UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBAJO FINAL DE GRADO

Grado en Ingeniería Biomédica

DETECCIÓN Y CLASIFICACIÓN DE DIFERENTES FORMAS ERITROCITARIAS ANÓMALAS MEDIANTE REDES NEURONALES PROFUNDAS.

Memoria y Anexos

Autor: Justel Pizarro, Joaquín Tomás
Director: Alférez Baquero, Edwin Santiago
Co-Director: Rodellar Benede, José Julián

Convocatoria: Mayo 2019

Resum

La observació de la morfologia eritrocitària, i per tant, la detecció de possibles alteracions eritrocitàries té un gran valor diagnòstic, ja que pot ser indici de diferents patologies com poden ser les anèmies.

Aquestes alteracions poden ser de diversos tipus, alteracions en la seva grandària, en la seva forma, o en la seva coloració hemoglobínica, així com també presentar diferents tipus d'inclusions.

El procés d'observació i classificació d'aquestes cèl·lules requereix de personal qualificat i amb experiència en el camp, ja que, en moltes ocasions, la diferència entre les diferents alteracions és molt subtil i el diagnòstic depèn d'elles.

Aquest treball pretén, gràcies al desenvolupament de tècniques d'*aprenentatge profund* del últims anys, aconseguir un algoritme automatitzat amb la capacitat per classificar adequadament i amb fiabilitat imatges de diferents tipus de cèl·lules sanguínies, en concret plaquetes, eritròcits i algunes de les seves principals alteracions morfològiques (esferòcits , drepanòcits, reticulòcits, cossos de Pappenheimer, cossos Howell-Jolly i puntejat basòfil).

S'ha utilitzat la tècnica de *transferència d'aprenentatge* amb fast.ai, que, en lloc d'entrenar una Xarxa Neuronal Convolucional des de zero, utilitza xarxes ja entrenades com a punt d'inici.

El conjunt d'imatges amb les que s'ha entrenat el model ha estat proporcionat per part del Laboratori Core de l'Hospital Clínic.

S'ha arribat a obtenir una precisió en la classificació de més del 96% amb tres models diferents, ResNet 34. ResNet 50 i DenseNet 169, i més del 97 % amb DenseNet 121. Per tant, l'ús d'aquestes tècniques dona un resultat prou satisfactori i pot ser un eina per ajudar al personal responsable.

Paraules clau:

Eritròcits, esferòcits , drepanòcits, reticulòcits, cossos de Pappenheimer, cossos Howell-Jolly i puntejat basòfil, xarxa neuronal convolucional, deep learning, aprenentatge automàtic, transfer learning, Fast.ai.

Resumen

La observación de la morfología eritrocitaria y, por ende, la detección de posibles alteraciones eritrocitarias, tiene un gran valor diagnóstico, ya que puede ser indicio de diferentes patologías como pueden ser las anemias.

Estas alteraciones pueden ser de diversos tipos, alteraciones en su tamaño, en su forma, o en su coloración hemoglobínica, así como también presentar diferentes tipos de inclusiones.

El proceso de observación y clasificación requiere de personal cualificado y con experiencia en el campo, ya que en muchas ocasiones la diferencia entre las distintas alteraciones es muy sutil, y el diagnóstico depende de ellas.

Este trabajo pretende, gracias al desarrollo de técnicas de aprendizaje profundo de los últimos años, conseguir un algoritmo automatizado con la capacidad para clasificar adecuadamente y con fiabilidad imágenes de diferentes tipos células sanguíneas, en concreto plaquetas, eritrocitos y algunas de sus principales alteraciones morfológicas (esferocitos, drepanocitos, reticulocitos, cuerpos de Pappenheimer, cuerpos Howell-Jolly y punteado basófilo).

Se ha utilizado la técnica de transferencia de aprendizaje con fast.ai, que, en lugar de entrenar una Red Neuronal Convolucional desde cero, utiliza redes ya entrenadas como punto de inicio.

El conjunto de imágenes con las que se ha entrenado el modelo ha sido proporcionado por parte del Laboratorio Core del Hospital Clínic.

Se ha llegado a obtener una precisión en la clasificación de más del 96 % con tres modelos diferentes, ResNet 34, ResNet 50 y DenseNet 169, y más del 97 % con DenseNet 121. Por tanto, el uso de estas técnicas da un resultado bastante satisfactorio y puede ser una buena herramienta de apoyo para el personal responsable.

Palabras clave:

Eritrocito, esferocito, drepanocito, reticulocito, cuerpos de Pappenheimer, cuerpos Howell-Jolly, punteado basófilo, red neuronal convolucional, deep learning, aprendizaje automático, transfer learning, Fast.ai.

Abstract

The observation of the erythrocyte morphology and, therefore, the detection of possible erythrocyte alterations has a great diagnostic value, since it can be indicative of different pathologies such as anemia.

These alterations can be of different types, alterations in their size, in their shape, or in their hemoglobin coloration, as well as presenting different types of inclusions.

The process of observation and classification requires qualified personnel with experience in the field, since in many cases the difference between the different alterations is very subtle and the diagnosis depends on them.

This work aims, thanks to the development of Deep Learning techniques of recent years, to achieve an automated algorithm with the ability to adequately and reliably classify images of different blood cells, specifically platelets, erythrocytes and some of their main morphological alterations (spherocytes, sickle cells, reticulocytes, Pappenheimer bodies, Howell-Jolly bodies and basophilic stippling).

It has been used the Transfer Learning technique with fast.ai, which, instead of training a Convolutional Neural Network from scratch, uses already trained networks as a starting point.

The set of images with which the model has been trained has been provided by the Core Laboratory of the Hospital Clínic.

It has been possible to obtain a classification accuracy of more than 96 % with three different models, ResNet 34, Resnet 50 and DenseNet 169, and 97 % with DenseNet 121. Therefore, the use of these techniques gives a very satisfactory result and can be a good support tool for the responsible staff.

Keywords:

Erythrocyte, spherocyte, sickle cell, reticulocyte, Pappenheimer bodies, Howell-Jolly bodies, basophilic stippling, convolutional neural network, deep learning, machine learning, transfer learning, Fast.ai.

Agradecimientos

El presente trabajo de fin de grado ha sido realizado bajo la supervisión de Santiago Alférez Baquero y José Rodellar Benede, a quienes me gustaría expresar mi más sincero agradecimiento por su paciencia, tiempo y dedicación, y por haberme dado a conocer el mundo del Deep Learning.

También agradecer al grupo del Hospital Clínic, en especial a la doctora Anna Merino y a Ángel Molina por la aportación de las imágenes que han hecho posible llevar a cabo este trabajo.

Glosario

CNN	Red Neuronal Convolucional
ConvNet	Red Neuronal Convolucional
Deep Learning	Aprendizaje profundo
ReLU	Función de pérdida que pone los valores negativos a cero

Índice

RESUM I

RESUMEN II

ABSTRACT III

AGRADECIMIENTOS IV

GLOSARIO V

ÍNDICE VI

ÍNDICE DE TABLAS Y ECUACIONES IX

ÍNDICE DE FIGURAS XI

1. PREFACIO 1

1.2. Origen del trabajo 1

1.3. Motivación 1

1.4. Requisitos previos 1

2. INTRODUCCIÓN 3

2.1. Objetivos del trabajo 3

3. BASES BIOLÓGICAS 5

3.1. Sistema circulatorio 5

3.2. La sangre 6

3.2.1. Funciones de la sangre 6

3.2.2. Componentes de la sangre 6

3.3. Células sanguíneas 7

3.3.1. Hematopoyesis 8

3.3.2. Eritrocitos 9

3.3.3. Leucocitos 10

3.3.4. Plaquetas 11

3.4. Alteraciones morfológicas eritrocitarias 12

3.4.1. Deformaciones 13

3.4.2. Inclusiones 15

4. APRENDIZAJE AUTOMÁTICO Y APRENDIZAJE PROFUNDO _____ 18

4.1. Redes neuronales 19

4.1.1. Funciones de activación 21

4.1.2. Aprendizaje de las redes neuronales: Back-propagation 22

4.1.3. Política de un ciclo (*one cycle policy*) 24

4.2. Redes neuronales convolucionales 25

4.2.1. Convolución 26

4.2.2. ReLU 27

4.2.3. Agrupación (*Pooling*) 27

4.2.4. Regularización 28

4.2.5. Clasificación. Capa totalmente conectada 28

4.2.6. Visualizando las ConvNets 29

4.3. Métodos propuestos: *transfer learning* 29

4.3.1. CNN como extractor fijo de características 30

4.3.2. Fine-Tuning 31

4.3.3. Modelos pre entrenados 31

5. CONJUNTO DE DATOS _____ 35

5.1. Distribución del conjunto de datos 35

5.2. Análisis del conjunto de datos 36

5.2.1. Overfitting y underfitting 36

5.2.2. Conjunto de entrenamiento y validación 37

6. MODELOS CREADOS: TRANSFER LEARNING _____ 39

6.1. ConvNet como extractor fijo de características 39

6.1.1. ResNet 42

6.1.2. Alexnet 49

6.1.3. VGG 51

6.1.4. Densenet 56

6.2. Fine tuning 61

6.2.1. ResNet 63

6.2.2. AlexNet 69

6.2.3. VGG16 71

6.2.4. DenseNet 76

6.3. Resultados 80

6.4. Aplicación web 82

7. HARDWARE Y SOFTWARE **84**

8. ANÁLISIS DEL IMPACTO AMBIENTAL **85**

CONCLUSIONES **86**

ANÁLISIS ECONÓMICO **88**

Servicios 88

Personal 88

Total 88

BIBLIOGRAFÍA **89**

Índice de tablas y ecuaciones

Tabla 1 Clasificación del conjunto de clases. 35

Tabla 2. Entrenamiento ResNet 43

Tabla 3. Clases más confundidas Resnet 18 48

Tabla 4. Clases más confundidas ResNet 34 48

Tabla 5. Clases más confundidas ResNet 50 48

Tabla 6. Entrenamiento AlexNet 49

Tabla 7. Clases más confundidas AlexNet 51

Tabla 8 Entrenamiento VGG 51

Tabla 9. Clases más confundidas VGG16 55

Tabla 10. Clases más confundidas VGG19 55

Tabla 11. Entrenamiento DenseNet 56

Tabla 12. Clases más confundidas DenseNet 121 60

Tabla 13. Clases más confundidas DenseNet 169 60

Tabla 14 Entrenamiento ResNet Fine Tuning 63

Tabla 15. Clases más confundidas ResNet 18 Fine Tuning 68

Tabla 16. Clases más confundidas ResNet 34 Fine Tuning 68

Tabla 17. Clases más confundidas ResNet 50 Fine Tuning 68

Tabla 18 Entrenamiento AlexNet Fine Tuning 69

Tabla 19. Clases más confundidas AlexNet Fine Tuning 71

Tabla 20 Entrenamiento VGG Fine Tuning 71

Tabla 21. Clases más confundidas VGG16 Fine Tuning 75

Tabla 22. Clases más confundidas VGG19 Fine Tuning 75

Tabla 23. Entrenamiento DenseNet Fine Tuning 76

Tabla 24 Clases más confundidas DenseNet121 Fine Tuning 79

Tabla 25 Clases más confundidas DenseNet169 Fine Tuning 80

Tabla 26. Resumen resultados de los modelos 81

Tabla 27. Presupuesto servicios 88

Tabla 28. Presupuesto mano de obra 88

Tabla 29. Presupuesto total 88

Ecuación 1 9

Ecuación 2 Sigmoide 21

Ecuación 3 Tanh(x) 21

Ecuación 4 ReLU 22

Índice de figuras

Figura 3.1 Diagrama del sistema circulatorio. (Fuente:(11)) _____ 5

Figura 3.2 Composición de la sangre. (Fuente:(13)) _____ 7

Figura 3.3 Hematopoyesis (Fuente:(14)) _____ 8

Figura 3.4 Eritrocitos normales (Fuente: (15)) _____ 10

Figura 3.5 Eritropoyesis. (Fuente: (16)) _____ 10

Figura 3.6 Eritropoyesis. (Fuente:(16)) _____ 11

Figura 3.7 Plaquetas junto a eritrocitos (Fuente: (17)) _____ 12

Figura 3.8 Abundantes esferocitos en caso de esferocitosis hereditaria. (Fuente: (20)) _____ 14

Figura 3.9 Drepanocitos o eritrocitos falciformes. (Fuente: (22)) _____ 14

Figura 3.10 Reticulocitos. (Fuente:(23)) _____ 15

Figura 3.11 Eritrocito joven con punteado basófilo. (Fuente: (18) p. 18) _____ 16

Figura 3.12 Eritrocito con cuerpos de Pappenheimer. (Fuente: (24) Cap. 3, p. 96) _____ 16

Figura 3.13 Eritrocitos con cuerpos de Howell-Jolly. (Fuente: (18) p. 20) _____ 17

Figura 4.1 Machine learning. (Fuente: (1) Cap. 1, p. 5) _____ 18

Figura 4.2 Enfoque machine learning vs deep learning. (Fuente: (25)) _____ 19

Figura 4.3 Representación del modelo matemático de una neurona. (Fuente: (26)) _____ 20

Figura 4.4 Ejemplo de red neuronal de 3 capas. (Fuente:(26)) 20

Figura 4.5 Función de activación sigmoide. 21

Figura 4.6 Función de activación tanh. 22

Figura 4.7. Función de activación ReLU. 22

Figura 4.8 Esquema del funcionamiento de una red neuronal. (Fuente: (1) Cap.1, p.11) 23

Figura 4.9 Learning Rate (Fuente:(28)) 24

Figura 4.10 One cycle (evolución learning rate). Fuente: (3) 24

Figura 4.11 Esquema de una ConvNet. (Fuente:(33)) 25

Figura 4.12 Convolución (Fuente: (34)) 26

Figura 4.13 Ejemplo de aplicación de ReLU. (Fuente: (35)) 27

Figura 4.14 Max Pooling. (Fuente: (36)(37)) 27

Figura 4.15 Ejemplo de una ConvNet para clasificar números escritos a mano. (Fuente: (39)) 28

Figura 4.16. Visualización primera capa CNN. (Fuente: (40)) 29

Figura 4.17. Cambio de clasificador en Transfer Learning.(Fuente: (1) Cap. 5, p. 144) 30

Figura 4.18 Residual block. (Fuente: (45)) 32

Figura 4.19 Arquitectura ResNet34. (Fuente:(45)) 32

Figura 4.20 Arquitetura DenseNet. (Fuente:(6)) 33

Figura 4.21. Arquitectura AlexNet. (Fuente:(45)) 33

Figura 4.22 Arquitectura VGG16 vs VGG19. (Fuente:(51)) 34

Figura 5.1 Overfitting y underfitting (Fuente:(53)) 37

Figura 6.1. Evolución de pérdidas ResNet 18 44

Figura 6.2. Evolución de pérdidas ResNet 34 44

Figura 6.3. Evolución de pérdidas ResNet 50 44

Figura 6.4. Imágenes con mayor pérdida ResNet 18 45

Figura 6.5. Imágenes con mayores pérdidas ResNet 34 45

Figura 6.6. Imágenes con mayores pérdidas ResNet 50 46

Figura 6.7 Matriz de Confusión ResNet 18. 46

Figura 6.8. Matriz de Confusión ResNet 34 47

Figura 6.9. Matriz de Confusión ResNet 50 47

Figura 6.10. Evolución de pérdidas AlexNet 50

Figura 6.11. Matriz de Confusión AlexNet 50

Figura 6.12. Evolución de pérdidas VGG16 52

Figura 6.13. Evolución de pérdidas VGG19 52

Figura 6.14. Imágenes con mayores pérdidas VGG16 53

Figura 6.15. Imágenes con mayores pérdidas VGG19 53

Figura 6.16. Matriz de confusión VGG16 54

Figura 6.17. Matriz de Confusión VGG19 54

Figura 6.18. Evolución pérdidas DenseNet 121 57

Figura 6.19. Evolución pérdidas DenseNet 169 57

Figura 6.20. Imágenes con mayor pérdida DenseNet 121 58

Figura 6.21. Imágenes con mayor pérdida DenseNet 169 58

Figura 6.22. Matriz de Confusión DenseNet 121 59

Figura 6.23 Matriz de Confusión DenseNet 169 59

Figura 6.24 Evolución pérdida - Learning rate 61

Figura 6.25. Evolución pérdidas ResNet 18 Fine Tuning 63

Figura 6.26. Evolución pérdidas ResNet34 Fine Tuning 64

Figura 6.27 Evolución pérdidas ResNet50 Fine Tuning 64

Figura 6.28. Imágenes con mayores pérdidas ResNet18 Fine Tuning 65

Figura 6.29. Imágenes con mayores pérdidas ResNet 34 Fine Tuning 65

Figura 6.30 Imágenes con mayores pérdidas ResNet50 Fine Tuning 66

Figura 6.31 Matriz de confusión ResNet18 FineTuning 66

Figura 6.32 Matriz de Confusión ResNet34 Fine Tuning 67

Figura 6.33. Matriz de confusión ResNet50 Fine Tuning 67

Figura 6.34. Evolución pérdidas AlexNet Fine Tuning 69

Figura 6.35 Imágenes con mayores pérdidas AlexNet Fine Tuning 70

Figura 6.36 Matriz de confusión AlexNet Fine Tuning 70

Figura 6.37Evolución pérdidas VGG16 Fine Tuning _____ 72

Figura 6.38 Evolución pérdidas VGG19 Fine Tuning_____ 72

Figura 6.39 Imágenes con mayores pérdidas VGG16 Fine Tuning _____ 73

Figura 6.40 Imágenes con mayores pérdidas VGG19 Fine Tuning _____ 73

Figura 6.41 Matriz de confusión VGG16 Fine Tuning _____ 74

Figura 6.42 Matriz de confusión VGG19 Fine Tuning _____ 74

Figura 6.43 Evolución pérdidas DenseNet121 Fine Tuning _____ 76

Figura 6.44 Evolución pérdidas DenseNet 169 Fine Tuning _____ 77

Figura 6.45 Imágenes con mayores pérdidas DenseNet121 Fine Tuning_____ 77

Figura 6.46 Imágenes con mayores pérdidas DenseNet169 Fine Tuning_____ 78

Figura 6.47 Matriz de confusión DenseNet121 Fine Tuning _____ 78

Figura 6.48 Matriz de confusión DenseNet169 Fine Tuning _____ 79

Figura 6.49 Primera prueba WebApp. Fuente imagen: (58) _____ 83

Figura 6.50 Segunda prueba WebApp. Fuente imagen: (59) _____ 83

1. Prefacio

1.2. Origen del trabajo

El origen de este trabajo viene de querer hacer un método de clasificación automática de imágenes de diferentes tipos de células sanguíneas, en este caso de la serie eritrocitaria principalmente, para que se pueda hacer de manera rápida y fiable.

En un principio, para llevar a cabo el trabajo se iba a utilizar Keras, del que ya se tenía algo de conocimiento, pero finalmente se optó por utilizar la librería fastai con PyTorch.

1.3. Motivación

La motivación para llevar a cabo el tema de este trabajo nace, en primer lugar, al realizar la asignatura "Procesado de Imágenes Biomédicas" del grado en ingeniería biomédica en la UPC, ya que me despertó un interés sobre todo lo relacionado con las imágenes biomédicas y su procesado. En segundo lugar, y el principal, fue al cursar la asignatura optativa de "Aprendizaje Bioestadístico" también del mismo grado e impartida por Santiago Alférez y José Rodellar, director y co-director del trabajo respectivamente. En esta última asignatura fue donde me interesé por el tema de las redes neuronales y, por ello, al momento de la elección del tema acudí a ellos.

1.4. Requisitos previos

Uno de los principales requisitos previos para poder llevar a cabo este trabajo es una base de programación en Python y ciertos conocimientos sobre redes neuronales. Para poder obtener estos requisitos se procedió al estudio del libro "Deep Learning with Python" de François Chollet (1), la realización de un curso ofrecido por la Universidad de Stanford, *CS231n: Convolutional Neural Networks for Visual Recognition* (2), y por último la realización del curso de Fast.ai, *Practical Deep Learning for coders* (3). Éste último es muy importante ya que el trabajo se ha llevado a cabo con la librería de Fast.ai.

Otro requisito es tener conocimientos sobre las bases biológicas del estudio, es decir, las células sanguíneas, para poder hacer una mejor interpretación de los resultados obtenidos.

2. Introducción

El aprendizaje automático ha despertado un gran interés en los últimos años, sobre todo una rama en particular debido a su gran potencial, el aprendizaje profundo. El aprendizaje profundo emplea redes neuronales artificiales y ha demostrado un gran desempeño en tareas como la detección y clasificación de imágenes. Aunque es un campo con un largo recorrido, el auge de su popularidad se remonta cerca del 2010, cuando una red neuronal convolucional (AlexNet (4)) logró ganar con una amplia diferencia una competición internacional para la detección y clasificación de imágenes (ILSVRC).

Este aumento de popularidad se debe, en gran parte, gracias a la mejora en la obtención de imágenes y a la capacidad de obtenerlas en grandes cantidades, y también a la mejora del hardware y software, que permiten su implementación.

El creciente uso del *deep learning* y su continuo desarrollo, junto a la mejora en la adquisición de imágenes en el ámbito médico, ha presentado un campo de investigación muy interesante. Esta unión puede resultar muy beneficiosa para la investigación y para el diagnóstico médico, como, por ejemplo, puede ser en la detección de tumores o anormalidades celulares. Por consecuente, esta unión también es muy beneficiosa para la salud del paciente.

El presente trabajo se basa en la unión de estos dos campos, aplicando técnicas del aprendizaje profundo para la clasificación de distintos tipos de células sanguíneas, que pueden llegar a ser indicativos de distintas enfermedades.

En primer lugar, se introduce el marco teórico biológico (capítulo 3), explicando los diferentes tipos de células sanguíneas y alteraciones de los eritrocitos. En el siguiente capítulo (capítulo 4) se explican las bases del aprendizaje automático y el aprendizaje profundo, en especial las redes neuronales convolucionales y el método de transferencia de aprendizaje, junto con las técnicas aplicadas.

Posteriormente se analiza el conjunto de datos (capítulo 5), se procede a la creación de los diferentes modelos y se analizan los resultados obtenidos (capítulo 6).

2.1. Objetivos del trabajo

El objetivo marcado para este trabajo es el de crear un modelo de red neuronal convolucional capaz de clasificar, de manera automática y con un excelente desempeño, imágenes digitales de distintos tipos de células sanguíneas de la serie roja. Los tipos de células en cuestión corresponden a: plaquetas, eritrocitos y diferentes anormalidades en su morfología, que son esferocitos, drepanocitos,

reticulocitos, cuerpos de Pappenheimer, cuerpos Howell-Jolly y punteado basófilo, que pueden llegar a ser indicio de alguna enfermedad.

Para la consecución del objetivo general es necesario realizar los siguientes objetivos específicos:

Estudiar y aprender los fundamentos del aprendizaje profundo, en concreto de las redes neuronales convolucionales (ConvNet o CNN). También el estudio de diferentes arquitecturas de ConvNets populares ya creadas, que son VGG (5), ResNet (6), DenseNet (7) y AlexNet (4).

Estudiar y aprender el método de transferencia de aprendizaje (*transfer learning*), así como dos de sus técnicas: utilizar ConvNet como extractor fijo de características y ajuste fino (*fine tuning*).

Utilizar estas diferentes arquitecturas pre entrenadas como extractor fijo de características del conjunto de datos.

Ajustar la arquitectura pre entrenada partiendo de los pesos calculados al extraer las características para la clasificación de los diferentes tipos de células.

Comparar los resultados obtenidos con las diferentes arquitecturas y las dos técnicas de transferencia de aprendizaje.

Crear una aplicación web sencilla en donde se implemente el mejor modelo obtenido

3. Bases biológicas

3.1. Sistema circulatorio

El sistema circulatorio es el sistema encargado de la circulación de la sangre a través de todos los tejidos del organismo. Atiende así sus diversas necesidades tales como el aporte de oxígeno y nutrientes, el retiro los deshechos (producto del metabolismo) y del dióxido de carbono, el transporte de las hormonas y, en general, mantener un entorno adecuado para la óptica funcionalidad de las células. [(8)Cap. 14, p. 432]

El sistema circulatorio está formado por dos subdivisiones: el sistema cardiovascular y el sistema linfático. El sistema cardiovascular lo forman el corazón, la sangre y los vasos sanguíneos, por otra parte, el sistema linfático lo componen los vasos linfáticos y los tejidos linfoides. [(9) Cap. 13, p. 406]

La circulación de la sangre se consigue gracias al corazón y a un gradiente de presiones. Además, está dividida en circulación pulmonar y circulación sistémica. La primera, como su nombre indica, se encarga de llevar la sangre a los pulmones, y la segunda es la encargada de llevar la sangre a todos los tejidos del organismo, por lo que también recibe el nombre de circulación periférica o mayor. [(10) Cap. 5, p. 112]

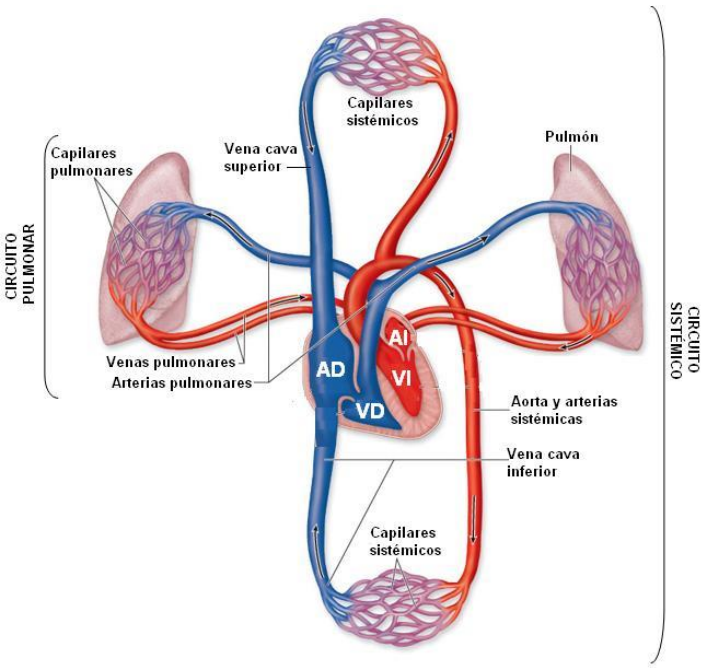

Figura 3.1 Diagrama del sistema circulatorio. ***(Fuente:***(11)***)***

3.2. La sangre

La sangre es un tejido conectivo compuesto por una matriz extracelular líquida llamada plasma, en la cual se disuelven diversas sustancias y se encuentran numerosas células y fragmentos celulares en suspensión (elementos formes).

Su característico color rojo varía según el contenido de oxígeno, cuando está saturada, sangre arterial, es rojo brillante y cuando está insaturada, sangre venosa, es rojo oscuro. Constituye cerca del 20 % del líquido extracelular y representa el 8 % de la masa corporal total del individuo. En un hombre adulto promedio el volumen sanguíneo está entre los 5 y 6 litros y entre los 4 y 5 en el caso de la mujer. [(12) Cap. 19, p 729]

3.2.1. Funciones de la sangre

La sangre tiene tres funciones principales:

- ***Transporte.*** Es la encargada del transporte de oxígeno desde los pulmones hacia las distintas células del cuerpo, así como también del transporte de nutrientes desde el tracto gastrointestinal y de hormonas desde las glándulas endocrinas. Por otra parte, también se encarga de transportar el dióxido de carbono y los distintos deshechos hacia diferentes órganos para su expulsión del cuerpo.
- ***Regulación.*** La sangre ayuda a mantener la homeostasis de todos los líquidos del organismo. Gracias a las proteínas plasmáticas y otros solutos que actúan como sistemas amortiguadores o sistemas tampón, controla los cambios en el pH, que repercutirían en el óptimo funcionamiento celular. También ayuda en el ajuste de temperatura corporal transportando el calor excedente hasta la piel y los pulmones para disiparlo al exterior.
- ***Protección.*** La sangre tiene la posibilidad de controlar sus pérdidas debido a una lesión gracias a su capacidad de coagulación, donde intervienen plaquetas y otras proteínas plasmáticas. También gracias a los glóbulos blancos y diversas proteínas sanguíneas como anticuerpos e interferones presentes en ella se consigue la protección frente a una gran variedad de enfermedades. [(10)Cap. 4, p. 85; (12) Cap. 19, p.729]

3.2.2. Componentes de la sangre

La sangre está constituida por dos componentes:

- ***Plasma sanguíneo.*** Es una matriz extracelular líquida con apariencia amarillenta compuesto principalmente por agua. El resto son solutos, la mayoría de los cuales son proteínas. Constituye el 55 % del volumen total de la sangre.

