

INSTITUTO TECNOLÓGICO DE CHIHUAHUA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“MODELO DE APRENDIZAJE PROFUNDO PARA ANÁLISIS DE MUESTRAS
UROLÓGICAS, FROTIS SANGUÍNEO Y CULTIVOS MICROSCÓPICOS”

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS EN INGENIERÍA ELECTRÓNICA

PRESENTA:

FLAVIO EMMANUEL ACOSTA ESPINOZA

DIRECTOR DE LA TESIS:
DR. JUAN ALBERTO RAMIREZ QUINTANA
CODIRECTOR DE LA TESIS:
DR. MARIO IGNACIO CHACÓN MURGUÍA

CHIHUAHUA, CHIH., ENERO 2024



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®





Chihuahua, Chih. 11 de enero de 2024

**C. FLAVIO EMMANUEL ACOSTA ESPINOZA
PRESENTE**

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica y a propuesta de su Comité Tutorial, la División de Estudios de Posgrado e Investigación le concede la autorización para imprimir la tesis titulada **"Modelo de aprendizaje profundo para análisis de muestras urológicas, frotis sanguíneo y cultivos microscópicos"** dirigida por el Dr. Juan Alberto Ramírez Quintana con el siguiente contenido de capítulos:

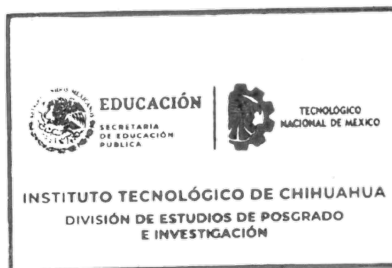
- I Marco Teórico
- II Antecedentes
- III Red neuronal profunda
- IV Metodología de diseño
- V Algoritmos de entrenamiento
- VI Resultados
- VII Conclusiones
- VIII Referencias

ATENTAMENTE

Excelencia en Educación Tecnológica®
"La Técnica por el Engrandecimiento de México"

LEONCIO OCHOA CERVANTES
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

LOC/adcs





Chihuahua, Chih. 11 de enero de 2024

LEONCIO OCHOA CERVANTES
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica, le notificamos que el documento de tesis del alumno **C. FLAVIO EMMANUEL ACOSTA ESPINOZA**, titulado **"Modelo de aprendizaje profundo para análisis de muestras urológicas, frotis sanguíneo y cultivos microscópicos"** dirigido por el Dr. Juan Alberto Ramírez Quintana, ha sido aprobado y aceptado para su impresión.

Por lo anterior, proponemos le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted:

ATENTAMENTE

*Excelencia en Educación Tecnológica-
La técnica por el Engrandecimiento de México*

Juan Ramírez

DR. JUAN ALBERTO RAMÍREZ QUINTANA
MIEMBRO DEL COMITÉ TUTORIAL

DR. MARIO IGNACIO CHACÓN MURGUÍA
MIEMBRO DEL COMITÉ TUTORIAL

DR. JESÚS ALFONSO MEDRANO HERMOSILLO
MIEMBRO DEL COMITÉ TUTORIAL

MTRA. ALMA DELIA CORRAL SÁENZ
MIEMBRO DEL COMITÉ TUTORIAL





CARTA CESIÓN DE DERECHOS

En la ciudad de Chihuahua el día 18 de enero de 2024, el que suscribe, C. FLAVIO EMMANUEL ACOSTA ESPINOZA con número de control G22060300, de la Maestría en Ciencias en Ingeniería Electrónica, adscrita a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del el Dr. Juan Alberto Ramírez Quintana y cede los derechos del trabajo titulado "Modelo de aprendizaje profundo para análisis de muestras urológicas, frotis sanguíneo y cultivos microscópicos", al Tecnológico Nacional de México y/o Instituto Tecnológico de Chihuahua para su difusión, divulgación, transmisión, reproducción, así como su digitalización con fines académicos y de investigación.

Flavio Acosta

C. FLAVIO EMMANUEL ACOSTA ESPINOZA





DECLARACIÓN DE ORIGINALIDAD

En la ciudad de Chihuahua el día 18 de enero de 2024, el que suscribe, C. FLAVIO EMMANUEL ACOSTA ESPINOZA de la Maestría en Ciencias en Ingeniería Electrónica, con número de control G22060300, adscrito a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual de la tesis titulada "Modelo de aprendizaje profundo para análisis de muestras urológicas, frotis sanguíneo y cultivos microscópicos" bajo la dirección el Dr. Juan Alberto Ramírez Quintana; que el contenido es original y que las fuentes de información consultadas para su fundamentación están debidamente citadas y referenciadas.

C. FLAVIO EMMANUEL ACOSTA ESPINOZA



Chihuahua, Chihuahua, enero de 2024

Dr. María Elena Álvarez-Buylla Roces
Directora de Conahcyt

Sirva la presente para saludarla e informarle que a la fecha he obtenido el grado de **Maestro en Ciencias en Ingeniería Electrónica** en la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, motivo por el cual agradezco todo el apoyo brindado por este Consejo que usted representa, con el otorgamiento de una beca que permitió dedicarme de manera exclusiva a la realización de mis estudios de posgrado y de esta manera lograr el objetivo principal del convenio establecido.

Sin otro particular quedo a sus órdenes no sin antes reiterar mi agradecimiento.
Atentamente

Flavio Emmanuel Acosta Espinoza

AGRADECIMIENTOS

Quiero expresar mi profundo agradecimiento a Dios, cuya guía y bendiciones han iluminado mi camino hacia la culminación de esta etapa académica. A mis padres, fuente inagotable de apoyo incondicional, les agradezco por motivarme constantemente a alcanzar la excelencia, brindándome no solo una educación académica, sino también inculcándome valores éticos y morales que han forjado mi carácter. Por siempre apoyarme, amarme, educarme y cuidarme, gracias.

Agradezco de manera especial a mi amada Isabel, cuyo apoyo inquebrantable ha sido mi roca en este viaje. Gracias por acompañarme en cada momento, por sostener mi mano en las adversidades y ser la luz en los momentos más oscuros. Tu amor y motivación han sido mi impulso para superar mis propias expectativas y levantarme con determinación tras cada caída, así como en los momentos de fastidio y frustración. Gracias por ser mi mayor motivación y admiración, por mostrarme que siempre es posible ir más allá.

Agradezco a mis amigos cercanos que han estado a mi lado durante este desafiante viaje. Sus palabras de aliento, risas compartidas y momentos de distracción han sido un bálsamo invaluable en los momentos de tensión.

A mis compañeros de laboratorio que, con su diversidad de perspectivas, han enriquecido mis experiencias académicas. A través de nuestras interacciones, he aprendido a apreciar la importancia de la colaboración y el intercambio de ideas en el proceso de aprendizaje, entendiendo el mundo desde distintos panoramas.

A mi asesor, agradezco sinceramente por su paciencia, orientación y disposición para escuchar mis inquietudes. Gracias por brindarme la oportunidad de crecer académica y personalmente, así como por abrir nuevos horizontes en mi desarrollo profesional, y una percepción del mundo distinta a la que tenía al momento de iniciar este recorrido.

También quiero expresar mi gratitud a todos los profesores y mentores que han contribuido a mi formación académica. Sus enseñanzas y orientación han sido fundamentales para mi desarrollo intelectual y profesional. Cada lección compartida ha dejado una impresión duradera en mi trayectoria.

Por último, agradezco a todas las personas anónimas que de alguna manera contribuyeron a mi éxito. Cada interacción, ya sea pequeña o grande, ha dejado una marca en mi travesía y ha sido parte integral de mi crecimiento.

Este logro no solo es mío, sino también de aquellos que han dejado su huella en mi viaje académico y personal. A todos ustedes, mi más sincero agradecimiento por su apoyo incondicional.

RESUMEN

MODELO DE APRENDIZAJE PROFUNDO PARA ANÁLISIS DE MUESTRAS UROLÓGICAS, FROTIS SANGUÍNEO Y CULTIVOS MICROSCÓPICOS

Flavio Emmanuel Acosta Espinoza

Maestría en Ciencias en Ingeniería Electrónica

División de Estudios de Posgrado e Investigación del

Instituto Tecnológico de Chihuahua

Chihuahua, Chih. 2023

Director de Tesis: Dr. Juan Alberto Ramírez Quintana

Codirector de Tesis: Dr. Mario Ignacio Chacón Murguía

Los estudios de sedimento urinario, frotis sanguíneo y la identificación de microorganismos han sido útiles para la prevención y diagnóstico de enfermedades renales e infecciosas. Para esto, las redes neuronales profundas se han convertido en una opción prometedora para la eficiencia y precisión en estos análisis, ya que aprenden y reconocen patrones complejos en datos de interés médico. Para contribuir en dichos esfuerzos, esta tesis propone una red neuronal profunda llamada *EfficientNet* de baja profundidad (LDE-Net) que analiza muestras microscópicas de sedimento urinario, frotis sanguíneo y cultivos de microorganismos como bacterias y levaduras. Como datos de referencia, se utilizaron tres conjuntos para validar y evaluar LDE-Net, los cuáles son: *Urinary Sediment Dataset*(USD) para urología, *Rabin-wbc* para frotis sanguíneo y *Bactery and Yeast Real Images* (BYRI) para bacterias y levaduras. LDE-Net logró un rendimiento mayor a 95% de exactitud en validación para las tres bases de datos y un tiempo de procesamiento de 11 ms por imagen, con 2.45 millones de parámetros. Las pruebas de validación cruzada mostraron que el modelo generaliza en los distintos conjuntos de datos, y que no existen indicios de sesgo.

CONTENIDO

INTRODUCCIÓN	xv
1. MARCO TEÓRICO	1
1.1 Conceptos básicos de urología	1
1.1.1. Urología	1
1.1.2. Muestras urológicas.....	1
1.1.3. Métodos de análisis	1
1.1.4. Tipos de objetos en muestras	2
1.2. Conceptos básicos de frotis sanguíneo	2
1.2.1. Frotis sanguíneo	2
1.2.2. Tipos de leucocitos.....	3
1.2.3. Relación de leucocitos con padecimientos médicos	3
1.3. Conceptos básicos de cultivos de microorganismos	4
1.3.1. Caldo nutritivo	4
1.3.2. Microorganismos	4
1.3.3. Padecimientos asociados a microorganismos.....	5
1.4. Arquitectura de redes neuronales profundas.....	5
1.4.1. Redes neuronales profundas para clasificación	7
1.4.2. Redes neuronales profundas para generación de datos	9
2. ANTECEDENTES	11
2.1. Urinary Sediment Dataset (USD).....	11
2.2. Raabin-wbc	13
2.3. Bacteria and Yeast Real Image (BYRI).....	14
2.4. Estado del arte	17
2.4.1. Modelos de procesamiento de imágenes médicas.....	17
2.4.2. Modelos de procesamiento de imágenes de sedimento urinario.....	19
3. RED NEURONAL PROFUNDA.....	22
3.1. MobileNet	23
3.2. DenseNet.....	25
3.3. EfficientNet.....	27
4. METODOLOGÍA DE DISEÑO DE ARQUITECTURA.....	29
4.1. Análisis de la base de datos y homogeneización de imágenes	29
4.2. Pruebas preliminares con arquitecturas de modelos.....	30
4.3. Visualización y análisis de activaciones de capas.....	30
4.4. Comportamiento estadístico de los pesos en las capas.....	32

4.5.	Selección de la arquitectura EfficientNet y diseño de LDE-Net	33
4.6.	Pruebas con diferentes optimizadores y arquitecturas.....	34
4.7.	Ampliación de clases y diseño de LDE-Net V2.....	36
4.8.	Uso de LDE-Net V2 en otras bases de datos	38
5.	ALGORITMOS DE ENTRENAMIENTO	40
5.1.	Entrenamiento tradicional.....	40
5.2.	Cambios en el algoritmo de entrenamiento	40
5.2.1.	Algoritmo Glorot-Bengio	41
5.3.	Transferencia de aprendizaje	41
5.4.	Reentrenamiento.....	42
5.5.	Meta aprendizaje	¡Error! Marcador no definido.
6.	RESULTADOS	43
6.1.	Urinary Sediment Dataset.....	43
6.1.1.	Entrenamiento con 3 clases	44
6.1.2.	Entrenamiento con 5 clases	47
6.1.3.	Entrenamiento con 7 clases	51
6.1.4.	Entrenamiento con 7 clases mediante otros algoritmos de entrenamiento	53
6.1.5.	Validación cruzada.....	55
6.2.	Raabin-wbc	56
6.2.1.	Entrenamiento	57
6.2.2.	Validación cruzada	59
6.3.	BYRI.....	62
6.3.1.	Entrenamiento	62
6.3.2.	Validación cruzada	64
7.	CONCLUSIONES.....	68
8.	REFERENCIAS	70

Índice de figuras.

Figura 1.1 Diagrama general de neurona artificial.....	6
Figura 1.2 Red neuronal de tres capas.....	6
Figura 1.3 Red neuronal profunda de cuatro capas.....	7
Figura 2.1 Imagen de base de datos USD.....	12
Figura 2.2 Muestras recortadas por clase de USD.....	12
Figura 2.3 Muestras separadas de cada clase en base de datos Raabin-wbc.....	14
Figura 2.4 Muestra de base de datos BYRI de la clase candida albicans.....	15
Figura 2.5 Muestra de base de datos BYRI de la clase candida albicans.....	15
Figura 2.6 Muestra de base de datos BYRI de la clase candida albicans.....	16
Figura 2.7 Muestras separadas de cada clase en base de datos BYRI.....	16
Figura 3.1 Simbología de bloques a utilizar en redes.....	22
Figura 3.2 Arquitectura general de MobileNetV2.....	23
Figura 3.3 Arquitectura de bloque bottleneck.....	24
Figura 3.4 Arquitectura general de DenseNet.....	25
Figura 3.5 Arquitectura de bloque denso.....	26
Figura 3.6 Capa de transición de bloque denso.....	26
Figura 3.7 Arquitectura general de EfficientNet.....	27
Figura 3.8 Arquitectura de bloque MBConv.....	27
Figura 4.1 Imagen nl00030-cast_1.....	30
Figura 4.2 Activaciones de la capa 2 en las distintas arquitecturas.....	31
Figura 4.3 Activaciones de la capa 8 en las distintas arquitecturas.....	31
Figura 4.4 Activaciones de la capa 36 en las distintas arquitecturas.....	31
Figura 4.5 Activaciones en las capas 14 y 15 en EfficientNetV2L.....	32
Figura 4.6 Activaciones en las capas 32 y 33 en EfficientNetV2L.....	32
Figura 4.7 Gráficas de exactitud, media, desviación estándar y entropía, en la capa 14 de EfficientNetV2L.....	33
Figura 4.8 Diferencia entre EfficientNetV2 y LDE-Net.....	34
Figura 4.9 Activaciones en las capas 3 y 4 en LDE-Net.....	37
Figura 4.10 Activaciones en las capas 24 y 25 en LDE-Net.....	37
Figura 4.11 Comparación entre EfficientNetV2, LDE-Net y LDE-NetV2.....	38
Figura 6.1 Gráficas de exactitud y pérdida en distintas arquitecturas para 3 clases utilizando optimizador Adam.....	45
Figura 6.2 Gráficas de exactitud y pérdida en distintas arquitecturas para 3 clases utilizando optimizador SGD.....	47
Figura 6.3 Gráficas de exactitud y pérdida en distintas arquitecturas para 5 clases utilizando optimizador Adam.....	49
Figura 6.4 Gráficas de exactitud y pérdida en distintas arquitecturas para 5 clases utilizando optimizador SGD.....	50
Figura 6.5 Gráficas de exactitud y pérdida en LDE-Net para 7 clases utilizando optimizador Adam.....	51
Figura 6.6 Gráficas de exactitud y pérdida en LDE-Net y LDE-NetV2 para 7 clases utilizando optimizador Adam.....	53
Figura 6.7 Matriz de confusión obtenida en LDE-NetV2 + EG con 7 clases.....	54
Figura 6.8 Gráficas de exactitud en 5 subconjuntos para validación cruzada para 7 clases de USD_2.....	56
Figura 6.9 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos Raabin-wbc.....	57
Figura 6.10 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos Raabin-wbc utilizando transferencia de aprendizaje.....	58
Figura 6.11 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos Raabin-wbc utilizando reentrenamiento.....	59
Figura 6.12 Gráficas de exactitud en 5 subconjuntos para validación cruzada en Raabin-wbc.....	60
Figura 6.13 Gráficas de exactitud en 5 subconjuntos para validación cruzada en Raabin-wbc utilizando transferencia de aprendizaje.....	61

Figura 6.14 Gráficas de exactitud en 5 subconjuntos para validación cruzada en Raabin-wbc utilizando reentrenamiento.....61

Figura 6.15 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos BYRI.62

Figura 6.16 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos BYRI utilizando transferencia de aprendizaje.63

Figura 6.17 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos BYRI utilizando reentrenamiento.64

Figura 6.18 Gráficas de exactitud en 5 subconjuntos para validación cruzada en BYRI.65

Figura 6.19 Gráficas de exactitud en 5 subconjuntos para validación cruzada en BYRI utilizando transferencia de aprendizaje.66

Figura 6.20 Gráficas de exactitud en 5 subconjuntos para validación cruzada en BYRI utilizando reentrenamiento.....67

Índice de tablas.

Tabla 2.1 Cantidad de muestras por clase en base de datos USD_2.	13
Tabla 2.2 Cantidad de muestras por clase en base de datos Raabin-wbc.	14
Tabla 2.3 Cantidad de muestras por clase en base de datos BYRI.	16
Tabla 4.1 Comparación entre EfficientNetV2L y LDE-Net.	34
Tabla 4.2 Comparación entre EfficientNetV2L, LDE-Net y LDE-NetV2.	38
Tabla 6.1 Resultados de validación en distintas arquitecturas para 3 clases utilizando optimizador Adam.	44
Tabla 6.2 Resultados de validación en LDE-Net para 3 clases utilizando optimizador Adam.	45
Tabla 6.3 Resultados de validación en distintas arquitecturas para 3 clases utilizando optimizador SGD.	46
Tabla 6.4 Resultados de validación en distintas arquitecturas para 5 clases utilizando optimizador Adam.	47
Tabla 6.5 Resultados de validación en distintas arquitecturas para 5 clases utilizando optimizador SGD.	50
Tabla 6.6 Resultados de validación en LDE-Net para 7 clases utilizando optimizador Adam.	51
Tabla 6.7 Resultados de validación en LDE-NetV2 para 7 clases utilizando optimizador Adam.	52
Tabla 6.8 Resultados de validación en distintas versiones de LDE-Net y el uso de entropy gain.	53
Tabla 6.9 Resultados en transferencia de aprendizaje utilizando LDE-NetV2.	54
Tabla 6.10 Comparación de LDE-NetV2 + EG con trabajos en el estado del arte.	55
Tabla 6.11 Resultados validación cruzada USD_2 7 clases.	56
Tabla 6.12 Resultados de validación en base de datos Raabin-wbc utilizando LDE-NetV2 +EG.	57
Tabla 6.13 Resultados de validación en base de datos Raabin-wbc utilizando transferencia de aprendizaje.	58
Tabla 6.14 Resultados de validación en base de datos Raabin-wbc utilizando reentrenamiento.	58
Tabla 6.15 Resultados validación cruzada Raabin-wbc.	59
Tabla 6.16 Resultados validación cruzada Raabin-wbc utilizando transferencia de aprendizaje.	60
Tabla 6.17 Resultados validación cruzada Raabin-wbc utilizando reentrenamiento.	61
Tabla 6.18 Resultados de validación en base de datos BYRI utilizando LDE-NetV2 +EG.	62
Tabla 6.19 Resultados de validación en base de datos BYRI utilizando transferencia de aprendizaje.	63
Tabla 6.20 Resultados de validación en base de BYRI utilizando reentrenamiento.	63
Tabla 6.21 Resultados validación cruzada BYRI.	64
Tabla 6.22 Resultados validación cruzada BYRI utilizando transferencia de aprendizaje.	65
Tabla 6.23 Resultados validación cruzada BYRI utilizando reentrenamiento.	66

GLOSARIO

INTRODUCCIÓN

Los estudios clínicos de laboratorio desempeñan un papel fundamental en el diagnóstico, seguimiento y tratamiento de enfermedades ya que proporcionan información valiosa sobre el estado de salud de un paciente. Entre los estudios clínicos más comunes se encuentran el análisis de sedimento urinario, el estudio de frotis sanguíneo y la identificación de bacterias y levaduras a través de cultivos microbiológicos.

En la búsqueda constante de mejorar la eficiencia y la precisión en el análisis de muestras de laboratorio, la aplicación de técnicas de inteligencia artificial y redes neuronales profundas ha emergido como una opción prometedora [1][2]. Estas redes neuronales han demostrado ser capaces de aprender y reconocer patrones complejos en datos médicos, imagenología y señales biológicas[3][4]. Su capacidad para automatizar la clasificación y detección de patrones en muestras clínicas ha abierto nuevas perspectivas en el ámbito de la medicina de diagnóstico[5] [6]. Por ejemplo, en [7] se utiliza una red profunda para clasificación de células sanguíneas, mientras que en [8], se utilizan redes convolucionales para segmentación en imágenes del disco óptico para diagnóstico de glaucoma. Así mismo, en [9] se utilizan redes neuronales convolucionales para la segmentación de tomografías computarizadas, para la detección de cáncer en vejiga.

Sin embargo, la mayoría de los trabajos encontrados en el estado del arte se enfocan en un tipo de análisis en particular, limitando el alcance de los experimentos de una red neuronal a cierto tipo de padecimiento o muestra microscópica. Por ello, en esta tesis se propone el uso de redes neuronales profundas para el análisis de muestras de laboratorio, teniendo como alcance muestras de sedimento urinario, frotis sanguíneo y cultivos de bacterias y levaduras. Para llevar a cabo esta investigación, se utilizaron tres conjuntos de datos de referencia:

- USD, que contiene muestras de urología.
- Raabin-wbc, que incluye muestras de frotis sanguíneo.
- BYRI, que abarca muestras de bacterias y levaduras.

Estas bases de datos se han convertido en pilares fundamentales para la validación y evaluación de nuestros modelos.

Un enfoque clave de este estudio se centra en la exploración del rendimiento en la clasificación de muestras de sedimento urinario con tres arquitecturas de redes neuronales profundas: *MobileNetV2*, *DenseNet201* y *EfficientNetV2L*. A partir de los resultados obtenidos, se desarrollaron dos versiones de arquitectura basadas en *EfficientNetV2*, denominadas *EfficientNet* de baja profundidad (LDE-Net) y LDE-Net versión 2 (LDE-NetV2). Estos modelos representan un paso significativo hacia la creación de herramientas más precisas y eficientes para el análisis de muestras clínicas, con el potencial de mejorar la atención médica y la toma de decisiones en el ámbito de la medicina de diagnóstico.

La tesis se estructura en siete capítulos que abordan aspectos esenciales de la investigación. El Capítulo 1 establece un marco teórico que comprende conceptos clave en urología, frotis sanguíneo, cultivos de microorganismos y redes neuronales profundas. El Capítulo 2 presenta antecedentes, incluyendo bases de datos y el estado del arte en procesamiento de imágenes médicas. El Capítulo 3 se centra en las arquitecturas de redes neuronales profundas utilizadas. El Capítulo 4 detalla la metodología de diseño de las arquitecturas propuestas basadas en EfficientNetV2. El Capítulo 5 aborda los algoritmos de entrenamiento utilizados. El Capítulo 6 presenta resultados divididos por bases de datos, así como pruebas de validación cruzada. Finalmente, el Capítulo 7 resume las conclusiones derivadas de los hallazgos.

1. MARCO TEÓRICO

El presente capítulo aborda los fundamentos teóricos que sustentan y enriquecen el desarrollo de la investigación en el ámbito del análisis de muestras microscópicas mediante el empleo de técnicas de aprendizaje profundo. Las muestras que se revisarán en esta tesis provienen de bases de datos de sedimento urinario, frotis sanguíneo y cultivos de bacterias y levaduras. Con base en lo dicho, el presente capítulo se organiza de la siguiente manera: las secciones 1.1, 1.2 y 1.3 tratan los conceptos básicos de los microorganismos en las muestras utilizadas en esta tesis. La Sección 1.4 trata la arquitectura de redes neuronales profundas.

1.1 Conceptos básicos de urología

Con la finalidad de poner en contexto respecto al tema de investigación, se presentan de manera general conceptos relacionados al área de urología, métodos de análisis de muestras urológicas, así como la relevancia de la presencia de distintos objetos en las mismas.

1.1.1. Urología

Girona y Conejero definen la urología en [10], como la parte de la medicina que se refiere al estudio de la estructura, funciones y anomalías en el aparato urinario. La urología, se relaciona estrechamente con la anatomía, fisiología, fisiopatología, etiología, nosología, diagnóstico médico, entre otras áreas. El propósito principal de la urología, es conocer el comportamiento del aparato urinario y su interacción con los distintos sistemas del cuerpo humano, como con agentes y eventos ambientales.

1.1.2. Muestras urológicas

El uroanálisis o análisis de muestras urológicas, permite tener información necesaria para un posterior diagnóstico. Según Campos Otegui en [11], es necesario seguir una metodología para la extracción de la muestra, tratamiento y análisis. En el caso de la obtención de muestras, es indispensable la asepsia del recipiente contenedor. Además, la muestra debe ser recolectada con al menos cuatro horas de concentración, siendo recomendable 8 horas, con la finalidad de obtener suficiente concentración en los componentes de la orina. Finalmente, se sugiere un tiempo de espera entre la extracción de la muestra y el análisis de máximo dos horas.

1.1.3. Métodos de análisis

En el análisis de muestras urológicas, existen dos grandes áreas, las cuales son: el análisis macroscópico y microscópico. El análisis macroscópico consiste en observar las características fisicoquímicas de la orina, como el color y el potencial de hidrógeno (pH). Dependiendo de los resultados encontrados en este tipo de estudio, se

decide si se realiza un análisis microscópico. En el análisis microscópico o análisis de sedimento urinario, consiste en preparar una muestra de orina para distinguir componentes microscópicos dentro de la misma.

1.1.4. Tipos de objetos en muestras

Es posible encontrar varios tipos de objetos en muestras de urología que se asocian a distintas enfermedades que deben ser tratadas. Por ello, es importante conocer dichos objetos y su cantidad en estas muestras. Entre estos objetos se encuentran: eritrocitos, leucocitos, bacterias, cristales, células epiteliales y hongos. Dependiendo del tipo de objeto encontrado, es posible inferir un diagnóstico médico. En [12], se explica la importancia y posible diagnóstico a partir de los objetos encontrados, que pueden ser:

Eritrocitos: la presencia de eritrocitos en muestras es anormal y requiere una investigación más profunda. Posibles diagnósticos pueden ser cistitis, uretritis, inflamación de órganos cercanos al tracto urinario, entre otros.

Leucocitos: el hallazgo de más de cinco leucocitos por imagen, es considerado anormal. Puede indicar infección urinaria, tuberculosis, urolitiasis, entre otros padecimientos.

Bastones: en caso de encontrarse bastones de leucocitos, es sugerencia de pielonefritis. Bastones de eritrocitos pueden ser indicio de glomerulitis o vasculitis.

Cristales: pueden ser formados de forma regular en muestras con temperaturas menores al ambiente. Existen distintos tipos de cristales que indican distintos padecimientos.

Células epiteliales: las células epiteliales, indican contaminación de la muestra urinaria. En caso de presencia de gran cantidad de células o aquellas con morfología anormal (núcleos de gran tamaño, múltiples núcleos, etc.) son indicativos de proceso maligno afectando el urotelio.

1.2. Conceptos básicos de frotis sanguíneo

Con la finalidad de poner en contexto respecto el tema de investigación, se presentan de manera general conceptos relacionados al análisis de frotis sanguíneo, los tipos de leucocitos, su influencia en el organismo y su relación con padecimientos de salud.

1.2.1. Frotis sanguíneo

Un frotis sanguíneo es una prueba de laboratorio que se realiza para examinar la apariencia, cantidad y forma de los leucocitos, eritrocitos y plaquetas. El frotis sanguíneo se realiza al extender una gota de sangre sobre la superficie de un portaobjetos, con el fin de analizarla en el al microscopio [13]. El objetivo de esta prueba es

identificar y detallar las células de la sangre de manera directa, lo que permite detectar alteraciones en la forma de las células que pueden ser una identificación específica de algunas enfermedades. El frotis sanguíneo se utiliza para diagnosticar y vigilar muchas afecciones como trastornos de la sangre, insuficiencia renal repentina y el tratamiento de ciertos tipos de cáncer [14]. La calidad de la información obtenida de los frotis depende en gran parte de la calidad con que se hayan hecho estas preparaciones, por lo que es importante seguir un protocolo adecuado para la preparación del frotis sanguíneo [15].

