UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SEGMENTACIÓN Y CLASIFICACIÓN DE CABEZA DE ESPERMATOZOIDES PARA DIFERENCIAR SU NORMALIDAD UTILIZANDO DEEP LEARNING

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA

PILAR ANDREA NILO VÁSQUEZ

PROFESOR GUÍA:
VÍCTOR CASTAÑEDA ZEMAN

MIEMBROS DE LA COMISIÓN:
CARLOS NAVARRO CLAVERÍA
ANDRÉS CABA RUTTE

SANTIAGO DE CHILE
2025

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA
POR: PILAR ANDREA NILO VÁSQUEZ
FECHA: 2025
PROF. GUÍA: VÍCTOR CASTAÑEDA ZEMAN

SEGMENTACIÓN Y CLASIFICACIÓN DE CABEZA DE ESPERMATOZOIDES PARA DIFERENCIAR SU NORMALIDAD UTILIZANDO DEEP LEARNING

La infertilidad afecta al 15 % de las parejas a nivel mundial, según la Organización Mundial de la Salud (OMS), y entre el 20 % y el 30 % de los casos se asocia con infertilidad masculina, principalmente por alteraciones en la morfología espermática, factor clave para la fertilización. El análisis morfológico tradicional es manual, lento y subjetivo, mientras que los sistemas de Análisis Seminal Asistidos por Computadora (ASAC) son costosos y de acceso limitado. La sexta edición del manual de la OMS destaca la necesidad de evaluaciones estandarizadas. Bajo este contexto, las tecnologías de *deep learning* ofrecen una solución para transformar este análisis en un proceso objetivo, preciso y accesible.

Este trabajo de título desarrolla algoritmos de *deep learning* para segmentar y clasificar cabezas de espermatozoides en imágenes de microscopia óptica teñidas con hematoxilina-eosina para diferenciar su normalidad.

Se implementó un Modelo U-Net para segmentación y cuatro modelos de clasificación los cuales fueron ShuffleNetV2, CNN, ResNet-18 y una red basada en espacios latentes.

Para entrenar, validar y evaluar estos modelos se usó la base de datos SCIAN, dividiendo los datos en la proporción 70 % (entrenamiento), 15 % (validación) y 15 % (evaluación) para el modelo de segmentación, y utilizando validación cruzada Leave-One-Out para el modelo de clasificación.

Los resultados indican que el modelo U-Net logró un coeficiente DICE de 0,95, Intersection over Union de 0,91 y distancia Hausdorff de 1,91 píxeles en evaluación, mientras que en el modelo de clasificación, ShuffleNetV2 alcanzó un *accuracy* del 77 %.

En conclusión, se logró implementar modelos de *deep learning* para segmentar y clasificar espermatozoides para diferenciar su normalidad, donde el modelo U-Net mostró un rendimiento competitivo permitiendo su uso a nivel clínico. Por otro lado el modelo de clasificación requiere de mejoras para alcanzar estándares clínicos. Este trabajo valida el potencial de los algoritmos de *deep learning* para automatizar el diagnóstico de infertilidad masculina.

A mis padres y hermano.

Agradecimientos

Agradezco a mi familia por su sacrificio, apoyo y todas las herramientas que me entregaron durante este proceso y cada una de las etapas de mi vida.

Agradezco a todos mis amigos, que sin ellos este paso universitario no hubiera sido el mismo. A Sebastián Lemus por su apoyo y años de amistad desde el colegio. A Bastían, Luciano, Pedre, Rocío, Clau, Donko y Brian por ser un apoyo en esta etapa. Al grupo de eléctrica, Gonzalo, Kelly, Agustín, Diego, Pancha y Vale por su compañía y apoyo durante la especialidad y la pandemia. A Electrotutores, por ayudarme a crecer y confiar en mi durante esta etapa universitaria.

También agradecer a mi mascota Lukas, a quien extraño cada día.

Tabla de Contenido

1. Introducción **1**
1.1. Motivación y contexto . . . 1
1.2. Objetivos . . . 2
1.2.1. Objetivo general . . . 2
1.2.2. Objetivos específicos . . . 2

2. Marco teórico **3**
2.1. Conceptos Biológicos . . . 3
2.1.1. Estructura del espermatozoide . . . 3
2.1.2. Espermiograma . . . 3
2.1.3. Defectos en la estructura espermática . . . 4
2.1.4. Análisis morfológico espermático . . . 5
2.1.5. Sistema ASAC . . . 5
2.2. Procesamiento de imágenes . . . 6
2.2.1. *Deep Learning* . . . 6
2.2.2. Funciones de Pérdida . . . 7
2.2.2.1. Dice Loss . . . 7
2.2.2.2. Focal Loss . . . 7
2.2.2.3. Binary Cross Entropy With Logits Loss . . . 7
2.2.2.4. Cross Entropy Loss . . . 8
2.2.3. Validación Cruzada . . . 8
2.2.3.1. Leave-One-Out (LOO) . . . 8
2.2.4. Data leakage . . . 9
2.2.5. Machine Learning . . . 9
2.2.5.1. Random Forest . . . 9
2.2.5.2. XGBoost . . . 9
2.2.6. Segmentación imágenes . . . 9
2.2.6.1. U-net . . . 10
2.2.6.2. Mask Region Convolutional Neural Network . . . 11
2.2.6.3. Detectron2 . . . 11
2.2.7. Clasificación de imágenes . . . 11
2.2.7.1. Redes Convolutional Neural Network . . . 12
2.2.7.2. Familia Visual Geometry Group . . . 13
2.2.7.3. Arquitecturas Net clásicas . . . 13
2.2.7.4. Redes densas y de eficiencia . . . 15
2.2.7.5. Espacios latentes . . . 17
2.2.7.6. Bloque Squeeze-and-Excitation (SE) . . . 17

2.2.8. Características morfológicas . . . 17
2.3. Métricas de evaluación . . . 18
2.3.1. Clasificación de imágenes . . . 18
2.3.1.1. Matriz de confusión . . . 18
2.3.1.2. *Accuracy* . . . 19
2.3.1.3. *Recall* . . . 19
2.3.1.4. *Precision* . . . 20
2.3.1.5. Area Under Curve (AUC) . . . 20
2.3.1.6. F1-Score . . . 20
2.3.2. Segmentación . . . 21
2.3.2.1. Coeficiente de Sorense-DICE . . . 21
2.3.2.2. Distancia de Hausdorff . . . 21
2.3.2.3. Intersection Over Union . . . 21
2.3.3. AveragePrecision . . . 22
2.4. Base de datos . . . 22
2.4.1. SCIAN-MorphoSpermGS . . . 22
2.4.2. HuSHeM . . . 23
2.4.3. SMIDS . . . 23

3. Estado del arte **25**
3.1. Procesamiento de imágenes . . . 25
3.1.1. Clasificación . . . 25
3.1.2. Segmentación . . . 28

4. Desarrollo **30**
4.1. Metodología . . . 30
4.1.1. Justificación metodológica . . . 30
4.1.2. Entornos de desarrollo . . . 31
4.2. Algoritmo de segmentación . . . 31
4.2.1. Preprocesamiento de datos . . . 32
4.2.2. Modelo U-net . . . 32
4.2.3. Mask R-CNN . . . 33
4.3. Algoritmos de clasificación . . . 34
4.3.1. Preprocesamiento de datos . . . 34
4.3.1.1. Extracción de características . . . 35
4.3.2. Convolutional Neural Network (CNN) . . . 36
4.3.3. ShuffleNetV2 . . . 37
4.3.4. Resnet-18 . . . 38
4.3.5. Red espacios latentes . . . 39
4.3.6. Random Forest . . . 40
4.3.7. XGBoost . . . 41

5. Resultados y análisis **42**
5.1. Segmentación . . . 42
5.1.1. Discusión general . . . 45
5.2. Clasificación . . . 46
5.2.1. Convolutional Neuronal Network (CNN) . . . 46
5.2.2. ShuffleNetV2 . . . 47

5.2.3. ResNet-18 . . . 48
5.2.4. Espacios Latentes . . . 49
5.2.5. Comparación . . . 49
5.2.6. Discusión general . . . 50

6. Conclusiones **53**
6.1. Trabajo futuro . . . 54
6.1.1. Segmentación . . . 54
6.1.2. Clasificación . . . 54

Bibliografía **55**

Anexos **59**
A. Conjuntos de datos . . . 59
B. Segmentación . . . 60
C. CNN . . . 62
D. ShuffleNetV2 . . . 64
E. ResNet-18 . . . 66
F. Espacios Latentes . . . 67

Índice de Tablas

3.1. Resumen de métricas de evaluación de modelos de *deep learning* para la clasificación de espermatozoides descritos en el estado del arte. Todos los valores expresados en %. 28
3.2. Métricas de los métodos de segmentación descritos en el estado del arte. 29
4.1. Características de forma. 35
4.2. Descriptores de forma. 36
4.3. Descriptores de textura. 36
4.4. Grilla hiperparámetros RandomForest. 41
4.5. Grilla de hiperparámetros algoritmo XGBoost. 41
5.1. Métricas de evaluación en conjunto de validación y evaluación del modelo U-Net. 44
5.2. Métricas de evaluación del modelo U-Net diseñado y el estado del arte. 45
5.3. Tabla de métricas de evaluación para el modelo CNN utilizando tres funciones de pérdida. 47
5.4. Tabla de métricas de evaluación del modelo ShuffleNetV2 usando tres funciones de pérdida. 48
5.5. Tabla de métricas de evaluación de los mejores modelos de *deep learning* diseñados y los algoritmos de *machine learning*. 49
5.6. Tabla de métricas de evaluación del mejor modelo de clasificación diseñado y el estado del arte. 50
5.7. Parámetros utilizados en cada algoritmo diseñado. 50
A.1. División de conjuntos para el algoritmo de segmetnación. 59
A.2. Cantidad de clases 0, 1 y 2 por paciente. 59
A.3. Cantidad de clases 0 y 1 por paciente, después de limpiar el *dataframe*. 60
B.1. Métricas de evaluación modelo U-Net utilizando el 60 % de la base de datos. . 60
B.2. Métricas de evaluación modelo Mask R-CNN 62
C.1. Métricas de evalución para iteraciones sobre hiperparametros en la función de pérdida Focal Loss en el modelo CNN. 63
D.1. Métricas de evaluación para iteraciones sobre hiperparametros en la función de pérdida Focal Loss en el modelo ShuffleNetV2 65
E.1. Tabla resumen de las métricas de evaluación del modelo ResNet-18 usnado tres diferentes funciones de pérdida. 67
F.1. Iteración sobre hiperparámetros para la función de pérdida Focal Loss en el modelo de espacios latentes. 68

Índice de Ilustraciones

2.1. Dibujos esquemáticos de algunas formas anormales de espermatozoides humanos. Imagen extraída de [3] 5
2.2. Arquitectura U-net (ejemplo para 32×32 píxeles), según [14] 10
2.3. Esquema MaskR-CNN para la segmentación por instancias, según [15] 11
2.4. Arquitectura común de una CNN según [17] 13
2.5. Arquitectura de Visual Geometry Group (VGG). 13
2.6. Arquitectura de GoogleNet. 14
2.7. Arquitectura de ResNet50. 15
2.8. Bloques de construcción de ShuffleNetV1 y V2. (a)arquitectura básica ShuffleNetV1; (b)ShuffleNetV1 para un muestreo espacial descendente; (c)arquitectura básica ShuffleNetV2; (d)ShuffleNetV2 para un muestreo espacial descendente. Según [21]. 15
2.9. Una DenseNet profunda con tres bloques densos. Según [22] 16
2.10. Arquitecturas de MobileNet. 17
2.11. Estructura general Block SE según [25]. 17
2.12. Matriz de confusión, elaboración propia. 19
2.13. Curva ROC, según [5] 20
2.14. Intersection Over Union (IoU). a) Cálculo de IoU mediante la división entre la intersección de dos conjuntos entre la unión de estos. b) tres ejemplos de cálculos de IoU, Según [5] 22
2.15. Enfoque de adquisición de datos utilizado para obtener imágenes de SMIDS según [34] 23
2.16. Ejemplos de imágenes en cada base de dato internacional. (a) SCIAN-MorphoSpermGS (b) HuSHeM y (c) SMIDS 24
4.1. Arquitectura U-Net modificada. 33
4.2. Arquitectura de CNN 36
4.3. Arquitectura ShuffleNetV2 38
4.4. Arquitectura de ResNet-18. 39
4.5. Arquitectura de modelo espacios latentes. 40
5.1. Gráfico de pérdidas en entrenamiento y validación - modelo U-Net. 42
5.2. Ejemplo de resultados de segmentación utilizando arquitectura U-Net diseñada, siguen el siguiente orden: Imagen original, máscara real, predicción, errores. En color verde se tienen los aciertos entre la predicción y la máscara real, en azul lo que segmentó adicional el modelo y en rojo lo que faltó por segmentar. . . . 43
5.3. Gráfico comparativo de modelo U-Net diseñado con el estado del arte. 45
5.4. Comparación de métricas de evaluación del modelo CNN usando tres diferentes funciones de pérdida. 47

5.5. Comparación de métricas de evaluación del modelo ShuffleNetV2 usando tres diferentes funciones de pérdida. 48
5.6. Gráfico de métricas de evaluación de los mejores modelos de *deep learning* diseñados y los algoritmos de *machine learning*. 49
5.7. Gráfico de comparación de métricas de evaluación entre el mejor modelo de clasificación diseñado y el estado del arte. 50
B.1. Gráfico de pérdidas en entrenamiento y validación modelo U-Net entrenada con el 60 % de la base de datos. 60
B.2. Gráficos de pérdida en entrenamiento de sus componentes. 61
C.1. Gráfico de pérdidas de entrenamiento y validación del modelo CNN utilizando función de pérdida CrossEntropyLoss y BCEWithLogitsLoss. 62
C.2. Gráfico de pérdidas en entrenamiento y validación del mejor rendimiento utilizando Focal Loss en el modelo CNN. 62
C.3. Gráfico de comparación de métricas de evaluación entre iteraciones sobre hiperparametros para la funcion de perdida Focal Loss en el modelo CNN. 63
D.1. Gráfico de pérdidas de entrenamiento y validación del modelo ShuffleNetV2 utilizando función de pérdida CrossEntropyLoss y BCEWithLogitsLoss. 64
D.2. Gráfico de pérdidas en entrenamiento y validación en el mejor rendimiento utilizando Focal Loss en el modelo ShuffleNetV2. 64
D.3. Gráfico de comparación de métricas de evaluación entre iteraciones sobre hiperparametros para la funcion de perdida Focal Loss en el modelo ShuffleNetV2 65
E.1. Gráficos de pérdida en entrenamiento y validación del modelo ResNet-18 para tres funciones de pérdida diferentes. 66
F.1. Gráfico de pérdidas en entrenamiento y validación del mejor modelo de espacios latentes. 67
F.2. Gráfico de comparación de métricas de evaluación entre iteraciones sobre hiperparámetros para la función de pérdida Focal Loss en el modelo espacios latentes. 68

Capítulo 1

Introducción

1.1. Motivación y contexto

La infertilidad afecta al 15 % de las parejas a nivel mundial, según la Organización Mundial de la Salud (OMS). Entre el 20 % y el 30 % de estos casos están relacionados con la infertilidad masculina, frecuentemente asociada a alteraciones en la calidad del semen [1].

Un factor clave en la calidad espermática es la morfología de los espermatozoides, ya que esta, representada en su forma, influye directamente en su capacidad de fertilización. El análisis morfológico permite detectar defectos en la cabeza, cuello o cola, los cuales comprometen la funcionalidad y dificultan el éxito de la fecundación.

El análisis espermático, esencial para evaluar la fertilidad masculina, incluye parámetros como motilidad, concentración y morfología. Estos análisis tradicionalmente se realizan de forma manual bajo el microscopio, lo que genera procesos lentos y con resultados variables debido a la subjetividad de los expertos. Aunque los sistemas de Análisis Seminal Asistido por Computadora (ASAC) han mejorado la precisión y reproducibilidad, su uso está limitado por su costo elevado y, por ende, por su acceso restringido [2].

La sexta edición del manual de la OMS subraya la importancia de estandarizar las evaluaciones morfológicas, destacando que una clasificación precisa y reproducible es esencial para el diagnóstico [2].

En este contexto, las tecnologías basadas en *deep learning* tienen el potencial de transformar el análisis morfológico de los espermatozoides de uno visual y subjetivo a uno objetivo y preciso. Estas herramientas pueden segmentar y clasificar cabezas de espermatozoides con mayor precisión, consistencia y velocidad, reduciendo costos y aumentando su reproducibilidad.

Su implementación podría no solo optimizar los tratamientos de fertilidad, sino también facilitar el acceso a estas evaluaciones, mejorando significativamente la atención en este ámbito.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar algoritmos de *deep learning* que permitan segmentar y clasificar cabezas de espermatozoides en imágenes de microscopía óptica de campo claro teñidos con hematoxilina-eosina, considerando una baja resolución espacial, para diferenciar su normalidad.

1.2.2. Objetivos específicos

1. Desarrollar un algoritmo de segmentación de imágenes basado en *deep learning* para segmentar cabezas de espermatozoides .

2. Desarrollar un algoritmo de clasificación para evaluar la morfología de la cabeza de espermatozoides.

3. Validar los algoritmos de segmentación y clasificación en base de datos internacionales para medir su precisión.

4. Evaluar el rendimiento de los modelos con los algoritmos del estado del arte para comparar sus resultados.

Capítulo 2

Marco teórico

2.1. Conceptos Biológicos

2.1.1. Estructura del espermatozoide

El espermatozoide humano es una célula reproductora masculina que transporta material genético. Tiene una forma alargada, similar a una elipse, y su estructura está formada por una cabeza, un cuello (también llamada pieza media) y una cola.

De acuerdo con la OMS [3], la cabeza del espermatozoide es ovalada, lisa y de contorno regular, con una longitud de $4{,}0-5{,}0\ \mu m$ y ancho de $2{,}5-3{,}5\ \mu m$. Se compone principalmente de un acrosoma, el cual contiene enzimas que permiten al espermatozoide penetrar el óvulo, ocupando $\frac{2}{3}$ del volumen total de la cabeza. Además, no debe contener vacuolas grandes ni más de dos vacuolas pequeñas. También dentro de esta estructura se encuentra el núcleo, el cual contiene los 23 cromosomas.

El cuello, también llamado pieza media, almacena mitocondrias que permiten el movimiento del espermatozoide, ya que esta parte entrega la energía necesaria para el movimiento de la cola. Está unido de forma axial a la cabeza, es delgado (aproximadamente 1 μm) y alcanza una longitud equivalente a una cabeza y media.

La cola tiene la función de permitir la movilidad espermática. Su longitud es de 45 μm, y debe ser recta, uniforme y estar desenrollada para permitir el nado a una velocidad de aproximadamente 3 mm por minuto. Esta condición indica que el espermatozoide es progresivo, es decir, que avanza en línea recta para poder llegar al óvulo.

Para considerar un espermatozoide como normal, las gotas citoplasmáticas deben ser menores que la mitad del tamaño de una cabeza normal.

2.1.2. Espermiograma

El espermiograma es un examen utilizado para evaluar la fertilidad masculina mediante el análisis de variables macroscópicas y microscópicas del semen humano. Estas últimas son especialmente relevantes, ya que permiten determinar las características morfológicas y funcionales de los espermatozoides, siendo un factor predictor de su capacidad de fertilización.

Según la OMS [3], las principales variables microscópicas evaluadas en un espermiograma incluyen la concentración, motilidad, vitalidad y morfología de los espermatozoides.

La primera de ellas se refiere al número de espermatozoides presentes en un mililitro (ml) de semen. La OMS considera normal tener al menos 15 millones de espermatozoides/ml en el espermiograma o una concentración total de al menos 39 millones de espermatozoides en el volumen total del eyaculado, ya que niveles inferiores son indicativos de oligozoospermia, condición que afecta la fertilidad.

La motilidad se refiere a la capacidad de los espermios para moverse, estableciéndose que al menos el 40 % de los espermatozoides deben ser móviles. Una motilidad deficiente puede indicar astonozoospermia, lo cual limita la capacidad del espermio para alcanzar el óvulo.

Por otro lado, la vitalidad mide el porcentaje de espermatozoides vivos en una muestra de semen, considerándose normal un mínimo del 58 % de espermatozoides vivos. Un porcentaje inferior a este valor de referencia indicaría necrospermia, lo que influye negativamente en la fertilidad.

Finalmente, la morfología evalúa la forma y estructura de los espermatozoides, estableciendo que al menos el 4 % de estos deben tener una morfología normal, conforme a los criterios ya mencionados en la sección anterior.

2.1.3. Defectos en la estructura espermática

Los espermatozoides pueden sufrir alteraciones o malformaciones, lo cual lleva a que tengan un menor potencial de fecundación. Es por esto que un mínimo defecto en los criterios descritos anteriormente sobre la estructura del espermatozoide humano permite clasificarlos como un espermatozoide anormal.

La OMS [3] clasifica las anomalías estructurales que se pueden presentar en un espermatozoide en diferentes categorías, correspondientes a una parte específica de la estructura morfológica.

En la cabeza se pueden encontrar defectos de tamaño y forma, tales como cabezas grandes, pequeñas, acintadas y/o piriformes. También se encuentran las cabezas amorfas y vacuoladas, donde estas últimas presentan más del 20 % de su área ocupada por vacuolas no teñidas. Asimismo, es posible encontrar anomalías en el área acrosomal, donde esta es menor al 40 % del área total de la cabeza.

En lo que respecta al cuello, las anomalías incluyen cuello doblado, que forma un ángulo mayor a 90° respecto al eje de la cabeza, además de una inserción asimétrica. También se encuentran cuellos de grosor irregular, ya sea más grueso o anormalmente fino.

Los defectos de cola abarcan malformaciones como cola corta, múltiple, en forma de horquilla, rota o doblada en ángulo superior a 90°. También se observan casos con colas de grosor irregular o enrolladas.

Por último, las gotas citoplasmáticas constituyen una anomalía significativa cuando ocupan más de la mitad de la superficie de la cabeza de un espermatozoide normal.

En la Figura 2.1 se observan los defectos espermáticos mencionados anteriormente.

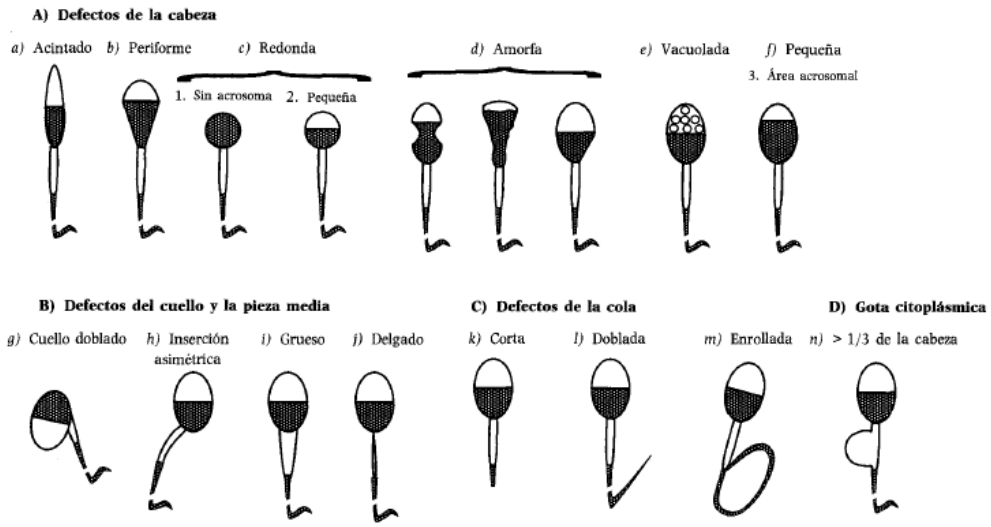


Figura 2.1: Dibujos esquemáticos de algunas formas anormales de espermatozoides humanos. Imagen extraída de [3]

2.1.4. Análisis morfológico espermático

El análisis morfológico de espermatozoides es fundamental para el diagnóstico de la infertilidad masculina, ya que permite evaluar la calidad de los espermios y detectar anomalías que puedan afectar su funcionalidad y, en consecuencia, la fertilidad.

Los expertos utilizan diversos métodos de tinción y técnicas de imagen para llevar a cabo el análisis morfológico. Entre estos, la técnica de tinción Hematoxilina-Eosina proporciona un buen contraste para la evaluación morfológica de la cabeza de los espermatozoides, superando métodos de tinción como Papanicolaou, Diff-Quick y de Wright. En cuanto a la obtención de imágenes, la microscopía de campo claro es el método más común, ya que proporciona los detalles suficientes para una evaluación morfológica precisa [2].