- ***Elementos formes.*** Los componen tres tipos principales que son los glóbulos rojos o **eritrocitos**, los glóbulos blancos o **leucocitos** y las **plaquetas**. Componen el 45 % de volumen total sanguíneo restante, de los cuales el 99 % corresponden a eritrocitos.
  Dentro de los leucocitos también hay diferentes clases, que se pueden clasificar en:
  A. Granulocitos (contienen gránulos visibles en microscopio)
     1. Neutrófilos
     2. Eosinófilos
     3. Basófilos
  B. Agranulocitos (no se ven gránulos en microscopio)
     1. Linfocitos T y B y células *natural killer*
     2. Monocitos

  [(10)Cap. 4, p. 84; (12) Cap. 19, p. 729-732]

En la siguiente imagen se puede observar cuales son los porcentajes y las cantidades de los diferentes elementos que forman la sangre:

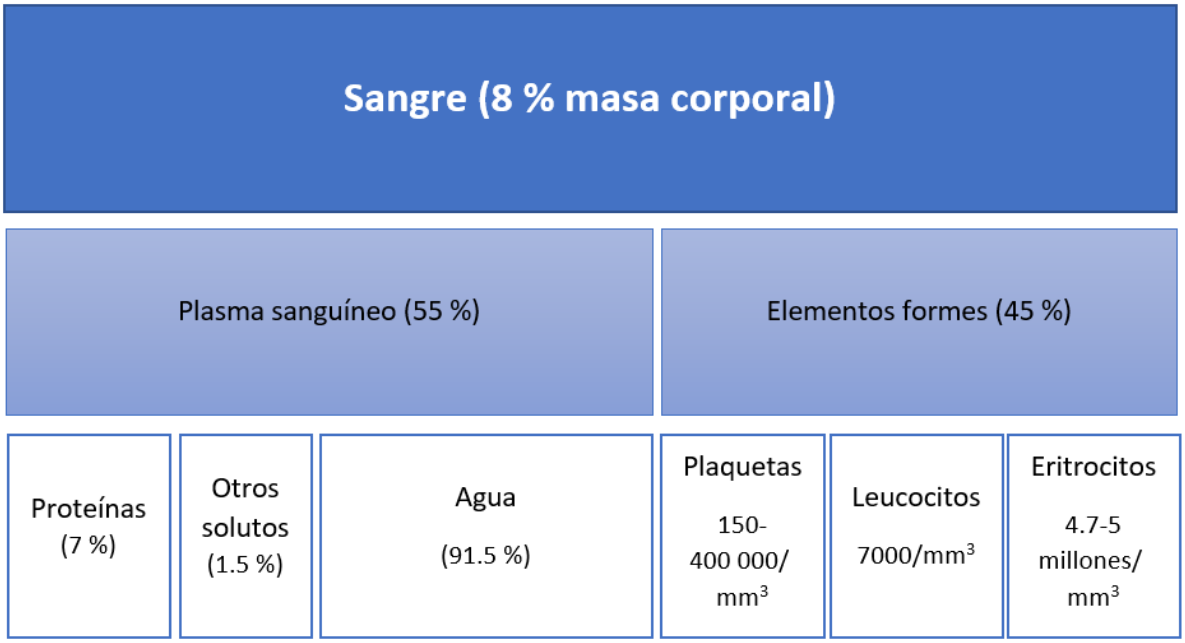

Figura 3.2 Composición de la sangre. ***(Fuente:***(13)***)***

3.3. Células sanguíneas

Como ya se ha comentado en el apartado anterior (3.2.2) los elementos formes o corpusculares de la sangre se pueden clasificar en ***eritrocitos***, ***leucocitos*** y ***plaquetas***, siendo los primeros los más numerosos. Todos ellos surgen gracias a un proceso llamado *hematopoyesis*.

3.3.1. Hematopoyesis

La *hematopoyesis* o hemopoyesis es el proceso por el cual se desarrollan los diferentes elementos formes sanguíneos.

El proceso se produce en diferentes lugares según la etapa embrionaria (tres primeros meses de gestación), la etapa fetal o la etapa posterior al nacimiento. Durante el primer mes de vida se produce exclusivamente en el saco vitelino, a partir de la quinta semana se empieza a producir en el hígado, que será el principal productor desde el tercer al sexto mes. A partir de este momento la producción en la médula ósea comienza a ser importante. Después del nacimiento, todas las células sanguíneas, menos algunos linfocitos, se producen en ella. En la niñez, el proceso se produce tanto en el esqueleto axial, que incluye cráneo, costillas, esternón, vértebras y pelvis, como en los huesos de las extremidades. Una vez ya se es adulto, la producción se limita al esqueleto axial y a los extremos proximales de fémur y húmero, el resto se va reemplazando por tejido adiposo. [(10) Cap. 4, p. 86]

Aun siendo diferentes y con una función propia, todas las células sanguíneas proceden de un precursor común indiferenciado, denominado ***célula madre pluripotencial***. Estas células producen dos tipos de células madre capaces de transformarse en varios tipos celulares, estas son las *células madre mieloides* y las *células madre linfoides*.

Las primeras empiezan su desarrollo en la médula ósea y, tras varios pasos previos, dan origen a eritrocitos, plaquetas, monocitos, neutrófilos, eosinófilos y basófilos. Las células madre linfoides empiezan en la médula ósea, pero lo completan en tejidos linfáticos, dando origen a los linfocitos y células *natural killer* [(12) Cap. 19, p. 734]. En la siguiente imagen se puede observar claramente el proceso de la hematopoyesis.

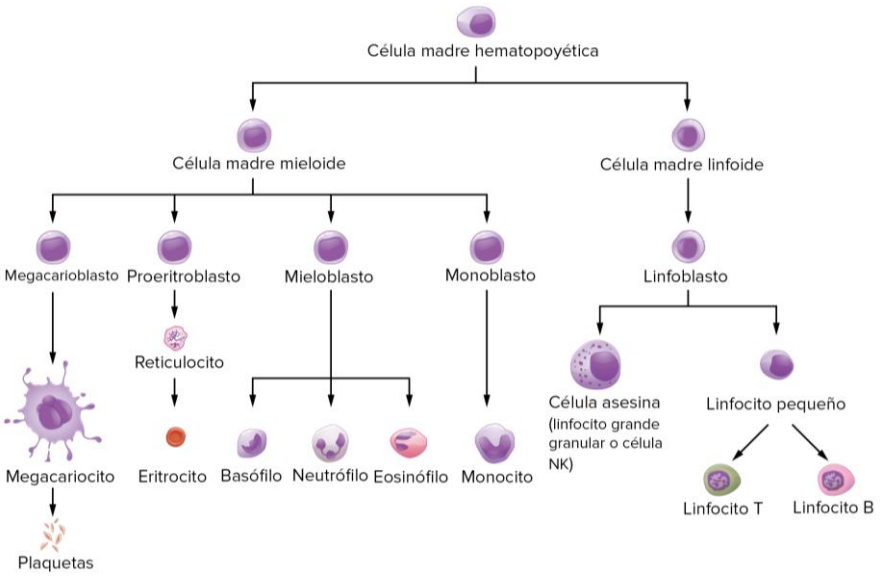

Figura 3.3 Hematopoyesis ***(Fuente:***(14)***)***

3.3.2. Eritrocitos

Los eritrocitos, glóbulos rojos o hematíes son el tipo más numeroso de las células sanguíneas. Son los encargados de transportar la *hemoglobina (Hb),* que es una proteína encargada del transporte de oxígeno por todo el cuerpo, desde los pulmones hasta los tejidos, y la que le da a la sangre su característico color rojo.

Los eritrocitos son discos bicóncavos de unos 7 u 8 µm de diámetro y 2-2.5 µm de grosor. Gracias a esta forma y a su ausencia de núcleo, su relación superficie-volumen es bastante elevada, lo que favorece a su función de transporte. Tiene una gran plasticidad, lo que le permite cambiar mucho su forma y comprimirse a medida que pasa por lo capilares. Además, al carecer de mitocondrias y generar ATP (energía) de forma anaeróbica (sin oxígeno), no utilizan nada de lo que transportan.

Además del transporte, el eritrocito lleva a cabo otras funciones. Una de ellas es el transporte del 23 % de dióxido de carbono, también gracias a la hemoglobina, haciendo el recorrido inverso al oxígeno. La hemoglobina también está involucrada en la regulación del flujo sanguíneo y la tensión arterial.

A parte de la hemoglobina, los eritrocitos también contienen la enzima anhidrasa carbónica (CA), que cataliza la reacción reversible entre el dióxido de carbono (CO_2) y el agua para formar ácido carbónico (H_2CO_3), lo que aumenta la velocidad de la reacción posibilitando que el agua que tiene la sangre pueda transportar grandes cantidades de dióxido de carbono en forma de ion bicarbonato (HCO_3^-) hasta los pulmones. Allí se convierte de nuevo en CO_2 y se expulsa.

$$\boldsymbol{CO_2 + H_2O \overset{CA}{\rightleftharpoons} H_2CO_3 \rightleftharpoons H^+ + HCO_3^-}$$

Ecuación 1

También son responsables de la mayor parte del poder amortiguador de la sangre.

Desde que salen de la médula ósea hacia el sistema circulatorio tienen una media de vida de unos 120 días. Esto se debe a que al no tener núcleo y otros orgánulos no pueden sintetizar ciertos componentes para ir sustituyéndolos por los que se van desgastando con el tiempo. Esto hace que cuando la membrana plasmática se va volviendo más vieja es más frágil y por lo tanto tiene una mayor propensión a estallar. Los macrófagos son los encargados de retirar los eritrocitos rotos y destruirlos. [(8) Cap. 33; (12) Cap.19, p.734-736]

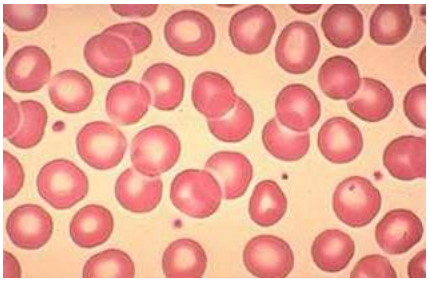

Figura 3.4 Eritrocitos normales ***(Fuente:*** (15) ***)***

Eritropoyesis

La eritropoyesis es como se denomina a la producción de eritrocitos. Empieza en la médula ósea con una célula precursora llamada *proeritroblasto*, que es la primera célula que puede identificarse como perteneciente a la serie eritrocítica, que se forma a partir de la célula madre mieloide.

A partir del *proeritroblasto* se producen los *eristoblastos basófilos*, estas células aún tienen muy poca hemoglobina. En las siguientes generaciones aumenta la concentración de hemoglobina y se condensa el núcleo hasta finalmente ser absorbido o expulsado de la propia célula.

En este momento se obtiene lo que se llama *reticulocito*, que todavía tiene una pequeña cantidad de material basófilo correspondientes a algunas mitocondrias, ribosoma y retículo endoplasmático. Durante este estado la célula se traslada de la médula ósea hacia los capilares sanguíneos. Posteriormente maduran y se transforman, 1 o 2 días después, en *eritrocitos*.

Este proceso está regulado dentro de unos límites para, por un lado, disponer de un número óptimo de eritrocitos para el transporte de oxígeno, y por el otro, para evitar el exceso de eritrocitos que puedan impedir o dificultar el correcto flujo sanguíneo. Uno de los principales encargados de ello es la *eritropoyetina*, que es una hormona producida mayoritariamente en el riñón y actúa sobre la diferenciación de la célula progenitora al proeritroblasto. [(8) Cap. 33, (12) Cap.19, p.737]

Figura 3.5 Eritropoyesis. ***(Fuente:*** (16)***)***

3.3.3. Leucocitos

Los leucocitos o glóbulos blancos son el tipo de célula menos abundante de las tres, aunque tienen una tarea muy importante en la *defensa* del organismo frente a infecciones y tumores. La relación con respecto a los eritrocitos es de inferioridad de 700:1.

Se forman, en parte, en la médula ósea (granulocitos y monocitos, y pocos linfocitos) y, en parte, en el tejido linfático (linfocitos y células plasmáticas). Tras formarse se transportan mediante la sangre allá donde sean necesarios.

Como se comenta en el apartado 3.2.2, la clasificación es la siguiente:

A. Granulocitos (contienen gránulos visibles en microscopio)
   1. Neutrófilos (60 - 70 %)
   2. Eosinófilos (2 - 4 %)
   3. Basófilos (0.5 - 1 %)
B. Agranulocitos (no se ven gránulos en microscopio)
   1. Linfocitos T y B y células *natural killer* (20 - 25 %)
   2. Monocitos (3 - 8 %)

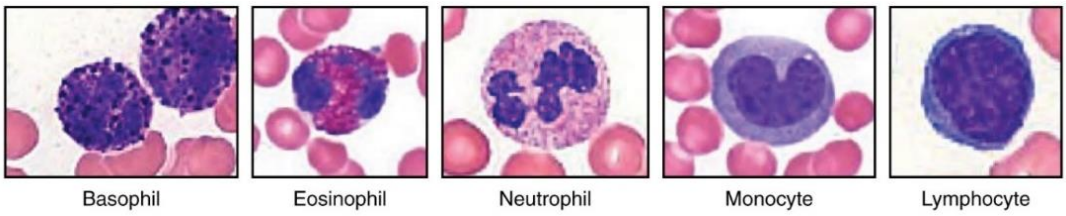

Figura 3.6 Eritropoyesis. ***(Fuente:***(16) ***)***

Los granulocitos y monocitos son los encargados de hacer frente a los microorganismos invasores sobre todo ingiriéndoles, *fagocitosis*, o mediante sustancias antimicrobianas o inflamatorias. Por otra parte, los linfocitos actúan con el sistema inmunitario. Las células B son más efectivas contra las bacterias e inactivando sus toxinas, las células T atacan virus, hongos, células trasplantadas, células cancerosas y algunas bacterias, son las responsables de las reacciones transfusionales, las reacciones alérgicas y el rechazo a órganos trasplantados.

El aumento de número de leucocitos circulantes indica inflamación o infección, por ello, si se hace un recuento de cada tipo de leucocito un médico puede llegar a detectar una infección o inflamación, evaluar afectaciones hemáticas o efectos de quimioterapia, detectar alergias… [(8) Cap. 34; (12) Cap. 19, p. 740]

3.3.4. Plaquetas

Las plaquetas o trombocitos son el tercer grupo de elementos corpusculares de la sangre, son discos de 1 a 4 µm de diámetro. Se forman en la médula a partir de los megacariocitos, células hematopoyéticas muy grandes de la médula ósea, que se van fragmentando formando plaquetas.

Las plaquetas ni poseen núcleo ni pueden reproducirse, aun así, comparten muchas de las características funcionales de las células completas. Es una estructura activa con una vida de 8 a 12 días en la sangre, siendo eliminadas posteriormente por macrófagos.

Su principal función es intervenir en la coagulación de la sangre gracias a sustancias que contienen sus orgánulos y el frenado de pérdida sanguínea en los vasos sanguíneos dañados formando lo que se denomina un tapón plaquetario. [(8) Cap. 37]

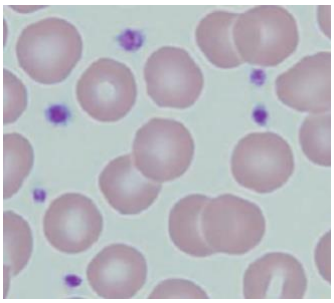

Figura 3.7 Plaquetas junto a eritrocitos **(*Fuente:*** (17)**)**

3.4. Alteraciones morfológicas eritrocitarias

Existen diversas alteraciones en la morfología de la serie eritrocitaria en sangre periférica, ya sean alteraciones en su tamaño, formas anormales, así como también alteraciones de la coloración hemoglobínica o diferentes tipos de inclusiones. Su observación e identificación es de gran importancia a nivel de diagnóstico, ya que puede ser indicio de anemia, entre otros. Las alteraciones más relevantes son las siguientes: (18)

- Alteraciones del tamaño:
  - Anisocitosis
  - Microcitosis
  - Macrocitosis
- Alteraciones en la forma:
  - *Esferocitos*
  - Eliptocitos
  - Ovalocitos
  - Dacriocitos
  - Dianocitos
  - Estomatocitos
  - Esquistocitos
  - Equinocitos
  - Acantocitos
  - *Drepanocitos*
  - Excentrocitos

- Alteraciones de la coloración hemoglobínica:
  - Hipocromía
  - Hipercromía
  - Policromasia
- Inclusiones eritrocitarias:
  - *Reticulocitos*
  - *Punteado Basófilo*
  - *Cuerpos de Pappenheimer*
  - *Cuerpos de Howell-Jolly*
  - Anillos de Cabot

A continuación, se explican las alteraciones objeto del estudio, que son alteraciones de la forma o deformaciones (esferocitos y drepanocitos) e inclusiones (reticulocitos, punteado basófilo, cuerpos de Pappenheimer y cuerpos de Howell-Jolly).

3.4.1. Deformaciones

3.4.1.1. Esferocitos

Los esferocitos son células que tienen una forma esférica o semejante. Son células que han perdido la membrana debido a una anormalidad, adquirida o heredada, del citoesqueleto eritrocitario y la membrana. En un frotis se les puede observar la pérdida de la palidez central típica de un eritrocito. [(19) Cap. 3, p. 79-81]

Hay diversos motivos por los que esto ocurre. En la esferocitosis hereditaria existe una anormalidad del citoesqueleto, atribuible a un defecto hereditario en las proteínas de éste o en alguna proteína relacionada con su bicapa lipídica causando una desestabilización de la membrana y pérdida de parte de ella. En la adquirida, por otro lado, la esferocitosis puede venir como resultado de un daño directo sobre la membrana de la célula como puede ser por calor, o el contacto con veneno o toxinas. La pérdida de membrana puede llevar al recubrimiento de la célula por anticuerpos, siendo después eliminada por macrófagos.

Esta transformación de la forma de la célula puede provocar que sea menos deformable y se atasque con mayor facilidad provocando un daño aún mayor. [(19) Cap. 2, p. 112-115]

Los esferocitos son frecuentes en determinadas anemias hemolíticas congénitas (esferocitosis hereditaria) o adquiridas (anemia hemolítica autoinmune) [(18) p. 2]

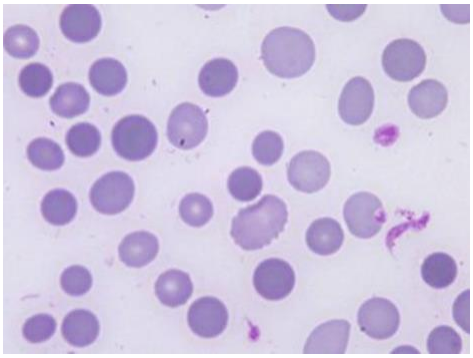

Figura 3.8 Abundantes esferocitos en caso de esferocitosis hereditaria. ***(Fuente:*** (20)***)***

3.4.1.2. Drepanocitos

Los drepanocitos o hematíes falciformes presentan una forma alargada y estrecha (semilunar). Contienen una hemoglobina anormal, llamada hemoglobina S (Hb-S), resultado de una mutación genética. Cuando la Hb-S libera el oxígeno al líquido intersticial, forma estructuras alargadas y rígidas que le dan la forma alargada a los eritrocitos. Cuando los eritrocitos se deforman, se rompen con mayor facilidad y prematuramente. Además, con la forma que adoptan los eritrocitos afectados dificulta su movilidad a través de los vasos sanguíneos y tienden a aglutinarse, pudiendo obstruir los vasos sanguíneos. Esta situación disminuye la cantidad de oxígeno que llega a los tejidos, además de la posibilidad de producir dolores e infecciones. [(21) Cap. 11, p. 45]

Esta afectación es hereditaria. Los homocigotos, es decir, los que han heredado un gen anormal de cada uno de los progenitores, sufren anemias graves; mientras que los que tienen un solo gen defectuoso, heterocigotos, padecen problemas menores. [(12) Cap. 19, p. 751]

Son típicos de la anemia falciforme o drepanocítica. [(18) p.13]

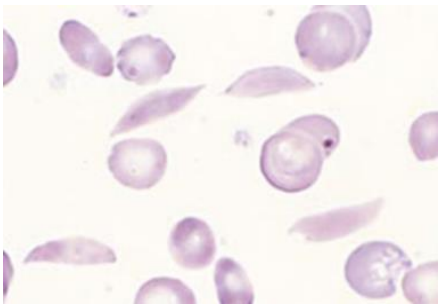

*Figura 3.9 Drepanocitos o eritrocitos falciformes. (****Fuente:*** (22)***)***

3.4.2. Inclusiones

3.4.2.1. Reticulocitos

Los reticulocitos son células en transición entre los eritroblastos ortocromáticos y los eritrocitos maduros. Su fase de maduración es de aproximadamente cuatro días, tres de los cuales generalmente en la médula ósea, durando así sólo unas 24 horas en circulación. Se definen por la presencia de material citoplasmático, compuesto por ácido ribonucleico ribosomal (RNA). Además del RNA ribosomal puede contener varios orgánulos como mitocondrias, restos del retículo endoplasmático, centriolos y moléculas de ferritina, entre otros.

Al contrario que los eritrocitos maduros, los reticulocitos pueden tener una forma más irregular, a veces lobulada. [(19) Cap. 2, p. 96]

Constituyen entre el 0,5 y el 2 % de hematíes totales, en condiciones normales. Este porcentaje puede incrementarse en caso de una hemorragia intensa, en anemias hemolíticas, esplenectomía (extirpación del bazo) o en el tratamiento de las anemias carenciales.

Su cuantificación permite determinar el carácter regenerativo, mayor cifra de reticulocitos, o arregenerativo, menor cifra de reticulocitos, de una anemia. [(18) p. 17-18]

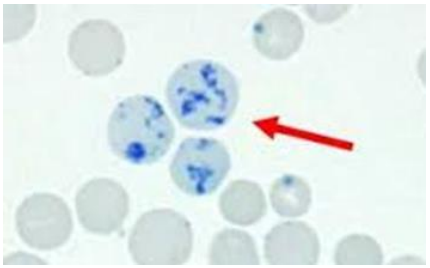

Figura 3.10 Reticulocitos. ***(Fuente:**(23)**)***

3.4.2.2. Punteado Basófilo

El punteado basófilo indica la presencia de numerosos gránulos de color azul oscuro correspondientes a ribosomas precipitados, compuestos principalmente por RNA. Es atribuible a una estructura o composición anómalas de los ribosomas o un defecto en el mecanismo enzimático requerido para su catabolismo, favoreciendo así su precipitación. Por lo general este defecto está asociado con anormalidades en la síntesis de la hemoglobina. [(19) Cap. 2, p. 128]

Son bastante comunes y no tienen una importancia significativa en el diagnóstico. Se pueden observar en la talasemia, anemia megaloblástica, intoxicación por plomo, ingesta crónica de alcohol, anemia

sideroblástica y deficiencia de pirimidina 5-nucleotidasa, una enzima que participa en la degradación de las moléculas de ARN. [(21) Cap. 55, p. 290]

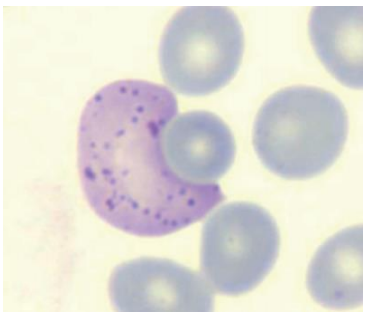

Figura 3.11 Eritrocito joven con punteado basófilo. ***(Fuente:*** (18) ***p. 18)***

3.4.2.3. Cuerpos de Pappenheimer

Los cuerpos de Pappenheimer son inclusiones basófilas que pueden estar presentes en pequeños grupos en la periferia de los eritrocitos y se puede demostrar la presencia de hierro. Están compuestos por agregados de ferritina, o de mitocondrias o fagosomas que contiene ferritina agregada. Los eritrocitos con este tipo de inclusiones se denominan siderocitos. [(24) Cap. 3, p. 96]

Se observan con la anemia sideroblástica, talasemia, síndromes mielodisplásicos, anemia diseritropoyética y otras anemias graves. [(21) Cap. 55, p. 291]

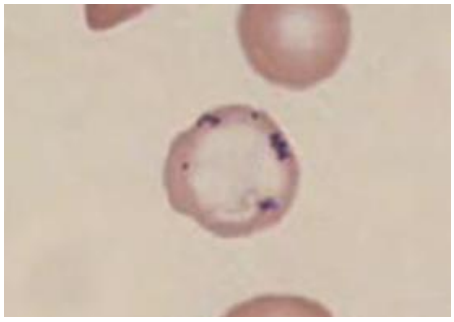

Figura 3.12 Eritrocito con cuerpos de Pappenheimer. ***(Fuente:*** (24) ***Cap. 3, p. 96)***

3.4.2.4. Cuerpos de Howell-Jolly

Los cuerpos de Howell-Jolly son inclusiones eritrocitarias citoplasmáticas, de tamaño medio, que son fragmentos de material nuclear y están compuestas por ácido desoxirribonucleico (ADN).

Puede surgir a partir de la rotura del núcleo, por expulsión nuclear incompleta o puede ser un cromosoma que se ha separado del huso mitótico durante una mitosis anormal. [(24) Cap. 3, p. 93]

Se relaciona con la anemia megaloblástica, eritropoyesis acelerada como en la anemia hemolítica, drepanocitosis, posesplenectomía o hipoesplenismo. [(21) Cap. 55, p. 291]

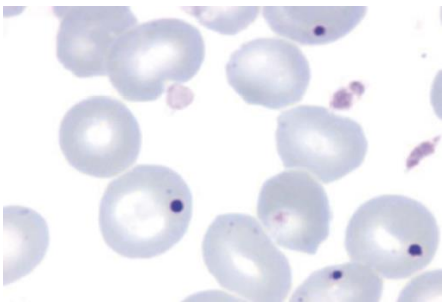

Figura 3.13 Eritrocitos con cuerpos de Howell-Jolly. ***(Fuente:*** (18) ***p. 20)***

4. Aprendizaje automático y aprendizaje profundo

Para definir el aprendizaje automático se comparará con la programación clásica. En la programación clásica se introducen reglas (programa) y datos para procesarlos siguiendo estas reglas, obteniendo así resultados. En cambio, en el aprendizaje automático o *machine learning* se introducen los datos con las respuestas esperadas de estos, obteniendo como resultado las reglas. Estas reglas que se obtienen se pueden aplicar a nuevos datos para producir las respuestas deseadas. Por este motivo se dice que un sistema de machine learning se "entrena" (1).

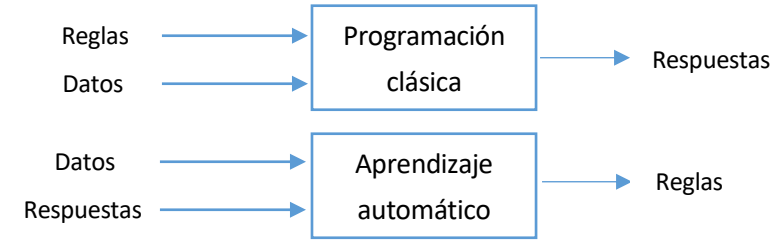

Figura 4.1 Machine learning. (Fuente: (1) Cap. 1, p. 5)

Por tanto, se puede decir que machine learning es un tipo de inteligencia artificial donde el ordenador aprende a hacer algo sin haberlo programado.

Poniendo un ejemplo, se puede programar para identificar a un animal como gato escribiendo un código que haga que el programa responda "gato" cuando vea una imagen de un gato en particular. Esto sólo funciona correctamente si el único gato que el programa va a ver es el gato de la imagen. En cambio, fallaría si el programa va a ver diferentes animales, entre ellos diferentes gatos, y tenga que elegir cuales de ellos son gatos.

En el aprendizaje automático se suele trabajar con un conjunto de datos grande y complejo.

Para llevar a cabo machine learning se necesitan tres cosas:

- ***Datos** de entrada*, que pueden ser imágenes, textos o sonidos
- *Ejemplos del **resultado esperado***, en el caso de imágenes una etiqueta como "eritrocito" o "drepanocito"
- *Una **métrica** para observar si el algoritmo lo está haciendo correctamente*, esto es para ir viendo la diferencia entre el resultado obtenido y el esperado. Esta medida se usa como una señal de retroalimentación para ajustar el funcionamiento del algoritmo. Este ajuste es al que se le llama aprendizaje.

El *aprendizaje profundo*, más conocido como *deep learning,* es un campo del aprendizaje automático que se basa en el aprendizaje a partir de datos a través de una sucesión de capas cada vez más complejas. A mayor cantidad de capas, más "profundo" es el modelo. Esta sucesión de capas forma lo que se denomina *redes neuronales*.

Las redes empiezan aprendiendo algo simple en las primeras capas, pasando la información de cada una a la siguiente. A medida que pasan las capas esta información se va colectando y combinando en algo más complejo hasta llegar a la última capa que ya es capaz de reconocer el objetivo.

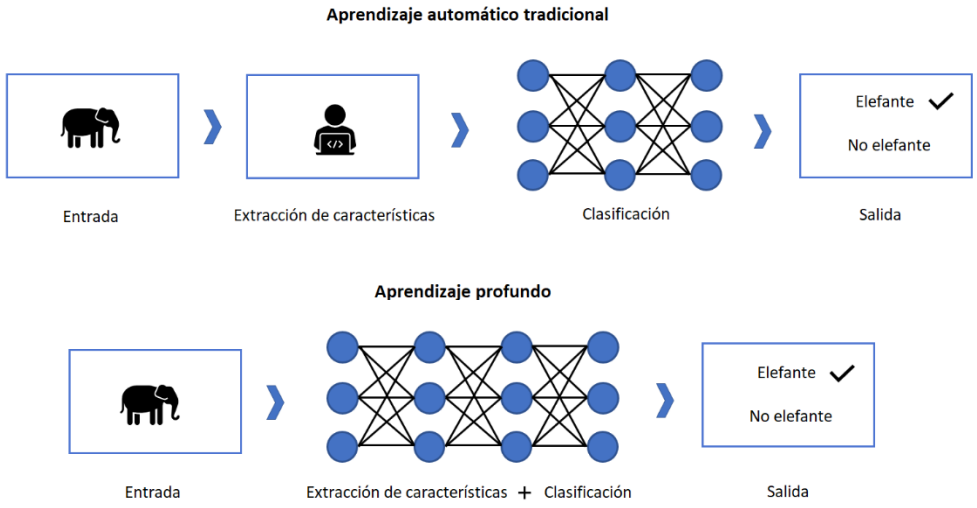

Figura 4.2 Enfoque machine learning vs deep learning. ***(Fuente:*** (25)***)***

4.1. Redes neuronales

Como se ha comentado, las redes neuronales son la base del *deep learning*. Estas están constituidas por una sucesión de capas conectadas, donde en cada una de ellas se pueden encontrar varias unidades básicas, llamadas *neuronas artificiales*. La conexión entre estas capas se produce a través de estas unidades básicas y depende de la arquitectura del modelo.

Estas unidades básicas están inspiradas en las neuronas del cerebro, de ahí el nombre. En el cerebro humano se encuentran millones de neuronas, que se conectan entre ellas mediante *sinapsis.* Cada neurona recibe señales de entrada de sus dendritas y produce señales de salida a través de su axón. El axón a su vez está conectado mediante sinapsis a dendritas de otras neuronas. Haciendo una analogía y apoyándose en la siguiente figura, en el modelo computacional de la neurona, las señales se transmiten a través de los axones (x_0), mediante sinapsis, a las dendritas de la siguiente neurona siendo ponderadas en base a la fuerza de esa conexión (ω_0). Esta fuerza de conexión, los llamados *pesos o parámetros* (ω), son aprendibles y controlan la fuerza de influencia o inhibición entre estas dos neuronas. Las dendritas llevan esta señal hacia el cuerpo celular donde se suman ponderadamente

todas. Si la suma supera cierto umbral, la neurona propaga la información. Este umbral que permite o no la propagación viene definida por lo que se denomina una *función de activación* (26).