1.2.2. Tipos de leucocitos

En el análisis de un frotis sanguíneo se pueden identificar diferentes tipos de leucocitos tales como los basófilos, eosinófilos, linfocitos, monocitos y neutrófilos. Cada uno de estos tipos de leucocitos tiene una función específica en el sistema inmunológico del cuerpo que son:

- Basófilos. Liberan histamina y otros químicos en respuesta a una infección o alérgeno.
- Eosinófilos. Combaten las infecciones parasitarias y las alergias.
- Linfocitos. Responsables de la producción de anticuerpos y la eliminación de células infectadas.
- Monocitos. Encargados de fagocitar y eliminar bacterias y otros microorganismos.
- Neutrófilos. Son los primeros en responder a una infección y fagocitar bacterias y otros microorganismos.

La identificación de alteraciones en la cantidad o forma de estos leucocitos en un frotis sanguíneo puede ser una identificación específica de algunas enfermedades [16].

1.2.3. Relación de leucocitos con padecimientos médicos

Conocer los tipos de leucocitos presentes en la muestra permite realizar diagnósticos médicos. En [17], se explica la importancia y posible diagnóstico a partir de los objetos encontrados, que pueden ser:

Basófilos: la leucemia está asociada a un incremento de basófilos, mientras que la disminución de estos es indicador de hipertiroidismo e infecciones agudas.

Eosinófilos: el incremento de eosinófilos indica reacciones alérgicas, infección parasitaria y malignidad. Por su parte, un decremento es indicador de síndrome de Cushing y estrés derivado por traumas.

Linfocitos: al existir un incremento, es indicador de leucemia crónica, reacciones hipersensibles e infecciones virales. Un decremento puede indicar síndrome de inmunodeficiencia adquirida, sepsis y anemia aplásica.

Monocitos: un incremento en la cantidad de monocitos está asociado con enfermedades autoinmunes e infecciones por hongos o protozoarios. Por su parte, un decremento en los mismos indica anemia aplásica, leucemia de células peludas e infecciones agudas.

Neutrófilos: un conteo alto de neutrófilos sugiere inflamaciones crónicas e infecciones, mientras que un conteo bajo está asociado con el síndrome *Chediak-Hagashi*, el síndrome de *Kostmann* y neutropenia autoinmune.

1.3. Conceptos básicos de cultivos de microorganismos

Existen distintos medios para realizar cultivos de microorganismos, pero esta Sección se enfoca en el cultivo mediante caldo nutritivo. A continuación, se profundiza respecto a este tipo de medio y los microorganismos cuyo crecimiento es favorecido mediante distintos padecimientos.

1.3.1. Caldo nutritivo

Un caldo nutritivo es un medio de cultivo líquido utilizado para el crecimiento de microorganismos. Es ampliamente utilizado en laboratorios para fomentar la reproducción de microorganismos, con la finalidad de poder posteriormente analizar las muestras. El caldo nutritivo contiene nutrientes como peptona de gelatina y extracto de carne, dependiendo del tipo de microorganismo al cual se desea fomentar el crecimiento [18].

1.3.2. Microorganismos

Al utilizar el caldo nutritivo como medio de cultivo, se tienen los nutrientes necesarios para favorecer el crecimiento de distintos microorganismos, tal es el caso de *escherichia coli*, *staphylococcus epidermidis*, *staphylococcus aureus* y *candida albicans* [19][20]. A continuación, se abarca de manera cada uno de estos microorganismos.

Escherichia coli: este microorganismo, comúnmente conocido como E. coli, es una bacteria que forma parte del microbiota intestinal de los seres humanos y otros animales.

Staphylococcus epidermidis: es una bacteria que se encuentra comúnmente en la piel humana. Aunque generalmente es inofensiva, puede causar infecciones en personas con sistemas inmunológicos debilitados o cuando se introduce en el torrente sanguíneo por de procedimientos médicos invasivos.

Staphylococcus aureus: staphylococcus aureus es otra especie de bacteria estafilococo que se encuentra en la piel y las mucosas. Es más patógena en comparación con staphylococcus epidermis y puede causar una variedad de infecciones, cutáneas o en órganos internos.

Candida albicans: a diferencia de los microorganismos mencionados anteriormente, candida albicans es una levadura (hongo). El caldo nutritivo también puede ser adaptado para el crecimiento de hongos como Candida albicans, proporcionando fuentes de carbono y nitrógeno adecuadas, así como vitaminas y minerales necesarios para su desarrollo. Candida albicans es conocida por causar las infecciones de la candidiasis.

1.3.3. Padecimientos asociados a microorganismos

Los tres microorganismos analizados en este trabajo son *staphylococcus aureus*, *escherichia coli* y *candida albicans*. *Staphylococcus aureus* es una bacteria que puede causar infecciones cutáneas como furúnculos y celulitis. Además, es responsable de infecciones graves como neumonía, osteomielitis, endocarditis y sepsis. Algunas cepas son resistentes a varios antibióticos y se denominan *staphylococcus aureus* resistente a la meticilina (SARM), lo que dificulta aún más su tratamiento [21][22]. *Candida albicans* es un hongo que puede causar candidiasis, una infección que afecta áreas diversas del cuerpo, como la piel, las mucosas, la garganta, el esófago y los genitales. Estas infecciones pueden variar desde leves, como la candidiasis oral (muguet), hasta graves, como la candidiasis sistémica, especialmente en pacientes inmunocomprometidos [23][24]. *Escherichia coli* está generalmente asociada a enfermedades gastrointestinales, sin embargo algunas cepas pueden producir toxinas que conducen a infecciones graves, como la colitis hemorrágica y el síndrome urémico hemolítico (SUH), que pueden resultar en daño renal y, en casos graves, la muerte.[25].

Es importante destacar que estos microorganismos forman parte del microbiota del cuerpo humano y pueden ser beneficiosos en ciertos contextos. Sin embargo, si el sistema inmunológico está debilitado o cuando se produce una proliferación excesiva, pueden causar infecciones y enfermedades que requieren tratamiento médico adecuado. La identificación y el estudio de estos microorganismos son esenciales para comprender y abordar las enfermedades asociadas con ellos.

1.4. Arquitectura de redes neuronales profundas

Dentro del análisis de imágenes médicas, una de las técnicas que van tomando cada vez más importancia es el uso de modelos de aprendizaje profundo, ya que éstos son capaces de aprender características abstractas de las muestras para realizar tareas como clasificación y segmentación con un alto grado de exactitud y con tiempos de procesamiento aceptables.

La unidad básica de las redes neuronales son las neuronas artificiales, que a su vez basan su funcionamiento en las neuronas cerebrales. Cada neurona artificial tiene pesos asociados (w), que determinan la influencia de la misma en neuronas siguientes o en la salida (y). En la Figura 1.1 se puede observar el diagrama general de una neurona artificial, donde se muestran las entradas representadas con la variable x , los pesos con w y la salida con

y. Así mismo, se puede observar la analogía con una neurona cerebral, donde las entradas corresponden a las dendritas, los pesos con la sinapsis, la sumatoria con el cuerpo celular y la salida con el axón.

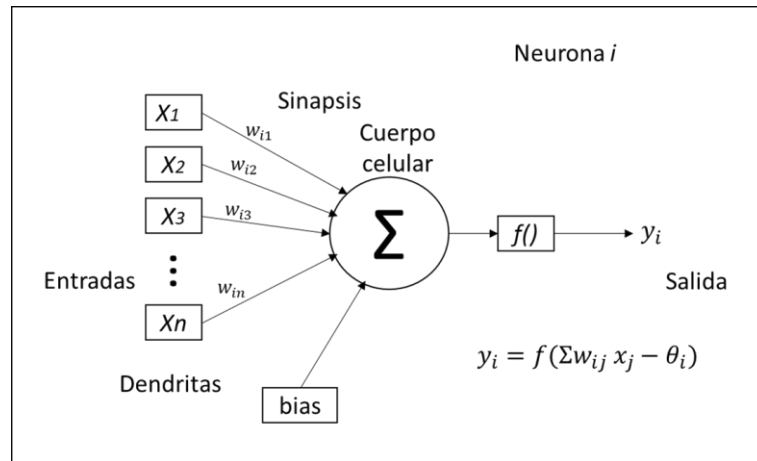


Figura 1.1 Diagrama general de neurona artificial.

A un conjunto de neuronas distribuidas en un mismo nivel se le llama capa. Cada neurona de una capa puede estar conectada únicamente con capas anteriores y posteriores, pero no entre neuronas de la misma capa. La estructura general de una red neuronal se basa en capa de entrada, capas ocultas y capa de salida. La capa de entrada es la capa inicial de la red donde se ingresan los datos que serán procesados en el resto de la arquitectura. Las capas ocultas (o intermedias) son aquellas que están entre la capa de entrada y de salida en la arquitectura. Finalmente, la capa de salida es la última dentro de la arquitectura, y es mediante la cual el sistema dará algún resultado. En la Figura 1.2 se muestra la arquitectura básica de una red neuronal de tres capas.

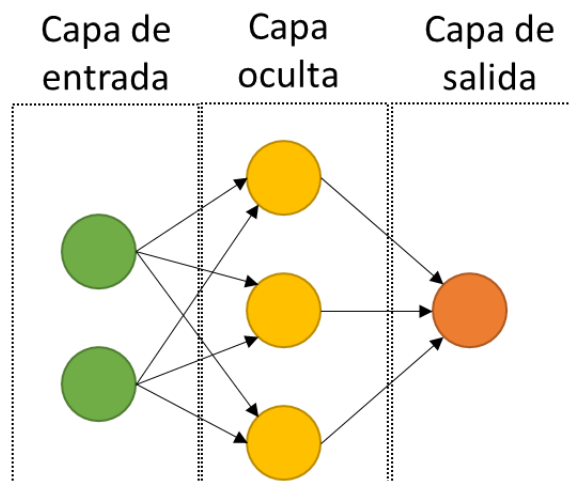


Figura 1.2 Red neuronal de tres capas.

En el caso del aprendizaje profundo, se tienen arquitecturas de redes neuronales con más de una capa oculta (de allí el nombre de profundo). En la Figura 1.3 se muestra un ejemplo de red neuronal profunda, como se puede observar (a diferencia de la Figura 1.2 que muestra una red no profunda) tiene dos capas ocultas.

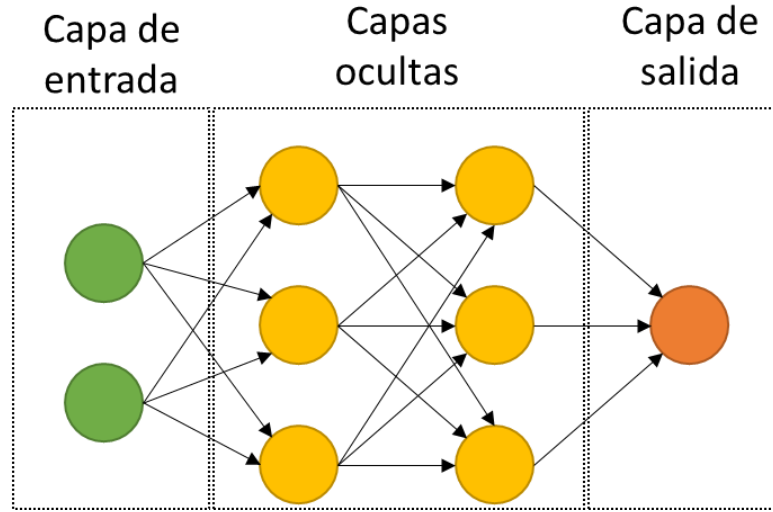


Figura 1.3 Red neuronal profunda de cuatro capas

1.4.1. Redes neuronales profundas para clasificación

Las redes neuronales profundas han demostrado un gran éxito en la clasificación de imágenes debido a su capacidad para aprender representaciones complejas de características visuales y abstractas. Sin embargo, la elección de la arquitectura de red adecuada es un desafío importante en la clasificación de imágenes debido a la complejidad del problema y la cantidad de variables a considerar. Una arquitectura de red neuronal profunda para clasificación de imágenes puede incluir varios elementos, que pueden variar en complejidad y cantidad, según la tarea a resolver. Algunos de los elementos que se pueden encontrar en una arquitectura típica de clasificación de imágenes son:

Capas convolucionales: son capas que aplican filtros para extraer características de la imagen. Estas capas son responsables de la extracción de características visuales y suelen ser el elemento principal de las redes neuronales convolucionales (CNN). Formalmente, están definidas por la ecuación (1.1).

$$h(i, j) = \sum_p \sum_q W(p, q)X(i - p + 1, j - q + 1) + b(p, q) \quad (1.1)$$

donde X es la salida de la capa anterior, W es el kernel o filtro aplicado y h es la salida de la capa actual. Así mismo, p y q son los índices correspondientes a los distintos valores tanto de la entrada de la red como del kernel,

mientras que i y j son los índices referentes a la salida de la capa. Para la primera capa, X son las entradas de la red.

Capa de no linealidad: son funciones de activación que aplican una función de no linealidad a la salida de las capas para permitir a la red neuronal que aprenda relaciones entre las características de entrada y la salida. Estas funciones evitan la saturación en el rango positivo, promueven el flujo no lineal de información y facilitan la activación dispersa, contribuyendo así a mantener gradientes significativos durante la retropropagación. Existen distintas funciones que pueden ser utilizadas como la tangente hiperbólica, sigmoide, LeakyReLU y ReLU dada por la Ecuación (1.2).

$$r(h) = \max(0, h) \quad (1.2)$$

donde h es la salida de la capa anterior y r es la salida de la actual.

Capas de agrupamiento: son capas que reducen la dimensionalidad de la salida de la capa convolucional. Esto se hace mediante la aplicación de una operación de reducción (como la media o el máximo) en ventanas de la salida. En el caso de agrupamiento por máximo, el cuál es ampliamente utilizado, está definido por la Ecuación (1.3).

$$X(r) = \max(r_m) \quad (1.3)$$

donde r es la salida de la capa anterior, r_m es la matriz de valores de la ventana a analizar y X es la salida de la capa.

Capas totalmente conectadas: Albawi et al. comentan en [26] que son capas que conectan todas las neuronas de la capa anterior con todas las neuronas de la capa siguiente. Estas capas son responsables de la clasificación final y suelen ser el último elemento de la red neuronal. Formalmente, se define por la Ecuación (1.4).

$$Z(X, W) = WX + b \quad (1.4)$$

donde W se refiere a la matriz de pesos, X a la salida de la capa anterior, b al sesgo y Z a la salida de la capa.

Abandono (Dropout): es una técnica introducida por Hinton et al. en [27] que ayuda a evitar el sobreajuste al "desactivar" aleatoriamente algunas neuronas durante el entrenamiento, lo que favorece al aprendizaje de las distintas neuronas.

Capa softmax: es una función de distribución de probabilidad que se utiliza como capa en problemas de clasificación. Esta capa toma como entrada un vector de números reales y produce como salida un vector de la misma dimensión, donde cada elemento del vector de salida representa la probabilidad de que la entrada pertenezca a una de las clases posibles. La función *softmax* se define como:

$$Y(Z) = \frac{e^{Z_i}}{\sum_{j=1}^n e^{Z_j}} \quad (1.5)$$

donde Z_i es la salida de la capa anterior, siendo a su vez el valor de entrada para la i -ésima neurona y n es el número total de neuronas en la capa. La capa *softmax* se utiliza comúnmente como la última capa en una red neuronal para problemas de clasificación, y se combina con una función de pérdida como la entropía cruzada para entrenar la red.

Estos son solo algunos de los elementos que se pueden encontrar en una arquitectura típica de red neuronal profunda para la clasificación de imágenes. La selección y combinación de estos elementos dependerá del problema específico a resolver y de la complejidad del conjunto de datos.

1.4.2. Redes neuronales profundas para generación de datos

Las redes neuronales generativas son una herramienta muy útil en el campo de la inteligencia artificial y el aprendizaje automático. Una de las principales utilidades de estas redes es su capacidad para generar datos sintéticos a partir de un conjunto de datos de entrenamiento. Estos datos pueden ser utilizados para aumentar el conjunto de datos de entrenamiento y mejorar el rendimiento de los modelos de aprendizaje automático. Por ejemplo, si se está construyendo un modelo para clasificar imágenes, se puede utilizar una red generativa para aumentar el conjunto de datos de entrenamiento con imágenes generadas sintéticamente que se asemejen a las imágenes reales. Así mismo, permiten mejorar la generalización de las redes, ya que se generan datos semejantes mas no iguales a los originales, ampliando la distribución de casos observados por la red.

Comúnmente, las redes generativas constan de un generador y un discriminador. El generador se encarga de crear datos artificiales con características similares a los reales, mientras que el discriminador busca diferenciar los datos creados por el generador de los reales. Ambas partes son entrenadas de tal manera que el discriminador sea incapaz de diferenciar entre datos reales y artificiales, debido a la semejanza entre ellos. De manera general, la arquitectura en redes generativas incluye las siguientes capas:

Capas de convolución transpuesta: estas capas están definidas por la ecuación (1.6):

$$h(i, j) = \sum_p \sum_q W(p, q) X(i - p, j - q) \quad (1.6)$$

donde X es la salida de la capa anterior, W es el kernel aplicado y h es la salida de la capa actual en el punto dado. Así mismo, p y q son los índices correspondientes a los distintos valores tanto de la entrada de la red como del kernel, mientras que i y j son los índices referentes a la salida de la capa. Estas capas se utilizan en el generador para crear datos a partir de un vector de ruido. La convolución transpuesta se utiliza para aumentar la resolución de la imagen generada.

Capas de convolución: estas capas están definidas por la ecuación (1.1) y en el caso de la generación de datos, son utilizadas en el discriminador para procesar los datos de entrada y distinguir entre los datos reales y los obtenidos por el generador.

Capas de normalización: estas capas se utilizan en ambas sub-redes para normalizar los datos de entrada y mejorar la estabilidad y eficacia del entrenamiento. Estas capas están definidas por la ecuación (1.7)

$$X(r) = \gamma \frac{\alpha_r - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta \quad (1.7)$$

donde α son las activaciones de la capa anterior, μ la media de estas activaciones, σ^2 la varianza y ε una constante pequeña para evitar la división entre cero. Por otro lado, X es la salida de la capa actual, mientras que γ y β son parámetros aprendidos por la red durante el entrenamiento.

Capas de activación: las capas de activación se utilizan en ambas sub-redes para introducir no linealidad en la red neuronal. Las funciones de activación comunes utilizadas en GAN incluyen ReLU, LeakyReLU y la tangente hiperbólica. La función ReLU está definida por la ecuación (1.3), mientras que las funciones LeakyReLU y tangente hiperbólica están definidas por las ecuaciones (1.8) y (1.9) respectivamente.

$$r(h) = \begin{cases} h & \text{si } h > 0 \\ \alpha h & \text{si } h \leq 0 \end{cases} \quad (1.8)$$

donde h es la entrada de la capa anterior, r la salida de la capa actual y α un parámetro que corresponde a la atenuación de los valores menores a cero.

$$r(h) = \frac{e^{2h} - 1}{e^{2h} + 1} \quad (1.9)$$

donde h es la entrada de la capa anterior y r la salida de la capa actual.

2. ANTECEDENTES

Los enfoques basados en el aprendizaje profundo han demostrado su capacidad para extraer patrones complejos y mejorar la precisión en la interpretación de imágenes médicas. En este contexto, el presente capítulo se enfoca en las bases de datos que respaldan el desarrollo del modelo de aprendizaje profundo y en un análisis del estado del arte. Con base en lo dicho, el presente capítulo se organiza de la siguiente manera: la Sección 2.1 presenta la base de datos de urología. La Sección 2.2 presenta la base de datos de frotis sanguíneo. La Sección 2.3 presenta la base de datos correspondiente a cultivos de bacterias y levaduras. Finalmente, la Sección 2.4 menciona trabajos relevantes encontrados en el estado del arte relacionados con procesamiento de imágenes médicas.

2.1. Urinary Sediment Dataset (USD)

La base de datos de sedimento urinario (USD, por sus siglas en inglés) fue presentada por Liang et al. en [28] durante el 2018, y posteriormente utilizada en trabajos como los presentados por Liang et al. en [29] y por Yan et al. en [30]. Está compuesta por 5,376 imágenes etiquetadas con 7 categorías de objetos que pueden ser encontrados en muestras de sedimento urinario. Dichas clases son: bacterias, cristales, células epiteliales, núcleos epiteliales, eritrocitos, leucocitos y hongos.

Las imágenes en la base de datos contienen distintos objetos por muestra, tal como se observa en la Figura 2.1, en la que se visualizan elementos tales como eritrocitos, leucocitos, células epiteliales y núcleos epiteliales. Las imágenes se acompañan de un archivo *.xml* que contiene información relevante de las imágenes, entre las que se encuentran las coordenadas espaciales de cada objeto, así como la clase a la que pertenecen. Esta información es utilizada para recortar de forma automática cada uno de los objetos de interés y se forma una nueva base de datos, que en esta tesis se llamará USD_2 y está compuesta por 42,255 imágenes de los objetos de interés que se categorizan en las clases mencionadas. En la Figura 2.2 se observan distintas muestras recortadas por clase que conforman la base de datos USD_2 y la Tabla 2.1 muestra la distribución de muestras por clase de USD_2. Se puede observar un desbalance de clases, esto se debe a la naturaleza misma de las muestras, siendo los eritrocitos la muestra más encontrada, mientras que los núcleos epiteliales son los más escasos. No obstante, esta base de datos cuenta con una variación importante en coloración, morfología y ruido presente en las muestras, lo cual conlleva a una posible mejor generalización de los modelos entrenados con esta base de datos.

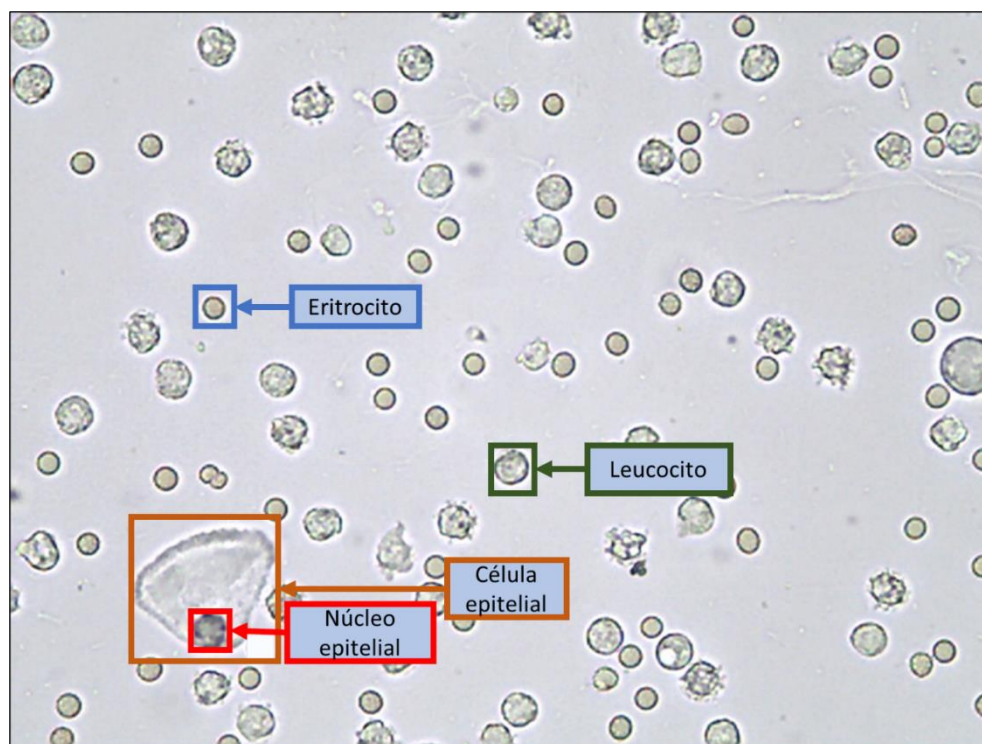


Figura 2.1 Imagen de base de datos USD. [28]

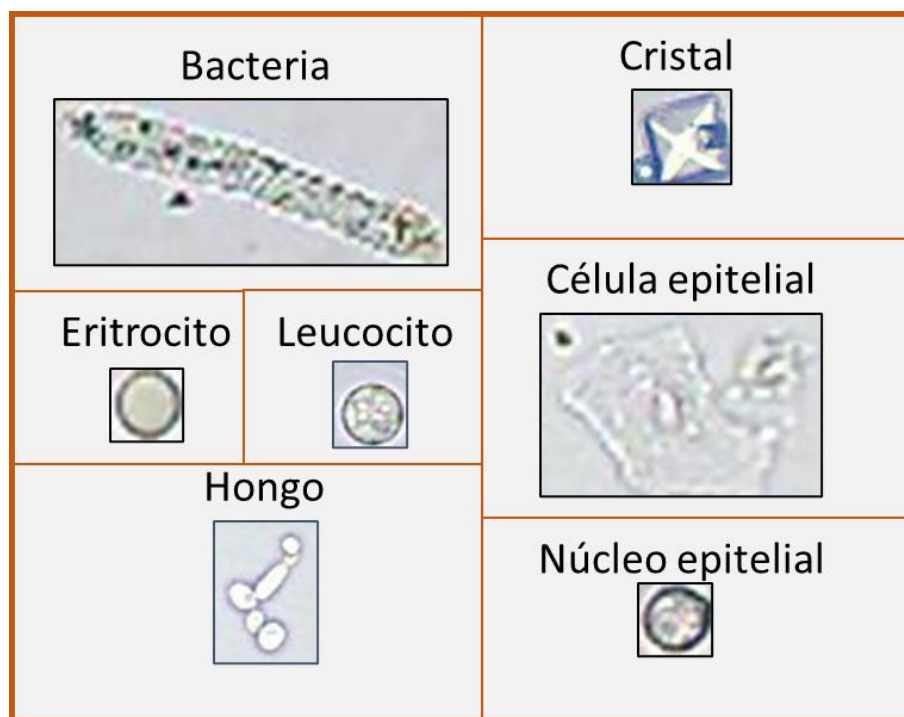


Figura 2.2 Muestras recortadas por clase de USD.

Tabla 2.1 Cantidad de muestras por clase en base de datos USD_2.

Clase	Cantidad de imágenes
Bacterias	3,662
Cristales	1,664
Células epiteliales	6,175
Núcleos epiteliales	687
Eritrocitos	21,815
Leucocitos	6,169
Hongos	2,083
Total	42,255

2.2. Raabin-wbc

Kouzehkanan et al. presentan en [31] la base de datos Raabin-wbc, que contiene imágenes de glóbulos blancos en sangre periférica. Cuenta con 14,514 imágenes recortadas de leucocitos de distinto tipo: basófilos, eosinófilos, linfocitos, monocitos y neutrófilos. En la Figura 2.3 se muestran ejemplos de cada clase contenida en la base de datos, mientras que en la Tabla 2.2, se presenta la cantidad de muestras por clase. Se puede observar que existe un desbalance de clases, teniendo una cantidad de basófilos escasa en comparación con el resto de muestras. Esto se debe a la naturaleza misma de estos, así como en la frecuencia en que se encuentran en el organismo. La base de datos Raabin-wbc es muy utilizada en el estado del arte debido a su disponibilidad, así como las diferencias en las muestras de coloración, orientación y ruido, las cuales permiten obtener mejores desempeños en etapas de prueba de modelos profundos.