Dado que existe un alto grado de variabilidad entre los observadores expertos que clasifican la morfología de los espermatozoides, se ha generado la necesidad de estandarizar el método de clasificación en los laboratorios. Por esta razón, muchos laboratorios han adoptado el análisis de espermatozoides asistido por computadora, conocido como ASAC o CASA por sus siglas en inglés, con el fin de reducir la variabilidad entre observadores expertos.

2.1.5. Sistema ASAC

El análisis de espermatozoides asistido por computadora (ASAC) o *Computer Assisted Sperm Analysis* (CASA) es una técnica avanzada que utiliza software especializado para analizar automáticamente la morfología y motilidad de los espermatozoides. Este método proporciona un análisis cuantitativo detallado, reduciendo la subjetividad a la que está ex-

puesta la evaluación manual.

Este sistema emplea cámaras que capturan múltiples imágenes de la muestra en tiempo real, permitiendo que el software analice tanto la motilidad como la morfología de los espermatozoides. Además, mide parámetros como la velocidad de desplazamiento, la forma de cabeza, la pieza media y cola, alcanzando una precisión del 92 % en el conteo y clasificación de los parámetros morfológicos y cinéticos [2].

2.2. Procesamiento de imágenes

El procesamiento de imágenes hace referencia a la manipulación y análisis de imágenes con el propósito de extraer información útil o mejorar su calidad para tareas específicas.

Entre las tareas que se realizan en el procesamiento de imágenes se encuentran la clasificación y segmentación, las cuales pueden llevarse a cabo mediante técnicas de *deep learning* utilizando redes Convolutional Neural Network (CNN)

2.2.1. *Deep Learning*

El *deep learning* o aprendizaje profundo, según el artículo [4], es una técnica de aprendizaje automático basada en redes neuronales artificiales profundas. Esta técnica permite procesar grandes cantidades de datos, aprendiendo de forma autónoma a partir de estos, mejorando su rendimiento con el tiempo y realizando tareas complejas como el reconocimiento de voz, reconocimiento visual de objetos y la detección de objetos, entre otros.
Una red de *deep learning* se compone de múltiples capas de neuronas interconectadas, que dependen de la información procesada en la capa anterior para optimizar la predicción a realizar. La primera capa, llamada capa de entrada, recibe los datos en bruto y no estructurados; las capas de agrupamiento que reducen la dimensionalidad, mejorando eficiencia y robustez; las capas intermedias, conocidas como capas ocultas que incluyen una o más capas de convolución, realizan la extracción de características y patrones; finalmente, en la última capa de salida, se producen los resultados finales.

El entrenamiento de los modelos de *deep learning* se realizan mediante el algoritmo de retropropagación (*backpropagation*) que ajusta los parámetros de la red para minimizar el error de las predicciones. Esto permite que los modelos aprendan a realizar tareas complejas, logrando mejores resultados que los enfoques tradicionales con una o dos capas de neuronas.

Entre los tipos de redes neuronales artificiales en las cuales se basa el *deep learning* se encuentran las siguientes:

- Redes Neuronales Convolucionales (CNN): procesan y clasifican imágenes a través de los objetos que aparecen en ellas.
- Redes Neuronales Recurrentes (RNN): utilizan datos secuenciales para su procesamiento, como el lenguaje natural. Por ejemplo Google Translate.
- Redes Neuronales Generativos (GAN): emplean dos redes neuronales que compiten entre sí para generar contenido nuevo que simula ser real, como imágenes o textos.

- Redes Neuronales Profundas (DNN); realizan tareas complejas sin necesidad de ser programadas explícitamente. Por ejemplo el recorte automático de imágenes
- Redes Neuronales Modulares(MNN): están construidas a partir de módulos predefinidos, seleccionados de acuerdo con la tarea específica a realizar.

2.2.2. Funciones de Pérdida

Las funciones de pérdida son funciones matemáticas que permiten evaluar o medir qué tan bien está modelando el conjunto de datos de ejemplo entregado al modelo, cuantificando el error entre las predicciones y los valores reales, donde un valor bajo sugiere un mejor ajuste del modelo a los datos durante el entrenamiento y un valor alto indica necesidad de ajustar parámetros. Durante el entrenamiento, la función de pérdida busca minimizar el error ajustando los pesos a través de técnicas como el descenso de gradiente [5].
Estas se clasifican según el tipo de tarea de aprendizaje, en este trabajo en particular solo se trabajará con tareas de clasificación y segmentación.

2.2.2.1. Dice Loss

Dice Loss también conocida como Dice Similarity Coefficiente (DSC), es una función comúnmente usada en tareas de segmentación. Evalúa la similitud entre la predicción del modelo y la máscara verdadera.
Mide la superposición entre la predicción y la etiqueta verdadera mediante le Ecuación 2.2.2.1, donde *pred* son los píxeles predichos y *gt* los píxeles verdaderos. Un valor cercano a 0 indica una alta similitud, mientras que valores más altos reflejan un menor rendimiento en la segmentación [5].

$$\text{DiceLoss} = 1 - \frac{2 \cdot \cap(pred, gt)}{|pred| + |gt|} \tag{2.1}$$

2.2.2.2. Focal Loss

La función de pérdida Focal Loss para la tarea de clasificación es una variación de la función Cross Entropy Loss que aborda el problema de desbalance de clases, enfocándose en ejemplos difíciles y reduciendo el impacto de los ejemplos fáciles para mejorar la capacidad del modelo en aprender patrones de la clase minoritaria [5].
Su fórmula es la de la Ecuación 2.2, donde p_t es la probabilidad predicha para la clase correcta, α_t ajusta el peso de las clases desbalanceadas entre 0,1 a 1,0, y γ que es un hiperparámetro entre 2,0 a 4,0, modula el enfoque en ejemplo difíciles. La función reduce la contribución de ejemplos bien clasificados, es decir cuando la probabilidad p_t es cercana a 1, al elevar $(1 - p_t$) a la potencia γ, priorizando la optimización de ejemplos mal clasificados [5].

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{2.2}$$

2.2.2.3. Binary Cross Entropy With Logits Loss

La función de pérdida Binary Cross Entropy With Logits Loss (BCEWithLogitsLoss), implementada por PyTorch [6], es una combinación entre Binary Cross Entropy (BCE) [5] y

una activación Sigmoide [7]. Calcula la pérdida entre las predicciones (logits) de un modelo y las clases verdaderas, en este caso 0 o 1 ya que es clasificación binaria, aplicando la activación sigmoide a los logits y luego calcula la pérdida a través de la Ecuación 2.3.

$$L = \frac{1}{N}\sum_{i=1}^{N}[\log_e(1 + e^{-z}) + z(1 - y_i)] \tag{2.3}$$

La ecuación anterior es la resultante de reemplazar p_i de la Ecuación 2.5 de BCE del artículo [8], con la función sigmoide de la Ecuación 2.4.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{2.5}$$

Donde:

- N: número de ejemplos.
- y_i: clase verdadera del ejemplo i.
- p_i: probabilidad predicha del ejemplo i para la clase verdadera.

2.2.2.4. Cross Entropy Loss

La función de pérdida Cross Entropy Loss es utilizada en tareas de clasificación y mide la diferencia entre la distribución de los valores predichos y los verdaderos. Según el artículo [9], esta pérdida pertenece a una familia de funciones *composite losses*, las cuales aplican una transformación Softmax a las salidas del modelo y luego calculan la pérdida.

La pérdida se calcula mediante la Ecuación 2.6 donde t_i es la etiqueta verdadera y s_i la probabilidad predicha.

$$\text{CE} = -\sum_{i=1}^{C} t_i \log(s_i) \tag{2.6}$$

2.2.3. Validación Cruzada

Según James, G. et al. [10] la validación cruzada es una técnica para evaluar el rendimiento de un modelo estadístico, evitando que se sobreajuste. Esto dividiendo el conjunto de datos en subconjuntos, de los cuales algunos serán utilizados para entrenar y otros para evaluar, con el fin de asegurar que el modelo generalice ante la presencia de nuevos datos.
Existen varios tipos de validación cruzada y en este trabajo se utilizará la de tipo Leave-One-Out.

2.2.3.1. Leave-One-Out (LOO)

Este método implica dividir el conjunto de datos en dos parte, pero en vez de que estos sean de igual tamaño, se utiliza una única observación para el conjunto de prueba, y las restantes constituyen el conjunto de entrenamiento, luego el error de predicción se calcula para la observación dejada fuera. Se repite este proceso n veces, una por cada observación, y el error de LOO es el promedio de los errores individuales [10].

2.2.4. Data leakage

Según Bouke, M. A et al. [11] Data leakage también conocido como Pattern leakage, ocurre durante el pre-procesamiento de los datos cuando los datos del conjunto de evaluación son utilizados en el conjunto de entrenamiento, llevando a un sobreajuste y precisión elevada en el rendimiento del modelo.

2.2.5. Machine Learning

Machine learning es una técnica de la inteligencia artificial que a través de algoritmos y modelos estadísticos permiten realizar tareas como predicciones, clasificaciones o toma de decisiones sin recibir una instrucción explicita.

Hay dos tipos de técnicas: aprendizaje supervisado, que entrena un modelo con datos de entrada y salida conocidos para que de predicciones en un futuro, y aprendizaje no supervisado, que encuentra patrones ocultos en los datos de entrada, es utilizada para extraer inferencias de conjuntos de datos por ejemplo la agrupación de clústeres.

2.2.5.1. Random Forest

El algoritmo Random Forest (RF) propuesto por Breiman, L. [12] es un método de aprendizaje supervisado para tareas de clasificación y regresión, basado en múltiples árboles de decisión que utilizan el método *bagging*, es decir que cada árbol de decisión es entrenado con un subconjunto diferente de datos generados aleatoriamente. Una de las ventajas de RF es la capacidad de manejar un gran volumen de datos con alta dimensionalidad, proporcionando predicciones robustas.

La configuración del algoritmo depende de los hiperparámetros, como el número de árboles (n_estimators), que controla la cantidad de predictores, y el número de características consideradas por división (max_features), introduciendo aleatoriedad. La profundidad máxima (max_depth) y el número mínimo de muestras por hoja (min_samples_leaf), regulan la complejidad de los árboles individuales

2.2.5.2. XGBoost

El algoritmo XGBoost es un método de aprendizaje supervisado basado en árboles de decisión propuesto por Chen, T et al. [13], que utiliza la técnica *boosting*, que construye árboles de manera secuencial, donde cada árbol corrige los errores del anterior.
Sus hiperparámetros son tasa de aprendizaje (learning_rate), número de árboles (n_estimators), profundidad máxima (max_depth) y términos de regularización (λ y α), los cuales permiten ajustar el algoritmo.

2.2.6. Segmentación imágenes

La segmentación de imágenes se puede realizar mediante un modelo de aprendizaje profundo que divide la imagen en múltiples segmentos. Esto permite simplificar la detección de objetos y clasificar la información contenida en cada segmento.

Existen tres tipos de segmentación [5]:

- Segmentación Semántica: Consiste en dividir una imagen en diferentes regiones y asignar una etiqueta a cada píxel.

- Segmentación por instancias: Es capaz de detectar cada objeto presente en la imagen y asignarle una máscara con etiqueta única.
- Segmentación Panóptica: Es una combinación de la segmentación semántica y la segmentación por instancias.

Dentro de las arquitecturas capaces de realizar segmentación, encontramos la U-Net y la Mask Region Convolutional Neural Network (Mask R-CNN)

2.2.6.1. U-net

La U-Net es una arquitectura en forma de U, como se muestra en la Figura 2.2, diseñada para resolver problemas de segmentación semántica basada en redes convolucionales de codificación y decodificación.

La codificación consiste en 5 bloques que aplican repetidamente dos convoluciones de 3×3, seguidas de una unidad de activación lineal rectificada (ReLU) y un agrupamiento máximo (*max pooling*) para reducir la dimensión de la imagen y extraer sus características.

En cuanto a la decodificación, que realiza el proceso inverso de la codificación, también utiliza 5 bloques que permiten aumentar la resolución de la imagen. Esto se logra mediante un aumento en el tamaño del mapa de características usando un escalado (*upsampling*), seguido por una convolución de 2×2 que reduce el número de características. Luego, se aplican dos capas de convolución 3×3 con activación ReLU. Finalmente, en el último bloque, se realiza una convolución de 1×1 para convertir las características extraídas en clases, generando así una máscara de segmentación [14].

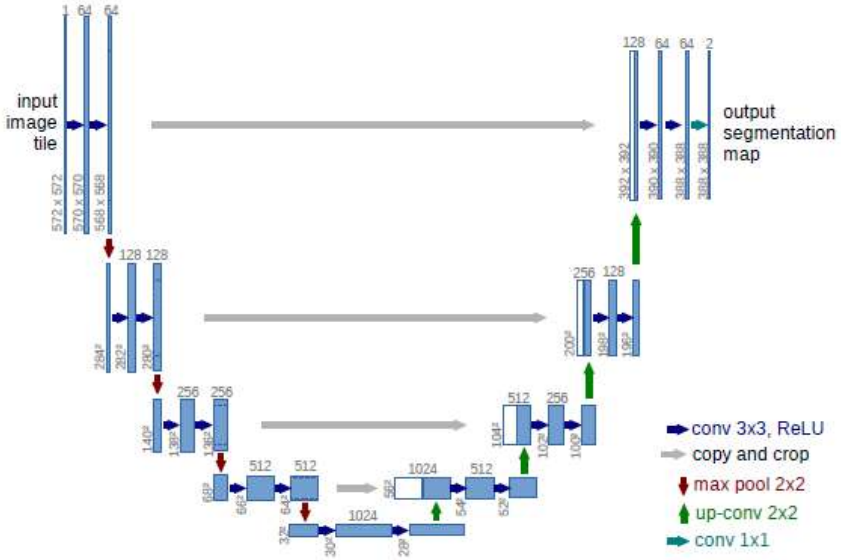


Figura 2.2: Arquitectura U-net (ejemplo para 32×32 píxeles), según [14]

2.2.6.2. Mask Region Convolutional Neural Network

Este modelo es una extensión de la arquitectura Faster R-CNN y realiza segmentación basada en redes convolucionales de región, es decir, una vez detectado el objeto, realiza la clasificación y segmentación de instancias en imágenes [15].
Se compone de:

- *Backbone*: Usa una red convolucional como ResNet para extraer características de la imagen.
- *Region Proposal Network* (RPN): Genera propuestas de regiones candidatas que podrían contener objetos.
- *Region of Interest Align* (RoIAlign): Alinea las regiones propuestas con precisión, corrigiendo problemas presentes en capas anteriores, con el fin de mejorar la precisión de la segmentación.
- Cabezas de predicción: Incluye ramas paralelas para clasificación de objetos, regresión de cajas delimitadoras y generación de máscaras de segmentación.

Su arquitectura se visualiza en la Figura 2.3.

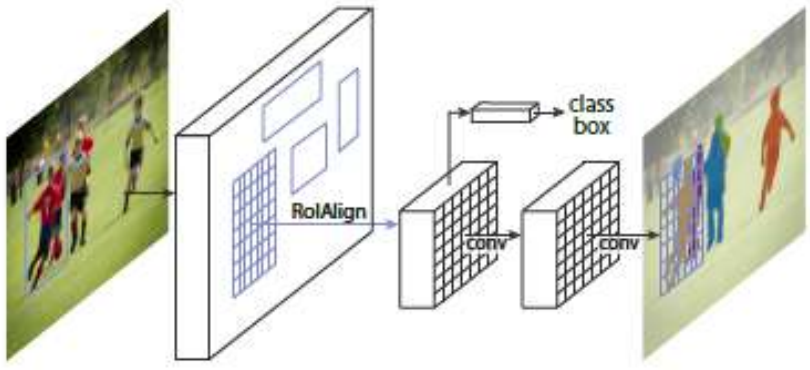


Figura 2.3: Esquema MaskR-CNN para la segmentación por instancias, según [15]

2.2.6.3. Detectron2

Detectron2 es una biblioteca de código abierto desarrollada por Meta AI (ex-Facebook AI Research), orientada a tareas de detección y segmentación de objetos. Está basada en Pytorch y proporciona implementaciones de modelos como Faster R-CNN, Mask R-CNN y RetinaNet, entre otros [16].

2.2.7. Clasificación de imágenes

La clasificación de imágenes consiste en entrenar un modelo de aprendizaje automático para reconocer y asignar una etiqueta o clase a una imagen, basada en su contenido.

Existen tres tipos de clasificación [5]:

- Clasificación Binaria: Se categoriza la imagen en una de dos tipos de clases. Por ejemplo: Verdadero y Falso, 1 y 0 o Positivo y Negativo.
- Clasificación Multiclase: Se categoriza la imagen en una de las múltiples clases. Por ejemplo: color de autos (rojo, verde, negro, etc.), tipo de animal (perro, gato, león, etc.) o tallas de ropa (pequeña, mediana, grande).
- Clasificación Multietiquetas: Se categoriza la imagen con más de una clase a la vez. Por ejemplo, una película puede ser clasificada como documental y otra como de ciencia ficción y acción.

Dentro de las arquitecturas diseñadas para la tarea de clasificación de imágenes se encuentran las siguientes:

2.2.7.1. Redes Convolutional Neural Network

Las redes Convolutional Neural Network (CNN) básicas son la base de muchas arquitecturas para detección de objetos, destacándose por la extracción de características de las imágenes gracias a las capas convolucionales.

La unión de tres tipos de capas, como la convolucional, de agrupación (*pooling*) y totalmente conectada (*fully connect*), conforma la arquitectura CNN mostrada en la Figura 2.4. La primera mencionada es el bloque principal de una CNN, que aplica filtros (*kernels*) sobre los datos de entrada para detectar características. Además, se utiliza una función de activación, como la unidad de rectificación lineal (ReLU), introduciendo no linealidad para mejorar la capacidad de representación de la CNN. Finalmente, se genera un mapa de características donde se almacena el resultado del producto escalar entre los datos de entrada y el filtro [17].

La capa de agrupación permite reducir la dimensión de los mapas de características, consiguiendo una disminución de la carga computacional. Esto se realiza con dos tipos de técnicas:

1. Agrupación máxima (*max pooling*): A medida que el filtro recorre los datos de entrada, se selecciona el mayor valor para enviarlo a la matriz de salida.
2. Agrupación media (*average pooling*): A medida que el filtro recorre los datos de entrada, calcula el promedio de los valores para enviarlo a la matriz de salida, lo que sirve para descartar detalles redundantes.

Finalmente, la capa totalmente conectada realiza la clasificación basándose en las características extraídas de las capas anteriores y sus diferentes filtros. A diferencia de las otras capas, esta utiliza una función de activación llamada Softmax que convierte un conjunto de valores en un vector de probabilidades entre 0 y 1, donde la suma de todos los valores es 1.

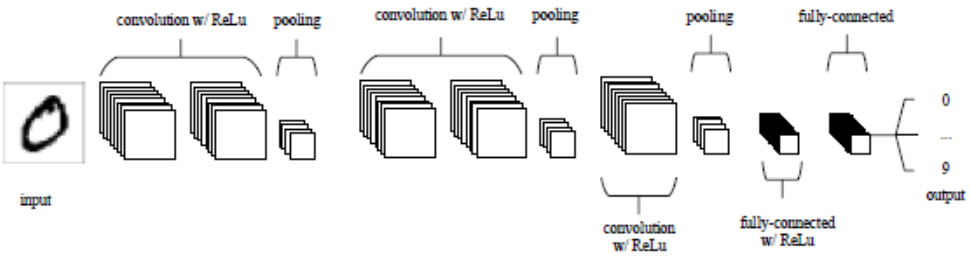


Figura 2.4: Arquitectura común de una CNN según [17]

2.2.7.2. Familia Visual Geometry Group

Las arquitecturas Visual Geometry Group (VGG) son redes neuronales profundas, que se centra en el uso de pequeñas convoluciones de 3×3 píxeles, alcanzando profundidades de 16 o 19 capas. Esto permite capturar características complejas y detalladas de las imágenes.

Su arquitectura se compone de capas convolucionales pequeñas que utilizan filtros de tamaño 3×3, como se mencionó anteriormente, además de utilizar la función de activación ReLU. Luego, siguen las capas de agrupación con filtros de tamaño 2×2 para reducir dimensiones. Finalmente, se agregan tres capas totalmente conectadas: las dos primeras tienen 4096 neuronas y la última tiene tantas neuronas como clases en el problema, añadiendo una capa final de Softmax para generar una salida del tipo vector de probabilidades [18].

En la Figura 2.5 se puede visualizar la arquitectura de la VGG.

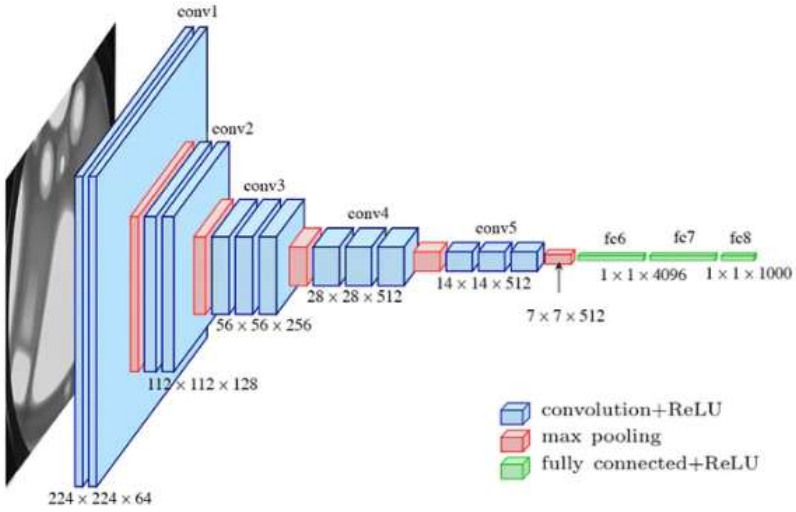


Figura 2.5: Arquitectura de Visual Geometry Group (VGG).

2.2.7.3. Arquitecturas Net clásicas

GoogleNet es una red convolucional profunda que introdujo el concepto de módulos Inception, permitiendo que la red elija entre diferentes tamaños de filtros en cada capa, mejorando su desempeño.

Su arquitectura se compone de 22 capas, donde la capa inicial es de convolución con un tamaño 7×7 y 64 filtros, seguida de una agrupación con *max pooling* de tamaño 3×3. Luego sigue el módulo Inception, donde cada módulo combina convoluciones de 1×1, 3×3, 5×5 y *max pooling*. La función de estos módulos es reducir dimensiones. Finalmente, las capas finales son Global Average Pooling (GAP) en lugar de una capa *fully connect* como en las redes VGG y CNN, que también reduce dimensiones, y una capa de salida con Softmax [19]. En la Figura 2.6 se muestra un resumen de las capas que componen la arquitectura GoogleNet.

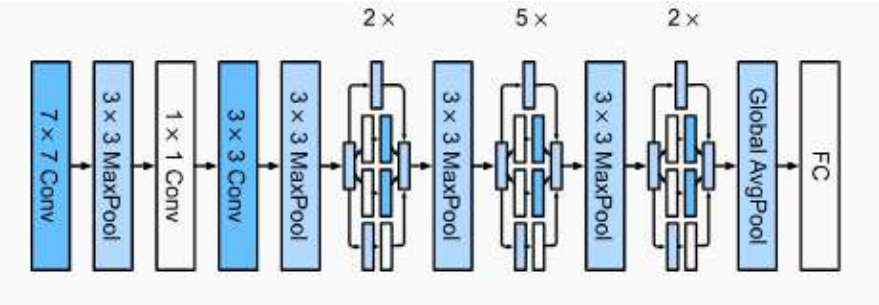


Figura 2.6: Arquitectura de GoogleNet.

ResNet permite la construcción de redes extremadamente profundas, desde 50 hasta 152 capas, gracias a las conexiones residuales, con las cuales también se aborda el problema de la degradación del rendimiento en redes muy profundas. Esto facilita la propagación del gradiente y mejora la eficiencia del aprendizaje.

Su arquitectura comienza con una capa inicial de convolución de tamaño 7×7 y 64 filtros, seguida de una capa de agrupamiento de *max pooling* de 3×3 para reducir dimensiones. Luego, si integran bloques residuales, de los cuales existen de dos tipos:

- **Bloques básico:** usados en versiones poco profundas como ResNet-18 y ResNet-34, consisten en dos capas convolucionales de tamaño 3×3.
- **Bloques con cuello de botella:** utilizados en versiones más profundas como ResNet-50, ResNet-101 y ResNet-152. Estos bloques comprimen y expanden las dimensiones usando tres capas: una de convolución 1×1, otra de 3×3 y finalmente 1×1.

Posteriormente, se agrega una capa de GAP que promedia todas las activaciones en cada mapa de características. Finalmente, se utiliza una capa de clasificación con Softmax [20].