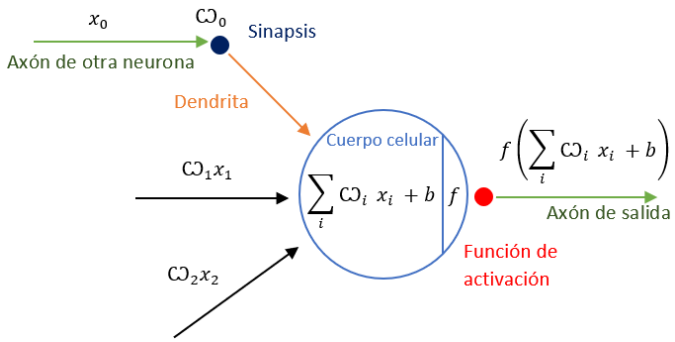

Figura 4.3 Representación del modelo matemático de una neurona. ***(Fuente:*** (26)***)***

Por lo tanto, cada capa contiene unos pesos o parámetros, que son los que contienen el conocimiento adquirido por la red. El objetivo del aprendizaje de la red es el de hallar el valor de estos parámetros que consigan que, de los valores de entrada, se obtenga la salida esperada. Por eso es importante obtener un buen y amplio conjunto de datos.

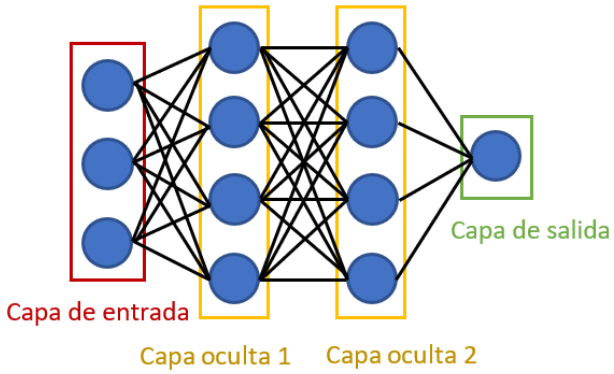

Figura 4.4 Ejemplo de red neuronal de 3 capas. ***(Fuente:***(26)***)***

Una neurona artificial en sí, es considerada como la red más sencilla, aunque, como es obvio, sus capacidades son muy limitadas. A mayor complejidad de la red neuronal, mayor capacidad de aprendizaje y por tanto puede afrontar problemas más complejos.

Al hablar de la cantidad de capas de una red neuronal solo se tienen en cuenta aquellas que tienen entradas. Como se puede observar en la anterior figura, se puede hablar de tres tipos de capa:

- *Capas de entrada* (input layers). Es la que recibe los datos de entrada.
- *Capas ocultas* (hidden layers). Son las capas internas que contienen los parámetros.
- *Capas de salida* (output layers). Es la capa que proporciona los resultados de la red, es el clasificador (26).

4.1.1. Funciones de activación

Las funciones de activación cogen un solo número y aplican una operación matemática. Las más comunes son:

- ***Sigmoide.*** Esta función de activación coge el valor de la salida y la comprime a un rango [0,1]. Los valores negativos grandes se convierten en 0 y los valores positivos grandes se convierten en 1. Tiene dos grandes inconvenientes, el primero es que cuando la activación de la neurona satura, ya sea a 0 o a 1, el gradiente es casi cero y hace empeorar mucho el aprendizaje de las neuronas. El segundo es que al no estar centrada en el cero puede producir una indeseable dinámica de zig-zag en las actualizaciones de los pesos.

$$\boldsymbol{\sigma(x) = \frac{1}{(1 + e^{-x})}}$$

Ecuación 2 Sigmoide

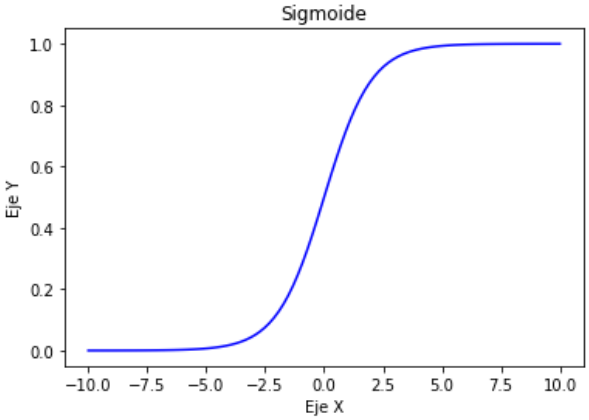

Figura 4.5 Función de activación sigmoide.

- ***Tanh.*** Esta función coge el valor de salida y lo comprime a un rango [-1,1]. Como en la sigmoide, su activación puede saturar, pero a diferencia de la sigmoide, está centrada en cero.

$$\boldsymbol{tanh(x) = 2\sigma(2x) - 1}$$

Ecuación 3 Tanh(x)

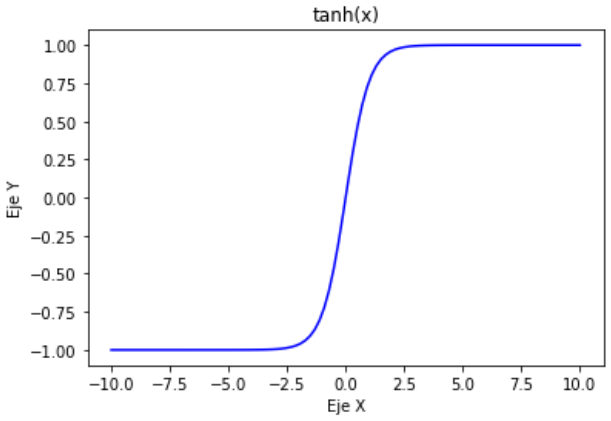

Figura 4.6 Función de activación tanh.

- ***ReLU.*** Esta función es de las más populares en la actualidad, ya que es utilizada en casi todas las redes neuronales convolucionales. Básicamente pone todos los valores negativos a cero.

$$\boldsymbol{f(x) = max(0, x)}$$

Ecuación 4 ReLU

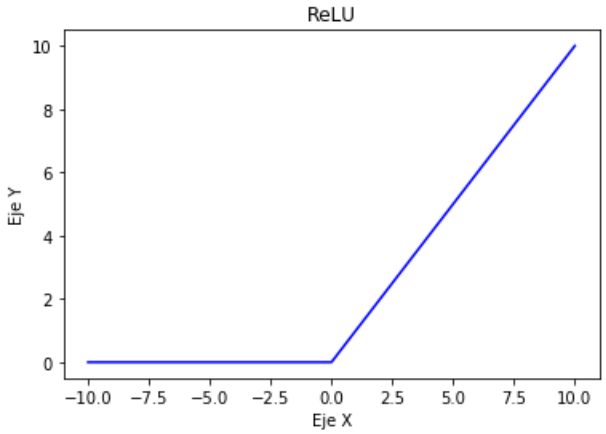

Figura 4.7. Función de activación ReLU.

4.1.2. Aprendizaje de las redes neuronales: Back-propagation

Como ya se ha comentado, una red neuronal está formada por diferentes capas, la de entrada, las capas ocultas y la de salida. También que las conexiones entre nodos de capas adyacentes contienen parámetros asociados con ellos. Y que el objetivo del aprendizaje es asignar los valores correctos a estos parámetros para que, dada una entrada, se obtenga la salida deseada.

La retro propagación de errores o *Back-propagation* es una de las varias maneras que tiene una red neuronal artificial para aprender. Es un entrenamiento supervisado, lo que significa que aprende de un

conjunto de datos ya etiquetado, es decir, que para cada entrada ya se sabe la salida esperada. Se basa en el aprendizaje a partir de los errores, el "supervisor" va corrigiendo la red allá donde se produzcan los errores.

Al crear la red neuronal, todos los parámetros son asignados aleatoriamente. Entonces, al entrenar e introducir los datos de entrenamiento, la red se activa y se produce una salida. Esta salida se compara con la deseada que se conoce. El error, es decir, la diferencia entre la salida deseada y la obtenida, se propaga atrás hacia las capas anteriores. Es entonces cuando los parámetros se ajustan de acorde al gradiente de este error, buscando siempre el mínimo. Cuando todo el conjunto de datos ha pasado por este proceso se completa lo que se denomina un ***epoch***. Para entrenar una red neuronal normalmente hacen falta varios epochs. Como normalmente los conjuntos de datos no son muy pequeños, no se puede introducir todos a la vez a la red, por eso se divide el conjunto de datos en varios lotes (***batch***). Al terminar este proceso se dice que la red neuronal ha sido entrenada (27).

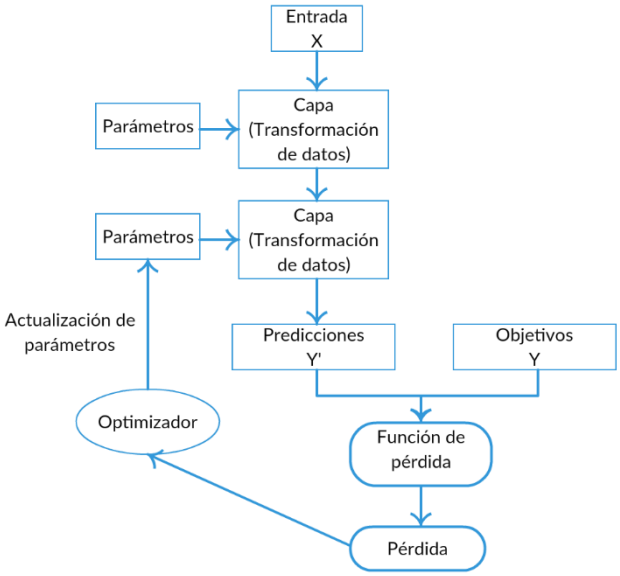

Figura 4.8 Esquema del funcionamiento de una red neuronal. ***(Fuente:*** (1) ***Cap.1, p.11)***

En el aprendizaje de las redes neuronales entran en juego los siguientes conceptos:

- ***Función de perdida***. Determina como la red medirá el desempeño en el conjunto de entrenamiento y como será capaz de orientarse en la dirección correcta.
- ***Optimizador.*** Mecanismo a través del cual la red se actualizará basándose en los datos que observa y su función de pérdida.
- ***Métricas*** para monitorear el entrenamiento y las pruebas. Puede ser la tasa de error, la precisión, etc. [(1) Cap. 2, p. 29]
- ***Learning rate (tasa de aprendizaje).*** Es el valor que establece la velocidad de aprendizaje, se multiplica por el gradiente. Si es muy alto, no convergirá al mínimo error. Además, la pérdida

aumenta y el modelo dejará de aprender. En cambio, si es muy bajo, el modelo aprenderá muy lentamente, cosa que tampoco interesa. Su correcta elección es un factor muy importante.

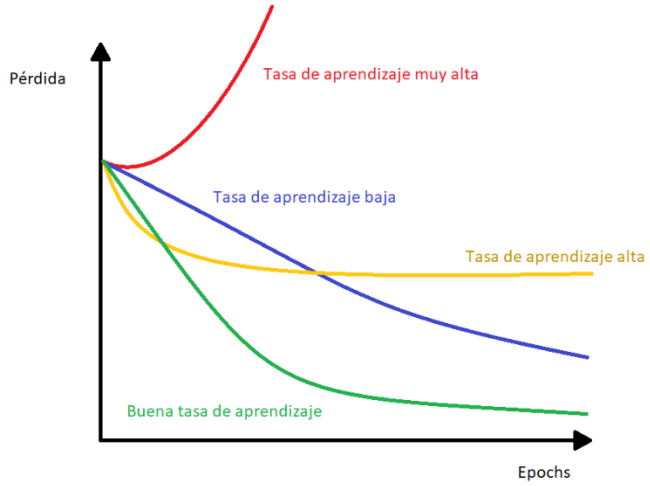

Figura 4.9 Learning Rate (Fuente:(28)*)*

4.1.3. Política de un ciclo (*one cycle policy*)

Al momento de entrenar la red, en lugar de utilizar un learning rate fijo, es recomendable usar la política de un ciclo o *one cycle policy* (29). Esta política recomienda hacer un ciclo de learning rate de dos etapas de igual longitud, es decir, en la primera etapa se va aumentando el learning rate desde un valor bajo a uno alto, y en la segunda etapa al revés. Esta subida y bajada forma lo que se llama un ciclo.

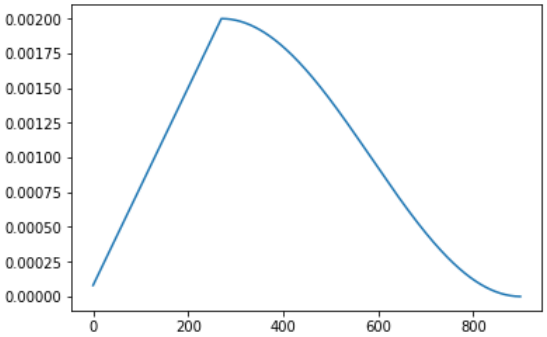

Figura 4.10 One cycle (evolución learning rate). Fuente: (3)

Para escoger el valor máximo de *learning rate* se grafican las pérdidas obtenidas con respecto al progresivo aumento de *learning rate*. Al hacer esto se ve que a medida que aumenta el learning rate las pérdidas van disminuyendo, hasta que llega cierto punto en que estas vuelven a subir. Entonces, el máximo debe ser el valor de *learning rate* justo antes de que las pérdidas vuelvan a crecer. El mínimo debería ser unas diez veces menor que el máximo.

El motivo detrás de esto es que, durante la mitad del aprendizaje, es decir, cuando el *learning rate* es mayor, éste actúa como un método de regularización y evita el sobreajuste de la red. La parte de subida ayuda a la red a evitar los mínimos locales. Por otra parte, cuando el learning rate vuelve a bajar permite llegar correctamente al mínimo de la función, donde es más suave. En esta bajada es cuando se observan las mejoras en las pérdidas y la exactitud (29)–(32).

4.2. Redes neuronales convolucionales

Las *redes neuronales convolucionales*, *ConvNets* o *CNN* son una categoría de redes neuronales muy efectivas en el reconocimiento y clasificación de imágenes. Una CNN es un algoritmo que puede coger una imagen de entrada, asignarles importancia a varios aspectos de la imagen y posteriormente es capaz de diferenciarla de otra distinta.

En la figura siguiente se puede observar un ejemplo de una arquitectura de ConvNet que clasifica una imagen de entrada en cuatro categorías. Mirando las predicciones de salida, la clasificación es correcta.

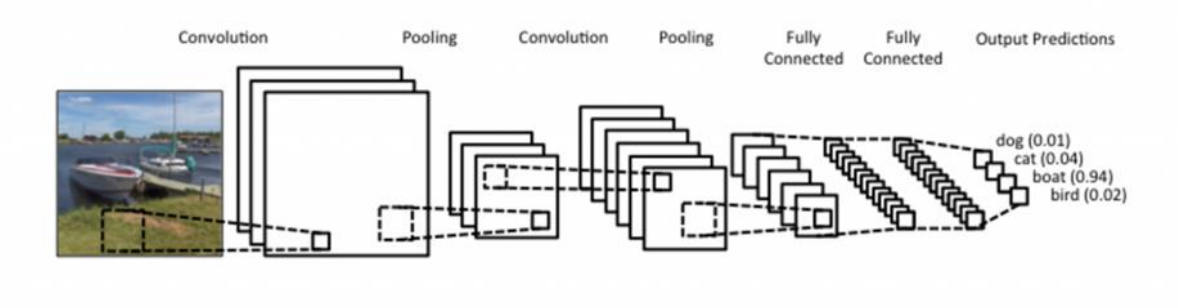

Figura 4.11 Esquema de una ConvNet. ***(Fuente:***(33)***)***

Una red neuronal convolucional se puede dividir en las siguientes operaciones:

- Convolución
- ReLU
- Agrupación (*pooling*)
- Clasificación

Estas operaciones son los componentes básicos de toda red neuronal convolucional (34).

Primero de todo, definir que una imagen puede ser representada por una matriz de valores. Según los canales que tenga (si es en color tiene tres: rojo, verde y azul; y si es en escala de grises solo tiene uno) tendrá la misma cantidad de matrices, una para cada canal.

4.2.1. Convolución

Las ConvNets deben su nombre al operador matemático de la convolución. El principal objetivo de la convolución en este caso es extraer características de la imagen de entrada.

Estas características son extraídas con la aplicación de diversos filtros sobre la imagen de entrada. Estos filtros suelen ser pequeños en cuanto ancho por alto, pero en profundidad son iguales a la imagen de entrada.

Por ejemplo, se observa la imagen siguiente. La matriz verde corresponde a la imagen de entrada y la matriz amarilla corresponde a un filtro. En la fila de abajo se observa como el filtro se va deslizando sobre la imagen haciendo la convolución, obteniendo así otra matriz. A esta matriz se le denomina *mapa de activación* o mapa de características. Según el filtro se obtiene un mapa de características diferente. Esto permite que cada filtro detecte una característica diferente, como pueden ser curvas, esquina, líneas, etc.

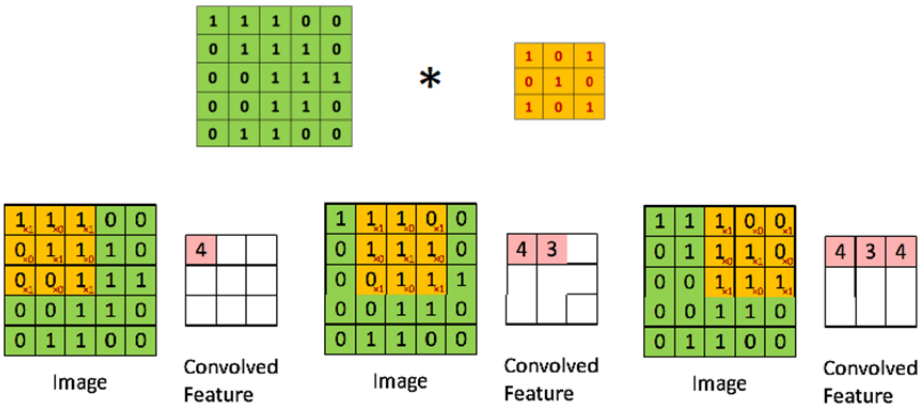

Figura 4.12 Convolución ***(Fuente:*** (34)***)***

En la práctica, una CNN aprende los valores de estos filtros por su cuenta durante el proceso de entrenamiento. Lo que se deberá que especificar son otros parámetros como el número de filtros, el tamaño de estos, la arquitectura de la red, etc.

A mayor cantidad de filtros, más características de la imagen se pueden extraer y mejor se reconocerá los patrones en imágenes no vistas.

El tamaño de los mapas de características depende de tres parámetros:

- *Profundidad.* La profundidad corresponde al número de filtros que se usan para la convolución.
- *Zancada (Stride).* La zancada es el número de píxeles por los que se desliza la matriz de filtro sobre la matriz de entrada. Cuando es uno, el filtro se mueve cada un pixel, si es dos, cada dos.

- *Relleno de ceros (Zero-padding*). Para poder aplicar el filtro a los elementos de los bordes, a veces es conveniente rellenar la matriz de entrada con ceros alrededor suya (34)**.**

4.2.2. ReLU

ReLU es una operación aplicada a cada elemento de los mapas de características y, como se comenta en el apartado 4.1.1, reemplaza todos los valores de píxeles negativos en el mapa de activación por cero. El propósito es introducir la no linealidad en la ConvNet, ya que la mayoría de datos que se quiere que aprenda la red son no lineares.

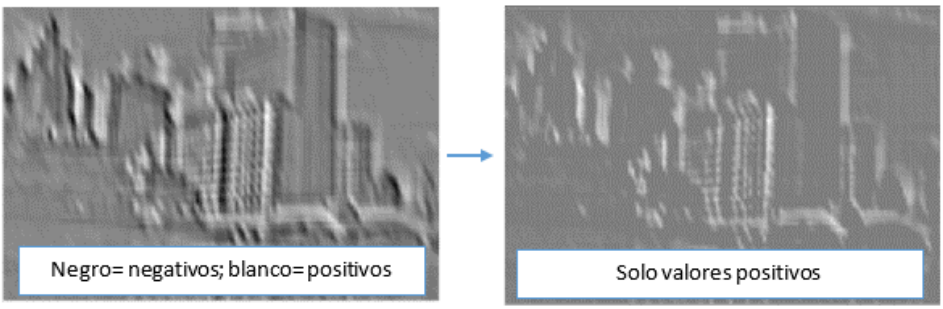

Figura 4.13 Ejemplo de aplicación de ReLU. ***(Fuente:*** (35)***)***

4.2.3. Agrupación (*Pooling*)

La agrupación espacial o submuestreo es una operación que reduce la dimensión de cada mapa de características, pero conservando la información más relevante. Esta agrupación puede ser de diferentes tipos como *Max (máximo), Average (media), Sum (suma)*…

En el caso de *Max Pooling*, que es el usado por lo general en CNNs, se define una ventana y se coge el elemento más grande del mapa de características rectificado dentro de esa ventana. En la siguiente figura se puede entender de manera sencilla. En la imagen de la derecha se lleva a cabo un *max pooling* con filtros de 2x2 y con una zancada de 2, quedando así la matriz de 2x2 (34).

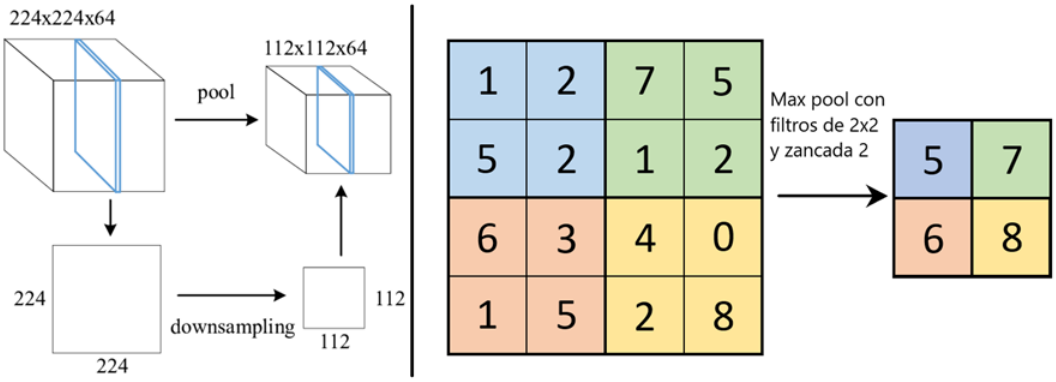

Figura 4.14 Max Pooling. ***(Fuente:*** (36)(37) ***)***

Por tanto, con el uso de *pooling* se consigue:

- Reducir el tamaño espacial de la representación de entrada haciéndola más manejable, no la profundidad.
- Reducir el número de parámetros de la red y controlar el sobreajuste.
- Hacer a la red más robusta frente a pequeñas distorsiones de la imagen de entrada.
- Conseguir una representación casi invariante de la imagen (38).

4.2.4. Regularización

Puede aparecer también algún método de regularización para evitar el sobreajuste. El más utilizado es el *Dropout* ya que es bastante efectivo. El término "dropout" se refiere a ignorar unidades (ocultas y visibles) en una red neuronal. Al ignorar una unidad, se refiere a su eliminación temporal de la red, junto con todas sus conexiones entrantes y salientes, de manera que queda una red más reducida.

4.2.5. Clasificación. Capa totalmente conectada

Una vez pasado por todas estas operaciones se obtienen varios mapas de activación, dependiendo la cantidad de capas, y se llega a la última capa. En esta última etapa se trata de una capa completamente conectada, es decir, cada neurona de la anterior capa está conectada a todas las neuronas de la siguiente capa.

El propósito de la capa completamente conectada es usar estas características obtenidas por las capas anteriores para clasificar la imagen de entrada en alguna de las clases del conjunto de entrenamiento.

En la siguiente imagen se puede observar esta última capa con claridad.

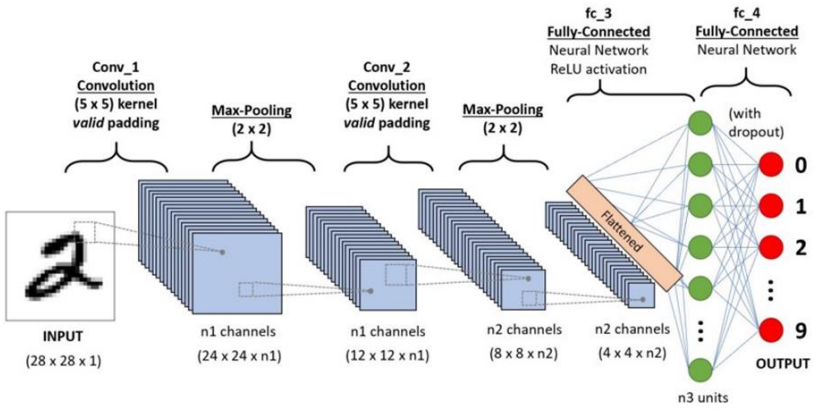

Figura 4.15 Ejemplo de una ConvNet para clasificar números escritos a mano. ***(Fuente:*** (39)***)***

4.2.6. Visualizando las ConvNets

Para hacerse una idea de cómo aprende una red neuronal convolucional existen varias técnicas. En primer lugar, se podría observar los diferentes mapas de características, otra manera sería observar qué características extrae cada filtro de la imagen, otra sería observar los mapas de calor para cada capa de la red o, por último, buscar la salida de cada capa convolucional en la imagen de entrada.

Esta última técnica fue propuesta por Zeiler y Fergus (40), lo que se hace es el mismo proceso que una ConvNet pero al revés.

Al llevar a cabo esta técnica se observa el aumento de complejidad de las características extraídas a medida que se avanza en la red. Es decir, en las primeras capas se obtienen líneas, bordes, esquinas o gradientes, como se puede observar en la imagen, y, a medida que se avanza se van combinando formando características más complejas como diferentes texturas. En las últimas capas ya sería capaz de reconocer partes de la imagen, como pueden ser ojos, nariz, boca, ruedas… Hasta que al final ya es capaz de reconocer el objeto entero como una cara, un coche, una bicicleta… (41).

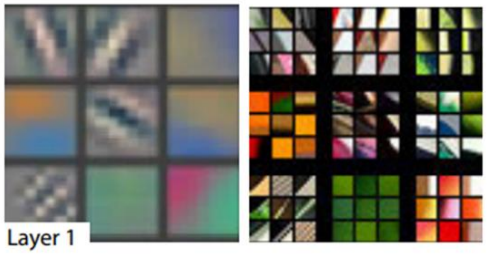

Figura 4.16. Visualización primera capa CNN. ***(Fuente:*** (40)***)***

4.3. Métodos propuestos: *transfer learning*

Como ya se ha comentado, las redes neuronales convolucionales o ConvNets han demostrado ser muy efectivas en reconocimiento de imágenes. Sin embargo, muy pocas veces se entrena una red neuronal convolucional desde cero (con inicialización aleatoria). Esto pasa porque es extraño poseer un conjunto de datos de un tamaño suficientemente grande. En lugar de eso se suele utilizar una técnica llamada *transfer Learning* (transferencia de aprendizaje). En este trabajo se llevarán a cabo dos técnicas importantes de *transfer Learning*.

El *Transfer Learning* es una técnica muy utilizada para lidiar con el problema de la falta de un buen conjunto de datos. Esta técnica se basa en utilizar una CNN pre entrenada con un conjunto de datos muy grande, como por ejemplo ImageNet (42) (con 1.2 millones de imágenes y mil categorías), como punto de partida o como un extractor de características para la tarea de interés (43). Esto es posible

gracias a que, si el conjunto de datos con los que se ha entrenado es suficientemente grande, las características aprendidas por esta red pueden funcionar como un modelo genérico, siendo útil también para clases totalmente nuevas [(1)Cap. 5, p. 143]. Las principales técnicas de *transfer learning* son las siguientes dos: ***ConvNet como extractor fijo de características y Fine Tuning.***

4.3.1. CNN como extractor fijo de características

En este caso se coge una ConvNet pre entrenada en ImageNet, se "congelan" los parámetros de toda la red excepto la última capa totalmente conectada, el clasificador, que es extraída. Una vez extraída se sustituye por una nueva con parámetros aleatorios para las nuevas predicciones (43). Este proceso se puede observar gráficamente en la figura siguiente.

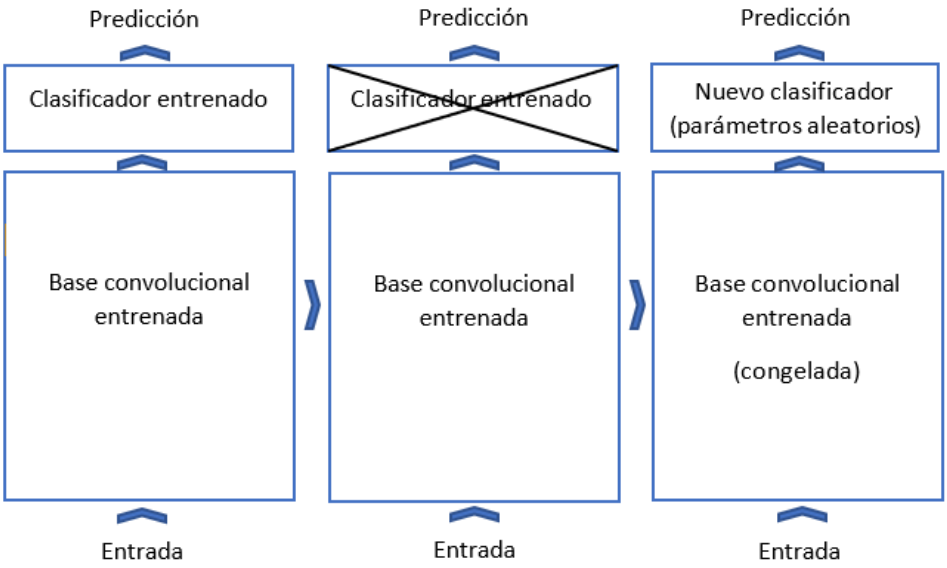

Figura 4.17. Cambio de clasificador en Transfer Learning. ***(Fuente:*** (1) ***Cap. 5, p. 144)***

En este punto ya se puede coger el conjunto de datos con el que se trabaje y entrenar el modelo, pero en esta técnica solo se entrena esta última capa añadida.