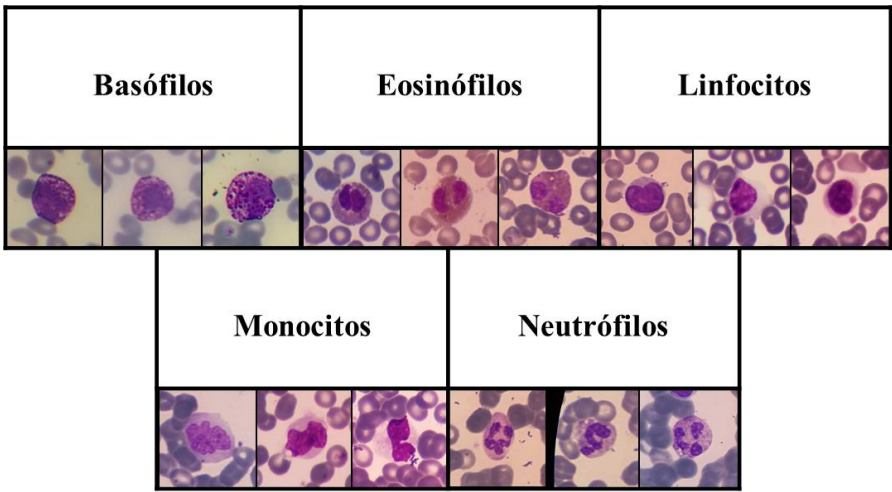


Figura 2.3 Muestras separadas de cada clase en base de datos Raabin-wbc.

Tabla 2.2 Cantidad de muestras por clase en base de datos Raabin-wbc.

Clase	Cantidad de imágenes
Basófilos	301
Eosinófilos	1,066
Linfocitos	3,461
Monocitos	795
Neutrófilos	8,891
Total	14,514

2.3. Bacteria and Yeast Real Image (BYRI)

La base de datos BYRI fue presentada por Salazar E. en [32], la cual contiene imágenes de bacterias y levaduras encontradas en cultivos de microorganismos. Se incluyen 3 clases de microorganismos: candida albicans, escherichia coli y staphylococcus aureus. La base de datos cuenta con 200 muestras de cada clase y en las Figuras Figura 2.4, Figura 2.5 y Figura 2.6 se muestran ejemplos de candida albicans, escherichia coli y staphylococcus aureus respectivamente.

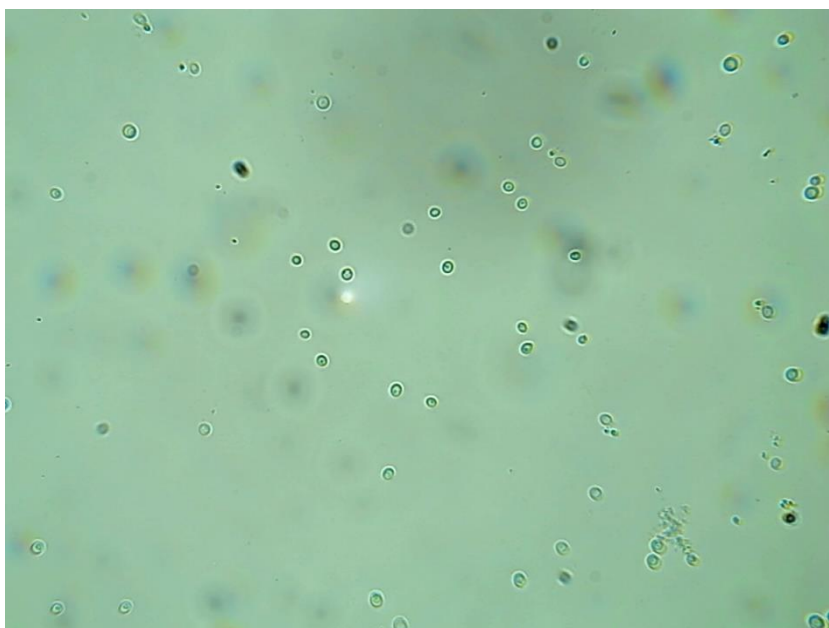


Figura 2.4 Muestra de base de datos BYRI de la clase candida albicans.

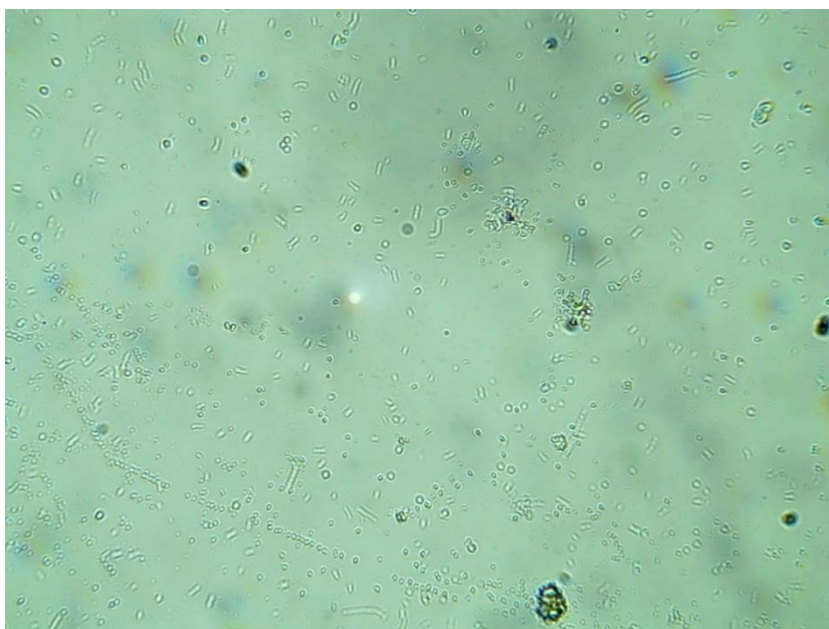


Figura 2.5 Muestra de base de datos BYRI de la clase candida albicans.

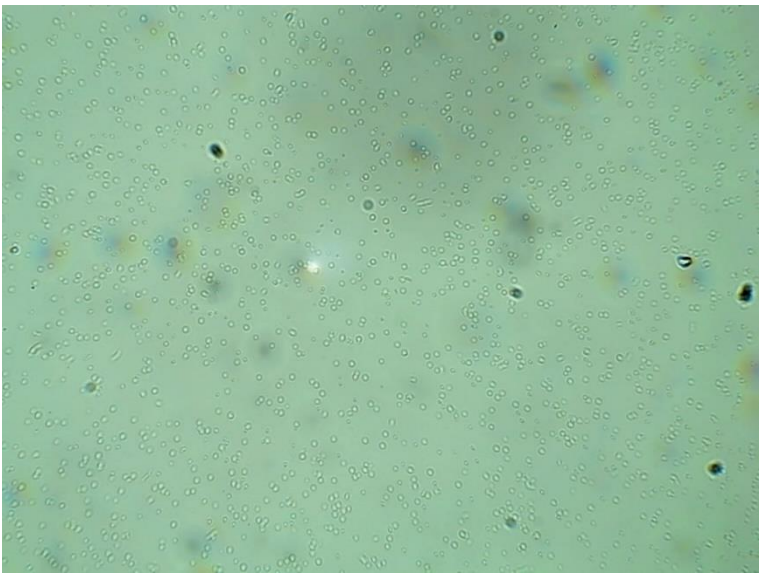


Figura 2.6 Muestra de base de datos BYRI de la clase candida albicans.

Como se puede observar, cada imagen tiene varias muestras de microorganismos, por lo cual se realizó un recorte de las mismas mediante los *ground truth* de la base de datos. Con esta información, se obtuvieron los centroides de cada objeto, para posteriormente realizar el recorte en un espacio de 20x20 pixeles alrededor del mismo. En la Tabla 2.3, se indica la cantidad de muestras por clase obtenidas para un total de 224,988 muestras.

Tabla 2.3 Cantidad de muestras por clase en base de datos BYRI.

Candida albicans	Escherichia coli	Staphylococcus aureus	Total
28,117	85,055	111,816	224,988

Es importante señalar que cada muestra obtenida fue redimensionada mediante interpolación bicúbica a un tamaño de 50x50 pixeles [33]. En la Figura 2.7 se observan ejemplos de las muestras de cada clase recortadas.

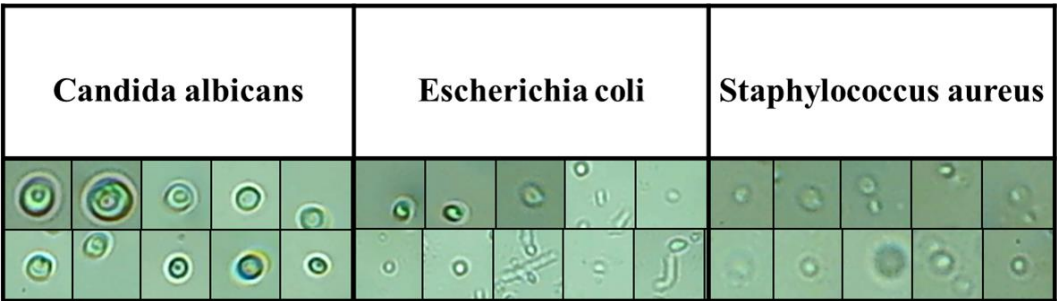


Figura 2.7 Muestras separadas de cada clase en base de datos BYRI.

2.4. Estado del arte

En la literatura se encuentran diversos trabajos sobre procesamiento de imágenes médicas con redes neuronales profundas para tareas de reconocimiento, segmentación semántica, identificación y clasificación. A continuación, se presentan algunos trabajos relevantes relacionados al procesamiento de imágenes médicas. Los modelos presentados muestran el uso de aprendizaje profundo para realizar distintas tareas relacionadas con análisis médico.

2.4.1. Modelos de procesamiento de imágenes médicas

Las siguientes investigaciones abordan temas de visión por computadora y procesamiento de imágenes, destacando enfoques para segmentación semántica, aplicaciones médicas, detección de tumores cerebrales, clasificación de células sanguíneas y otros. Se utilizan diversas técnicas, como redes totalmente convolucionales, transferencia de aprendizaje y métodos de registro. Los resultados indican avances significativos en estas áreas.

Shelhamer et al. presentan en [34] un método para segmentación semántica mediante redes totalmente convolucionales (FCN). Para ello, se utilizan las CNN AlexNet, Visual Geometry Group (VGG) y GoogLeNet con transferencia de aprendizaje y las bases de datos Pattern Analysis Statistical modelling and Computational Learning Visual Object Classes (PASCAL VOC), New York University Depth en su segunda versión (NYUDv2) y Pattern Analysis Statistical modelling and Computational Learning Context (PASCAL-Context, por sus siglas en inglés). Los resultados presentados muestran una mejora relativa de 30% de exactitud en la segmentación, mientras que la inferencia tomó una décima de segundo. Por su parte Ji et al. proponen en [35] una red neuronal paralela completamente convolucional para segmentación semántica. Para ello, se integra una red de detección de bordes holísticamente anidada para capturar información del borde de la imagen. La red se prueba con las bases de datos PASCAL VOC 2012, PASCAL-Context y Cityscapes. Los resultados presentados muestran un mejor desempeño en comparación con los obtenidos mediante los métodos DeepLab, red completamente convolucional-8s/32s, red de segmentación (Segmentation Network o SegNet), Pyramid Scene Parsing Network (PSPNet) y U-Net, ya que la red propuesta alcanza un 66.76% en la métrica de intersección sobre la unión (mean IoU). Este resultado representa una mejora de hasta un 14.34% en la tarea de segmentación semántica. En otro sentido Wen et al. proponen en [36] una red generativa adversaria profunda para tareas de segmentación semántica mediante un método de aprendizaje débilmente supervisado. Para ello, se utiliza una red generativa adversaria, así como un método de aprendizaje débilmente supervisado y técnicas de transferencia de aprendizaje. Los resultados presentados muestran ventajas del método propuesto en comparación con la generalización de la segmentación semántica. Así mismo, el método disminuye significativamente la dependencia con datos etiquetadas asegurando la precisión de la red.

CAPÍTULO II. ANTECEDENTES

Cui et al. proponen en [37] un método basado en redes neuronales bayesianas completamente convolucionales para registro de imágenes médicas no rígidas. Para ello, se utiliza una red neuronal bayesiana completamente convolucional y la base de datos Mindboggle 101. Los resultados presentados muestran una mejora promedio de 2.8% respecto al método de registro rápido de imagen (FAIM, por sus siglas en inglés) en la métrica DICE. Además de esto, el método propuesto es capaz de estimar la incertidumbre de los resultados.

Hu et al. proponen en [38] un algoritmo de procesamiento de imágenes cerebrales para predicción de enfermedades. Para ello, se utiliza un sistema basado en agrupamientos difusos y Hierarchical Probabilistic U-Net (HPU-NET) e imágenes de resonancia magnética cerebrales recolectadas de un hospital. Los resultados presentados muestran que el algoritmo propuesto tiene mayor cantidad de nodos, menor consumo de energía y una mayor estabilidad frente a cambios que otros modelos en las mismas condiciones. Así mismo, el algoritmo alcanzó un 93.6% de precisión en la predicción.

Gobhinath et al. proponen en [39] un método de detección y clasificación de tumores cerebrales mediante procesamiento de imágenes. Para ello, se utiliza la transformada wavelet para construir características y análisis de componentes principales (PCA, por sus siglas en inglés) para minimizar las dimensiones de las características. Así mismo, se utilizan Máquinas de Vectores de Soporte con Kernel (KSVM) clasificación, *K-fold* para validación cruzada, y la base de datos con 240 imágenes de resonancia magnética. Los resultados presentados muestran un máximo de precisión del 98.17%, y un tiempo promedio de 0.0511 segundos para la predicción.

Banik et al. presentan en [40] un modelo de red neuronal convolucional para clasificar imágenes de glóbulos blancos en sangre. Para ello, se utiliza una arquitectura de 5 capas convolucionales, 3 max-pooling y una red totalmente conectada con una capa oculta, además de un modelo combinado de red neuronal convolucional y recurrente para fines de comparación. Los resultados presentados muestran que se obtuvo una precisión de entre 90.39% y 90.79%.

Van Opbroek et al. proponen en [41] un método para minimizar la discrepancia máxima media entre datos de entrenamiento y prueba de imágenes de lesiones cerebrales, lesiones en materia blanca y segmentación de hipocampo. Para ello, se utiliza aprendizaje de kernel, máquinas de vectores de soporte, clasificación por árboles aleatorios, así como cinco bases de datos distintas. Estas bases de datos son dos del estudio de escaneo de Rotterdam, una del reto MSBrainS y dos del repositorio de segmentación cerebral de Internet o IBSR, por sus siglas en inglés. Los resultados presentados muestran una mejora significativa en los tres casos de estudio, teniendo una precisión máxima de 87.14% en lesiones cerebrales, 92.99% en lesiones de materia blanca y 84.6% en segmentación de hipocampo.

Zhou et al. proponen en [42] un método basado en máquinas de vectores de soporte y una red de resolución adaptativa (RANet, por sus siglas en inglés), para clasificación de imágenes de tumores. Para ello, utilizan un

detector de anomalías con máquinas de vectores de soporte y la red mencionada, además de la base de datos BACH del 2018. Se obtuvieron resultados en exactitud para imágenes de prueba de 97.75%, además de reducir el tiempo computacional aproximadamente en 50% respecto a redes como ResNet y DenseNet.

Wang et al. presentan en [43] un modelo basado en una red neuronal profunda con mecanismos de atención para segmentación de imágenes de ultrasonidos. Para esto, se utiliza una red neuronal convolucional con transformer (ConvTrans-Net) y la base de datos abierta de imágenes de ultrasonido mamario (Breast Ultrasound Images Dataset). Los resultados mostrados presentan una precisión del 85.17% y un 85.21% en la métrica Jaccard, siendo superiores a lo obtenido mediante EfficientNet.

2.4.2. Modelos de procesamiento de imágenes de sedimento urinario

Shen et al. proponen en [44] un método para reconocimiento de componentes encontrados en muestras de sedimento urinario. Para ello, se utiliza máquinas de vectores de soporte (SVM), el algoritmo de aceleración de cascada, PCA y una base de datos con 14,000 imágenes por clase. Los resultados presentados muestran una precisión en el reconocimiento alrededor al 90%, la cuál es superior a la alcanzada por otros métodos. Ji et al. proponen en [45] un método de reconocimiento basado en CNN y el algoritmo de características de área para reconocimiento de 10 clases distintas en muestras de reconocimiento urinario. Los autores propusieron su propia base de datos, la cual contiene 300,000 imágenes de sedimento urinario. Los resultados presentados muestran que se alcanzó una precisión de prueba de hasta 97%. Por su parte, Tangsuksant et al. proponen en [46] un método de segmentación y detección de glóbulos rojos y blancos en imágenes de sedimento urinario. Para ello, se utiliza una red neuronal artificial en el espacio de color de matiz, saturación y valor (HSV), operaciones morfológicas para eliminación de ruido y la transformada de Hough para la detección de círculos. Los resultados presentados muestran un porcentaje promedio de error de 5.28% en la detección de glóbulos rojos y un 8.35% en glóbulos blancos. Aunado a esto Kaul et al. proponen en [47] una técnica para segmentación de cáncer de piel y lesiones pulmonares, basada en la incorporación de atención a CNN. Para ello, se utiliza un modelo de red totalmente convolucional y una base de datos de 2750 imágenes para la segmentación de cáncer de piel y otra de 267 imágenes para el caso de segmentación de lesiones pulmonares. Los resultados presentados muestran una precisión mínima de 97.57% y una máxima de 99.81%, siendo mayor a la arquitectura U-Net (96.67% - 98.78%).

Hortinela et al. proponen en [48] un método de identificación de cristales presentes en muestras de sedimento urinario, basada en máquinas de vectores de soporte. Para ello, se utiliza una Raspberry Pi 2 con cámara para la adquisición de las imágenes, el algoritmo Adaptive Boost y máquinas de vectores de soporte. Los resultados presentados muestran una precisión de 90%.

Noothout et al. proponen en [49] un método de localización automático en imágenes médicas, basado en redes completamente convolucionales. Para ello, se utiliza una arquitectura de redes neuronales totalmente

CAPÍTULO II. ANTECEDENTES

convolucionales basada en red residual en su versión 34 (ResNet34). También, se utilizan tres bases de datos, una con escaneos de angiografía coronaria por tomografía computarizada (CCTA) en 3D, otra con resonancia magnética en 3D y otra con cefalométricas en 2D. Los resultados presentados muestran un rendimiento superior respecto otros algoritmos en el estado del arte, obteniendo una precisión promedio de 95%.

Xiang et al. proponen en [50] un método de identificación basado en redes completamente convolucionales, para reconocimiento de cristales en muestras de sedimento urinario. Para ello, se utiliza una red totalmente convolucional ResNet50, una red piramidal de características, una subred de clasificación y una base de datos obtenida del hospital Nanshan de Shenzhen. Los resultados presentados alcanzan hasta un 74% de precisión en el reconocimiento de los cristales.

Fernández et al. proponen en [51] un sistema de automatizado para reconocimiento de constituyentes en muestras de uroanálisis basado en redes neuronales artificiales. Para ello, se utiliza un microcontrolador PIC18F25J50 junto con motores de pasos y cámara para la adquisición de muestras, así como métodos de reconocimiento de bordes utilizando operadores Prewitt para mover los lentes. Por otra parte, se utiliza una red neuronal artificial creada en Matlab para el reconocimiento y conteo de los constituyentes, así como una base de datos de 100 muestras. Los resultados presentados muestran una precisión promedio de 89.5% y una velocidad promedio de 3.66 minutos, debido al tiempo empleado en la captura y reconocimiento de las muestras.

Li et al. presentan en [52] un método de clasificación de objetos en muestras de sedimento urinario, basado en aprendizaje profundo y CNN. Para ello, utilizan un modelo basado en la arquitectura LeNet-5, enfocado en el análisis morfológico de las imágenes. Se realizaron modificaciones a LeNet-5 cambiando el número de capas convolucionales. Los resultados presentados muestran una exactitud obtenida de 92.3% promedio para las distintas clases.

Li et al. presentan en [53] un método para clasificación de componentes en muestras de sedimento urinario, mediante la integración de filtros de Gabor y transformada de dispersión para análisis de textura. Por otra parte, utiliza una máquina de vectores de soporte (SVM, por sus siglas en inglés) para la clasificación, tomando las características extraídas por los filtros de Gabor y la transformada de dispersión. Los resultados presentados muestran una exactitud de 90% promedio en las clases analizadas.

Kang et al. proponen en [54] método de detección de objetos en muestras de sedimento urinario mediante CNN. Para ello, se exploran dos métodos, redes neuronales convolucionales recurrentes rápidas (Faster R-CNN) y un detector de único tiro (SSD, por sus siglas en inglés). Así mismo, se utilizó la base de datos USD, con 5376 imágenes etiquetadas y utilizando las 7 clases contenidas en la misma. Los resultados presentados muestran que la mejor exactitud obtenida fue de 84.1%, con un tiempo de procesamiento de 72 milisegundos por imagen, utilizando una GPU NVIDIA Titan X.

CAPÍTULO II. ANTECEDENTES

Li et al. presentan en [55] un método para segmentación de elementos presentes en muestras de sedimento urinario fuera de foco basado en transformadas wavelet, procesamiento morfológico y métodos de combinación. En este sentido, se utilizan las transformadas de wavelet y procesamiento morfológico para mejorar el enfoque en los elementos presentes de las muestras. Posteriormente, se realiza detección de bordes con umbrales adaptivos para finalmente realizar la segmentación de los elementos. Los resultados presentados muestran que el 97% de los elementos fueron segmentados correctamente.

3. RED NEURONAL PROFUNDA

Este capítulo presenta distintas arquitecturas de redes neuronales profundas que se reportan en la literatura como aquellas que han sido utilizadas para la tarea de clasificación de elementos presentes en muestras clínicas de laboratorio. Así mismo, se describe la composición y funcionamiento de cada una de las arquitecturas incluidas.

Con la finalidad de facilitar la presentación de las redes neuronales, se realizó un conjunto de bloques que representan las diferentes capas de ellas. Dichos conjuntos se presentan en la Figura 3.1 y para entender mejor, se recuerda al lector que algunas de las capas que ya se definieron formalmente en la Sección 1.4. Dichas capas son: la Sección 1.4: capa convolucional, capa totalmente conectada, capa de agrupamiento, normalización por lotes y función de activación. Por otro lado, el resto de las capas se definirán en las secciones 3.1, 3.2 y 3.3 según sean mencionadas.

Input layer	Capa de entrada	Bottleneck Residual	Capa de cuello de botella residual
Conv	Capa convolucional	DenseBlock	Bloque denso
Fully Connected	Capa totalmente conectada	MBConv	Bloque convolucional con cuello de botella residual móvil
Pooling	Capa de agrupamiento	Transition Layer	Capa de transición
Add	Capa de suma	Activation Function	Función de activación
Batch Normalization	Normalización por lotes	DepthWise Conv	Convolución en profundidad

Figura 3.1 Simbología de bloques a utilizar en redes.

El capítulo está organizado de la siguiente manera: en la Sección 3.1 se presenta la arquitectura MobileNet, así como su variación MobileNetV2. En la Sección 3.2 se desglosa la arquitectura DenseNet. Finalmente, en la Sección 3.3 se analiza la arquitectura EfficientNet, así como la variación EfficientNetV2.

3.1. MobileNet

MobileNet es una arquitectura de red neuronal profunda presentada en 2017 por Howard et al. en [56]. Está diseñada para ejecutarse eficientemente en dispositivos móviles y de baja potencia al utilizar convoluciones profundas separables, lo cual reduce significativamente el costo computacional. En la Figura 3.2 se muestra la arquitectura general de esta arquitectura en su segunda versión.

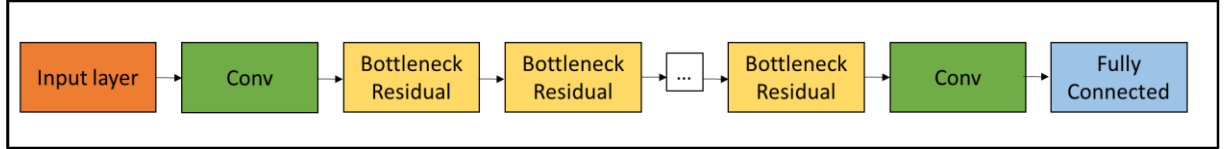


Figura 3.2 Arquitectura general de MobileNetV2.

Convolución profunda separable: divide la operación de convolución estándar definida en la ecuación (1.1) en dos etapas: una convolución espacial y una convolución en profundidad. La convolución espacial se realiza en la entrada de la imagen y se aplica un conjunto de filtros de tamaño reducido (por ejemplo, 3x3), mientras que la convolución en profundidad se aplica a cada uno de los canales de entrada y se aplica un conjunto de filtros de tamaño 1x1. Esta técnica reduce significativamente el número de operaciones y parámetros requeridos, lo que hace que la red sea más eficiente en términos de cómputo. La convolución espacial está definida por la ecuación (3.1):

$$h(i, j) = \sum_{m=1}^M \sum_{n=1}^N X(i + m, j + n) W(m, n) \quad (3.1)$$

Donde X es la salida de la capa anterior, W es el kernel aplicado y h es la salida de la capa actual en el punto dado. Así mismo, m y n son los índices correspondientes a los distintos valores tanto de la entrada de la red como del kernel, mientras que i y j son los índices referentes a la salida de la capa. La convolución en profundidad está definida por la ecuación (3.2):

$$h(i, j, k) = \sum_{m=1}^M \sum_{n=1}^N \sum_{l=1}^L X(i + m - 1, j + n - 1, l) W(m, n, l, k) \quad (3.2)$$

donde l corresponde al índice del canal de entrada y k al del canal de salida.

Capas de separación de canal: dividen los canales de entrada en grupos y aplican la convolución en profundidad por separado a cada grupo. Esto reduce aún más la cantidad de operaciones y parámetros necesarios, mientras mantiene una buena capacidad de representación [57].

Otra técnica utilizada es la capa de punto de anclaje, que se utiliza para reducir la resolución espacial de la entrada de la imagen. Esta capa aplica una convolución espacial de tamaño reducido seguida de una operación de agrupación máxima, lo que reduce la resolución espacial de la imagen sin perder demasiada información.

En 2018, Sandler et al presentaron en [58] MobileNetV2, la cual es una mejora a la arquitectura original. Una de las principales mejoras de esta versión es el uso de bloques residuales lineales invertibles (*bottleneck blocks*). Estos bloques permiten aumentar la profundidad de la red sin aumentar significativamente el número de operaciones y parámetros. También incluye una combinación de capas convolucionales normales y capas convolucionales separables, que optimiza la relación entre el costo computacional y la precisión de la red. En la Figura 3.3 se muestra la arquitectura de dicho bloque.

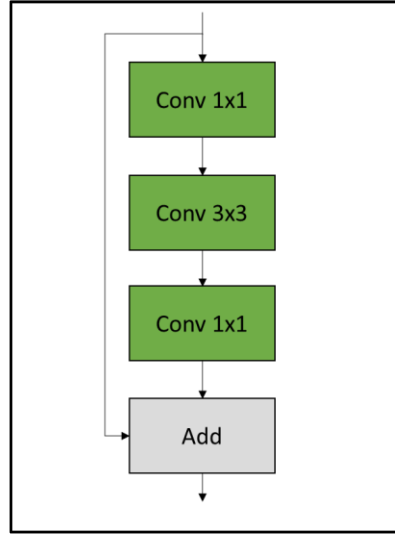


Figura 3.3 Arquitectura de bloque bottleneck.

Capa de suma: funge como punto de suma, realizando la adición correspondiente entre la salida de la última convolución y la salida del bloque anterior. Así mismo, utiliza una técnica llamada atenuación lineal de la tasa de aprendizaje (*linear decay of learning rate*), que permite que la red tenga una tasa de aprendizaje alta en las primeras etapas del entrenamiento y una tasa de aprendizaje baja en las etapas posteriores. Esto ayuda a evitar que la red se atasque en óptimos locales y permite que la red alcance una precisión óptima de clasificación y está definida por la ecuación (3.3).

$$lr_n = lr_0 \left(1 - \frac{n}{N}\right) \quad (3.3)$$

Donde lr_0 es la tasa de aprendizaje inicial y lr_n la tasa de aprendizaje de la iteración actual. Por otro lado, n y N representan el número de iteración actual y el número total de iteraciones respectivamente.

Finalmente, MobileNetV2 utiliza una técnica de normalización de lotes ajustable (*adjustable batch normalization*), que permite ajustar el tamaño del lote de entrada en tiempo de ejecución sin requerir un entrenamiento previo. Para ello, el modelo puede ajustar los parámetros γ y β de la ecuación (1.7) durante el entrenamiento del modelo para adaptarse a cambios en la distribución de los datos.

3.2. DenseNet

DenseNet fue presentada por Huang et al. en [59] en 2016 y es una arquitectura de red neuronal profunda que se caracteriza por su alta eficiencia y su capacidad para evitar el problema de la desaparición del gradiente. En la Figura 3.4 se muestra la arquitectura general de esta.