A continuación, la Figura 2.7 se visualiza la arquitectura de una red ResNet-50 en específico.

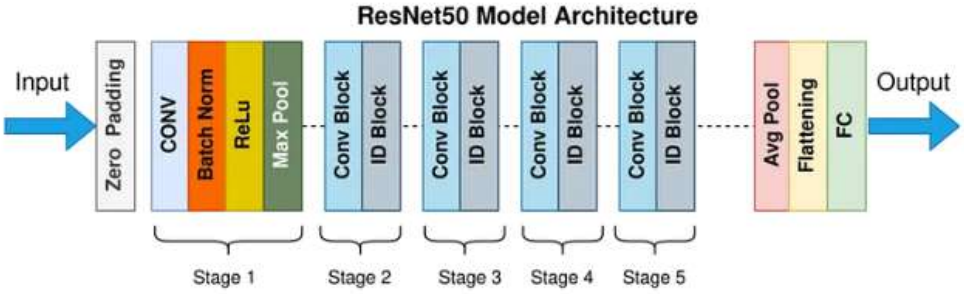


Figura 2.7: Arquitectura de ResNet50.

ShuffleNet V2 es altamente eficiente en velocidad de ejecución y consumo de memoria, lo que permite que sea utilizada en dispositivos con recursos limitados, como *smartphones*.

Su arquitectura, visualizada en la Figura 2.8, se compone de una capa de convolución de 3×3 y 24 filtros, seguida de una capa de agrupamiento *max pooling* reduciendo las dimensiones. Luego, se añaden bloques ShuffleNet, donde cada bloque se compone de una división de canales en dos partes, convoluciones separables (es decir, primero a una parte de los canales), luego convoluciones 1×1 y finalmente una mezcla de canales. Posterior al bloque, se añade una capa GAP para reducir la salida a un vector de características, y este pasa por una última capa de clasificación con Softmax [21].

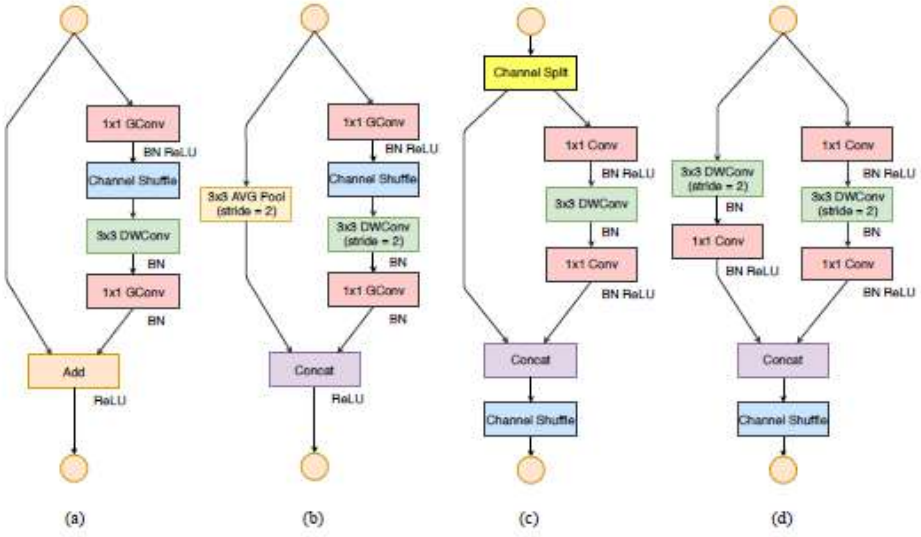


Figura 2.8: Bloques de construcción de ShuffleNetV1 y V2. (a)arquitectura básica ShuffleNetV1; (b)ShuffleNetV1 para un muestreo espacial descendente; (c)arquitectura básica ShuffleNetV2; (d)ShuffleNetV2 para un muestreo espacial descendente. Según [21].

2.2.7.4. Redes densas y de eficiencia

DenseNet es una red neuronal profunda de conexiones densas entre sus capas, mejorando el flujo de información y de gradientes a través de la red. Esta red es densa, ya que las

características de todas las capas previas se concatenan y sirven de entrada a la siguiente capa.

La arquitectura, mostrada en la Figura 2.9, comienza con una capa de convolución de 7×7, seguida de una capa de agrupación *max pooling* de 3×3. Luego, se añade un bloque denso donde sus capas constan de una convolución 1×1 que reduce dimensiones, y otra convolución de 3×3 que captura características especiales. Se concatenan todas las capas anteriores y se pasan a la siguiente capa, la cual es una de transición. En esta última, se reduce la dimensión con una convolución de 1×1 y agrupación *average pooling* de 2×2. Finalmente, se utiliza una capa de GAP para reducir la salida a un vector compacto, seguido de una capa de clasificación con Softmax [22].

Cabe mencionar que la DenseNet se puede configurar con diferentes profundidades, como por ejemplo: DenseNet-121, DenseNet-169, DenseNet-201 y DenseNet-264, donde el número corresponde a la cantidad de capas.

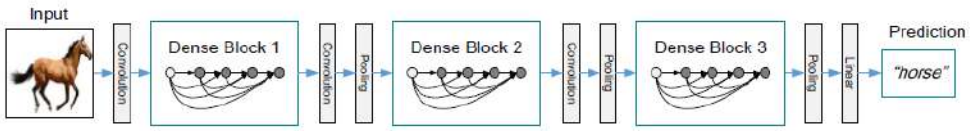


Figura 2.9: Una DenseNet profunda con tres bloques densos. Según [22]

MobileNet utiliza convoluciones de profundidad separable que dividen las operaciones en convoluciones por canal y puntuales, reduciendo así la carga computacional.

La arquitectura, visualizada en la Figura 2.10, se compone de 28 capas, comenzando con una capa convolucional de 3×3 y 32 filtros, seguida de una normalización por lotes (*bath normalization*) y función de activación ReLU. Luego, siguen bloques de convoluciones separable en profundidad. Estos consisten en dos etapas: la primera aplica una convolución en profundidad, donde un filtro opera en cada canal de la entrada procesando las características de forma separada. La segunda etapa consiste en una convolución puntual de 1×1, que combina los canales generados en la etapa 1 para producir la salida final. Estas etapas se intercalan a lo largo del bloque y utilizan normalización por lotes y una función de activación ReLU también. Al final de la red, se utiliza una capa de GAP, seguida de una capa de clasificación con Softmax [23].

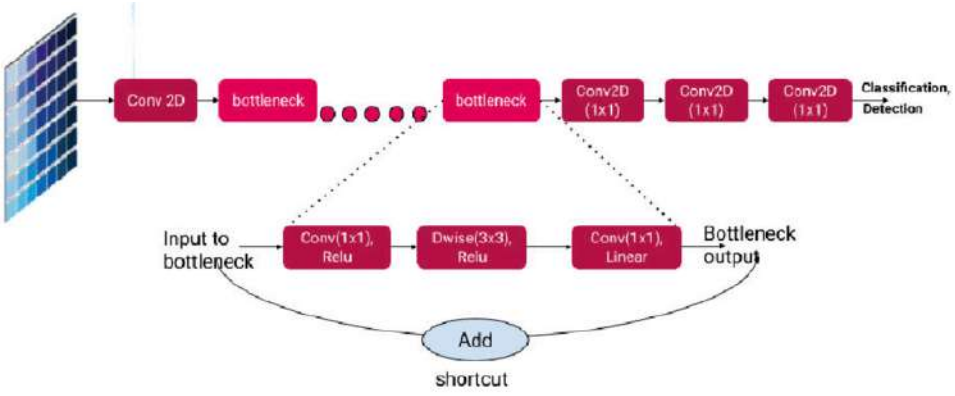


Figura 2.10: Arquitecturas de MobileNet.

2.2.7.5. Espacios latentes

El espacio latente es una representación comprimida y abstracta de datos, en la que las características relevantes se capturan en un espacio de menor dimensión [24]. Esto ocurre porque, tras el entrenamiento de una red neuronal, las capas de convolución transforman los datos de entrada en representaciones compactas que capturan patrones esenciales, descartando información redundante o irrelevante. En este trabajo en particular, se capturan características desde la capa anterior a la capa clasificadora, que servirán de base para otra red de *deep learning*.

2.2.7.6. Bloque Squeeze-and-Excitation (SE)

El bloque Squeeze-and-Excitation (SE), propuesto por Hu, J. et al. [25], es un mecanismo de atención que actúa sobre los canales de una red convolucional, permitiendo una recalibración dinámica de los mapas de características. Se conforma por tres etapas:

- *Squeeze*: Reduce la información espacial usando *global average pooling*.
- *Excitation*: Aprende qué canales son más importantes mediante una red neuronal ligera.
- *Scale*: Aplica una ponderación canal por canal a los mapas originales.

En la Figura 2.11 se muestra la arquitectura que sigue este bloque de atención.

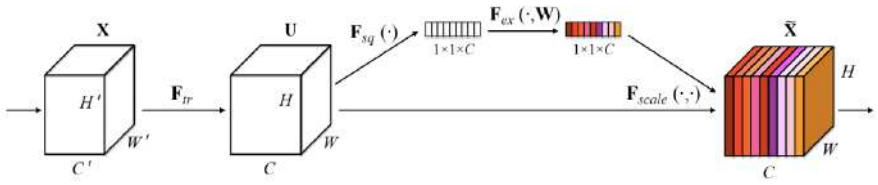


Figura 2.11: Estructura general Block SE según [25].

2.2.8. Características morfológicas

Como parte del procesamiento de imágenes, las características morfológicas y los descriptores de forma son herramientas utilizadas para caracterizar y analizar objetos en una imagen, permitiendo tareas como reconocimiento, clasificación y segmentación.

Estas características capturan propiedades geométricas, estadísticas y frecuenciales de los objetos, proporcionando representaciones robustas que facilitan la comparación y el análisis. Las características de forma utilizadas en este trabajo son:

- Momentos Zernike: Los momentos de Zernike son descriptores de región usados para caracterizar la forma de un objeto en una imagen proyectándola sobre una base ortogonal de polinomios complejos definidos en el disco unitario. Estos momentos son invariantes a la rotación [26].
- Momentos Geométricos: Los momentos geométricos son descriptores de región que capturan propiedades estadísticas de la distribución de píxeles en una imagen, como el área, centroide y orientación. Son menos robustos frente a transformaciones complejas [26].
- Coeficientes de Gabor: Los filtros de Gabor son descriptores de textura que combinan análisis espacial y frecuencial para capturar patrones locales. Están basados en una función gaussiana modulada por una onda sinusoidal y son sensibles a la orientación y escala de texturas [27].
- Coeficientes de Haralick: Estos coeficientes son descriptores de textura derivados de la matriz de co-ocurrencia de niveles de grises (GLCM), que analiza la distribución estadística de pares de píxeles. Ofrece robustez frente a rotaciones y traslaciones [28].
- Descriptores de Fourier: Los descriptores de Fourier sirven para describir los contornos de los objetos en una imagen utilizando la transformada de Fourier. Estos son ideales para describir contornos cerrados y clasificar formas [26].
- Local Binary Pattern (LBP): Los patrones binarios locales son descriptores de textura que codifican la relación entre un píxel y sus vecinos. Son invariantes a rotaciones [29].

2.3. Métricas de evaluación

Para evaluar el rendimiento de los modelos de clasificación o segmentación de imágenes, se utilizan diferentes métricas.

2.3.1. Clasificación de imágenes

Para la clasificación de imágenes las métricas de evaluación que se tendrán en cuenta son Matriz de confusión, *Accuracy*, *Recall*, *Precision*, *AUC* y *F1-Score* [5].

2.3.1.1. Matriz de confusión

La matriz de confusión se utiliza para evaluar un algoritmo de clasificación de manera resumida. Esta matriz incluye cuatro valores esenciales que describen el tipo de predicción realizada, que se pueden observar en la Figura 2.12:

- Verdaderos Positivos (VP): cantidad de instancias donde tanto la variable predicha y la real son positivas.
- Verdaderos Negativos (VN): cantidad de instancias donde tanto la variable predicha y la real son negativas.

- Falsos Positivos (FP): cantidad de instancias donde la variable predicha es positiva y la variable real es negativa.
- Falsos Negativos (FN): cantidad de instancias donde la variable predicha es negativa y la variable real positiva

Con lo anterior se pueden obtener las métricas de rendimiento como *Accuracy, Precision, Recall* y *F1-Score*.

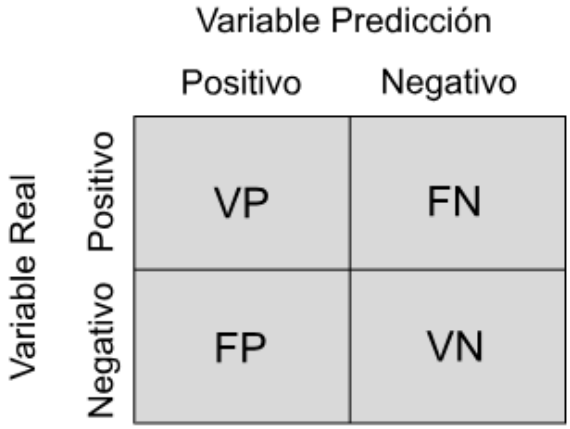


Figura 2.12: Matriz de confusión, elaboración propia.

2.3.1.2. *Accuracy*

El *accuracy*, también conocido como exactitud, es la tasa de valores correctamente clasificados por sobre la cantidad total de valores a clasificar.

Su rango va entre $[0, 1]$, donde 1 indica una perfecta clasificación.
La Ecuación 2.7 corresponde a la manera en la cual se obtiene el valor de esta métrica utilizando los 4 valores esenciales de la matriz de confusión.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.7}$$

2.3.1.3. *Recall*

El *Recall*, también conocido como sensibilidad o Tasa de Verdadero Positivos (TPR), es la proporción de instancias positivas reales que fueron identificadas correctamente por el modelo de clasificación.

Mediante la Ecuación 2.8 se calcula esta métrica considerando los 4 valores esenciales de la matriz de clasificación.

$$Recall = \frac{VP}{VP + FN} \tag{2.8}$$

Un alto *recall* significa que el modelo está detectando la mayoría de los valores positivos reales. En el caso contrario, indica que el modelo pasa por alto los valores positivos.

2.3.1.4. *Precision*

La precisión indica la tasa entre la cantidad de predicciones positivas que fueron realmente correctas, sobre la cantidad total de valores positivos.

La Ecuación 2.9 muestra la forma en que se calcula esta métrica utilizando los 4 valores esenciales de la matriz de clasificación.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

Un alto *precision* significa que, cuando el modelo predice un valor positivo, es altamente probable que sea correcto. En cambio, un bajo *precision* indica que las predicciones de valores positivos realizadas por el modelo son incorrectas.

2.3.1.5. Area Under Curve (AUC)

El área bajo la curva Receiver Operating Characteristic (ROC), representada en la Figura 2.13, (AUC, por sus siglas en inglés) el cual representa el rendimiento del modelo de clasificación en distinguir entre valores positivos y negativos. A mayor área bajo la curva, mejor es el rendimiento del modelo.

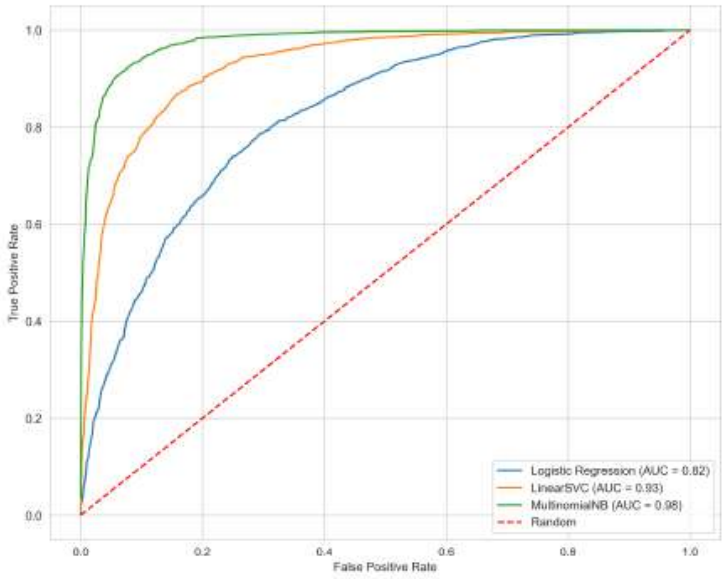


Figura 2.13: Curva ROC, según [5]

2.3.1.6. F1-Score

El F1-Score, también conocido como medida F, es una combinación de las métricas *precision* y *recall*. Específicamente, entrega la media armónica entre las dos métrica mencionadas anteriormente.

Para calcular el F1-score, se utiliza la Ecuación 2.10, donde se utilizan los 4 valores esen-

ciales de la matriz de clasificación.

$$\text{F1-Score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot VP}{2 \cdot VP + FP + FN} \tag{2.10}$$

Esta métrica refleja la capacidad del modelo para identificar correctamente los ejemplos relevantes del conjunto de datos sin cometer demasiados errores.

2.3.2. Segmentación

Las métricas de evaluación para la segmentación de imágenes que se utilizan con mayor frecuencia son las siguientes [5]

2.3.2.1. Coeficiente de Sorense-DICE

El coeficiente de Sorence-Dice es una métrica que mide la superposición entre el área de la segmentación predicha y el área de la segmentación real.

Respecto al uso de este coeficiente en la segmentación de imágenes, los rangos que se manejan son entre 0 y 1, donde 1 indica una coincidencia perfecta entre la segmentación real y la segmentación predicha. En cambio, un valor de 0 indica que no hay superposición.

La fórmula en que se calcula está representada por la Ecuación 2.11, donde A y B son conjuntos de datos.

$$DICE = \frac{2|A \cap B|}{|A| + |B|} \tag{2.11}$$

2.3.2.2. Distancia de Hausdorff

La distancia Hausdorff mide la distancia máxima entre los puntos de los bordes de la segmentación real y la segmentación predicha, como se muestra en la Ecuación 2.13. Una forma intuitiva de entender esta métrica es que cuando hay una distancia Hausdorff pequeña significa que los bordes de las áreas segmentadas están cerca entre sí.

$$h(A, B) = max_{a \epsilon A} min_{b \epsilon B} \left\| a - b \right\|_2 \tag{2.12}$$

$$d_H(A, B) = max(h(A, B), h(B, A)) \tag{2.13}$$

donde $h(A, B)$ es la distancia dirigida de un conjunto A a un conjunto B, la cual calcula la mayor distancia desde cualquier punto A al punto más cercano en B utilizando la distancia euclidiana $\left\| a - b \right\|_2$, como aparece en la Ecuación 2.12. $h(B, A)$ es la distancia dirigida del conjunto B al conjunto A [30].

2.3.2.3. Intersection Over Union

La Intersection over Union (IoU), también conocido como índice de Jaccard, mide la proporción de la intersección entre las áreas de segmentación real y predicha por sobre la unión de estas.
Se calcula mediante la Ecuación 2.14 donde A y B son conjuntos, para más claridad se tiene la Figura 2.14.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.14}$$

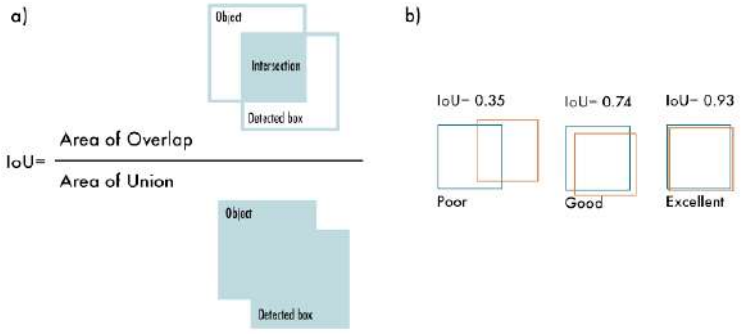


Figura 2.14: Intersection Over Union (IoU). a) Cálculo de IoU mediante la división entre la intersección de dos conjuntos entre la unión de estos. b) tres ejemplos de cálculos de IoU, Según [5]

2.3.3. AveragePrecision

Métrica de evaluación utilizada en *deep learning*, especialmente para evaluar modelos de detección de objetos y recuperación de información. La AveragePrecision(AP) mide la calidad de las predicciones al calcular el promedio de los valores *precision* en cada elemento en diferentes niveles de recuperación de información, comúnmente en tareas como la detección de cajas delimitadoras.

Se calcula como el área bajo la curva *precision-recall*, reflejando la capacidad del modelo para identificar y clasificar objetos correctamente, tomando valores entre 0 y 1, donde 0 indica que el modelo no es capaz de identificar correctamente ningún objeto y 1, un desempeño perfecto [5].

2.4. Base de datos

A continuación se describen las bases de datos internacionales y en la Figura 2.16 se observan muestras de cada base de datos.

2.4.1. SCIAN-MorphoSpermGS

Base de datos pública de Scientific Image Analysis Lab (SCIAN) de la Universidad de Chile, consiste en 2665 imágenes de 40×40 píxeles, teñidas con hematoxilina-eosina y posee cinco clases, una de normalidad y cuatro restantes para identificar diferentes tipos de anormalidades en la cabeza de los espermatozoides.
Las imágenes fueron capturadas utilizando microscopía óptica de campo claro (Axiostar Plus, Carl Zeiss Inc, Wetzlar, Alemania), un zoom de 63x (oil, NA 1.4) con un adaptador de 0.63x y una cámara digital (scA780-54gc, Basler AG, Ahrensburg, Alemania) [31].

2.4.2. HuSHeM

Base de datos pública Human Sperm Head Morphology (HuSHeM) del Isfahan Fertility and Infertility Center. Consiste en 216 imágenes de 131×131 píxeles, teñidas con el método Diff-Quick y posee cuatro clases, una de normalidad y tres de defectos [32]. Las imágenes fueron obtenidas utilizando un microscopio Olympus CX21 con un lente objetivo de 100x y un ocular de 10x, junto con una cámara a color Sony (modelo No SSC-DC58AP).

2.4.3. SMIDS

Base de datos internacional The Sperm Morphology Image Dataset (SMIDS) que posee 3000 imágenes teñidas con un ensayo modificado de hematoxilina-eosina (puede incluir cambios en las concentraciones de hematoxilina y eosina, diferente tiempo de exposición a los tintes o adaptación al equipo óptico), capturadas con un sistema basado en *smartphones*, validado para detección y conteo de espermatozoides móviles. Posee tres clases: normalidad, anormalidad y no es espermatozoide [33].
Si bien el mecanismo de obtención no especifica el modelo del *smartphone* utilizado en la captura de imágenes, el sistema empleó un diseño que acopla este teléfono inteligente a un adaptador óptico, como se observa en la Figura 2.15. Las especificaciones ópticas de la microscopia y la información de las imágenes oculares registradas son: microscopio Olympus BX50, magnificación de 20x, iluminación 12V/100W lámpara halógena, resolución 3890x3000 y resolución del espacio de objetos $1\mu m/px$ [34].

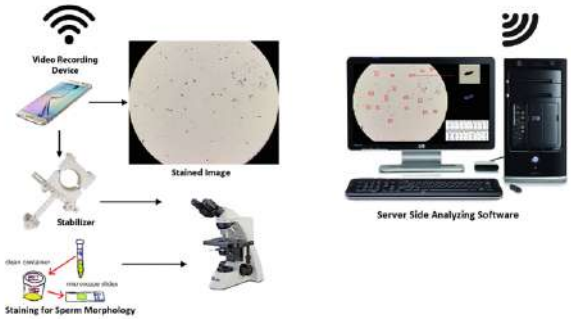
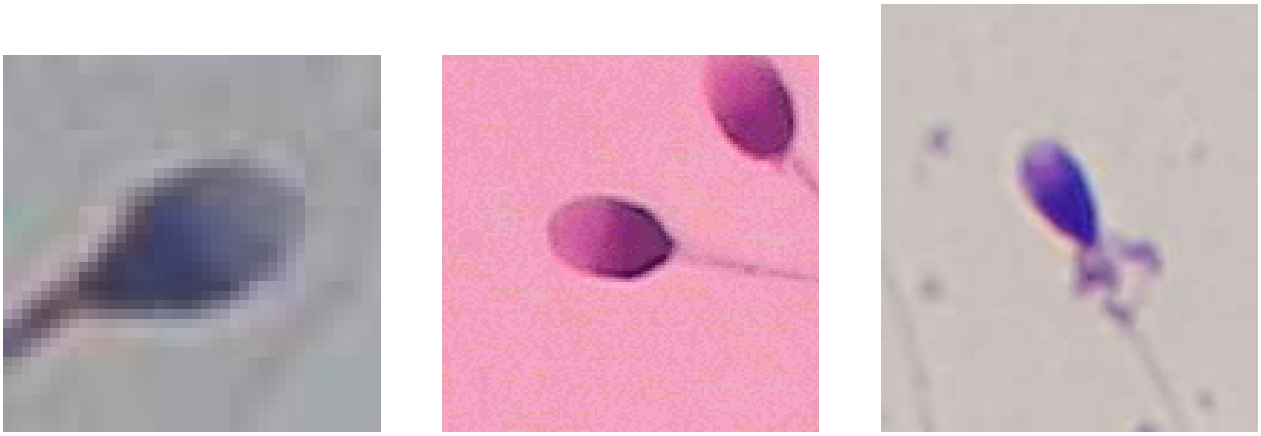


Figura 2.15: Enfoque de adquisición de datos utilizado para obtener imágenes de SMIDS según [34]



(a) SCIAN (b) HuSHeM (c) SMIDs

Figura 2.16: Ejemplos de imágenes en cada base de dato internacional. (a) SCIAN-MorphoSpermGS (b) HuSHeM y (c) SMIDS

Capítulo 3

Estado del arte

3.1. Procesamiento de imágenes

En esta sección se detallan los antecedentes de los principales enfoques y avances del procesamiento de imágenes orientados al análisis morfológico de espermatozoides.