La extracción de esta última capa es debido principalmente a dos cosas, la primera es que esta última capa al haber sido entrenada con ImageNet, es una matriz de parámetros que tiene 1000 columnas y difícilmente se trabajaran con tantas categorías. Y la segunda, aunque se tuviesen mil categorías, estas no serían las mismas que en ImageNet, por tanto, tampoco sería bueno. Por tanto, esta última capa es reemplazada por una nueva con parámetros aleatorios y es la única que se entrena.

4.3.2. Fine-Tuning

La técnica de fine-tuning es otra de las técnicas para reusar un modelo ya existente y complementaria a la extracción de características. Esta segunda estrategia no es solo reemplazar y volver a entrenar el clasificador de la ConvNet con el nuevo conjunto de datos, sino que también se descongelan los parámetros de la red pre entrenada y se modifican al continuar la backpropagation. Es posible aplicar fine-tuning a todas las capas de la ConvNet o también dejar alguna de las primeras capas fijas y solo descongelar una porción de alto nivel. Esto se debe a la observación de que los primeros parámetros de la ConvNet contienen características más genéricas que pueden ser útiles para varias tareas, en cambio, a medida que avanzan las capas se vuelven más específicas a los detalles de las clases del conjunto de datos original (43).

4.3.3. Modelos pre entrenados

Los modelos pre entrenados, por tanto, son modelos que han sido entrenados previamente con un conjunto de datos de gran tamaño, que contiene los parámetros que representan las características del que haya sido el conjunto de datos con el que se entrenó.

En la actualidad existen muchos modelos pre entrenados, pero algunos de los principales y más populares son los siguientes:

4.3.3.1. ResNet

Las Redes Residuales o ResNet (Residual Networks) son una de las mejores opciones y la que más se utiliza actualmente (44).

Mientras más profunda es una ConvNet mejores resultados se obtienen, ya que los modelos tienen una mayor capacidad para aprender. Sin embargo, se ha observado que, al llegar a cierta profundidad, el rendimiento se degrada. Este problema es el denominado *vanishing gradient* (desvanecimiento del gradiente). Surge al hacer backpropagation (apartado 224.1.2) al ir retrocediendo en la red y actualizando los parámetros de acorde con el gradiente del error. Los gradientes tienden a ir haciéndose cada vez más pequeños a medida que se retrocede en la red, lo que hará que el ajuste de los parámetros en las capas anteriores sea muy pequeño y, por lo tanto, lento. En el peor de los casos, esto puede detener completamente el entrenamiento de la red neuronal.

Las ResNets están constituidas por unos bloques llamadas bloques residuales, que introducen lo que se llama una *skip connection* o atajo, que ofrece un camino alternativo (saltándose varios pasos) a la información que se transmite para poder llegar más profundo sin perderla.

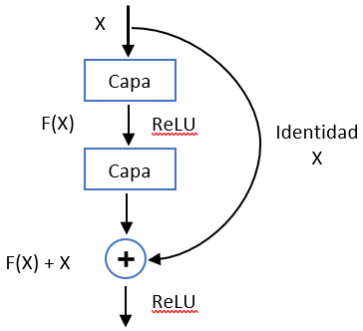

Figura 4.18 Residual block. ***(Fuente:*** (45)***)***

Existen diferentes tipos de ResNet según su tamaño, está la ResNet 18, la ResNet 34, la ResNet 50, la ResNet 101 o la ResNet 150.

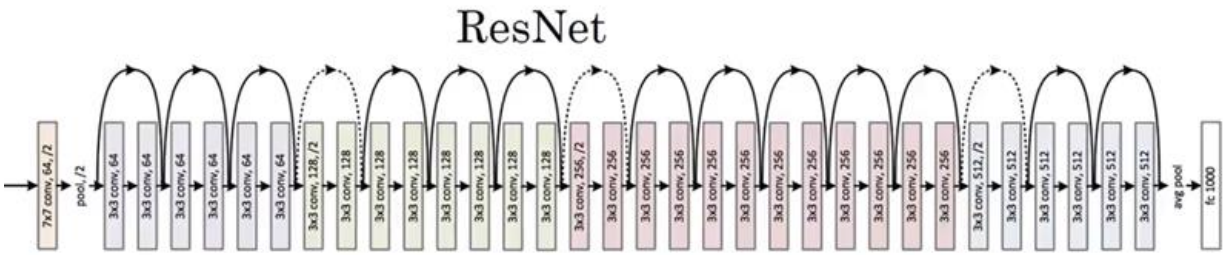

Figura 4.19 Arquitectura ResNet34. ***(Fuente:***(45)***)***

Como se puede observar, la arquitectura consta de una primera capa convolucional y un pooling seguido de cuatro secciones de capas convolucionales y por último un clasificador softmax. Según la cantidad de capas convolucionales que tienen cada sección se configuran las diferentes ResNets (6)(46)–(48).

4.3.3.2. DenseNet

Como se ha comentado, al aumentar la profundidad de una red neuronal convolucional se produce el problema del desvanecimiento del gradiente. Para resolver este problema, las DenseNet utilizan un patrón de conectividad simple para garantizar el máximo flujo de información entre las capas tanto hacia adelante como hacia atrás. Para cada capa, los mapas de activación de todas las capas anteriores se utilizan como entradas, y sus propios mapas de activación se utilizan como entradas en todas las capas posteriores. Esta estructura se denomina *dense block* (7)(46).

La arquitectura de las DenseNet está formada por una capa convolucional al principio, varios dense blocks y varias capas transicionales entre ellos conformadas por una capa de convolución y un pooling. Al final se encuentra otro pooling y el clasificador.

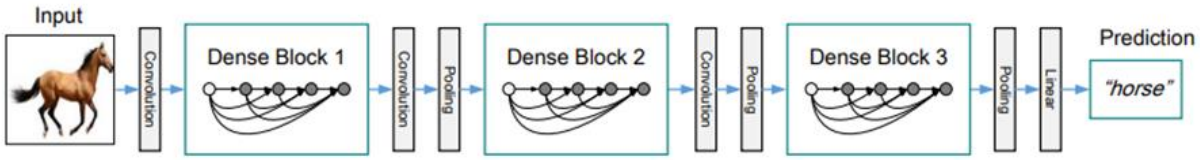

Figura 4.20 Arquitetura DenseNet. ***(Fuente:***(6)***)***

4.3.3.3. AlexNet

Esta arquitectura fue una de las primeras redes profundas. Es una arquitectura de red simple pero potente, allanó el camino hacia una mayor investigación de *deep learning* (49).

AlexNet está constituida por 5 capas convolucionales y 3 capas completamente conectadas. Las dos primeras capas convolucionales están seguidas por capas de Max Pooling. La tercera, cuarta y quinta capa están conectadas directamente. La última capa convolucional vuelve a estar seguida por una de Max Pooling y su salida se dirige a una serie de dos capas completamente conectadas, terminando con un clasificador softmax con 1000 canales (una para cada clase). También añadir que después de todas las capas se aplica ReLU.

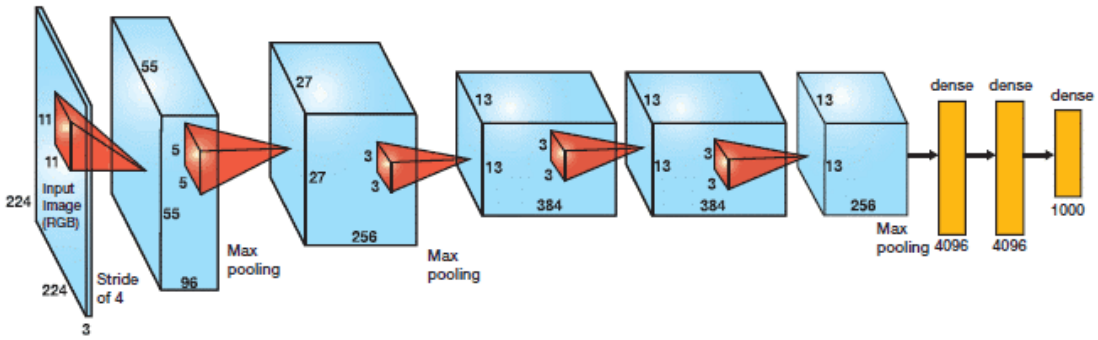

Figura 4.21. Arquitectura AlexNet. ***(Fuente:***(45)***)***

4.3.3.4. VGG

Esta arquitectura es del grupo VGG, de Oxford. Es una mejora sobre AlexNet reemplazando el tamaño los filtros que esta usa en las primeras capas convolucionales (11 y 5 para las dos primeras capas respectivamente) por múltiples filtros de 3 x 3 uno detrás de otro. La estructura consta de una serie de capas convolucionales acompañadas por capas Max-Pooling, seguidas de tres capas totalmente conectadas. La última capa de todas es un clasificador softmax con 1000 canales (una para cada clase de ImageNet). En todas las capas convolucionales se aplica la rectificación no linear ReLU (50).

Existen diversas variaciones de esta arquitectura según el número de capas convolucionales que tengan: VGG11, VGG13, VGG16 o VGG19. Las más utilizadas son las últimas dos.

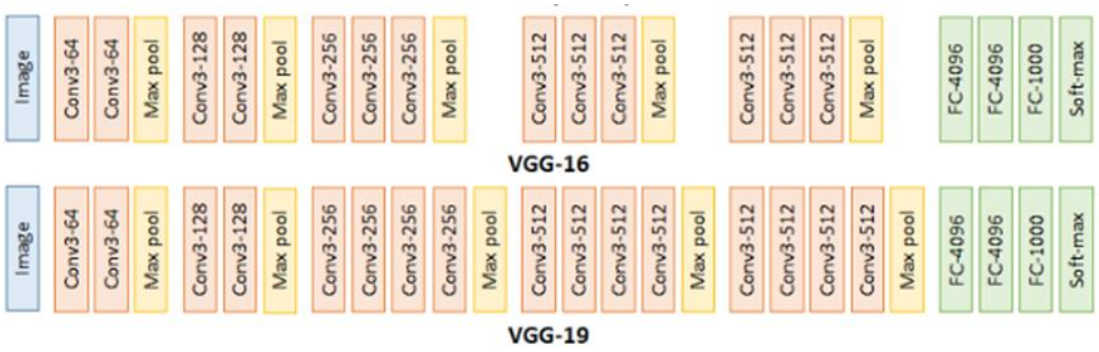

Figura 4.22 Arquitectura VGG16 vs VGG19. ***(Fuente:***(51) ***)***

5. Conjunto de datos

5.1. Distribución del conjunto de datos

Se dispone de un conjunto de datos que contiene ocho clases diferentes de imágenes, cada una con carpeta propia, correspondientes a eritrocitos, plaquetas y seis alteraciones morfológicas típicas de la serie eritrocitaria, comentadas en el *apartado **3.4***, que son esferocitos, drepanocitos, reticulocitos, punteado basófilo, cuerpos de Pappenheimer y cuerpos de Howell-Jolly. En la siguiente tabla se muestra cómo se organizan las imágenes según el tipo de célula, con el nombre de clase asignado y la cantidad de imágenes correspondiente a cada clase.

TIPO DE CÉLULA	CLASE	CANTIDAD DE IMÁGENES
Eritrocitos normales	Normal	9664
Plaquetas	PLT	611
Esferocitos	Esferocito	1035
Drepanocitos	Drepanocitos	550
Reticulocitos	Reticulocito	387
Punteado Basófilo	PB	1204
Cuerpos de Pappenheimer	Pappenheimer	1729
Cuerpos de Howell-Jolly	HJ	1134

Tabla 1 Clasificación del conjunto de clases.

En un principio las imágenes de las células no eran independientes, si no que formaban parte de un frotis con muchas células. Por ello, se procedió a recortarlas en imágenes de 452 x 452 píxeles, donde la célula en cuestión queda generalmente aislada del resto de células y centrada. La obtención de las imágenes digitales de las células, así como su etiquetaje y recorte se llevó a cabo por el grupo de especialistas del Hospital Clínic. El hecho de que queden centradas y aisladas ayuda mucho a la hora de entrenar el modelo, ya que, si la mayor parte de la imagen corresponde a la célula que se estudia, habrá menor ruido, facilitando el aprendizaje.

5.2. Análisis del conjunto de datos

Observando el conjunto de datos se puede apreciar que hay clases que contienen muchas imágenes, como puede ser la de eritrocitos normales, y otras que tienen muchas menos, como es el caso de los reticulocitos. A este problema se le denomina ***desbalanceo de datos.*** Por lo general presenta un problema al entrenar el modelo ya que puede provocar una tendencia hacia las clases más numerosas.

Existen diferentes técnicas para tratar este problema, algunas de las principales son:

- ***Oversampling*.** Es uno de los métodos más utilizados en deep learning. Sencillamente se basa en realizar copias de la clase con menor cantidad de imágenes. Es efectiva, aunque puede inducir *overfitting,* explicado más adelante. En muchas ocasiones es el mejor método (52).
- ***Undersampling.*** Es otro método muy utilizado y es lo contrario al anterior. En lugar de copiar las clases con menor cantidad, elimina de las clases con mayor cantidad de datos.
- ***Data augmentation.*** Es otra técnica bastante utilizada. Consiste en procesar las imágenes generando X variaciones de cada una de ellas. Estas variaciones pueden ser rotaciones, aumento de tamaño, cambio de saturación, cambio en el brillo, entre otros. Hay que tener en cuenta que según qué tipo de variación puede interesar o no. Por ejemplo, en el caso del presente trabajo de clasificación de células no interesaría un cambio de forma, ya que distorsionaría totalmente el resultado. Esta técnica, además de servir para el desbalanceo de datos también es útil para aumentar el tamaño del conjunto de datos y hacerlo más robusto al overfitting.
- Otra opción posible es llevar a cabo una ***combinación*** de ellas, como por ejemplo multiplicar la que tiene menos y reducir la que tiene más, para llegar así a un conjunto de datos más parejo.

Sin embargo, Jeremy Howard, fundador de Fast.ai, en la segunda lección de su curso (3), cuando le preguntan sobre cómo proceder ante clases desbalanceadas, invita a probar a entrenar el modelo aun estando desbalanceado, ya que comenta “Pruébalo. Funciona […] He realizado muchos análisis con datos no balanceados en los últimos años y simplemente no puedo hacer que no funcionen. Siempre funciona”. Por tanto, se realizarán las pruebas con los datos tal y como están, pudiendo comprobar que, efectivamente, funciona y además bastante bien.

5.2.1. Overfitting y underfitting

El *overfitting* o sobreajuste de datos es un problema que surge al entrenar las redes neuronales. Sucede cuando el modelo entrenado tiene un alto rendimiento en el conjunto de datos de entrenamiento, pero cuando trata con datos nuevos, su rendimiento empeora considerablemente. Esto pasa porque

el modelo no sólo ha aprendido a reconocer ciertos tipos de imágenes, si no que ha empezado a aprender dichas imágenes en concreto. Esto se puede deber a varias cosas, pero por lo general suele ser por un número elevado de epochs, un learning rate alto o un conjunto de datos demasiado pequeño.

Para observar si un modelo entrenado tiene sobreajuste se crea un conjunto de datos de validación, que son un conjunto de imágenes que el modelo no ha visto aún. Entonces se comparan los resultados de las métricas utilizadas en el conjunto de entrenamiento y el conjunto de validación. Existen varios métodos para lidiar con este problema como añadir más datos al conjunto, usar *data augmentation*, usar normalización, añadir regularización o reducir la complejidad de la red, evitando así que el modelo se ajuste en exceso al conjunto de datos de entrenamiento.

El caso contrario es el denominado *undefitting,* esto sucede cuando el modelo aún no ha aprendido toda la información y relaciones en el conjunto de entrenamiento. Puede ocurrir cuando los datos son insuficientes, cuando la complejidad del modelo es demasiado baja, pocos *epochs* o *learning rate* demasiado bajo.

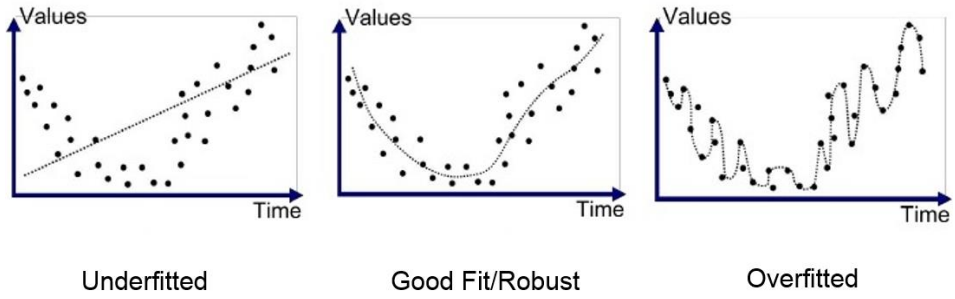

Figura 5.1 Overfitting y underfitting ***(Fuente:***(53)***)***

5.2.2. Conjunto de entrenamiento y validación

Para llevar a cabo los modelos y la manipulación del conjunto de datos se ha utilizado la librería de Fast.ai, que utiliza PyTorch como entorno de trabajo.

Para entrenar el modelo y valorar su rendimiento es necesaria la separación del conjunto de datos entre entrenamiento y validación.

Para utilizar el conjunto de datos con el que se entrenará a la red, y dividirlo en conjuntos de entrenamiento y validación, en fast.ai se crea lo que se denomina un *DataBunch*, donde dentro de él se encuentran estos dos grupos independientes.

Este *Databunch* se crea a partir de la siguiente comanda, que se encarga de coger las imágenes de cada clase y crear los conjuntos de entrenamiento y validación aleatoriamente según un porcentaje proporcionado. En este caso se elige que el conjunto de validación sea el 20 % de los todos datos que se poseen (valid_pct=0.2).

```
data= ImageDataBunch.from_folder("data", valid_pct=0.2, ds_tfms=tfms, size=224).normalize(imagenet_stats)
```

Código 5.1 Creación de los conjuntos de entrenamiento y validación.

Además, también permite aplicar *aumento de datos*, aplicando transformaciones a las imágenes. Estas transformaciones se definen gracias a la función get_transforms().

Las transformaciones que se han aplicado a todos los modelos son las siguientes:

```
tfms= get_transforms(do_flip= True, flip_vert= True, max_warp=0.,max_rotate =360.,
                     max_zoom=1.1, max_lighting=0.2, p_affine=0.75, p_lighting=0.75)
```

Código 5.2 Transformaciones aplicadas

- Giro horizontal
- Giro vertical
- Sin deformaciones, ya que no es producente para el conjunto de datos, cambiaria la forma de la célula.
- Rotaciones, sin máximo porque al ser células no afecta.
- Zoom, con un máximo de 1.1
- Cambio de brillo, con un máximo de 0.2
- Además, se definen los porcentajes con los que cada transformación se aplicará sobre las imágenes (75%)

Después de esto queda un conjunto de entrenamiento con 13052 imágenes y uno de validación con 3262 imágenes.

6. Modelos creados: transfer learning

En el presente trabajo, el objetivo es crear un modelo capaz de clasificar las distintas imágenes de células sanguíneas que se tienen en el conjunto de datos. Para ello se ha utilizado el transfer learning (apartado 4.3), donde se reutiliza un modelo previo desarrollado para una tarea, como punto de inicio para otro modelo en una segunda tarea. Se han llevado a cabo los métodos de ConvNet como extractor fijo de características y el Fine Tuning con diferentes modelos pre entrenados para observar cual es el más preciso.

Todo el código llevado a cabo para crear estos modelos se puede encontrar en el repositorio siguiente de GitHub: https://github.com/JoaquinJustel/TFG (54)

Los modelos pre entrenados (apartado 4.3.3) que se han utilizado para crear los nuevos son los siguientes:

- ResNet 18
- ResNet 34
- ResNet 50
- AlexNet
- VGG 16
- VGG 19
- DenseNet 121
- DenseNet 169

6.1. ConvNet como extractor fijo de características

En primer lugar, se lleva a cabo el método que utiliza la ConvNet pre entrenada como extractor fijo de características, donde al modelo pre entrenado que se escoge se le quita la última capa. Como ya se ha explicado, esta última capa es la que actúa como clasificador en las distintas clases que tenía el conjunto de datos con el que se entrenó. Una vez extraída esta capa, se añade una nueva con parámetros aleatorios, que será la encargada de clasificar los nuevos datos en las nuevas clases. Al llevar a cabo el entrenamiento del modelo se congela todo menos esta última capa, que es la que se entrena y aprende.

Primero se explicará el algoritmo utilizado, sin tener en cuenta qué modelo pre entrenado se ha utilizado.

En primer lugar, se importan y actualizan las librerías que se utilizan:

```
import torch

!pip install fastai --upgrade

%reload_ext autoreload
%autoreload 2
%matplotlib inline

from fastai.vision import *
from fastai.metrics import error_rate, accuracy
```

Código 6.1 Importación de librerías

A continuación, se observa cual es el conjunto de datos, con todas las clases que se tienen.

```
ls data/
```

```
Drepanocitos/  HJ/      PB/   Pappenheimer/  models/
Esferocito/    Normal/  PLT/  Reticulocito/
```

Código 6.2 Conjunto de datos

Habiendo observado cuales son los datos, ahora hay que dividirlos en los conjuntos de entrenamiento y validación, ya que solo están clasificados por la clase a la que pertenecen. Esto se hace a través de una función de fast.ai denominada ImageDataBunch. Esta función permite crear un conjunto de datos de entrenamiento y validación separando los datos que se le proporcionan. Esta separación la lleva a cabo a partir de un porcentaje que se le introduce que indica la cantidad de datos que se cogerán para crear el conjunto de validación. En este caso se ha escogido que el 20 % de los datos vayan al conjunto de validación. También se le especifica el tamaño de las imágenes con el que se trabajará, definido a 224x224 píxeles, que es un tamaño que su uso es muy común y suele dar buenos resultados.

Otro punto interesante es que también permite aplicar ciertas transformaciones a las imágenes, haciendo así *aumento de datos*. Las transformaciones aplicadas están explicadas en el apartado 5.2.2.

```
tfms= get_transforms(do_flip= True, flip_vert= True, max_warp=0.,max_rotate =360.,
                     max_zoom=1.1, max_lighting=0.2, p_affine=0.75, p_lighting=0.75)

data= ImageDataBunch.from_folder("data", valid_pct=0.2, ds_tfms=tfms, size=224).normalize(imagenet_stats)
```

Código 6.3 Creación de los conjuntos de entrenamiento y validación.

Una vez ya se tiene el *DataBunch* con los conjuntos de entrenamiento y validación creados, se procede a crear el modelo. Esto se lleva a cabo con otra función de fast.ai denominada *cnn_learner*. Esta función crea un objeto llamado *Learner a* partir de los datos proporcionados y se encarga de crear un nuevo modelo.

```
arch = 'introducir modelo'
learn = cnn_learner(data, models.arch, metrics=accuracy)
```

Código 6.4 Nuevo modelo creado

Específicamente, carga el modelo pre entrenado definido por la arquitectura que se escoge y lo corta por la última capa convolucional para eliminarla. Después de extraerla añade un seguido de capas que se puede observar en la siguiente imagen. También se define la métrica con la que se juzgará el nuevo modelo, en este caso la exactitud (en el conjunto de validación).

```
(1): Sequential(
  (0): AdaptiveConcatPool2d(
    (ap): AdaptiveAvgPool2d(output_size=1)
    (mp): AdaptiveMaxPool2d(output_size=1)
  )
  (1): Flatten()
  (2): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): Dropout(p=0.25)
  (4): Linear(in_features=4096, out_features=512, bias=True)
  (5): ReLU(inplace)
  (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): Dropout(p=0.5)
  (8): Linear(in_features=512, out_features=8, bias=True)
```

Código 6.5 Capas añadidas al modelo

Las capas añadidas se tratan de una capa de Max Pool y Average Pool concatenadas, seguidas por una capa *Flatten,* y un seguido de bloques de diferentes capas (dropout, batch normalization, transformaciones lineales y ReLU). Este conjunto de capas se añade con parámetros aleatorios y es el que se entrenará a continuación.

Se escoge entrenar a 20 epochs, ya que después de hacer varias pruebas es una cantidad que suele entrenar suficiente al modelo sin caer en el sobreajuste, dando una muy buena precisión.

```
learn.fit_one_cycle(20)
```

Código 6.6 One cycle

Al momento de entrenar la red, en lugar de utilizar un learning rate fijo, es recomendable usar la política de un ciclo o *one cycle policy,* apartado 4.1.3 (55)(29)(30).

Llegados a este punto ya se tiene el nuevo modelo entrenado, lo que se hace a continuación es observar e interpretar los resultados. Para ello, fast.ai cuenta con un seguido de funciones expuestas a continuación que permiten la visualización de ciertos elementos muy útiles para analizar los resultados.

```
interp = ClassificationInterpretation.from_learner(learn)

losses,idxs = interp.top_losses()

len(data.valid_ds)==len(losses)==len(idxs)
```

Código 6.7 Interpretación de resultados

Con la siguiente función se graficar el desarrollo del entrenamiento. Se puede observar cómo han ido evolucionando las pérdidas de entrenamiento y validación durante el entrenamiento.

```
learn.recorder.plot_losses()
```

Código 6.8 Evolución pérdidas

Con la siguiente función se puede observar cuales han sido las imágenes con más pérdida, además están representadas con un mapa de calor que muestra en que parte de la imagen se centró la CNN para clasificarla.

```
interp.plot_top_losses(9, figsize=(15,11))
```

Código 6.9 Mayores pérdidas

Con la siguiente función se dibuja la matriz de confusión.

```
interp.plot_confusion_matrix(figsize=(7,7), dpi=100, normalize = True , norm_dec=2)
```

Código 6.10 Matriz de confusión

Por último, también se puede observar cuales han sido las clases que han resultado con mayor número de imágenes confundidas de clase, se mostrarán las cinco principales.

```
interp.most_confused(min_val=1)
```

Código 6.11 Mayor confusión

Ahora, siguiendo el método explicado, se probarán los diferentes modelos pre entrenados y se observará como se desenvuelven con el conjunto de datos y cuál es el que consigue un mejor desempeño en la clasificación.

6.1.1. ResNet

El primer modelo a analizar es el de las redes residuales (Apartado 4.3.3.1), se probarán tres de ellas (ResNet 18, ResNet 34 y ResNet 50). Existen ResNet de mayor tamaño y por tanto que probablemente ofrezcan mejores resultados en la clasificación, sin embargo, por limitaciones de hardware no han

podido entrenarse. A continuación se hará una comparativa con los resultados del entrenamiento de los tres tipos de ResNet utilizados.

ResNet 18				ResNet 34				ResNet 50			
Epoch	Train loss	Valid loss	Accuracy	Epoch	Train loss	Valid loss	Accuracy	Epoch	Train loss	Valid loss	Accuracy
0	1,036626	0,663381	0,809013	0	1,098042	0,613362	0,833538	0	0,749687	0,500109	0,857756
1	0,544396	0,373075	0,866953	1	0,535764	0,345840	0,878296	1	0,413225	0,298828	0,891784
2	0,381236	0,285514	0,894237	2	0,384470	0,252132	0,902820	2	0,321496	0,219103	0,919988
3	0,321720	0,238144	0,909871	3	0,295735	0,227514	0,912324	3	0,259820	0,185717	0,933783
4	0,270891	0,222403	0,921827	4	0,251307	0,179536	0,932863	4	0,211121	0,157555	0,941140
5	0,228436	0,188552	0,931330	5	0,231817	0,163034	0,936542	5	0,206606	0,145364	0,944206
6	0,216612	0,177729	0,939914	6	0,208599	0,142628	0,947885	6	0,168705	0,138410	0,9491110
7	0,199115	0,152502	0,945432	7	0,188482	0,135597	0,950337	7	0,169820	0,128832	0,954016
8	0,201253	0,153017	0,946045	8	0,169419	0,134627	0,950337	8	0,149421	0,115056	0,959227
9	0,179933	0,149789	0,951563	9	0,154811	0,118581	0,958614	9	0,136593	0,110072	0,960147
10	0,166549	0,135287	0,954016	10	0,151075	0,115048	0,957388	10	0,135140	0,117222	0,958308
11	0,158930	0,130533	0,955242	11	0,131921	0,114995	0,959227	11	0,112905	0,102738	0,963519
12	0,151105	0,129796	0,952790	12	0,141292	0,109802	0,958001	12	0,119056	0,105799	0,961067
13	0,137030	0,126600	0,958001	13	0,129109	0,106172	0,963826	13	0,099969	0,102785	0,963826
14	0,130483	0,126987	0,956775	14	0,125478	0,105536	0,963826	14	0,099329	0,098600	0,964132
15	0,129242	0,121820	0,955855	15	0,111963	0,105522	0,961067	15	0,094687	0,095884	0,965052
16	0,130537	0,119980	0,958614	16	0,116784	0,104648	0,963213	16	0,090970	0,093320	0,964439
17	0,123763	0,120378	0,958001	17	0,112320	0,103733	0,961986	17	0,081066	0,096054	0,965359
18	0,129768	0,124066	0,958001	18	0,108242	0,103282	0,963519	18	0,085701	0,094589	0,965359
19	0,120656	0,120347	0,958308	19	0,100800	0,104066	0,962293	19	0,081830	0,094166	0,965359

Tabla 2. Entrenamiento ResNet

Como se puede observar en los resultados obtenidos al entrenar el modelo usando ResNet 18, ResNet 34 y ResNet 50, a mayor profundidad de la red neuronal convolucional residual, mejor es el resultado obtenido. Esto se debe a que al tener más capas y, por tanto, mayor número de parámetros, tienen mayor capacidad de aprendizaje.

También se observa que a medida que se va entrenando más el modelo (mayor número de epochs) los errores de entrenamiento y validación van disminuyendo. En el caso del entrenamiento, al principio esta mejora es más brusca porque al principio del entrenamiento los parámetros de la última capa son aleatorios.

Otro punto a observar es el progreso de las pérdidas de entrenamiento y validación. En un principio se observa que las pérdidas de entrenamiento son bastante más altas que las de validación, esto significa que el modelo aún no se ha entrenado lo suficiente. A medida que aumentan los epoch las dos pérdidas se van igualando hasta que llega un punto que la de entrenamiento es menor. Cuando llega este punto

y la precisión de validación para de mejorar es importante no entrenar mucho más el modelo, ya que se correría el riesgo del *overfitting*. Cuando ambas convergen significa que se ha escogido un buen learning rate. A continuación, se pueden observar las tres gráficas de la evolución de las pérdidas de los diferentes modelos, en general todos siguen un buen progreso y consiguen un buen entrenamiento.

