Exploración de Vision Transformer para la clasificación de células normales de sangre periférica

Belén Ryba Maciejewska
Máster universitario en Bioinformática y Bioestadística
Área de Machine Learning

Nombre Director
Edwin Santiago Alférez Baquero
Fecha Entrega: 06/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Descripción del trabajo*
Nombre del autor:	*Belén Ryba Maciejewska*
Nombre del consultor/a:	*Edwin Santiago Alférez Baquero*
Nombre del PRA:	*Laura Calvet Liñán*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Máster Universitario en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Machine Learning*
Idioma del trabajo:	*Español*
Número de créditos:	15
Palabras clave	*Deep Learning, clasificación de imágenes, Visual Transformer*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Pese a que las Redes Neuronales Convolucionales hayan revolucionado el mundo de la visión artificial, ante el creciente aumento del tamaño de los conjuntos de datos y el continuo desarrollo de nuevas técnicas, estos modelos empiecen a presentar limitaciones, especialmente debido a su elevado tiempo de procesamiento.

En el siguiente trabajo se analiza *Vision Transformer*, una nueva arquitectura propuesta que permitiría superar estas restricciones. Centrando su aplicación en el campo de las ciencias de la computación en el ámbito sanitario para el reconocimiento automático de células en sangre periférica.

Los resultados obtenidos, con una precisión superior al 0.96 en la clasificación, sugieren que los modelos basados en ViT son una prometedora alternativa para el desempeño de este tipo de tareas frente a los ya conocidos modelos basados en CNNs. Asimismo, se implementa una sencilla interfaz gráfica para acercar a los usuarios a la utilización de este tipo de algoritmos para la clasificación, sin tener que disponer de conocimientos informáticos avanzados.

Abstract (in English, 250 words or less):

Although Convolutional Neuronal Networks have revolutionized the world of computer vision, with the increasing size of datasets and the continuous development of new techniques, these models are beginning to present limitations, especially due to their high processing time.

The following work analyzes Vision Transformer, a new proposed architecture that would allow overcoming these restrictions. Focusing its application in the field of computer science for the automatic recognition of cells in peripheral blood.

The results obtained, with a higher classification accuracy than 0.96, show that ViT-based models are a promising alternative to the well-known CNN-based models for the performance of this type of tasks. In addition, a simple graphical interface is implemented with the aim of bringing users closer to the use of this type of algorithms for classification tasks, without any deep computer knowledge.

ÍNDICE

Listado de figuras

Listado de tablas

1. INTRODUCCIÓN

1.1. CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

1.1.1. DESCRIPCIÓN GENERAL

El reconocimiento e inspección de las células sanguíneas de la sangre periférica constituye una de las tareas más importantes para el diagnóstico de múltiples enfermedades. Debido a la complejidad de este proceso, tanto por la cantidad de diferentes tipos celulares como de sus distintas morfologías, esta tarea se ha tenido que realizar en gran medida de forma manual por especialistas. Constituyendo un proceso tedioso y monótono al que los profesionales sanitarios deben enfrentarse a diario, invirtiendo grandes cantidades de tiempo, y sujeto a una variabilidad intra e inter observador.

Es por ello por lo que la comunidad científica se encuentra volcada en el desarrollo de algoritmos basados en Inteligencia Artificial (IA) capaces de automatizar estos procesos con una precisión igual o mayor a la obtenida por los facultativos sanitarios. Buscando obtener herramientas de diagnóstico mucho más eficientes y rápidas que los métodos tradicionales.

El desarrollo en los últimos años de diferentes arquitecturas, basadas en *Deep Learning*, especialmente en el caso redes neuronales convolucionales (CNNs) ha demostrado obtener muy buenos resultados en este tipo de cometidos. Esto, junto con la búsqueda de cada vez soluciones más eficientes ha centrado la atención en nuevas propuestas de modelos basados en arquitecturas de *Visual Transformer* (ViT), que permitirían obtener resultados muy similares a las CNNs pero con un menor tiempo de procesamiento.

1.1.2. JUSTIFICACIÓN DEL TFM

Pese a que las CNNs hayan revolucionado el mundo de la visión artificial, ante el creciente aumento del tamaño de los conjuntos de datos y el continuo desarrollo de nuevos métodos, tanto supervisados como no supervisados, estos modelos empiecen a presentar limitaciones.

Estas limitaciones respecto al tiempo de procesamiento del algoritmo despiertan un creciente interés por encontrar nuevas arquitecturas que nos permitan realizar entrenamientos más eficientes sobre conjuntos de datos cada vez mayores.

Una de estas nuevas propuestas se centra en las arquitecturas basadas en ViT (1). En este trabajo, concretamente, estudiaremos su aplicación para el reconocimiento automático de células en sangre periférica. Para, posteriormente, comparar su desempeño con los modelos basados en CNNs (2), con el fin de demostrar que estos nuevos modelos pueden ser el paso preliminar hacia arquitecturas genéricas y escalables que puedan resolver muchas tareas en el campo de la visión artificial.

Con la elección de un estudio comparativo entre esta nueva arquitectura y las ya conocidas CNNs podemos comprobar si estos nuevos modelos realmente consiguen abordar un mismo problema con una alta precisión y un menor tiempo de entrenamiento. A la vez que, analizamos si verdaderamente su arquitectura, basada en *self-atenttion* de parches segmentados a partir de las imágenes originales, supone alguna ventaja.

La importancia de estos resultados radica en que si estos modelos cumplen con las expectativas generadas permitirían procesar mayores cantidades de datos en un menor tiempo, suponiendo un consecuente ahorro de tiempo y de recursos económicos para numerosos proyectos que podrían avanzar de forma mucho más rápida.

1.2. OBJETIVOS DEL TRABAJO

El objetivo principal de este trabajo es el estudio y aplicación práctica de un modelo basado en *Vision Transformer* (ViT) en el reconocimiento automático de células normales de sangre periférica.

Para la consecución de este objetivo se pueden fijar una serie de objetivos específicos:

1. Familiarizarse con el reconocimiento de los diferentes tipos de células de sangre periférica en imágenes digitales.
2. Adquirir nuevos conocimientos de *Deep Learning* y reforzar otros. En particular, los relacionados con ViT.
3. Desarrollar y validar los modelos de detección automática basado en CNNs y ViT, a partir de las imágenes digitales de células de sangre periférica obtenidas en el paso anterior.
4. Comparación del desempeño y porcentaje de éxito del modelo frente a los algoritmos convencionales basados en CNNs.

1.3. ENFOQUE Y MÉTODO SEGUIDO

El objetivo final de este proyecto es desarrollar una herramienta para la detección de células normales de sangre periférica, a partir de un conjunto de imágenes de frotis de sangre periférica. En este caso, el elemento diferenciador es la utilización de *Vision Transformer* (ViT), arquitecturas basadas en *self-atenttion* de parches segmentados a partir de las imágenes originales que permiten obtener resultados similares a las convencionales CNNs de forma más eficiente, con un menor tiempo de procesamiento.

Podemos distinguir diferentes fases en el proyecto:

1.- Implementación de un modelo basado en CNNs para la detección de células sanguíneas de sangre periférica.
2.- Implementación de la arquitectura basada en ViT para la detección de células sanguíneas de sangre periférica.
3.- Entrenamiento de la redes y ajuste de parámetros.
4.- Evaluación de los resultados y comparación con los obtenidos por los diferentes modelos.

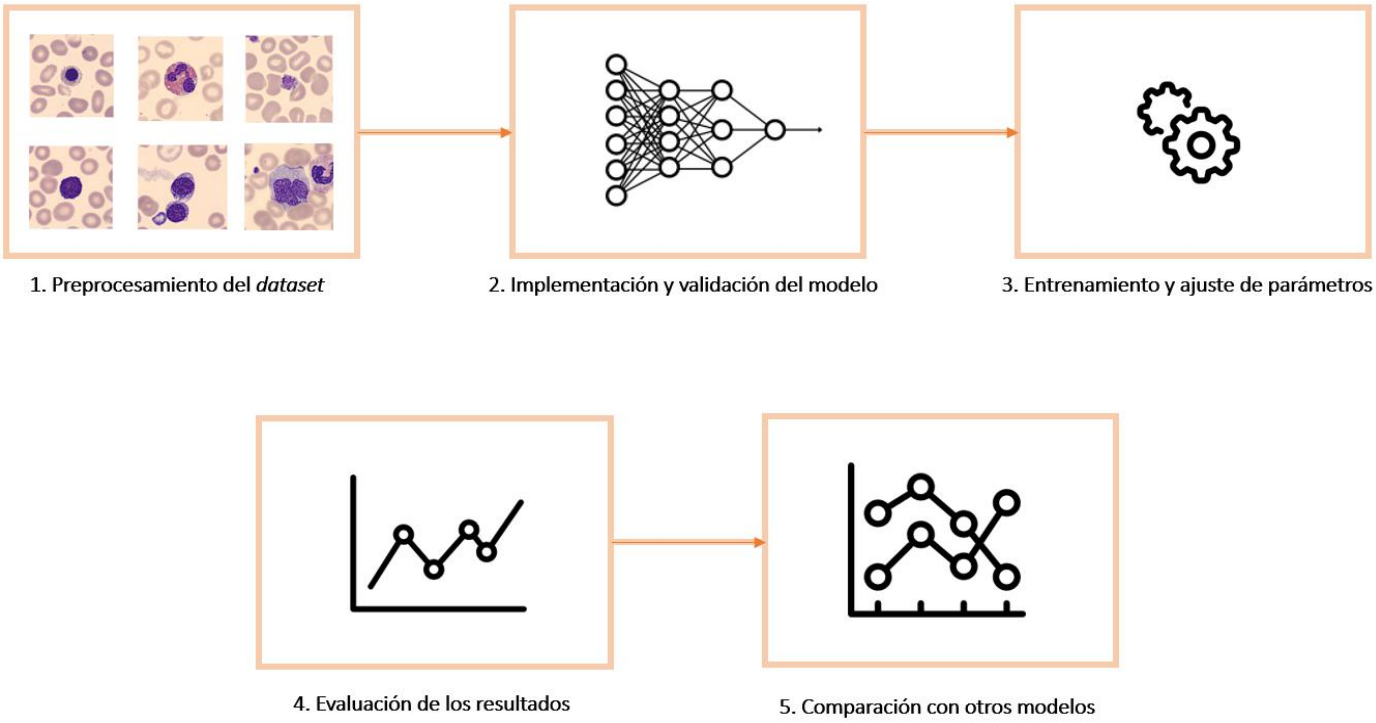

Figura 1. Esquema del procedimiento seguido.

1.4. PLANIFICACIÓN DEL TRABAJO

1.4.1. TAREAS

Podemos dividir las tareas en dos fases consecutivas, una primera de adquisición de los conocimientos necesarios e implementaciones básicas. Seguida de una segunda fase, centrada en el desarrollo, funcionamiento y perfeccionamiento de los modelos.

FASE 1

1. Instalación del entorno de desarrollo óptimo para trabajar con Python utilizando Google Colab, con las librerías correspondientes para trabajar con PyTorch y Fast.ai. (10 horas).

2. Formación en el *framework* utilizado para el desarrollo, Pytorch y Fast.ai, a través de cursos online. Junto con el entorno de desarrollo (Google Colab) (40 horas).

3. Formación en *Vision Transformer* por medio de la revisión y el estudio de la bibliografía relacionada con este tipo de modelos. (15 horas).

4. Análisis del estado de arte de las aplicaciones y modelos propuestos de CNNs y Vit. (10 horas).

FASE 2

1. Creación del *pipeline* automatizado para el tratamiento de las imágenes. (15 horas).
2. Implementación del modelo basado en CNNs en Fast.ai (20 horas).
3. Implementación modelo basado en ViT en Fast.ai (35 horas).
4. Entrenamiento y validación de los modelos. (50 horas).
5. Creación de un repositorio en GitHub para compartir el código desarrollado. (10 horas)
6. Redacción de la memoria (25 horas).

1.4.2. CALENDARIO

Para facilitar la organización del trabajo se desarrollan tres diagramas de *Grantt*, mostrados a continuación, en los que se muestra la planificación global, en función de las fechas de entrega correspondientes a las diferentes pruebas de evaluación continua (PEC). Junto con la planificación detallada de la PEC 2 y PEC 3, con sus respectivas tareas.

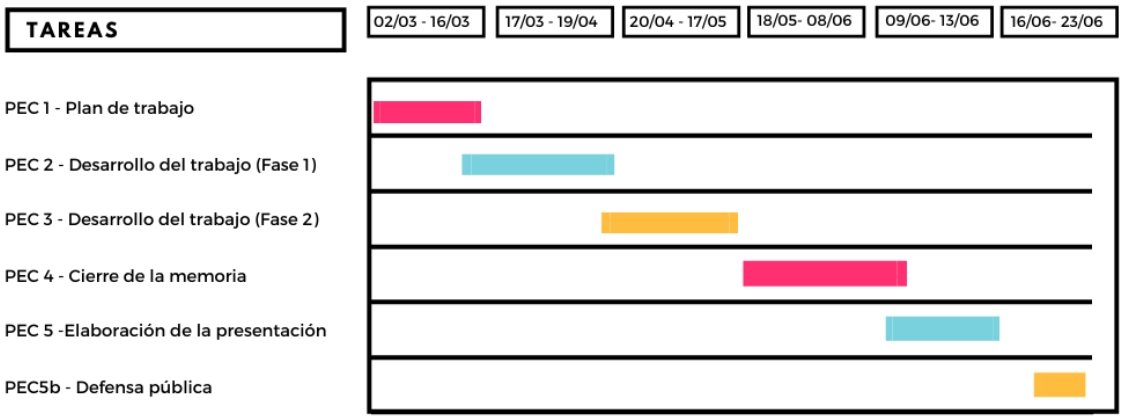

Figura 2. Planificación Global

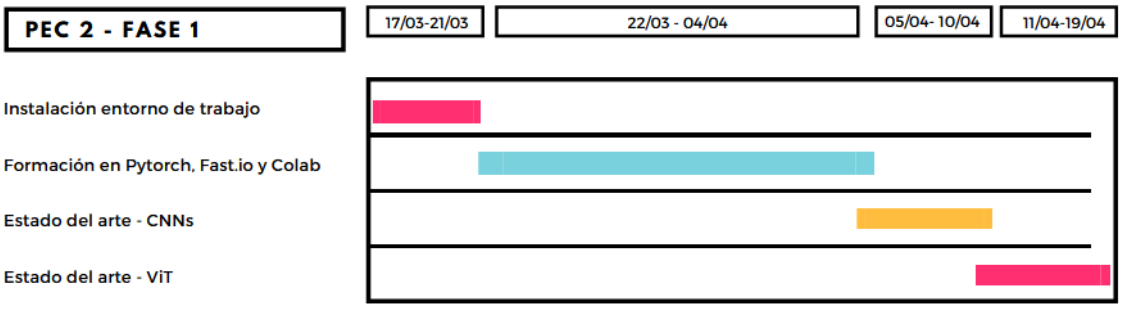

Figura 3. Planificación PEC 2.

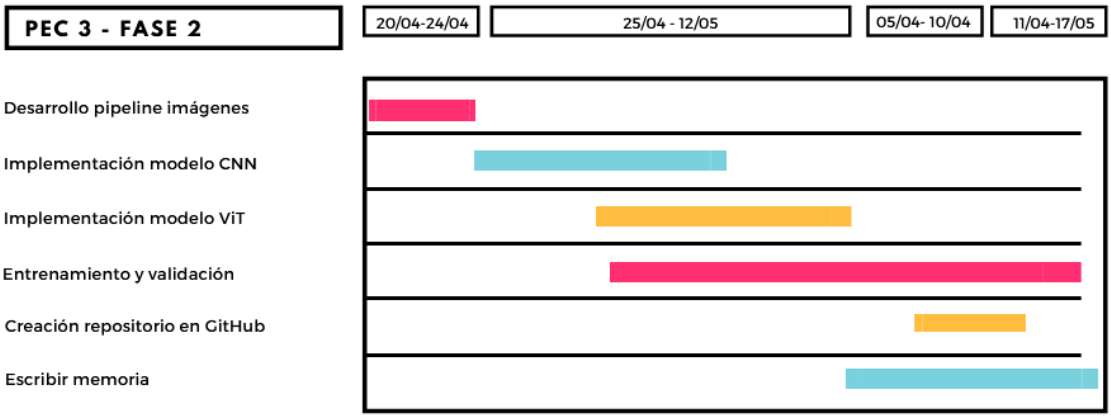

Figura 4. Planificación PEC 3.