3.1.1. Clasificación

El artículo de Ilhan, H. et al. [35] propone una técnica basada en *transfer learning* con las arquitecturas GoogleNet y VGG16 pre-entrenadas con ImageNet. Los conjuntos de datos utilizados fueron SCIAN-MorphoSpermGS, Human Sperm Head Morphology (HuSHeM) y Sperm Morphology Image (SMIDS), y se utilizaron técnicas de aumento de datos como rotaciones, reescalados y volteos en cada conjunto de datos. Además, utilizó una validación cruzada de 5 conjuntos, dividiendo los datos en un 80 % para entrenamiento y un 20 % para prueba.
Las arquitecturas fueron ajustadas usando *fine-tunning* en dos etapas: primero con SMIDS para mejorar la adaptación a las imágenes de espermatozoides y luego con los datos de HuSHeM y SCIAN-Morpho. Posteriormente, se combinaron las predicciones de ambas redes mediante la estrategia de *soft-voting*, lo que mejoró la clasificación, logrando un *accuracy* del 73,2 % para SCIAN-Morpho.

En la investigación de Riordon, J. et al. [36], se utilizó arquitectura VGG16 con *transfer learning* pre-entrenada con ImageNet, donde se reemplazaron las dos últimas capas *fully connect* con nuevas capas *fully-connect* de tamaño 128 y activación ReLu. Se agregaron dos capas de *dropout* con el fin de prevenir el sobreajuste y finalmente se añadió una capa *Softmax* para realizar la clasificación, logrando un 62 % de *accuracy*.
En cuanto a los datos, se realizaron aumentos mediante rotaciones, desplazamientos, volteo y conversión a Red, Green, Blue (RGB) con el fin de asegurar consistencia en el formato de las imágenes durante el procesamiento. También se utilizó una validación cruzada de 5 conjuntos, destinando un 80 % de los datos al entrenamiento y un 20 % para prueba.

La publicación [37] propone un modelo llamado Morphological Classification of Human Sperm Heads (MC-HSH), una arquitectura especializada de CNN compuesta por 53 capas convolucionales distribuidas en 5 bloques, que utilizan filtros de distintos tamaños como 1×1, 3×3 y 5×5 para la extracción de características. El modelo también incluye la normalización por lotes, activación LeakyReLu y concatenación para mejorar la clasificación.

Entre las técnicas de aumento de datos se encuentran las rotaciones, desplazamientos, volteos y la conversión de imágenes a canal RGB antes de que ingresen al modelo. Se utilizó validación cruzada de 5 conjuntos, donde el 80 % de los datos fueron destinados al entrenamiento y el 20 % a pruebas.
Cabe destacar que en este artículo se realiza la comparación entre los datos SCIAN con parcial acuerdo y total acuerdo, logrando un mejor desempeño en este último con un *accuracy* de 77 %.

La investigación realizada por Spencer, L. et al. [38] presenta el uso de un modelo estilo ensamble de CNNs que combina las arquitecturas VGG16, VGG19, DenseNet-161 y Resnet-34 modificada, eliminando la última serie de capas convolucionales. La unión de las arquitecturas mencionadas forman un meta-clasificador para clasificar la morfología de las cabezas de espermatozoides.
Para el conjunto de datos SCIAN con total acuerdo, se ajustó la capa de salida de las arquitecturas a la cantidad de clases del conjunto, en este caso 5 clases. Además, se optimizaron los hiperparámetros mediante búsqueda bayesiana, y se añadió al meta-clasificador dos capas ocultas, normalización por lotes y *dropout*.
El modelo realizó aumento de datos realizando solo volteos verticales, además de utilizar validación cruzada repetida de 3×5 conjuntos, dejando un 80 % de los datos para el entrenamiento y un 20 % para pruebas. Finalmente, se evaluó el desempeño de cada una de las arquitecturas y la del meta-clasificador (arquitecturas unidas), donde este último obtiene el *accuracy* más alto de 86,29 %.

El artículo [39] utiliza un modelo basado en seis diferentes arquitecturas de redes neuronales convolucionales, diseñadas desde cero para clasificar cabezas de espermatozoides. Estas redes, denominadas modelos A, B, C, D y F, incluyen entre 21 a 46 capas, cada una con configuraciones específicas en cuanto a filtros, tamaño de ventana, técnicas de normalización y capas de *dropout* para optimizar el rendimiento.
Las seis redes se combinaron utilizando técnicas de fusión a nivel de decisión como *soft-voting* y *hard-voting*, logrando un *accuracy* de 71,91 %.
Por otro lado, se realizó aumento de datos utilizando técnicas como *zoom*, desplazamiento, rotación y volteos con una limitación de rangos específicos para mantener las características originales de las imágenes. También, se utilizó validación cruzada de 5 conjuntos, donde 4 de ellos se utilizaron para el entrenamiento y el restante para pruebas, repitiendo el proceso hasta que cada conjunto sirviera como conjunto de prueba, para finalmente calcular la media de los resultados obtenidos.

En la investigación realizada por Ortumlu, O. et al. [40] se utilizan dos arquitecturas: MobileNet V1 con 91 capas y MobileNet V2 con un módulo mejorado con la estructura residual inversa y eliminando los estados no lineales en capas estrechas. Ambas redes son pre-entrenadas con ImageNet y utilizan *transfer learning*.
Los modelos mencionados utilizaron el conjunto de datos SCIAN-Morpho, al cual se le aplicó aumento de datos con técnicas de escalado, redimensionar, desplazamiento y recortes. Además, utilizó validación cruzada de 5 conjuntos dejando 906 imágenes para el entrenamiento y 226 para prueba.
MobileNet V1 fue el modelo que obtuvo el mayor *accuracy* de 67 %.

El artículo [41] propone un método basado en tres etapas, que consisten en generar pseudo-máscaras jerárquicas para segmentar la cabeza de espermatozoides. Luego, se usa la arquitectura DenseNet-201 pre-entrenada con ImageNet y se aplican las técnicas de destilación de información anatómica (AID) y*soft tuning* para la clasificación.
Los datos utilizados fueron los de SCIAN-Morpho, al cual se le realizó aumento de datos con ajuste de dimensiones, rotaciones, volteos verticales, brillo y contraste. Además, se usó validación cruzada de 5 conjuntos. Finalmente, este modelo obtiene un *accuracy* de 65,9 %.

En la publicación [42] presenta el modelo Sperm Head Morphology Classification (SHMCNet), que utiliza redes neuronales basadas en ShuffleNet V2. Inicialmente, se aplica un módulo de generación y refinamiento de máscaras para obtener los límites de la cabeza del espermatozoide. Luego, estas máscaras se introducen a la red de imágenes y a la red de máscaras del codificador de fusión respectivamente. Las características obtenidas por cada red se fusionan y pasan por un clasificador lineal para obtener la predicción de clases.
Se emplea la técnica de *soft mixup* en las imágenes y las máscaras para aumentar datos, además de utilizar las técnicas de rotación, volteo, *zoom* a 64×64 píxeles y desplazamiento. Se usó la validación cruzada de 5 conjuntos, aunque no se informa la partición de los conjuntos de datos
Por otro lado, logra un 73,5 % en SCIAN con parcial acuerdo y un 86,9 % con total acuerdo.

En la Tabla 3.1 se resumen las métricas de evaluación de cada modelo mencionado en los artículos anteriores.

Tabla 3.1: Resumen de métricas de evaluación de modelos de *deep learning* para la clasificación de espermatozoides descritos en el estado del arte. Todos los valores expresados en %.

Método *deep learning*	Accuracy	Recall	Precision	F1-Score
Data Augmentation Transfer Learning using VGG16 and GoogleNet (ImageNet + SMIDS) Decision Level Fusion (VGG16 +GoogleNet)	73,2	-	-	-
Data Augmentation Transfer Learning using VGG16 and GoogleNet (ImageNet)	72,08			
MC-HSH - Partial Acuerdo	63	68	56	61
MC-HSH - Total Acuerdo	77	88	64	74
DenseNet-161	77,01	87,36	65,37	70,26
Modified ResNet-34	74,67	87,36	60,43	65,81
VGG16	77,26	86,71	64,01	68,15
VGG19	76,9	87,13	62,77	67,54
Meta-Classifier	86,29	91,77	76,95	81,24
10 × Data Augmentation over Balanced Dataset Design of Six CNNBased Deep Learning Models Decision Level Soft Fusion	71,91	-	-	-
MobileNetV1	67	-	-	-
MobileNetV2	66	-	-	-
DenseNet-201 + Unsupervised anatomical feature distillation	65,9	68,9	58,7	63,2
SHMC-Net - Parcial Acuerdo	73,6	63,6	66,5	65
SHMC-Net - Total Acuerdo	86,9	83,4	86,7	85

3.1.2. Segmentación

En el artículo [43] se comparan las arquitecturas U-net y Mask-RCNN para segmentar partes del espermatozoide (cabeza, acrosoma y núcleo) utilizando el *dataset* SCIAN-

SpermSegGS. Antes de procesar los datos, se implementó una estrategia de validación cruzada de 5 conjuntos, dividiendo el conjunto de datos en un 70 % para entrenamiento y un 30 % para evaluación en cada iteración. Posteriormente, dentro del conjunto de entrenamiento, se aplicaron técnicas de aumento de datos, de los cuales un 20 % se reservó para validación.
El aumento de datos mencionado anteriormente consistió en utilizar 54 transformaciones geométricas que incluyeron rotaciones y técnicas de volteo, manteniendo forma y tamaño de los espermatozoides, logrando así, aumentar el tamaño del conjunto de datos en un 2500 %.

En el caso de la U-net, la arquitectura original fue modificada aplicando *transfer learning*, pre-entrenando el modelo con el conjunto de datos Data Science Bowl 2018, dada la similitud en tareas. Se utilizaron los pesos obtenidos de este pre-entrenamiento para mejorar la segmentación. Por otro lado, en Mask-RCNN se eliminaron las capas finales y se optimizaron los pesos usando el método de descenso de gradiente estocástico (SGD).

Los mejores resultados se obtuvieron con U-net utilizando *transfer learning*, logrando un coeficiente Dice de 0,96 para la segmentación de cabeza de espermatozoides.

En la Tabla 3.2 se muestran los coeficientes DICE de los modelos mencionados anteriormente para la segmentación.

Tabla 3.2: Métricas de los métodos de segmentación descritos en el estado del arte.

Método *deep learning*	Coeficiente DICE
Unet from scratch	0,92
Mask-RCNN from scratch	0,87
Unet con Transfer Learning	0,96
Mask-RCNN con Transfer Learning	0,92

Capítulo 4

Desarrollo

En este capítulo se detallan las actividades realizadas para llevar a cabo este trabajo de título.

4.1. Metodología

El desarrollo de este trabajo se estructuró en dos etapas principales:

- Segmentación de cabezas de espermatozoides mediante modelos basados en *deep learning.*
- Clasificación de cabezas de espermatozoides utilizando modelos de *deep learning* para diferenciar su normalidad.

4.1.1. Justificación metodológica

El primer paso de este trabajo consistió en la segmentación de imágenes para identificar regiones de interés, eliminando información irrelevante como el fondo y posibles manchas o *debris* propios del proceso de tinción. Esto reduce la complejidad del análisis posterior y permite que la clasificación se realice sobre datos más representativos.

En esta primera etapa, se segmentaron todas las imágenes de la base de datos SCIAN sin considerar su clase, con el objetivo de extraer regiones relevantes de manera uniforme, independientemente de su etiqueta. Posteriormente, la clasificación permitió determinar la clase a la que pertenece cada imagen.

Para la segmentación, se seleccionó la arquitectura U-Net debido a su destacado rendimiento en tareas de segmentación de imágenes, según lo revisado en el estado del arte. Además, se evaluó el modelo Mask R-CNN para comparar su desempeño en el conjunto de datos SCIAN.

En la etapa de clasificación de cabezas de espermatozoides, se optó por redes de aprendizaje profundo como ResNet-18, CNN y ShuffleNetV2, seleccionadas por su equilibrio entre precisión y eficiencia computacional, lo que las hace adecuadas para conjuntos de datos de tamaño pequeño y baja resolución, como el utilizado en este trabajo. También se implementó una estrategia de combinación de redes basadas en espacios latentes, integrando las características extraídas por el mejor modelo de clasificación (ResNet-18, CNN o ShuffleNetV2) y las

características morfológicas y descriptores de forma obtenidas desde el procesamiento de las máscaras resultantes del modelo de segmentación para mejorar la representación de los datos y optimizar el rendimiento de la clasificación inspirado en el artículo [38]. Se descartaron arquitecturas más complejas, como VGG, MobileNet, DenseNet, GoogleNet y ResNet-34 o superiores, debido a su mayor número de parámetros y capas más profundas, que aumentan el riesgo de sobreajuste en conjuntos de datos pequeños.

Los algoritmos de segmentación siguieron un proceso estructurado basado en la literatura del estado del arte, para garantizar un desarrollo robusto y una evaluación confiable, dividido en las siguientes etapas:

- Entrenamiento: El algoritmo ajusta sus parámetros para aprender patrones y relaciones en los datos, minimizando los errores de predicción.
- Validación: Se evalúa el rendimiento del modelo en un subconjunto de datos independiente. Esta etapa no modifica los parámetros, si no que sirve para ajustar los hiperparámetros del modelo (ejemplo: tasa de aprendizaje, número de épocas, etc.) y prevenir el sobreajuste, asegurando la capacidad de generalización ante nuevos datos.
- Evaluación: Se mide el rendimiento del modelo en un conjunto de datos independiente, no utilizado en las etapas previas, simulando un escenario real para evaluar su capacidad predictiva en datos desconocidos.

Por otra parte, los algoritmos de clasificación siguieron un esquema basado en el estado del arte, que consiste en realizar entrenamiento y validación con el método de validación cruzada del tipo Leave-One-Out (LOO), descrito en sección 2.2.3, debido al desbalance del conjunto de datos:

- Entrenamiento: El modelo se entrena utilizando todas las muestras del conjunto de datos, excepto una, ajustando sus parámetros para aprender patrones y relaciones en las imágenes.
- Validación: La muestra excluida del conjunto de entrenamiento se utiliza como conjunto de validación para evaluar el rendimiento del modelo. Este proceso se repite para cada muestra del conjunto de datos.

4.1.2. Entornos de desarrollo

Para el desarrollo de este trabajo se utilizó un Intel Core i5 de 4 núcleos, 16 GB de RAM y tarjeta GPU GeForce GTX 260 1 GB. Además, se contó con un computador servidor con procesador Intel Core i7-4930K de 6 núcleos, 64 GB de memoria RAM. Este posee la disponibilidad de trabajar con una tarjeta GPU GeForce GTX 980 4 GB y una tarjeta GPU GeForce RTX 4060 Ti 8 GB.
Todo el equipamiento está ubicado en el programa de Biología Integrativa del ICBM de la Facultad de Medicina de la Universidad de Chile.

4.2. Algoritmo de segmentación

4.2.1. Preprocesamiento de datos

La base de datos SCIAN se dividió en tres conjuntos: 70 % para entrenamiento, 15 % para validación y 15 % para evaluación. Esta proporción es ampliamente utilizada en la literatura del estado del arte.

La división se realizó a nivel de pacientes para evitar la filtración de datos (*data leakage*), ya que las muestras de un mismo paciente suelen presentar similitudes. Al asignar todas las muestras de un paciente exclusivamente a un solo conjunto, se previene que el modelo acceda a información de un paciente presente en múltiples conjuntos, lo que podría introducir sesgos y sobrestimar el rendimiento. Este enfoque garantiza una evaluación más realista y mejora la capacidad del modelo para generalizar a nuevos pacientes.

La distribución detallada de pacientes en cada conjunto se presenta en la Tabla A.1 del Anexo A.

4.2.2. Modelo U-net

El modelo U-Net se implementó utilizando la biblioteca MONAI (versión 1,4,0), diseñada específicamente para la segmentación de imágenes médicas en formato 2D RGB (dos dimensiones y tres canales). La red se configuró con parámetros adaptados a las características de la base de datos SCIAN:

- Dimensiones y canales: Se establecieron dimensiones espaciales de 2 (spatial_dims=2) para procesar imágenes 2D, con 3 canales de entrada (RGB) (in_channels=3) y 1 canal de salida (out_channels=1) para generar máscaras de segmentación binaria.
- Estructura del codificador: Se diseñaron cuatro capas en el trayecto descendente con canales $(32, 64, 128, 256)$, donde el último canal corresponde al del cuello de botella. Los pasos (*strides*) se configuraron como $(2, 2, 2)$ para reducir la resolución espacial por un factor de 2 en cada capa, salvo en el cuello de botella, permitiendo capturar características de mayor escala.
- Bloques residuales: Se incorporaron dos bloques residuales por nivel (num_res_units=2) para mejorar el flujo de gradientes y la capacidad de aprendizaje de la red.
- Optimización: Se utilizó la función de pérdida DiceLoss con una función de activación sigmoide, adecuada para segmentación binaria ya que permite calcular la similitud entre la máscara predicha y real, manejando el desbalance entre fondo y objeto. El optimizador *Adam* se configuró con una tasa de aprendizaje de 5×10^{-4} para estabilizar el entrenamiento.
- Entrenamiento: El modelo se entrenó durante 100 épocas, límite establecido para evitar el sobreajuste tras evaluar el rendimiento en el conjunto de validación.

La arquitectura se detalla en la Figura 4.1.

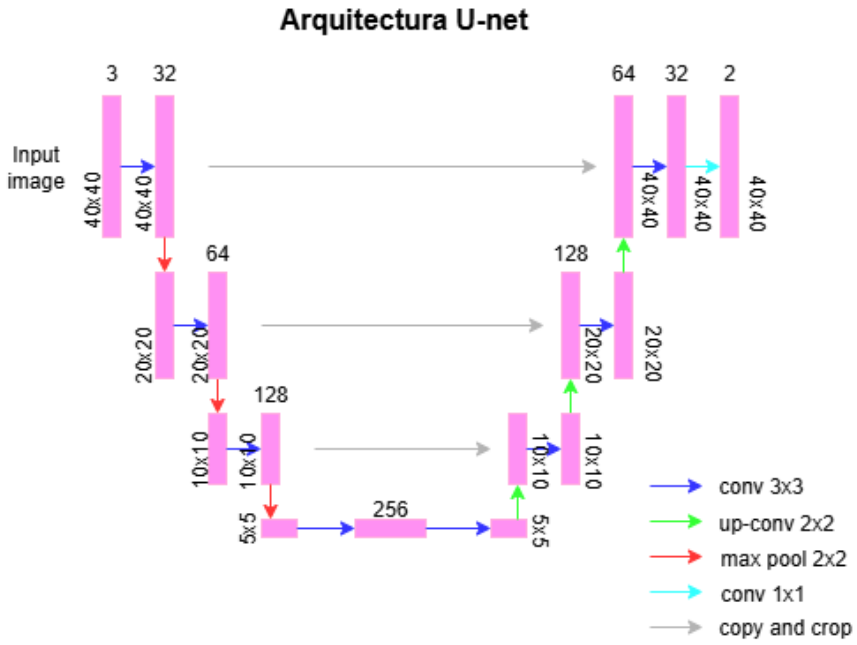


Figura 4.1: Arquitectura U-Net modificada.

Para los conjuntos de entrenamiento, validación y evaluación del modelo U-Net, se definieron transformaciones específicas utilizando la biblioteca MONAI, con el objetivo de aumentar datos sin interpolar información de las imágenes. Para el conjunto de entrenamiento, se aplicó la siguiente secuencia de transformaciones:

- *LoadImaged*: Carga imágenes y máscaras de segmentación desde los archivos.
- *EnsureChannelFirstd*: Garantiza que el canal de la imagen esté en la dimensión correcta, cumpliendo con el formato esperado por el modelo.
- *ScaleIntensityd*: Normaliza la intensidad de los píxeles para estandarizar los valores de las imágenes.
- *ThresholdIntensityd*: Aplica un umbral de 165 por canal, en las imágenes, asignando un valor 0 a los píxeles por debajo de este umbral, lo que reduce el ruido y mejora la calidad de las imágenes.
- *RandRotated*: Aplica rotaciones aleatorias con una probabilidad de 0,5 y un rango de 0° a 360°, preservando el tamaño original de las imágenes.

Para los conjuntos de validación y evaluación, se aplicaron las transformaciones *LoadImage, EnsureChannelFirstd, ScaleIntensityd* y *ThresholdIntensityd* con el mismo umbral de 165, omitiendo el aumento de datos para mantener la integridad de las imágenes y evaluar el modelo en condiciones representativas de los datos originales.

4.2.3. Mask R-CNN

El modelo Mask R-CNN utilizado se basa en la arquitectura GeneralizedRCNN propuesta por He, K et al. [15], diseñada para tareas de detección y segmentación de instancias. Se

implementó mediante la biblioteca Detectron2, desarrollada por Meta AI, debido a que Detectron1, su versión anterior, no es compatible con versiones recientes de Python.
La arquitectura consta de tres componentes principales:

- *Backbone*: Basado en una red ResNet-50 con una Pirámide de Características (FPN), extrae representaciones de características a múltiples escalas. La FPN emplea capas convolucionales de 1×1 para reducir la dimensionalidad de las características y capas de 3×3 para refinarlas en los niveles res2 a res5, que corresponden a las etapas de salida de las capas convolucionales de ResNet-50. Cada nivel (res2 a res5) presentan diferente resolución espacial. Además, se incorpora un bloque de *max pooling* para generar representaciones más robustas.

- Generador de propuestas de regiones (RPN): Este módulo identifica las regiones potenciales que podrían contener objetos. Utiliza una cabeza convolucional que predice *logits* de objeto (probabilidad de que una región contenga un objeto) y realiza ajustes de anclajes para refinar las coordenadas propuestas, optimizando la selección de áreas relevantes en la imagen.

- Cabezales de región de interés (ROIHeads): Procesa las regiones identificadas mediante ROIAlign, una técnica que alinea las características extraídas a nivel subpíxel, mejorando la localización precisa de estas regiones. Los ROIHeads incluyen dos ramas: una para la clasificación de objetos y la regresión de cajas delimitadoras, y otra para la segmentación de máscaras, que utiliza capas convolucionales seguidas de una deconvolución para generar máscaras binarias por instancia.

Se utilizó la configuración base de Mask R-CNN R50-FPN 3× del modelo zoo de Detectron2, sin pre-entrenamiento. El modelo se configuró para una sola clase de interés, con una tasa de aprendizaje de 1×10^{-4} y un máximo de iteraciones que aseguró un entrenamiento equivalente a 100 épocas (aproximadamente 31400 iteraciones).

4.3. Algoritmos de clasificación

4.3.1. Preprocesamiento de datos

El preprocesamiento para la clasificación de cabezas de espermatozoides consistió en asignar tres clases: Clase 0 (normalidad), Clase 1 (anormalidad) y Clase 2 (imágenes con discordancia en el etiquetado, debido a la falta de consenso entre etiquetadores).
A partir de las imágenes, se extrajeron características morfológicas y de textura, detalladas en las Tablas 4.1, 4.2, 4.3. Estas características se organizaron en un *dataframe* que incluía la ruta de cada imagen y su clase correspondiente. Posteriormente, se filtró el *dataframe* eliminando las imágenes de la clase 2 y aquellas con valores nulos en sus características, garantizando un conjunto de datos limpio para el entrenamiento y validación de los modelos.