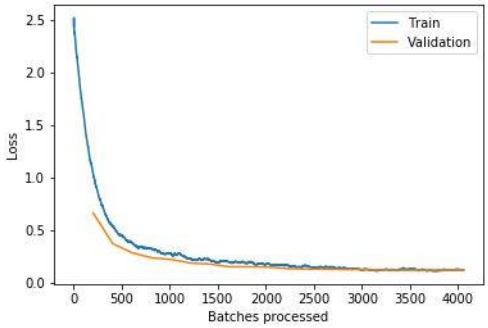

Figura 6.1. Evolución de pérdidas ResNet 18

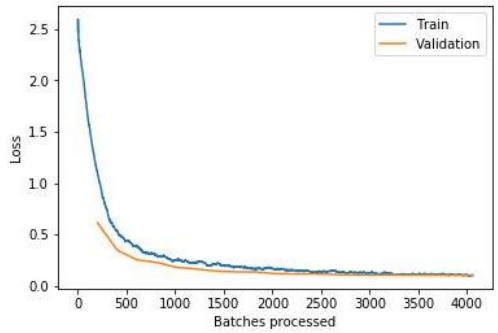

Figura 6.2. Evolución de pérdidas ResNet 34

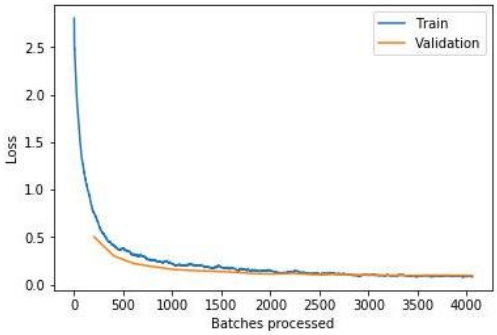

Figura 6.3. Evolución de pérdidas ResNet 50

A continuación se muestran las seis imágenes con mayor pérdida de cada modelo. Esto significa que el modelo está “convencido” de que pertenecen a una clase, sin embargo, pertenecen a otra distinta. Como se observa, en la mayoría la pérdida es bastante grande, ya que la probabilidad que le asigna a

la clase correcta de la célula es nula o casi nula. Además, se puede ver el mapa de calor en cada una de ellas que representa los puntos en los que la CNN se ha centrado para clasificarlas, esto ayuda a ver que, en estas imágenes, en lugar de centrarse en la propia célula, en la gran mayoría se centra en los alrededores.

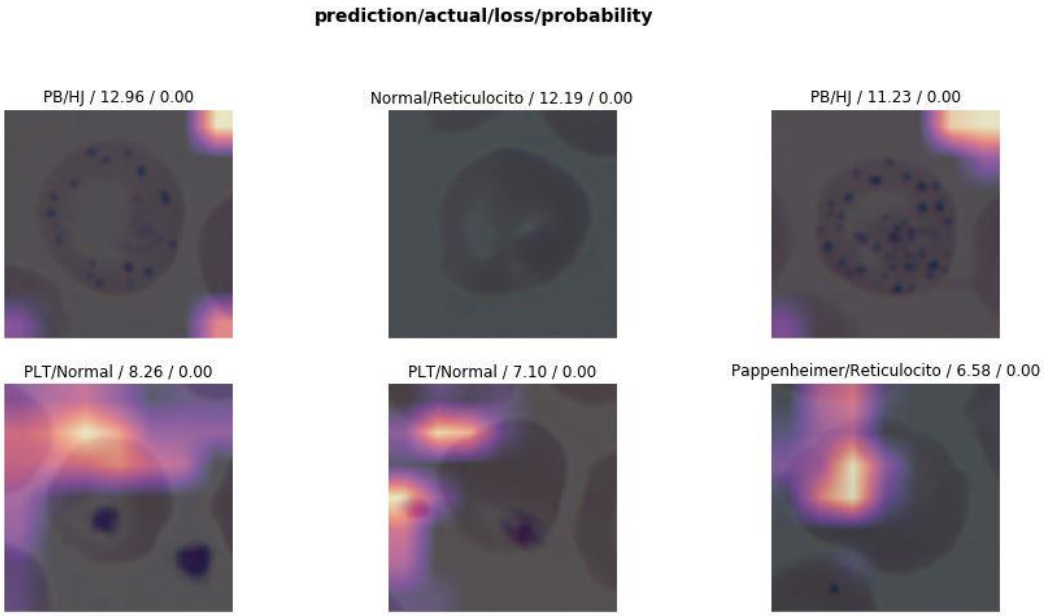

Figura 6.4. Imágenes con mayor pérdida ResNet 18

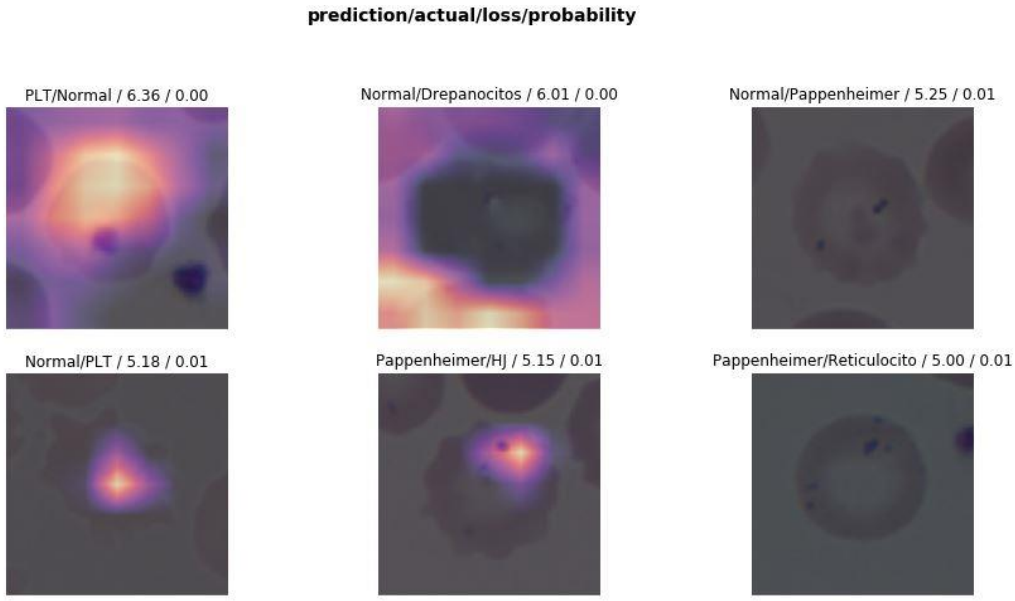

Figura 6.5. Imágenes con mayores pérdidas ResNet 34

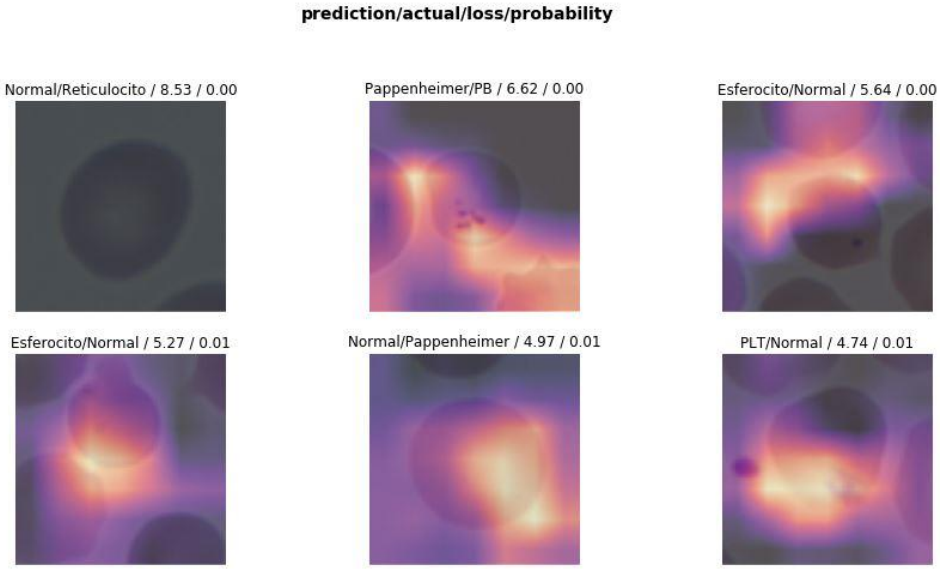

Figura 6.6. Imágenes con mayores pérdidas ResNet 50

Lo siguiente a observar son las matrices de confusión. En ellas se observa cómo ha rendido la red para clasificar las imágenes, mostrando cuan bien ha clasificado cada categoría y con cual las ha confundido con mayor frecuencia. En general los resultados son bastantes satisfactorios, rozando el 100 % en muchas clases. La clase que genera mayor error es la de reticulocitos, pero seguramente se deba a que es la clase con menos imágenes de todas. Sin embargo, con la ResNet 50 también se consigue un muy buen resultado en esa clase.

Figura 6.7 Matriz de Confusión ResNet 18.

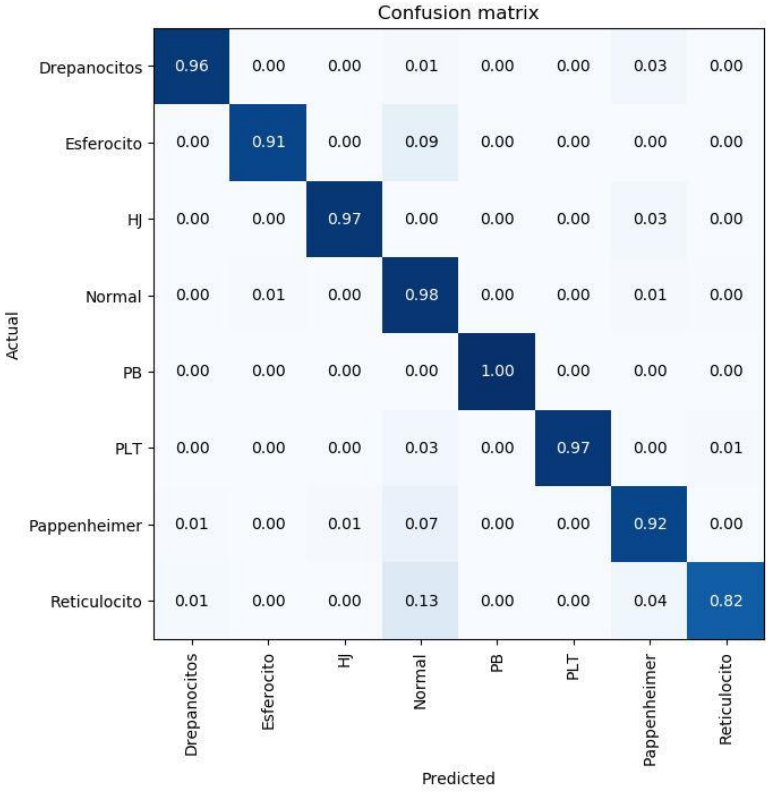

Figura 6.8. Matriz de Confusión ResNet 34

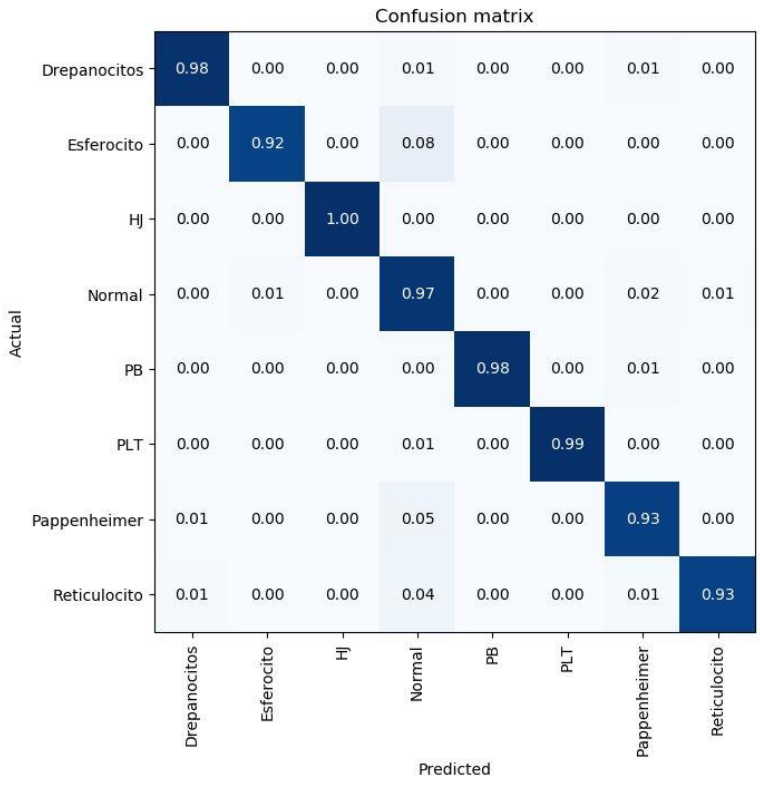

Figura 6.9. Matriz de Confusión ResNet 50

Seguidamente, se observan las clases con mayor número de imágenes mal clasificadas. Observando, las clases que provocan más fallos son las células eritrocitarias normales con las que tienen cuerpos de Pappenheimer y viceversa. Esto puede deberse a las propias imágenes, ya que en algunas imágenes es difícil observar los cuerpos de Pappenheimer a simple a vista. Además, recordar que la clase normal tiene muchas más imágenes, por ello puede haber más fallos que en otras clases, pero representan un pequeño porcentaje del total de ellas.

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	37
Pappenheimer	Normal	21
Normal	Esferocito	19
Esferocito	Normal	17
Reticulocito	Normal	7

Tabla 3. Clases más confundidas Resnet 18

Clase predicha	Clase correcta	Cantidad de errores
Pappenheimer	Normal	25
Normal	Pappenheimer	20
Esferocito	Normal	16
Normal	Reticulocito	12
Reticulocito	Normal	11

Tabla 4. Clases más confundidas ResNet 34

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	30
Normal	Esferocito	17
Esferocito	Normal	16
Pappenheimer	Normal	16
Normal	Reticulocito	10

Tabla 5. Clases más confundidas ResNet 50

6.1.2. Alexnet

El siguiente modelo con el que se llevaron a cabo pruebas es AlexNet (apartado 4.3.3.3). Observando los resultados, es el que peor rendimiento ha tenido y el que mayor número de fallos ha cometido.

AlexNet			
Epoch	**Train loss**	**Valid loss**	**Accuracy**
0	1,043985	0,623959	0,824341
1	0,553787	0,366638	0,883507
2	0,407395	0,298007	0,893930
3	0,392063	0,271808	0,908032
4	0,358608	0,260006	0,905886
5	0,337521	0,250331	0,907112
6	0,322862	0,237253	0,917535
7	0,309648	0,223714	0,920294
8	0,306231	0,220171	0,920294
9	0,302813	0,224409	0,917842
10	0,288934	0,223765	0,920601
11	0,290174	0,204326	0,923666
12	0,257328	0,205190	0,923053
13	0,264397	0,208373	0,922747
14	0,249537	0,197540	0,930717
15	0,243959	0,198520	0,931944
16	0,243457	0,195888	0,931330
17	0,242184	0,192309	0,932863
18	0,223155	0,191503	0,932863
19	0,234188	0,192371	0,932557

Tabla 6. Entrenamiento AlexNet

Analizando el entrenamiento a 20 epochs, se puede observar el descenso de pérdidas a medida que se suceden los epochs como debe ser. Sin embargo, en el último, la pérdida de entrenamiento sigue siendo mayor que la de validación, esto lleva a pensar que todavía le falta entrenar unos epochs más, esta puede ser una posible causa a su menor precisión. A pesar de esto, se observa que en los últimos epochs la precisión no mejora más.

En la gráfica que muestra la evolución de los errores a lo largo del entrenamiento se puede observar también lo comentado, que las dos pérdidas, aunque convergen correctamente no llegan a alcanzarse.

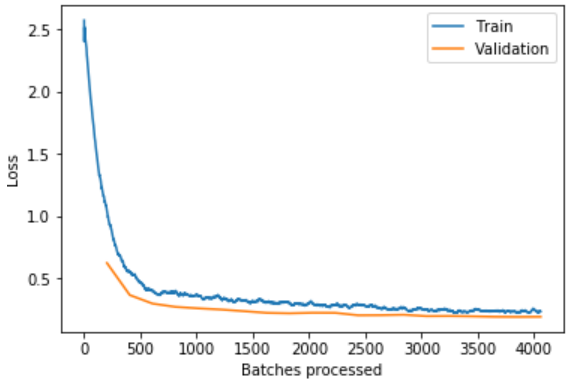

Figura 6.10. Evolución de pérdidas AlexNet

En la matriz de convolución también se puede observar un rendimiento más pobre, sobre todo al clasificar los reticulocitos, que se confunden más de la mitad con eritrocitos normales.

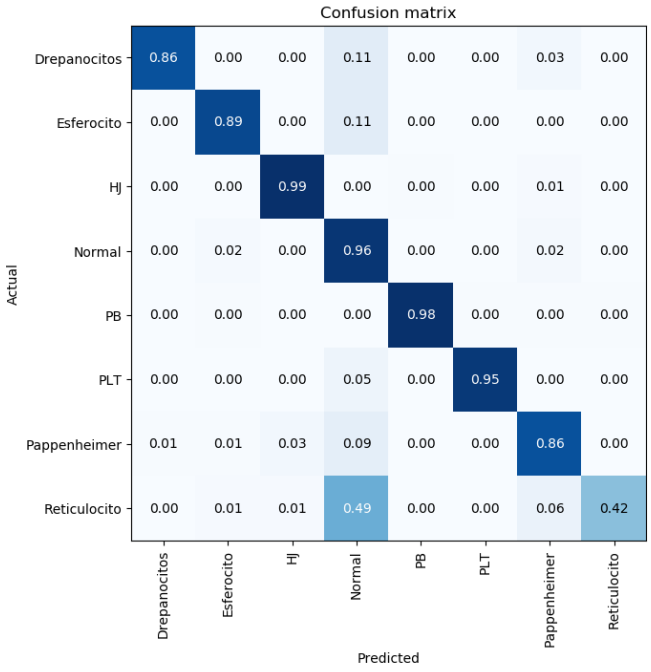

Figura 6.11. Matriz de Confusión AlexNet

Con la información siguiente llama la atención el hecho de que el error de predicción del reticulocito por uno normal no aparezca en la tabla. Esto significa que, aunque se hayan errado más de la mitad de las imágenes de reticulocitos, éstas siguen siendo pocas en número absoluto.

Clase predicha	Clase correcta	Cantidad de errores
Reticulocito	Normal	46
Normal	Pappenheimer	36
Pappenheimer	Normal	31
Normal	Esferocito	30
Esferocito	Normal	21

Tabla 7. Clases más confundidas AlexNet

6.1.3. VGG

Los siguientes modelos a analizar son los VGG 16 y VGG 19 (apartado 4.3.3.4).

VGG 16				VGG 19			
Epoch	**Train loss**	**Valid loss**	**Accuracy**	**Epoch**	**Train loss**	**Valid loss**	**Accuracy**
0	1,102961	0,700351	0,797364	0	1,130012	0,725931	0,789393
1	0,542874	0,386237	0,862661	1	0,584743	0,400090	0,856836
2	0,381284	0,281617	0,892091	2	0,411010	0,311267	0,885653
3	0,318833	0,228766	0,919068	3	0,332571	0,261208	0,905886
4	0,276835	0,199729	0,925812	4	0,267848	0,218275	0,918761
5	0,235503	0,188061	0,932557	5	0,246130	0,204632	0,923666
6	0,219332	0,156599	0,941754	6	0,212392	0,183622	0,932863
7	0,203087	0,145223	0,946045	7	0,194120	0,175149	0,938994
8	0,208813	0,132924	0,946965	8	0,176724	0,165690	0,940221
9	0,176628	0,134887	0,947885	9	0,172743	0,168390	0,938381
10	0,170330	0,134695	0,948191	10	0,171220	0,157438	0,943286
11	0,169456	0,123277	0,953096	11	0,159343	0,142020	0,950337
12	0,139944	0,125982	0,951563	12	0,143800	0,137255	0,947885
13	0,148546	0,122238	0,953709	13	0,139508	0,136414	0,947578
14	0,138971	0,118521	0,953403	14	0,132144	0,129320	0,949724
15	0,130752	0,116790	0,953709	15	0,118575	0,128574	0,950950
16	0,125353	0,114474	0,955549	16	0,126281	0,125312	0,950644
17	0,123015	0,113703	0,954323	17	0,110560	0,122926	0,954016
18	0,130225	0,111291	0,955242	18	0,117208	0,123544	0,953096
19	0,123814	0,112013	0,956468	19	0,116158	0,123272	0,954016

Tabla 8 Entrenamiento VGG

En este caso las pérdidas también disminuyen correctamente y en ambos modelos la precisión empieza a parar de mejorar. Ambos modelos obtienen una precisión bastante buena, entre el 95 y el 96 %, aunque un poco menor que la que se consigue en ResNet 34 y ResNet 50.

En las gráficas de la evolución se puede observar claramente como las dos pérdidas van convergiendo como se espera que lo hagan.

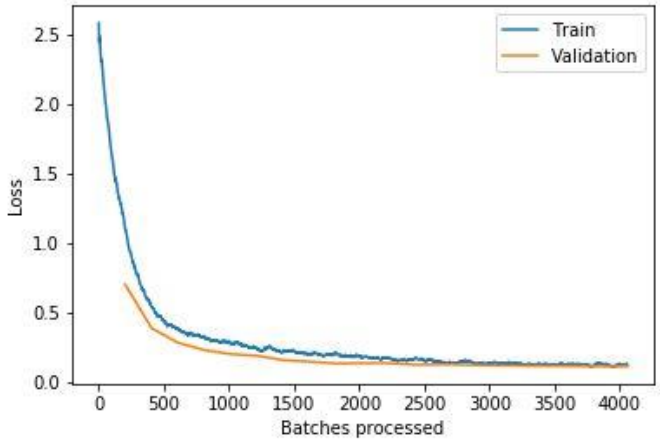

Figura 6.12. Evolución de pérdidas VGG16

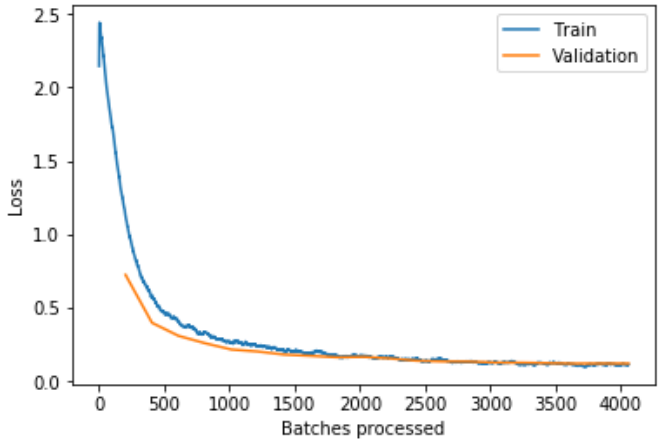

Figura 6.13. Evolución de pérdidas VGG19

Observando las imágenes con más pérdidas, en el caso de la VGG16 vuelven a aparecer los cuerpos de Pappenheimer, en este caso seguramente se debe a la propia imagen que haga difícil el reconocerlos, ya que, según el mapa de calor, la CNN sí que se fijó en las células correctas. También apuntar que son imágenes con mucha pérdida porque la probabilidad de su correcta clasificación vuelve a ser nula o casi nula.

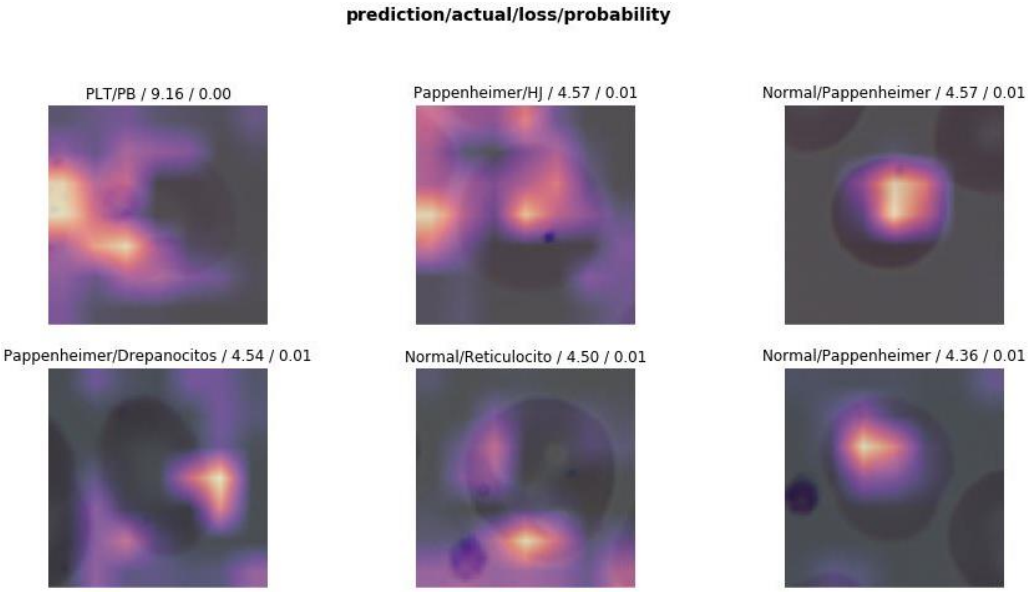

Figura 6.14. Imágenes con mayores pérdidas VGG16

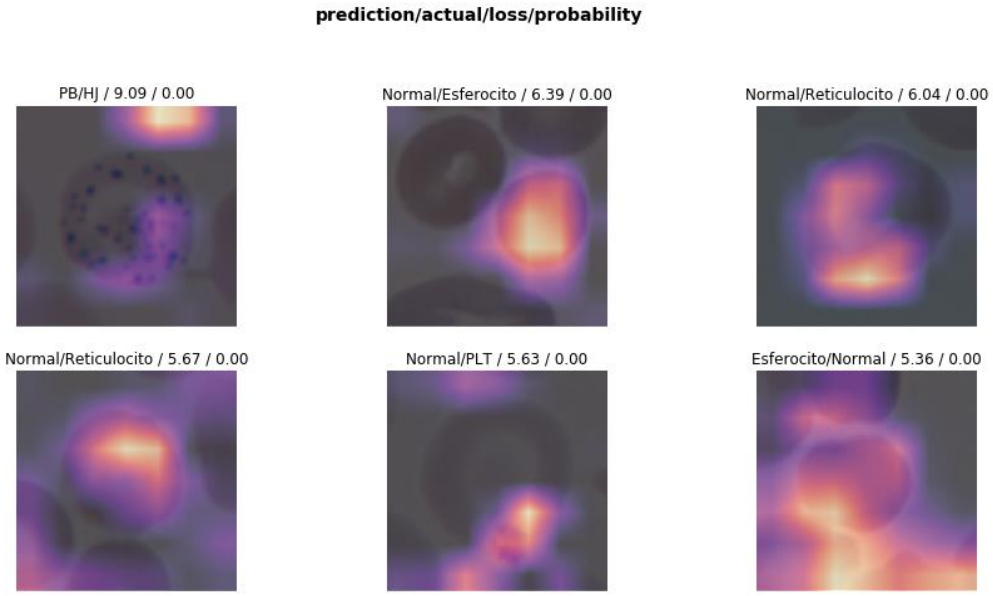

Figura 6.15. Imágenes con mayores pérdidas VGG19

Observando las matrices de confusión se aprecia que, aunque con la mayoría de clases se obtienen unos muy buenos resultados, con los reticulocitos fallan bastante ambas, clasificándolas como eritrocitos normales. Esto, como ya se comentó con anterioridad seguramente se deba al desbalance que existe entre ambas clases, con un conjunto de datos más balanceados probablemente disminuiría el fallo.

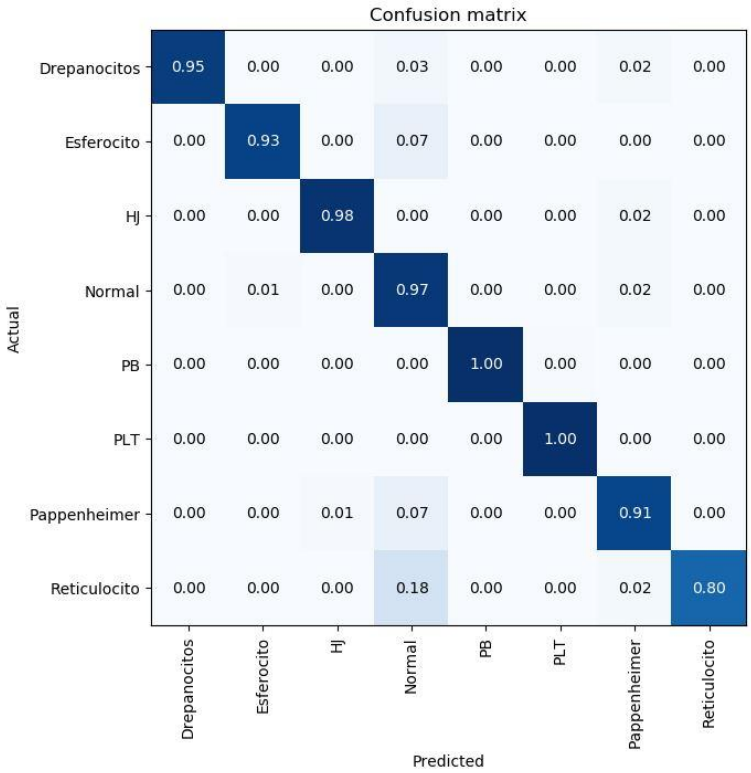

Figura 6.16. Matriz de confusión VGG16

Figura 6.17. Matriz de Confusión VGG19

En cuanto a las clases con mayor número de imágenes mal predichas se vuelve a encontrar las que contienen cuerpos de Pappenheimer con las normales. También los esferocitos confundidos con eritrocitos normales.