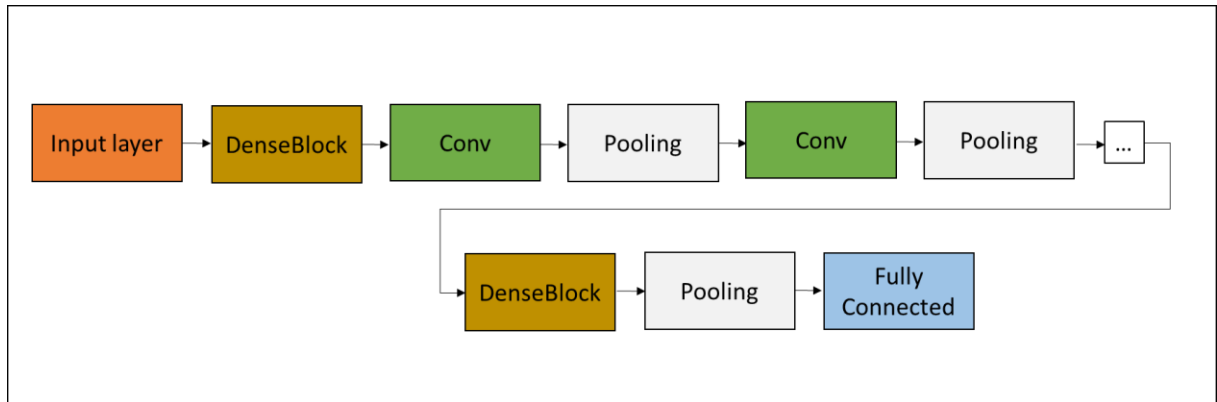


Figura 3.4 Arquitectura general de DenseNet.

Bloque denso: en lugar de tener capas completamente conectadas, como en las redes neuronales tradicionales, DenseNet utiliza bloques densamente conectados (*densely connected blocks*). En estos bloques, cada capa recibe como entrada las características de todas las capas anteriores, lo que significa que todas las capas están conectadas directamente entre sí.

Este tipo de conexión tiene varias ventajas. En primer lugar, permite que la información fluya a través de la red de manera más eficiente, ya que cada capa puede recibir información de todas las capas anteriores. En segundo lugar, hace que la red sea más fácil de entrenar y menos propensa a la desaparición del gradiente, ya que cada capa recibe información directa de todas las capas anteriores.

Los bloques densamente conectados de DenseNet se dividen en dos partes: la de características y la de clasificación. La parte de características se compone de varias capas convolucionales, que extraen las características de la imagen de entrada. La parte de clasificación se compone de una capa de agrupamiento global seguida de una capa completamente conectada que produce la salida de la red.

Además de esto, utiliza de capas de transición entre los bloques densamente conectados. Estas capas reducen el número de canales de características y el tamaño de la imagen de salida antes de pasar a la siguiente capa de bloques densamente conectados, lo que reduce la complejidad computacional y evita el sobreajuste. En la Figura 3.5 se muestra la arquitectura de los bloques densamente conectados.

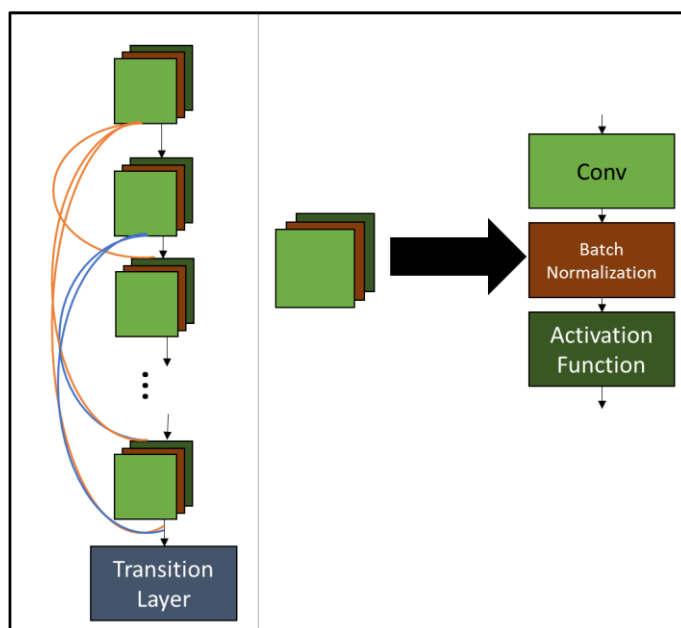


Figura 3.5 Arquitectura de bloque denso.

Capa de transición: las capas de transición están compuestas por una capa de normalización definida por la ecuación (1.7), una capa de activación ReLU según la ecuación (1.2), una capa de convolución de 1x1 definida en la ecuación (1.1) y una capa de agrupamiento descrita por la ecuación (1.3). Esto se puede observar de manera gráfica en la Figura 3.6.

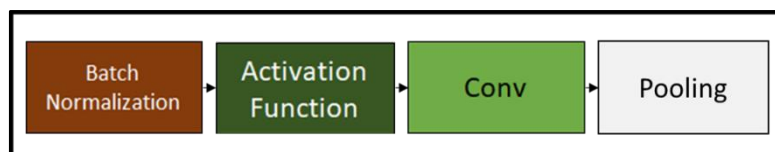


Figura 3.6 Capa de transición de bloque denso.

Existen distintas versiones de DenseNet, según su complejidad. DenseNet201 es una versión más grande y profunda de la arquitectura DenseNet original. Mientras que la arquitectura original utiliza cuatro bloques densamente conectados, DenseNet201 utiliza tres bloques densamente conectados más grandes y profundos, por lo que tiene una mayor capacidad para extraer características complejas y sutilmente detalladas de las imágenes de entrada. Además, utiliza capas de transición con una tasa de compresión menor que la arquitectura DenseNet original, lo que significa que conserva más información y características en las capas de transición.

En términos de rendimiento, DenseNet201 suele ser más preciso que la arquitectura DenseNet original, pero también requiere más recursos computacionales para entrenar y ejecutar debido a su mayor tamaño y profundidad.

3.3. EfficientNet

En 2019, Tan et al. presentaron en [60] una arquitectura de red neuronal profunda llamada EfficientNet, cuyo funcionamiento se basa en una estrategia de escalado compuesto que busca equilibrar el aumento del tamaño del modelo con la eficiencia computacional. La idea es escalar el ancho, la profundidad y la resolución de la red de manera proporcional para obtener un modelo que procese la mayor cantidad de características, pero con un aumento mínimo en los recursos computacionales requeridos. En la Figura 3.7 se muestra la arquitectura general de EfficientNet.

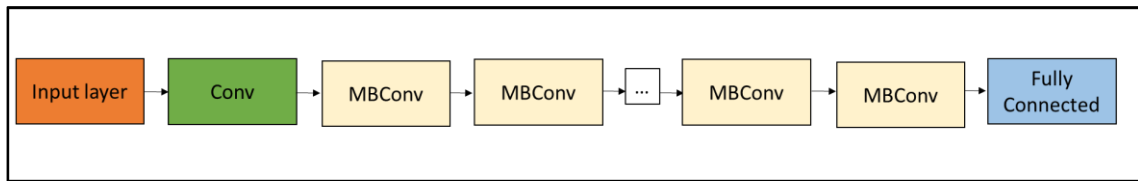


Figura 3.7 Arquitectura general de EfficientNet.

Bloques convolucionales con cuello de botella residual móviles: consisten en una combinación de capas de convolución y agrupamiento dadas por las ecuaciones (1.1) y (1.3) respectivamente, así como conexiones residuales. Estos bloques (llamados MBConv) son eficientes computacionalmente y permiten que la información fluya a través de la red de manera más rápida y efectiva. En la Figura 3.8 se muestra la arquitectura de los bloques MBConv.

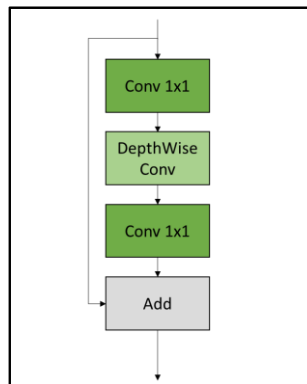


Figura 3.8 Arquitectura de bloque MBConv.

Una característica única de EfficientNet es el uso de un parámetro de escalado llamado factor de escalamiento compuesto, que se aplica a las dimensiones del modelo, incluyendo el ancho, la profundidad y la resolución. Este parámetro permite ajustar el modelo para diferentes tamaños de imágenes y recursos computacionales disponibles, lo que hace que EfficientNet sea altamente adaptable y flexible. El ancho, la profundidad y la resolución en esta arquitectura varían según la ecuación (3.4):

$$\begin{aligned} A &= \alpha^\phi \\ B &= \beta^\phi \\ \Gamma &= \gamma^\phi \end{aligned} \tag{3.4}$$

Donde A , B y Γ representan la anchura, profundidad y resolución que el modelo tendrá, mientras que α , β y γ son constantes predefinidas basadas en experimentación. Finalmente, el parámetro ϕ es el factor de escalamiento compuesto, el cual será definido por el modelo durante el entrenamiento y determina el tamaño del modelo en las distintas dimensiones.

EfficientNet ha demostrado ser una arquitectura de red neuronal eficiente y efectiva en tareas de clasificación de imágenes, logrando un rendimiento competitivo en comparación con otras arquitecturas más grandes y computacionalmente costosas.

En 2021, Tan et al. presentaron una nueva versión de EfficientNet en [61], llamada EfficientNetV2. EfficientNetV2 es una versión mejorada de la arquitectura EfficientNet original.

La principal diferencia entre ambas es la forma en que se construyen los bloques MBConv. En EfficientNet, los bloques MBConv se construyen mediante una combinación de operaciones de convolución 3x3 y 1x1, mientras que en EfficientNetV2 se utilizan operaciones de convolución dada por la ecuación (1.1) con filtros de 3x3 y 5x5 en combinación con una operación de convolución con filtro de 1x1.

Además, EfficientNetV2 utiliza una nueva técnica de regularización llamada profundidad estocástica, que consiste en eliminar aleatoriamente algunas capas durante el entrenamiento. Esto permite que la red se adapte mejor a diferentes tamaños de lotes de entrenamiento y mejora el rendimiento de la red en conjuntos de datos pequeños.

Otra diferencia importante es que utiliza una estrategia de escalado uniforme en lugar de la estrategia de escalado compuesto utilizada en la arquitectura original. Esto significa que los modelos EfficientNetV2 se escalan de manera uniforme en todas las dimensiones, incluyendo la profundidad, el ancho y la resolución, lo que permite una mayor simplicidad y flexibilidad en la construcción de modelos.

En general, EfficientNetV2 se considera una mejora significativa sobre la arquitectura original EfficientNet, con una mayor eficiencia computacional, una mejor capacidad de generalización y un entrenamiento más rápido.

4. METODOLOGÍA DE DISEÑO DE ARQUITECTURA

En el presente capítulo, se describe la metodología de experimentación de esta tesis para proponer un modelo novedoso de red neuronal profunda para procesamiento de imágenes clínicas. Primero, se realizó un análisis de la base de datos USD_2 para determinar el tamaño de las imágenes de entrada. Posteriormente, se realizaron pruebas con las arquitecturas MobileNet, DenseNet y EfficientNet. Las pruebas consistieron en implementar los modelos con tres clases que son la clasificación de bacterias, eritrocitos y células epiteliales. Dichas clases se implementaron para visualizar las activaciones de estos para analizar su funcionamiento. También, se realizó un análisis del comportamiento estadístico de los pesos en las distintas capas.

Al realizar los análisis anteriores, se optó por proponer una variante de la red EfficientNet que se denominó EfficientNet de baja profundidad (LDE-Net). LDE-Net se comparó con MobileNetV2, EfficientNetV2 y DenseNet. Una vez obtenidos resultados de las pruebas con tres clases (bacterias, eritrocitos y células epiteliales), se decidió aumentar la cantidad de clases a 7, incluyendo todas las presentes en la base de datos USD_2, utilizando para ello únicamente la arquitectura LDE-Net, ya con esta se obtuvieron los mejores resultados para 3 clases. Se hicieron pruebas con 5 y 7 clases, y se diseñó una nueva arquitectura, depurando aún más capas redundantes, la cual lleva por nombre LDE-NetV2. Finalmente, se decidió probar el desempeño del modelo LDE-NetV2 para la tarea de clasificación en muestras de imágenes médicas, distintas a USD, las cuáles fueron Raabin-wbc y BYRI.

Este capítulo está organizado de la siguiente manera: la Sección 4.1 describe el análisis de la base de datos USD y el proceso de homogeneización de imágenes. La Sección 4.2 detalla las pruebas preliminares realizadas con las redes MobileNet, DenseNet y EfficientNet. La Sección 4.3 incluye el proceso de visualización y análisis de las activaciones en distintas capas. La Sección 4.4 menciona el análisis del comportamiento estadístico de los pesos en las capas. La Sección 4.5 describe la selección de la arquitectura EfficientNet como modelo inicial para el diseño de la arquitectura LDE-Net. La Sección 4.6 detalla las pruebas realizadas con distintos optimizadores y arquitecturas. La Sección 4.7 describe el proceso de ampliación de clases a analizar, y el diseño de la arquitectura LDE-NetV2. Finalmente, la Sección 4.8 menciona el uso de LDE-NetV2 para clasificación de microorganismos presentes en muestras de frotis sanguíneo y cultivos de bacterias y levaduras.

4.1. Análisis de la base de datos y homogeneización de imágenes

Como se menciona en la Sección 2.1, la base de datos USD contiene muestras de distintas clases de componentes, por lo cual es necesario recortar cada uno de los sedimentos por separado. Una vez realizado el recorte, las imágenes obtenidas tienen distinta dimensionalidad, debido a la naturaleza misma de los elementos. Los modelos de aprendizaje profundo utilizados tienen una capa de entrada de tamaño definido, es decir, requiere que las entradas sean de la misma dimensión. Es por esto que se optó por establecer un tamaño de imagen al cual redimensionar las distintas muestras, para ser procesadas por los modelos profundos.

Para ello, se realizó un análisis estadístico de las muestras que indicara un tamaño adecuado de las imágenes que altere lo menos posible la morfología de los objetos de interés. Este análisis se basó en obtener media, desviación estándar, máximos y mínimos de los tamaños de los objetos contenidos en las muestras. Como conclusión de dicho análisis, se decidió utilizar para las imágenes es de 100x100 píxeles. Con esto, se redimensionaron todas las imágenes a 100x100 mediante interpolación bicúbica. A la base de datos generada con este proceso se le llamó USD_2

4.2. Pruebas preliminares con arquitecturas de modelos

Se realizó la implementación de tres arquitecturas distintas: MobileNetV2, EfficientNetV2L y DenseNet201. Como se mencionó en el capítulo 3, estas arquitecturas tienen un comportamiento distinto en la forma de analizar los datos de entrada, además de una variación importante en el número de parámetros, lo cual se traduce en carga computacional. MobileNetV2 tiene 3.5 millones de parámetros, DenseNet201 cuenta con 20.2 millones y EfficientNetV2L utiliza 119 millones.

La implementación se realizó utilizando modelos entrenados previamente con la base de datos ImageNet [62], la cual consta de más de un millón de imágenes de mil clases distintas. Las redes fueron entrenadas con los mismos hiperparámetros definidos para una justa comparación. Se utilizó la entropía categórica cruzada como función de pérdida, que está definida por la ecuación (4.1).

$$L_{CE} = -\sum_{i=1}^n t_i \log(p_i) \quad (4.1)$$

donde t_i es el objetivo de la inferencia de la i ésima clase, p_i la probabilidad de la misma, y n es la cantidad de clases. El optimizador utilizado fue Adam [63] y se requirieron 100 épocas sin mecanismos de paro temprano, para observar el comportamiento de las redes durante el entrenamiento.

4.3. Visualización y análisis de activaciones de capas

Una vez entrenadas MobileNetV2, EfficientNetV2L y DenseNet201, se observó que el comportamiento de ellas variaba dependiendo de la arquitectura. Con la finalidad de entender dicho comportamiento, se decidió analizar de manera visual las activaciones de las distintas capas de las redes. Para realizar esto, se propagaron imágenes de entrada de las distintas clases y se tomaron salidas parciales de los modelos. Como ejemplo, se muestran las activaciones derivadas de la imagen nl00030-cast_1 que se muestra en la Figura 4.1.



Figura 4.1 Imagen nl00030-cast_1

Las Figuras Figura 4.2, Figura 4.3 y Figura 4.4, se muestran las activaciones obtenidas de las capas 2, 8 y 36 en las arquitecturas MobileNetV2, DenseNet201 y EfficientNetV2. Nótese que en las primeras capas se tienen comportamientos similares en las distintas redes. Sin embargo, al avanzar por la red, cada capa comienza a extraer características distintas. Así mismo, en capas más profundas se obtienen características más abstractas.

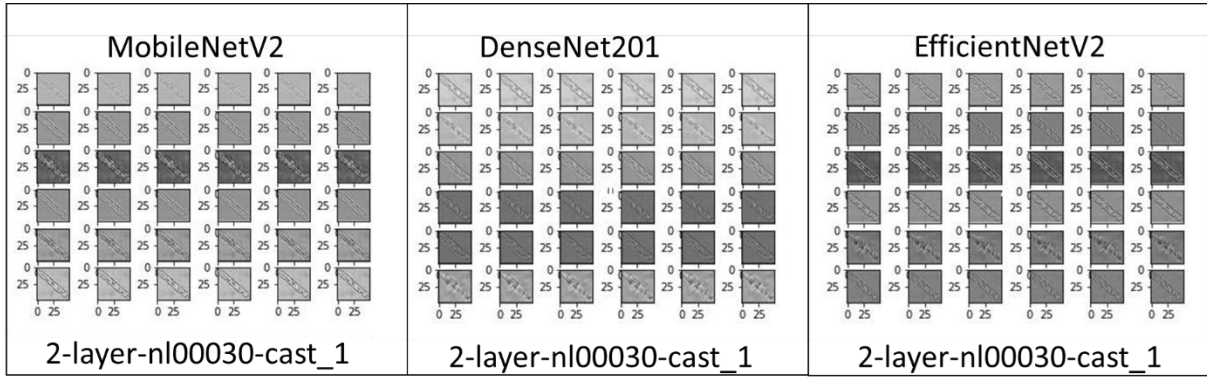


Figura 4.2 Activaciones de la capa 2 en las distintas arquitecturas.

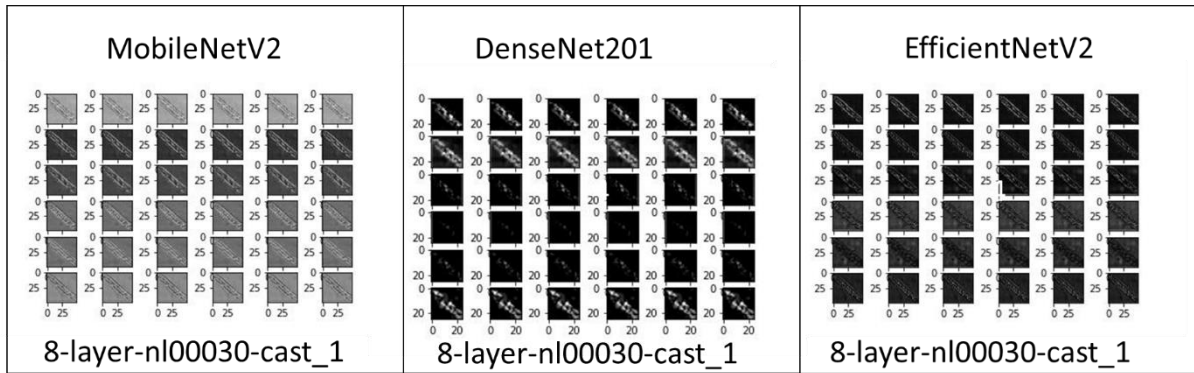


Figura 4.3 Activaciones de la capa 8 en las distintas arquitecturas.

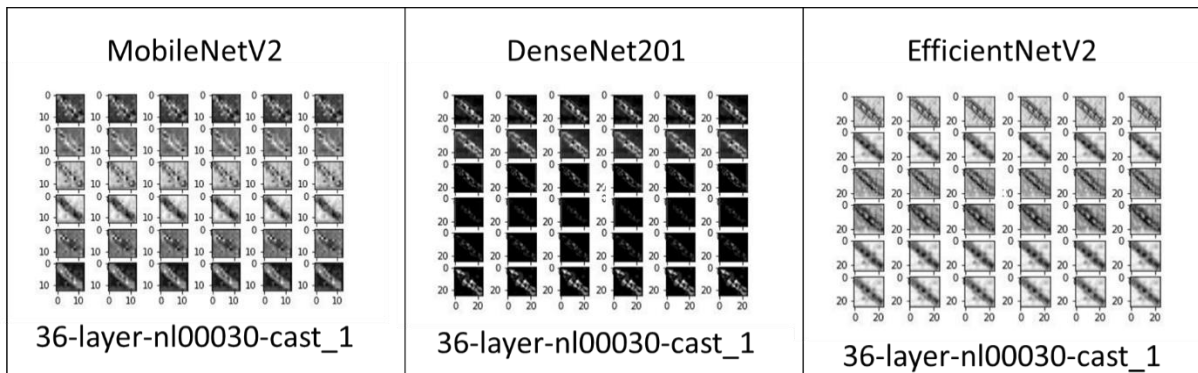


Figura 4.4 Activaciones de la capa 36 en las distintas arquitecturas.

En general, se observó redundancia en algunas capas, tal como se puede observar en las Figuras Figura 4.5 y Figura 4.6. Posteriormente se comentará más al respecto. Esta redundancia se debe a que la capacidad del modelo es demasiado grande, por lo que no todas las capas aprenden características únicas e importantes de las imágenes analizadas, repitiéndose patrones en las capas. Esto puede afectar generando sobreajuste, ya que, al tener una capacidad demasiado grande, el modelo es capaz de memorizar las muestras de entrenamiento, lo que dificultará la clasificación en muestras no conocidas.

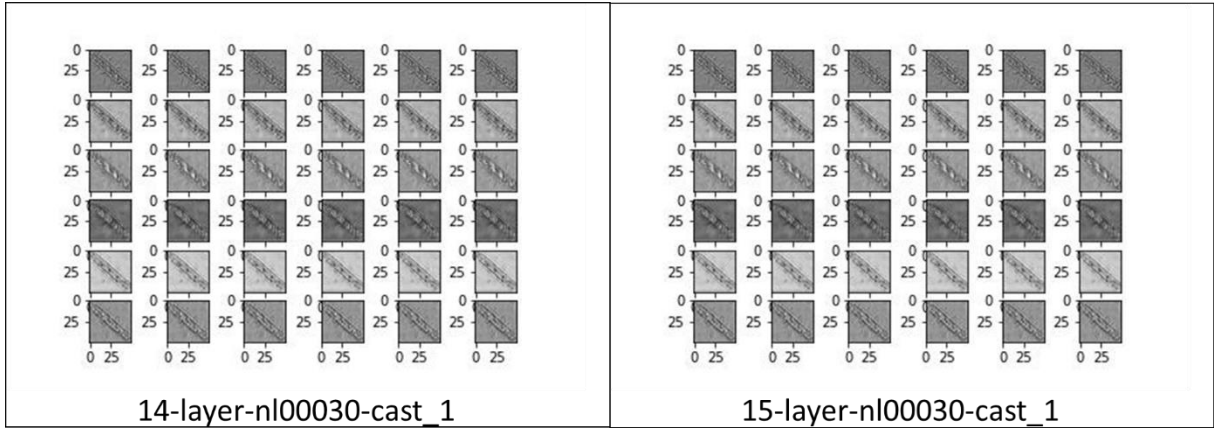


Figura 4.5 Activaciones en las capas 14 y 15 en EfficientNetV2L.

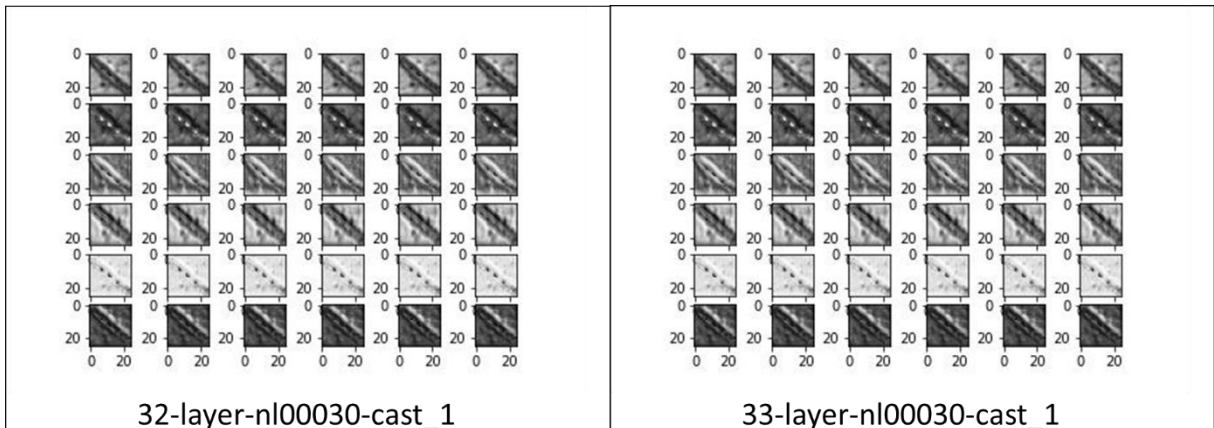


Figura 4.6 Activaciones en las capas 32 y 33 en EfficientNetV2L.

4.4. Comportamiento estadístico de los pesos en las capas

Para comprender el a mayor profundidad el comportamiento de los modelos, se calcularon la media, desviación estándar y entropía de los pesos en las capas, y se buscó una posible relación con el desempeño. En la Figura 4.9 se muestran de manera gráfica los distintos valores estadísticos, así como la exactitud a través de un entrenamiento de 100 épocas.

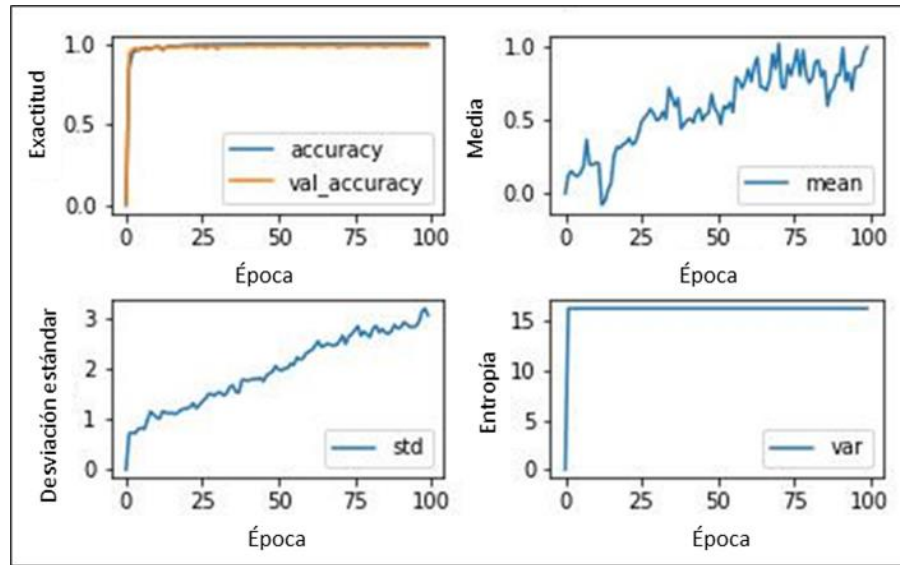


Figura 4.7 Gráficas de exactitud, media, desviación estándar y entropía de los pesos, en la capa 14 de EfficientNetV2L.

Cómo se puede observar, la entropía y exactitud muestran un comportamiento similar, lo cual puede ser indicador de una posible relación entre ambas métricas ya que se puede observar que ambas curvas son semejantes en todas las épocas, mientras que la media y desviación estándar muestran un aumento progresivo, lo que indica que conforme se realiza el entrenamiento, los valores de los filtros aumentan, sin embargo, esto no repercute directamente en una mejor exactitud.