Debido al notorio desbalance entre las clases 0 y 1, una división tradicional como $70-30$ o $80-20$ no aseguraba una distribución equilibrada de imágenes por clase, especialmente para la clase minoritaria. Por ello, se empleó el método de validación cruzada de tipo *Leave-One-Out* (LOO) a nivel de paciente. En este método, los datos de cada paciente se utiliza una vez como conjunto de validación, mientras que los datos de los demás pacientes se emplean para el entrenamiento, repitiendo el proceso hasta que todos los pacientes hayan sido utilizados

como validación. Este enfoque maximiza el uso de los datos disponibles y mejora la robustez del modelo frente al desbalance.
La distribución de pacientes por clase se detalla en la Tabla A.2 del Anexo A, así como también se observa en la Tabla A.3 del Anexo A el desbalance del *dataframe* filtrado, donde 508 muestras son normales y 1763 son anormales, es decir una proporción 1 : 3 aproximadamente.

Además se utilizaron las siguientes transformaciones para el conjunto de entrenamiento:

- *RandomRotation*: Aplica rotaciones aleatorias dentro de un rango de 0° a 360°.
- *RandomHorizontalFlip*: Invierte aleatoriamente las imágenes de izquierda a derecha.
- *RandomVerticalFlip*: Invierte aleatoriamente las imágenes de arriba hacia abajo
- Normalización: Ajusta los valores de los píxeles de las imágenes para que tengan una media y desviación estándar especifica, estandarizando las imágenes y facilitando el entrenamiento.

Para el conjunto de validación se utilizó solo la normalización como transformación.

4.3.1.1. Extracción de características

La extracción de características se realizó a partir de las máscaras generadas por el modelo U-Net en la etapa de segmentación, las cuales incluían la cabeza del espermatozoide etiquetado como objetivo, junto con partes de otros espermatozoides ubicados en los bordes de la imagen (cabezas de espermatozoides incompletos o cortados). Para abordar este problema, se adaptó el código de extracción de características para excluir los objetos detectados en los bordes, asegurando que solo se consideraran las cabezas de espermatozoides completas dentro de la máscara. Posteriormente, se extrajeron las características morfológicas y de textura mencionadas, las cuales se detallan en las Tablas 4.1, 4.2 y 4.3, obteniendo un vector de 77 características.

Tabla 4.1: Características de forma.

Características morfológicas	Número de características
Área	1
Perímetro	1
Excentricidad	1
Regularidad	1
Rectangularidad	1
Circularidad	1
Curvatura Máxima	1
Curvatura Mínima	1

Tabla 4.2: Descriptores de forma.

Descriptores de forma	Número de características
Morfológicas	8
Fourier	10
Momentos geométricos	5
Momentos de Zernike	25
Convexidad	1
Elipticidad	1

Tabla 4.3: Descriptores de textura.

Descriptores de textura	Número de características
Coeficientes de Gabor	4
LBP	10
Haralick	13

4.3.2. Convolutional Neural Network (CNN)

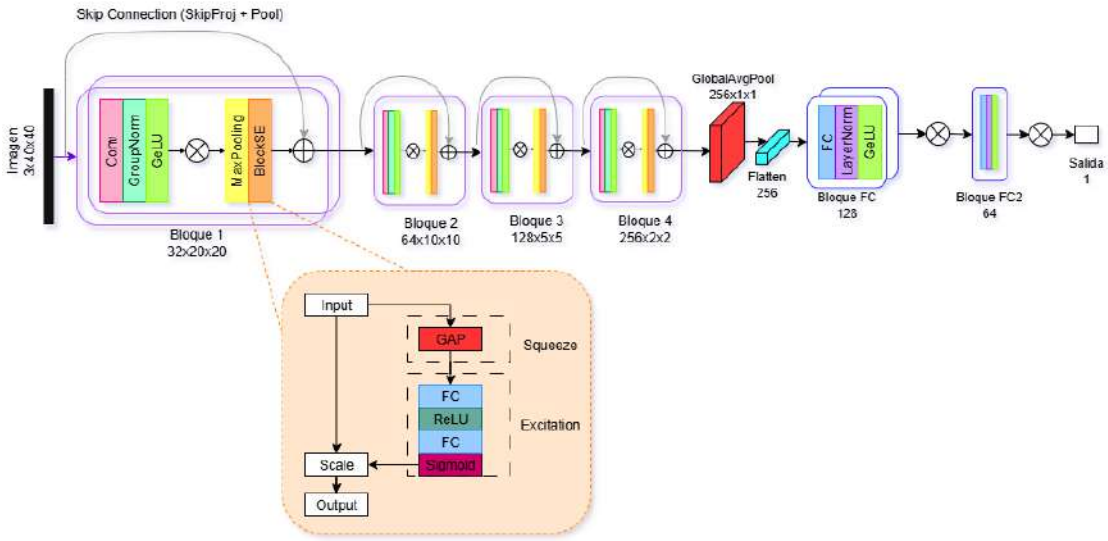


Figura 4.2: Arquitectura de CNN

La red CNN diseñada para clasificar imágenes de 40×40 píxeles, ilustrada en la Figura 4.2 se implementó con las siguientes características:

- Arquitectura: Consta de cuatro bloques convolucionales, cada uno con una capa convolucional de 3×3 y padding de 1 para detectar bordes y texturas, seguida de un max pooling de 2×2 para reducir la dimensionalidad. Cada bloque incluye:
  - Normalización por grupos para estabilizar el aprendizaje

  - Activación GeLU para introducir no linealidades suaves, mejorando el manejo de gradientes.
  - Dropout de 0,1 para aplicar regularización, desactivando neuronas aleatoriamente durante el entrenamiento y mitigando el sobreajuste.
  - Módulo Squeeze-and-Excitation (SE) con reducción de 16 canales y activación sigmoide, que pondera los canales más relevantes.
  - Conexiones residuales con proyección: Cada bloque incorpora una conexión residual con un factor α aprendido por la red, que pondera la contribución de la conexión residual. Cuando las dimensiones de entrada y salida del bloque difieren, se utiliza una proyección aprendida *(skip projection)* implementada mediante una capa convolucional de 1×1, asegurando compatibilidad dimensional para la combinación de características.

- Capa de salida: Incluye un pool global adaptativo para reducir la dimensionalidad de los mapas de características, seguido de tres capas totalmente conectadas (256 a 128, 128 a 64, 64 a 1) con:
  - Normalización por capas (LayerNorm)
  - Activación GeLU y dropout de 0,3 para regularización.
  - Activación sigmoide final, que genera una salida normalizada entre 0 y 1 para la clasificación binaria.

- Entrenamiento: Se entrenó durante 100 épocas con una tasa de aprendizaje de 1×10^{-3}, seleccionada tras observar sobreajuste con tasas de aprendizaje de 1×10^{-4} y 5×10^{-4}, y así como con más de 100 épocas.

Se realizaron tres experimentos con diferentes funciones de pérdida: (1) Cross Entropy Loss, (2) BCEWithLogitsLoss y (3) Focal Loss. Las dos primeras utilizaron submuestreo para balancear la base de datos, mientras que Focal Loss se aplicó directamente sobre la base de datos desbalanceada. Estas configuraciones permitieron evaluar el impacto de la función de pérdida y las estrategias de manejo del desbalance en la clasificación binaria.

4.3.3. ShuffleNetV2

La red shuffleNetV2 se adaptó para clasificar imágenes de 40×40 píxeles con las siguientes modificaciones:

- Convolución inicial: Una capa convolucional de 3×3 (stride=2, padding=1) transforma los 3 canales de entrada a 64 canales, seguida de normalización por lotes (BatchNorm) y activación ReLU. No se aplicó max pooling adicional para preservar la resolución espacial en imágenes pequeñas.

- Etapas y bloques: Se diseñaron tres etapas con 4, 8 y 4 bloques *InvertedResidual*, respectivamente, y canales de salida $(128, 256, 512)$. Cada bloque incluye convoluciones puntuales (1×1), convoluciones profundas (3×3), BatchNorm y ReLU. Cuando stride=1, los canales se dividen en dos ramas (una sin procesar y otra transformada), con stride=2 ambas ramas se procesan para reducir la resolución.

- Channel shuffle: Configurado con groups=1 para simplificar la mezcla de canales, adaptándose a imágenes de baja resolución.
- Capa final: Una convolución de 1×1 eleva los canales a 1024, seguida de un *global average pooling* y una capa totalmente conectada ajustada a 1 unidad para BCEWithLogitsLoss y Focal Loss en clasificación binaria, y ajustada a 2 para Cross Entropy Loss.
- Entrenamiento: La red se entrenó durante 100 épocas con una tasa de aprendizaje de 1×10^{-3}, seleccionada tras observar sobreajuste con tasas de 1×10^{-4}, 5×10^{-4} y más de 100 épocas.

La arquitectura se detalla en la Figura 4.3.

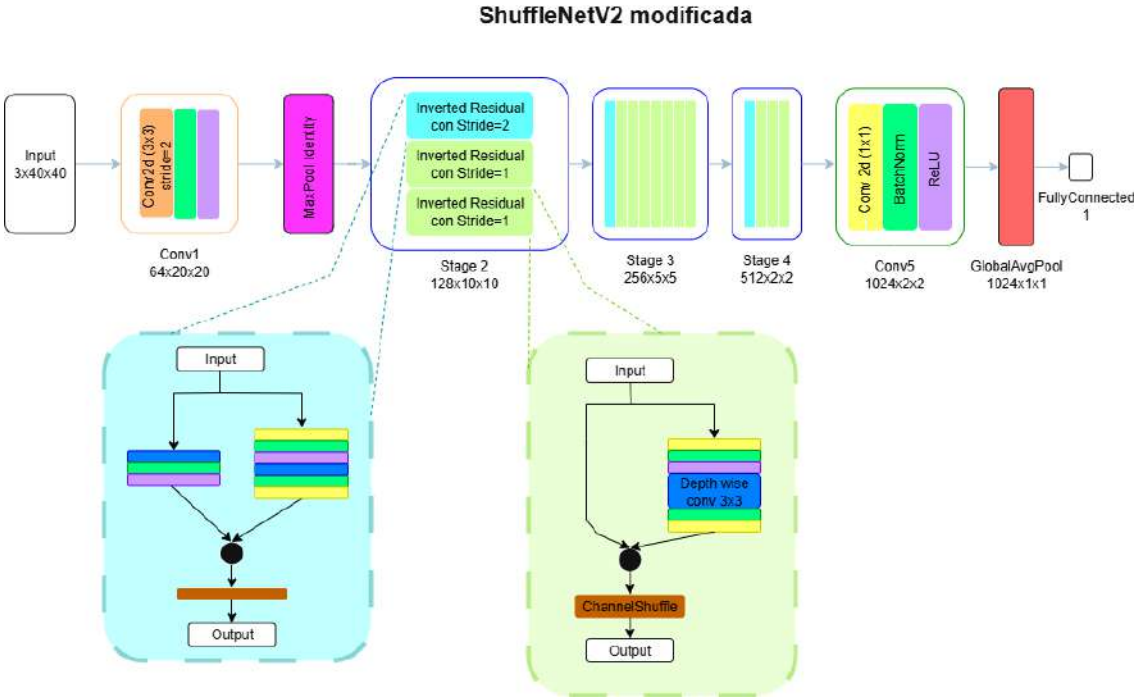


Figura 4.3: Arquitectura ShuffleNetV2

Se realizaron tres experimentos con diferentes funciones de pérdida: (1) Cross Entropy Loss, (2) BCEWithLogitsLoss y (3) Focal Loss. Las dos primeras utilizaron submuestreo para balancear la base de datos, mientras que Focal Loss se utilizó sobre la base de datos desbalanceada. Estas configuraciones permitieron evaluar el impacto de la función de pérdida y el manejo del desbalance de clases en la clasificación binaria.

4.3.4. Resnet-18

La arquitectura ResNet-18 se adaptó con las siguientes características:

- Convolución inicial: Una capa convolucional de 3×3 (stride=1, padding=1) con 32 canales de salida en lugar de los 64 estándar, seguida de BatchNorm y ReLU. El *max pooling* se ajustó a un kernel de 2×2 (stride=1, padding=1) , lo que mantiene la resolución espacial en comparación a otras.

- Estructura de capas: Cuatro etapas con bloques residuales básicos (2 bloques por etapa) y canales de salida $(32, 64, 128, 256)$. Los strides se configuraron como $(1, 1, 2, 1)$ para controlar la reducción espacial por etapa, adaptándose a imágenes de entrada pequeñas.

- Capas de salida: Un AdaptiveAvgPool2d reduce los mapas de características a 1×1, seguido de una capa totalmente conectada ajustada a 1 unidad para generar una salida escalar compatible con BCEWithLogitsLoss y Focal Loss, o ajustada a 2 para Cross Entropy Loss.

- Entrenamiento: La red se entrenó durante 100 épocas con una tasa de aprendizaje de 1×10^{-3}, seleccionada tras observar sobreajuste con tasas de 1×10^{-4}, 5×10^{-4} y más de 100 épocas.

La arquitectura ResNet-18 adaptada se detalla en la Figura 4.4.

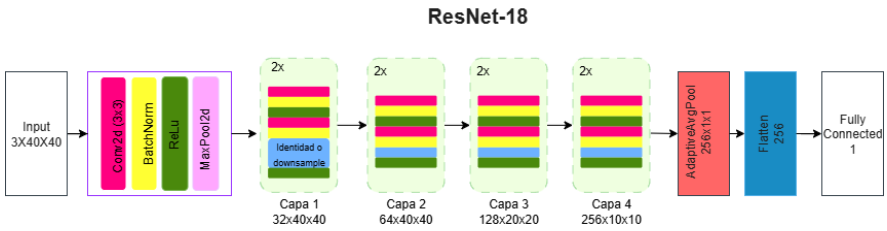


Figura 4.4: Arquitectura de ResNet-18.

Se realizaron tres experimentos con diferentes funciones de pérdida: (1) Cross Entropy Loss, (2) BCEWithLogitsLoss y (3) Focal Loss. Las dos primeras utilizaron submuestreo para balancear la base de datos, mientras que Focal Loss se utilizó sobre la base de datos desbalanceada. Estas configuraciones permitieron evaluar el impacto de la función de pérdida y el manejo del desbalance de clases en la clasificación binaria.

4.3.5. Red espacios latentes

Se diseñó una estrategia basada en espacios latentes, seleccionando la red de mejor rendimiento (evaluada mediante métricas como *Accuracy*, *Precision*, *Recall*, *F1-score* y *AUC*) entre las arquitecturas descritas previamente, de las cuales el mejor rendimiento fue por parte de ShuffleNetV2. Las modificaciones y el proceso fueron los siguientes:

- Extracción de características de la red de deep learning: Se congeló la última capa totalmente conectada de la red seleccionada para conservar las características del espacio latente extraídas por las capas convolucionales. Estas características se obtuvieron tras aplicar un *global average pooling*, generando un vector de características por imagen.

- Concatenación: Los vectores de características extraídos de la red se concatenaron con las características morfológicas y de textura, detalladas en las Tablas 4.2 y 4.3, para formar un vector de entrada combinado.

- Clasificador de vectores: Se implementó una red totalmente conectada en PyTorch para procesar el vector concatenado, con la siguiente arquitectura:

  – Capa lineal que reduce la dimensionalidad del vector de entrada a 32 unidades.

- Normalización por lotes para estabilizar el aprendizaje.
- Dropout 0,3 para mitigar el sobreajuste.
- Activación ReLU para introducir no linealidad.
- Segundo dropout de 0,5 para mayor regularización.
- Capa lineal final que genera una salida escalar para clasificación binaria.

- Entrenamiento: La red de espacios latentes se entrenó durante 100 épocas con una tasa de aprendizaje de 1×10^{-6}, utilizando el optimizador *Adam* y la función de pérdida Focal Loss.

La arquitectura se detalla en la Figura 4.5.

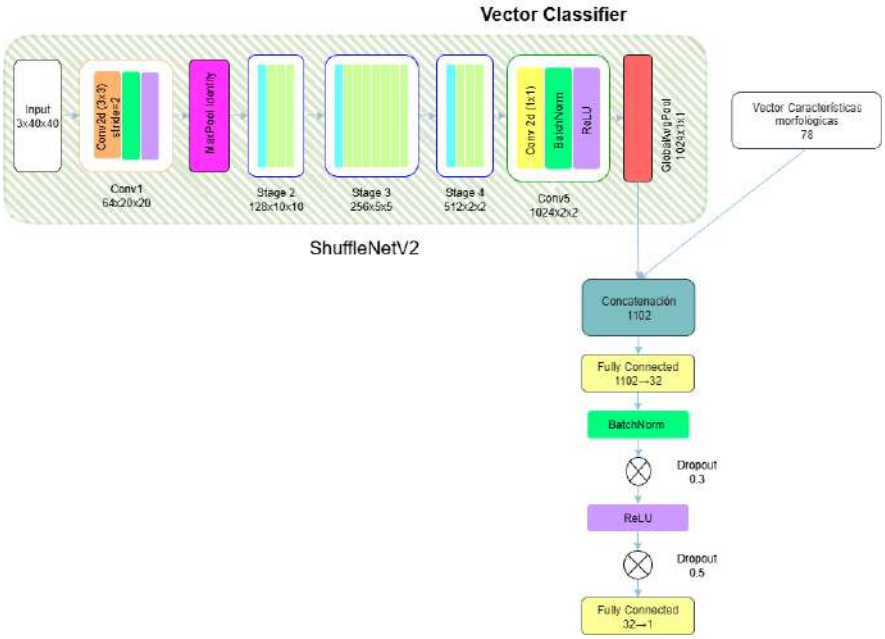


Figura 4.5: Arquitectura de modelo espacios latentes.

4.3.6. Random Forest

Para el algoritmo de machine learning RandomForest (RF) se realizó una búsqueda de hiperparámetros con GridSearchCV, optimizando para la métrica *Accuracy* sobre los parámetros detallados en la Tabla 4.4. Los mejores hiperparámetros seleccionados fueron:

- criterion: gini
- max_depth: None
- max_features: sqrt
- max_leaf_nodes: 10
- min_samples_leaf: 2
- min_samples_split: 2

- n_estimators: 200

El modelo se entrenó con estos hiperparámetros, utilizando random_state=42 para garantizar reproducibilidad. Se evaluó en cada paciente mediante métricas de *accuracy*, *f1_score*, *precision*, *recall* y *AUC* .

Tabla 4.4: Grilla hiperparámetros RandomForest.

Hiperparámetro	Valores
Criterion	[gini, entropy]
max_features	[5, 10, 20, 40]
max_leaf_nodes	[2, 4, 6, 8, 10]
n_estimators	[50, 100, 200]
max_depth	[None, 10, 20]
min_samples_split	[2, 5]
min_samples_leaf	[1, 2]
max_features	[sqrt, log2]

4.3.7. XGBoost

Se realizó una búsqueda de hiperparámetros con GridSearchCV optimizando para la métrica *Accuracy* sobre los parámetros detallados en la Tabla 4.5. Los mejores hiperparámetros fueron:

- gamma: 0,1
- learning_rate: 0,01
- max_depth: 6
- n_estimators: 100

El algoritmo XGBoost se configuró con objective=binary:logistic y eval_metric=logloss, utilizando random_state=42 para garantizar reproducibilidad.

Tabla 4.5: Grilla de hiperparámetros algoritmo XGBoost.

Hiperparámetro	Valores
Criterion	[gini, entropy]
max_features	[5, 10, 20, 40]
max_leaf_nodes	[2, 4, 6, 8, 10]
n_estimators	[50, 100, 200]
max_depth	[None, 10, 20]
min_samples_split	[2, 5]
min_samples_leaf	[1, 2]
max_features	[sqrt, log2]

Capítulo 5

Resultados y análisis

En este capítulo se presentan los resultados obtenidos de los algoritmos desarrollados, junto con su respectivo análisis y discusión.

5.1. Segmentación

El modelo U-Net desarrollado en este trabajo mostró un desempeño sobresaliente al segmentar la base de datos SCIAN. Durante el entrenamiento, las pérdidas en los conjuntos de entrenamiento y validación disminuyeron de manera estable, sin un aumento en la pérdida de validación tras 100 épocas, es decir sin sobreajuste, lo que indica una buena capacidad de aprendizaje y generalización. Este comportamiento se observa en la Figura 5.1.

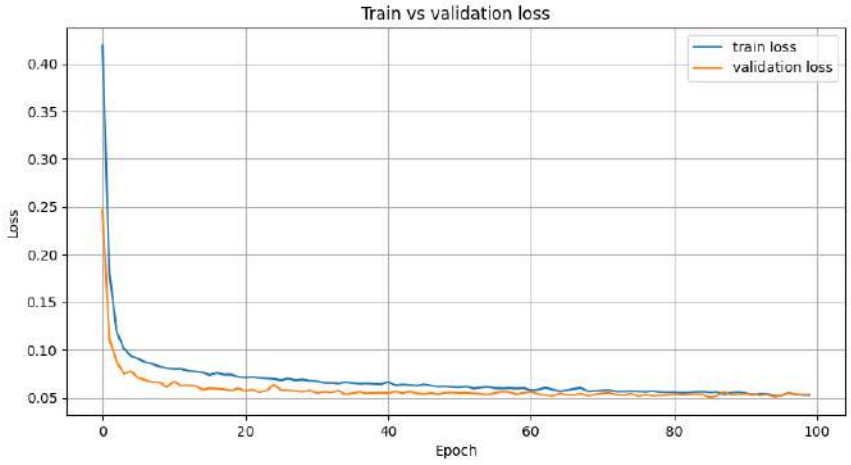
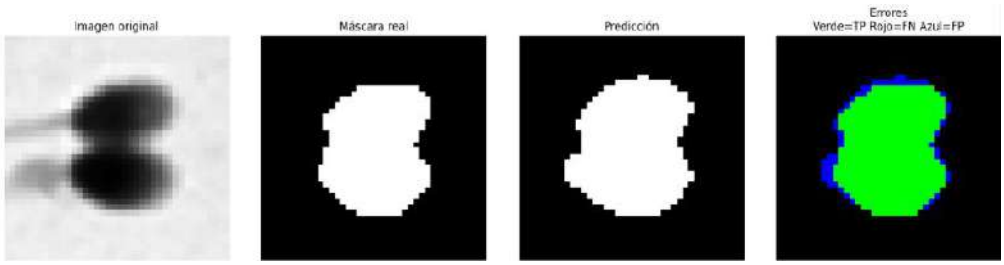


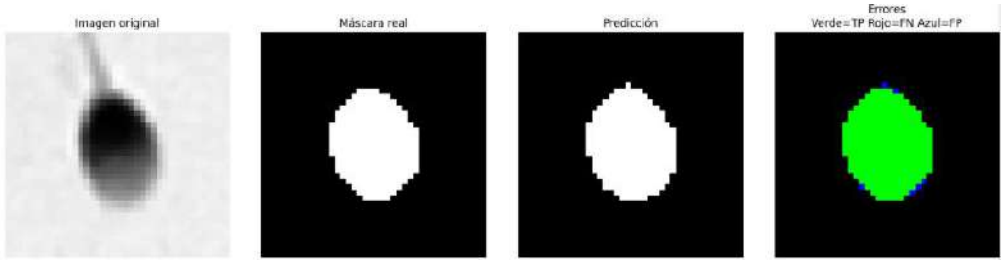
Figura 5.1: Gráfico de pérdidas en entrenamiento y validación - modelo U-Net.

Las métricas de evaluación del modelo de segmentación se presentan en la Tabla 5.1. El modelo alcanzó un coeficiente DICE promedio de 0,9471 y un IoU de 0,9038 en validación, y coeficiente DICE 0,9530 y IoU de 0,9132 en el conjunto de evaluación, reflejando una alta superposición entre las segmentaciones predichas y reales. Además, la distancia hausdorff promedio de 2,1249 píxeles en validación y de 1,9104 píxeles en evaluación, indica que los bordes segmentados están muy próximos a los reales, confirmando la precisión del modelo.

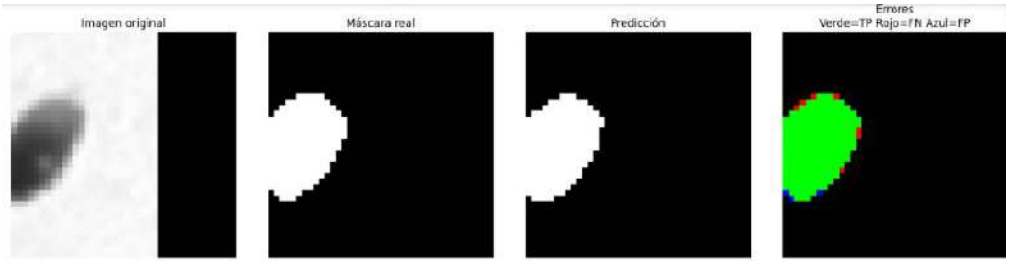
En la Figura 5.2 se pueden apreciar algunos ejemplos de la segmentación realizada por la U-Net, donde encontramos de izquierda a derecha la imagen real, máscara real, predicción y errores, donde en color verde se tienen los aciertos entre la predicción y la máscara real, en azul lo que segmentó adicional el modelo y en rojo lo que faltó por segmentar.