1.4.3. HITOS

- Configuración del entorno de desarrollo.
- Creación del *pipeline* de preprocesamiento de las imágenes.
- Implementación y validación del modelo basado en CNNs.
- Implementación y validación del modelo basado en arquitectura ViT.
- Entrenamiento del modelo.
- Evaluación de los resultados obtenidos y comparación de sus resultados frente a modelos basados en CNNs.
- Redacción de la memoria.
- Elaboración de la defensa del trabajo.

1.4.4. ANÁLISIS DE RIESGO

Para evaluar los posibles riesgos asociados a este proyecto los dividimos en las siguientes categorías, según su naturaleza:

- **Problemas técnicos:** problemas relacionados con el equipo informático utilizado para el desarrollo del proyecto. Para evitar este tipo de imprevistos se realizan periódicamente *backups* en un disco duro externo y en Google Drive. De esta forma, en caso de ser necesario, se podría retomar el proyecto desde otro equipo sin pérdidas de información.
- **Riesgos de calendario:** ya sea por una mala planificación de los objetivos o por la imposibilidad de conciliar en algún momento puntual el horario laboral con el desarrollo del proyecto se procedería a replanificar el calendario en función de la nueva disponibilidad horaria con el fin de cumplir con todos los objetivos fijados para la fecha final del proyecto.
- **Enfermedad o accidente**: en caso de ser necesario se procedería a reorganizar los objetivos y de no serlo, se replanificarían las tareas restantes según el nuevo tiempo disponible, tratando de compensar las horas.

- Riesgos externos: situaciones de crisis, ya sea económica, política, sanitaria o medioambiental que impidan el desarrollo normal del proyecto según su planificación y los recursos necesarios.

1.4.5. COSTOS ASOCIADOS AL PROYECTO

Para la realización del presupuesto de este proyecto se tienen en cuenta tanto los gastos directos como los indirectos. Los gastos directos se obtienen como la suma del coste de la mano de obra junto con el coste de los recursos materiales empleados.

En este caso se estima una dedicación de 375 horas para el desarrollo del trabajo, en los que se empleará diferentes recursos. El ordenador personal (procesador i7, 16 GB de RAM y gráfica Nvidia MX250), por no disponer de una gráfica adecuada para el trabajo con *Deep Learning* se hará uso del servicio de *Google Colab* y *Google Drive* en su versión gratuita. No se requiere del pago de ninguna licencia ya que el software utilizado es de uso libre o ya se disponía de él.

Los gastos indirectos se calculan como un 15% de los gastos directos totales. Además, se tiene en cuenta el beneficio industrial, que se corresponde con aproximadamente el 6% de los gastos totales, es decir, los gastos directos junto a los indirectos. Obteniendo así, finalmente un presupuesto total de 8.435,10€.

COSTE MANO DE OBRA	Horas	Precio/hora	Total
Trabajador implicado	375	15 €	5.625 €

COSTE RECURSOS MATERIALES	Precio	Tiempo de uso (meses)	Amortización (años)	Total
Ordenador personal	900 €	5	4	93,75 €

Costes directos	5.718,75 €
Gastos generales (indirectos)	857,81 €
Beneficio industrial	394,59 €

SUBTOTAL PRESUPUESTO	6.971,15 €
IVA (21%)	1.463,94 €
PRESUPUESTO TOTAL	8.435,10 €

Tabla 1. Presupuesto del proyecto.

1.5. BREVE RESUMEN DE LOS PRODUCTOS OBTENIDOS

Listado de productos obtenidos una vez finalizado el trabajo final de máster:

→ Plan de trabajo
→ Memoria
→ Repositorio público en *GitHub* con el código desarrollado.
→ Presentación visual para apoyar la defensa del TFM.
→ Autoevaluación del proyecto

1.6. ESTRUCTURA DEL PROYECTO

El TFM se divide en siete capítulos:

- En el **primer capítulo** se hace una breve introducción sobre la problemática a la que se pretende dar solución en este proyecto. Además, se aporta un marco teórico sobre los distintos componentes de la sangre, (tipos de células, trombocitos, anticuerpos…), y la importancia de su análisis. Esto permitirá facilitar el entendimiento del trabajo desarrollado y justificar su elección. Al final de este apartado se fija el objetivo global que se pretende alcanzar y unos objetivos específicos necesarios para su consecución.

- En el **segundo capítulo** se realiza una introducción al concepto de red convolucional, detallando su estructura y sus aplicaciones en el reconocimiento de células de sangre periférica poniendo como ejemplo algunas de las arquitecturas de red más utilizadas. Posteriormente, se explica la idea de *Vision Transfomer*, su estructura y los ámbitos en los que ha sido aplicado hasta ahora. De esta manera se pretende conseguir una visión global de los métodos existentes y las líneas de investigación seguidas para su mejora.

- En el **tercer capítulo** se detalla el diseño e implementación de las redes junto con el proceso de entrenamiento seguido para el desarrollo del proyecto.

- En el **cuarto capítulo** se muestran los resultados del entrenamiento llevados a cabo y se realiza un análisis crítico sobre las diferencias existentes entre los resultados obtenidos en comparación con trabajos similares abordados con el uso de CNNs.

- En el **quinto capítulo** se elabora una conclusión basada en los resultados obtenidos y se analizan las posibles líneas futuras del proyecto, por tratarse de un tema de gran potencial e interés clínico.

- En el **sexto capítulo** un glosario con las abreviaturas o términos específicos utilizados.

- En el **séptimo capítulo** la bibliografía que se ha utilizado para elaborar este trabajo.

2. ESTADO DEL ARTE

En las últimas décadas el campo de la Visión Artificial en el sector sanitario ha experimentado una gran evolución debido al cambio de paradigma en los métodos utilizados para el reconocimiento, clasificación y tratamiento de imágenes.

Desde la aparición de las técnicas basadas en Inteligencia Artificial, concretamente del *Deep Learning*, se han podido abordar multitud de tareas de forma mucho más eficiente. La utilización de las redes neuronales en las que pequeños cambios en los datos no necesitan de grandes modificaciones en los algoritmos de aprendizaje, supone una de las grandes ventajas que presentan estas nuevas tecnologías frente a los métodos utilizados anteriormente.

Esto es posible gracias a que los métodos de *Deep Learning* basados en redes neuronales permiten conformar una representación jerárquica de las características contenidas en las imágenes con una increíble capacidad de modelado al estar conformadas por un elevado número de capas. Pudiendo así, a partir de una base de datos lo suficientemente grande y un correcto entrenamiento, otorgarles una alta capacidad de generalización con la que poder realizar predicciones de forma exitosas sobre un conjunto de datos completamente nuevo.

Para entender la complejidad detrás de estas afirmaciones, en este apartado, después de una breve introducción sobre la sangre, sus componentes y algunos modelos propuestos hasta el momento para su análisis, se procede a analizar en mayor detalle una de las arquitecturas de red de aprendizaje profundo más utilizada en este campo, la red neuronal convolucional o CNN. Para posteriormente abordar sus diferencias respecto a una de las nuevas arquitecturas propuestas para este mismo cometido, los *Vision Transformer* o ViT.

2.1. CLASIFICACIÓN DE CÉLULAS SANGUÍNEAS

2.1.1. LA SANGRE

La sangre es un fluido complejo formado por plasma, líquido extracelular, y elementos sólidos: glóbulos rojos, glóbulos blancos y plaquetas. Por tanto, podemos considerar la sangre como una suspensión de elementos celulares en plasma (3).

2.1.2. COMPONENTES DE LA SANGRE

Si tratamos de explicar cada uno de los elementos que componen la sangre en mayor profundidad debemos tener en cuenta que estos a su vez se encuentran conformados por diferentes compuestos.

- Plasma, es una solución acuosa de color blanco pálido de electrolitos, proteínas plasmáticas, hidratos de carbono y lípidos. Siendo las tres proteínas más abundantes en su composición: la albúmina, el fibrinógeno y las γ-globulinas. Las γ-globulinas incluyen las **inmunoglobulinas** o anticuerpos, que pueden separarse a su vez en IgA, IgD, IgE, IgG e IgM (3).

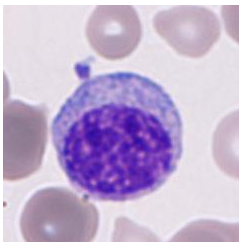

Figura 5. Imagen inmunoglobulina del dataset utilizado.

Los diferentes tipos celulares que podemos encontrar en la sangre son producto de un proceso conocido como **hematopoyesis** (4). Gracias a la complejidad de este proceso podemos distinguir una gran variedad de tipos celulares, que a su vez se encuentran implicados en importantes funciones de nuestro organismo como son: la respuesta inmunitaria, el transporte de gases o la hemostasia (5) .

- Glóbulos rojos, constituyen el grupo celular más abundante de la sangre, compuesto por células bicóncavas no nucleadas. El mantenimiento de su morfología es posible gracias a un citoesqueleto anclado a la membrana plasmática a través de glicoforina y el intercambiador de Cl-HCO_3. La forma distintiva del glóbulo rojo proporciona una relación superficie-volumen mucho mayor que la de una célula esférica; esto maximiza el área de difusión y minimiza las distancias de difusión intracelular para el intercambio de gases (3).

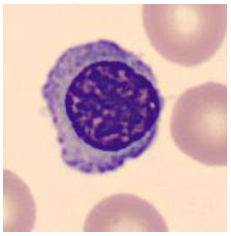

Figura 6. Imagen eritroblasto del dataset utilizado.

- Glóbulos blancos, podemos dividirlos en dos grandes grupos: los granulocitos, por un lado, y los linfocitos y monocitos, por otro. Su función principal se relaciona con la respuesta inmune frente a infecciones.

Los granulocitos se denominan así por sus gránulos citoplasmáticos, que en un frotis de sangre teñido con MGG aparecen rojos (**eosinófilos**), azules (**basófilos**) o intermedios (**neutrófilos**) (3).

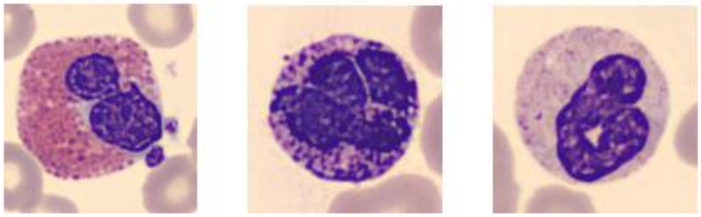

Figura 7. Granulocitos: eosinófilo, basófilo y neutrófilo del dataset utilizado.

Frente a los **monocitos** y **linfocitos** que no presentan gránulos citoplasmáticos.

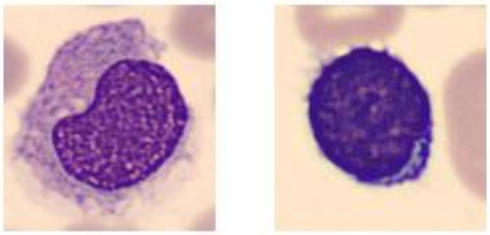

Figura 8. Monocito y linfocito del dataset utilizado.

- **Plaquetas,** son fragmentos sin núcleo. Las plaquetas se forman en la médula ósea a partir de células grandes llamadas megacariocitos, cuya maduración depende de la TPO y la IL-3. Cada megacariocito puede producir hasta varios miles de plaquetas (3) . Se trata de elementos esenciales en la hemostasia.

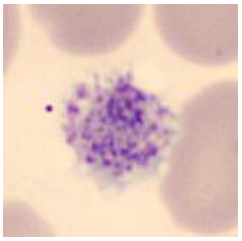

Figura 9. Imagen plaquetas del dataset utilizado.

2.1.3. IMPORTANCIA DEL ANÁLISIS SANGUÍNEO

El **análisis sanguíneo** es una de las herramientas más utilizadas para determinar el estado de salud de un paciente y el paso inicial para la prevención de muchas otras enfermedades. Es por ello por lo que en muchas ocasiones se recomienda realizar este tipo de análisis de forma rutinaria.

En ocasiones, debido a la naturaleza inespecífica de los síntomas y signos de muchas enfermedades con el **hemograma**, en el que se realiza el recuento de los diferentes componentes de la sangre y se comprueba si estos se encuentran dentro de los parámetros normales no es suficiente. Es necesario complementarlo con un análisis microscópico cuidadoso del **frotis sanguíneo** teñido, para poder estudiar las características morfológicas, como la forma del núcleo o la textura de dichos compuestos.

Este tipo de estudios nos permite determinar enfermedades como la leucemia (6) o la malaria (7).

2.1.4. MODELOS PROPUESTOS PARA LA CLASIFICACIÓN DE CELULAS SANGUÍNEAS

Las tecnologías basadas en aprendizaje profundo aplicada a la imagen médica pueden llegar a ser las tecnologías con mayor impacto desde la aparición de la imagen digital. La mayoría de los investigadores creen que, en los próximos 15 años, las aplicaciones basadas el aprendizaje profundo no solo determinaran la mayoría de los diagnósticos, pero también ayudará a predecir enfermedades, prescribir medicamentos y guiar tratamientos (8).

Como ya hemos visto en el apartado anterior algunos ejemplos de la importancia de la clasificación celular en algunas enfermedades, vamos a describir en mayor detalle algunas de las arquitecturas de red de aprendizaje profundo propuestas en la literatura para automatizar el proceso de la clasificación de células sanguíneas con diferentes propósitos.

- **CNN desarrollada para la clasificación de los diferentes tipos de glóbulos blancos**. Este proceso es especialmente importante en la monitorización de la efectividad de la quimioterapia en pacientes con cáncer. El modelo propuesto consta de dos capas convolucionales y una de *pooling* seguida de una capa densa con una capa oculta y una capa de salida (9).

- **Modelo VGG-SVM desarrollado para la detección de la malaria** (7). En el caso de la malaria el parásito *Plasmodium* infecta los glóbulos rojos del organismo huésped provocando su ruptura. Un diagnóstico temprano de esta enfermedad permite frenarla a tiempo. En este caso la red propuesta para la detección de los glóbulos rojos infectados se genera a partir de la unión de dos modelos distintos: una *Visual Geometry Group* (VGG) (10), una de las arquitecturas de red CNN más conocidas, junto con un algoritmo una arquitectura *Support Vector Machine* (SVM) (11).

- Modelo híbrido para la detección de linfoblastos de tipo B en pacientes con leucemia (12). En este caso los linfoblastos presentan una morfología muy difícil de analizar, por lo que disponer de modelos de aprendizaje automático que puedan realizar esta tarea de forma eficiente puede suponer un importante avance en el diagnóstico y tratamiento de esta enfermedad. Concretamente el modelo propuesto en este trabajo se basa en un modelo híbrido de VGG (10) y MobilNet (13).

2.2. REDES CONVOLUCIONALES (CNNS)

Las redes convolucionales (CNNs) se pueden definir como un conjunto de unidades neuronales interconectadas, donde cada una presenta sus propios pesos y sesgos inicializados de forma aleatoria y difundidos por toda la red en el paso de *forward propagation*.

Es durante el entrenamiento de la red cuando los pesos y sesgos, se van actualizando y propagando en función del error cometido durante el proceso de *backward propagation*. Asociando a cada una de las entradas sus correspondientes valores en función de la importancia relativa que estas tengan en la salida.