4.5. Selección de la arquitectura EfficientNet y diseño de LDE-Net

Al analizar el desempeño obtenido en las distintas épocas de entrenamiento para las distintas redes, se definió que EfficientNetV2L era la mejor opción para el diseño del modelo debido a sus desempeños altos y estables, así como la convergencia del mismo desde épocas tempranas del entrenamiento. Esto se verá con más detalle en el capítulo 6.

Cómo se mencionó en el capítulo 4.3, se encontró redundancia en las capas, con lo cual se optó por eliminar las que se encontrasen en dicha situación. El proceso de depuración se realizó eliminando las capas cuyas activaciones fueran similares a las de las capas siguientes. Para ello se definió un umbral de cambio de 0.2 en la desviación estándar promedio entre capas, es decir, en bloques donde la desviación estándar de las activaciones fuese mayor o menor en 0.2 al bloque MBConv anterior, se toma como redundante. De manera visual, la disminución del modelo se presenta la diferenciación en la Figura 4.8.

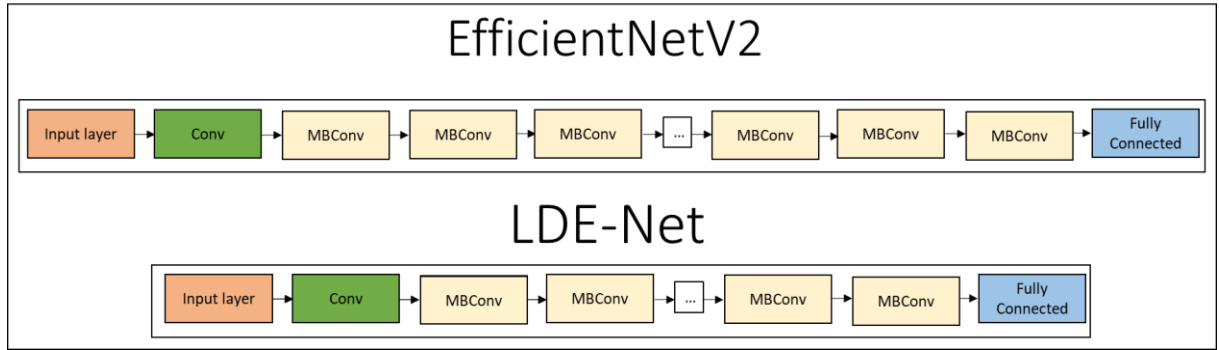


Figura 4.8 Diferencia entre EfficientNetV2 y LDE-Net.

Una vez hecho esto, se obtuvo un modelo con menor profundidad el cuál denominamos LDE-Net. Dicho modelo tiene únicamente 5.3 millones de parámetros, en comparación con los 119 millones de EfficientNetV2L, con lo cual se reduce la carga computacional de este.

Tabla 4.1 Comparación entre EfficientNetV2L y LDE-Net.

Arquitectura	Capas	Cantidad de parámetros
EfficientNetV2L	781	119 millones
LDE-Net	273	5.3 millones

4.6. Pruebas con diferentes optimizadores y arquitecturas

Una vez diseñada la arquitectura LDE-Net, se realizaron pruebas con distintas combinaciones de optimizador y arquitectura. Dichas pruebas consistieron en implementar los optimizadores RMSProp, Adam y NAdam con las redes MobileNetV2, EfficientNetV2L, DenseNet201 y LDE-Net.

El optimizador de gradiente estocástico descendiente (SGD, por sus siglas en inglés), se rige en base al siguiente algoritmo:

Algoritmo SGD

Requiere: Razón de aprendizaje ϵ_k

Requiere: Parámetro inicial θ

Mientras no se cumplan los criterios de paro:

Obtener el lote de m ejemplos del conjunto de entrenamiento $\{x^{(1)}, \dots, x^{(m)}\}$ con las etiquetas $y^{(i)}$ correspondientes.

Calcular el gradiente estimado: $\hat{g} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

Actualizar parámetros $\theta \leftarrow \theta - \varepsilon \hat{g}$

Donde m es el número de ejemplos en el lote utilizado, θ los parámetros a actualizar y $L(f(x^{(i)}; \theta), y^{(i)})$ la función de pérdida utilizada.

En general, los demás optimizadores siguen un esquema semejante a este algoritmo, cambiando los parámetros a actualizar y la regla de aprendizaje (es decir, la manera en que son actualizados los mismos). Para el optimizador de propagación de raíz cuadrada media (RMSProp, por sus siglas en inglés), el algoritmo es el siguiente:

Algoritmo RMSProp

Requiere: Razón de aprendizaje ε_k , razón de decaimiento ρ

Requiere: Parámetro inicial θ

Inicializar variables de acumulación $r = 0$

Mientras no se cumplan los criterios de paro:

Obtener el lote de m ejemplos del conjunto de entrenamiento $\{x^{(1)}, \dots, x^{(m)}\}$ con las etiquetas $y^{(i)}$ correspondientes.

Calcular el gradiente estimado: $\hat{g} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

Calcular el acumulado del gradiente cuadrático: $r \leftarrow \rho r + (1 - \rho) \hat{g} \odot \hat{g}$

Calcular la actualización de los parámetros: $\Delta\theta = -\frac{\varepsilon}{\sqrt{\delta + r}} \odot \hat{g}$

Actualizar pesos $\theta \leftarrow \theta + \Delta\theta$

Como se puede observar, a diferencia del optimizador SGD, en RMSProp se utiliza no únicamente el valor actual del gradiente, sino también el acumulado de los gradientes cuadráticos en iteraciones pasadas del entrenamiento. Así mismo, se incluye un parámetro de decaimiento ρ para da menor peso al valor de la acumulación pasado. Así mismo, se tiene una constante δ que evita divisiones entre cero que tiene un valor muy pequeño, para evitar que tenga un efecto relevante al calcular la actualización de los parámetros.

Por su parte, el algoritmo de momento adaptivo (Adam) utiliza el valor del gradiente para la actualización de los pesos y un parámetro de momento utilizado para evitar que el entrenamiento quede atrapado en mínimos locales. El algoritmo de entrenamiento del optimizador Adam es el siguiente:

Algoritmo Adam

Requiere: Razón de aprendizaje ε_k , razones de decaimiento ρ_1 y ρ_2

Requiere: Parámetro inicial θ

Inicializar variables para primer y segundo momento $s = 0, r = 0$

Inicializar variable de tiempo $t = 0$

Mientras no se cumplan los criterios de paro:

Obtener el lote de m ejemplos del conjunto de entrenamiento $\{x^{(1)}, \dots, x^{(m)}\}$ con las etiquetas $y^{(i)}$ correspondientes.

Calcular el gradiente estimado: $\hat{g} \leftarrow +\frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

$t \leftarrow t + 1$

Actualizar estimación del primer momento: $s \leftarrow \rho_1 s + (1 - \rho_1) \hat{g}$

Actualizar estimación del segundo momento: $r \leftarrow \rho_2 r + (1 - \rho_2) \hat{g} \odot \hat{g}$

Corregir sesgo en primer momento: $\hat{s} \leftarrow \frac{s}{1 - \rho_1^t}$

Corregir sesgo en segundo momento: $\hat{r} \leftarrow \frac{r}{1 - \rho_2^t}$

Calcular el acumulado del gradiente cuadrático: $r \leftarrow \rho r + (1 - \rho) \hat{g} \odot \hat{g}$

Calcular la actualización de los parámetros: $\Delta \theta = -\frac{\hat{s}}{\sqrt{\delta + \hat{r}}}$

Actualizar pesos $\theta \leftarrow \theta + \Delta \theta$

Como se puede observar, el optimizador Adam incluye el primer y el segundo momento estadístico. Además, se realizan correcciones de sesgo en ambos momentos.

Finalmente, el optimizador Nadam sigue el mismo algoritmo de Adam, con la diferencia de utilizar el momentum de Nesterov como primer momento en el cual, en lugar de calcular el gradiente de la posición actual, se calcula el de una posición futura [64].

La combinación en que se encontró un mejor desempeño fue LDE-Net con el optimizador SGD, teniendo resultados semejantes con LDE-Net y Adam. En el capítulo 6 se ahondará en el análisis para llegar a esta conclusión.

4.7. Ampliación de clases y diseño de LDE-Net V2

Debido al alto desempeño obtenido mediante la arquitectura LDE-Net con el optimizador SGD y con Adam, se decidió aumentar el número de clases de las 3 iniciales a las 7 que comprenden la base de datos. Se realizaron pruebas con 3, 5 y 7 clases para observar el comportamiento del desempeño al agregar paulatinamente categorías a clasificar. Se observó que, al incrementar el número de clases, el desempeño disminuye, debido al aumento en la complejidad del problema a resolver.

CAPÍTULO IV. METODOLOGÍA DE DISEÑO DE ARQUITECTURA

Con esto, se analizó nuevamente la arquitectura LDE-Net, siguiendo la misma metodología comentada anteriormente: se realizó una implementación preliminar, se visualizaron las activaciones de las distintas capas y se analizaron estadísticamente los pesos de las mismas. Al realizar este análisis, se observó que la arquitectura LDE-Net seguía teniendo redundancia dentro de las activaciones, lo que indica que existen demasiados kernels para el problema a resolver, lo que conlleva el efecto de sobreajuste. En las Figuras Figura 4.9 y Figura 4.10 se puede observar dicha redundancia.

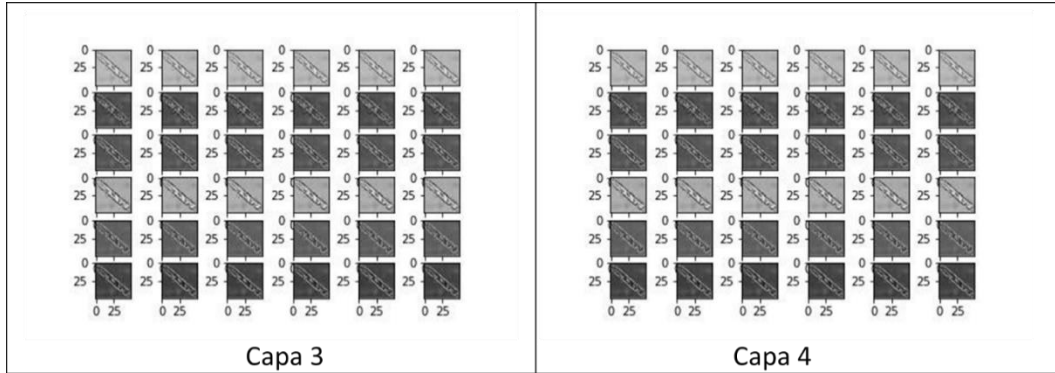


Figura 4.9 Activaciones en las capas 3 y 4 en LDE-Net.

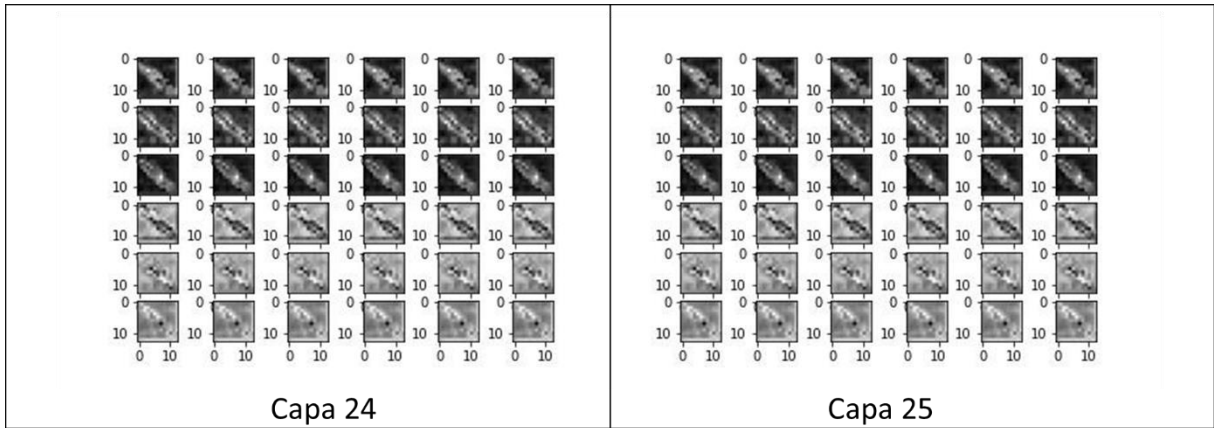


Figura 4.10 Activaciones en las capas 24 y 25 en LDE-Net.

Entendiendo lo anterior, se procedió a realizar una nueva depuración de capas, con la finalidad de reducir la cantidad de kernels y el sobreajuste. El nuevo modelo generado fue llamado LDE-NetV2, el cual reduce la cantidad de parámetros de 5.3 millones a 2.45 millones. De manera gráfica, se puede observar la reducción del modelo de la versión 2 de LDE-Net en comparación con las utilizadas anteriormente en la Figura 4.11.

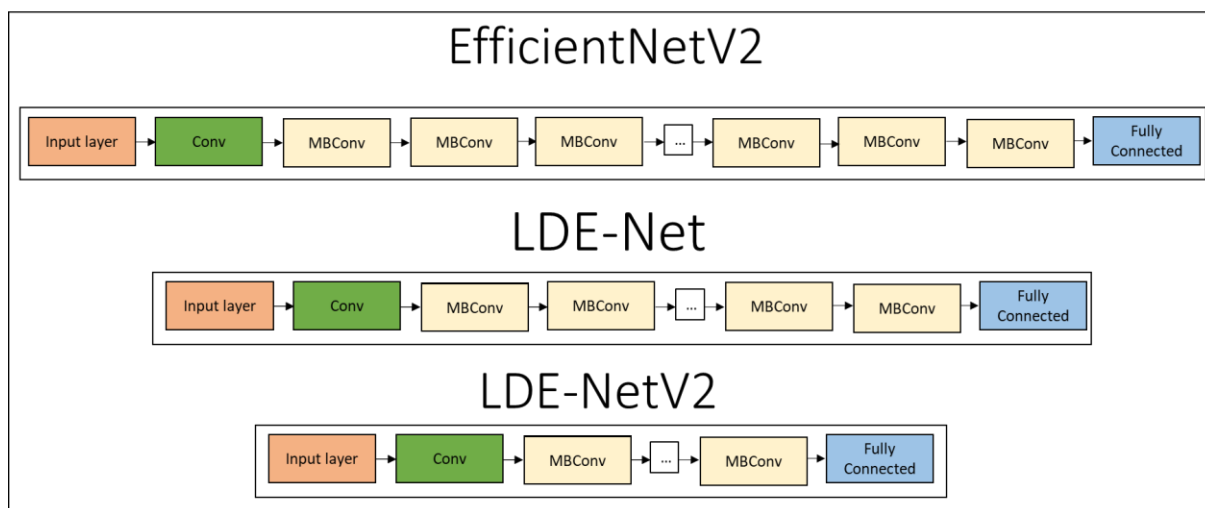


Figura 4.11 Comparación entre *EfficientNetV2*, *LDE-Net* y *LDE-NetV2*.

En la Tabla 4.2 se muestra la cantidad de capas y parámetros de las distintas versiones de LDE-Net, así como de EfficientNetV2L.

Tabla 4.2 Comparación entre *EfficientNetV2L*, *LDE-Net* y *LDE-NetV2*.

Arquitectura	Capas	Cantidad de parámetros
EfficientNetV2L	781	119 millones
LDE-Net	273	5.3 millones
LDE-NetV2	176	2.45 millones

4.8. Uso de LDE-Net V2 en otras bases de datos

El modelo LDE-NetV2, inicialmente fue concebido para muestras de sedimento urinario y se probó con la base de datos de USD. Los resultados obtenidos de LDE-NetV2 en USD motivaron la exploración de su aplicabilidad más allá de su contexto original. Por ello, se probó las LDE-NetV2 con las bases de datos Raabin-wbc y BYRI para probar la capacidad de generalización del modelo y evaluar su rendimiento en condiciones diversas. La base de Raabin-wbc fue seleccionada debido a que las muestras que contiene son representativas de una variedad importante de leucocitos, ya que poseen distintas coloraciones, ruido y las clases más comúnmente encontradas en análisis clínicos. Por su parte, la base de datos de BYRI fue seleccionada al ser generada de un proyecto en conjunto con la Universidad Autónoma de Chihuahua, además de ser de las pocas bases de datos que contienen muestras de cultivos de microorganismos en caldo nutritivo, a pesar de ser ampliamente utilizados en los análisis clínicos.

CAPÍTULO IV. METODOLOGÍA DE DISEÑO DE ARQUITECTURA

Se realizaron distintos experimentos, utilizando los algoritmos de entrenamiento descritos en el siguiente capítulo, que son: entrenamiento tradicional, reentrenamiento, transferencia de aprendizaje y meta aprendizaje. Es importante destacar que en cada uno de estos se utilizaron los mismos hiperparámetros, cambiando únicamente el algoritmo de entrenamiento.

Finalmente, se realizaron pruebas de validación cruzada para verificar la generalización de los modelos generados con la arquitectura LDE-NetV2 utilizando los distintos algoritmos de entrenamiento para las bases de datos USD_2, Raabin-wbc y BYRI.

5. ALGORITMOS DE ENTRENAMIENTO

El presente capítulo describe los distintos algoritmos de entrenamiento implementados para el entrenamiento de la arquitectura LDE-NetV2 para la clasificación de muestras de sedimento urinario, frotis sanguíneo y cultivos de bacterias y levaduras. Dicho entrenamiento se realizó de diversas vertientes: entrenamiento tradicional, cambios en el algoritmo de entrenamiento al reinicializar los kernels que presenten indicios de parálisis durante el mismo, reentrenamiento, transferencia de aprendizaje y meta aprendizaje. Cada una de estas vertientes representa enfoques distintos para abordar el entrenamiento y el ajuste de modelos de aprendizaje automático.

La comparación y evaluación de sus resultados del entrenamiento pueden brindar información valiosa sobre cuál de estas estrategias es más efectiva para la tarea y los datos específicos en cuestión. De esta manera, el propósito de este capítulo es describir cada una de estos algoritmos y está organizado de la siguiente manera: la Sección 5.1 describe el entrenamiento tradicional. La Sección 5.2 analiza la vertiente de cambios en el algoritmo de entrenamiento. La Sección 5.3 comenta la transferencia de aprendizaje. Finalmente, en la Sección 5.4 se presenta la vertiente de reentrenamiento.

5.1. Entrenamiento tradicional

Esta estrategia implica la construcción completa del modelo y el ajuste de sus parámetros a través de un proceso iterativo de optimización, como el descenso de gradiente estocástico. Este método sirve como punto de referencia para comparar las otras vertientes y evaluar su eficacia en términos de rendimiento.

5.2. Cambios en el algoritmo de entrenamiento

En el capítulo 4.4 se observó en el análisis estadístico que existe relación entre el desempeño en la métrica de exactitud y la entropía de los filtros dada por la ecuación (5.1).

$$H(S) = -\sum_{z=1}^n P(s_z) \log_2(P(s_z)) \quad (5.1)$$

Donde $H(S)$ es la entropía de la variable aleatoria S y n es el número de distintos valores posibles que puede tomar s .

Debido a esto, se implementó el algoritmo entropy gain, propuesto por Camacho M. en [65]. Este algoritmo tiene como objetivo estimular los filtros en diferentes capas del modelo para favorecer un aprendizaje equilibrado en cada uno de ellos. Para lograrlo, se reinician los filtros cuya entropía cae fuera de un rango predefinido. La reinicialización se realiza utilizando el algoritmo de Glorot Bengio, lo que podría ayudar a evitar el estancamiento del aprendizaje en ciertas capas.

5.2.1. Algoritmo Glorot-Bengio

En el proceso de entrenamiento de modelos de aprendizaje profundo es fundamental mantener un equilibrio en la inicialización de los filtros de las capas neuronales. Una inicialización adecuada puede contribuir significativamente a evitar problemas como el estancamiento del aprendizaje y el desvanecimiento de gradiente, permitiendo que el modelo converja de manera más eficiente hacia soluciones óptimas. En este contexto, el algoritmo de Glorot-Bengio (conocido como inicialización Glorot o inicialización Xavier), se ha destacado como un enfoque valioso para establecer una inicialización ponderada de los pesos de las redes neuronales.

El algoritmo de Glorot-Bengio toma su nombre de los investigadores Xavier Glorot y Yoshua Bengio, quienes propusieron este enfoque en [66]. La idea central detrás de esta técnica es adaptar la inicialización de los pesos de las neuronas de manera que se controle la propagación de las señales y los gradientes a lo largo de las capas de la red, evitando que se vuelvan demasiado pequeños o grandes a medida que profundizamos en la arquitectura. En términos simples, el algoritmo de Glorot-Bengio inicializa los pesos de las neuronas utilizando una distribución uniforme o gaussiana con varianza específica. Esta varianza se calcula con el número de neuronas en la capa actual y con el número de neuronas en la capa anterior. La ecuación para calcular la varianza está definida por la ecuación (5.2).

$$Var[W^i] = \frac{2}{n_i + n_{i+1}} \quad (5.2)$$

donde W^i son los pesos de la i -ésima capa y n_i la cantidad de neuronas en la i -ésima capa. Esto permite que los pesos se inicialicen de manera que la propagación hacia adelante y hacia atrás de las señales se mantenga en un rango adecuado, lo que puede mejorar la estabilidad del entrenamiento y la convergencia del modelo.

La reinicialización de los filtros utilizando el algoritmo de Glorot-Bengio, tiene como objetivo abordar el riesgo de estancamiento del aprendizaje en ciertas capas del modelo que se está entrenando, asegurando así una progresión más suave del proceso de optimización.

El algoritmo de Glorot-Bengio para la reinicialización de filtros en el análisis de muestras urológicas, frotis sanguíneo y cultivos microscópicos contribuye a mantener la estabilidad y el equilibrio en el aprendizaje de cada capa, permitiendo que el modelo explore y aprenda patrones complejos de manera más eficaz y evitando los obstáculos que pueden surgir debido a una mala inicialización de los pesos.

5.3. Transferencia de aprendizaje

La transferencia de aprendizaje es una técnica que implica cargar un modelo previamente entrenado en una tarea y ajustar solo las últimas capas del modelo para adaptarlo a la nueva tarea. Las capas iniciales se mantienen congeladas para preservar los patrones de características aprendidos anteriormente, entrenando únicamente la parte final del modelo. Esto es particularmente útil cuando se dispone de un modelo bien entrenado en una tarea inicial

y se desea aprovechar ese conocimiento para mejorar el rendimiento en una tarea específica y similar a la tarea inicial. Entre las ventajas de esta técnica se incluyen el ahorro de tiempo y recursos, la mejor generalización a nuevas tareas y su efectividad con conjuntos de datos pequeños. No obstante, puede no funcionar bien en tareas y dominios muy diferentes, y es importante gestionar el ajuste fino para evitar el sobreajuste y elegir adecuadamente las redes preentrenadas basándose en el conocimiento de dominio.

5.4. Reentrenamiento

El reentrenamiento consiste en cargar un modelo previamente entrenado y utilizarlo como punto de partida para continuar el proceso de aprendizaje. A diferencia de la transferencia de aprendizaje, en el reentrenamiento se somete todo el modelo a un proceso de adaptación. Esto significa que cada capa se ajusta para ser funcional en la nueva tarea, sin congelar ninguna. De esta manera, el reentrenamiento parte de los conocimientos adquiridos por el modelo en una etapa anterior. Luego, se ajusta el modelo completo utilizando nuevos datos o ajustes específicos para la nueva tarea. Esto permite afinar cada capa del modelo para que sea más efectiva en la tarea actual. A diferencia de la transferencia de aprendizaje, el reentrenamiento permite una adaptación completa del modelo a la nueva tarea, lo que puede ser beneficioso cuando las características de la nueva tarea son diferentes de la tarea original. En algunos casos, el reentrenamiento puede mejorar el rendimiento y la convergencia del modelo en comparación con el entrenamiento tradicional, ya que el modelo tiene condiciones iniciales definidas por una base de datos previa. Al entrenar completamente el modelo, existe el riesgo de perder información valiosa adquirida en la etapa anterior, lo que puede afectar negativamente el rendimiento en la nueva tarea. El reentrenamiento no garantiza una convergencia adecuada, y es posible que se requieran más datos o ajustes para lograr resultados óptimos.

En resumen, el reentrenamiento se diferencia de la transferencia de aprendizaje en que adapta todo el modelo a la nueva tarea, lo que lo hace especialmente útil cuando las características de la nueva tarea son muy diferentes de la tarea original. Sin embargo, existe el riesgo de perder información previa y no se garantiza la convergencia adecuada.

6. RESULTADOS

Este capítulo presenta los resultados obtenidos a partir de la evaluación de nuestro modelo LDE-Net, tanto en la primera como segunda versión, con las bases de datos de USD_2, Raabin-wbc y BYRI. Los experimentos llevados a cabo en estas diversas bases de datos brindan una visión amplia y completa del rendimiento de nuestro modelo en una variedad de contextos y escenarios. Además, para garantizar la robustez y la fiabilidad de los hallazgos, se han aplicado validaciones cruzadas para cada conjunto de datos utilizando los distintos algoritmos de entrenamiento presentados en el capítulo 5. Cada base de datos presenta sus propios desafíos y requisitos específicos, lo que permite evaluar cómo el modelo se adapta a diferentes situaciones y demuestra su capacidad de generalización. Las bases de datos se dividieron en 70% para el conjunto de entrenamiento y 30% en el conjunto de validación y prueba. Para abordar el desbalance de clases se utilizó submuestreo para la clase más escasa. Cabe aclarar que todos los resultados presentados en este capítulo son referentes a los conjuntos de validación de las respectivas bases de datos. La computadora utilizada para los experimentos posee una tarjeta gráfica NVIDIA 3060, 9 GB de RAM, y un procesador Intel Xeon E5620.

Las métricas utilizadas fueron exactitud, precisión, recall y F1, las cuales están definidas por las ecuaciones 4.1 a 4.4.

$$Acc = \frac{Tp+Tn}{Tp+Tn+Fp+Fn} \quad (4.1)$$

$$P = \frac{Tp}{Tp+Fp} \quad (4.2)$$

$$Rc = \frac{Tp}{Tp+Fn} \quad (4.3)$$

$$F1 = \frac{(P)(Rc)}{P+Rc} \quad (4.4)$$

Donde Acc es exactitud, P es precisión, Rc es recall, mientras que Tp, Tn, Fp y Fn se refiere a verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos respectivamente.

Este capítulo está organizado de la siguiente manera: la Sección 6.1 presentan los resultados correspondientes a los experimentos realizados con la base de datos USD_2. La Sección 6.2 incluye los resultados referentes a la base de datos Raabin-wbc. Finalmente, en la Sección 6.3 se encuentran los resultados con la base de datos BYRI.

6.1. Urinary Sediment Dataset

El trabajo realizado se enfocó principalmente en la base de datos USD_2, debido a los objetivos planteados en la propuesta de tesis. Por lo tanto, a continuación, se presentan los resultados obtenidos en las distintas fases del desarrollo del proyecto, desde el objetivo inicial con 3 clases asociadas con infección urinaria, hasta llegar a la totalidad de la base de datos. Así mismo, se realizaron pruebas con los distintos algoritmos de entrenamiento presentados en el capítulo 5, para finalmente verificar la generalización del modelo mediante validación cruzada.

6.1.1. Entrenamiento con 3 clases

Inicialmente, se realizaron pruebas con las tres arquitecturas mencionadas en el capítulo 3 (MobileNetV2, DenseNet201 y EfficientNetV2L). Las redes se entrenaron utilizando el optimizador Adam, la función de costo de entropía cruzada definida por la ecuación (4.1) y 100 épocas de entrenamiento. Así mismo, se analizaron 3 de las 7 clases contenidas en la base de datos: bacterias, eritrocitos y células epiteliales. En la Tabla 6.1 se muestran los resultados obtenidos.

Tabla 6.1 Resultados de validación en distintas arquitecturas para 3 clases utilizando optimizador Adam.

Métricas	MobileNetV2	DenseNet201	EfficientNetV2L
Accuracy	0.94807792	0.969994705	0.98483413
F1	0.94832027	0.960004182	0.9848485
P	0.94886661	0.969994705	0.98483413
R	0.94786727	0.969994705	0.98483413
Parámetros	3.5M	20.2M	119.0M
Tiempo proc (por imagen)	8 ms	37 ms	76 ms
RAM pico (MB)	1507570688	5604800768	4084061184
RAM prom (MB)	26368	74496	128768

Se puede observar en la tabla que las tres redes tuvieron resultados superiores al 94% en las distintas métricas. Es destacable a su vez que los resultados tienen valores semejantes en exactitud, F1, P y R, mostrando con ello que son capaces de determinar en medidas semejantes los verdaderos positivos y negativos. La red con mejores resultados fue EfficientNetV2L, la cual tiene una cantidad muy superior de parámetros y de consumo de memoria RAM en comparación a las otras arquitecturas, lo cual trae consigo un mayor tiempo de procesamiento por imagen.