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	33
Pappenheimer	Normal	26
Normal	Esferocito	20
Reticulocito	Normal	17
Esferocito	Normal	15

Tabla 9. Clases más confundidas VGG16

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	42
Normal	Esferocito	20
Esferocito	Normal	19
Reticulocito	Normal	19
Pappenheimer	Normal	17

Tabla 10. Clases más confundidas VGG19

6.1.4. Densenet

Los últimos modelos a analizar son los DenseNet 121 y DenseNet 169 (apartado 4.3.3.2)

DenseNet 121				DenseNet 169			
Epoch	**Train loss**	**Valid loss**	**Accuracy**	**Epoch**	**Train loss**	**Valid loss**	**Accuracy**
0	0,878742	0,504674	0,858063	0	0,701923	0,464811	0,871551
1	0,412599	0,251231	0,911097	1	0,345235	0,233052	0,917535
2	0,287775	0,178741	0,934396	2	0,261248	0,173888	0,940834
3	0,225221	0,156832	0,947885	3	0,205621	0,148685	0,944819
4	0,191637	0,152059	0,946045	4	0,186623	0,132426	0,954323
5	0,172091	0,110623	0,960454	5	0,158435	0,124892	0,952483
6	0,155849	0,124629	0,951257	6	0,142244	0,116594	0,960147
7	0,144209	0,101238	0,965972	7	0,132809	0,103896	0,961373
8	0,126785	0,095030	0,965972	8	0,106259	0,101681	0,963826
9	0,118798	0,090057	0,965972	9	0,112085	0,103326	0,963213
10	0,103750	0,094100	0,966585	10	0,099429	0,094139	0,967198
11	0,106008	0,090226	0,965665	11	0,088322	0,091672	0,967198
12	0,101310	0,086117	0,967811	12	0,082286	0,096959	0,964439
13	0,090491	0,085107	0,968424	13	0,079145	0,094808	0,965665
14	0,089517	0,080816	0,971183	14	0,070374	0,096609	0,965665
15	0,081897	0,081878	0,969651	15	0,063406	0,094180	0,965972
16	0,075107	0,081100	0,969957	16	0,066659	0,088528	0,968118
17	0,071323	0,078587	0,972103	17	0,052827	0,091160	0,967198
18	0,063549	0,080870	0,970877	18	0,051358	0,091203	0,966278
19	0,066757	0,078730	0,971183	19	0,044355	0,091320	0,968118

Tabla 11. Entrenamiento DenseNet

Observando el progreso de las pérdidas se aprecia el mismo patrón que las demás, que es el adecuado, las dos pérdidas disminuyen y la de entrenamiento baja más que la de validación. Sin embargo, en el caso del modelo de DenseNet 169 la pérdida de entrenamiento es aproximadamente la mitad que la de validación, lo que junto a la observación de que la precisión tampoco mejora, puede llevar a pensar que empieza a haber un poco de overfitting. Seguramente se deba a un exceso de entrenamiento.

En las gráficas se ve que en DenseNet 121 sigue un patrón correcto, mientras que en la de DenseNet 169 al final se puede observar como las pérdidas se vuelven a separar, confirmando lo comentado anteriormente.

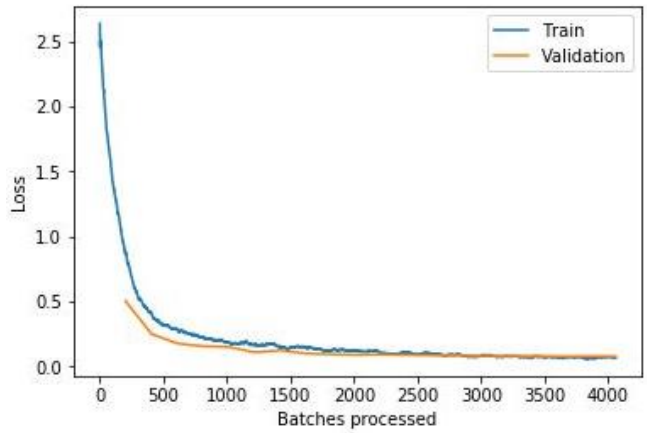

Figura 6.18. Evolución pérdidas DenseNet 121

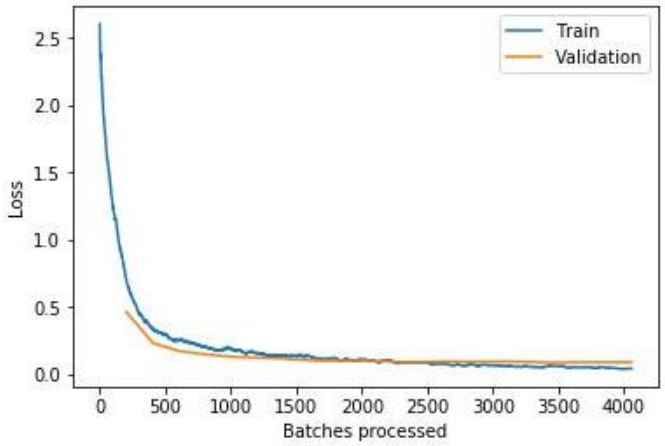

Figura 6.19. Evolución pérdidas DenseNet 169

En las imágenes con mayor pérdida se observa que en algunas de ellas la CNN se centra más en los alrededores de la célula que en la propia célula, lo que puede llevar a su mala clasificación.

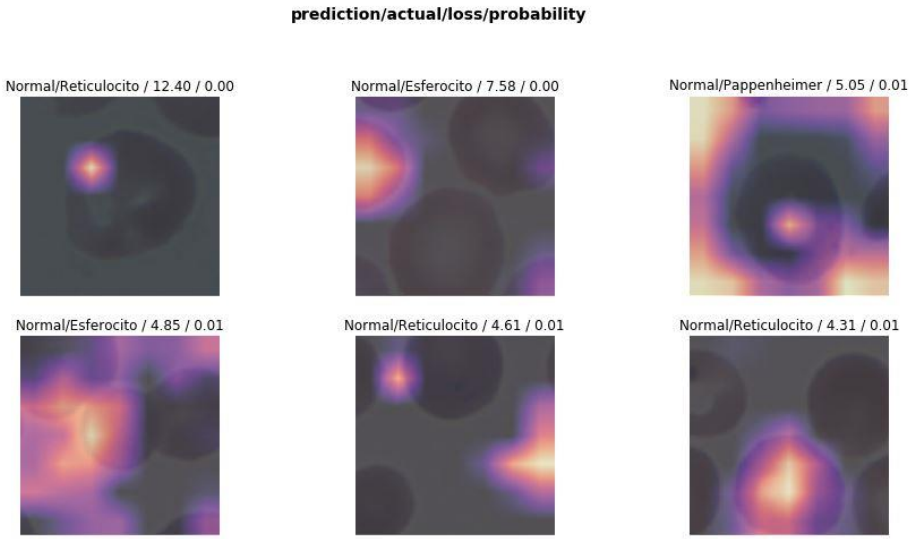

Figura 6.20. Imágenes con mayor pérdida DenseNet 121

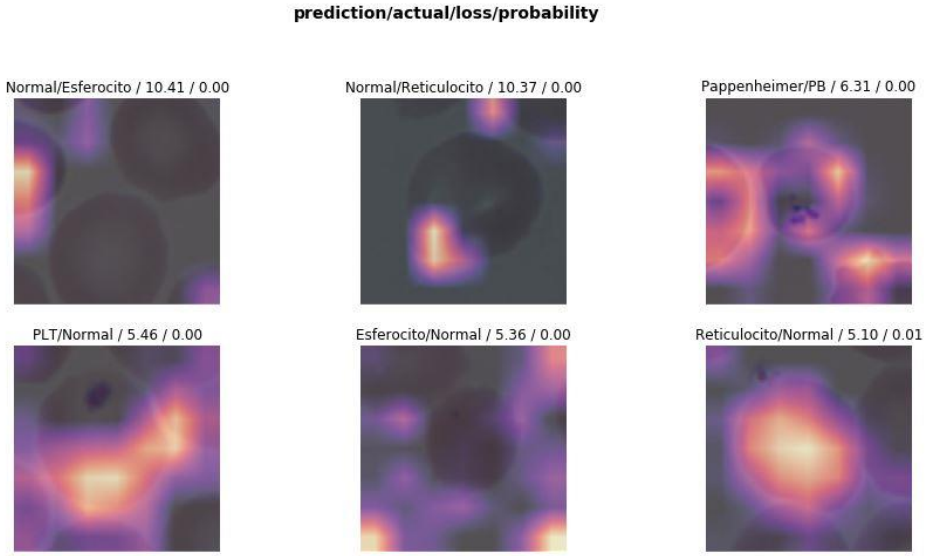

Figura 6.21. Imágenes con mayor pérdida DenseNet 169

Observando las matrices de confusión se puede apreciar que en el modelo de DenseNet 121 se tiene una clasificación casi perfecta en la mayoría de las clases. La única clase que tiene un mayor porcentaje de fallos es, otra vez, el de reticulocitos confundidos con células normales. En el modelo de DenseNet 169 mejora en esta clase, pero empeora un poco con los drepanocitos.

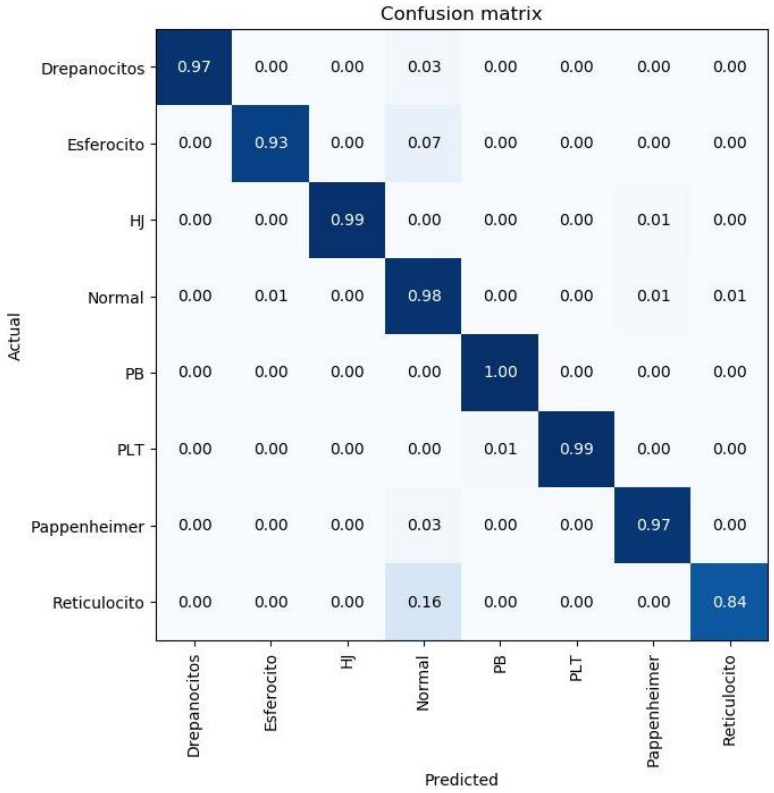

Figura 6.22. Matriz de Confusión DenseNet 121

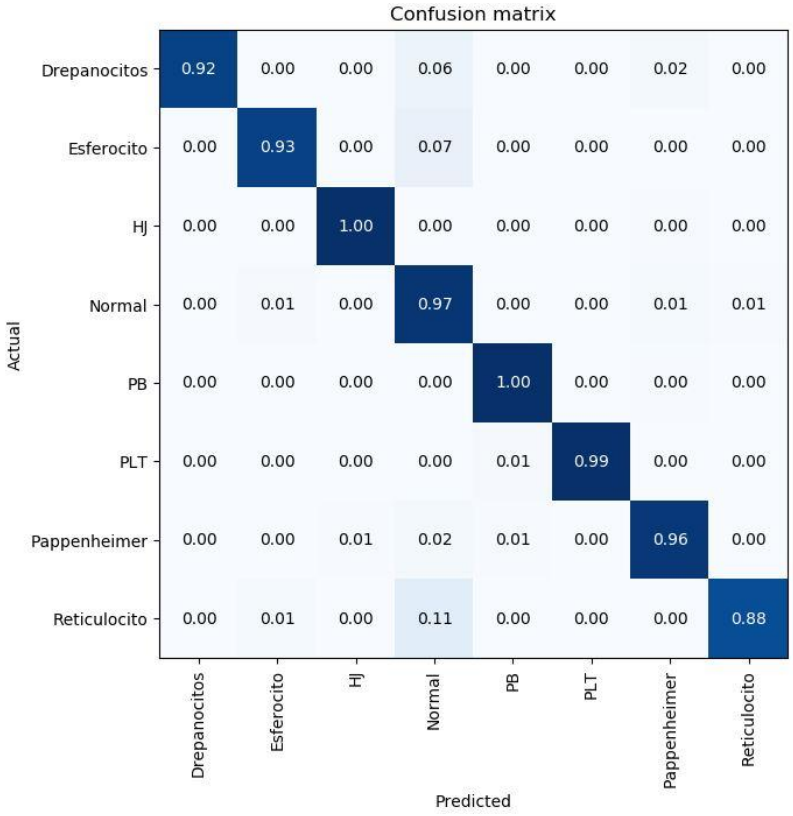

Figura 6.23 Matriz de Confusión DenseNet 169

Como en el resto de modelos vistos, la mayor cantidad de imágenes mal clasificadas corresponde a eritrocitos con cuerpos de Pappenheimer clasificados como normales. Las siguientes clases que también tienen mayor cantidad de imágenes con fallos son las de la relación esferocito con las normales.

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	18
Esferocito	Normal	15
Normal	Esferocito	15
Reticulocito	Normal	12
Normal	Reticulocito	11
Pappenheimer	Normal	10

Tabla 12. Clases más confundidas DenseNet 121

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	24
Normal	Esferocito	17
Esferocito	Normal	15
Normal	Reticulocito	13
Pappenheimer	Normal	8
Reticulocito	Normal	8

Tabla 13. Clases más confundidas DenseNet 169

6.2. Fine tuning

Fine tuning (apartado 4.3.2) es la otra técnica utilizada. En este caso, además de reemplazar la última capa del modelo pre entrenado por una nueva, se descongelan los parámetros del modelo pre entrenado y se entrena todo con el nuevo conjunto de datos. En este método, en lugar de crear modelos nuevos otra vez, se utilizarán los modelos ya creados en el anterior, teniendo así el cuerpo de la ConvNet inicial y la última capa personalizada ya entrenada.

Entonces, siguiendo el último paso del método anterior se procede a descongelar los parámetros:

```
learn.unfreeze()
```

Código 6.12 Descongelado de parámetros

Después de descongelar los parámetros, se vuelve a entrenar el modelo, pero antes de eso se busca averiguar cuál es el *learning rate* más adecuado para llevarlo a cabo. Para observarlo se hace lo siguiente:

```
learn.lr_find()
```

Código 6.13 Búsqueda de learning rate

```
learn.recorder.plot()
```

Código 6.14 Gráfico learning rate

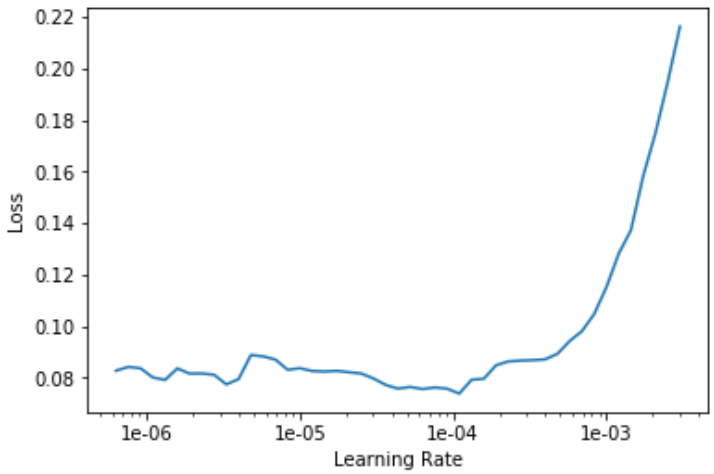

Figura 6.24 Evolución pérdida - Learning rate

En este gráfico se muestra la pérdida en función de cómo aumenta el *learning rate*. Para escoger un buen *learning rate* se debe escoger el máximo que se usará en el punto donde la pérdida se dispara

(1e-04). A continuación, se entrena el modelo. En este punto, en lugar de entrenar siempre con un mismo valor de *learning rate*, se hará por divisiones, se le pasa un rango. Esto se hace así porque como las primeras capas del modelo pre entrenado son bastante generales y ya funcionan bien, se pueden entrenar a un ritmo más rápido que las últimas. Es decir, las primeras capas las entrena con un learning rate de 1e-6 y las últimas con un learning rate de 1e-4, y entremedio se distribuyen las capas entre esos dos valores.

```
learn.fit_one_cycle(5, max_lr=slice(1e-6,1e-4))
```

Código 6.15 Entrenamiento Fine Tuning

Una vez ya entrenado la nueva red, se procede a observar los resultados de la misma manera que con el método anterior.

```
interp = ClassificationInterpretation.from_learner(learn)

losses,idxs = interp.top_losses()

len(data.valid_ds)==len(losses)==len(idxs)
```

Código 6.16 Interpretación resultados

```
learn.recorder.plot_losses()
```

Código 6.17 Evolución pérdidas

```
interp.plot_top_losses(9, figsize=(15,11))
```

Código 6.18 Mayores pérdidas

```
interp.plot_confusion_matrix(figsize=(7,7), dpi=100, normalize = True , norm_dec=2)
```

Código 6.19 Matriz de confusión

```
interp.most_confused(min_val=1)
```

Código 6.20 Mayor confusión

6.2.1. ResNet

Ahora es el turno de aplicar fine tuning a las tres ResNets. Después de descongelar los parámetros y elegir el learning rate adecuado según se ha comentado con anterioridad, se procede a entrenar los modelos a 5 epochs.

ResNet 18				ResNet 34				ResNet 50			
Epoch	Train loss	Valid loss	Accuracy	Epoch	Train loss	Valid loss	Accuracy	Epoch	Train loss	Valid loss	Accuracy
0	0,121416	0,124171	0,956162	0	0,109943	0,109140	0,960454	0	0,087084	0,093848	0,966278
1	0,116608	0,120259	0,956775	1	0,123102	0,102257	0,962906	1	0,092577	0,097511	0,965665
2	0,112572	0,117441	0,957082	2	0,095686	0,101908	0,963213	2	0,084529	0,095763	0,966278
3	0,109399	0,116771	0,960147	3	0,094127	0,095835	0,963826	3	0,080964	0,090033	0,966585
4	0,103275	0,118065	0,959841	4	0,087711	0,093760	0,963213	4	0,071295	0,090115	0,968118

Tabla 14 Entrenamiento ResNet Fine Tuning

Se consiguen valores de precisión bastante buenos, sobre todo en la ResNet 50 con un 96.8 %, en los tres casos mejoran la precisión con respecto al resultado anterior.

En este método se ve cómo las pérdidas de entrenamiento aumentan en un principio y después van disminuyendo poco a poco. Esto es consecuencia de usar one cycle, porque al principio el learning rate sube un poco antes de ir bajando. Lo que hace es que empiece bajo, suba, y vuelva a bajar.

Este patrón se puede divisar en los tres casos. Observando cómo evolucionan las pérdidas de entrenamiento y validación en los tres casos, se puede apreciar que, tanto en ResNet 18 como ResNet 50, ambas van bajando, pero llega un punto en el que las pérdidas de validación vuelven a subir y se separan de las de entrenamiento. Esto podría ser un indicativo de comienzo de overfitting.

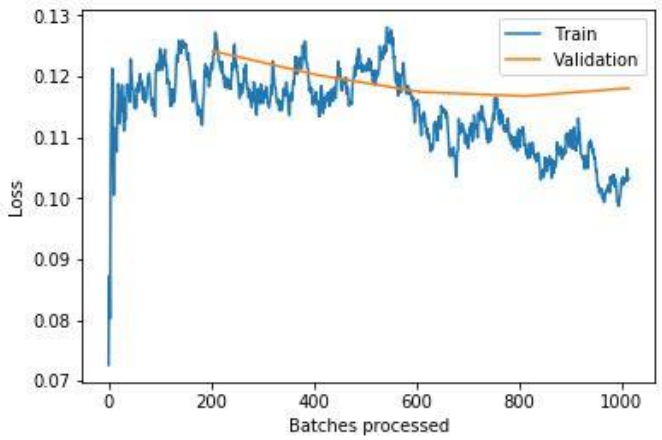

Figura 6.25. Evolución pérdidas ResNet 18 Fine Tuning

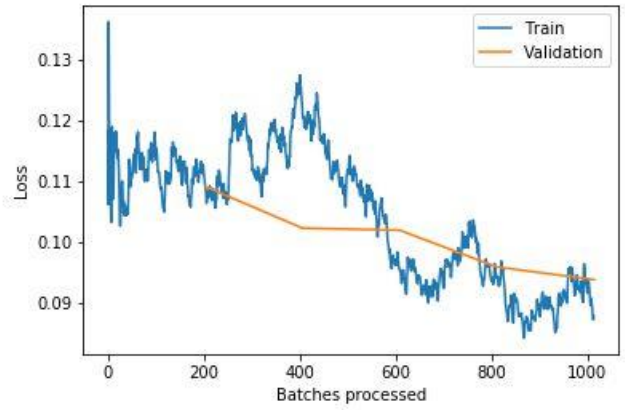

Figura 6.26. Evolución pérdidas ResNet34 Fine Tuning

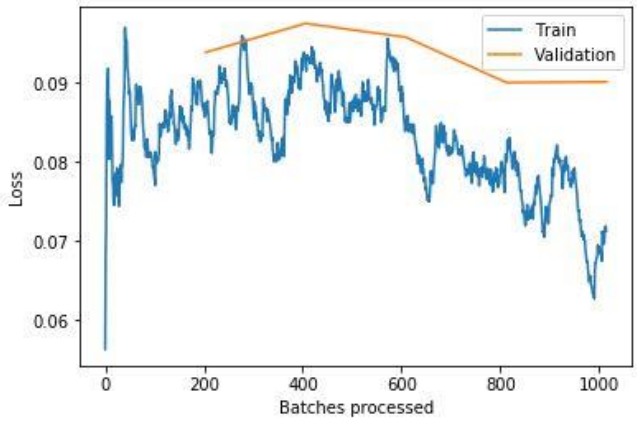

Figura 6.27 Evolución pérdidas ResNet50 Fine Tuning

Mirando las imágenes con mayores pérdidas, se observa que en muchas de ellas la CNN vuelve a focalizarse en los alrededores de la célula para su clasificación, haciendo que esta sea incorrecta. También hay un par de casos en la ResNet 18 donde las clasifica como plaqueta cuando en realidad corresponden a normales, observando las imágenes se puede ver que efectivamente el eritrocito es normal, pero que también hay presencia de plaquetas.

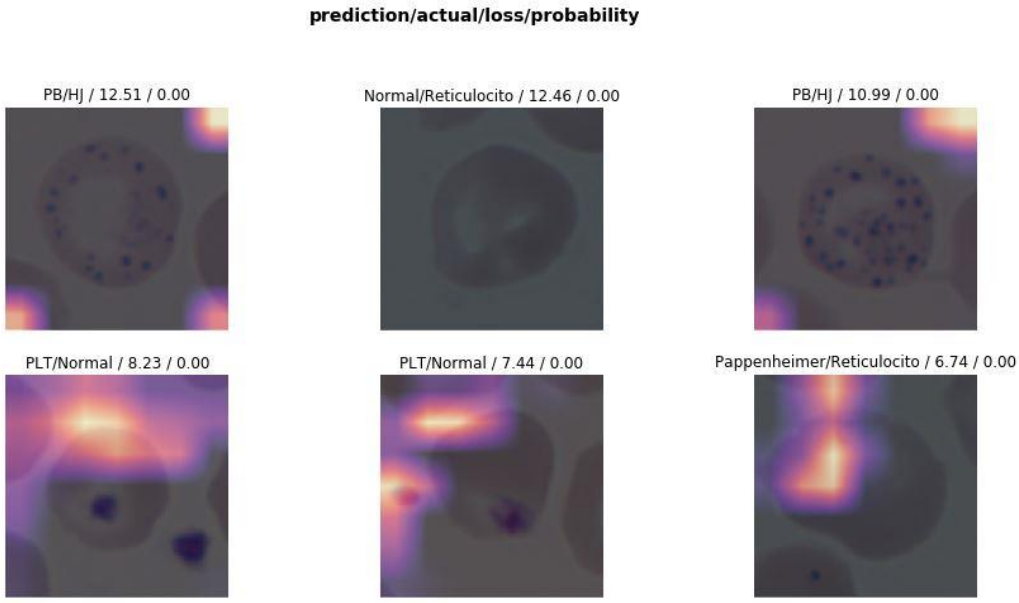

Figura 6.28. Imágenes con mayores pérdidas ResNet18 Fine Tuning

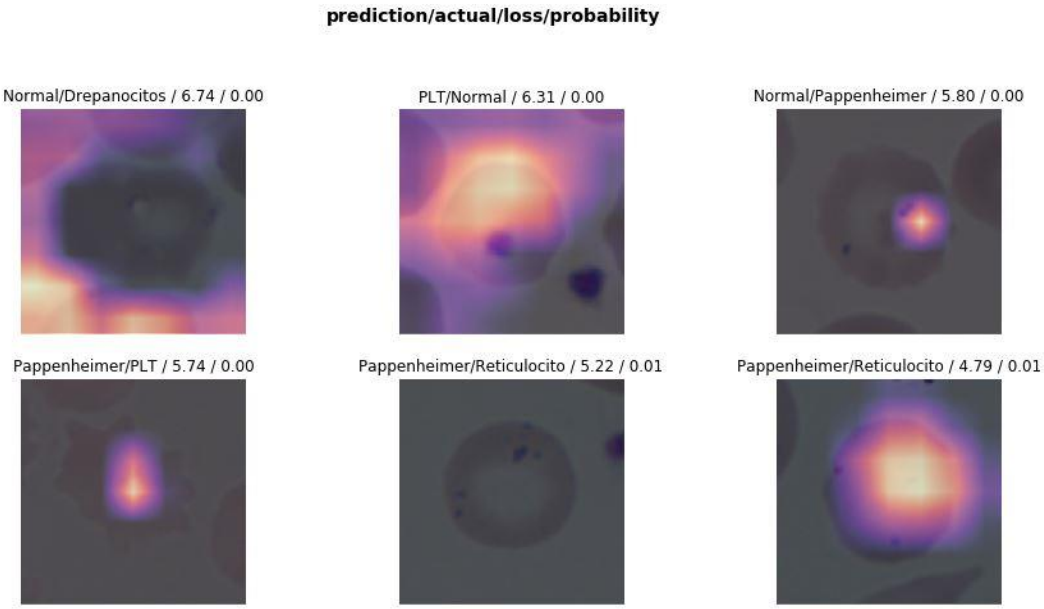

Figura 6.29. Imágenes con mayores pérdidas ResNet 34 Fine Tuning

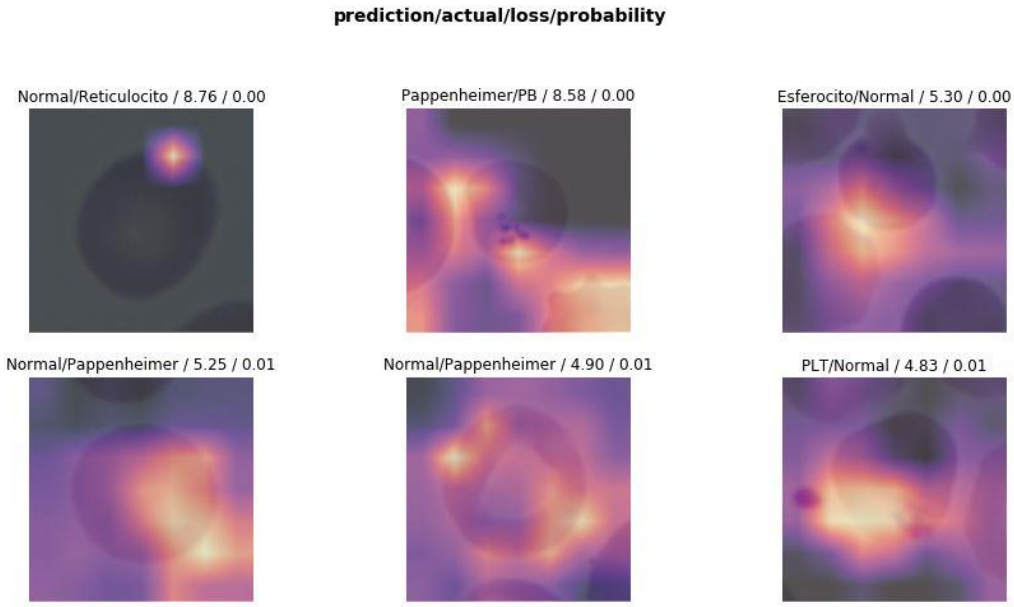

Figura 6.30 Imágenes con mayores pérdidas ResNet50 Fine Tuning

Observando el resultado de las matrices de confusión, vuelve a presentarse un menor desempeño con la clasificación de reticulocitos (bastante menos acentuado en ResNet 50). El resto de clases sigue teniendo un buen resultado en su clasificación.