En la Figura 10 podemos ver la principal diferencia de este tipo de redes de las redes neuronales convencionales es que parten de la asunción de que los datos de entrada son imágenes. Permitiendo así, introducir ciertas modificaciones en la arquitectura de la red para hacerla más eficiente en el proceso de *forward propagation* y para poder reducir el número de parámetros con el fin de evitar el *overfitting* (14).

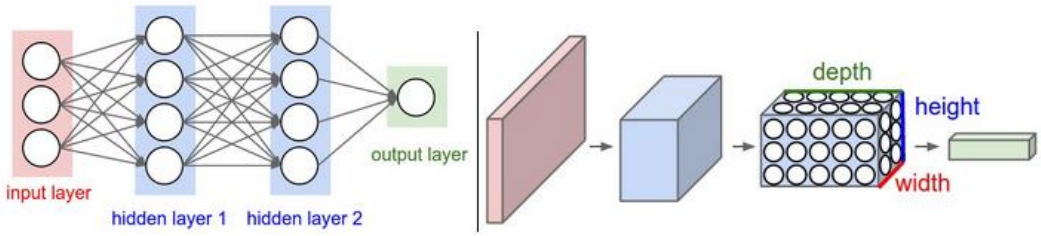

Figura 10. Diferencia entre una red neuronal convencional (imagen izquierda) y una red convolucional (imagen derecha) (14).

2.2.1. ARQUITECTURA DE UNA CNN

En la arquitectura de una CNN se pueden distinguir diferentes tipos de capas, las cuales además pueden tener o no, tanto parámetros como hiperparámetros.

Capa convolucional

Son las capas más importantes de la red, ya que en ellas se realizan las operaciones de mayor gasto computacional, las convoluciones. Esta operación matemática se realiza sobre la imagen de entrada aplicando una serie de filtros de aprendizaje (14).

Cada filtro posee unas dimensiones mucho más pequeñas que la imagen estudiada, pero de la misma profundidad y se convoluciona con la imagen de entrada, realizando el producto punto a

punto de los valores del filtro con cada una de las posiciones de la imagen. De esta forma los filtros aplicados extraen las características más importantes de la imagen original con el fin de reducir el número de parámetros a procesar posteriormente, conformando un mapa bidimensional de valores de activación. El resultado final de aplicar esta capa es un volumen compuesto por cada uno de los mapas bidimensionales obtenidos para cada uno de los filtros aplicados (14).

Para realizar de forma eficiente los cálculos del paso de *forward propagation* en las capas convolucionales, estas son implementadas como matrices de multiplicación. Podemos visualizar el proceso en la Figura 11 en la que a medida que el filtro se va desplazando por la imagen tridimensional, con el valor de salto que corresponda, se van multiplicando sus valores por los valores originales de la imagen. Finalmente, las consecutivas multiplicaciones se suman obteniendo un único valor por cada posición en la que se encuentre el filtro, obteniendo un volumen de salida mucho más pequeño mostrado en verde.

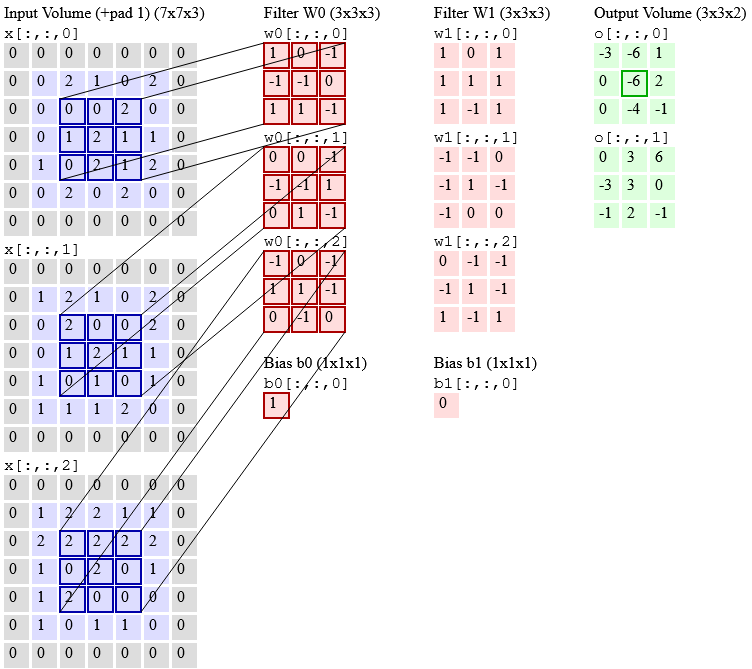

Figura 11. Ilustración de una capa de convolución (14).

Capa de reducción (*pooling*)

Además de las convoluciones discretas, las operaciones de *pooling* constituyen otro elemento importante de las CNN. Estas operaciones de *pooling* reducen el tamaño de los mapas de características utilizando alguna función como la media (*mean-pooling*) o el valor máximo (*max-polling*) calculado a partir de las subregiones. En cierto sentido, el *pooling* funciona como una convolución discreta, pero sustituye la combinación lineal descrita por el núcleo por otra función (15). En la Figura 12 podemos observar un ejemplo de *max-pooling* utilizando un filtro 3x3 y un salto de 1x1.

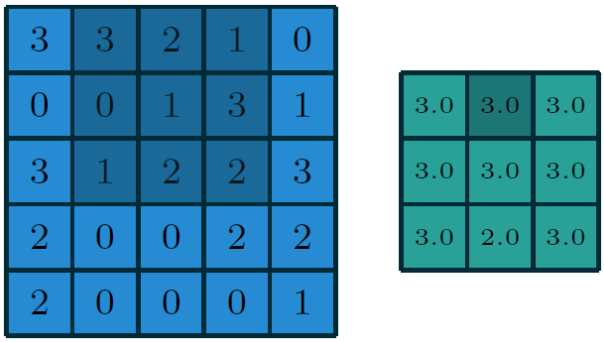

Figura 12. Ejemplo de max-pooling con un filtro 3x3 y salto 1x1. (15)

Capa densa (*fully connected layer)*

Las capas densas están formadas por neuronas totalmente conectadas a las neuronas de las capas adyacentes, considerando cada píxel como una neurona en sí. Esta disposición es análoga a la forma en que se disponen las neuronas en las redes neuronales tradicionales como las del perceptrón multicapa (2).

2.3. VISION TRANSFORMER

2.3.1. ¿QUÉ ES UN TRANSFORMER?

Para poder entender correctamente el funcionamiento de ViT es necesario entender el funcionamiento de su elemento principal el *Transformer.* Este tipo de arquitecturas fueron desarrolladas con el fin de mejorar las limitaciones que las unidades LSTM (*Long short-erm memory*) y GRU (*Gated recurrent unit)* presentaban a la hora de tener en cuenta la relación temporal entre las diferentes variables.

El elemento diferenciador de los *Transformer* se encuentra en su arquitectura, creada para deshacerse de los sistemas recurrentes de los que los modelos anteriormente mencionados hacían uso. Para ello se sirvieron de los mecanismos de *self-attention*.

En la Figura 13 se muestra un esquema de la arquitectura del *Transformer*:

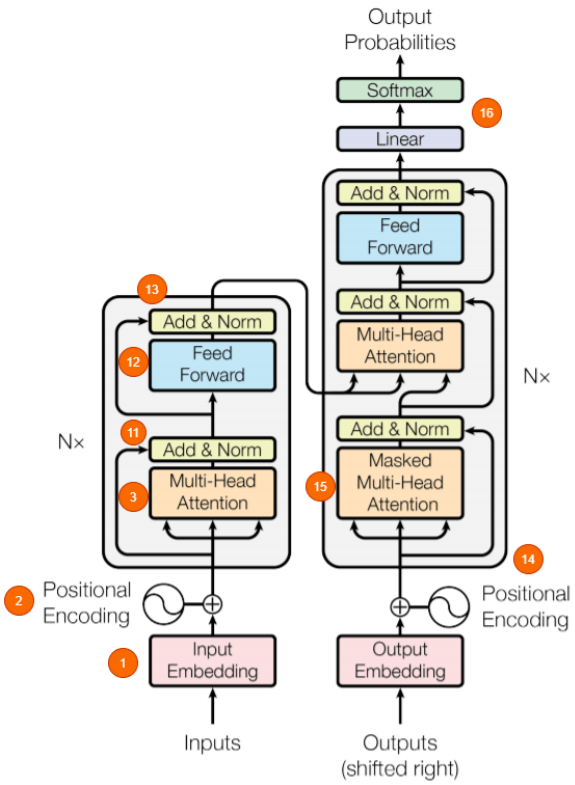

Figura 13. Arquitectura de un Transformer (16)

A grandes rasgos podemos distinguir en su arquitectura dos partes principales: el **decodificador** en la parte izquierda y el **codificador** en la derecha del esquema de la Figura 13. El codificador es el encargado de tomar la entrada y generar a partir de ella una representación intermedia que posteriormente será decodificada por el decodificador. Este, a su vez, se encargará de generar la salida correspondiente.

Explicando en mayor detalle este proceso, los datos de entrada son integrados en un vector gracias a la capa superpuesta o *embedding layer* [1], permitiendo obtener una representación vectorial para cada unidad de información. En la siguiente etapa se realiza una codificación posicional [2] con el fin de representar el orden de la secuencia. Para poder utilizar el mecanismo de *multi-headed attention* [3]. En él, mostrando en mayor detalle en la Figura 14, se generan tres vectores (*query*, *key* y *value*) [4] para cada uno de los vectores de entrada. A partir de *query* y *key* se obtiene una matriz de puntuación [5] que representa la relación que guarda cada unidad de información con el resto, de tal forma que a mayor puntuación mayor relación y viceversa. Esta matriz de puntuación es reducida [6] posteriormente a las dimensiones de *query* y *key* con el fin de asegurar la estabilidad de los gradientes (16).

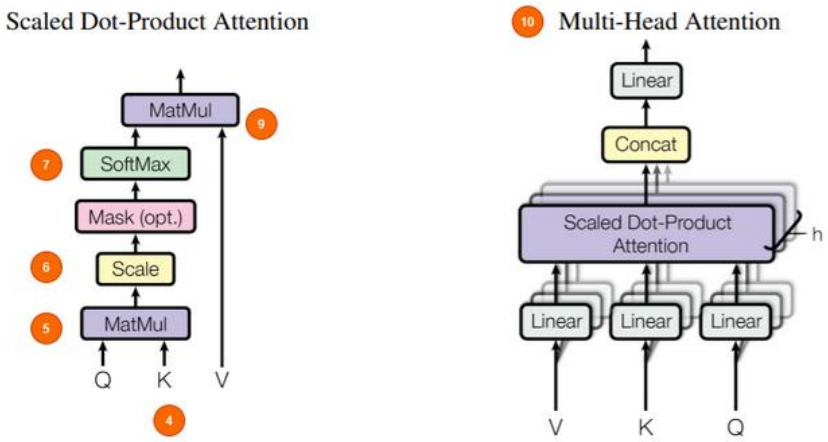

Figura 14. Arquitectura multi-headed-attention (16).

En la siguiente etapa del proceso la matriz de puntuaciones se somete a un proceso de *softmaxing* [7] para convertir las puntuaciones de atención en probabilidades. La matriz resultante [9] se multiplica por el vector de *value*, haciendo que las probabilidades más altas cobren mayor importancia y las de menor puntuación sean consideradas como irrelevantes. Ocurriendo todo este proceso explicado en detalle en el *Scaled Dot-Product Attention* [10], cuya salida concatenada de los vectores *query, key* y *value* pasa por una capa lineal para su posterior procesamiento (16).

El mecanismo de *self-attention* es aplicado a cada elemento considerado como unidad de información por la red, de tal forma que, al ser independientes unas de otras, se utiliza una copia del módulo de *self-attention* para procesar todo de forma simultánea (*multi-headed*). Luego los vectores de valores de salida son concatenados y se añaden a la conexión residual proveniente de la capa de entrada y la representación resultante se normaliza a través de una capa de normalización. (La conexión residual ayuda a que los gradientes fluyan a través de la red, mientras que la capa de normalización ayuda a reducir el tiempo de entrenamiento y a estabilizar la red (16).

Por último, la salida obtenida pasa por una red *feed-forward,* es decir un perceptrón en el que las conexiones entre las neuronas son unidireccionales, para obtener una mejor representación. Las salidas se vuelven a normalizar y se les añaden los residuos de la capa anterior. De esta forma la salida del codificador junto con las entradas de los pasos anteriores pasan al decodificador (16). Donde siguen un mecanismo muy similar al descrito para el codificador para dar como resultado final las probabilidades correspondientes de salida para cada una de las entradas iniciales.

2.3.2. ARQUITECTURA DE VIT

Los *Vision Transformer* (ViT) son una de las nuevas arquitecturas propuestas en el campo de la Visión Artificial. Su elemento diferenciador es no necesitar recurrir al uso de capas convolucionales en su arquitectura para poder realizar complejos cálculos sobre las imágenes. Esta importante ventaja les permite realizar entrenamientos más rápidos, gracias a un menor

gasto computacional, al haber renunciado a una de las operaciones que mayores recursos consume: la convolución.

Los *Vision Transformer* surgen a partir de modelos propuestos inicialmente para el procesamiento de lenguaje natural (NPL). Estos modelos están basados en *self-attention*, un mecanismo de atención que relaciona diferentes posiciones de una misma secuencia para calcular una representación de la misma (17) .

En este caso concretamente, ViT fue desarrollado utilizando el modelo original de *Transformer* propuesto por Vaswani (17). Constituido por una arquitectura general de capas totalmente conectadas apiladas de *self-attention* y operaciones punto a punto, tanto para el codificador como para el decodificador.

El mecanismo central de la arquitectura ViT es el *self-attention*, que permite comprender la conexión entre las entradas. Cuando los *Transformers* se aplican al NPL, calculan la relación entre las palabras de forma bidireccional, lo que significa que el orden de entrada no importa, a diferencia de las RNN (18). Un modelo con arquitectura *Transformer* maneja entradas de tamaño variable utilizando pilas de capas de *self-attention* en lugar de CNNs y RNNs (19).

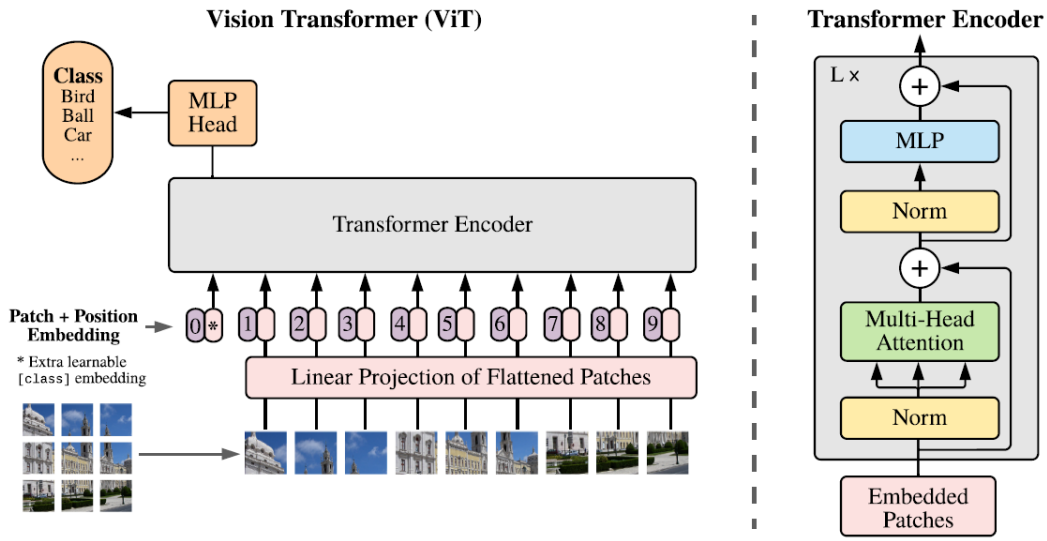

Figura 15. Esquema de la estructura de un ViT. (1)

Debido a la alta dimensionalidad de las imágenes de entrada para evitar realizar el cálculo de *self-attention* sobre la totalidad de la imagen, esta es divida en pequeños parches sobre los que se realizan las operaciones. Pasando así a considerar los parches como la unidad de estudio frente a los píxeles con el fin de reducir considerablemente la dimensionalidad del cómputo.