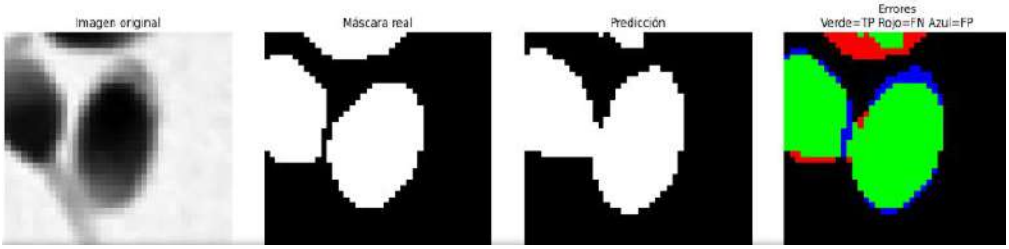
(a) Ejemplo 1.



(b) Ejemplo 2.



(c) Ejemplo 3.



(d) Ejemplo 4.

Figura 5.2: Ejemplo de resultados de segmentación utilizando arquitectura U-Net diseñada, siguen el siguiente orden: Imagen original, máscara real, predicción, errores. En color verde se tienen los aciertos entre la predicción y la máscara real, en azul lo que segmentó adicional el modelo y en rojo lo que faltó por segmentar.

Tabla 5.1: Métricas de evaluación en conjunto de validación y evaluación del modelo U-Net.

Conjunto	Coeficiente DICE	IoU	Distancia Hausdorff
Validación	$0{,}9471 \pm 0{,}0584$	$0{,}9038 \pm 0{,}0783$	$2{,}3705 \pm 3{,}1628$
Evaluación	$0{,}9530 \pm 0{,}0491$	$0{,}9132 \pm 0{,}0660$	$1{,}9104 \pm 2{,}6166$

Este modelo también se entrenó con un 60 % de la base de datos, y los resultados, incluyendo el gráfico de pérdidas en entrenamiento y las métricas de evaluación, se detallan en la Tabla B.1 y Figura B.1 del Anexo B.

Por otro lado, el modelo Mask R-CNN, mostró un comportamiento estable durante el entrenamiento en sus componentes: loss_mask (precisión de la máscara segmentada), loss_rpn_cls (clasificación de regiones de objeto versus fondo), loss_box_reg (ajuste de cajas delimitadoras) y loss_rpn_loc (precisión de localización en propuestas de región). Estas pérdidas convergieron a valores cercanos a cero, como se observa en la Figura B.2 del Anexo B. Sin embargo, las métricas de evaluación, presentadas en la Tabla B.2 del Anexo B, indican un rendimiento insuficiente. Aunque el modelo detectó objetos con buena precisión en umbrales bajos de IoU (AP50), su desempeño disminuyó significativamente en umbrales más estrictos (AP75), sugiriendo menor precisión en la segmentación de bordes. Además, el valor bajo de APs refleja un mal desempeño al segmentar estructuras pequeñas, un aspecto crítico en la base de datos SCIAN. Por estas razones, se decidió no utilizar Mask R-CNN para la segmentación de cabezas de espermatozoides.

En resumen, el modelo U-Net demostró un desempeño sólido y consistente en la segmentación de cabezas de espermatozoides en la base de datos SCIAN. La convergencia estable durante el entrenamiento, junto con las métricas de evaluación obtenidas, confirma su capacidad de generalización y precisión. La alta superposición entre máscaras predichas y reales, combinada con una baja distancia de hausdorff, indica que las segmentaciones son confiables. En contraste, Mask R-CNN no alcanzó un rendimiento comparable, por lo que no se continuó utilizando.

Finalmente, la comparación del modelo U-Net con estado del arte se presenta en la Figura 5.3 y la Tabla 5.2, donde se detallan las métricas de rendimiento frente a otros enfoques.

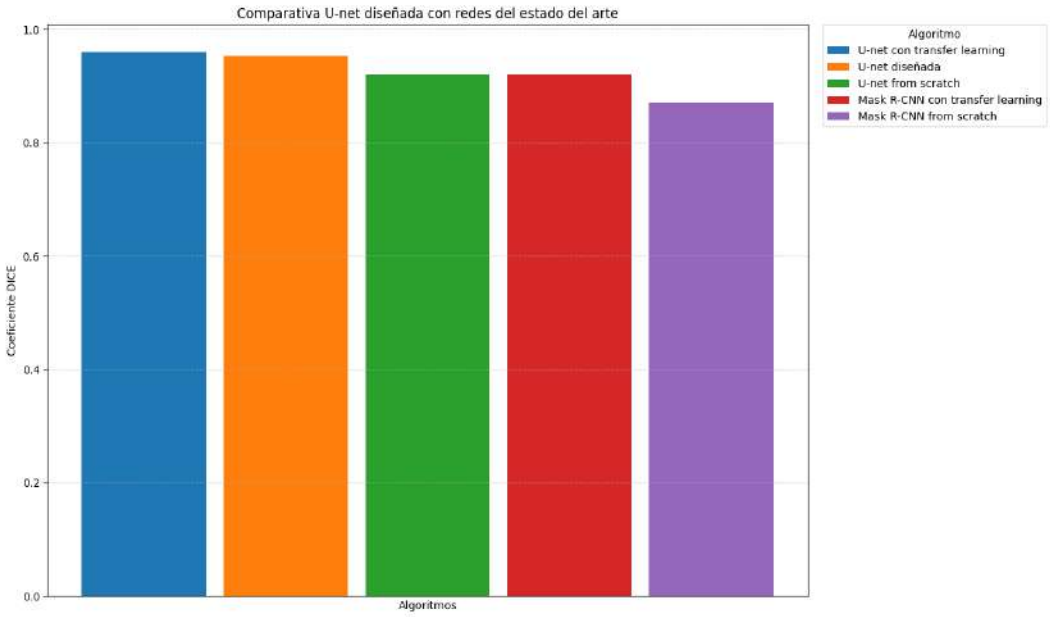


Figura 5.3: Gráfico comparativo de modelo U-Net diseñado con el estado del arte.

Tabla 5.2: Métricas de evaluación del modelo U-Net diseñado y el estado del arte.

Algoritmo de segmentación	Coeficiente DICE	IoU	Distancia Hausdorff
U-net con transfer learning	0,96	-	-
U-net diseñada	0,9530	0,9132	1,9104
U-net from scratch	0,92	-	-
Mask R-CNN con transfer learning	0,92	-	-
Mask R-CNN from scratch	0,87	-	-

5.1.1. Discusión general

En general, los resultados obtenidos con el modelo U-Net son prometedores, alcanzando un coeficiente DICE 0,9530 en el conjunto de evaluación, cercano al reportado en el estado del arte (0,96) para tareas similares. Este rendimiento es significativo, ya que se logró sin emplear *transfer learning* ni pre-entrenamiento, lo que destaca la robustez de la arquitectura diseñada para aprender características relevantes de SCIAN.

La evaluación con métricas complementarias, como IoU y distancia hausdorff, indica que el modelo no solo logra una segmentación precisa en términos de superposición, sino que también mantiene alta fidelidad en la forma y localización de las regiones segmentadas. Sin embargo, el valor de la distancia hausdorff sugiere pequeños errores locales en los contornos, un fenómeno esperado debido a la baja resolución de las imágenes (40×40 píxeles), donde cada píxel afecta aún más.

Como se evidenció en los resultados, los errores se deben en su parte a la máscara real de la base de datos, ya que en la Figura 5.2 ejemplo 1, se puede observar que en la máscara real no se segmentaron por separado los dos espermatozoides presentes y esto puede afectar el

aprendizaje del modelo, bajando la precisión de la segmentación predicha por U-Net.

Por lo anterior, el modelo U-Net propuesto resulta competitivo y eficiente, considerando las limitaciones de la base de datos, como el tamaño reducido de las imágenes. Sin embargo, se identificaron oportunidades de mejora para optimizar su rendimiento:

- Aumentar los datos de entrenamiento: Incorporar más imágenes o aplicar técnicas de aumento de datos para mejorar la generalización del modelo.
- Ajustar función de perdida: Explorar funciones de pérdida alternativas, como HausdorffDTLoss para mejorar la segmentación en bordes.
- Aumentar la complejidad de la red: Evaluar arquitecturas avanzadas como U-Net++ o Attention U-Net, para capturar características más complejas.

Este trabajo también evidencia que, para imágenes de 40×40 píxeles con una sola clase, la segmentación semántica es más adecuada, eficiente y precisa que la segmentación por instancias. Esto se debe a que la segmentación semántica no requiere diferenciar instancias individuales, simplificando la tarea y evitando la complejidad de modelos como Mask R-CNN. Además, la baja resolución de las imágenes dificulta la detección precisa de instancias, mientras que la segmentación semántica aprovecha mejor la información de cada píxel, generando máscaras de mejor calidad.

5.2. Clasificación

En esta sección se presentan los resultados obtenidos al evaluar cuatro modelos de *deep learning* (CNN, ShuffleNetV2, ResNet-18 y red combinada de espacios latentes) con tres funciones de pérdida: BCEWithLogitsLoss, Cross Entropy Loss y Focal Loss. Para esta última, se exploraron combinaciones de los hiperparámetros α (0,5y0,75) y γ (2,0, 3,0y4,0). Las métricas de evaluación incluyeron *Accuracy*, *Precision*, *Recall*, *F1-Score* y *AUC*.

5.2.1. Convolutional Neuronal Network (CNN)

El modelo CNN se evaluó inicialmente con Cross Entropy Loss y BCEWithLogitsLoss, que requieren una base de datos balanceada, por lo que se aplicó submuestreo. En la Figura C.1 en el Anexo C, se observan las curvas de pérdidas de entrenamiento y validación para ambas funciones de pérdida, mostrando una disminución general a lo largo de las épocas, lo que indica aprendizaje. No se observaron signos de sobreajuste, aunque las fluctuaciones en las curvas sugieren cierta inestabilidad en el entrenamiento con estas funciones de pérdida.

Para abordar el desbalance de clases, se empleó Focal Loss, iterando sobre los hiperparámetros α: 0,5 y 0,75 , y γ: 2,0, 3,0 y 4,0. La configuración óptima fue Focal Loss con $\alpha = 0{,}75$ y $\gamma = 2{,}0$, como se detalla en la Figura C.3 y la Tabla C.1 del Anexo C. Las curvas de pérdidas de entrenamiento y validación (Figura C.2, Anexo C) muestran una disminución estable, sin sobreajuste y con menor inestabilidad que con Cross Entropy Loss y BCEWithLogitsLoss.

En la Figura 5.4 y Tabla 5.3 se comparan las métricas de evaluación para CNN con Cross Entropy Loss, BCEWithLogitsLoss y Focal Loss ($\alpha = 0{,}75$ y $\gamma = 2{,}0$). Esta última configuración obtuvo el mejor rendimiento, con valores superiores en todas las métricas. Además, las

pérdidas con Focal Loss oscilan entre 0,065 y 0,01, mientras que con BCEWithLogitsLoss y Cross Entropy Loss varían entre 0,8 y 0,35, indicando que Focal Loss reduce significativamente las pérdidas durante el entrenamiento, mejorando la capacidad del modelo para manejar clases desbalanceadas.

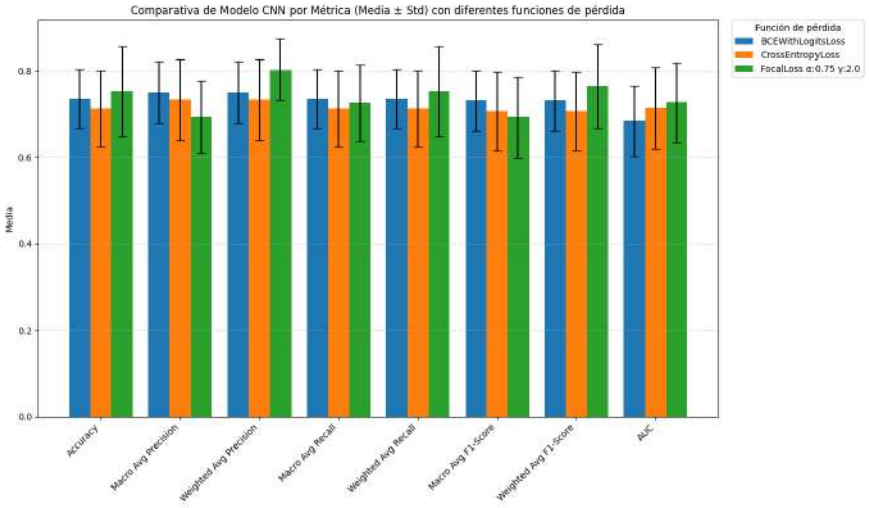


Figura 5.4: Comparación de métricas de evaluación del modelo CNN usando tres diferentes funciones de pérdida.

Tabla 5.3: Tabla de métricas de evaluación para el modelo CNN utilizando tres funciones de pérdida.

Métrica	BCEWithLogitsLoss	CrossEntropyLoss	Focal Loss
Accuracy	$0{,}7357 \pm 0{,}0681$	$0{,}7131 \pm 0{,}0887$	$0{,}7527 \pm 0{,}1045$
Macro Avg Precision	$0{,}7506 \pm 0{,}0724$	$0{,}7333 \pm 0{,}0933$	$0{,}6931 \pm 0{,}0843$
Weighted Avg Precision	$0{,}7506 \pm 0{,}0724$	$0{,}7333 \pm 0{,}0933$	$0{,}8033 \pm 0{,}0714$
Macro Avg Recall	$0{,}7357 \pm 0{,}0681$	$0{,}7131 \pm 0{,}0887$	$0{,}7267 \pm 0{,}0886$
Weighted Avg Recall	$0{,}7357 \pm 0{,}0681$	$0{,}7131 \pm 0{,}0887$	$0{,}7527 \pm 0{,}1045$
Macro Avg F1-Score	$0{,}7317 \pm 0{,}0695$	$0{,}7064 \pm 0{,}0913$	$0{,}6928 \pm 0{,}0935$
Weighted Avg F1-Score	$0{,}7317 \pm 0{,}0695$	$0{,}7064 \pm 0{,}0913$	$0{,}7649 \pm 0{,}0966$
AUC	$0{,}6837 \pm 0{,}0821$	$0{,}7145 \pm 0{,}0949$	$0{,}7272 \pm 0{,}0927$

El uso de submuestreo para balancear la base de datos resultó contraproducente, ya que redujo la cantidad de datos disponibles, afectando el aprendizaje del modelo. En cambio, Focal Loss con $\alpha = 0{,}75$ asigna mayor ponderación a la clase minoritaria (0,75) y menor a la mayoritaria (0,25), mejorando el manejo del desbalance sin sacrificar datos.

5.2.2. ShuffleNetV2

Para ShuffleNetV2, se aplicó submuestreo para trabajar con CrossEntropyLoss y BCEWithLogitsLoss. Las curvas de pérdida de entrenamiento y validación (Figura D.1 Anexo D) muestran una disminución a lo largo de las épocas, indicando aprendizaje, aunque con fluc-

tuaciones que indican inestabilidad y no se observó sobreajuste. Posteriormente, se utilizó Focal Loss, iterando sobre los hiperparámetros α y γ. La configuración óptima fue Focal Loss con $\alpha = 0{,}75$ y $\gamma = 4{,}0$, como se detalla en la Figura D.3 y Tabla D.1 del Anexo D. Las curvas de pérdida (Figura D.2, Anexo D) son más estables que con las otras funciones, aunque se observa un ligero sobreajuste a partir de la época 80.

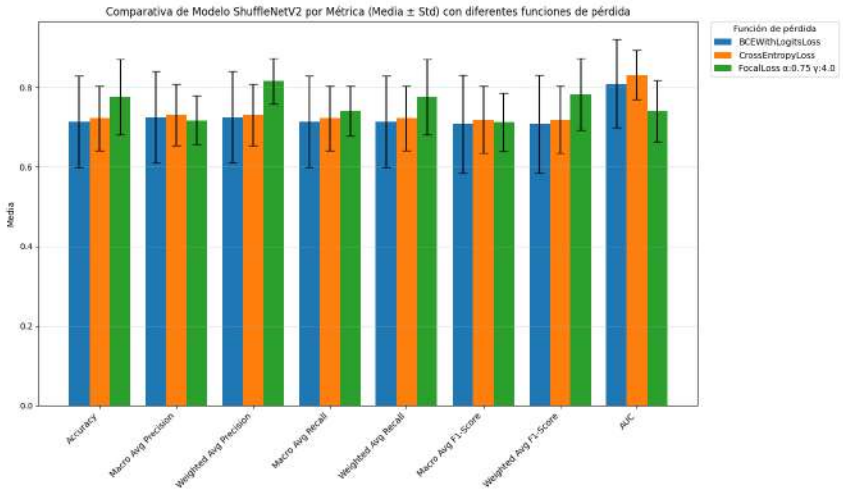


Figura 5.5: Comparación de métricas de evaluación del modelo ShuffleNetV2 usando tres diferentes funciones de pérdida.

En la Figura 5.5 y Tabla 5.4 se comparan las métricas de evaluación, donde Focal Loss ($\alpha = 0{,}75$ y $\gamma = 4{,}0$) obtuvo el mejor rendimiento, con perdidas entre 0,02 y 0,01, frente a 0,75 y 0,55 para Cross Entropy Loss y BCEWithLogitsLoss.

Tabla 5.4: Tabla de métricas de evaluación del modelo ShuffleNetV2 usando tres funciones de pérdida.

Métrica	BCEWithLogitsLoss	CrossEntropyLoss	Focal Loss
Accuracy	$0{,}7142 \pm 0{,}1153$	$0{,}7225 \pm 0{,}0810$	$0{,}7754 \pm 0{,}0943$
Macro Avg Precision	$0{,}7249 \pm 0{,}1149$	$0{,}7314 \pm 0{,}0773$	$0{,}7170 \pm 0{,}0607$
Weighted Avg Precision	$0{,}7249 \pm 0{,}1149$	$0{,}7314 \pm 0{,}0773$	$0{,}8158 \pm 0{,}0575$
Macro Avg Recall	$0{,}7142 \pm 0{,}1153$	$0{,}7225 \pm 0{,}0810$	$0{,}7402 \pm 0{,}0629$
Weighted Avg Recall	$0{,}7142 \pm 0{,}1153$	$0{,}7225 \pm 0{,}0810$	$0{,}7754 \pm 0{,}0943$
Macro Avg F1-Score	$0{,}7075 \pm 0{,}1229$	$0{,}7188 \pm 0{,}0841$	$0{,}7126 \pm 0{,}0724$
Weighted Avg F1-Score	$0{,}7075 \pm 0{,}1229$	$0{,}7188 \pm 0{,}0841$	$0{,}7817 \pm 0{,}0906$
AUC	$0{,}8090 \pm 0{,}1113$	$0{,}8311 \pm 0{,}0625$	$0{,}7405 \pm 0{,}0768$

5.2.3. ResNet-18

ResNet-18 s evaluó con las tres funciones de pérdida. A pesar de un buen rendimiento en métricas de evaluación observadas en Tabla E.1 Anexo E, las curvas de pérdidas de entrenamiento y validación (Figura E.1, Anexo E) muestran un comportamiento fluctuante, indicando inestabilidad. Las perdidas con Cross Entropy Loss y BCEWithLogitsLoss oscilan

entre 2,5 y 0,5, mientras que con Focal Loss ($\alpha = 0{,}75$ y $\gamma = 2{,}0$), varían entre 0,09 y 0,06, pero con mayores fluctuaciones que las otras funciones. Debido a esta inestabilidad y alto rango de pérdidas, se decidió no continuar utilizando esta red para clasificación.

5.2.4. Espacios Latentes

Dado el buen desempeño de Focal Loss en los modelos anteriores, la red de espacios latentes se evaluó únicamente con esta función de pérdida, iterando sobre α (0,5 y 0,75) y γ (2,0, 2,0 y 4,0). La configuración óptima fue Focal Loss con $\alpha = 0{,}75$ y $\gamma = 3{,}0$ (Figura F.1 y Tabla F.1 Anexo F. En la Figura F.1 Anexo F, las curvas de pérdidas de entrenamiento y validación disminuyen, aunque la de validación muestra un sobreajuste a partir de la época 50, donde comienza a aumentar de a poco. A diferencia de otros modelos, esta red presenta fluctuaciones mínimas, indicando mayor estabilidad.

5.2.5. Comparación

En la Figura 5.6 y Tabla 5.5, se presentan las métricas de evaluación de las mejores configuraciones de cada modelo, destacando que ShuffleNetV2 obtiene el mejor rendimiento. Además, se compararon los algoritmos de *machine learning* (Random Forest y XGBoost), confirmando que ShuffleNetV2 supera a ambos.

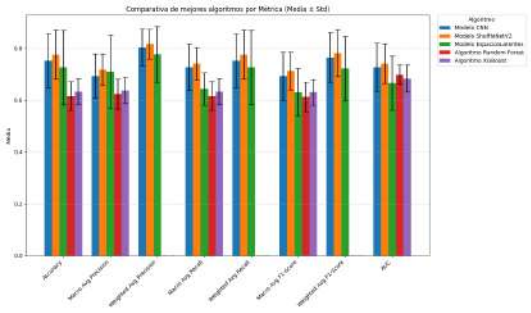


Figura 5.6: Gráfico de métricas de evaluación de los mejores modelos de *deep learning* diseñados y los algoritmos de *machine learning*.

Tabla 5.5: Tabla de métricas de evaluación de los mejores modelos de *deep learning* diseñados y los algoritmos de *machine learning*.

Métrica	ShuffleNetV2	RandomForest	XGBoost	CNN	Espacios Latentes
Accuracy	$0{,}7754 \pm 0{,}0943$	$0{,}6166 \pm 0{,}0559$	$0{,}6332 \pm 0{,}0494$	$0{,}7527 \pm 0{,}1045$	$0{,}6431 \pm 0{,}1806$
MacroAvg Precision	$0{,}7170 \pm 0{,}0607$	$0{,}6232 \pm 0{,}0579$	$0{,}6377 \pm 0{,}0505$	$0{,}6931 \pm 0{,}0843$	$0{,}6230 \pm 0{,}1687$
WeightedAvg Precision	$0{,}8158 \pm 0{,}0575$	-	-	$0{,}8033 \pm 0{,}0714$	$0{,}7364 \pm 0{,}2042$
MacroAvg Recall	$0{,}7402 \pm 0{,}0629$	$0{,}6166 \pm 0{,}0559$	$0{,}6332 \pm 0{,}0494$	$0{,}7267 \pm 0{,}0886$	$0{,}6608 \pm 0{,}0825$
WeightedAvg Recall	$0{,}7754 \pm 0{,}0943$	-	-	$0{,}7527 \pm 0{,}1045$	$0{,}6431 \pm 0{,}1806$
Macro Avg F1-Score	$0{,}7126 \pm 0{,}0724$	$0{,}6123 \pm 0{,}0570$	$0{,}6303 \pm 0{,}0496$	$0{,}6928 \pm 0{,}0935$	$0{,}5810 \pm 0{,}1571$
Weighted Avg F1-Score	$0{,}7817 \pm 0{,}0906$	-	-	$0{,}7649 \pm 0{,}0966$	$0{,}6362 \pm 0{,}2070$
AUC	$0{,}7405 \pm 0{,}0768$	$0{,}6980 \pm 0{,}0372$	$0{,}6829 \pm 0{,}0526$	$0{,}7272 \pm 0{,}0927$	$0{,}7317 \pm 0{,}0663$

La comparación con el estado del arte se presenta en la Figura 5.7 y la Tabla 5.6, donde ShuffleNetV2 muestra un rendimiento competitivo, aunque inferior a modelos SHMC-Net y Meta-Classifier, diseñados para conjuntos más grandes.

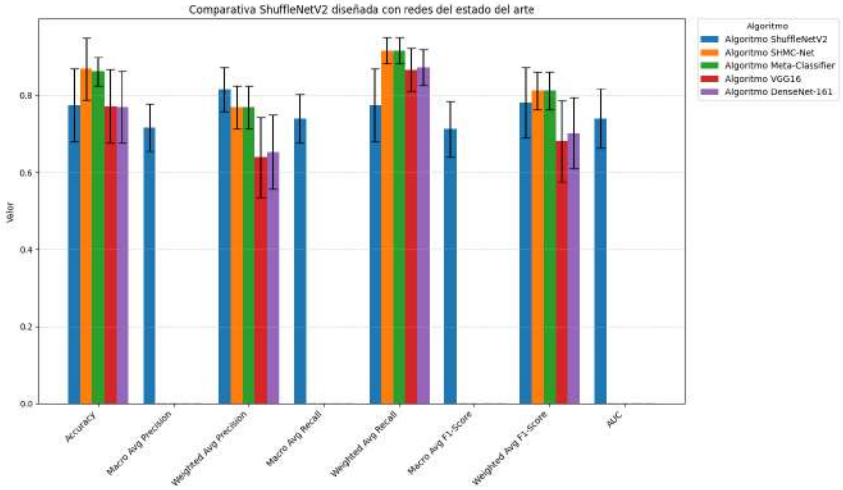


Figura 5.7: Gráfico de comparación de métricas de evaluación entre el mejor modelo de clasificación diseñado y el estado del arte.