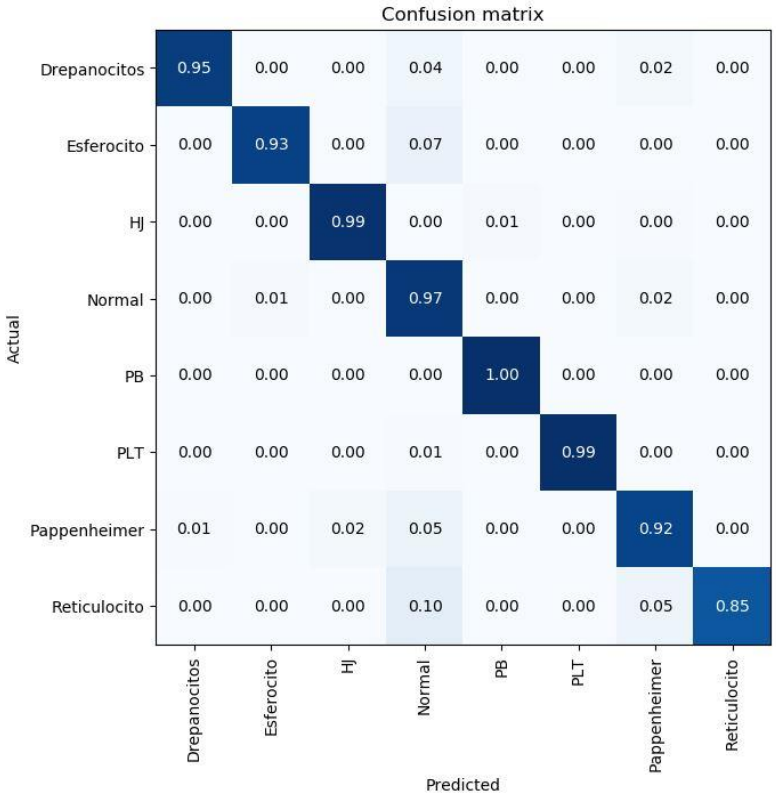

Figura 6.31 Matriz de confusión ResNet18 FineTuning

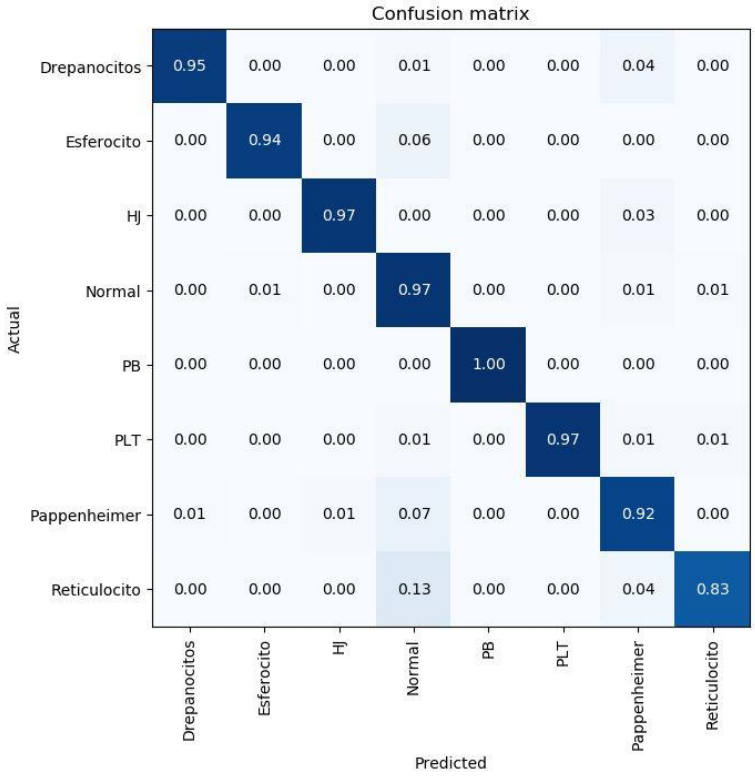

Figura 6.32 Matriz de Confusión ResNet34 Fine Tuning

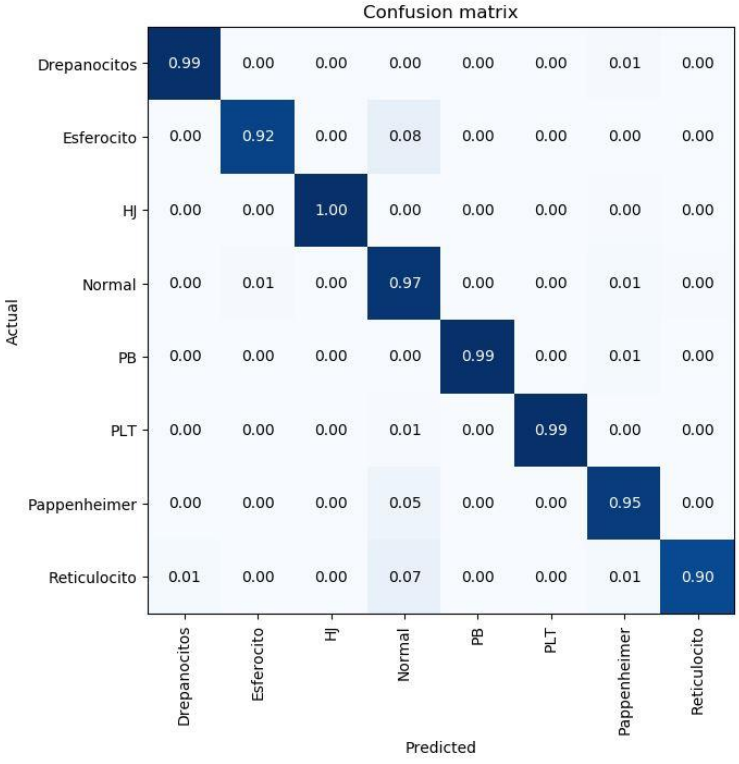

Figura 6.33. Matriz de confusión ResNet50 Fine Tuning

Viendo las clases que producen mayor cantidad de imágenes mal clasificadas, se encuentra otra vez la clase de cuerpos de Pappenheimer confundidos con eritrocitos normales y también a la inversa. Otra confusión remarcable es la que se produce entre esferocito y normal, y viceversa.

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	34
Normal	Esferocito	21
Pappenheimer	Normal	19
Esferocito	Normal	13
Reticulocito	Normal	8

Tabla 15. Clases más confundidas ResNet 18 Fine Tuning

Clase predicha	Clase correcta	Cantidad de errores
Pappenheimer	Normal	24
Normal	Pappenheimer	18
Normal	Esferocito	12
Normal	Reticulocito	12
Esferocito	Normal	11

Tabla 16. Clases más confundidas ResNet 34 Fine Tuning

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	26
Normal	Esferocito	17
Esferocito	Normal	16
Pappenheimer	Normal	15
Normal	Reticulocito	8

Tabla 17. Clases más confundidas ResNet 50 Fine Tuning

6.2.2. AlexNet

El siguiente modelo en aplicar fine tuning es el de AlexNet, también entrenado a 5 epochs.

AlexNet			
Epoch	**Train loss**	**Valid loss**	**Accuracy**
0	0,224679	0,176156	0,935316
1	0,198831	0,154817	0,945739
2	0,165653	0,149928	0,947272
3	0,178592	0,140663	0,952177
4	0,161028	0,141842	0,951257

Tabla 18 Entrenamiento AlexNet Fine Tuning

Mirando los valores de la tabla de las pérdidas de entrenamiento y validación, éstas van disminuyendo correctamente, aunque al llegar al final del entrenamiento la de validación sigue siendo más baja, por lo que significa que se podría seguir entrenando el modelo un poco más.

Observando la gráfica de la evolución de ambas perdidas se puede decir que presenta un buen perfil de evolución, donde la pérdida de entrenamiento primero sube bruscamente para después ir bajando y la pérdida de validación haciendo lo mismo.

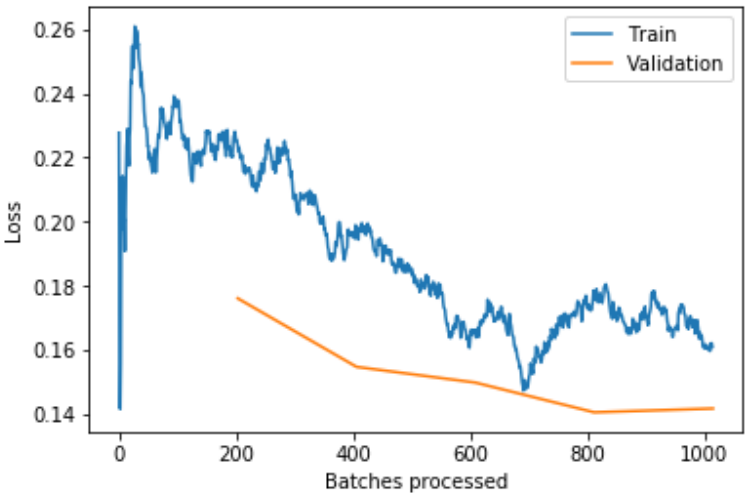

Figura 6.34. Evolución pérdidas AlexNet Fine Tuning

Como se puede observar a continuación, cuatro de las seis imágenes con mayores pérdidas corresponden a la mala clasificación de reticulocitos.

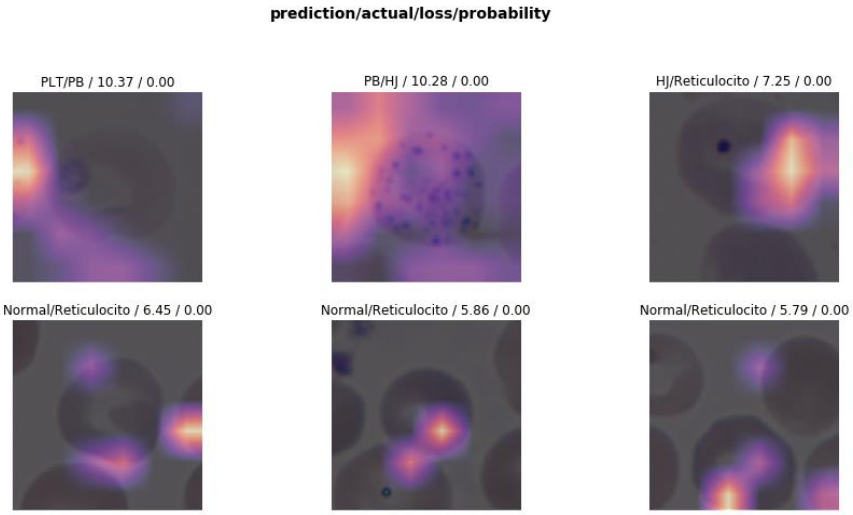

Figura 6.35 Imágenes con mayores pérdidas AlexNet Fine Tuning

En cuanto a la matriz de confusión, se obtienen buenos resultados en la mayoría, aunque con los reticulocitos los resultados son realmente malos, llegando a fallar el 40 %. Sin embargo, existe mejoría con respecto a los resultados anteriores.

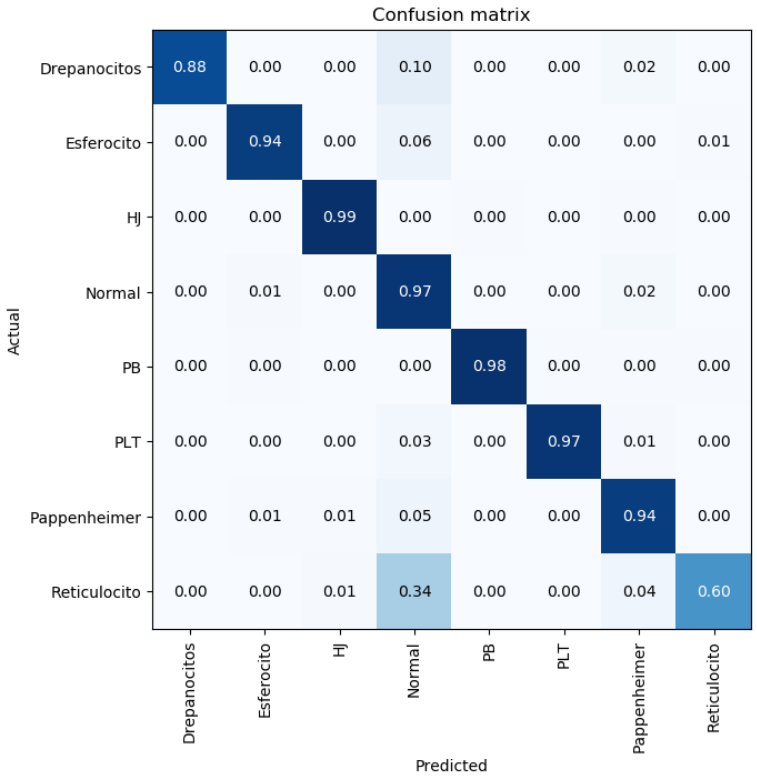

Figura 6.36 Matriz de confusión AlexNet Fine Tuning

En este modelo también destaca la cantidad de imágenes mal predichas, empezando por las de Pappenheimer como normales terminando por los reticulocitos como normales. Llama la atención en este último caso comparándolo con la matriz de confusión, ya que las imágenes de reticulocitos predichas como normales son sólo 5, sin embargo, corresponden al 34 % de las totales de esta clase. Aquí se acentúa el problema de tener un desajuste tan grande con esta clase. Con un conjunto de datos con mayor cantidad de imágenes de reticulocitos se podría observar mejor la precisión de los modelos.

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	38
Reticulocito	Normal	32
Normal	Esferocito	21
Pappenheimer	Normal	16
Esferocito	Normal	11
Drepanocito	Normal	10
Normal	Reticulocito	5

Tabla 19. Clases más confundidas AlexNet Fine Tuning

6.2.3. VGG16

VGG 16				VGG 19			
Epoch	**Train loss**	**Valid loss**	**Accuracy**	**Epoch**	**Train loss**	**Valid loss**	**Accuracy**
0	0,129184	0,112343	0,955242	0	0,119802	0,123259	0,953403
1	0,118790	0,110078	0,955855	1	0,123396	0,119950	0,949418
2	0,111215	0,108219	0,953709	2	0,108088	0,112209	0,955855
3	0,101816	0,104994	0,955242	3	0,096031	0,107728	0,958308
4	0,099929	0,104806	0,956162	4	0,086993	0,104284	0,963213

Tabla 20 Entrenamiento VGG Fine Tuning

Con los modelos de VGG también se observa esta subida inicial de pérdidas con la posterior bajada. Se llegan a obtener muy buenos resultados, en el caso de VGG 19 supera la precisión conseguida anteriormente.

En los gráficos también se puede observar una correcta evolución de ambas pérdidas, lo que significa que se ha llevado a cabo un buen entrenamiento.

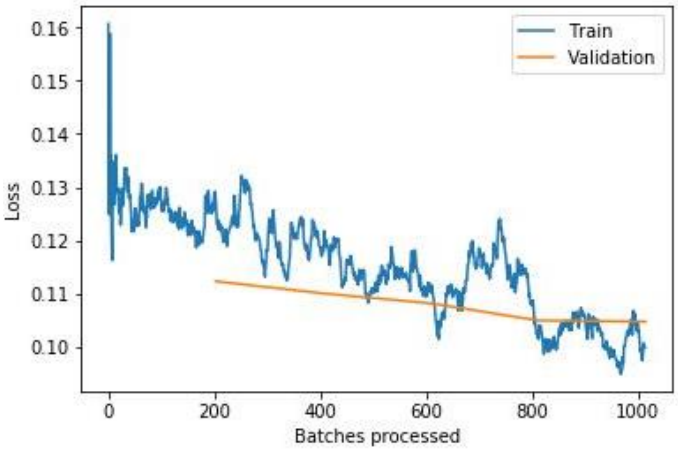

Figura 6.37Evolución pérdidas VGG16 Fine Tuning

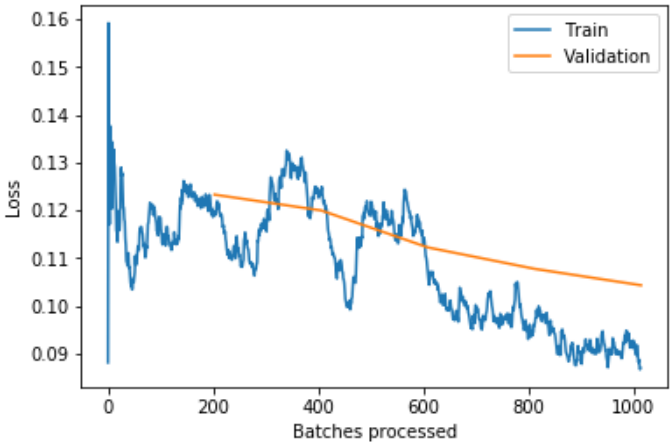

Figura 6.38 Evolución pérdidas VGG19 Fine Tuning

Las siguientes imágenes son las que han presentado una mayor pérdida para cada modelo. Observándolas, se podría deber ya sea a cierta similitud entre algunas clases en la imagen, o que la CNN se ha centrado más en el lugar equivocado para su correcta clasificación.

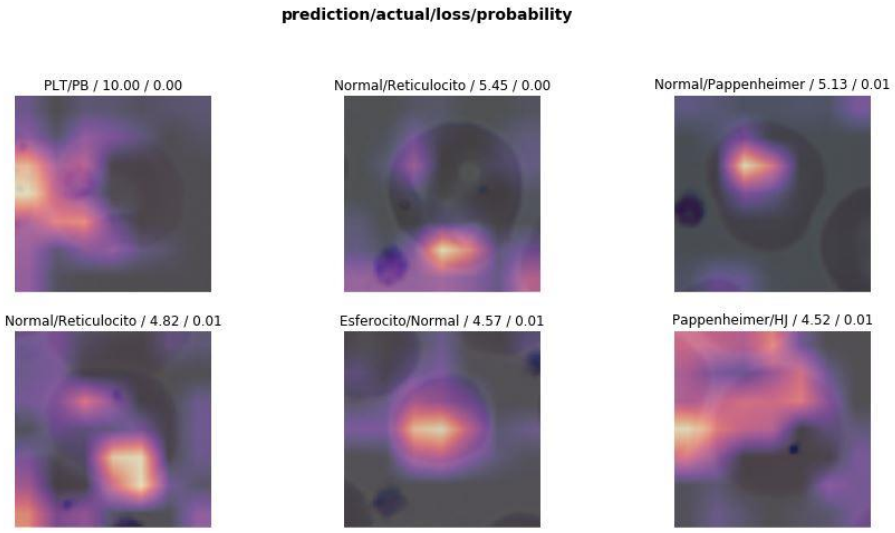

Figura 6.39 Imágenes con mayores pérdidas VGG16 Fine Tuning

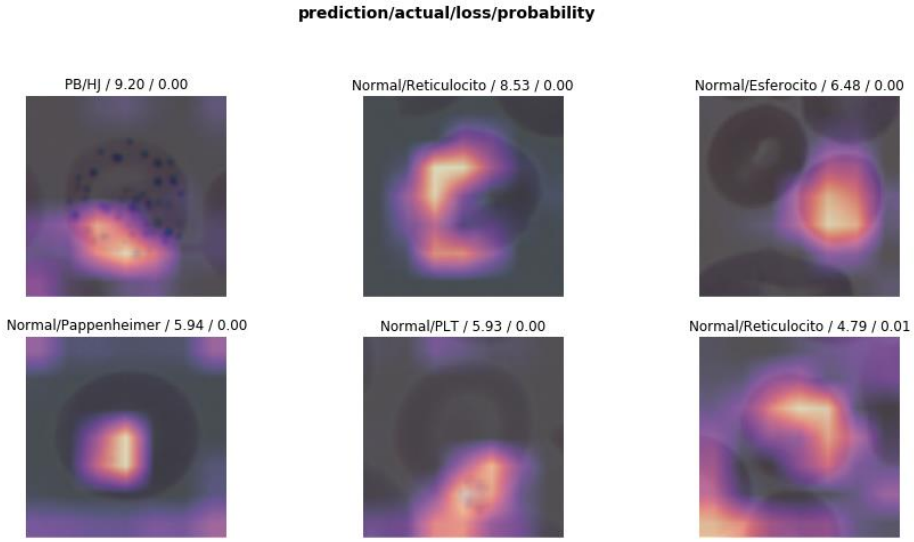

Figura 6.40 Imágenes con mayores pérdidas VGG19 Fine Tuning

Observando las matrices de confusión se consiguen unos muy buenos resultados en ambos modelos, aunque vuelven a fallar en la clasificación de los reticulocitos, aunque mejoran el resultado obtenido con la otra técnica.

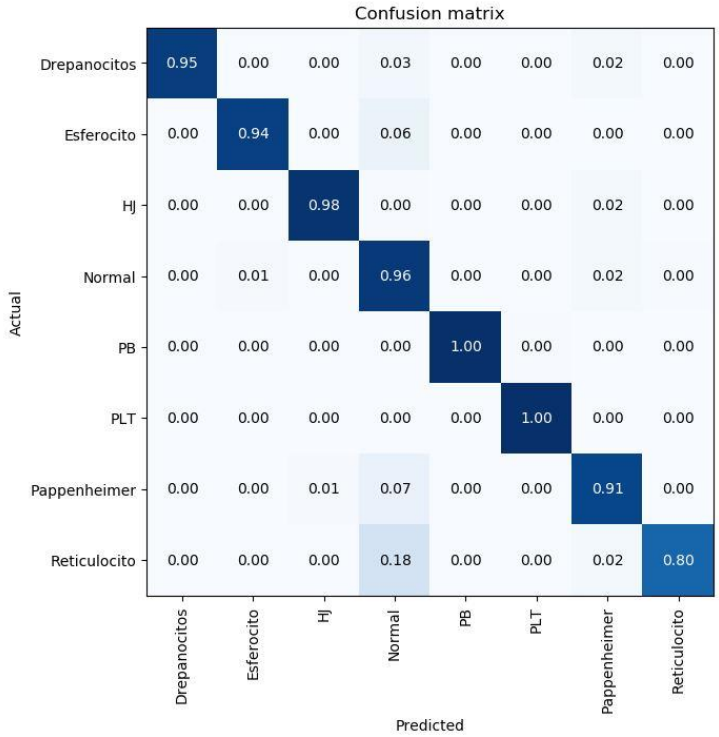

Figura 6.41 Matriz de confusión VGG16 Fine Tuning

Figura 6.42 Matriz de confusión VGG19 Fine Tuning

Con las clases más confundidas se llega a la misma conclusión de antes con respecto a la relación que causa más imágenes mal predichas (Pappenheimer-normal o normal-Pappenheimer) y con la cantidad de reticulocitos mal predichos, que no entra ni en las principales cinco.

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	34
Pappenheimer	Normal	25
Normal	Esferocito	20
Reticulocito	Normal	17
Esferocito	Normal	13

Tabla 21. Clases más confundidas VGG16 Fine Tuning

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	33
Normal	Esferocito	18
Esferocito	Normal	17
Reticulocito	Normal	13
Pappenheimer	Normal	11

Tabla 22. Clases más confundidas VGG19 Fine Tuning

6.2.4. DenseNet

Los últimos dos modelos a analizar son los correspondientes a las DenseNets.

DenseNet 121				DenseNet 169			
Epoch	**Train loss**	**Valid loss**	**Accuracy**	**Epoch**	**Train loss**	**Valid loss**	**Accuracy**
0	0,068984	0,077035	0,971183	0	0,059138	0,089318	0,967198
1	0,067018	0,081138	0,967198	1	0,049167	0,092039	0,965359
2	0,066245	0,080666	0,970264	2	0,046781	0,093081	0,965665
3	0,065192	0,081706	0,969037	3	0,050137	0,093507	0,966891
4	0,061842	0,081093	0,969957	4	0,044844	0,094095	0,965665

Tabla 23. Entrenamiento DenseNet Fine Tuning

En este caso se observa como las pérdidas son pequeñas ya desde el primer epoch en ambos modelos y además con una muy buena precisión. De hecho, a medida que avanza el modelo, las pérdidas de entrenamiento bajan mucho, sobre todo en el caso de DenseNet 169, pero la precisión y las pérdidas de validación se mantienen constantes o incluso empeoran. Esto puede significar que se está cayendo en el sobreajuste, ya sea por exceso de epochs o por un learning rate demasiado alto.

En las gráficas se puede observar lo comentado como a medida que avanza el entrenamiento las pérdidas se separan. En el caso de la DenseNet 169 se ve claramente que desde el principio la separación es muy grande.

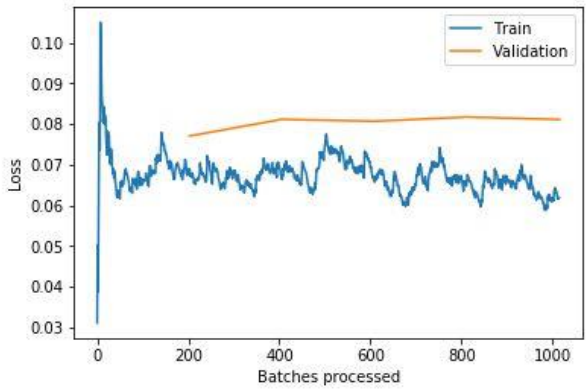

Figura 6.43 Evolución pérdidas DenseNet121 Fine Tuning

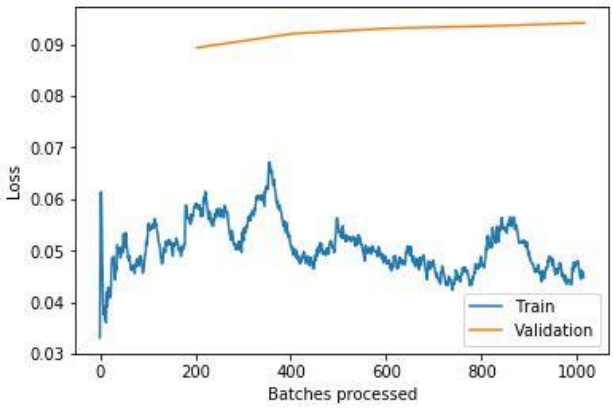

Figura 6.44 Evolución pérdidas DenseNet 169 Fine Tuning

En las imágenes con mayores pérdidas se vuelven a observar varias en las que la CNN se ha centrado en los alrededores de la célula de interés, o simplemente porque la célula de interés está movida.

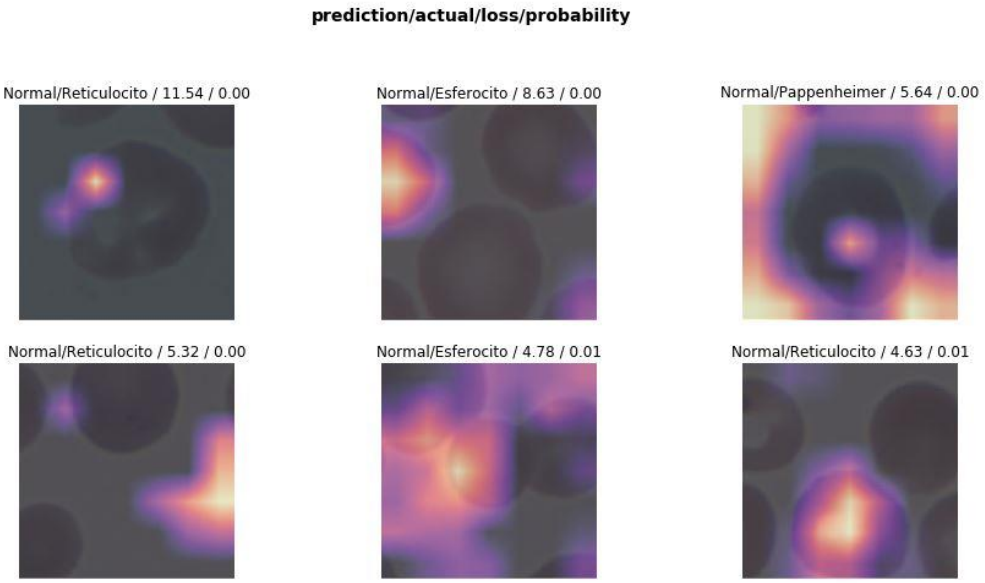

Figura 6.45 Imágenes con mayores pérdidas DenseNet121 Fine Tuning

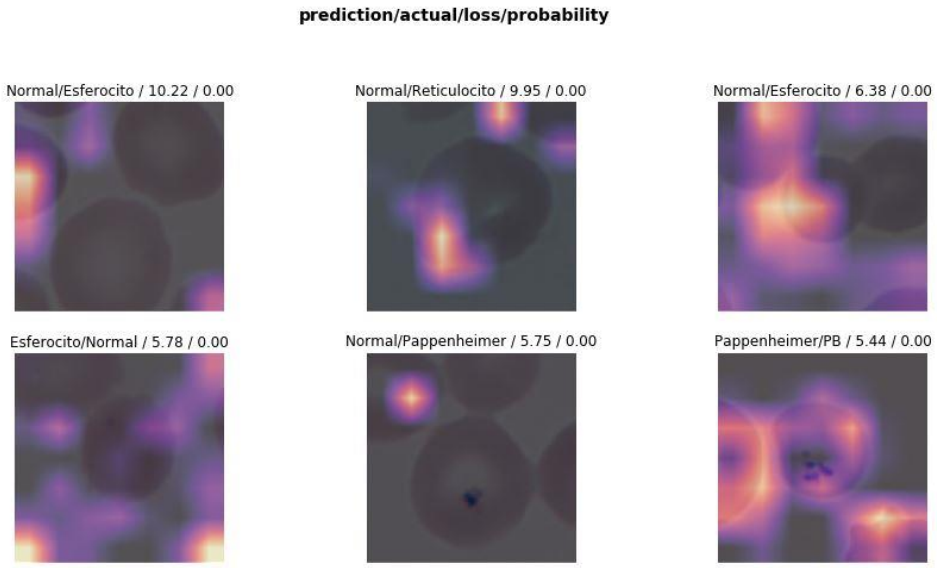

Figura 6.46 Imágenes con mayores pérdidas DenseNet169 Fine Tuning

Por último, en las matrices de confusión se observa un buen desempeño por parte de los dos modelos, llegando incluso al 100 % en alguna clase. Sin embargo, vuelve a suceder el problema con los reticulocitos.