En la Figura 15 podemos ver como la primera capa del ViT proyecta linealmente los parches aplanados en un espacio de menor dimensión. Después de la proyección, se integra la posición aprendida a las representaciones del parche. A través de las cuales el modelo aprende a codificar la distancia dentro de la imagen en función de la similitud de las posiciones integradas. Es decir, los parches más cercanos presentan posiciones similares al igual que los parches pertenecientes a la misma estructura fila-columna. De esta forma ViT integra información de toda la imagen, incluyendo las capas más bajas, haciendo que el modelo sea capaz de aprender qué regiones de la imagen son semánticamente relevantes para la clasificación (1).

3. DISEÑO E IMPLEMENTACIÓN

3.1. DATASET

El conjunto de datos sobre el que se realiza el estudio se corresponde con 17092 imágenes de células normales individuales adquiridas por medio del *CellaVision DM96* en el Laboratorio Central del Hospital Clínic de Barcelona (20). Trabajar con este repositorio nos ofrece múltiples ventajas puesto que se encuentra constituido por imágenes de alta calidad y de las mismas dimensiones (363x360x3), que, a su vez, se encuentran divididas en subdirectorios según sus tipos celulares. Gracias a esto se pudo ahorrar mucho tiempo del preprocesamiento que muchas veces requieren los repositorios públicos.

En este caso concreto se podía optar entre dos implementaciones diferentes: la primera utilizando como clases los 8 grupos principales celulares, es decir, en función de la división de subdirectorios. O en función de 13 clases obtenidas a partir de los nombres de las imágenes. Para esta segunda composición los grupos principales *neutrophil* se subdivide en *band* y *segmented*, mientras que *ig* se divide en *promyelocyte, metamyelocyte* y *myelocyte.*

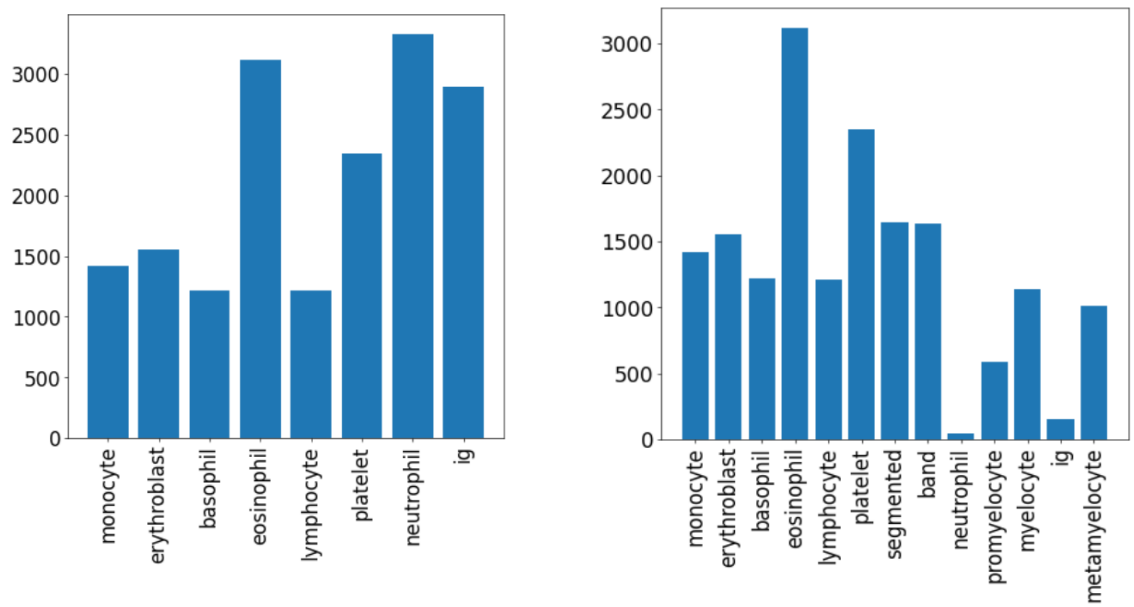

Figura 16. Gráfico de barras para 8 clases (izquierda) y para 13 clases (derecha).

En vista de los resultados obtenidos al analizar la cantidad de casos disponibles para cada uno de los grupos se opta finalmente por realizar el entrenamiento con las clases principales, es decir, 8 tipos celulares principales.

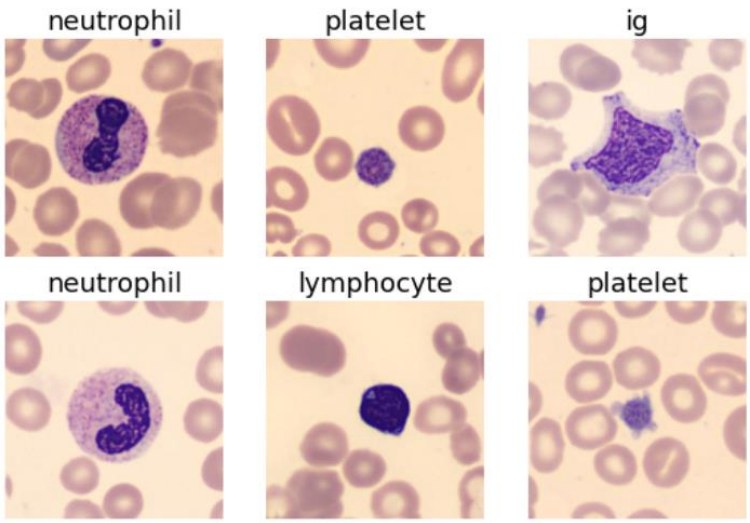

Figura 17. Ejemplo de imágenes del dataset con sus correspondientes etiquetas

Para la preparación de los datos se hizo uso de la API *data block* de *fast.ai*, definiendo como conjunto de test el 20% de las imágenes. Las correspondientes etiquetas de clasificación se obtuvieron a partir de los nombres de cada uno de los archivos.

Además, se fijó el mismo tamaño para todas las imágenes con el fin de prevenir algún caso conflictivo en el que pudiese existir alguna ligera diferencia de dimensiones. Por último, puesto que el conjunto de datos no es muy extenso se definieron método de *data augmentation* con un *presizing* de 0.75.

3.2. LIBRERÍAS Y PAQUETES

Para la implementación de las redes se utilizó la plataforma *Google Colab*. Definiendo en las opciones de configuración la utilización de GPU como entorno de ejecución. En este caso al trabajar con un entorno en la nube no se pudo seleccionar el tipo de GPU a utilizar, si no que este fue seleccionado de forma automática por la plataforma en función de la disponibilidad en el momento de la ejecución.

Las librerías de las que se hizo uso el trabajo fueron:

- ***Fast.ai***: biblioteca de código abierto diseñada especialmente para *Deep Learning* y construida sobre *Pytorch* cuya ventaja principal se sustenta en proveer de una API al usuario con multitud de recursos implementados para facilitar el entrenamiento y validación de diferentes modelos en un entorno mucho más cómodo.
- ***Timm***: librería de código abierto en la que podemos encontrar multitud de las arquitecturas de aprendizaje profundo propuestas en la bibliografía.
- ***Os:*** librería de Python para facilitar el manejo de los directorios.
- ***Pandas:*** librería de Python para facilitar la creación de tablas.
- ***Matplotlib***: librería de Python para la creación y edición de elementos gráficos.

3.3. RESNET

3.3.1. DISEÑO

Para el desarrollo de esta implementación se hizo uso de la librería *fast.ai* en la que se encuentran disponibles los modelos preentrenados de *torchvision.*

En este caso, con el fin de posteriormente poder comprobar si el desempeño de las arquitecturas ViT para nuestra tarea es comparable al de las arquitecturas más utilizadas hasta el momento para el reconocimiento de imágenes, las CNNs se decide probar la implementación de arquitecturas *ResNet* (21). Concretamente dos de sus implementaciones preentrenadas, *ResNet18* y *ResNet34*, como su propio nombre indica de 18 y 34 capas respectivamente. Pudiendo resumir su estructura en la Figura 18 (22):

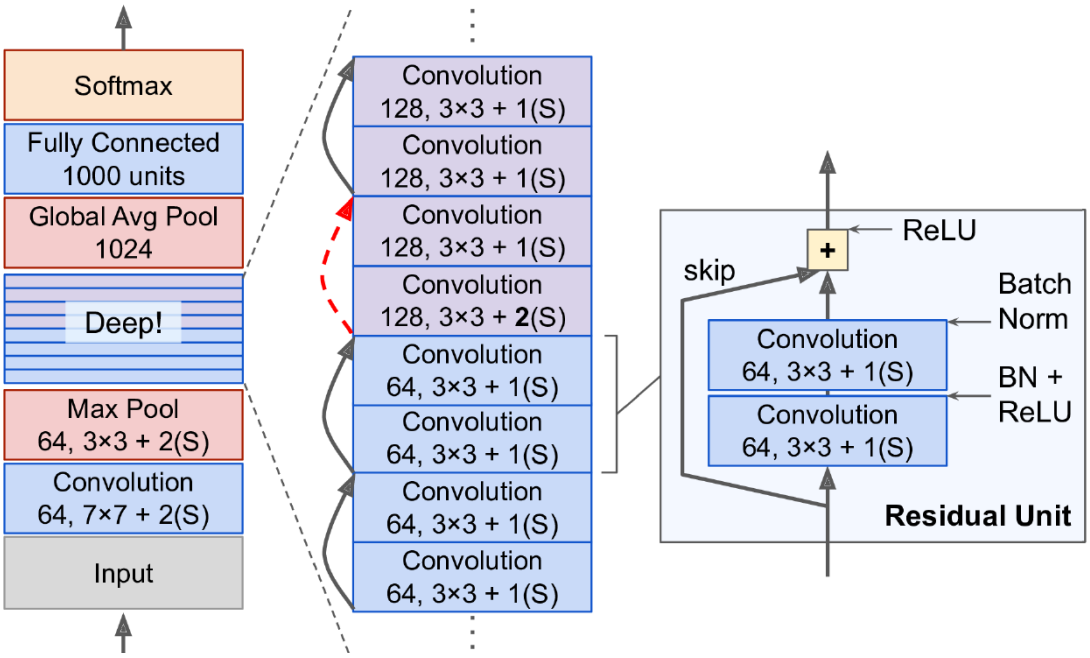

Figura 18. Arquitectura ResNet (22)

3.3.2. IMPLEMENTACIÓN

La técnica de *transfer learning*, consiste actualizar los parámetros de un modelo preentrenado para una tarea concreta, entrenándolo por una serie de *epochs* con un conjunto de datos distinto al que se usó para su preentrenamiento, puesto que se desea resolver un problema distinto. Este proceso permite ajustar de forma eficiente y con muy buenos resultados la red para el desempeño de un nuevo cometido. Además, al no disponer de un conjunto de datos muy grande la utilización de *transfer learning* puede suponer una importante ventaja. En ambos modelos se emplea un *unfreezing* gradual, eligiendo de forma manual sus valores de *learning rate*, por sus buenos resultados en modelos CNN.

3.4. VIT

3.4.1. DISEÑO

Debido a que la arquitectura ViT ha sido propuesta recientemente aún no la encontramos implementada en las librerías de *fast.ai*, por lo que utilizaremos la librería de *timm* para llevar a cabo el despliegue del modelo. En dicha librería podemos encontrar diferentes arquitecturas ViT definidas en función el número de capas. Además, para cada una de ellas existen dos versiones de tamaño de parche utilizado: 16 y 32, respectivamente.

Para el desarrollo del proyecto se decide optar por utilizar las versiones preentrenadas de ViT sobre del conjunto de datos de *ImageNet* (23), ya que al trabajar con la versión gratuita de Colab contamos con limitaciones de tiempo, memoria y capacidad de procesamiento que podrían comprometer los resultados al intentar entrenar un modelo desde cero, por lo que se decide optar por técnicas de *transfer learning* también.

3.4.2. IMPLEMENTACIÓN

En una primera aproximación al problema se decide probar el modelo de menor coste computacional disponible, es decir, el de 12 capas y tamaño de parche 32. Un tamaño de parche mayor supone un coste computacional menor dado que la longitud de la secuencia del transformador es inversamente proporcional al cuadrado del tamaño del parche. Pero los resultados no son los esperados por lo que se opta por la utilización de modelos de parches de tamaño de 16.

Por otra parte, llevar a cabo un *unfreezing* de algunas capas a la vez ofrece mejores resultados que su implementación de forma gradual, para los modelos NLP, en los que está basado ViT, por lo que este es el planteamiento seguido para el entrenamiento de todos los modelos propuesto.

Además, hacemos uso de *Big Transfer* (BiT) (24), al utilizar redes preentrenadas sobre un conjunto de datos de un tamaño mucho mayor al utilizado convencionalmente, como es el caso de *ImageNet-21k* compuesto por una gran cantidad de datos genéricos. Permitiendo al modelo extraer las características generales necesarias para poder utilizarlas con un pequeño ajuste de hiperparámetros sobre conjuntos de datos completamente diferentes y de mucho menor tamaño, obteniendo muy buenos resultados.

Además, se realizan diferentes pruebas para determinar el mejor ajuste de valores para el modelo:

- Determinación del tamaño de *batch*
  Para determinar el tamaño de *batch* se realizaron dos implementaciones de *ViT-Base* con *unfreezing* no gradual durante 15 *epochs*. El primero con el tamaño de *batch* por defecto, 64 y otro, de las mismas características con un tamaño de 128, observando un mejor desempeño para un tamaño de 64 por lo que se determina continuar con las siguientes pruebas con dicho tamaño.

- ***MixUp***
  Como técnicas de aumento de datos para intentar mejorar los resultados. Se opta por una primera implementación con un modelo de *ViT-Base* utilizando *MixUp* con el fin de mejorar las predicciones. Se trata de una técnica de *data augmentation* que permite entrenar con ejemplos virtuales construido a partir de la interpolación lineal de dos ejemplos aleatorios del conjunto de entrenamiento y sus correspondientes etiquetas (25).

- ***Label Smoothing***
  El modelo está entrenado para devolver 0 para todas las categorías excepto una, para la que devuelve 1. Esto hace que incluso 0,999 no sea "suficientemente bueno". Provocando que en muchas ocasiones el modelo aprenda a predecir activaciones con una confianza mayor a la que debería, fomentando el sobreajuste (26).
  Una manera de evitar este exceso de confianza se corresponde con lo que conocemos como *label smoothing* (27). Esta técnica consiste en reemplazar en nuestro modelo todos los valores de 0 y 1, respectivamente, con un número ligeramente menor. Evitando así el exceso de confianza y creando un modelo mucho más robusto que dará mejores resultados incluso si existen datos con etiquetas erróneas en nuestro conjunto de entrenamiento (26).

- Número de capas
  Se entrena con modelos de tamaño de parche 16, en dos de sus versiones disponibles (***ViT-Base*** de 12 capas y ***ViT-Large*** de 24 capas) con el fin de determinar cómo influye la profundidad de la red en el desempeño sobre nuestro conjunto de datos.
  Se descarta la utilización de ***ViT-Huge*** de 32 capas debido al elevado tiempo de procesamiento que consumiría su entrenamiento. Además, al no disponer de con conjunto de datos muy grande podría resultar sencillo que tendiese al *overfitting*.

3.5. INTERFAZ GRÁFICA

Con el fin hacer más accesible la herramienta desarrollada se realiza una sencilla interfaz gráfica para su uso, mostrada en la Figura 19. En ella el usuario puede seleccionar la imagen que desea clasificar desde el botón *Upload,* y posteriormente seleccionar el tipo de red en el desplegable de la que desea hacer uso para realizar la clasificación. Por último, haciendo *click* en el botón *Clasificar*, se muestra por pantalla el resultado de la clasificación con su probabilidad correspondiente.