Observando el comportamiento en las distintas arquitecturas, se desarrolló la primera versión de LDE-Net, según lo comentado en el capítulo 4, con lo cual se obtuvieron los resultados presentados en la Tabla 6.2

CAPÍTULO VI. RESULTADOS

Tabla 6.2 Resultados de validación en LDE-Net para 3 clases utilizando optimizador Adam.

Accuracy	F1	P	R	Parámetros	Tiempo proc (por imagen)	RAM pico (MB)	RAM prom (MB)
0.98578197	0.98575425	0.98578197	0.98578197	5.3M	17 ms	5604800757	6144

En esta tabla, se puede ver que se obtuvieron resultados bastante semejantes a los conseguidos con EfficientNetV2L, incluso superiores. Así mismo, es destacable que esto se logró con una reducción de más del 95% en la cantidad de parámetros, y disminuyendo con ello el tiempo de procesamiento por imagen a 17 ms.

Siguiendo este esquema, en la Figura 6.1 se presentan las gráficas correspondientes al comportamiento de la exactitud y pérdida para las distintas arquitecturas. En cada gráfica se muestran las métricas tanto en entrenamiento como en validación.

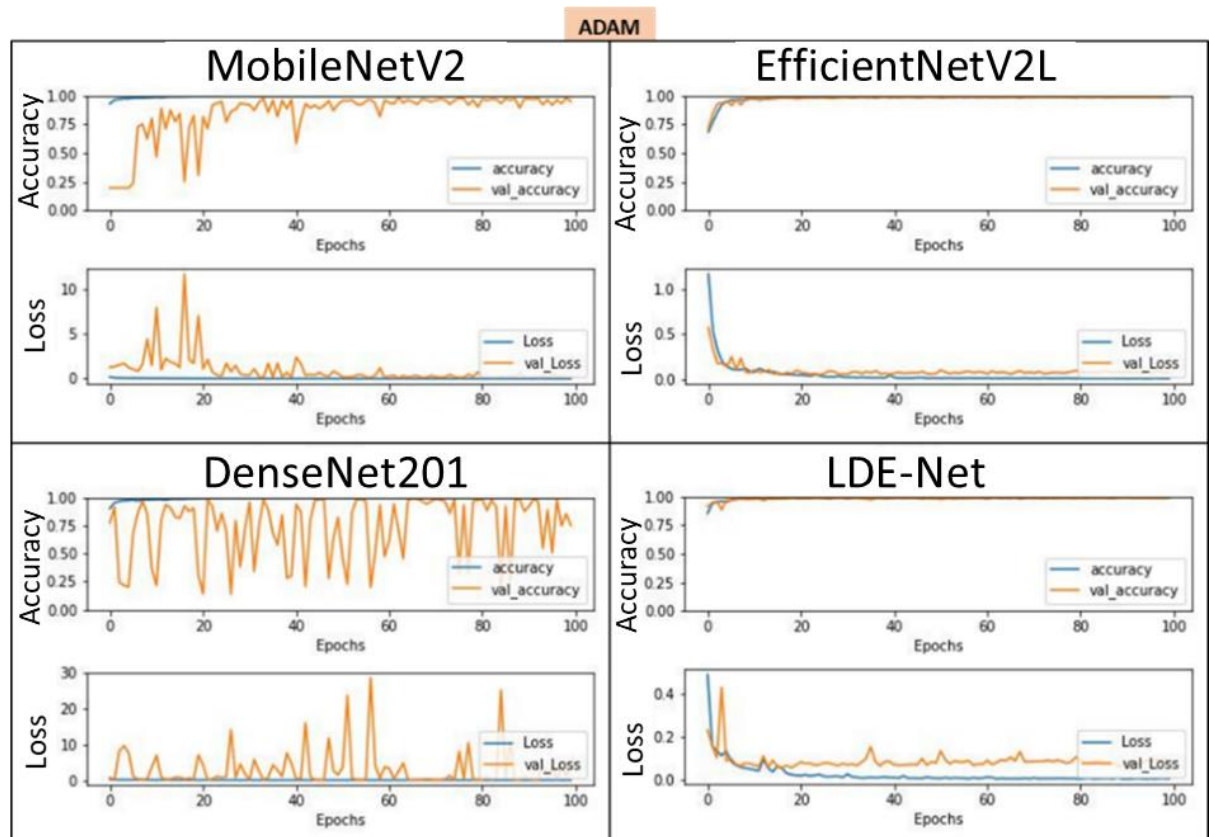


Figura 6.1 Gráficas de exactitud y pérdida en distintas arquitecturas para 3 clases utilizando optimizador Adam.

CAPÍTULO VI. RESULTADOS

En las distintas gráficas se puede observar que en todos los casos se tienen valores cercanos al 100% en la exactitud del entrenamiento. Sin embargo, en la exactitud de validación existen distintas fluctuaciones a lo largo de las épocas para las distintas arquitecturas. La menor presencia de este efecto se encuentra en EfficientNetV2L y en LDE-Net, mientras que DenseNet201 presenta la mayor variación.

Aunado a esto, se realizaron pruebas con el optimizador de gradiente estocástico descendiente (SGD), utilizando un tamaño de lote de uno. En la Tabla 6.3 se presentan los resultados en las distintas métricas, mientras que en la Figura 6.2 se muestran las gráficas correspondientes.

Tabla 6.3 Resultados de validación en distintas arquitecturas para 3 clases utilizando optimizador SGD.

Métricas	MobileNetV2	DenseNet201	EfficientNetV2L	LDE-NET
Accuracy	0.97072142	0.989994705	0.97651398	0.98451817
F1	0.97074914	0.990004182	0.97653621	0.98449165
P	0.97072142	0.989994705	0.97651398	0.98451817
R	0.97072142	0.989994705	0.97651398	0.98451817
Parámetros	3.5M	20.2M	119.0M	5.3M
Tiempo proc.	8 ms	37 ms	76 ms	17 ms

En esta tabla se puede visualizar que DenseNet201 fue la arquitectura con mejor desempeño en las distintas métricas, mientras que LDE-Net presenta resultados cercanos con poco más de la cuarta parte de parámetros, y menos de la mitad en tiempo de procesamiento por imagen.

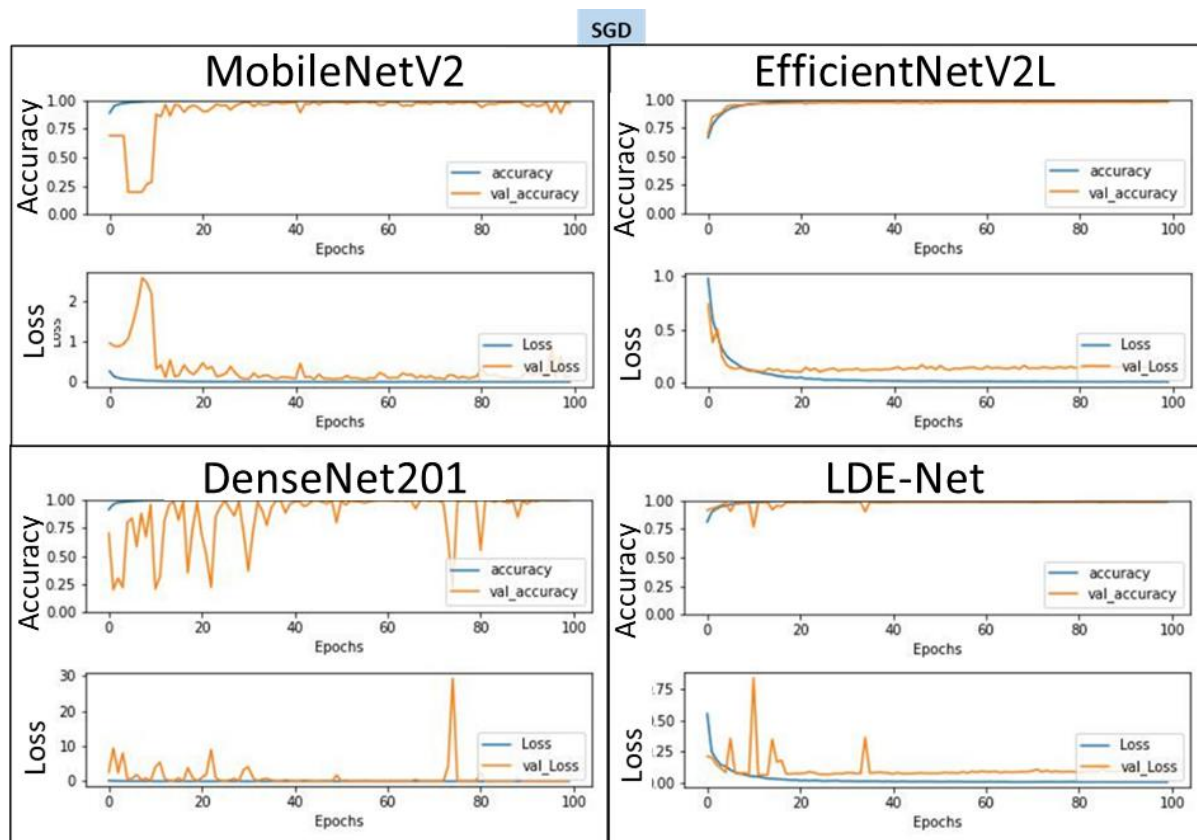


Figura 6.2 Gráficas de exactitud y pérdida en distintas arquitecturas para 3 clases utilizando optimizador SGD.

En la figura se puede observar que, si bien DenseNet201 obtuvo un mejor desempeño que LDE-Net según la Tabla 6.3, la convergencia del modelo fue en épocas más tempranas para este último. Por otra parte, es importante destacar que al utilizar el optimizador SGD redujo significativamente el sobreajuste para las distintas arquitecturas respecto a Adam en MobileNetV2, DenseNet201 y EfficientNetV2L mientras que aumentó en las primeras épocas de entrenamiento en LDE-Net.

6.1.2. Entrenamiento con 5 clases

Una vez obtenidos resultados aceptables con LDE-Net para la clasificación de 3 clases, se decidió aumentar de manera paulatina las clases para analizar las 7 categorías pertenecientes a la base de datos. Como primer paso, se agregaron al análisis 2 clases más: cristales y hongos. Se eligieron éstas debido a la diferencia morfológica significativa que tiene con las otras clases analizadas previamente. En la Tabla 6.4 se muestran los resultados obtenidos en las distintas arquitecturas utilizando el optimizador Adam.

Tabla 6.4 Resultados de validación en distintas arquitecturas para 5 clases utilizando optimizador Adam.

CAPÍTULO VI. RESULTADOS

Métricas	MobileNetV2	DenseNet201	EfficientNetV2L	LDE-NET
Accuracy	0.90592051	0.67802107	0.92173558	0.94079483
F1	0.90761089	0.67667091	0.92176044	0.94123292
P	0.9083876	0.68438399	0.92370462	0.94148719
R	0.90470397	0.67356044	0.91808599	0.93957824
Parámetros	3.5M	20.2M	119.0M	5.3M
Tiempo proc.	8 ms	37 ms	76 ms	17 ms

De igual manera que el entrenamiento con 3 clases, EfficientNetV2L y LDE-Net presentaron los desempeños más altos, siendo superiores al 92% en las distintas métricas de desempeño, donde LDE-Net es la red con mejores resultados. En la Figura 6.3 se incluyen las gráficas correspondientes al entrenamiento para 5 clases utilizando el optimizador Adam, y se puede observar en MobileNetV2 y DenseNet201 que los desempeños en las primeras épocas de entrenamiento son muy bajos, incluso aparentando una parálisis momentánea en MobileNetV2. Sin embargo, una vez alcanzadas las 20 épocas, este modelo comienza a aprender de mejor manera. De manera contrastante, en EfficientNetV2L y LDE-Net se alcanzan valores de exactitud altos tanto para los conjuntos de entrenamiento como los de validación en épocas tempranas del entrenamiento. Así mismo, en estas arquitecturas se tiene un ligero efecto de sobreajuste, sin embargo, las curvas de exactitud y pérdida en entrenamiento y validación permanecen cercanas entre ellas. Caso contrario, MobileNetV2 y DenseNet201 muestran un sobreajuste notablemente superior en varios momentos del entrenamiento.

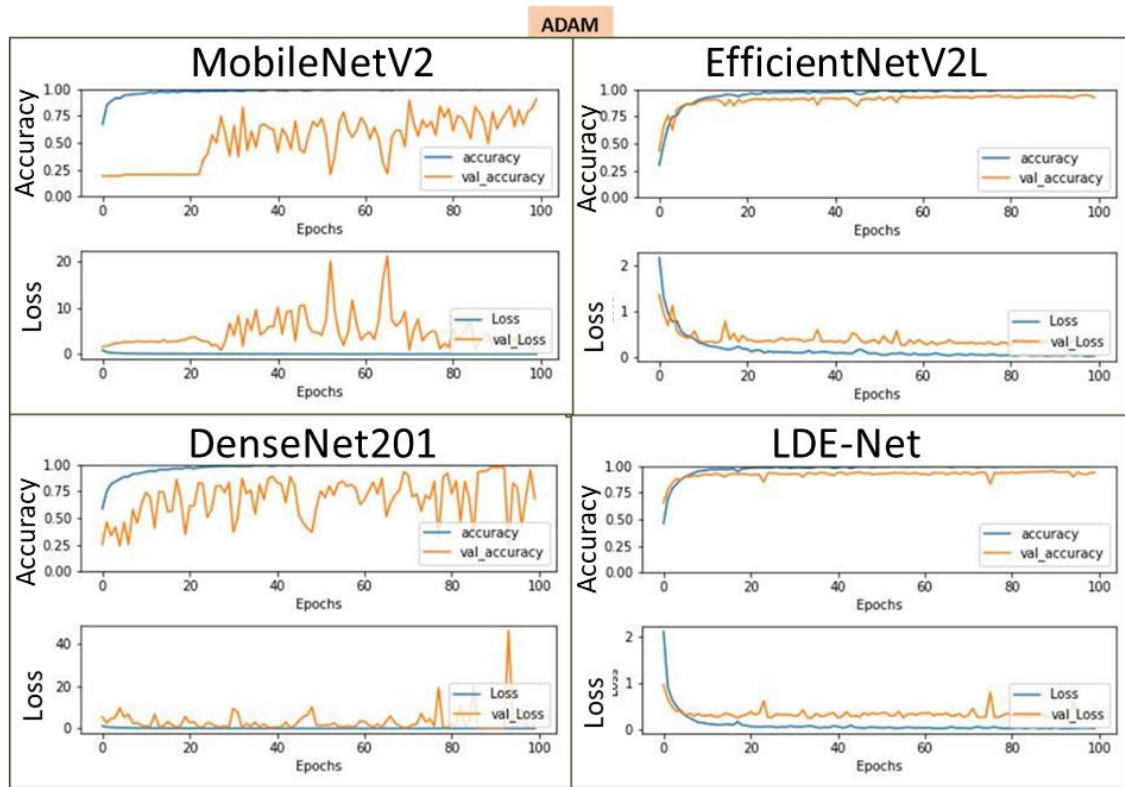


Figura 6.3 Gráficas de exactitud y pérdida en distintas arquitecturas para 5 clases utilizando optimizador Adam.

De igual manera, en la Tabla 6.5 se muestran los resultados obtenidos utilizando el optimizador SGD. Según la información mostrada en la Tabla 6.5 y en la Figura 6.4, MobileNetV2 obtuvo el mejor desempeño utilizando el optimizador SGD, aunque logró converger hasta después de 30 épocas. Por su parte, el resto de las redes lograron una convergencia en épocas más tempranas, destacando EfficientNetV2L y LDE-Net. Al igual que en los experimentos anteriores, se observa un sobreajuste importante tanto en MobileNetV2 como en DenseNet201, mientras que en las otras dos arquitecturas este efecto es menor.

Tabla 6.5 Resultados de validación en distintas arquitecturas para 5 clases utilizando optimizador SGD.

Métricas	MobileNetV2	DenseNet201	EfficientNetV2L	LDE-NET
Accuracy	0.9566099	0.84428221	0.88767236	0.93592864
F1	0.95730942	0.84611559	0.88335538	0.93696576
P	0.95736909	0.84546566	0.89002037	0.93775427
R	0.95620435	0.84306568	0.88605028	0.93471211
Parámetros	3.5M	20.2M	119.0M	5.3M
Tiempo proc.	8 ms	37 ms	76 ms	17 ms

Finalmente, en la Figura 6.4 se incluyen las gráficas correspondientes.

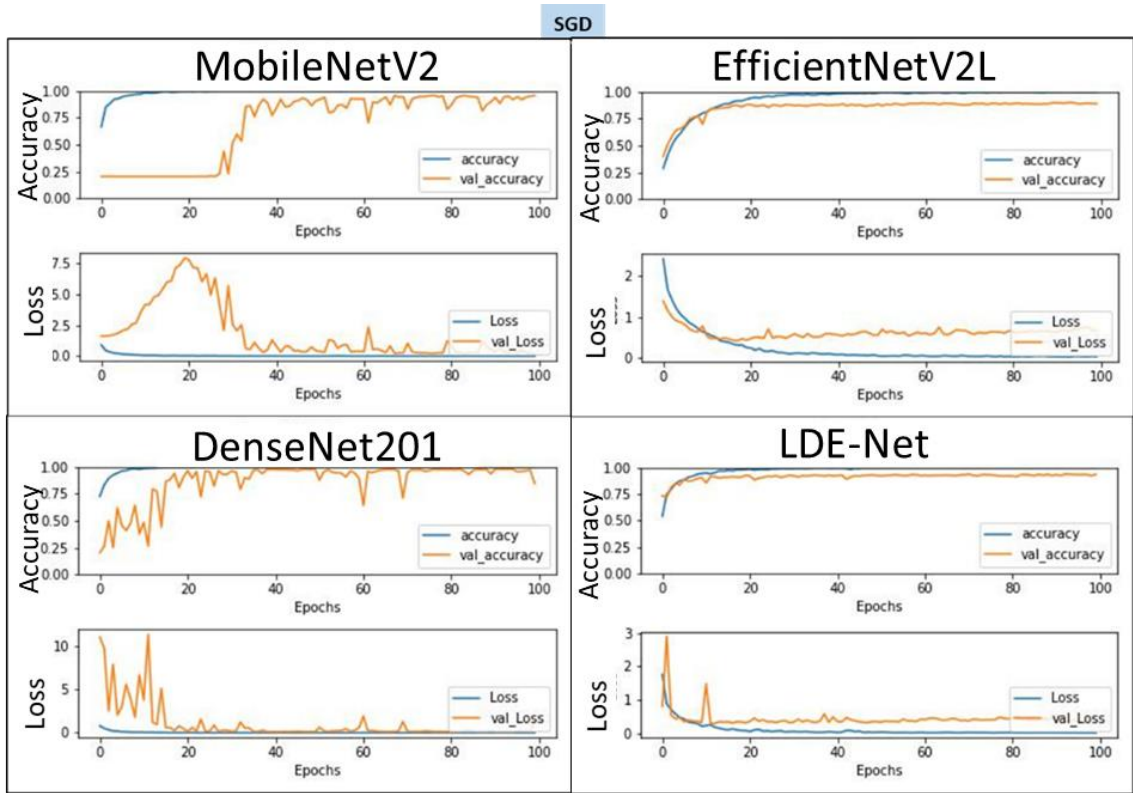


Figura 6.4 Gráficas de exactitud y pérdida en distintas arquitecturas para 5 clases utilizando optimizador SGD.

6.1.3. Entrenamiento con 7 clases

Finalmente, se agregaron a las pruebas las clases faltantes: leucocitos y núcleos epiteliales. En este caso, se optó por realizar pruebas únicamente en la red propuesta LDE-Net, debido a que en los experimentos anteriores se tuvieron comportamientos similares en las otras arquitecturas. En la Tabla 6.6 se presentan los resultados para 7 clases con la arquitectura LDE-Net.

Tabla 6.6 Resultados de validación en LDE-Net para 7 clases utilizando optimizador Adam.

Métricas	LDE-NET
Accuracy	0.93238544
F1	0.93238544
P	0.93322784
R	0.93322784
Parámetros	5.3M
Tiempo proc.	17 ms

En la Figura 6.5 se muestran las gráficas correspondientes a la clasificación de 7 clases con la arquitectura LDE-Net, utilizando el optimizador Adam.

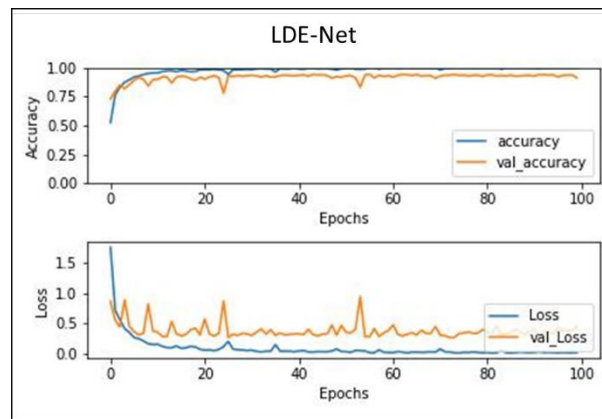


Figura 6.5 Gráficas de exactitud y pérdida en LDE-Net para 7 clases utilizando optimizador Adam.

La convergencia del modelo es visible desde el inicio del entrenamiento, y se alcanzan valores superiores al 93% en las distintas métricas de desempeño. De igual manera que en los experimentos anteriores, los valores de

CAPÍTULO VI. RESULTADOS

las métricas son muy parecidos entre sí, indicando clasificación correcta cercana en verdaderos positivos y verdaderos negativos.

En este punto, es donde se creó la segunda versión de LDE-Net (LDE-NetV2) según lo comentado en el capítulo 4.7, con lo que se realizaron pruebas utilizando las 7 clases de USD_2. En la Tabla 6.7 se presentan los resultados obtenidos con LDE-NetV2, así como los obtenidos con LDE-Net para fines de comparación. Se puede observar que los resultados fueron superiores a la versión 1 de LDE-Net, además de reducir los parámetros en un 54%, siendo una reducción del 98% en comparación con EfficientNetV2L.

Tabla 6.7 Resultados de validación en LDE-NetV2 para 7 clases utilizando optimizador Adam.

Métricas	LDE-NetV2	LDE-Net
Accuracy	0.95249606	0.93238544
F1	0.95249606	0.93238544
P	0.95328266	0.93322784
R	0.95328266	0.93322784
Parámetros	2.45 M	5.3M
Tiempo proc.	11 ms	17 ms

Así mismo, para fines de comparación, se presentan las gráficas en ambas versiones de LDE-Net en la Figura 6.6. Es visible que el comportamiento en ambas es similar, alcanzando valores de exactitud en el conjunto de validación superior al 90% desde las primeras épocas, y se mantiene a lo largo del entrenamiento. Aunado a esto, existe un menor sobreajuste en LDE-NetV2 respecto a LDE-Net, lo cual muestra que la capacidad reducida de la segunda versión, favorece una mejor generalización.

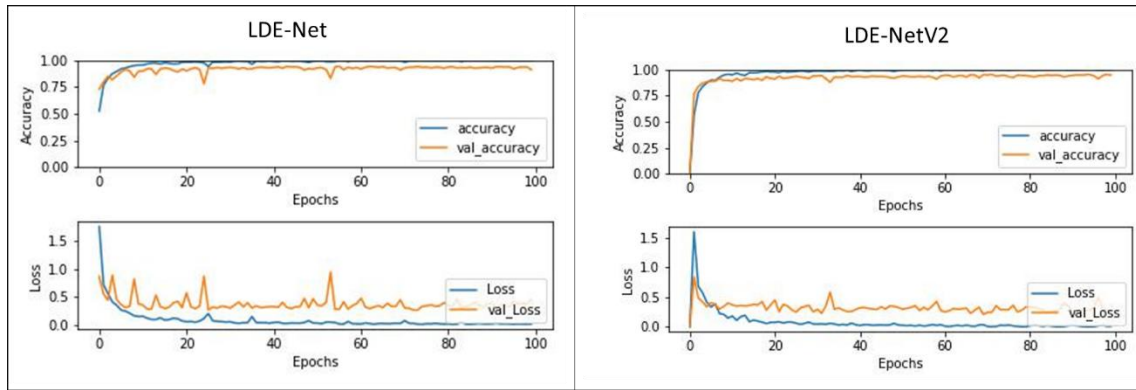


Figura 6.6 Gráficas de exactitud y pérdida en LDE-Net y LDE-NetV2 para 7 clases utilizando optimizador Adam.

6.1.4. Entrenamiento con 7 clases mediante otros algoritmos de entrenamiento

Como se mencionó en el capítulo 5, se analizaron distintos algoritmos de entrenamiento para buscar mejorar el desempeño del modelo LDE-NetV2. Inicialmente, se añadió el algoritmo *entropy gain* durante el entrenamiento. En la Tabla 6.8 se incluyen los resultados obtenidos al utilizar este algoritmo, en comparación con LDE-Net y LDE-NetV2 sin el mismo.

Tabla 6.8 Resultados de validación en distintas versiones de LDE-Net y el uso de *entropy gain*.

Métricas	LDE-NET	LDE-NETV2	LDE-NETV2 + EG
Accuracy	0.93238544	0.95249606	0.95638544
Parámetros	5.3M	2.45 M	2.45 M
Tiempo proc.	17 ms	11 ms	11 ms

El uso del algoritmo *entropy gain* favoreció el aprendizaje del modelo según los resultados presentados. Si bien el aumento fue relativamente bajo, indica un punto de oportunidad para la obtención de mejores resultados al modificar el algoritmo de entrenamiento. Así mismo, con la finalidad de visualizar los casos en que se presenta mayor cantidad de errores, en la Figura 6.7 se incluye la matriz de confusión obtenida en el experimento de LDE-NetV2 utilizando *entropy gain*.

Clase predicha	Bacteria	665	2	8	4	1	3	8
	Cristal	3	670	5	5	1	2	6
	Célula epitelial	4	3	664	6	1	3	2
	Núcleo epitelial	2	4	3	636	15	10	5
	Eritrocito	4	5	1	18	651	12	3
	Leucocito	1	1	2	15	16	654	3
	Hongo	8	2	4	3	2	3	660
		Bacteria	Cristal	Célula epitelial	Núcleo epitelial	Eritrocito	Leucocito	Hongo
Clase real								

Figura 6.7 Matriz de confusión obtenida en LDE-NetV2 + EG con 7 clases.

La matriz de confusión indica que los casos en que mayores errores se presentaron fue en las clases de eritrocitos, leucocitos y núcleos epiteliales, lo cual puede deberse a la semejanza morfológica entre estas clases. No obstante, la gran mayoría de las muestras analizadas fueron clasificadas correctamente, concentrándose en la diagonal principal de la matriz. Además de esto, se realizaron pruebas de transferencia de aprendizaje, en las cuáles se congelaron las últimas 3 capas del modelo, y se tomaron como punto de partida el modelo LDE-NetV2 entrenado para 3 clases de USD_2 y el entrenado para 5 clases de USD_2. Así mismo, se realizó una transferencia de aprendizaje por etapas, tomando un modelo entrenado para 3 clases, transfiriendo el aprendizaje para 5 clases y finalmente ajustando a 7 clases. En la Tabla 6.9 se presentan los resultados obtenidos de estas pruebas, siendo la última columna los desempeños obtenidos sin utilizar transferencia de aprendizaje para fines comparativos. Se utilizó el modelo LDE-NetV2 sin *entropy gain*.

Tabla 6.9 Resultados en transferencia de aprendizaje utilizando LDE-NetV2.