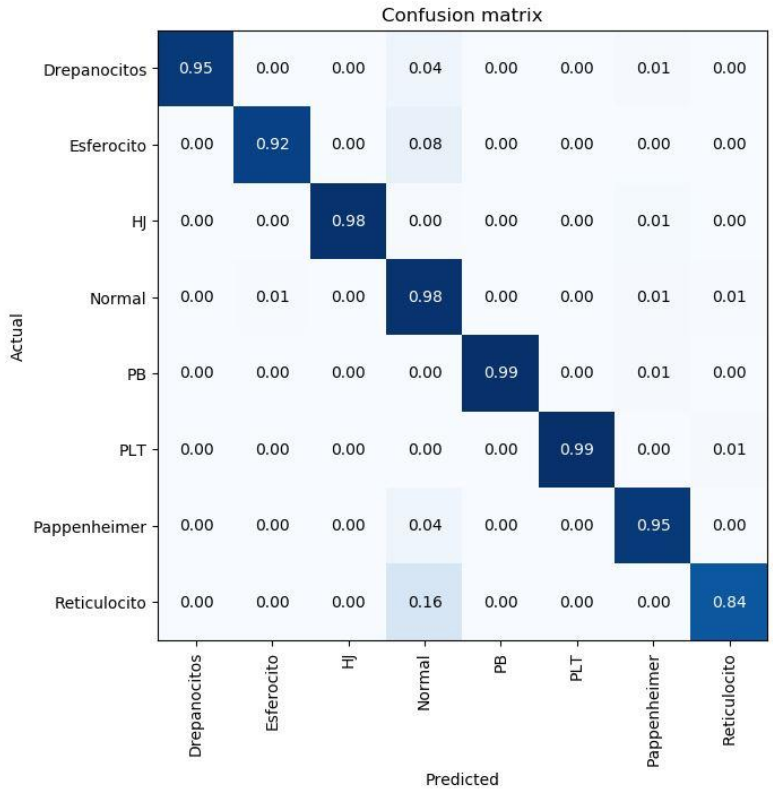

Figura 6.47 Matriz de confusión DenseNet121 Fine Tuning

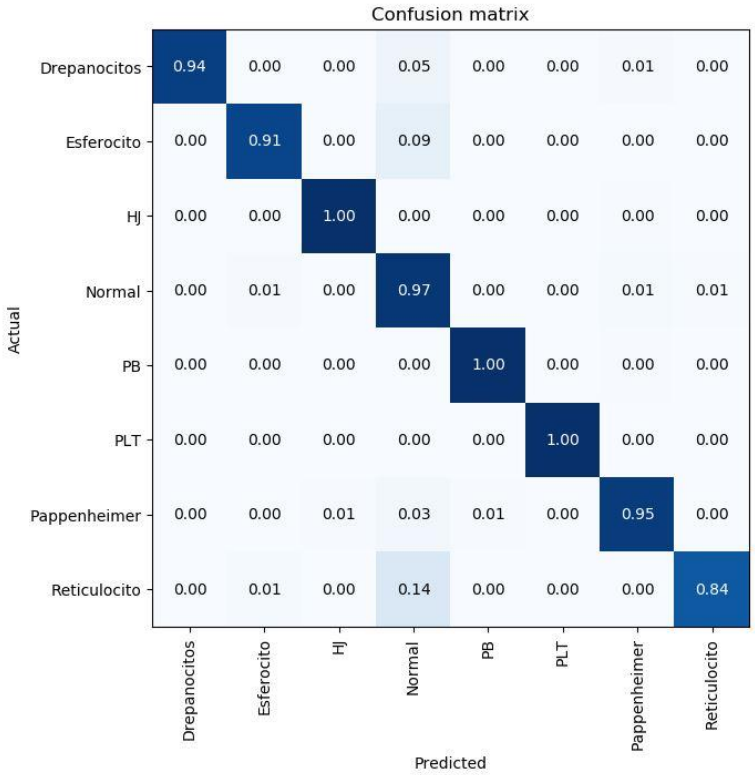

Figura 6.48 Matriz de confusión DenseNet169 Fine Tuning

En estos modelos, las clases que han originado mayor número de errores son las normales clasificadas como esferocitos y las de Pappenheimer clasificadas como normales.

Clase predicha	Clase correcta	Cantidad de errores
Esferocito	Normal	16
Normal	Pappenheimer	16
Pappenheimer	Normal	14
Normal	Reticulocito	12
Reticulocito	Normal	12

Tabla 24 Clases más confundidas DenseNet121 Fine Tuning

Clase predicha	Clase correcta	Cantidad de errores
Normal	Pappenheimer	27
Esferocito	Normal	18
Normal	Esferocito	16
Normal	Reticulocito	12
Pappenheimer	Normal	11

Tabla 25 Clases más confundidas DenseNet169 Fine Tuning

6.3. Resultados

A continuación, se presenta una tabla con todos los resultados obtenidos para cada modelo pre entrenado estudiado en los dos métodos de Transfer Learning. Los tres mejores resultados se destacan en verde, de más oscuro a más claro.

Modelo	Train_loss	Valid_loss	Precisión	Nº Epochs
FEATURE EXTRACTION				
ResNet 18	0.120656	0.120347	**0.958308**	**20**
ResNet 34	0.100800	0.104066	**0.962293**	**20**
ResNet 50	0.081830	0.094166	**0.965359**	**20**
AlexNet	0.234188	0.192371	**0.932557**	**20**
VGG 16	0.123814	0.112013	**0.956468**	**20**
VGG 19	0.116158	0.123272	**0.954016**	**20**
DenseNet 121	0.066757	0.078730	**0.971183**	**20**
DenseNet 169	0.044355	0.091320	**0.968118**	**20**

FINE TUNING				
ResNet 18	0.103275	0.118065	**0.959841**	**5**
ResNet 34	0.087711	0.093760	**0.963213**	**5**
ResNet 50	0.071295	0.090115	**0.968118**	**5**
AlexNet	0.161028	0.141842	**0.951257**	**5**
VGG 16	0.099929	0.104806	**0.956162**	**5**
VGG 19	0.086993	0.104284	**0.963213**	**5**
DenseNet 121	0.061842	0.081093	**0.969957**	**5**
DenseNet 169	0.044844	0.094095	**0.965665**	**5**

Tabla 26. Resumen resultados de los modelos

Se puede ver que, con el primer método, es decir, usando las CNN como extractor fijo de características, los modelos que consiguen una mejor precisión son DenseNet 121, DenseNet 169 y ResNet 50, en ese orden.

En el segundo método en cambio, el que sigue obteniendo mejor resultado es DenseNet 121, pero le sigue ResNet 50 y en tercera posición DenseNet 169. Que DenseNet 169 hay obtenido un peor resultado seguramente se deba a que se ha producido *overfitting,* como se ha podido observar con anterioridad en las gráficas.

Teniendo en cuenta los resultados obtenidos, el mejor de todos ellos es mediante la técnica de usar la CNN pre entrenada como extractor fijo de características, sin embargo, las diferencias son mínimas y en ambas técnicas se obtiene un muy buen resultado.

Al producirse el mejor resultado con esta primera técnica, con los parámetros congelados de la red pre entrenada, lleva a pensar que el conjunto de datos con la que se entrenó (ImageNet (42)) es eficaz para la clasificación de células sanguíneas, esto puede deberse a la gran cantidad de formas, texturas y colores que presenta este conjunto de datos.

Para averiguar si se consigue un mejor desempeño se podría llevar a cabo varias opciones, pudiendo implementarlas todas. La más obvia de todas es el aumento del conjunto de datos, ya que el tamaño del actual es pequeño. Otra opción sería balancear el conjunto de datos, esto se puede llevar a cabo

con *oversampling* en las clases más reducidas, y undersampling de las más numerosas, consiguiendo cierto equilibrio.

Una mejor calidad de imagen, así como una mejor separación de las células en ellas, también podría ayudar a mejorar, ya que hay imágenes en las que la calidad no es del todo buena o hay otras con células solapadas o cortadas.

También una mejor elección de learning rate o de la cantidad de epochs a la hora de entrenar.

Otra manera de mejorar los resultados, pero sin necesitar más datos, podría ser hacer *fine tuning* con el modelo que se ha obtenido, pero cambiando el tamaño de las imágenes, en lugar de 224x224 se podría usar el tamaño original. Esto es interesante hacerlo cuando casi se ha llegado al sobre ajuste porque es como si fuese un conjunto de datos nuevo para la ConvNet [(3) lección 3].

6.4. Aplicación web

Una vez creados y entrenados todos los modelos, se les puede poner en producción, es decir, a funcionar para cumplir el objetivo por el que fueron creados. Este objetivo era el de clasificar imágenes de células sanguíneas, no vistas con anterioridad, de manera correcta en alguna de las ocho clases distintas de células que se han estudiado.

Para probar los modelos se ha creado una aplicación web o *web app* donde está cargado el modelo elegido y se puede introducir una imagen nueva de alguna clase de las estudiadas y se clasifica según a cuál pertenezca.

El modelo elegido ha sido ResNet 50. Aunque haya modelos con mejores resultados (DenseNet 121), tiene una precisión muy buena, se puede observar su desempeño en la matriz de confusión, y además es menos costoso computacionalmente.

Para crearla se ha utilizado *Render* (56) y se ha seguido un sencillo tutorial de la página de fast.ai (57).

Se puede acceder a la aplicación web mediante el siguiente enlace:

https://clasificador-de-anomalias-eitrocitarias.onrender.com/

Se han llevado algunas pruebas con imágenes celulares eritrocitarias extraídas de internet que se pueden observar en las siguientes imágenes. En la primera prueba clasifica correctamente la imagen de un drepanocito y en la segunda hace lo mismo con una de cuerpos de Howell-Jolly.

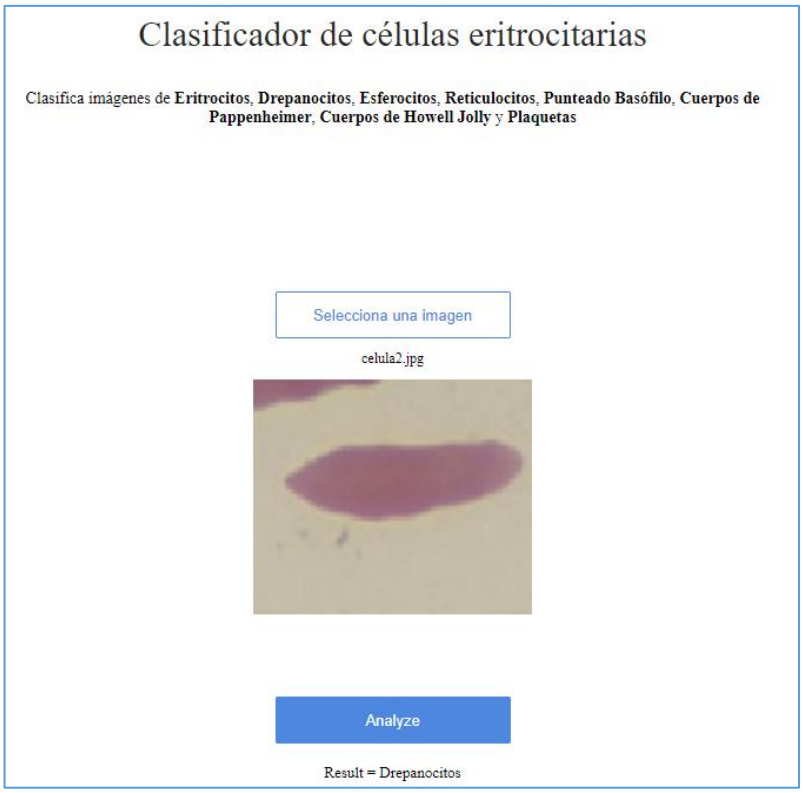

Figura 6.49 Primera prueba WebApp. Fuente imagen: (58)

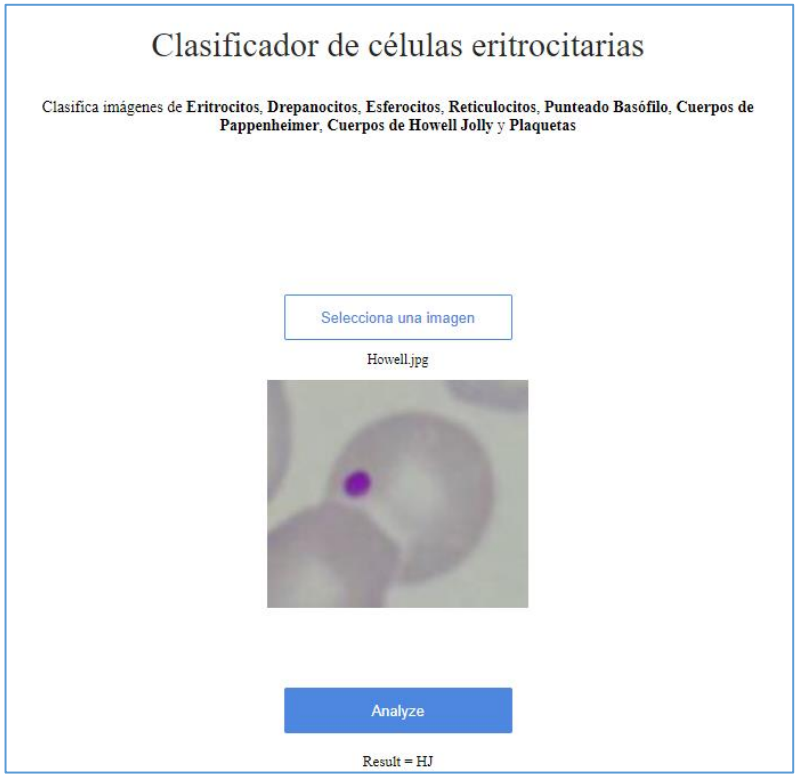

Figura 6.50 Segunda prueba WebApp. Fuente imagen: (59)

7. Hardware y software

El trabajo se ha llevado a cabo con un ordenador portátil HP Envy 15-k201ns:

- Procesador Intel ® Core ™ i7-5500U CPU @ 2.40 GHz
- RAM 8,00 GB
- Tarjeta gráfica GeForce GTX 850M
- Sistema operativo Windows 10 Home de 64 bits

La creación de los modelos se ha llevado a cabo con una conexión remota al servidor de la UPC. Para ello se ha requerido la instalación de Pulse Secure (60).

El software que se ha utilizado para llevar a cabo el trabajo es el siguiente:

- Python (61)
- Jupyter Notebook (62)
- Librería PyTorch (63)
- Librería fast.ai (3)
- Pulse Secure (60)

También se han utilizado varios servicios web:

- Render (56)
- GitHub (64)
- Google Drive (65)

8. Análisis del impacto ambiental

En este capítulo se evaluará el impacto que puede producir el proyecto y las consecuencias que puede producir tanto su correcto funcionamiento como su fallo.

La creación de un modelo que pueda clasificar correctamente las diferentes formas celulares, en este caso como ya se ha comentado, plaquetas, eritrocitos y anomalías morfológicas de estos últimos, puede producir varios impactos.

En caso de que se utilice como una herramienta de apoyo para el experto, su implementación puede reducir su tiempo de dedicación, ya que el modelo haría la mayor parte del trabajo. En este caso, al reducir las horas de dedicación también se reduce el gasto económico que conlleva.

También sería posible utilizarlo como herramienta para automatizar el diagnóstico, con lo que también se reduce la implicación de expertos en la materia.

Por tanto, su correcto funcionamiento sería un gran avance ya que agiliza el diagnóstico al paciente, pudiendo llevar a un diagnóstico con mayor antelación. Este ahorro de tiempo en el diagnóstico de según qué enfermedades puede ser realmente importante.

Por otro lado, en caso de utilizarlo como herramienta para automatizar el diagnóstico, si se produce un error y falla en el diagnóstico podrían suceder dos cosas, o no existe anomalía y se clasifica como si la hubiese, o en el peor caso, existe una anomalía y no la detecta y clasifica correctamente.

Por este último motivo, debido a que aún no se ha llegado a un método totalmente preciso, sería más conveniente usarlo como herramienta de apoyo para el experto responsable de la tarea.

Conclusiones

El uso del *deep learning* en el campo de la medicina supone un potencial muy grande, desde la investigación hasta el diagnóstico de enfermedades. Gracias a su aplicación se pueden acelerar tratamientos y conseguir diagnósticos mejores y más rápidos.

Hay que destacar que posiblemente en un futuro puedan llegar a sustituir a especialistas, pero hoy en día es aconsejable su uso como herramienta de soporte para ayudarlos y agilizar procesos, ya que como se ha visto, sus predicciones, aunque buenas, no son perfectas.

Por otra parte, el curso de Fast.ai, "Practical Deep Learning for Coders", es una excelente manera de introducirse al mundo del *deep learning* desde cero.

El objetivo del presente trabajo era crear un modelo capaz de clasificar automáticamente y con un gran desempeño células sanguíneas pertenecientes a ocho clases diferentes (plaquetas, eritrocitos, esferocitos, drepanocitos, reticulocitos, cuerpos de Pappenheimer, cuerpos Howell-Jolly y punteado basófilo).

Para ello se debió llevar a cabo el estudio de los fundamentos del *deep learning*, específicamente las redes neuronales convolucionales, que demuestran tener un gran desempeño en la clasificación automática de imágenes. También se requirió del estudio de dos métodos diferentes del denominado *transfer learning, feature extraction*, y *fine tuning.*

Gracias a esto fue posible la utilización de las diferentes arquitecturas pre entrenadas como extractor fijo de características, cambiando y entrenando solo las últimas capas encargadas de la clasificación, manteniendo el resto congelado. Con ello se obtuvo excelentes resultados como es el caso de ambas DenseNet y la ResNet 50, rozando e incluso superando el 97 % de exactitud.

También la utilización del *fine tuning,* descongelando los parámetros del modelo pre entrenado y ajustándolos al nuevo conjunto de datos, consiguiendo también resultados muy buenos, también en las tres anteriores.

Comparando los resultados se observa que el mejor es el obtenido con la DenseNet 121 utilizando *feature extraction.* Esto refleja que el conjunto de datos con el que se entrenó en un principio esta arquitectura, ImageNet, funciona muy bien para la clasificación de células sanguíneas. Esto puede deberse a la gran cantidad de formas y texturas que tiene.

Teniendo en cuenta que el conjunto de datos era de un tamaño reducido (la mayoría de clases tienen cerca de las mil imágenes o menos) y estando bastante desbalanceado, son unos resultados bastante

satisfactorios. De esto se deduce que, aunque aconsejable en algunos casos, el balanceo de datos no es incompatible con obtener buenos resultados.

Difícilmente se podría haber llegado a una precisión así creando una ConvNet desde cero, se llevaron a cabo varias pruebas, pero sin ningún resultado lo suficientemente satisfactorio.

Además, se ha conseguido la implementación del modelo en una sencilla aplicación web para poder llevar a cabo su utilización como clasificador. Su desempeño ha resultado ser bastante bueno, clasificando correctamente las diversas pruebas que se han realizado.

Análisis económico

A continuación, se detallan los costes asociados a la realización de este proyecto. Los únicos costes asociados a este proyecto son los relacionados al servicio de la aplicación web y al personal encargado del estudio y programación.

Servicios

Concepto	Coste (€/mes)	Tiempo (mes)	Total (€)
Render	5	3	15

Tabla 27. Presupuesto servicios

Personal

Personal	Coste (€/h)	Tiempo (h)	Total (€)
Ingeniero Junior	10	600	6000
Ingeniero Senior (Director proyecto)	30	60	1800
Ingeniero Senior (Co-director proyecto)	30	10	300

Tabla 28. Presupuesto mano de obra

Total

Tipo de coste	Coste (€)
Servicios	15
Personal	8100
TOTAL	8115

Tabla 29. Presupuesto total

Bibliografía

1. Chollet, F. *Deep Learning with Python*. Manning Shelter Island, 2018. ISBN 9781617294433.

2. CS231n Convolutional Neural Networks for Visual Recognition. En: [en línea]. [consulta: 25 octubre 2018]. Disponible en: http://cs231n.github.io/.

3. Fast.ai. Practical Deep Learning for Coders, v3 | fast.ai course v3. En: [en línea]. 2018. [consulta: 2 diciembre 2018]. Disponible en: course.fast.ai/.

4. Krizhevsky, A. y Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. En: [en línea]. p. 1-9. Disponible en: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

5. Simonyan, K. y Zisserman, A. Very deep convolutional networks for large-scale image recognition. En: [en línea]. 2015. [consulta: 1 febrero 2019]. Disponible en: http://www.robots.ox.ac.uk/.

6. He, K. et al. Deep Residual Learning for Image Recognition. En: [en línea]. 2015, Disponible en: http://arxiv.org/abs/1512.03385.

7. Huang, G. et al. Densely connected convolutional networks. En: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* [en línea]. 2017, vol. 2017-Janua, p. 2261-2269. DOI 10.1109/CVPR.2017.243. Disponible en: https://arxiv.org/pdf/1608.06993.pdf.

8. Hall, J.E. *Guyton y Hall, Tratado de Fisiología Médica*. Decimoterc. ELSEVIER, 2016. ISBN 978-84-9113-024-6.

9. Fox, S.I. *Fisiología Humana*. Decimoterc. Mc Graw Hill Education, 2014. ISBN 978-607-15-1151-5.

10. Tresguerres, J.A.F., Villanúa Bernués, M.Á. y López-Calderón Barreda, A. *Anatomía y fisiología del cuerpo humano*. Mc Graw Hill, 2009. ISBN 978-84-481-6890-2.

11. Genomasur. Sistema circulatorio. En: [en línea]. [consulta: 15 marzo 2019]. Disponible en: http://www.genomasur.com/BCH/BCH_libro/capitulo_13.htm.

12. Tortora, G.J. y Derrickson, B. *Principios de Anatomía y Fisiología*. Decimoterc. Editorial Médica Panamericana, 2011. ISBN 978-968-7988-77-1.

13. Colegio oficial de enfermeros de Barcelona. Sistema inmune y la sangre. En: [en línea]. [consulta: 16 marzo 2019]. Disponible en: https://www.infermeravirtual.com/files/media/file/102/Sangre.pdf?1358605574.

14. Componentes de la sangre (artículo) | Khan Academy. En: [en línea]. [consulta: 22 marzo 2019]. Disponible en: https://es.khanacademy.org/science/biology/human-biology/circulatory-pulmonary/a/components-of-the-blood.

15. Eritrocito | Biblioteca de Hematología. En: [en línea]. [consulta: 28 febrero 2019]. Disponible en: https://bibliotecadehematologia.wordpress.com/2014/10/05/eritrocito/.

16. Fisiologia de la Eritropoyesis | Sintesis de los Globulos Rojos. En: [en línea]. [consulta: 18 abril 2019]. Disponible en: https://cerebromedico.com/fisiologia-de-la-eritropoyesis/.

17. Laboratorio de Hematología: Recuento de Plaquetas. En: [en línea]. [consulta: 28 marzo 2019]. Disponible en: http://laboratoriohematologiadefelipe.blogspot.com/2015/08/recuento-de-plaquetas.html.

18. Merino, A. Alteraciones morfológicas de los eritrocitos. En: *Educacion continuada en el laboratorio clinico* [en línea]. 2015, vol. 20, p. 41-64. Disponible en: http://www.seqc.es/download/tema/3/2767/346271904/2987076/cms/tema-5-alteraciones-morfologicas-de-los-eritrocitos.pdf/.

19. D'Onofrio, G., Zini, G. y J Bain, B. *Morphology of Blood Disorders*. Second Edi. WILEY Blackwell, 2015.

20. Esferocitosis hereditaria. En: [en línea]. [consulta: 28 marzo 2019]. Disponible en: http://atlas.gechem.org/es/component/k2/item/229-esferocitosis-hereditaria.

21. Jaime Pérez, J.J. y Gómez Almaguer, D. *Hematología, La sangre y sus enfermedades*. Tercera ed. Mc Graw Hill, 2012. ISBN 978-607-15-0748-8.

22. Drepanocitosis - Hematología y oncología - Manual MSD versión para profesionales. En: [en línea]. [consulta: 28 marzo 2019]. Disponible en: https://www.msdmanuals.com/es/professional/hematología-y-oncología/anemias-causadas-por-hemólisis/drepanocitosis.

23. Hematología – Laboratorios Clínicos Zaragoza. En: [en línea]. [consulta: 28 marzo 2019]. Disponible en: http://centrodeespecialidadesdehuajuapan.com/laboratoriosclinicoszaragoza/perfilsapoydx/estudios-lab/hematologia/.

24. J. Bain, B. *Blood cells, A Practical Guide*. Fifth Edit. WILEY Blackwell, 2015.

25. Margaret Rouse. ¿Qué es Aprendizaje profundo (deep learning)? En: [en línea]. [consulta: 21 marzo 2019]. Disponible en: https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-profundo-deep-learning.

26. CS231n Convolutional Neural Networks for Visual Recognition. En: [en línea]. [consulta: 25 febrero 2019]. Disponible en: https://cs231n.github.io/neural-networks-1/.

27. Karn, U. A Quick Introduction to Neural Networks – the data science blog. En: [en línea]. [consulta: 20 marzo 2019]. Disponible en: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.

28. Hafidz Zulkifli. Understanding Learning Rates and How It Improves Performance in Deep Learning. En: [en línea]. [consulta: 1 abril 2019]. Disponible en: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

29. Smith, L.N. A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay. En: [en línea]. 2018, p. 1-21. Disponible en: http://arxiv.org/abs/1803.09820.

30. Smith, L.N. Cyclical learning rates for training neural networks. En: *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*. 2017, no. April, p. 464-472. DOI 10.1109/WACV.2017.58.

31. Smith, L.N. y Topin, N. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. En: [en línea]. [consulta: 2 marzo 2019]. Disponible en: https://arxiv.org/pdf/1708.07120.pdf.

32. Sylvian Gugger. Another data science student's blog – The 1cycle policy. En: [en línea]. [consulta: 2 marzo 2019]. Disponible en: https://sgugger.github.io/the-1cycle-policy.html.

33. Understanding Convolutional Neural Networks for NLP – WildML. En: [en línea]. [consulta: 20 marzo 2019]. Disponible en: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/.

34. Karn, U. An Intuitive Explanation of Convolutional Neural Networks. En: [en línea]. [consulta: 25 febrero 2019]. Disponible en: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

35. Jordan, M.I., Bishop, C.M. y Fergus, R. Neural networks. En: [en línea]. 2015, p. 32-50. Disponible en: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf.

36. CS231n Convolutional Neural Networks for Visual Recognition. Convolutional Neural Networks. En: [en línea]. [consulta: 16 marzo 2019]. Disponible en: https://cs231n.github.io/convolutional-networks/#fc.

37. Patrick Kamencay, Miroslav Benco, Tomas Mizdos, R.R. Max Pooling operation on feature map. En: [en línea]. [consulta: 20 marzo 2019]. Disponible en: https://www.researchgate.net/figure/Max-Pooling-operation-on-feature-map-22-window_fig8_321286547.

38. Leonardo Araujo Santos. Pooling Layer · Artificial Inteligence. En: [en línea]. [consulta: 3 abril 2019]. Disponible en: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/pooling_layer.html.

39. Sumit Saha. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. En: [en línea]. [consulta: 9 marzo 2019]. Disponible en: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

40. Zeiler, M.D. y Fergus, R. Visualizing and understanding convolutional networks. En: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014, vol. 8689 LNCS, no. PART 1, p. 818-833. ISSN 16113349. DOI 10.1007/978-3-319-10590-1_53.

41. Himanshu Rawlani. Visual Interpretability for Convolutional Neural Networks. En: [en línea]. [consulta: 30 marzo 2019]. Disponible en: https://towardsdatascience.com/visual-interpretability-for-convolutional-neural-networks-2453856210ce.

42. Stanford University, P.U. ImageNet. En: [en línea]. [consulta: 22 marzo 2019]. Disponible en: http://www.image-net.org/.

43. CS231n Convolutional Neural Networks for Visual Recognition. Transfer Learning. En: [en línea]. [consulta: 22 marzo 2019]. Disponible en: https://cs231n.github.io/transfer-learning/.

44. Stanford. Stanford DAWN Deep Learning Benchmark (DAWNBench) ·. En: [en línea]. [consulta: 16 abril 2019]. Disponible en: https://dawn.cs.stanford.edu/benchmark/.

45. CV-Tricks. ResNet, AlexNet, VGGNet, Inception: Understanding various architectures of Convolutional Networks – CV-Tricks.com. En: [en línea]. [consulta: 29 marzo 2019]. Disponible en: https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/.

46. Sik-Ho Tsang. Review: DenseNet — Dense Convolutional Network (Image Classification). En: [en línea]. [consulta: 7 abril 2019]. Disponible en: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803.

47. Pablo Ruiz Ruiz. Understanding and visualizing ResNets – Towards Data Science. En: [en línea]. 2018. [consulta: 6 marzo 2019]. Disponible en: https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8.

48. Ruiz, P. ResNets. En: [en línea]. 2018, no. August. Disponible en: http://pabloruizruiz10.com/resources/CNNs/ResNets.pdf.

49. Faizan Shaikh. 10 Advanced Deep Learning Architectures Data Scientists Should Know. En: [en línea]. [consulta: 3 abril 2019]. Disponible en: https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/.

50. Muneeb ul Hassan. VGG16 - Convolutional Network for Classification and Detection. En: [en línea]. [consulta: 10 abril 2019]. Disponible en: https://neurohive.io/en/popular-networks/vgg16/.

51. Sik-Ho Tsang. Review: VGGNet — 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014. En: [en línea]. [consulta: 29 marzo 2019]. Disponible en: https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11.

52. Buda, M., Maki, A. y Mazurowski, M.A. A systematic study of the class imbalance problem in convolutional neural networks. En: *Neural Networks* [en línea]. 2018, vol. 106, p. 249-259. ISSN 18792782. DOI 10.1016/j.neunet.2018.07.011. Disponible en: https://arxiv.org/abs/1710.05381.

53. Anup Bhande. What is underfitting and overfitting in machine learning and how to deal with it. En: [en línea]. [consulta: 29 marzo 2019]. Disponible en: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.

54. Justel Pizarro, J. Repositorio GitHub. En: [en línea]. 2019. Disponible en: https://github.com/JoaquinJustel/TFG.

55. Nachiket Tanksale. Finding Good Learning Rate and The One Cycle Policy. En: [en línea]. [consulta: 5 abril 2019]. Disponible en: https://towardsdatascience.com/finding-good-learning-rate-and-the-one-cycle-policy-7159fe1db5d6.

56. Render. Render · The Easiest Cloud For All Your Apps. En: [en línea]. Disponible en: https://dashboard.render.com/.

57. Fast.ai. Deploying on Render | fast.ai course v3. En: [en línea]. [consulta: 10 abril 2019]. Disponible en: https://course.fast.ai/deployment_render.html#per-project-setup.

58. Fichero de hematología: anormalidades eritrocitarias. Anormalidades la forma del eritrocito. Drepanocito. En: [en línea]. [consulta: 29 marzo 2019]. Disponible en: http://hemasextociclousac.blogspot.com/2014/10/anormalidades-eritrocitarias_5.html.

59. Grupo Español de Citología Hematológica. Cuerpos de Howell Jolly. En: [en línea]. [consulta: 30 marzo 2019]. Disponible en: http://atlas.gechem.org/es/component/k2/item/12-cuerpos-de-howell-jolly.

60. Pulse Secure. En: [en línea]. [consulta: 1 marzo 2019]. Disponible en: https://www.pulsesecure.net/.

61. Python. En: [en línea]. Disponible en: https://www.python.org/.

62. Project Jupyter | Home. En: [en línea]. Disponible en: https://jupyter.org/.

63. PyTorch. En: [en línea]. Disponible en: https://pytorch.org/.

64. GitHub. En: [en línea]. Disponible en: https://github.com/.

65. Google Drive. En: [en línea]. Disponible en: https://drive.google.com.