Para su implementación se han descargado los pesos correspondientes a las mejores versiones de cada una de las diferentes redes utilizadas, dando lugar a la posibilidad de elegir entre cuatro tipos diferentes de predicciones: las realizadas por *ResNet18, ResNet34*, *ViT-Base* o *ViT-Large*.

Debido a las limitaciones de espacio en plataformas como *GitHub* se han realizado dos versiones diferentes de dicha herramienta. La primera implementación, explicada anteriormente, se puede desplegar en nuestro servidor local haciendo uso de *Voila*. *Voila* es una plataforma flexible que permite convertir un *jupyter notebook* en una aplicación web y utilizar así cómodamente nuestro clasificador de células con cualquiera de los cuatro modelos de red propuestos.

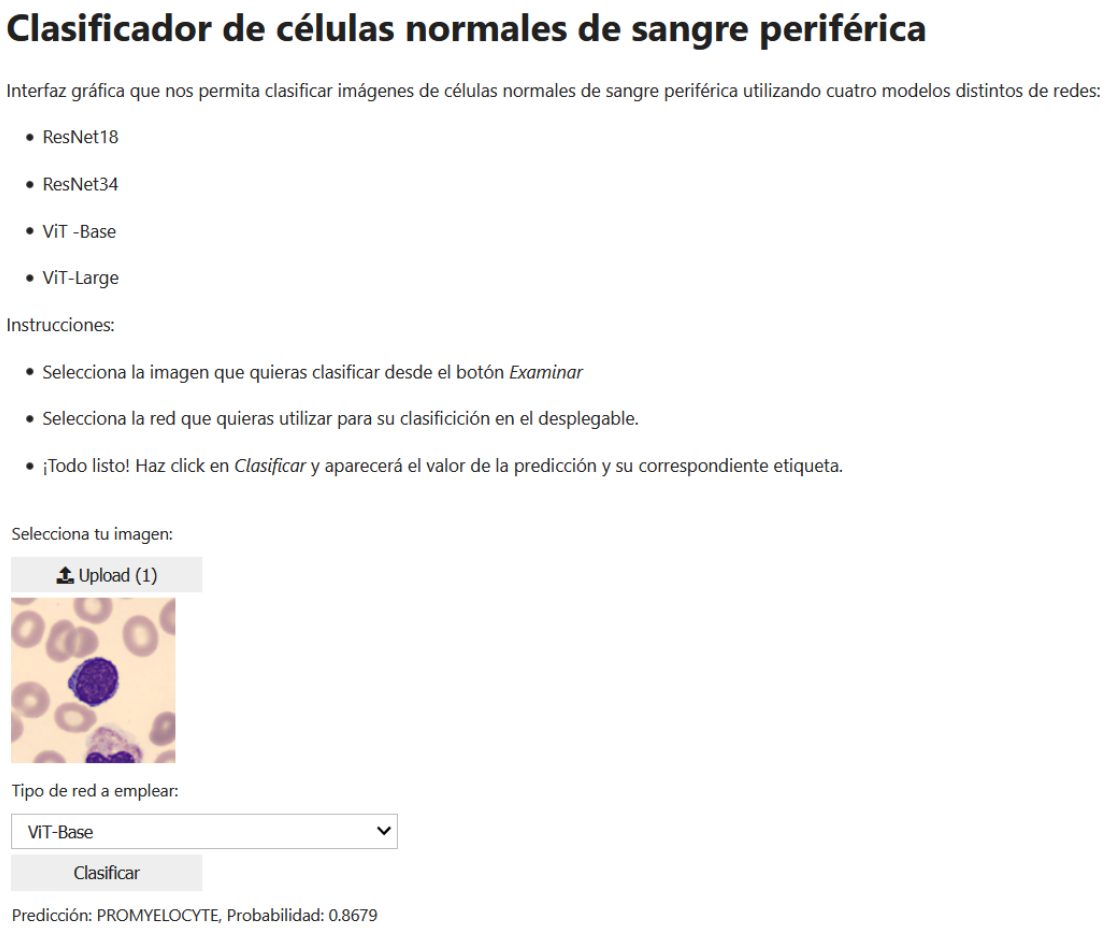

Figura 19. Interfaz gráfica completa

La segunda implementación, Figura 20, consiste en una versión reducida de dicha plataforma, en este caso solamente nos permite realizar predicciones utilizando ResNet18 y su despliegue se ha realizado utilizando *Voila* y *Binder,* obteniendo un *link* permanente a través del cual cualquier usuario de cualquier parte del mundo puede acceder de forma directa a la aplicación y hacer uso de ella. (Repositorio de la interfaz reducida: https://github.com/b-ryba/app_cell_classifier).

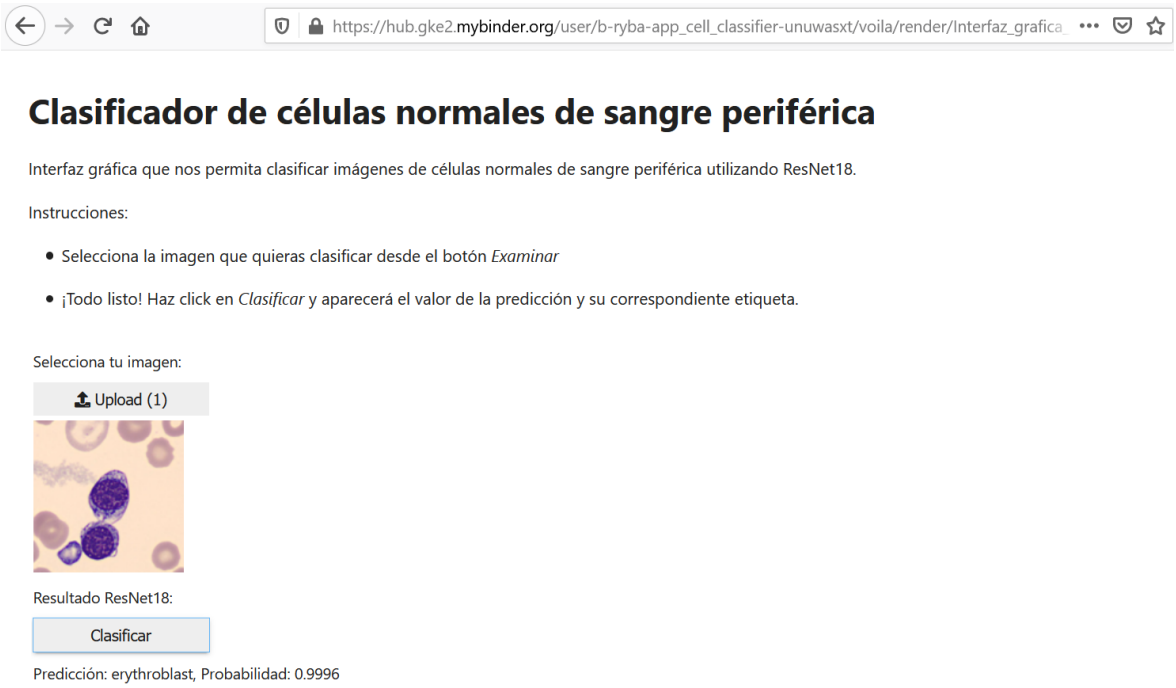

Figura 20. Interfaz gráfica reducida en la web

4. RESULTADOS Y DISCUSIÓN

4.1. RESULTADOS RESNET

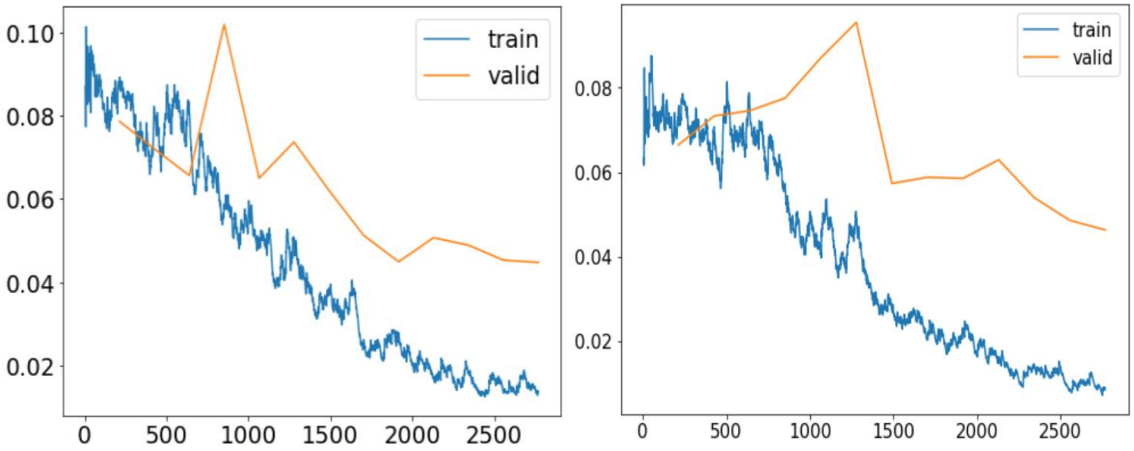

Figura 21. Entrenamiento ResNet18 (izquierda) y ResNet34 (derecha)

En las gráficas de *loss,* mostradas en la Figura 21, podemos observar como el modelo *ResNet34* es mejor al existir una mayor separación entre las curvas de entrenamiento y test para el modelo de *ResNet18*. Aun así, en general los resultados de clasificación de ambos modelos son similares y en ambos casos se llega a superar el 0.98 de *accuracy* para las predicciones. En cuanto al valor de *kappa score* y *balanced accuracy* llegan a superar ambos 0.98 para el primer modelo y el 0.98 y 0.99, respectivamente, para el segundo.

Puesto que los resultados son buenos para ambas arquitecturas se decide seleccionar ResNet18, para un análisis en mayor detalle, por ser el modelo de inferior tiempo de procesamiento debido a su menor profundidad. En la Figura 22 podemos ver como al seleccionar un ejemplo aleatorio del conjunto de test, podemos observar que acierta en la totalidad de sus predicciones.

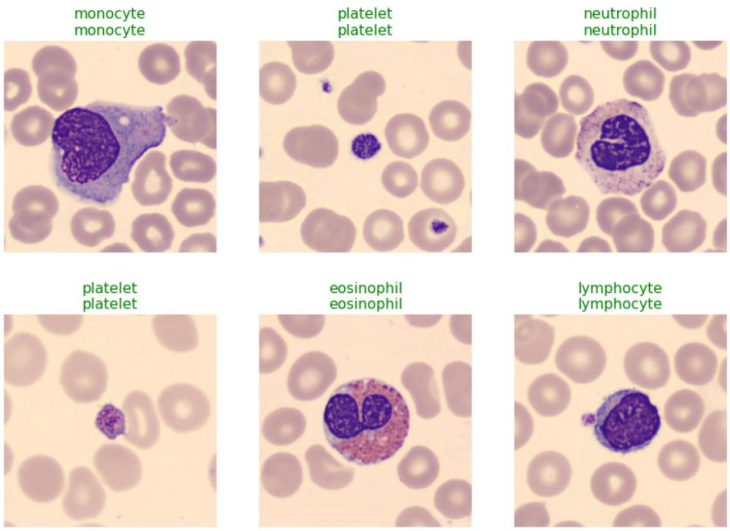

Figura 22. Ejemplo clasificación ResNet18.

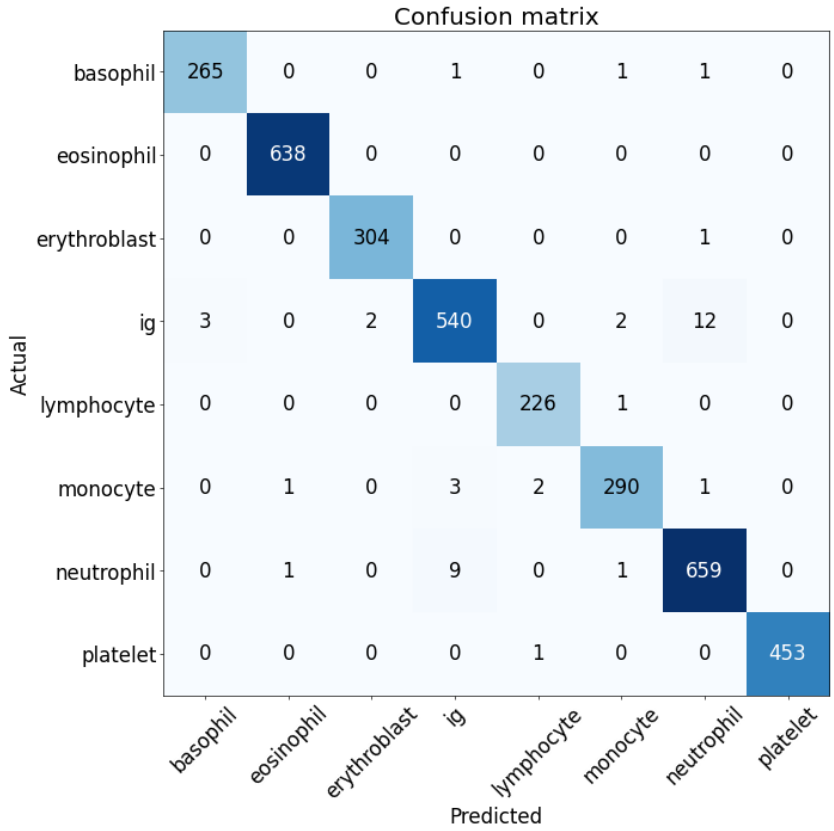

Figura 23. Matriz de confusión ResNet18

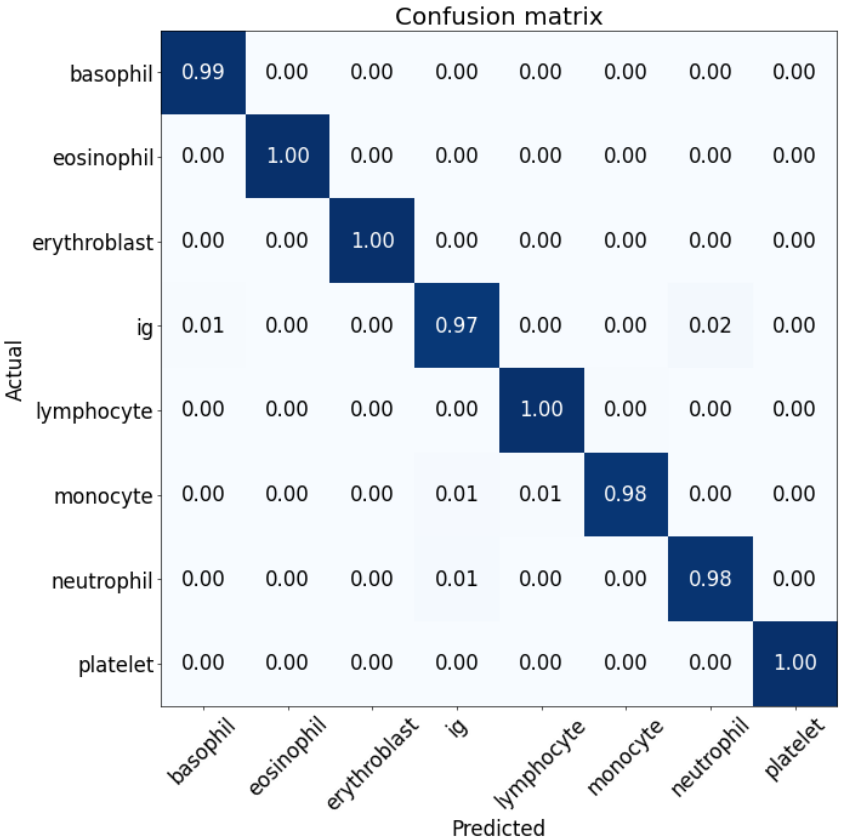

Figura 24. Matriz de confusión normalizada ResNet18

En las matrices de confusión, Figura 23 y Figura 24, mostrada anteriormente, podemos ver como las clasificaciones para todos los tipos celulares principales son muy buenas, siendo para la mitad de los casos exactos con una predicción de 1.0, mientras que para para el 50% de clases restantes iguales o superiores al 0.97 de acierto.