Tabla 5.6: Tabla de métricas de evaluación del mejor modelo de clasificación diseñado y el estado del arte.

Métrica	ShuffleNetV2	SHMC-Net total acuerdo	Meta-Classifier	VGG16	DenseNet-161
Accuray	$0{,}7754 \pm 0{,}0943$	$0{,}869 \pm 0{,}08$	$0{,}8629 \pm 0{,}0368$	$0{,}7726 \pm 0{,}0947$	$0{,}7701 \pm 0{,}0927$
MacroAvg Precision	$0{,}7170 \pm 0{,}0607$	-	-	-	-
WeightedAvg Precision	$0{,}8158 \pm 0{,}0575$	$0{,}867 \pm 0{,}14$	$0{,}7695 \pm 0{,}0565$	$0{,}6401 \pm 0{,}1049$	$0{,}6537 \pm 0{,}0968$
MacroAvg Recall	$0{,}7402 \pm 0{,}0629$	-	-	-	-
WeightedAvg Recall	$0{,}7754 \pm 0{,}0943$	$0{,}834 \pm 0{,}16$	$0{,}9177 \pm 0{,}0343$	$0{,}8671 \pm 0{,}0568$	$0{,}8736 \pm 0{,}0464$
MacroAvg F1-Score	$0{,}7126 \pm 0{,}0724$	-	-	-	-
WeightedAvg F1-Score	$0{,}7817 \pm 0{,}0906$	$0{,}85 \pm 0{,}15$	$0{,}8124 \pm 0{,}0485$	$0{,}6815 \pm 0{,}1055$	$0{,}7026 \pm 0{,}0922$
AUC	$0{,}7405 \pm 0{,}0768$	-	-	-	-

Por último, se tiene la Tabla 5.7 que indica la cantidad de parámetros utilizados por cada algoritmo diseñado.

Tabla 5.7: Parámetros utilizados en cada algoritmo diseñado.

Modelo	Parámetros
CNN diseñada	$485,445$
ShuffleNetV2	$380,033$
ResNet-18	$2,795,297$
Espacios Latentes	$35,361$

5.2.6. Discusión general

Se compararon diversos algoritmos para la clasificación de cabezas de espermatozoides, incluyendo una CNN diseñada, ResNet-18, ShuffleNetV2, una red basada en espacios latentes, RandomForest y XGBoost. Los resultados muestran que ShuffleNetV2 supera en rendimiento a las otras arquitecturas de *deep learning* y a los algoritmos de *machine learning* evaluados.

El desempeño superior de ShuffleNetV2 se atribuye a su diseño eficiente, detallado a continuación:

- CNN diseñada: Incorpora un bloque SE que recalibra adaptativamente los canales, mejorando la selección de características relevantes, y conexiones residuales que estabilizan el aprendizaje. Sin embargo, su rendimiento es inferior a ShuffleNetV2 debido a la presencia de varias capas densas al final de la arquitectura, lo que aumenta la complejidad y reduce la eficiencia.

- Red de espacios latentes: Esta arquitectura, diseñada para procesar vectores de entrada preprocesados como las características extraídas por el modelo shufflenetV2 y las extraídas a partir de las máscaras, es más simple y cuenta con menos parámetros. Aunque esta simplicidad reduce la carga computacional, limita la capacidad de capturar patrones complejos en comparación con las otras redes, resultando un rendimiento inferior.

- ResNet-18: Esta arquitectura de 18 capas, es más compleja y está diseñada para recibir un conjunto de datos grandes, como por ejemplo ImageNet, que contiene millones de imágenes. Sin embargo, en el conjunto de datos SCIAN, el cual tiene aproximadamente dos mil imágenes, ResNet18 es propensa al sobreajuste, generando fluctuaciones durante su entrenamiento. Además, su mayor número de parámetros la hace más lenta y menos eficiente que ShuffleNetV2 para clasificar.

- ShuffleNetV2: El rendimiento destacado por parte de esta red se debe a su diseño ligero y eficiente. La operación de channel shuffle mejora la diversidad de características al mezclar información entre grupos de canales, favoreciendo la generalización sin incrementar la complejidad computacional. Así mismo, el uso de convoluciones separables por profundidad (depth_wise) optimiza la eficiencia, haciendo que esta arquitectura sea adecuada para conjuntos de datos pequeños como SCIAN.

Aunque ShuffleNetV2 mostró un rendimiento sobresaliente con SCIAN, su aplicabilidad a otras bases de datos internacionales, como HuSHeM y SMIDs, está limitada por las diferencias en las tinciones de las imágenes, que requieren preprocesamiento adicional o ajustes en la arquitectura para mejorar la generalización.

Respecto a la función de perdida, Focal Loss resultó la más adecuada para mitigar el desbalance de clases presente en SCIAN, sin necesidad de recurrir a submuestreo, que reduce la cantidad de datos disponibles. Un valor de $\alpha = 0{,}75$ otorga mayor peso a la clase minoritaria, mejorando la clasificación sin desequilibrar la clase mayoritaria (a diferencia de $\alpha = 0{,}25$) ni tratar ambas clases por igual ($\alpha = 0{,}5$). Un $\gamma = 4{,}0$ en ShuffleNetV2 enfoca el entrenamiento en ejemplos difíciles, aumentando la precisión. En contraste, Cross Entropy Loss y BCEWithLogitsLoss generan pérdidas de validación más altas, ya que ponderan de manera uniforme los errores sin diferenciar su dificultad, mientras que Focal Loss reduce el peso de ejemplos fáciles mediante α y γ, siendo ideal para conjuntos desbalanceados.

En comparación con el estado del arte, ShuffleNetV2 logra un rendimiento competitivo en la clasificación morfológica de cabezas de espermatozoides en la base de datos SCIAN, aunque inferior al de modelos como SHMC-Net y Meta-Classifier. Estos últimos se benefician de técnicas avanzadas, como el pre-entrenamiento en ImageNet y un robusto aumento de datos (redimensiones a 64×64, recortes aleatorios múltiples, desplazamientos, entre otros).

Además, SHMC-Net integra la fusión con otra ShuffleNetV2 que utiliza máscaras de segmentación para resaltar características morfológicas. La menor precisión de ShuffleNetV2 se atribuye principalmente a la falta de pre-entrenamiento, aumento de datos, fusión con otra red que procese las máscaras y el ensamblaje de redes de *deep learning*.

Aun así, ShuffleNetV2 alcanza un 77 % de Accuracy sin emplear estas técnicas, lo que lo posiciona como un punto de partida prometedor. La incorporación de aumento de datos, una red para clasificar las máscaras segmentadas en paralelo para una posterior fusión, podrían permitir a ShuffleNetV2 alcanzar un rendimiento comparable al del estado del arte, manteniendo su ventaja en eficiencia computacional.

Por último, la evaluación morfológica de espermatozoides requiere de un alto nivel de *Accuracy* para otorgar un diagnostico confiable. Por ello, aunque el rendimiento de ShuffleNetV2 es de un 77 % de *Accuracy*, no es suficiente para una integración a nivel clínico, pero si como una herramienta de apoyo en laboratorios a modo de complemento a la evaluación manual.

Capítulo 6

Conclusiones

A partir de este trabajo realizado y los resultados obtenidos, se destaca el desarrollo de algoritmos de *deep learning* para la segmentación y clasificación de cabezas de espermatozoides en imágenes de microscopia óptica de campo claro teñidas con hematoxilina-eosina que han permitido alcanzar los objetivos planteados.

El modelo de segmentación basado en U-Net demostró un desempeño sobresaliente, con un coeficiente DICE de 0,9530, IoU de 0,9132 y una distancia Hausdorff de 1,9104 en el conjunto de evaluación, así como resultados comparables en la validación con la base de datos internacional SCIAN (Coeficiente DICE 0,9471, IoU de 0,903 y distancia Hausdorff 2,3605). Estos valores reflejan la robustez del modelo para detectar y segmentar con precisión las cabezas de espermatozoides, lo que lo posiciona como una herramienta confiable para aplicaciones en análisis morfológico automatizado.

Por otra parte, el modelo de clasificación basado en ShuffleNetV2 alcanzó una Accuracy del 77 % en validación cruzada tipo LOO, un rendimiento aceptable pero limitado para aplicaciones clínicas que requieren un estándar de Accuracy superior. Este resultado se atribuye principalmente al desbalance de clases (1 : 3) y al reducido número de muestras por clase en la base de datos SCIAN, lo que afectó la capacidad del modelo para generalizar. Sin embargo, el modelo puede servir como herramienta de asistencia para expertos en la evaluación morfológica, optimizando el proceso de clasificación.

La validación y evaluación de los modelos en la base de datos SCIAN permitió confirmar su eficacia, aunque no fue posible realizarlo con otras bases internacionales como HuSHeM y SMIDs debido a diferencias en los métodos de tinción.

La comparación con algoritmos del estado del arte, el modelo de segmentación mostró un rendimiento competitivo, con métricas cercanas a las reportadas en la literatura. Por otro lado, el modelo de clasificación, requiere mejoras en su capacidad de generalización para alcanzar los rendimientos de los algoritmos del estado del arte.

En resumen, este trabajo logró desarrollar, validar y evaluar algoritmos de *deep learning* que contribuyen al análisis automatizado de la morfología espermática, abriendo camino para futuras mejoras, como la optimización de los modelos considerando un aumento de datos, el manejo del desbalance de clases y la adaptación a diferentes técnicas de tinción. Estos

avances contribuyen hacia la automatización y estandarización en el diagnostico de fertilidad masculina.

6.1. Trabajo futuro

6.1.1. Segmentación

Para optimizar la segmentación de cabezas de espermatozoides en el conjunto de datos SCIAN, se propone lo siguiente:

- Implementar la función de pérdida HausdorffDTLoss, disponible en la librería MONAI (versión 1.4.0), basada en el artículo Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks [44]. Esta función minimizaría la distancia de hausdorff en la arquitectura U-net propuesta, mejorando la precisión de la segmentación.

- Implementar arquitectura Attention U-Net, que incorpora mecanismos de atención para destacar regiones relevantes de la imagen, permitiendo a la red aprender características más detalladas y mejorar la calidad de las segmentaciones.

6.1.2. Clasificación

Para mejorar la clasificación de cabezas de espermatozoides en el conjunto SCIAN, se propone lo siguiente:

- Explorar la incorporación de bloques de atención como SE, integrado en la arquitectura CNN diseñada en este trabajo, o el bloque Convolutional Block Attention Module (CBAM), que combina atención en canales y características espaciales, según artículo [45].

- Implementar una red que fusione las características extraídas de las imágenes a través de ShuffleNetV2 con las características obtenidas de las máscaras a través de otra ShuffleNetV2, simulando el algoritmo SHMC del estado del arte.

- Hacer más robusto el algoritmo de espacios latentes o concatenar los espacios latentes de dos redes *deep learning* en vez de una de *deep learning* con características morfológicas.

- Ampliar la base de datos para equilibrar las clases, incrementando la cantidad de datos disponibles para entrenar el modelo.

- Para abordar la validación y evaluación en bases de datos internacionales con diferentes tinciones, se propone entrenar los modelos simultáneamente con múltiples conjuntos de datos mediante un enfoque multi-task learning [46]. Esto permitirá al modelo aprender características específicas de cada tinción, seguido de un ajuste fino (*fine-tuning*) específico para cada conjunto, adaptando los modelos a las especificaciones de cada base de datos y aumentando su robustez ante variaciones en las imágenes.

Bibliografía

[1] Cui, W., "Mother of nothing: the agony of infertility", Bull World Health Org, no. 12, pp. 881–882, 2010, doi:10.2471/BLT.10.011210.

[2] World Health Organization, WHO Laboratory Manual for the Examination and Processing of Human Semen. Geneva: World Health Organization, 6th ed ed., 2010.

[3] Organización Mundial de la Salud, Manual de laboratorio de la OMS para el examen del semen humano y de la interacción entre el semen y el moco cervical. Organización mundial de la salud, 4ta ed ed., 2001.

[4] LeCun, Y., Bengio, Y., y Hinton, G., "Deep learning", Nature, 2015, doi:10.1038/nature14539.

[5] Terven, J., Cordova-Espar, y otros, "Loss functions and metrics in deep learning", Buscando, 2023, doi:10.48550/arXiv.2307.02694.

[6] PyTorch, "torch.nn.bcewithlogitsloss"., https://docs.pytorch.org/stable/generated/torch.nn.BCEWithLogitsLoss.html.

[7] Chen, Y., Li, L., Li, W., Guo, Q., Du, Z., y Xu, Z., Fundamentals of Neural Networks, cap. 2. Elsevier, 2024, doi:10.1016/B978-0-32-395399-3.00008-1.

[8] Chen, F., "Experiments on multi-task learning framework over bert for performing sentiment analysis, paraphrase detection, and semantic textual similarity simultaneously", default project, Department of Computer Science, Stanford University, 2023. Stanford CS224N.

[9] Mao, A., Mohri, M., y Zhong, Y., "Cross-entropy loss functions: Theoretical analysis and applications", 2023, https://arxiv.org/abs/2304.07288.

[10] James, G., Witten, D., Hastie, T., y Tibshirani, R., An Introduction to Statistical Learning: with Applications in R. Springer Texts in Statistics, Springer, 2013.

[11] Bouke, M. A. y Abdullah, A., "An empirical study of pattern leakage impact during data preprocessing on machine learning-based intrusion detection models reliability", Expert Systems with Applications, 2023, doi:10.1016/j.eswa.2023.120715.

[12] Breiman, L., "Random forests", Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[13] Chen, T. y Guestrin, C., "XGBoost: A scalable tree boosting system", arXiv preprint arXiv:1603.02754, June 2016, https://arxiv.org/abs/1603.02754. Presented at KDD'16, August 13-17, 2016, San Francisco, CA, USA.

[14] Ronneberger, O., Fischer, P., y Brox, T., "U-net: Convolutional networks for biomedical image segmentation", arXiv preprint arXiv:1505.04597, 2015, doi:10.48550/arXiv.1505.04597.

[15] He, K., Gkioxari, G., Dollár, P., y Girshick, R., "Mask r-cnn", arXiv preprint arXiv:1703.06870, 2018, doi:10.48550/arXiv.1703.06870.

[16] Meta AI, "Detectron2: A pytorch-based modular object detection library". https://ai.meta.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/, 2019.

[17] O'Shea, K. y Nash, R., "An introduction to convolutional neural networks", arXiv preprint arXiv:1511.08458, 2015, doi:10.48550/arXiv.1511.08458.

[18] Simonyan, K. y Zisserman, A., "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556, 2015, doi:10.48550/arXiv.1409.1556.

[19] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhn, D., Vanhoucke, V., y Rabinovich, A., "Going deeper with convolutions", arXiv preprint arXiv:1409.4842, 2014, doi:10.48550/arXiv.1409.4842.

[20] He, K., Zhang, X., Ren, S., y Sun, J., "Deep residual learning for image recognition", arXiv preprint arXiv:1512.03385, 2015, doi:10.48550/arXiv.1512.03385.

[21] Ma, N., Zhang, X., Zheng, H.-T., y Sun, J., "Shufflenet v2: Practical guidelines for efficient cnn architecture design", arXiv preprint arXiv:1807.11164, 2018, doi:10.48550/arXiv.1807.11164.

[22] Huang, G., Liu, Z., Maaten, L., y Weinberger, K., "Densely connected convolutional networks", arXiv preprint arXiv:1608.06993, 2018, doi:10.48550/arXiv.1608.06993.

[23] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., y Adam, H., "Mobilenets: Efficient convolutional neural networks for mobile vision applications", arXiv preprint arXiv:1704.04861, 2017, doi:10.48550/arXiv.1704.04861.

[24] Peyron, M., Fillion, A., Gürol, S., Marchais, V., Gratton, S., Boudier, P., y Goret, G., "Latent space data assimilation by using deep learning", Quarterly Journal of the Royal Meteorological Society, 2021, doi:10.1002/qj.4153. First published: 1 September 2021.

[25] Hu, J., Shen, L., Albanie, S., Sun, G., y Wu, E., "Squeeze-and-excitation networks", 2019, https://arxiv.org/abs/1709.01507.

[26] Nixon, M. y Aguado, A., Feature Extraction and Image Processing. Academic Press, 2 ed., 2008.

[27] Daugman, J. G., "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, pp. 1169–1179, July 1988. Invited Paper.

[28] Brynolfsson, P., Nilsson, D., Torheim, T., *et al.*, "Haralick texture features from apparent diffusion coefficient (ADC) MRI images depend on imaging and pre-processing parameters", Scientific Reports, vol. 7, p. 4041, 2017, doi:10.1038/s41598-017-04151-4.

[29] Ojala, T., Pietikainen, M., y Maenpaa, T., "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971–987, 2002, doi:10.1109/TPAMI.2002.1017623.

[30] Karimi, D. y Salcudean, S. E., "Reducing the hausdorff distance in medical image segmentation with convolutional neural networks", 2019, https://arxiv.org/abs/1904.10030.

[31] Chang, V., Garcia, A., Hitschfeld, N., y Härtel, S., "Gold-standard for computer-assisted morphological sperm analysis", Computers in Biology and Medicine, vol. 83, pp. 143–150,

2017, doi:10.1016/j.compbiomed.2017.03.004.

[32] Shaker, F., "Human Sperm Head Morphology dataset (HuSHeM)". Mendeley Data, V3, 2018, doi:10.17632/tt3yj2pf38.3.

[33] Ilhan, H., "Sperm Morphology Image Data Set (SMIDS)". Mendeley Data, V1, 2022, doi:10.17632/6xvdhc9fyb.1.

[34] Ilhan, H. O., Sigirci, I. O., Serbes, G., y Aydin, N., "A fully automated hybrid human sperm detection and classification system based on mobile-net and the performance comparison with conventional methods", Medical Biological Engineering Computing, vol. 58, pp. 1047–1062, December 2020, doi:10.1007/s11517-019-02101-y. Received: 24 May 2019 / Accepted: 16 December 2019.

[35] Ilhan, H. y Serbes, G., "Sperm morphology analysis by using the fusion of two-stage fine-tuned deep networks", Biomedical Signal Processing and Control, 2022, doi:10.1016/j.bspc.2021.103246.

[36] Riordon, J., McCallum, C., y Sinton, D., "Deep learning for the classification of human sperm", Computers in Biology and Medicine, 2019, doi:10.1016/j.compbiomed.2019.103342.

[37] Iqbal, I. y Ma, J., "Deep learning-based morphological classification of human sperm heads", Diagnostics, 2020, doi:10.3390/diagnostics10050325.

[38] Spencer, L., Fernando, J., Akbaridoust, F., Ackermann, K., y Nosrati, R., "Ensembled deep learning for the classification of human sperm head morphology", Advanced Intelligent Systems, 2022, doi:10.1002/aisy.202200111.

[39] Yuzkat, M., Ilhan, H., y Aydin, N., "Multi-model cnn fusion for sperm morphology analysis", Computers in Biology and Medicine, 2021, doi:10.1016/j.compbiomed.2021.104790.

[40] Tortumlu, O. y Ilhan, H., "The analysis of mobile platform based cnn networks in the classification of sperm morphology", en 2020 Medical Technologies Congress (TIPTEKNO), IEEE, 2020, doi:10.1109/TIPTEKNO50054.2020.9299281.

[41] Zhang, Y., Zhang, J., Zha, X., Zhou, Y., Cao, Y., y Chen, D., "Improving human sperm head morphology classification with unsupervised anatomical feature distillation", en 2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI), 2022, doi:10.1109/ISBI52829.2022.9761633.

[42] Sapkota, N., Zhang, Y., Li, S., Liang, P., y otros., "Shmc-net: A mask-guided feature fusion network for sperm head morphology classification", en 2024 IEEE International Symposium on Biomedical Imaging (ISBI), 2024, doi:10.1109/ISBI56570.2024.10635339.

[43] Marin, R. y Chang, V., "Impact of transfer learning for human sperm segmentation using deep learning", Computers in Biology and Medicine, 2021, doi:10.1016/j.compbiomed.2021.104687.

[44] Karimi, D. y Salcudean, S. E., "Reducing the hausdorff distance in medical image segmentation with convolutional neural networks", 2019, https://arxiv.org/abs/1904.10030.

[45] Woo, S., P. J. L. J. K. I., "Cbam: Convolutional block attention module", Computer Vision - ECCV 2018, 2018, doi:https://doi.org/10.1007/978-3-030-01234-2_1.

[46] Crawshaw, M., "Multi-task learning with deep neural networks: A survey", 2020, https:

//arxiv.org/abs/2009.09796.

Anexos

Anexo A. Conjuntos de datos

Tabla A.1: División de conjuntos para el algoritmo de segmetnación.

Conjunto de datos	Paciente	Cantidad imágenes	Total imágenes
Entrenamiento	001	841	1843
	004	482	
	005	301	
	006	96	
	007	123	
Validación	003	250	437
	010	187	
Prueba	008	201	385
	009	67	
	011	117	
Total			2665

Tabla A.2: Cantidad de clases 0, 1 y 2 por paciente.

Paciente	Cantidad Clase 0	Cantidad Clase 1	Cantidad Clase 2
001	128	568	145
003	44	176	30
004	78	346	58
005	111	135	55
006	23	67	6
007	35	88	0
008	59	141	1
009	13	54	0
010	40	147	0
011	32	85	0
Total	563	1807	295

Tabla A.3: Cantidad de clases 0 y 1 por paciente, después de limpiar el *dataframe*.

Paciente	Cantidad Clase 0	Cantidad Clase 1
001	122	546
003	37	173
004	70	339
005	96	131
006	21	66
007	31	88
008	55	138
009	11	52
010	37	145
011	28	85
Total	508	1763

Anexo B. Segmentación

Tabla B.1: Métricas de evaluación modelo U-Net utilizando el 60 % de la base de datos.

Conjunto	Coeficiente DICE	IoU	Distancia Hausdorff
Validación	$0{,}9445 \pm 0{,}0607$	$0{,}9001 \pm 0{,}0824$	$2{,}4477 \pm 3{,}1364$
Evaluación	$0{,}9518 \pm 0{,}0430$	$0{,}9105 \pm 0{,}0616$	$1{,}9370 \pm 2{,}4770$

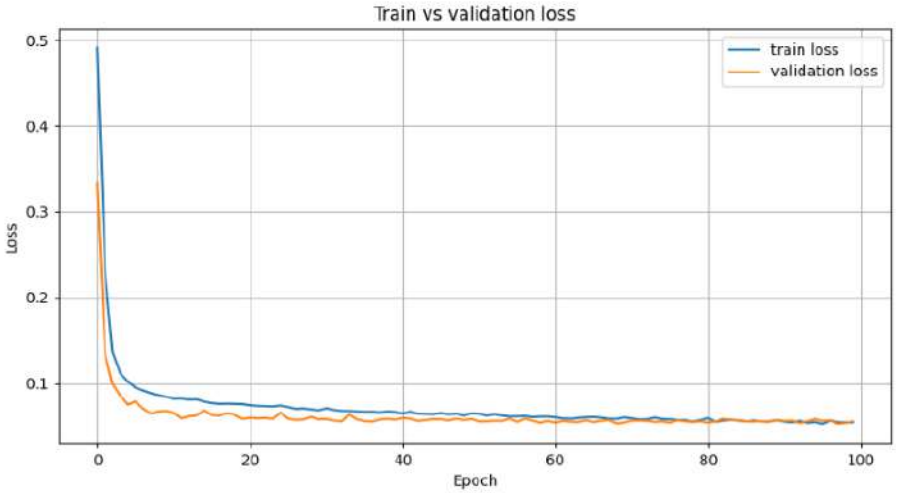
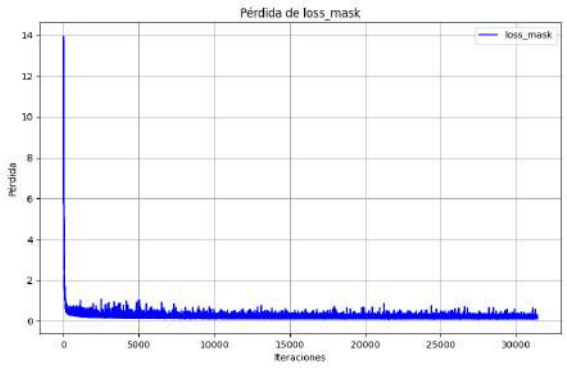
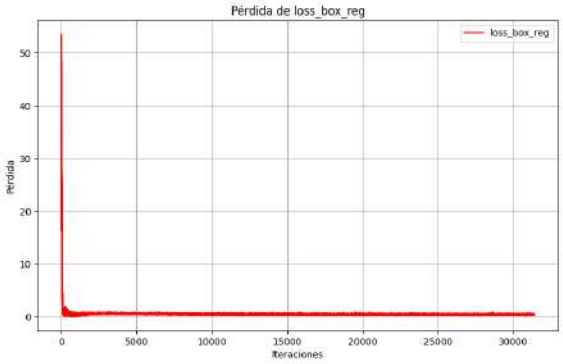


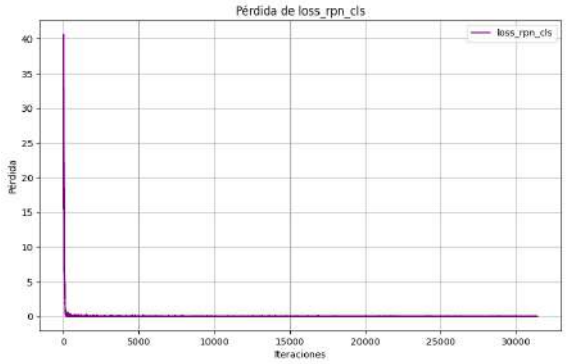
Figura B.1: Gráfico de pérdidas en entrenamiento y validación modelo U-Net entrenada con el 60 % de la base de datos.