Métricas	3 a 7 clases	5 a 7 clases	3 a 5 a 7 clases	7 clases
Accuracy	0.81692094	0.94244105	0.92335767	0.95249606
F1	0.55625403	0.6683538	0.72768116	0.95249606
P	0.38684139	0.50192779	0.57487011	0.95328266
R	0.98266298	0.99306518	0.98702353	0.95328266
Parámetros	2.45 M	2.45 M	2.45 M	2.45 M
Tiempo proc.	11 ms	11 ms	11 ms	11 ms

CAPÍTULO VI. RESULTADOS

En los experimentos realizados con transferencia de aprendizaje, no se obtuvo mejoría alguna en el desempeño del modelo, por el contrario, se presenta una disminución en el mismo. Sin embargo, dentro de las pruebas analizadas, la mejor metodología encontrada para transferencia de aprendizaje fue mediante el entrenamiento para 5 clases para posteriormente realizar la transferencia a 7 clases. Es importante también destacar que, al utilizar esta metodología, la métrica de precisión disminuyó en todos los casos, lo cual indica una menor tasa de aciertos en verdaderos positivos.

Finalmente, en la Tabla 6.10 se muestra la comparación del mejor resultado obtenido con otros trabajos del estado del arte.

Tabla 6.10 Comparación de LDE-NetV2 + EG con trabajos en el estado del arte.

Método	Accuracy	Tiempo procesamiento.
CNN + Area Feature Algorithm [45]	96.75%	61 ms
LDE-NETV2 + EG	95.63%	11 ms
CNN (LeNet-5) [52]	92.30%	
CNN (DFPN) [28]	91.40%	201 ms
Gabor + Scattering Transformation + SVM [53]	90%	
AdaBoost + SVM [44]	90%	
Faster RCNN+SSD [54]	84.10%	72 ms

En el estado del arte analizado, el modelo LDE-NetV2 con el algoritmo de *entropy gain* se sitúa en la segunda posición respecto a la métrica de exactitud, no obstante, se encuentra en primer lugar en tiempo de procesamiento.

6.1.5. Validación cruzada

Con la finalidad de validar la generalización del modelo, se realizaron pruebas de validación cruzada utilizando 5 subconjuntos de datos.

En la Tabla 6.11 se incluyen los resultados utilizando los distintos subconjuntos de validación, así como el promedio de estos.

CAPÍTULO VI. RESULTADOS

Tabla 6.11 Resultados validación cruzada USD_2 7 clases.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Promedio
Acc	0.95082939	0.94727486	0.9514218	0.94845974	0.95379144	0.95035545

En los distintos subconjuntos se obtuvo un desempeño similar, lo cual indica una buena generalización del modelo. Finalmente, en la Figura 6.8 se muestra de manera gráfica el comportamiento de la exactitud de validación durante el entrenamiento.

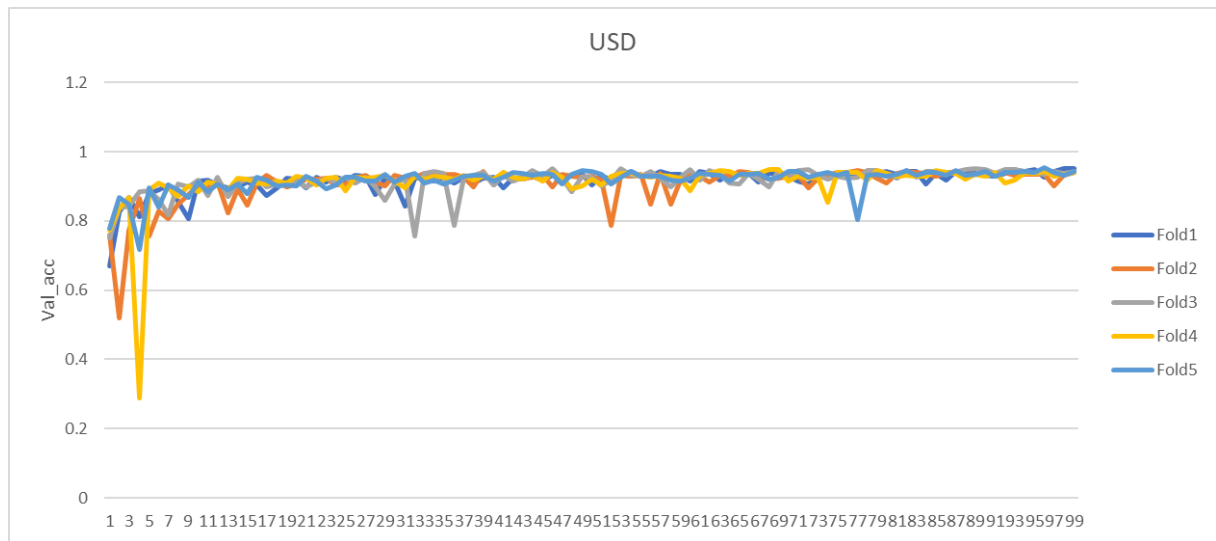


Figura 6.8 Gráficas de exactitud en 5 subconjuntos para validación cruzada para 7 clases de USD_2.

Cómo se puede observar, las gráficas correspondientes a la exactitud de validación en los distintos subconjuntos se comportan de manera similar, lo cual concuerda con lo mencionado respecto a la Tabla 6.11, y no existen indicios importantes de sesgo.

6.2. Raabin-wbc

Dado que los mejores resultados en la base de datos de urología fueron conseguidos utilizando la combinación de LDE-NetV2 con el algoritmo *entropy gain*, se utilizó esta combinación en la base de datos de Raabin-wbc. Los resultados presentados a continuación fueron obtenidos utilizando el optimizador Adam, la función de pérdida de entropía cruzada definida por la ecuación (4.1) y 100 épocas de entrenamiento.

6.2.1. Entrenamiento

Para este caso, se analizaron las 5 clases pertenecientes a la base de datos de Raabin-wbc: basófilos, eosinófilos, linfocitos, monocitos y neutrófilos. En la Tabla 6.12 se muestran los resultados obtenidos.

Tabla 6.12 Resultados de validación en base de datos Raabin-wbc utilizando LDE-NetV2 +EG.

Accuracy	F1	Precision	Recall
0.97728395	0.97753906	0.97774482	0.97728395

En la Figura 6.9 se muestran las gráficas correspondientes a la exactitud y pérdida utilizando la base de datos Raabin-wbc.

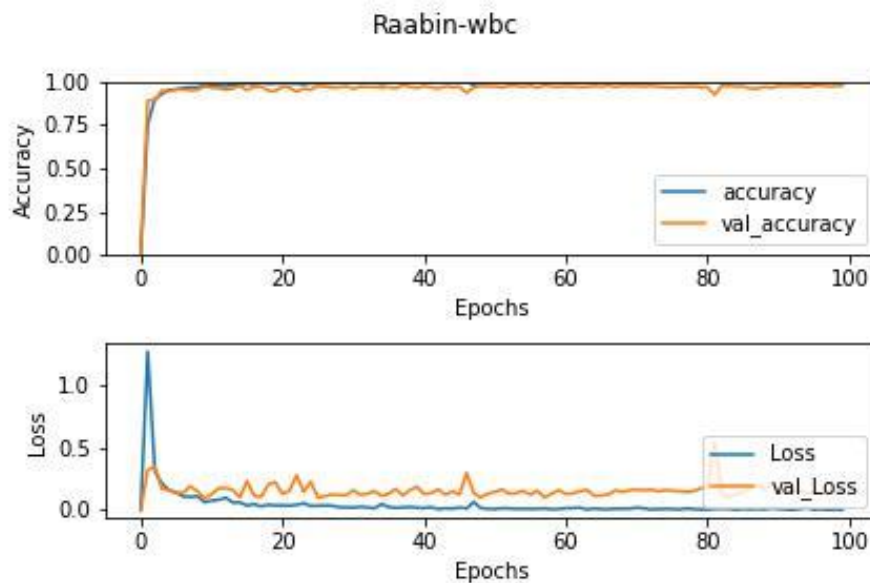


Figura 6.9 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos Raabin-wbc.

La Tabla 6.12 y la Figura 6.9 muestran que el modelo fue capaz de alcanzar un desempeño superior al 97% en las distintas métricas, además de alcanzar la convergencia desde las primeras épocas de entrenamiento y con un sobreajuste leve.

En el mismo sentido, en la Tabla 6.13 se muestran los resultados obtenidos utilizando transferencia de aprendizaje, tomando como modelo previo el modelo con mejor resultado obtenido en los experimentos de la base de datos USD_2, congelando el mismo y entrenando únicamente la capa softmax.

CAPÍTULO VI. RESULTADOS

Tabla 6.13 Resultados de validación en base de datos Raabin-wbc utilizando transferencia de aprendizaje.

Accuracy	F1	Precision	Recall
0.89157218	0.8917166	0.89328063	0.89107937

En la Figura 6.16 se muestran las gráficas del comportamiento del entrenamiento, utilizando transferencia de aprendizaje.

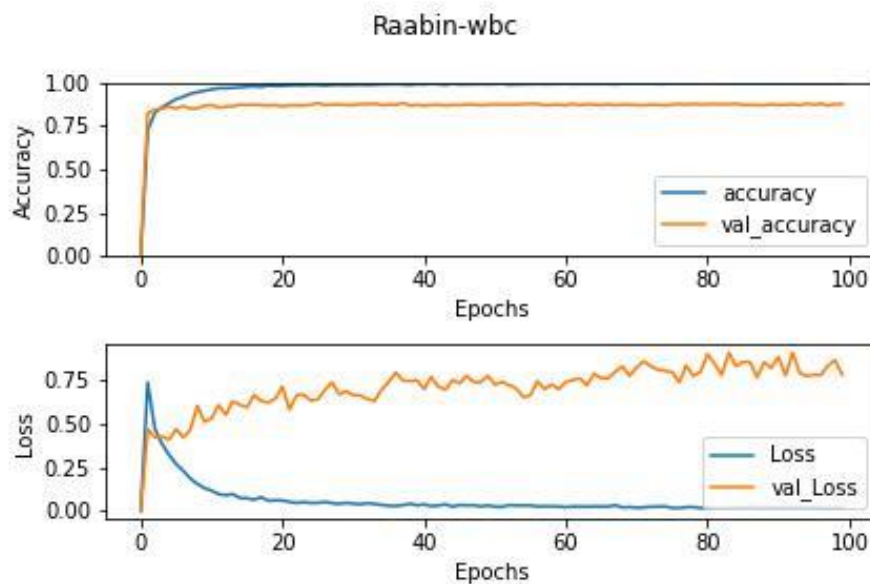


Figura 6.10 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos Raabin-wbc utilizando transferencia de aprendizaje.

Los resultados muestran que, al utilizar transferencia de aprendizaje, disminuyó el desempeño del modelo, además de presentar parálisis desde épocas tempranas del entrenamiento.

Así mismo, en la Tabla 6.14 se muestran los resultados obtenidos referentes al uso de reentrenamiento, tomando como punto de partida el modelo con el mejor resultado obtenido para USD_2.

Tabla 6.14 Resultados de validación en base de datos Raabin-wbc utilizando reentrenamiento.

Accuracy	F1	Precision	Recall
0.97725022	0.97777933	0.97773379	0.97725022

CAPÍTULO VI. RESULTADOS

Finalmente, en la Figura 6.8 se muestra de manera gráfica el comportamiento del entrenamiento tanto en el conjunto de entrenamiento como el de validación, utilizando reentrenamiento.

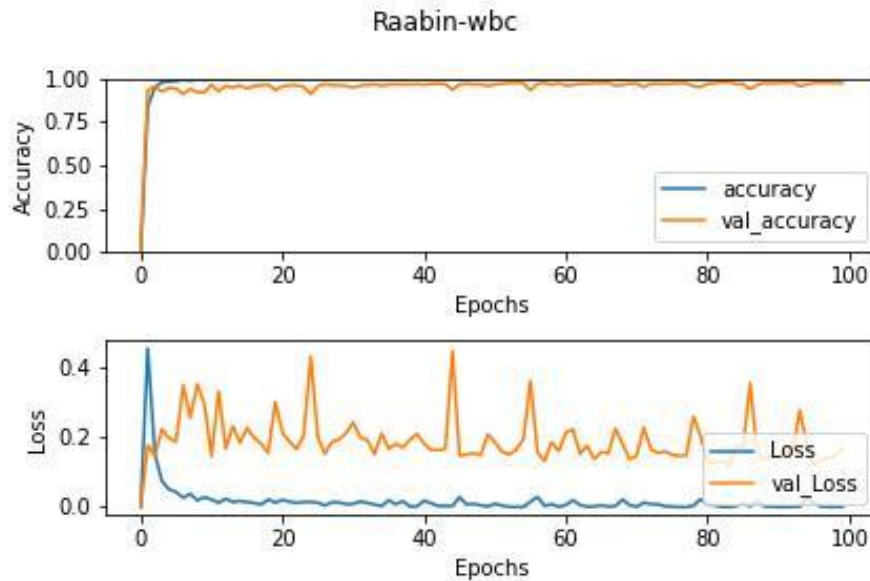


Figura 6.11 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos Raabin-wbc utilizando reentrenamiento.

Los resultados muestran que, al utilizar reentrenamiento, el desempeño del modelo fue similar al entrenamiento tradicional. Por otro lado, las gráficas muestran que existe un leve sobreajuste del modelo.

6.2.2. Validación cruzada

Buscando verificar la generalización del modelo, se realizaron pruebas de validación cruzada con los distintos algoritmos de entrenamiento. En todos los casos se utilizaron 5 subconjuntos de datos.

En la Tabla 6.15 se incluyen los resultados utilizando los distintos subconjuntos de validación, así como el promedio de estos. En ella se puede observar que la exactitud alcanzada en los distintos subconjuntos de validación es similar entre ellos, indicando una correcta generalización del modelo para la base de datos utilizada.

Tabla 6.15 Resultados validación cruzada Raabin-wbc.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Promedio
Acc	0.97634304	0.97634304	0.9758144	0.97728395	0.97725022	0.97660693

CAPÍTULO VI. RESULTADOS

De manera gráfica, en la Figura 6.12 se muestra el comportamiento de la exactitud de validación durante el entrenamiento.

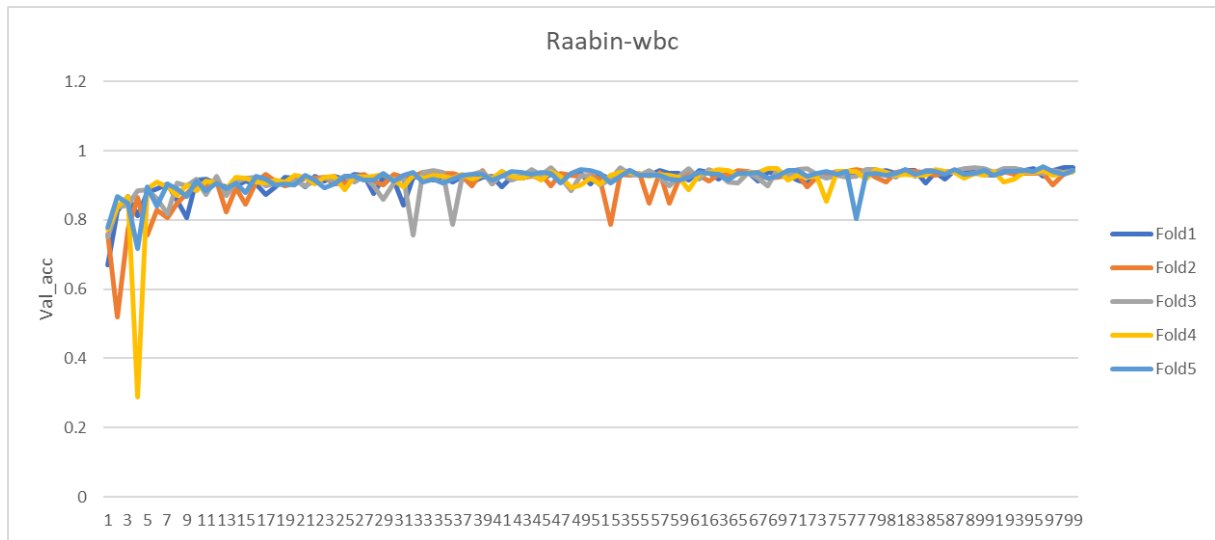


Figura 6.12 Gráficas de exactitud en 5 subconjuntos para validación cruzada en Raabin-wbc.

El comportamiento de las gráficas en los distintos subconjuntos de validación es similar a lo largo del entrenamiento, indicando que el modelo no está adecuándose únicamente a un subconjunto de validación específico. Este mismo comportamiento se observa en las pruebas de validación cruzada utilizando transferencia de aprendizaje y reentrenamiento, tal cómo se verá a continuación.

En la Tabla 6.16 se incluyen los resultados obtenidos en la validación cruzada utilizando transferencia de aprendizaje.

Tabla 6.16 Resultados validación cruzada Raabin-wbc utilizando transferencia de aprendizaje.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Promedio
Acc	0.89157218	0.87382948	0.87561697	0.88592595	0.8808111	0.88155113

En la Figura 6.13 se muestra el comportamiento de la exactitud de validación durante el entrenamiento.

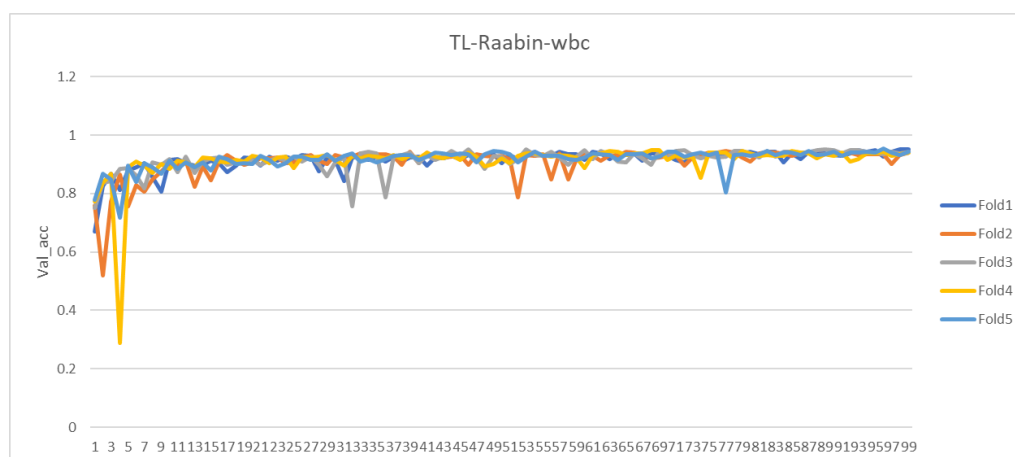


Figura 6.13 Gráficas de exactitud en 5 subconjuntos para validación cruzada en Raabin-wbc utilizando transferencia de aprendizaje.

En la Tabla 6.17 se incluyen los resultados de validación cruzada utilizando reentrenamiento.

Tabla 6.17 Resultados validación cruzada Raabin-wbc utilizando reentrenamiento.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Promedio
Acc	0.97634304	0.97585016	0.97433364	0.97481483	0.97725022	0.97571838

Finalmente, en la Figura 6.14 se observa gráficamente el comportamiento de la exactitud de validación en el reentrenamiento, tomando como punto de partida el modelo con mejor desempeño obtenido para la base de datos de urología.

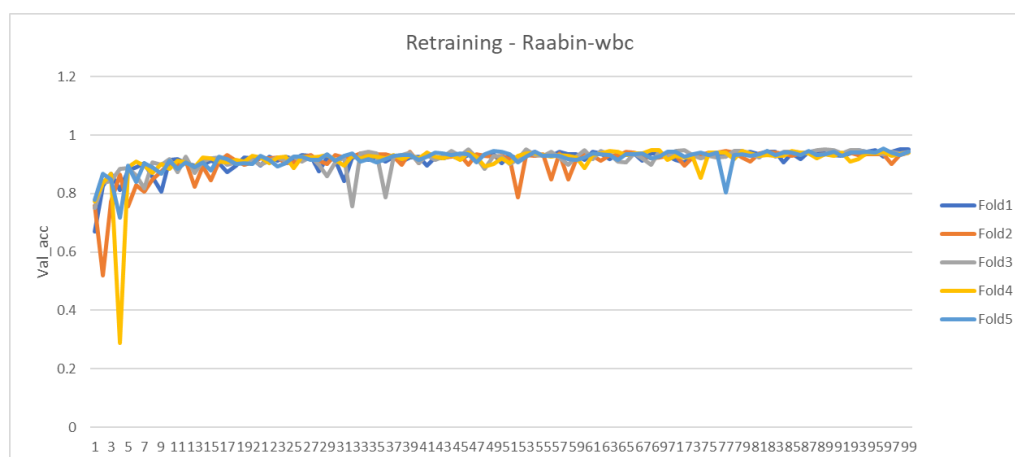


Figura 6.14 Gráficas de exactitud en 5 subconjuntos para validación cruzada en Raabin-wbc utilizando reentrenamiento.

6.3. BYRI

Para obtener los resultados mostrados a continuación, se utilizaron los mismos hiperparámetros establecidos para la base de datos de Raabin-wbc, es decir, se optó por la combinación de la arquitectura propuesta LDE-NetV2 con el algoritmo *entropy gain*, además del optimizador Adam, la función de pérdida de entropía cruzada definida por la ecuación (4.1) y 100 épocas de entrenamiento.

6.3.1. Entrenamiento

Para este caso, se analizaron las 3 clases pertenecientes a la base de datos BYRI: *candida albicans*, *escherichia coli* y *staphylococcus aureus*. En la

Tabla 6.18 se muestran los resultados obtenidos.

Tabla 6.18 Resultados de validación en base de datos BYRI utilizando LDE-NetV2 +EG.

Accuracy	F1	Precision	Recall
0.99147201	0.99148399	0.99147201	0.99147201

En la Figura 6.15 se muestran las gráficas correspondientes a la exactitud y pérdida utilizando la base de datos BYRI.

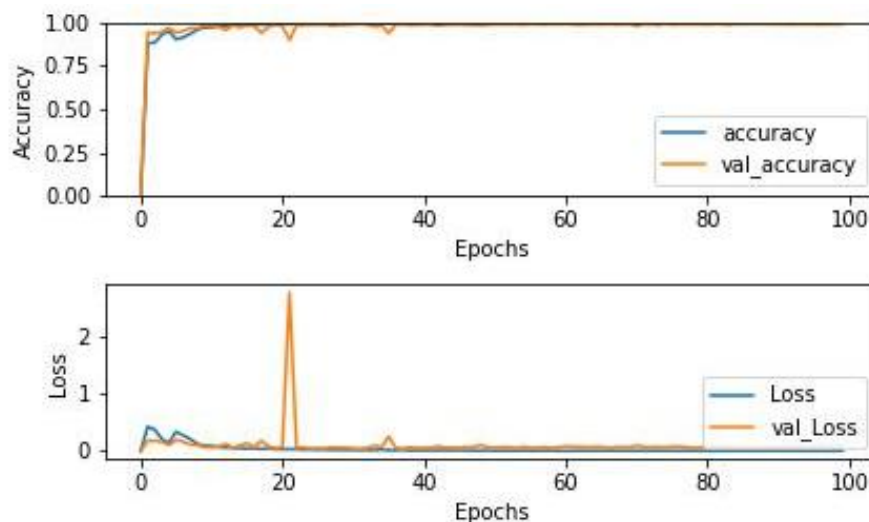


Figura 6.15 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos BYRI.

Estos resultados muestran que el desempeño para la clasificación en la base de datos BYRI es superior al 99%, y el modelo converge desde las primeras épocas del entrenamiento.

CAPÍTULO VI. RESULTADOS

En el mismo sentido, en la Tabla 6.19 se muestran los resultados obtenidos utilizando transferencia de aprendizaje, tomando como modelo previo el modelo con mejor resultado obtenido en los experimentos de la base de datos USD_2, congelando el mismo y entrenando únicamente la capa softmax.

Tabla 6.19 Resultados de validación en base de datos BYRI utilizando transferencia de aprendizaje.

Accuracy	F1	Precision	Recall
0.97487772	0.97500098	0.97516495	0.97480363

En la Figura 6.16 se muestran las gráficas del comportamiento del entrenamiento, utilizando transferencia de aprendizaje.

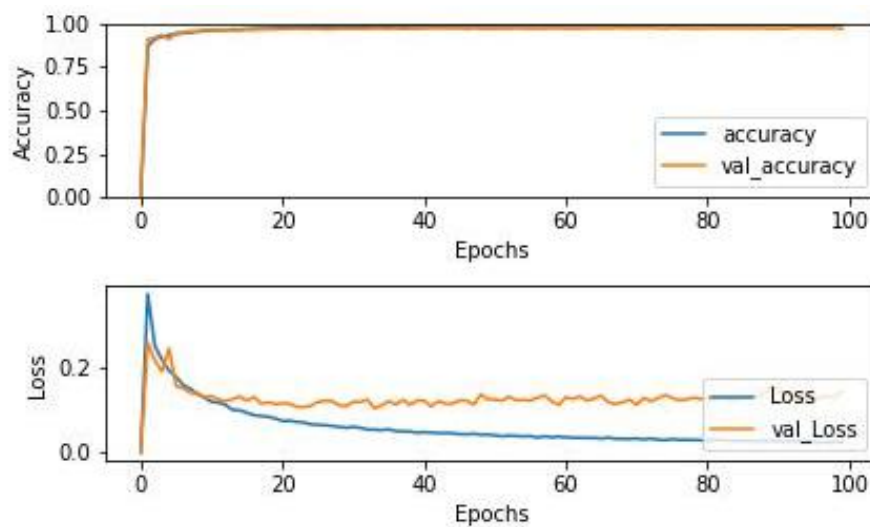


Figura 6.16 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos BYRI utilizando transferencia de aprendizaje.

Se puede observar que el desempeño se reduce al utilizar transferencia de aprendizaje de los conocimientos adquiridos en el entrenamiento con la base de datos de urología. También se puede observar que el efecto del sobreajuste es casi inexistente y que el modelo converge desde épocas tempranas del entrenamiento.

Así mismo, en la Tabla 6.20 se muestran los resultados obtenidos referentes al uso de reentrenamiento, tomando como punto de partida el modelo con el mejor resultado obtenido para USD_2.

Tabla 6.20 Resultados de validación en base de BYRI utilizando reentrenamiento.

Accuracy	F1	Precision	Recall
----------	----	-----------	--------

CAPÍTULO VI. RESULTADOS

0.99125147	0.99126184	0.99132371	0.99125147
------------	------------	------------	------------

Finalmente, en la Figura 6.17 se muestra de manera gráfica el comportamiento del entrenamiento tanto en el conjunto de entrenamiento como el de validación, utilizando reentrenamiento.

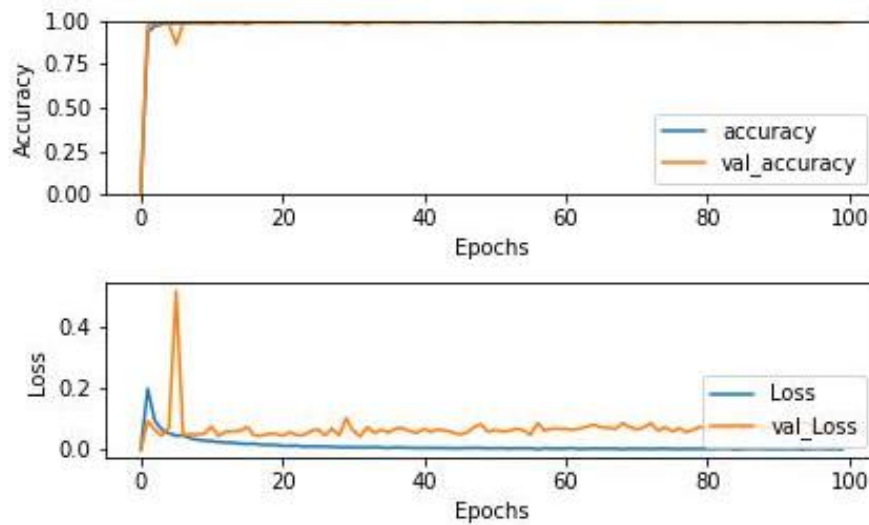


Figura 6.17 Gráficas de exactitud y pérdida en LDE-NetV2 en base de datos BYRI utilizando reentrenamiento.

Se puede observar que el desempeño es muy similar al entrenamiento tradicional, teniendo una ligera disminución en comparación con este. También se puede observar que el efecto del sobreajuste es prácticamente nulo en la mayor parte del entrenamiento y que alcanza una convergencia desde épocas iniciales del mismo.

6.3.2. Validación cruzada

Buscando verificar la generalización del modelo, se realizaron pruebas de validación cruzada con los distintos algoritmos de entrenamiento. En todos los casos se utilizaron 5 subconjuntos de datos.