Con el fin de analizar en mayor profundidad los errores de la red mostramos los ejemplos que peor han sido clasificados en la Figura 25.

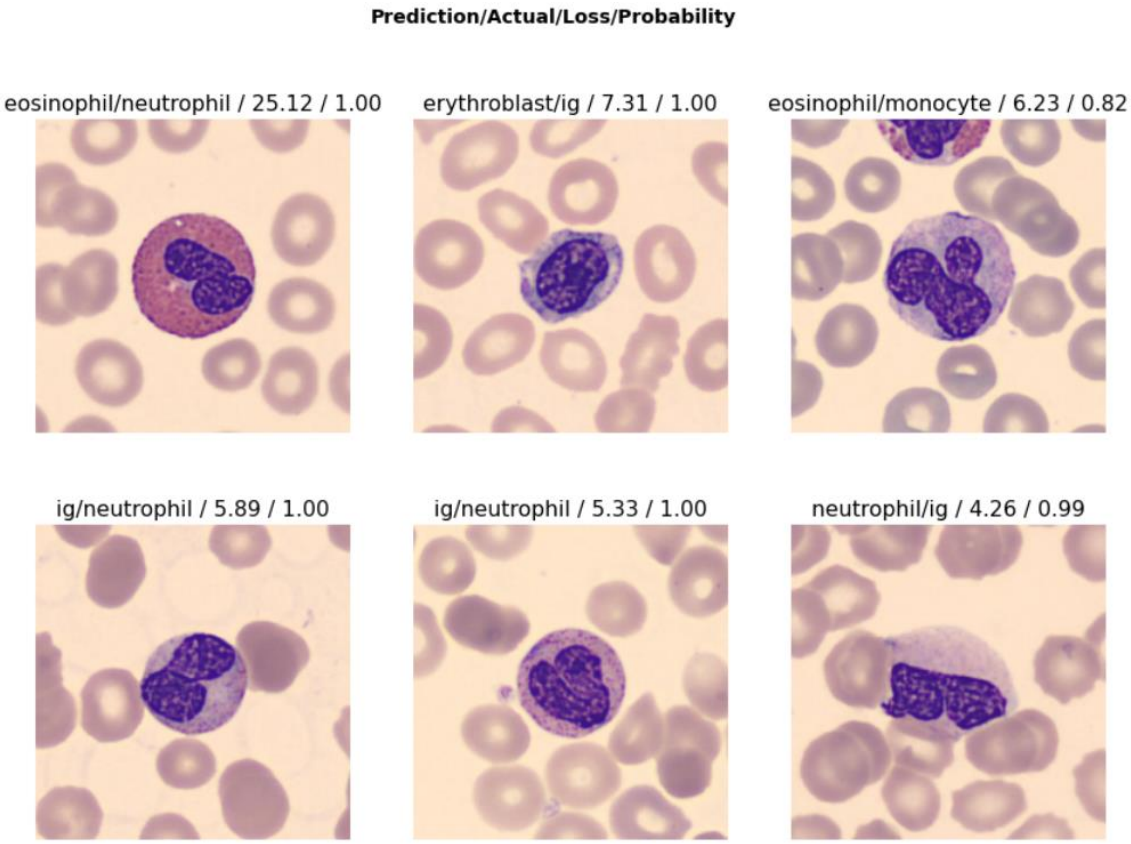

Figura 25. Casos peor clasificados ResNet18

Podemos observar como el caso *eosinophil/neutrophil* presenta un valor muy alto valor de *loss* en comparación al resto, en este caso concreto se trata de una imagen mal etiquetada puesto que sí que se trata de un eosinófilo en realidad. El grupo *neutrophil* se corresponde con uno de los grupos que, pese obtener un 0.98 en la matriz de confusión normalizada constituye uno de los valores más bajos.

En el resto de casos vemos que para tratarse de los ejemplos peor clasificados los valores de *loss* no son demasiado grandes, confirmando los buenos resultados vistos en sendas matrices de confusión. Además, podemos apreciar otro posible motivo que puede inducir al modelo a error: la presencia de dos tipos celulares en la misma imagen. En el tercer caso mostrado en la Figura 25 en el que aparecen dos tipos celulares distintos la predicción realizada por la red coincide con la célula superior mostrada en la imagen. Siendo completamente esperable que su resultado no sea el correcto en estas situaciones puesto que no ha sido entrenada para poder predecir varias etiquetas por imagen.

4.2. RESULTADOS VIT

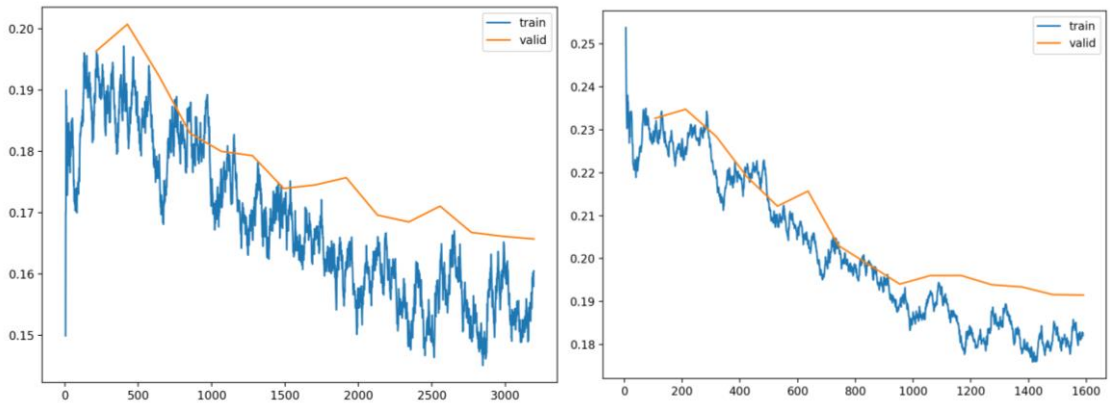

Figura 26. Entrenamiento ViT-Base batch size=64 (derecha) y batch size=128 (izquierda)

En las gráficas de *loss, mostradas en la* Figura 26, de los modelos *ViT-Base* utilizados para la **determinación del tamaño de *batch*** podemos ver pequeñas diferencias. Los valores obtenidos para un *batch* de 64 son ligeramente mejores puesto que conseguimos reducir en mayor medida los valores de *loss,* tanto para el conjunto de entrenamiento como para el de test. Esto puede apreciarse también en que las tendencias de ambos conjuntos se encuentran menos superpuestas a como ocurre para el tamaño de *batch* de 128.

Además, estas gráficas son respaldadas por sendos valores de *accuracy obtenidos durante el entrenamiento*, siendo para el primer modelo superiores al 0.94 mientras que para el segundo no llegan a superar este valor. En vista de estos resultados se determina definir el tamaño de *batch* final como 64 para el resto de los modelos.

Pese a que los resultados obtenidos son bastante buenos se intenta ver si se puede mejorar el modelo por medio de técnicas de ***data augmentation,*** concretamente *MixUp*. Por lo que se añade al modelo anterior y se obtiene la siguiente gráfica, Figura 27.

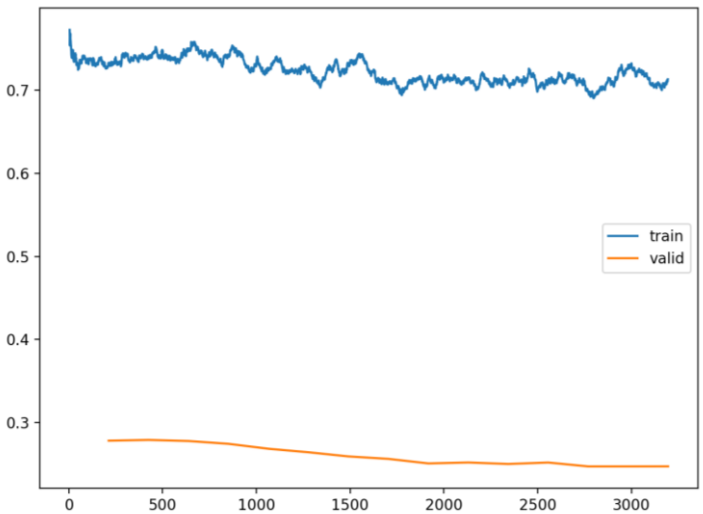

Figura 27. Entrenamiento ViT-Base con MixUp

En este caso nos encontramos ante una apariencia poco común puesto que los resultados de *loss* para el conjunto de entrenamiento y de test se encuentran muy alejados entre sí, mostrando además mucho mejores resultados para el conjunto de test que para el conjunto de entrenamiento, algo muy poco frecuente. En este caso la apariencia de la gráfica se debe principalmente a que la técnica de *MixUp* solamente es aplicada sobre el conjunto de entrenamiento, haciendo que las predicciones sean mucho más difíciles que las del propio conjunto de entrenamiento original, mientras que el conjunto de test sigue perteneciendo al conjunto de datos originales sin ningún tipo de modificaciones por lo que para la red es mucho más sencillo predecir su valor. En cuanto al *accuracy* se obtienen valores ligeramente superiores al modelo inicial sin *MixUp*, superando los 0.95. En cuanto al valor de *kappa score* y *balanced accuracy* llegan a superar en ambos casos el 0.94.

En la Figura 28, al mostrar los casos peor clasificados por dicha red es cuando podemos contrastar las notables diferencias respecto al modelo anterior, los valores de *loss* disminuyen considerablemente para estos casos.

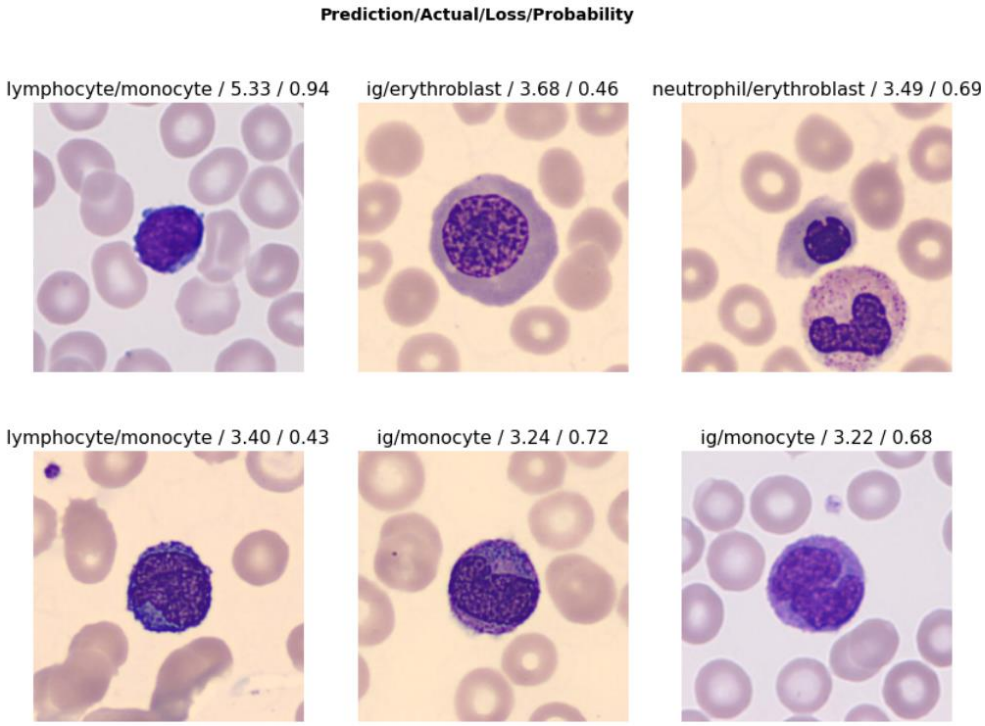

Figura 28. Casos peor clasificados ViT-Base con MixUp

En cambio, al mostrar la matrices de confusión, Figura 29 y Figura 29 , los resultados parecen peores. Esto se explica con el hecho de que al añadir *MixUp*, se evita que la red pueda estar tan segura como antes de que una imagen pertenece a una determinada clase, es por ello, que ningún valor es exactamente 1 en el caso de la matriz normalizada, aunque todos se acerquen considerablemente a este.