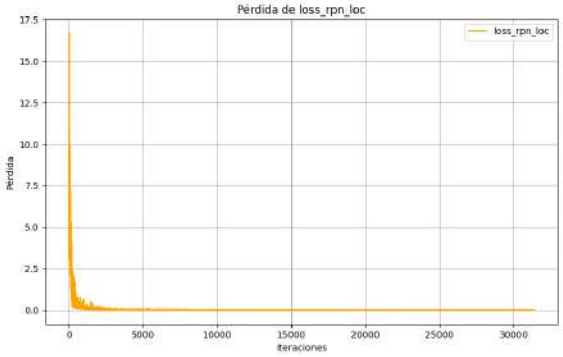
(a) Pérdidas en la precisión de la máscara segmentada.



(b) Pérdidas de ajuste de cajas delimitadoras.



(c) Pérdidas en clasificación de regiones de objeto versus fondo.



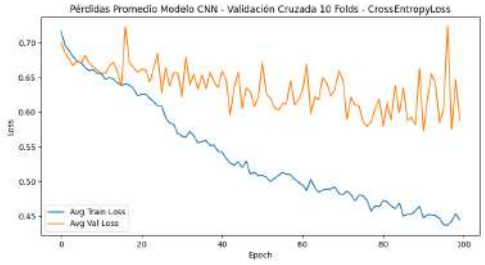
(d) Pérdidas en precisión de localización en propuestas de región.

Figura B.2: Gráficos de pérdida en entrenamiento de sus componentes.

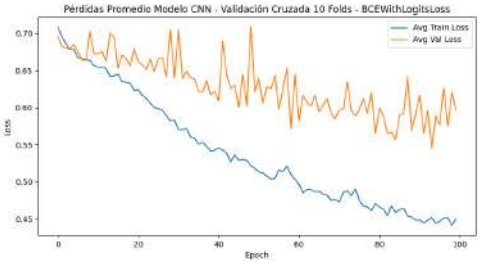
Tabla B.2: Métricas de evaluación modelo Mask R-CNN

AP	AP50	AP75	APs
0,6464	0,9576	0,7961	0,6464

Anexo C. CNN



(a) Función de pérdida Cross Entropy Loss.



(b) Función de pérdida BCEWithLogitsLoss.

Figura C.1: Gráfico de pérdidas de entrenamiento y validación del modelo CNN utilizando función de pérdida CrossEntropyLoss y BCEWithLogitsLoss.

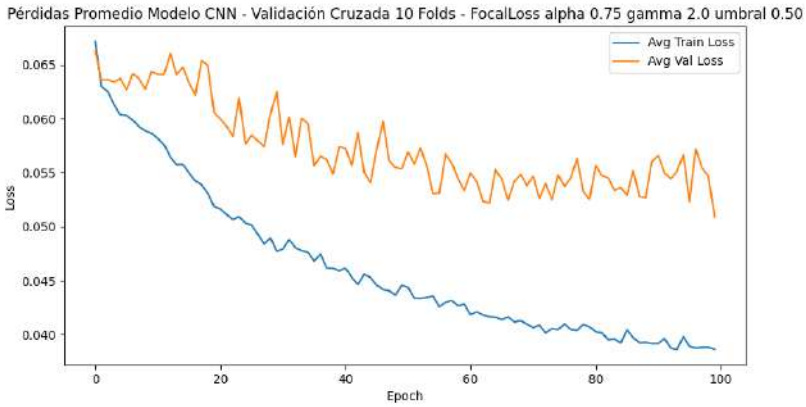


Figura C.2: Gráfico de pérdidas en entrenamiento y validación del mejor rendimiento utilizando Focal Loss en el modelo CNN.

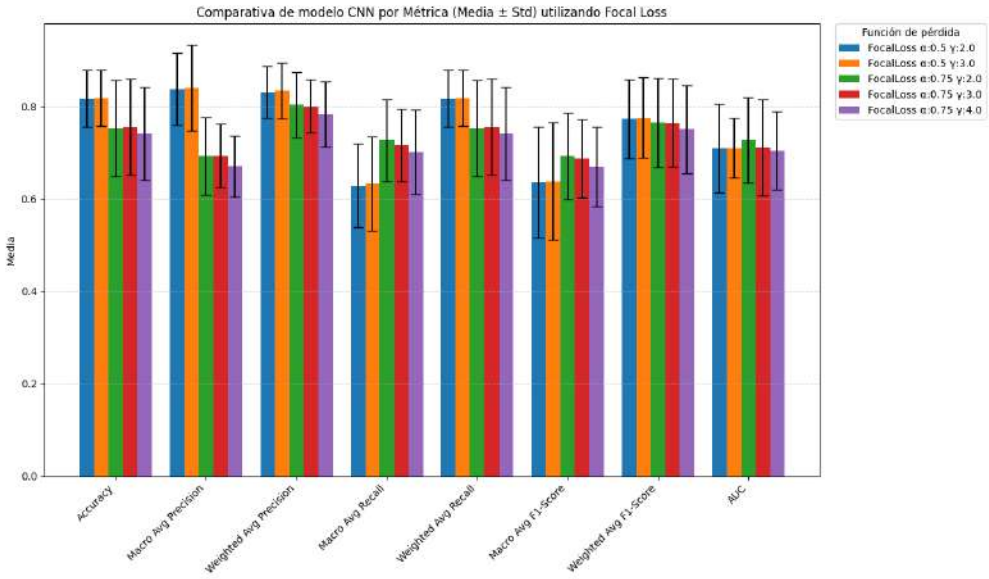
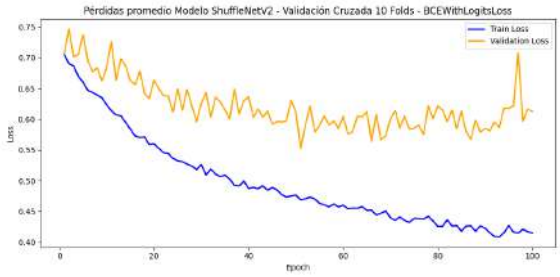


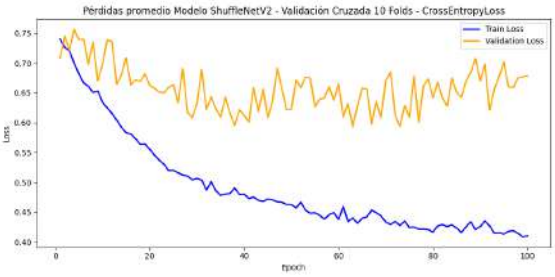
Figura C.3: Gráfico de comparación de métricas de evaluación entre iteraciones sobre hiperparametros para la funcion de perdida Focal Loss en el modelo CNN.

Tabla C.1: Métricas de evalución para iteraciones sobre hiperparametros en la función de pérdida Focal Loss en el modelo CNN.

Métrica	α=0,5 γ=2,0	α=0,5 γ=3,0	α=0,75 γ=2,0	α=0,75 γ=3,0	α=0,75 γ=4,0
Accuracy	$0{,}8172 \pm 0{,}0619$	$0{,}8189 \pm 0{,}0608$	$0{,}7527 \pm 0{,}1045$	$0{,}7563 \pm 0{,}1041$	$0{,}7417 \pm 0{,}1007$
Macro Avg Precision	$0{,}8376 \pm 0{,}0788$	$0{,}8405 \pm 0{,}0929$	$0{,}6931 \pm 0{,}0843$	$0{,}6935 \pm 0{,}0692$	$0{,}6703 \pm 0{,}0664$
Weighted Avg Precision	$0{,}8312 \pm 0{,}0555$	$0{,}8344 \pm 0{,}0614$	$0{,}8033 \pm 0{,}0714$	$0{,}8000 \pm 0{,}0576$	$0{,}7835 \pm 0{,}0703$
Macro Avg Recall	$0{,}6286 \pm 0{,}0908$	$0{,}6333 \pm 0{,}1024$	$0{,}7267 \pm 0{,}0886$	$0{,}7160 \pm 0{,}0779$	$0{,}7016 \pm 0{,}0914$
Weighted Avg Recall	$0{,}8172 \pm 0{,}0619$	$0{,}8189 \pm 0{,}0608$	$0{,}7527 \pm 0{,}1045$	$0{,}7563 \pm 0{,}1041$	$0{,}7417 \pm 0{,}1007$
Macro Avg F1-Score	$0{,}6362 \pm 0{,}1202$	$0{,}6381 \pm 0{,}1276$	$0{,}6928 \pm 0{,}0935$	$0{,}6875 \pm 0{,}0849$	$0{,}6689 \pm 0{,}0863$
Weighted Avg F1-Score	$0{,}7732 \pm 0{,}0854$	$0{,}7755 \pm 0{,}0872$	$0{,}7649 \pm 0{,}0966$	$0{,}7646 \pm 0{,}0947$	$0{,}7503 \pm 0{,}0948$
AUC	$0{,}7094 \pm 0{,}0953$	$0{,}7098 \pm 0{,}0648$	$0{,}7272 \pm 0{,}0927$	$0{,}7117 \pm 0{,}1040$	$0{,}7043 \pm 0{,}0844$



(a) Función de pérdida BCEWithLogitsLoss.



(b) Función de perdida Cross Entropy Loss.

Figura D.1: Gráfico de pérdidas de entrenamiento y validación del modelo ShuffleNetV2 utilizando función de pérdida CrossEntropyLoss y BCEWithLogitsLoss.

Anexo D. ShuffleNetV2

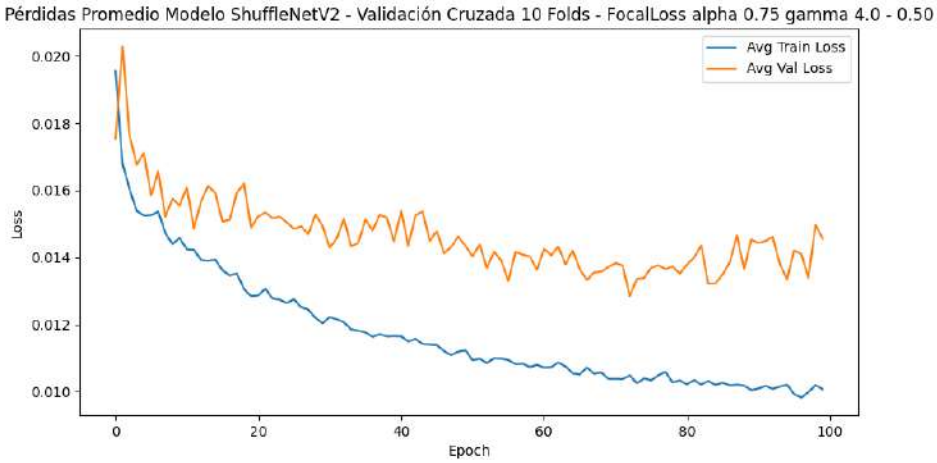


Figura D.2: Gráfico de pérdidas en entrenamiento y validación en el mejor rendimiento utilizando Focal Loss en el modelo ShuffleNetV2.

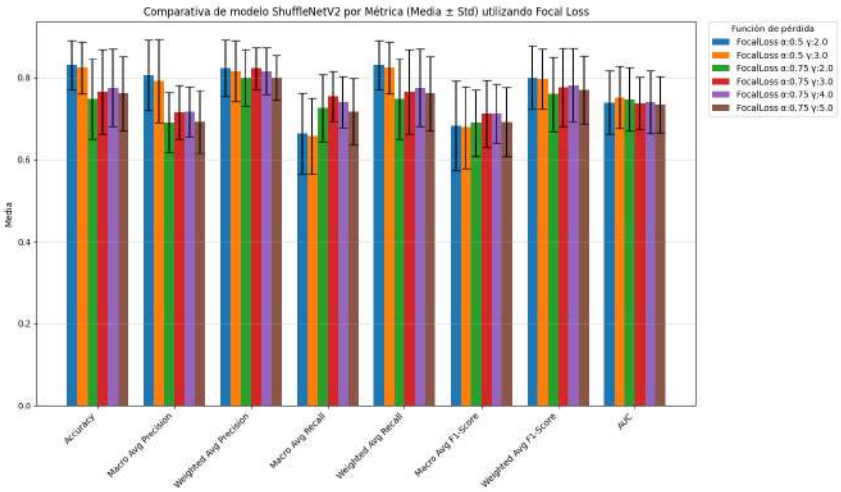
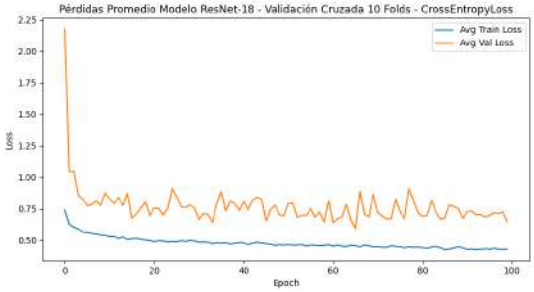


Figura D.3: Gráfico de comparación de métricas de evaluación entre iteraciones sobre hiperparametros para la funcion de perdida Focal Loss en el modelo ShuffleNetV2

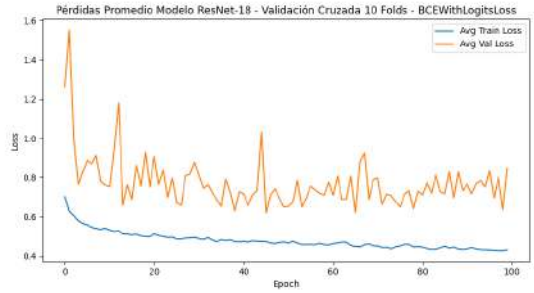
Tabla D.1: Métricas de evaluación para iteraciones sobre hiperparametros en la función de pérdida Focal Loss en el modelo ShuffleNetV2

Métrica	α=0,5 γ=2,0	α=0,5 γ=3,0	α=0,75 γ=2,0	α=0,75 γ=3,0	α=0,75 γ=4,0
Accuracy	0,8302 ± 0,0600	0,8237 ± 0,0635	0,7477 ± 0,0981	0,7651 ± 0,1033	0,7754 ± 0,0943
Macro Avg Precision	0,8056 ± 0,0855	0,7915 ± 0,1016	0,6903 ± 0,0734	0,7159 ± 0,0655	0,7170 ± 0,0607
Weighted Avg Precision	0,8233 ± 0,0689	0,8157 ± 0,0743	0,8004 ± 0,0688	0,8226 ± 0,0513	0,8158 ± 0,0575
Macro Avg Recall	0,6634 ± 0,0982	0,6578 ± 0,0915	0,7262 ± 0,0819	0,7541 ± 0,0611	0,7402 ± 0,0629
Weighted Avg Recall	0,8302 ± 0,0600	0,8237 ± 0,0635	0,7477 ± 0,0981	0,7651 ± 0,1033	0,7754 ± 0,0943
Macro Avg F1-Score	0,6821 ± 0,1096	0,6782 ± 0,1001	0,6893 ± 0,0821	0,7123 ± 0,0821	0,7126 ± 0,0724
Weighted Avg F1-Score	0,8000 ± 0,0773	0,7968 ± 0,0730	0,7596 ± 0,0914	0,7763 ± 0,0957	0,7817 ± 0,0906
AUC	0,7387 ± 0,0774	0,7513 ± 0,0755	0,7467 ± 0,0772	0,7364 ± 0,0644	0,7405 ± 0,0768

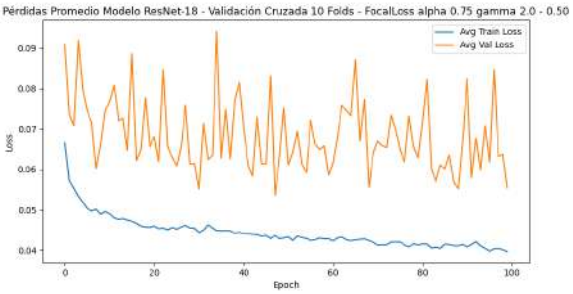
Anexo E. ResNet-18



(a) Función de pérdida CrossEntropyLoss



(b) Función de pérdida BCEWithLogitsLoss



(c) Función de pérdida Focal Loss

Figura E.1: Gráficos de pérdida en entrenamiento y validación del modelo ResNet-18 para tres funciones de pérdida diferentes.

Tabla E.1: Tabla resumen de las métricas de evaluación del modelo ResNet-18 usnado tres diferentes funciones de pérdida.

Métrica	BCEWithLogitsLoss	CrossEntropyLoss	Focal Loss α=0,75 γ=2,0
Accuracy	0,7383 ± 0,0730	0,7329 ± 0,0909	0,7898 ± 0,0695
Macro Avg Precision	0,7593 ± 0,0786	0,7722 ± 0,0775	0,7230 ± 0,0841
Weighted Avg Precision	0,7593 ± 0,0786	0,7722 ± 0,0775	0,8159 ± 0,0639
Macro Avg Recall	0,7383 ± 0,0730	0,7329 ± 0,0909	0,7370 ± 0,0688
Weighted Avg Recall	0,7383 ± 0,0730	0,7329 ± 0,0909	0,7898 ± 0,0695
Macro Avg F1-Score	0,7323 ± 0,0769	0,7193 ± 0,1078	0,7175 ± 0,0750
Weighted Avg F1-Score	0,7323 ± 0,0769	0,7193 ± 0,1078	0,7938 ± 0,0690
AUC	0,7605 ± 0,0758	0,7952 ± 0,0833	0,7556 ± 0,0652

Anexo F. Espacios Latentes

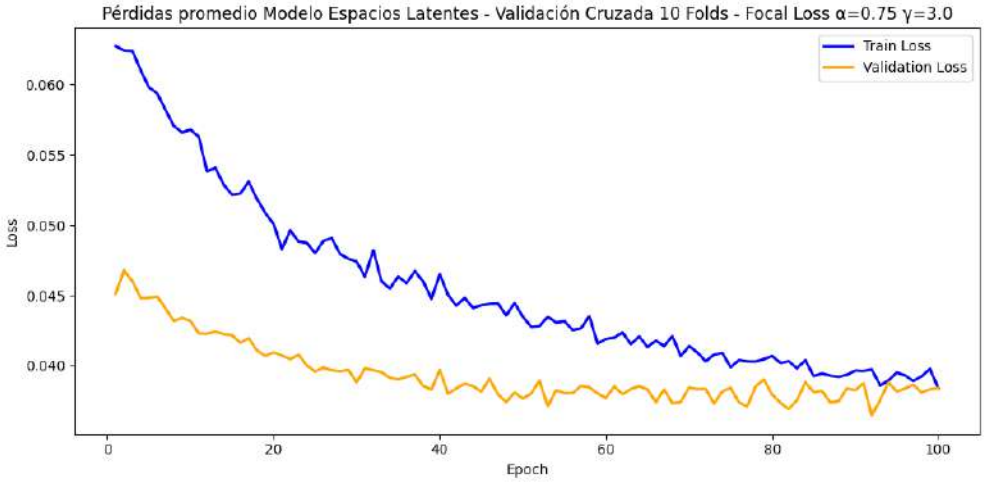


Figura F.1: Gráfico de pérdidas en entrenamiento y validación del mejor modelo de espacios latentes.

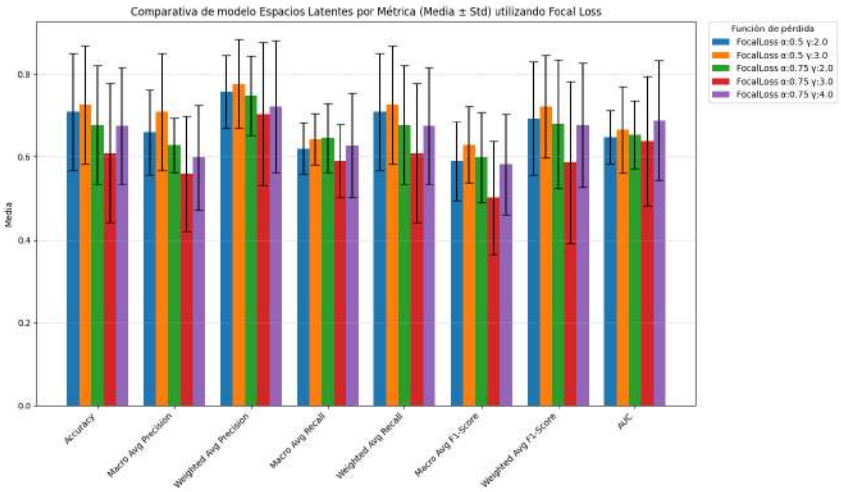


Figura F.2: Gráfico de comparación de métricas de evaluación entre iteraciones sobre hiperparámetros para la función de pérdida Focal Loss en el modelo espacios latentes.

Tabla F.1: Iteración sobre hiperparámetros para la función de pérdida Focal Loss en el modelo de espacios latentes.

Métrica	α=0,5 γ=2,0	α=0,5 γ=3,0	α=0,75 γ=2,0	α=0,75 γ=3,0	α=0,75 γ=4,0
Accuracy	$0{,}7091 \pm 0{,}1415$	$0{,}7260 \pm 0{,}1428$	$0{,}6770 \pm 0{,}1435$	$0{,}6104 \pm 0{,}1683$	$0{,}6747 \pm 0{,}1410$
Macro Avg Precision	$0{,}6603 \pm 0{,}1031$	$0{,}7092 \pm 0{,}1417$	$0{,}6291 \pm 0{,}0660$	$0{,}5597 \pm 0{,}1390$	$0{,}5989 \pm 0{,}1262$
Weighted Avg Precision	$0{,}7573 \pm 0{,}0884$	$0{,}7768 \pm 0{,}1072$	$0{,}7480 \pm 0{,}0959$	$0{,}7035 \pm 0{,}1726$	$0{,}7219 \pm 0{,}1585$
Macro Avg Recall	$0{,}6200 \pm 0{,}0620$	$0{,}6427 \pm 0{,}0621$	$0{,}6463 \pm 0{,}0838$	$0{,}5903 \pm 0{,}0876$	$0{,}6280 \pm 0{,}1254$
Weighted Avg Recall	$0{,}7091 \pm 0{,}1415$	$0{,}7260 \pm 0{,}1428$	$0{,}6770 \pm 0{,}1435$	$0{,}6104 \pm 0{,}1683$	$0{,}6747 \pm 0{,}1410$
Macro Avg F1-Score	$0{,}5903 \pm 0{,}0952$	$0{,}6302 \pm 0{,}0911$	$0{,}5990 \pm 0{,}1093$	$0{,}5016 \pm 0{,}1368$	$0{,}5826 \pm 0{,}1218$
Weighted Avg F1-Score	$0{,}6934 \pm 0{,}1365$	$0{,}7217 \pm 0{,}1244$	$0{,}6788 \pm 0{,}1550$	$0{,}5869 \pm 0{,}1951$	$0{,}6768 \pm 0{,}1496$
AUC	$0{,}6473 \pm 0{,}0653$	$0{,}6662 \pm 0{,}1047$	$0{,}6539 \pm 0{,}0820$	$0{,}6387 \pm 0{,}1557$	$0{,}6885 \pm 0{,}1445$