En la Tabla 6.21 se incluyen los resultados utilizando los distintos subconjuntos de validación, así como el promedio de estos.

Tabla 6.21 Resultados validación cruzada BYRI.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Promedio
Acc	0.99147201	0.98999113	0.99006748	0.98977321	0.99029195	0.99031916

CAPÍTULO VI. RESULTADOS

De manera gráfica, en la Figura 6.18 se muestra el comportamiento de la exactitud de validación durante el entrenamiento.

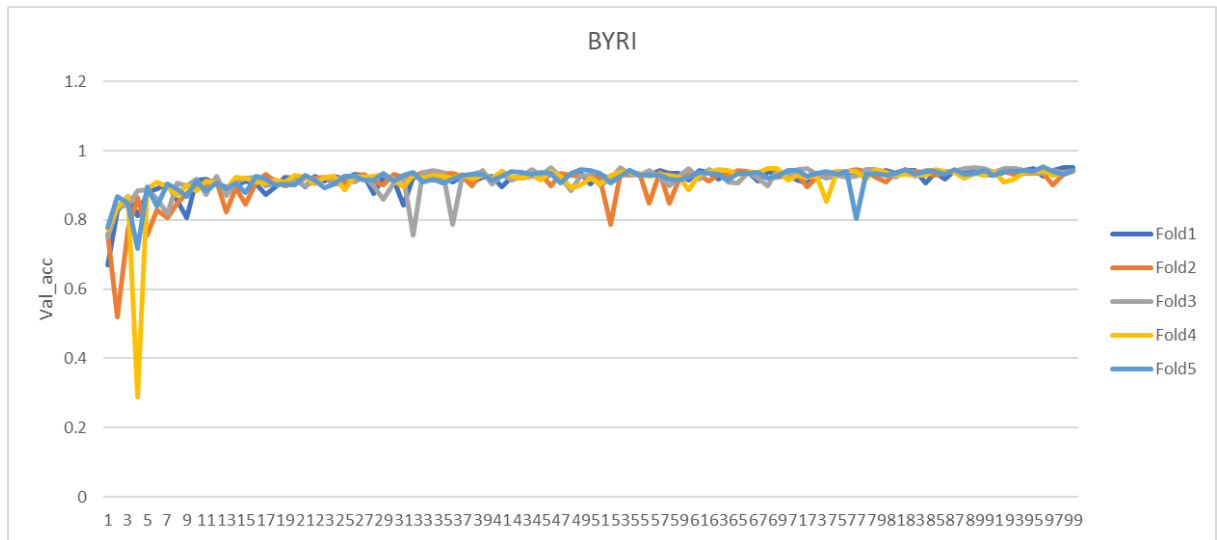


Figura 6.18 Gráficas de exactitud en 5 subconjuntos para validación cruzada en BYRI.

Según lo observado tanto en la Tabla 6.21 como en la Figura 6.18, los resultados obtenidos en exactitud de validación para los distintos subconjuntos son similares en cada uno de ellos. Así mismo, la gráfica muestra un comportamiento semejante a lo largo del entrenamiento. De igual manera que en los casos anteriores, esto indica una correcta generalización del modelo para la base de datos, y no existe indicio relevante de sesgo. Este efecto se presenta de la misma manera en las pruebas realizadas con transferencia de aprendizaje y reentrenamiento, lo cual se puede ver a continuación.

En la Tabla 6.22 se incluyen los resultados obtenidos en la validación cruzada utilizando transferencia de aprendizaje.

Tabla 6.22 Resultados validación cruzada BYRI utilizando transferencia de aprendizaje.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Promedio
Acc	0.97175264	0.9744274	0.97265452	0.97487772	0.97308123	0.9733587

En la Figura 6.19 se muestra el comportamiento de la exactitud de validación durante el entrenamiento.

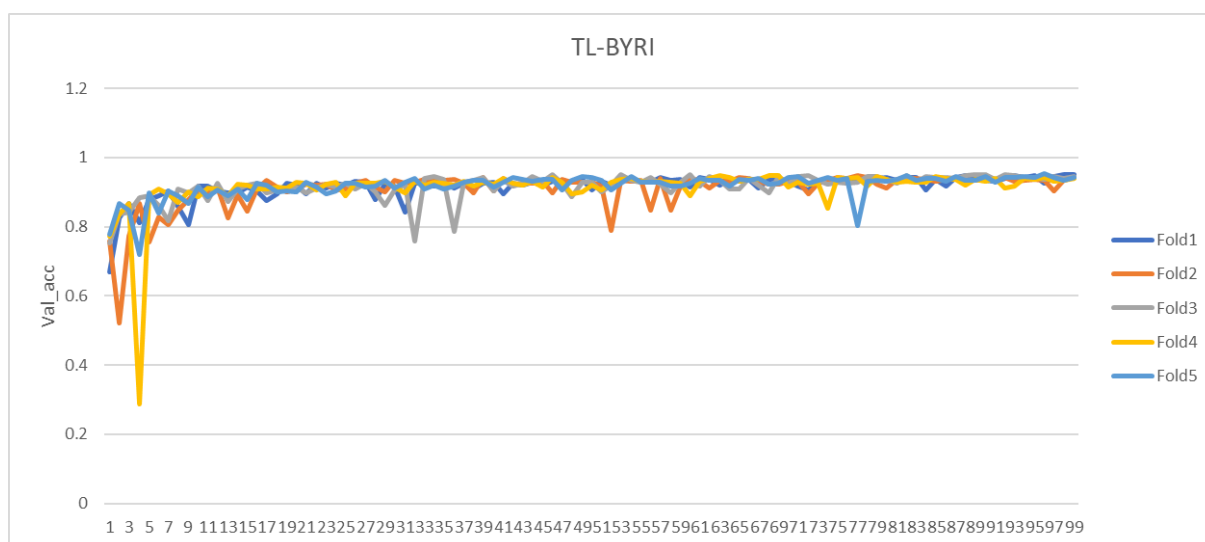


Figura 6.19 Gráficas de exactitud en 5 subconjuntos para validación cruzada en BYRI utilizando transferencia de aprendizaje.

En la Tabla 6.23 se incluyen los resultados de validación cruzada utilizando reentrenamiento.

Tabla 6.23 Resultados validación cruzada BYRI utilizando reentrenamiento.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Promedio
Acc	0.99065626	0.99125147	0.99095696	0.99044019	0.99058843	0.99077866

Finalmente, en la Figura 6.20 se observa gráficamente el comportamiento de la exactitud de validación en el reentrenamiento, tomando como punto de partida el modelo con mejor desempeño obtenido para la base de datos de urología.

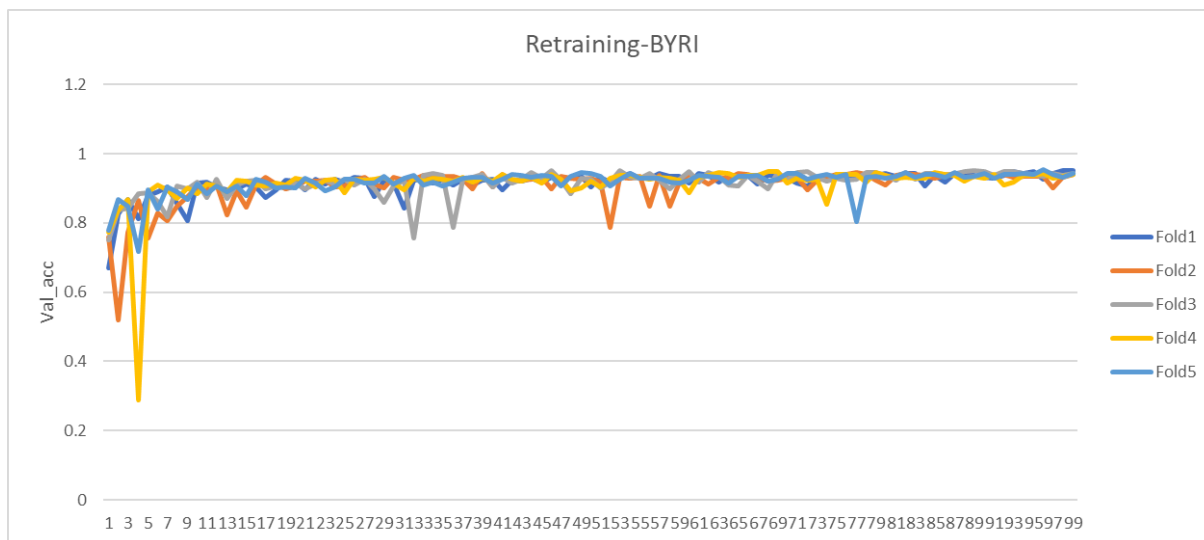


Figura 6.20 Gráficas de exactitud en 5 subconjuntos para validación cruzada en BYRI utilizando reentrenamiento.

7. CONCLUSIONES

El presente capítulo detalla las conclusiones obtenidas a partir del trabajo realizado y los resultados obtenidos. Así mismo, se proponen aspectos que pueden llevar a mejorar los resultados mediante trabajo futuro.

Los hallazgos de esta investigación indican que la reducción de profundidad en el modelo tiene un efecto positivo en el aprendizaje de la red para la tarea de clasificación de muestras microscópicas, al disminuir su capacidad y evitar la memorización. Además, Se observa un aumento en el rendimiento al utilizar el algoritmo Entropy Gain durante el entrenamiento, lo que sugiere una oportunidad para mejorar los resultados aún más.

Por otro lado, el uso de transferencia de aprendizaje no mostró mejoras significativas en los experimentos realizados, incluso llevando a una disminución en el desempeño. Aunque el reentrenamiento demostró resultados prometedores en casos específicos, no compensa la necesidad de un modelo previamente entrenado en términos de eficiencia y recursos.

Así mismo, el modelo es capaz de clasificar con una alta precisión, habiendo logrado reducir la cantidad de parámetros de manera significativa. Pasando de 119 millones de parámetros en EfficientNetV2L a 2.45 millones en LDE-NetV2, manteniendo un desempeño superior al 95% en exactitud en las tres bases de datos analizadas.

En general, las pruebas de validación cruzada indican que las implementaciones realizadas tienen la capacidad de generalizar en diferentes bases de datos, lo que sugiere la robustez y la aplicabilidad de nuestro modelo en diversas situaciones. Estos resultados respaldan la eficacia del enfoque en la reducción de profundidad y el uso de Entropy Gain en la mejora del rendimiento de las redes neuronales en tareas de clasificación.

Como direcciones de trabajo futuro, se pueden considerar las siguientes ideas:

Continuar investigando y realizando pruebas mediante la disminución de la anchura del modelo. Esto permitiría comprender mejor cómo los cambios en la anchura afectan el rendimiento y la eficiencia de las redes neuronales, complementando así los hallazgos actuales sobre la reducción de profundidad.

Emplear pruebas de búsqueda de arquitectura neuronal (NAS) para encontrar la arquitectura más eficiente para tareas específicas sería un paso superior a únicamente explorar la disminución de la anchura. Esto podría llevar a la creación de modelos altamente optimizados que se adapten a las necesidades de aplicaciones específicas.

Investigar a profundidad la aplicación de algoritmos de aprendizaje con pocas muestras y meta aprendizaje para adaptar el modelo a nuevas bases de datos de manera eficiente. Esto podría ampliar la capacidad de generalización del modelo y su capacidad de adaptarse a tareas de clasificación con datos limitados.

CAPÍTULO VII. CONCLUSIONES

Desarrollar la integración de la segmentación de elementos en las imágenes originales de las bases de datos con el modelo desarrollado. Esta sinergia entre la segmentación y la clasificación podría mejorar la precisión y utilidad del sistema en aplicaciones prácticas.

Trabajar en la implementación del sistema mencionado en dispositivos embebidos, con el objetivo de permitir su uso generalizado, especialmente en áreas donde no se disponga de acceso a centros de salud cercanos. Esto podría tener un impacto significativo en la atención médica en entornos con recursos limitados.

Estas direcciones de trabajo futuro tienen el potencial de avanzar en la investigación y la aplicabilidad de modelos de redes neuronales en el campo de la salud y la clasificación de imágenes, al mismo tiempo que abordan cuestiones de eficiencia y accesibilidad en el despliegue de dichos modelos.

8. REFERENCIAS

- [1] C. Aracena, F. Villena, F. Arias, y J. Dunstan, “Aplicaciones de aprendizaje automático en salud”, *Revista Médica Clínica Las Condes*, vol. 33, núm. 6, pp. 568–575, nov. 2022, doi: 10.1016/j.rmcl.2022.10.001.
- [2] B. J. Erickson, P. Korfiatis, Z. Akkus, y T. L. Kline, “Machine learning for medical imaging”, *Radiographics*, vol. 37, núm. 2, pp. 505–515, mar. 2017, doi: 10.1148/RG.2017160130.
- [3] A. Buetti-Dinh *et al.*, “Deep neural networks outperform human expert’s capacity in characterizing bioleaching bacterial biofilm composition”, *Biotechnology Reports*, vol. 22, p. e00321, jun. 2019, doi: 10.1016/J.BTRE.2019.E00321.
- [4] C. L. Chowdhary y D. P. Acharjya, “Segmentation and Feature Extraction in Medical Imaging: A Systematic Review”, *Procedia Comput Sci*, vol. 167, pp. 26–36, 2020, doi: 10.1016/j.procs.2020.03.179.
- [5] S. Albahra *et al.*, “Artificial intelligence and machine learning overview in pathology & laboratory medicine: A general review of data preprocessing and basic supervised concepts”, *Semin Diagn Pathol*, vol. 40, núm. 2, pp. 71–87, mar. 2023, doi: 10.1053/j.semdp.2023.02.002.
- [6] N. Rabbani, G. Y. E. Kim, C. J. Suarez, y J. H. Chen, “Applications of machine learning in routine laboratory medicine: Current state and future directions”, *Clin Biochem*, vol. 103, pp. 1–7, may 2022, doi: 10.1016/j.clinbiochem.2022.02.011.
- [7] Z. Zhu, Z. Ren, S. Lu, S. Wang, y Y. Zhang, “DLBCNet: A Deep Learning Network for Classifying Blood Cells”, *Big Data and Cognitive Computing 2023, Vol. 7, Page 75*, vol. 7, núm. 2, p. 75, abr. 2023, doi: 10.3390/BDCC7020075.
- [8] B. Al-Bander, B. M. Williams, W. Al-Nuaimy, M. A. Al-Tae, H. Pratt, y Y. Zheng, “Dense fully convolutional segmentation of the optic disc and cup in colour fundus for glaucoma diagnosis”, *Symmetry (Basel)*, vol. 10, núm. 4, abr. 2018, doi: 10.3390/SYM10040087.
- [9] K. H. Cha *et al.*, “Bladder Cancer Segmentation in CT for Treatment Response Assessment: Application of Deep-Learning Convolution Neural Network—A Pilot Study”, *Tomography*, vol. 2, núm. 4, pp. 421–429, dic. 2016, doi: 10.18383/j.tom.2016.00184.
- [10] L. Girona y J. Conejero, “Urología - Sociedad Española de Farmacia Hospitalaria”, *Farmacia Hospitalaria*, vol. 24, pp. 1601–1624, 2003.
- [11] V. de M. Campos Otegui, “Guía práctica para la estandarización del procesamiento y examen de las muestras de orina”, *Bio-Rad*, 2007.
- [12] W. J. C. Amend *et al.*, *Smith’s General Urology*, 17a ed. McGraw Hill, 2008.
- [13] D. Técnicos, P. El, y L. Clínico, “RECOMENDACIONES PARA LA INTERPRETACIÓN DEL FROTIS SANGUÍNEO DEL SUBPROGRAMA DE MORFOLOGÍA SANGUÍNEA”.
- [14] P. MÓNICA TORRENS, “INTERPRETACIÓN CLÍNICA DEL HEMOGRAMA”, *Revista Médica Clínica Las Condes*, vol. 26, núm. 6, pp. 713–725, nov. 2015, doi: 10.1016/J.RMCLC.2015.11.001.
- [15] C. B. Naranjo Aristizábal, *Atlas de hematología células sanguíneas*, 2nd ed. Manizales: Universidad Católica de Manizales, 2008.
- [16] J. H. Carr y B. F. Rodak, *Atlas de Hematología Clínica*, 3rd ed. Editorial Médica Panamericana, 2010.
- [17] Z. M. Kouzehkanan *et al.*, “Raabin-WBC: a large free access dataset of white blood cells from normal peripheral blood”, *bioRxiv*, p. 2021.05.02.442287, ene. 2021, doi: 10.1101/2021.05.02.442287.
- [18] F. Fernández Riverón, J. López Hernández, L. M. Ponce Martínez, y C. Machado Betarte, “Resistencia bacteriana”, *Revista Cubana de Medicina Militar*, vol. 32, 2003.
- [19] P. Gutiérrez Delgado, L. Iglesias Zaldivar, L. Torres Morales, C. Rodríguez Martínez, R. Zhurbenko, y E. González Lomba, “Evaluación de las Bases Nutritivas BioCen en los Medios para Propósito General”, en *Primer Taller Nacional de Medios de Cultivo*, La Habana, Cuba, 1993.
- [20] C. Rodríguez Martínez y R. Zhurbenko, *Manual de medios de cultivo*, 4th ed. Bejucal: Centro Nacional de Biopreparados, 2018.
- [21] E. Cervantes-García, R. García-González, y P. María Salazar-Schettino, “Características generales del *Staphylococcus aureus*”, *Rev Latinoam Patol Clin Med Lab*, vol. 61, núm. 1, pp. 28–40, 2014, Consultado: el 13 de septiembre de 2023. [En línea]. Disponible en: www.medigraphic.com/patologiaclinicawww.medigraphic.org.mx

8. REFERENCIAS

- [22] J. J. Camarena y R. Sánchez, “INFECCIÓN POR *Staphylococcus aureus* RESISTENTE A METICILINA”.
- [23] J. A. Cortés y G. Prada, “Protocolo de estudio y manejo de pacientes con candidiasis sistémica en adultos”, *Infectio*, vol. 16, pp. 118–122, dic. 2012, doi: 10.1016/S0123-9392(12)70037-1.
- [24] M. Nucci, F. Queiroz-Telles, A. M. Tobón, A. Restrepo, y A. L. Colombo, “Epidemiology of opportunistic fungal infections in latin America”, *Clinical Infectious Diseases*, vol. 51, núm. 5, pp. 561–570, sep. 2010, doi: 10.1086/655683.
- [25] G. Rodríguez Angeles, “Principales características y diagnóstico de los grupos patógenos de *Escherichia coli*”, *Salud Publica Mex*, vol. 44, sep. 2002.
- [26] S. Albawi, T. A. Mohammed, y S. Al-Zawi, “Understanding of a convolutional neural network”, en *2017 International Conference on Engineering and Technology (ICET)*, IEEE, ago. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.
- [27] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, y R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors”, jul. 2012, [En línea]. Disponible en: <http://arxiv.org/abs/1207.0580>
- [28] Y. Liang, Z. Tang, M. Yan, y J. Liu, “Object detection based on deep learning for urine sediment examination”, *Biocybern Biomed Eng*, vol. 38, núm. 3, pp. 661–670, ene. 2018, doi: 10.1016/J.BBE.2018.05.004.
- [29] Y. Liang, R. Kang, C. Lian, y Y. Mao, “An End-to-End System for Automatic Urinary Particle Recognition with Convolutional Neural Network”, *J Med Syst*, vol. 42, núm. 9, sep. 2018, doi: 10.1007/S10916-018-1014-6.
- [30] M. Yan, Q. Liu, Z. Yin, D. Wang, y Y. Liang, “A Bidirectional Context Propagation Network for Urine Sediment Particle Detection in Microscopic Images”, en *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, may 2020, pp. 981–985. doi: 10.1109/ICASSP40776.2020.9054367.
- [31] Z. M. Kouzehkanan *et al.*, “A large dataset of white blood cells containing cell locations and types, along with segmented nuclei and cytoplasm”, *Scientific Reports 2022 12:1*, vol. 12, núm. 1, pp. 1–14, ene. 2022, doi: 10.1038/s41598-021-04426-x.
- [32] E. A. Salazar González, “Método de aprendizaje de máquina embebido basado en redes autoencoder para detección de microorganismos”, Masther Thesis, Tecnm / Instituto Tecnológico de Chihuahua, Chihuahua, 2023.
- [33] Ó. Gálvez Borjas, “Estudio y selección de algoritmos de súper resolución para imágenes a color en diversas aplicaciones”, Colegio de Ciencia y Tecnología, Ciudad de México, 2016.
- [34] E. Shelhamer, J. Long, y T. Darrell, “Fully Convolutional Networks for Semantic Segmentation”, *IEEE Trans Pattern Anal Mach Intell*, vol. 39, núm. 4, pp. 640–651, abr. 2017, doi: 10.1109/TPAMI.2016.2572683.
- [35] J. Ji, X. Lu, M. Luo, M. Yin, Q. Miao, y X. Liu, “Parallel Fully Convolutional Network for Semantic Segmentation”, *IEEE Access*, vol. 9, pp. 673–682, 2021, doi: 10.1109/ACCESS.2020.3042254.
- [36] S. Wen, W. Tian, H. Zhang, S. Fan, N. Zhou, y X. Li, “Semantic Segmentation Using a GAN and a Weakly Supervised Method Based on Deep Transfer Learning”, *IEEE Access*, vol. 8, pp. 176480–176494, 2020, doi: 10.1109/ACCESS.2020.3026684.
- [37] K. Cui, P. Fu, Y. Li, y Y. Lin, “Bayesian Fully Convolutional Networks for Brain Image Registration”, *J Healthc Eng*, vol. 2021, pp. 1–12, jul. 2021, doi: 10.1155/2021/5528160.
- [38] M. Hu, Y. Zhong, S. Xie, H. Lv, y Z. Lv, “Fuzzy System Based Medical Image Processing for Brain Disease Prediction”, *Front Neurosci*, vol. 15, 2021, [En línea]. Disponible en: <https://www.frontiersin.org/articles/10.3389/fnins.2021.714318>
- [39] S. Gobhinath, S. Anandkumar, R. Dhayalan, P. Ezhilbharathi, y R. Haridharan, “Human Brain Tumor Detection and Classification by Medical Image Processing”, en *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, IEEE, mar. 2021, pp. 561–564. doi: 10.1109/ICACCS51430.2021.9441877.
- [40] P. P. Banik, R. Saha, y K.-D. Kim, “Fused Convolutional Neural Network for White Blood Cell Image Classification”, en *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, IEEE, feb. 2019, pp. 238–240. doi: 10.1109/ICAIIIC.2019.8669049.

8. REFERENCIAS

- [41] A. Van Opbroek, H. C. Achterberg, M. W. Vernooij, y M. De Bruijne, “Transfer Learning for Image Segmentation by Combining Image Weighting and Kernel Learning”, *IEEE Trans Med Imaging*, vol. 38, núm. 1, pp. 213–224, ene. 2019, doi: 10.1109/TMI.2018.2859478.
- [42] Y. Zhou, C. Zhang, y S. Gao, “Breast Cancer Classification From Histopathological Images Using Resolution Adaptive Network”, *IEEE Access*, vol. 10, pp. 35977–35991, 2022, doi: 10.1109/ACCESS.2022.3163822.
- [43] C. Wang, J. Zhang, y S. Liu, “Medical Ultrasound Image Segmentation With Deep Learning Models”, *IEEE Access*, vol. 11, pp. 10158–10168, 2023, doi: 10.1109/ACCESS.2022.3225101.
- [44] M. Shen y R. Zhang, “Urine Sediment Recognition Method Based on SVM and AdaBoost”, en *2009 International Conference on Computational Intelligence and Software Engineering*, IEEE, dic. 2009, pp. 1–4. doi: 10.1109/CISE.2009.5365881.
- [45] Q. Ji, X. Li, Z. Qu, y C. Dai, “Research on Urine Sediment Images Recognition Based on Deep Learning”, *IEEE Access*, vol. 7, pp. 166711–166720, 2019, doi: 10.1109/ACCESS.2019.2953775.
- [46] W. Tangsuksant, C. Pintavirooj, S. Taertulakarn, y S. Daochai, “Development algorithm to count blood cells in urine sediment using ANN and Hough Transform”, en *The 6th 2013 Biomedical Engineering International Conference*, IEEE, oct. 2013, pp. 1–4. doi: 10.1109/BMEiCon.2013.6687725.
- [47] C. Kaul, S. Manandhar, y N. Pears, “Focusnet: An Attention-Based Fully Convolutional Network for Medical Image Segmentation”, en *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, IEEE, abr. 2019, pp. 455–458. doi: 10.1109/ISBI.2019.8759477.
- [48] C. C. Hortinela, J. R. Balbin, J. C. Fausto, y K. K. Viray, “Identification of Crystals Present in a Urine Sediment based on Adaptive Boosting Algorithm”, en *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, IEEE, nov. 2019, pp. 1–4. doi: 10.1109/HNICEM48295.2019.9072865.
- [49] J. M. H. Noothout *et al.*, “Deep Learning-Based Regression and Classification for Automatic Landmark Localization in Medical Images”, *IEEE Trans Med Imaging*, vol. 39, núm. 12, pp. 4011–4022, dic. 2020, doi: 10.1109/TMI.2020.3009002.
- [50] H. Xiang *et al.*, “Urine Calcium Oxalate Crystallization Recognition Method Based on Deep Learning”, en *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, IEEE, abr. 2019, pp. 30–33. doi: 10.1109/ICACTM.2019.8776769.
- [51] E. O. Fernandez *et al.*, “Microcontroller-Based Automated Microscope for Image Recognition of Four Urine Constituents”, en *TENCON 2018 - 2018 IEEE Region 10 Conference*, IEEE, oct. 2018, pp. 1689–1694. doi: 10.1109/TENCON.2018.8650102.
- [52] T. Li *et al.*, “The image-based analysis and classification of urine sediments using a LeNet-5 neural network”, *Comput Methods Biomech Biomed Eng Imaging Vis*, vol. 8, núm. 1, pp. 109–114, ene. 2020, doi: 10.1080/21681163.2019.1608307.
- [53] C. Li, Y. Y. Tang, H. Luo, y X. Zheng, “Join Gabor and scattering transform for urine sediment particle texture analysis”, en *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, IEEE, jun. 2015, pp. 410–415. doi: 10.1109/CYBConf.2015.7175969.
- [54] R. Kang, Y. Liang, C. Lian, y Y. Mao, “CNN-Based Automatic Urinary Particles Recognition”, *J Med Syst*, vol. 42, núm. 9, mar. 2018, doi: 10.1007/s10916-018-1014-6.
- [55] Y.-M. Li y X.-P. Zeng, “A new strategy for urinary sediment segmentation based on wavelet, morphology and combination method”, *Comput Methods Programs Biomed*, vol. 84, núm. 2–3, pp. 162–173, dic. 2006, doi: 10.1016/j.cmpb.2006.07.010.
- [56] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, abr. 2017, [En línea]. Disponible en: <http://arxiv.org/abs/1704.04861>
- [57] E. T. Coletto, J. Rodríguez, V. Carlos, S. Pascual, y C. Cervera, “Métodos de aprendizaje profundo para la segmentación semántica de personas”, 2020.
- [58] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, y L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, ene. 2018, [En línea]. Disponible en: <http://arxiv.org/abs/1801.04381>
- [59] G. Huang, Z. Liu, L. van der Maaten, y K. Q. Weinberger, “Densely Connected Convolutional Networks”, ago. 2016, [En línea]. Disponible en: <http://arxiv.org/abs/1608.06993>
- [60] M. Tan y Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, may 2019, [En línea]. Disponible en: <http://arxiv.org/abs/1905.11946>

8. REFERENCIAS

- [61] M. Tan y Q. V. Le, “EfficientNetV2: Smaller Models and Faster Training”, abr. 2021, [En línea]. Disponible en: <http://arxiv.org/abs/2104.00298>
- [62] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge”, sep. 2014.
- [63] D. P. Kingma y J. Ba, “Adam: A Method for Stochastic Optimization”, dic. 2014.
- [64] T. Dozat, “Incorporating Nesterov Momentum into Adam”, 2016, Consultado: el 26 de noviembre de 2023. [En línea]. Disponible en: <http://mattmahoney.net/dc/text8.zip>
- [65] M. A. Camacho, “Clasificación morfológica de galaxias mediante aprendizaje profundo”, M.C. Thesis, ITCH, Chihuahua, 2022.
- [66] X. Glorot y Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, en *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*., Society for Artificial Intelligence and Statistics, 2010.