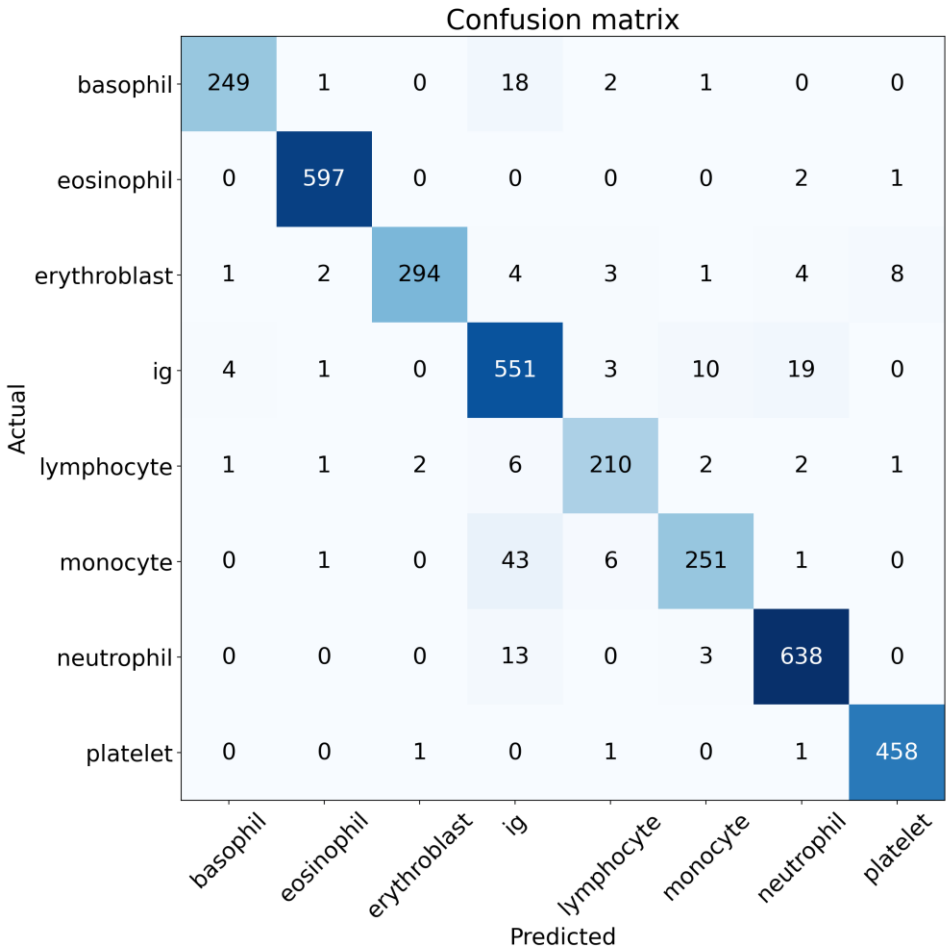

Figura 30. Matriz de confusión ViT-Base con MixUp

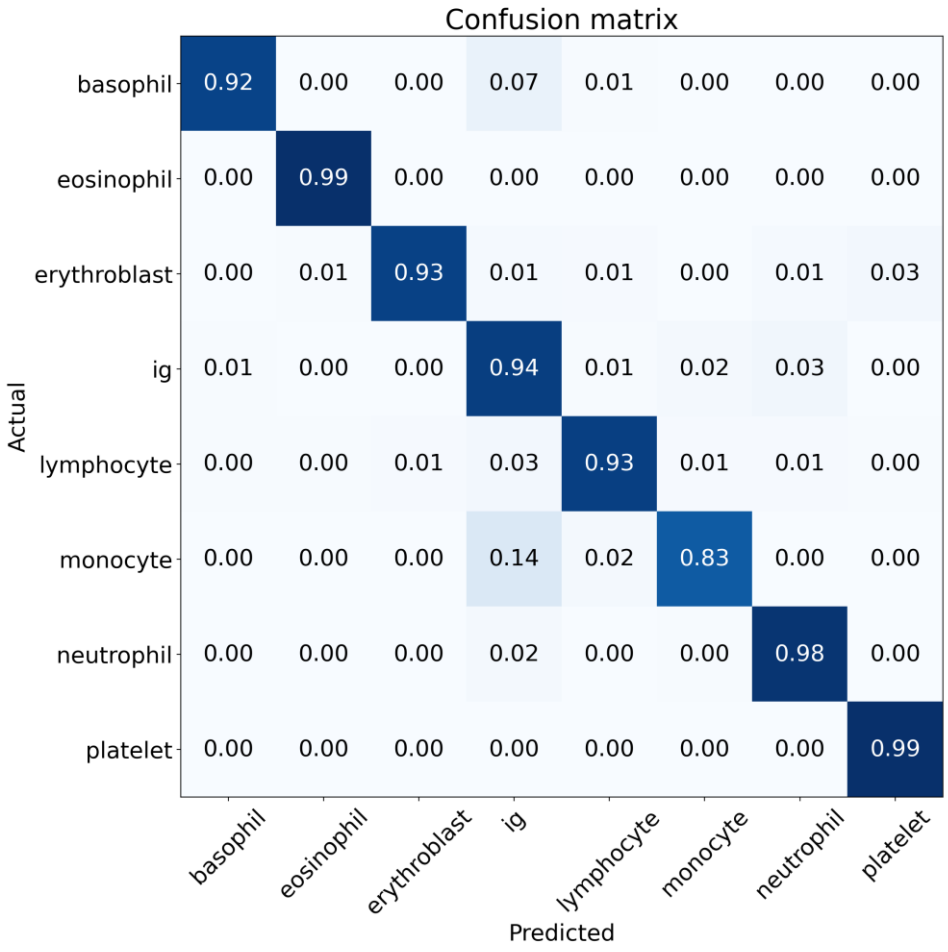

Figura 29. Matriz de confusión normalizada ViT-Base con MixUp

Para comprobar la **influencia del número de capas** en los resultados también se entrenan sendos modelos de *ViT-Large* de 24 capas, frente a las 12 de *ViT-Base*.

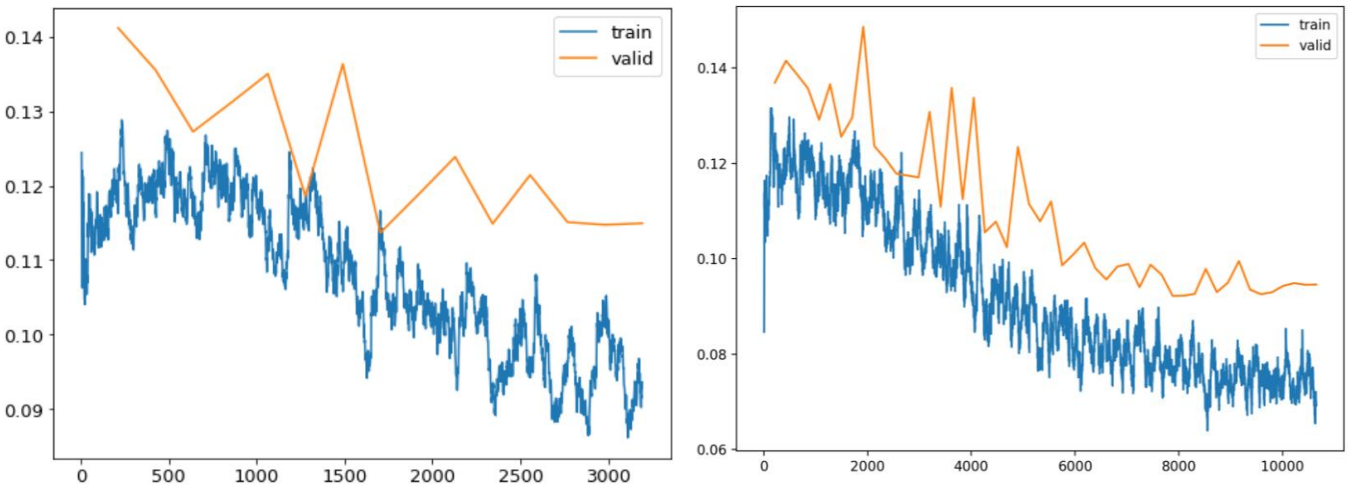

Figura 31. Entrenamiento ViT -Large 15 epochs (izquierda) y 50 epochs (derecha)

En la Figura 31 podemos ver como el primero se entrena durante 15 *epochs* seguidas y aunque los valores de *loss* sobre el conjunto de test se encuentren por encima de los de entrenamiento en ambos casos, podemos ver que los valores de *loss* en el caso de *ViT-Large* alcanzan valores más pequeños de *loss* para *ViT-Base* (Figura 26).

En vista de los buenos resultados se decide entrenar el mismo modelo durante 50 *epochs* seguidas para comprobar si seguirá mejorando o realmente como parece intuirse al final de la gráfica del entrenamiento de 15 *epochs* se entra en una zona constante. Finalmente se determina que efectivamente se trataba de una zona de meseta y que los resultados no mejoran apreciablemente por lo que se descarta la opción de añadir más *epochs*.

A continuación, se intenta mejorar el modelo *ViT-Large* añadiendo *MixUp* pudiendo comprobar que los resultados son muy similares a los obtenidos para el modelo de *ViT-Base* con *MixUp* (Figura 27). En cambio, al comparar los valores obtenidos durante el entrenamiento *se* consiguen superar los 0.95 para el *accuracy*, el *balanced accuracy* y el *kappa score*.

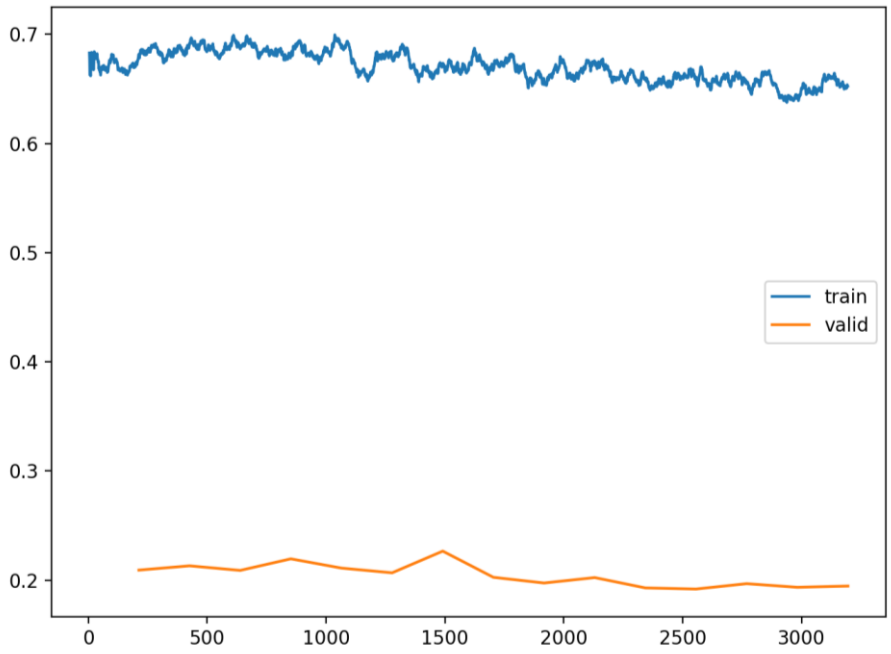

Figura 32. Entrenamiento ViT-Large con MixUp

Por último, se decide comprobar si al añadir *Label Smoothing, además* de *MixUp,* al modelo se puede mejorar su desempeño y se consigue superar los 0.96 de *accuracy*, y mantener los 0.95 para *balanced accuracy* y para *kappa score*. En vista de los resultados, este modelo parece ligeramente mejor al anterior, para realizar un análisis más certero se comparan las matrices de confusión normalizadas de ambos.

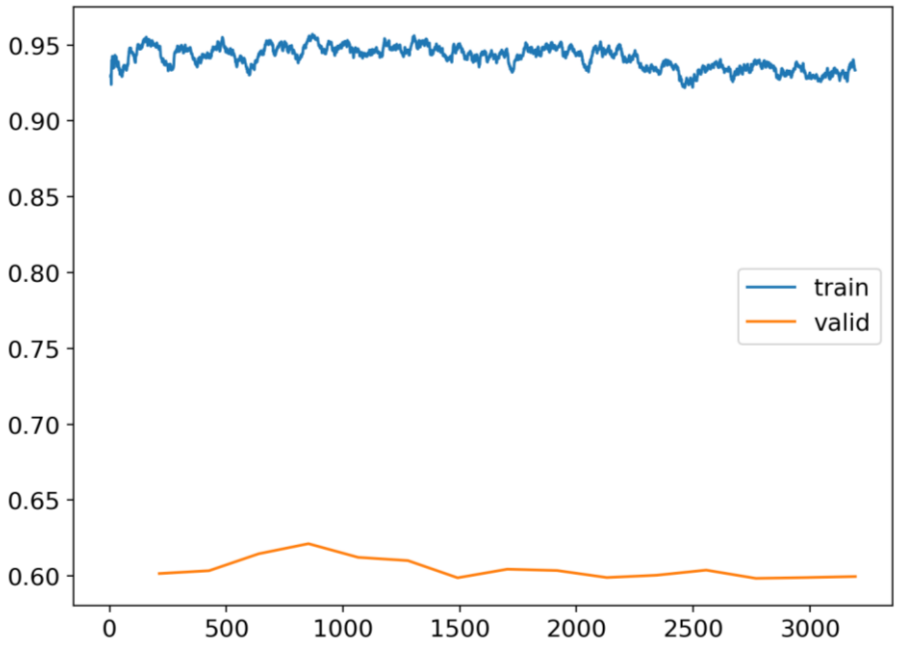

Figura 33. Entrenamiento ViT-Large con MixUp y Label Smoothing

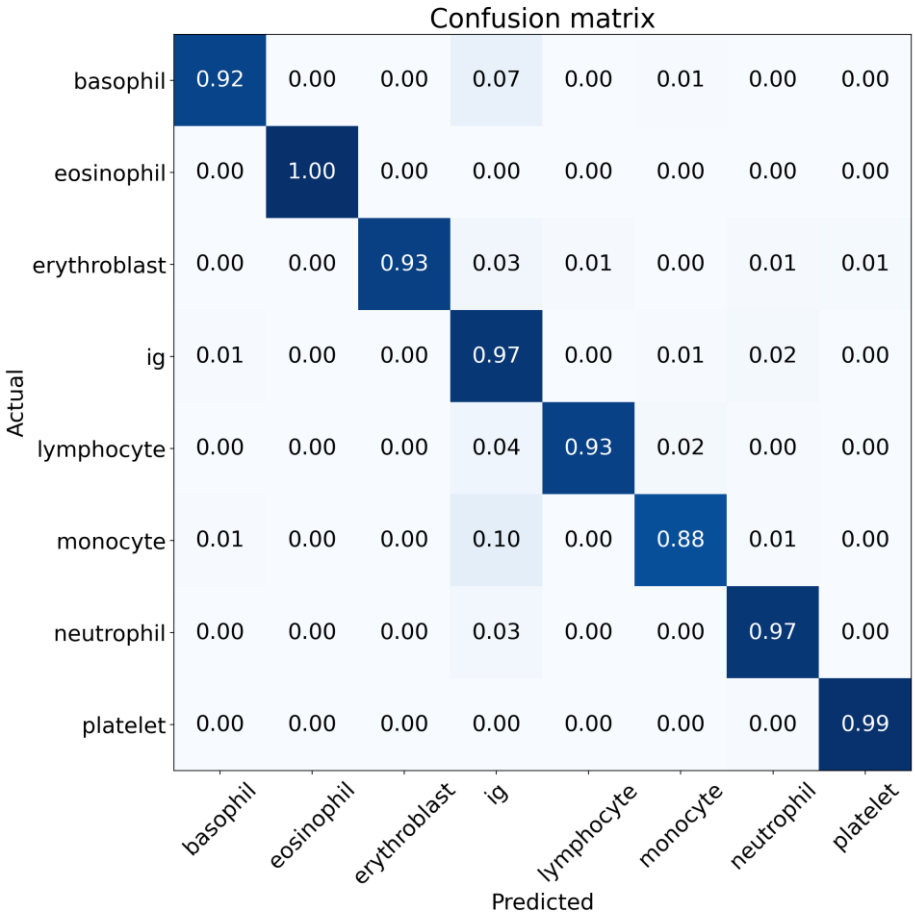

Figura 34. Matriz de confusión normalizada ViT-Large con MixUp

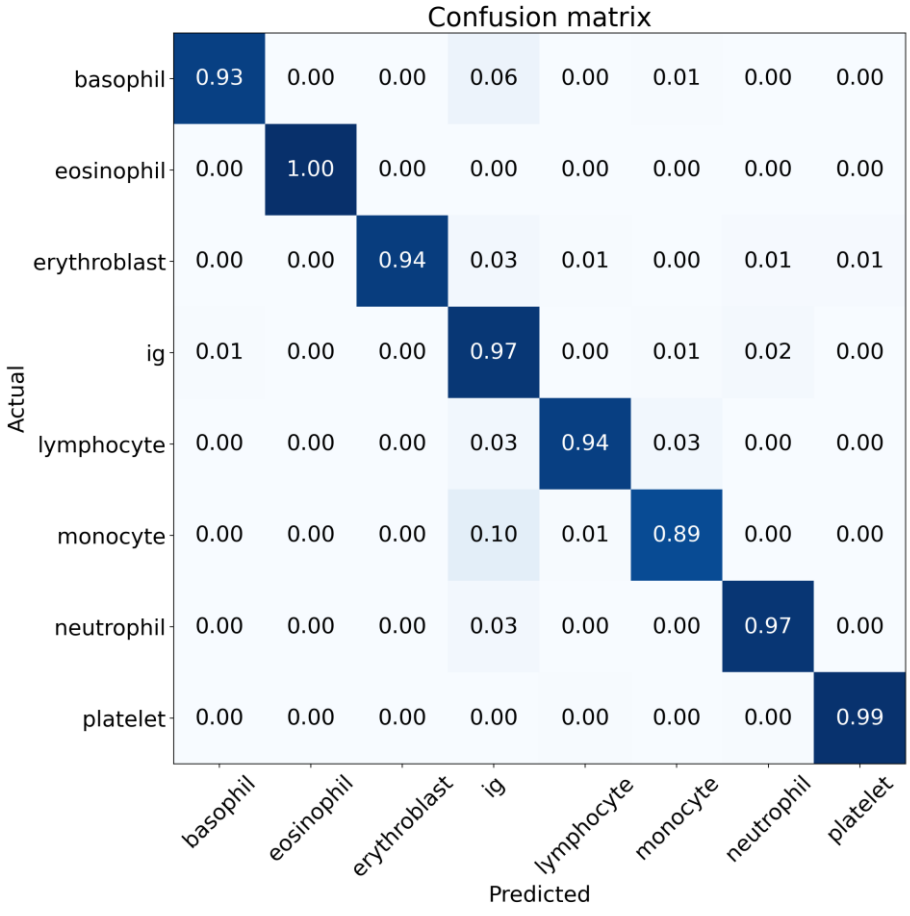

Figura 35. Matriz de confusión normalizada ViT-Large con MixUp y Label Smoothing

En vista de las matrices de confusión normalizadas de ambos modelos podemos confirmar que efectivamente el modelo con *MixUp* y *Label Smoothing* de *ViT-Large* es el modelo que presenta los mejores resultados.

La clase para cuya clasificación menor indice de aciertos tenemos son los monocitos, esto puede deberese a que se trata de una de las categorías que cuenta con una cantidad inferior de imágenes en el conjunto de datos que utilizamos, frente a categorías como: neutrófilos, inmunoglobulinas o eosinofilos que cuentan con la mayor representación dentro del *dataset*.

En el siguiente repositorio: https://github.com/b-ryba/TFM puede acceder al código completo desarrollado durante el proyecto.

4.3. DISCUSIÓN

A continuación, con el fin de poder comparar adecuadamente los mejores modelos de cada tipo se muestra un cuadro resumen con los mejores resultados de cada una de las mejores implementaciones para cada modelo estudiado.

	ResNet18	ResNet34	ViT-Base	ViT-Large
Balanced accuracy	0.98	0.99	0.94	0.95
Accuracy	0.99	0.99	0.95	0.96
Kappa score	0.98	0.98	0.94	0.95
Error rate	0.012	0.01	0.05	0.03

Tabla 2. Tabla resumen de los resultados de los distintos modelos

En general se puede observar como el desempeño de las CNNs sigue siendo mejor al de los modelos ViT, pese a ello cabe destacar que los modelos ViT presentan resultados lo suficientemente cercanos como para considerarlos una alternativa interesante.

Los modelos ViT constituyen una alternativa a tener en cuenta ya que según se defiende en el trabajo de Dosovitskiy et al. (1) su tiempo de cómputo es mucho más reducido que el de las CNN al prescindirse de las convoluciones en sus cálculos. Esto no se pudo comprobar en nuestro proyecto puesto que al trabajar en una plataforma como *Colab* los recursos son limitados e influyen muchos factores externos (cantidad de usuarios haciendo uso del servicio, conexión a internet…) en el tiempo de cómputo, por lo que no es posible comparar ejecuciones llevadas a cabo en momentos distintos.

Por otra parte, podemos ver que los grupos peor clasificados tanto por ViT como por CNN presentan la misma problemática ya mencionada anteriormente: existen algunos tipos celulares con una menor representación en el *dataset,* lo que se traduce en peores resultados en su clasificación y la presencia de más de un tipo celular en una imagen, dificultando su predicción.

En cambio, existen claras diferencias en los valores de *loss* otorgados a los ejemplos peor clasificados, en este caso el desempeño de ViT es mucho mejor puesto que obtiene valores más pequeños. Esto puede deberse a la ventaja que otorgan a este tipo de modelos los mecanismos

de *global attention,* permitiéndoles aprender sobre relaciones de alto nivel en fases mucho más tempranas que las redes convolucionales.

Además, es importante recalcar, que como se menciona en el trabajo de Dosovitskiy et al. (1) , los resultados de ViT solamente consiguen superar los obtenidos por las CNNs cuando se trabaja sobre conjuntos de datos muy superiores en tamaño del que disponemos en este trabajo. Asimismo, los métodos utilizados para afinar los hiperparámetros en los modelos ViT han sido diseñados teniendo en cuenta específicamente las arquitecturas de los modelos basados en CNNs o NLP, por lo que probablemente su desempeño no sea tan bueno al ser aplicado sobre una arquitectura novedosa como lo es ViT.

5. CONCLUSIONES

5.1. CONCLUSIONES

Tras el análisis e interpretación de los resultados obtenidos, se puede afirmar que las técnicas basadas en ViT presenta un gran potencial para la realización de tareas como la clasificación de células normales en sangre periférica de forma automática. Los resultados son fiables y cuentan con una precisión muy próxima a la obtenida por las CNNs que hasta el momento han sido consideradas como el mejor modelo para el desempeño de dicha tarea. A su vez, ambos modelos se asemejan mucho a los resultados logrados por expertos durante el desempeño de dicha clasificación de forma manual, puesto que es importante tener en cuenta la existencia de variabilidad en la clasificación para los casos más complejos entre los diferentes expertos que se enfrentan diariamente a este tipo de labores.

A lo largo del trabajo se ha podido comprobar la efectividad de técnicas como *Mixup* y *Label Smoothing* en la implementación de modelos para el reconocimiento de imágenes. A su vez, también se ha podido confirmar como, tanto en modelos basados en CNNs como en modelos basados en ViT, las arquitecturas con un mayor número de capas presentan un mejor desempeño. En cambio, un mayor número de *epochs* no garantiza mejores resultados.

Por otra parte, la presencia de ejemplos mal etiquetados puede llevar a un mal entrenamiento de los modelos, de ahí la importancia de trabajar con conjuntos de datos alta calidad, como en nuestro caso, en el que hemos podido constatar como la presencia de algún ejemplo mal etiquetado no ha perjudicado gravemente al desempeño del modelo.

El no disponer de un ordenador con una alta capacidad de cómputo no tiene por qué suponer un impedimento para ningún usuario a la hora de probar técnicas de *Deep Learning* gracias a la existencia de plataformas como *Colab* o *Google Cloud*. Pese a ello, sí que es cierto que disponer de un ordenador que nos proporcione un entorno estable de ejecución presenta múltiples ventajas como: no disponer de limitaciones de tiempo y no perder los datos generados durante el entrenamiento cada vez que se cierra la sesión.

Disponer de acceso a un servidor nos permitiría desplegar la versión completa de la interfaz gráfica en la web sin tener que vernos expuestos a las limitaciones de almacenamiento de *GitHub*.

El desarrollo de herramientas de automatización de procesos en medicina permite agilizar las líneas de la investigación de muchos proyectos. Esto es posible gracias a que el tiempo que antes se empleaba en estas tareas ahora puede ser empleado en las siguientes etapas del proyecto. Es por ello, que resulta de especial importancia no tan solo implementar dichas herramientas si no también crear entornos que favorezcan su utilización y que no requieran de conocimientos previos en informática, haciéndolas así, accesibles a todos los ámbitos cuya utilización pueda representar una importante ventaja. Siguiendo esta línea de pensamiento se ha habilitado dos repositorios públicos en *GitHub* que contiene todos los desarrollos implementados en este trabajo.

- https://github.com/b-ryba/app_cell_classifier con la interfaz gráfica interactiva.
- https://github.com/b-ryba/TFM con todos los notebooks de entrenamiento, evaluación de los modelos y con el código de la interfaz gráfica completa.

5.1.1. APRENDIZAJE DURANTE EL DESARROLLO DEL PROYECTO

Desde un punto de vista técnico, durante el desarrollo del proyecto se ha podido aprender a utilizar *fast.ai* y descubrir todas las ventajas que esta API proporciona. Debido a su alto nivel de abstracción permite al usuario centrarse en el propio entrenamiento y mejora de los modelos, además del análisis de sus resultados sin tener que invertir una gran parte del tiempo en la propia implementación del modelo o de *pipelines* para el preprocesado de las imágenes. Asimismo, se ha podido aprender a utilizar herramientas que permiten hacer nuestros trabajos más accesibles a otros usuarios como pueden ser *Voila*, *Binder* o *GitHub*.

Desde un punto de vista crítico, se ha podido comprobar la importancia de un buen sistema de organización del trabajo, que permita cumplir con los plazos acordados y hacer frente a posibles inconvenientes.

5.1.2. REFLEXIÓN CRÍTICA SOBRE EL DESARROLLO DEL PROYECTO

Se ha cumplido con el objetivo principal del proyecto, estudiar y aplicar en la práctica de un modelo basado en *Vision Transformer* (ViT) para el reconocimiento automático de células normales de sangre periférica. Además, se han realizado diferentes implementaciones de un mismo modelo con el fin de encontrar el que ofrezca los resultados más prometedores y próximos a los de las CNNs. En este caso se ha contrastado con el desempeño de uno de los modelos basados en CNNs más conocido: la *ResNet*. Y se ha podido constatar que efectivamente estas nuevas arquitecturas pueden suponer un antes y un después en el campo de la Visión Artificial.

El seguimiento del proyecto se ha realizado acorde a la planificación inicial por lo que no ha sido necesario la introducción de cambios en esta. Además, la metodología seguida ha contado con dos etapas diferenciadas de una duración similar.

- Primera etapa: Adquisición de los conocimientos teóricos previos de los modelos a implementar junto con las librerías y plataformas utilizadas para ello.
- Segunda etapa: Desarrollo práctico de los modelos con sus correspondientes pruebas y una sencilla interfaz gráfica.

Es importante incidir en el reparto equitativo de tiempo entre estas dos etapas, puesto que una formación teórica adecuada permite desarrollar una visión global de los elementos que conforman el proyecto y de los recursos disponibles para su implementación. Con esta metodología se agiliza la resolución de los posibles problemas que puedan aparecer durante la etapa práctica y garantizar el éxito del proyecto.

En este caso las mayores dificultades ante las que nos hemos encontrado durante el desarrollo del proyecto es trabajar con recursos como *Colab* que presentan limitaciones para su uso continuado. Además, otro problema importante ha sido la imposibilidad de integrar *Voila* en el entorno de *Colab*, por lo que ha sido necesario desarrollar también un entorno virtual en Python con los paquetes necesarios en local para poder trabajar en el desarrollo de la Interfaz gráfica.

5.2. LÍNEAS DE FUTURO

La posibilidad de ampliar el conjunto de datos, especialmente de aquellos casos que presentan mayores dificultades en su clasificación podría favorecer un mejor desempeño de la red. Además, también se podrían añadir imágenes que contenga células no normales, es decir, células cuyo tamaño/forma se ve modificado por una enfermedad para convertir la red en una posible herramienta de diagnóstico.

En las posibles mejoras a implementar en la red podría ser interesante tratar de modificar ViT mediante una técnica de *tokens-to-tokens* (28) haciéndolo así más efectivo sobre el conjunto sobre el que ha sido preentrenado y, por consiguiente, probablemente mejorando también los resultados de nuestro entrenamiento.

Otro planteamiento interesante podría ser implementar la red como un sistema de multietiqueta presentándole en una misma imagen diferentes tipos celulares a la vez para ver si ello se puede ayudar en el aprendizaje de los casos más similares entre sí o en aquellos en los que al aparecer diferentes tipos celulares en la misma imagen la red no puede reconocerlos correctamente.

Por otra parte, se podría mejorar la interfaz gráfica introduciendo la opción de cargar más de una imagen a la vez y realizar la clasificación correspondiente obteniendo como salida un archivo Excel con el nombre de cada imagen con su correspondiente clasificación y probabilidad. Además de implementar una página web con un diseño más elaborado, haciéndola más atractiva para el usuario.

Por último, la posibilidad de utilizar un equipo con unas características adecuadas para llevar a cabo los procesamientos que requieren las técnicas de *Deep Learning* nos permitiría disponer de

un entorno estable con el que poder comprobar las diferencias de tiempo existentes entre los entrenamientos de los distintos modelos.

6. GLOSARIO

ViT – *Vision Transformer*

CNN – *Convolutional Neuronal Network*

NPL – *Natural Language Processing*

LSTM – *Long short term*

GRU – *Gated recurrent unit*

API – *Application Programming Interface*

GPU – *Graphics Processing Unit*

Data augmentation – Técnicas utilizadas en Inteligencia Artificial para aumentar el conjunto de datos de entrenamiento, a partir de los datos de los que se dispone de tal forma que parezcan datos diferentes, pero sin cambiar las características del *dataset.*

Presizing – Método seguido para realizar el proceso de *data augmentation* minimizando la destrucción de datos y manteniendo el buen rendimiento (26) .

Transfer learning – Utilización de una red previamente entrenada para un nuevo propósito que para el que fue inicialmente concebida.

Epoch – Cada uno de los ciclos de pasada por el conjunto de entrenamiento.

Learning rate – Parámetro de entrenamiento de la red que determina por cuánto es multiplicado el gradiente. Generalmente se trata de un número comprendido entre 0.001 y 0.1

Unfreezing gradual - *Cambio* de los pesos de algunas de las capas de una red previamente entrenada de forma gradual.

7. BIBLIOGRAFÍA

1. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020;1–21.
2. O'Shea K, Nash R. An Introduction to Convolutional Neural Networks. 2015;1–11.
3. Boron, Walter F; Boulpaep EL. Medical Physiology 3rd Edition. J Chem Inf Model. 2019;53(9):1689–99.
4. Orkin SH, Zon LI. Hematopoiesis: An Evolving Paradigm for Stem Cell Biology. Cell. 2008;132(4):631–44.
5. Hoffman M, Iii DMM, Riddel JP, Aouizerat BE, Miaskowski C, Lillicrap DP, et al. A Cell-based Model of Hemostasis " For Internal Educational Purposes Only . Not for Dissemination ." " For Internal Educational Purposes Only . Not for Dissemination ." Arterioscler Thromb Vasc Biol. 2003;24(3):797–811.
6. Mohapatra S, Samanta SS, Patra D, Satpathi S. Fuzzy based blood image segmentation for automated leukemia detection. 2011 Int Conf Devices Commun ICDeCom 2011 - Proc. 2011;
7. Vijayalakshmi A, Rajesh Kanna B. Deep learning approach to detect malaria from microscopic images. Multimed Tools Appl. 2020;79(21–22):15297–317.
8. Razzak MI, Naz S, Zaib A. Deep learning for medical image processing: Overview, challenges and the future. Lect Notes Comput Vis Biomech. 2018;26:323–50.
9. Tiwari P, Qian J, Li Q, Wang B, Gupta D, Khanna A, et al. Detection of subtype blood cells using deep learning q. 2018;
10. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 3rd Int Conf Learn Represent ICLR 2015 - Conf Track Proc. 2015;1–14.
11. Noble WS. What is a support vector machine? Nat Biotechnol. 2006;24(12):1565–7.
12. Kassani SH, Kassani PH, Wesolowski MJ, Schneider KA, Deters R. A hybrid deep learning architecture for leukemic B-lymphoblast classification. arXiv. 2019;271–6.
13. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv. 2017;
14. CS231n Convolutional Neural Networks for Visual Recognition [Internet]. [cited 2021 Apr 19]. Available from: https://cs231n.github.io/convolutional-networks/
15. Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning. 2016;1–31.
16. Vision Transformer (ViT): Tutorial + Baseline | Kaggle [Internet]. [cited 2021 May 17]. Available from: https://www.kaggle.com/abhinand05/vision-transformer-vit-tutorial-baseline
17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Adv Neural Inf Process Syst. 2017;2017-Decem(Nips):5999–6009.
18. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. EMNLP 2014 - 2014 Conf Empir Methods Nat Lang Process Proc Conf. 2014;1724–34.
19. Are You Ready for Vision Transformer (ViT)? | by Yoshiyuki Igarashi | Towards Data Science [Internet]. [cited 2021 Apr 19]. Available from: https://towardsdatascience.com/are-you-ready-for-vision-transformer-vit-c9e11862c539
20. Acevedo A, Merino A, Alférez S, Molina Á, Boldú L, Rodellar J. A dataset of microscopic

peripheral blood cell images for development of automatic recognition systems. Data Br. 2020;30:105474.

21. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2016;2016-Decem:770–8.
22. Boehmke B, Greenwell B. Hands-On Machine Learning with R. Hands-On Machine Learning with R. 2019.
23. Jia Deng, Wei Dong, Socher R, Li-Jia Li, Kai Li, Li Fei-Fei. ImageNet: A large-scale hierarchical image database. 2009;248–55.
24. Kolesnikov A, Beyer L, Zhai X, Puigcerver J, Yung J, Gelly S, et al. Big Transfer (BiT): General Visual Representation Learning. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 2020;12350 LNCS:491–507.
25. Zhang H, Cisse M, Dauphin YN, Lopez-Paz D. Mixup: Beyond empirical risk minimization. arXiv. 2017;1–13.
26. Howard, Jeremy; Gugger S. Deep Learning for Coders with fastai. 2020;
27. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2016;2016-Decem:2818–26.
28. Yuan L, Chen Y, Wang T, Yu W, Shi Y, Jiang Z, et al. Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet. 2021;