PROYECTO_INFORME_FINAL

Luis Mateo Hincapié Martínez
1038417207

Clasificación de tipos de células sanguíneas periféricas a partir de imágenes

El diagnóstico de enfermedades basadas en la sangre a menudo implica la identificación y caracterización de muestras de sangre de pacientes. Los métodos automatizados para detectar y clasificar los subtipos de células sanguíneas tienen importantes aplicaciones médicas.

La propuesta es usar un conjunto de datos de ocho tipos de células de sangre periférica para clasificar nuevas muestras en el respectivo grupo de células, esto mediante el uso de redes neuronales.

Conjunto de datos

Se obtuvo un conjunto de datos de 17092 imágenes de ocho clases de células sanguíneas periféricas normales utilizando el analizador CellaVision DM96. Posteriormente un grupo de patólogos clasificaron todas las imágenes en los siguientes tipos: neutrófilos, eosinófilos, basófilos, linfocitos, monocitos, granulocitos inmaduros (mielocitos, metamielocitos y promielocitos), eritroblastos y plaquetas.

El conjunto de datos se distribuye en 8 carpetas de la siguiente manera:

CELL TYPE	TOTAL OF IMAGES BY TYPE
neutrophils	3329
eosinophils	3117
basophils	1218
lymphocytes	1214
monocytes	1420
immature granulocytes (metamyelocytes, myelocytes and promyelocytes)	2895
erythroblasts	1551
platelets (thrombocytes)	2348
Total	17,092

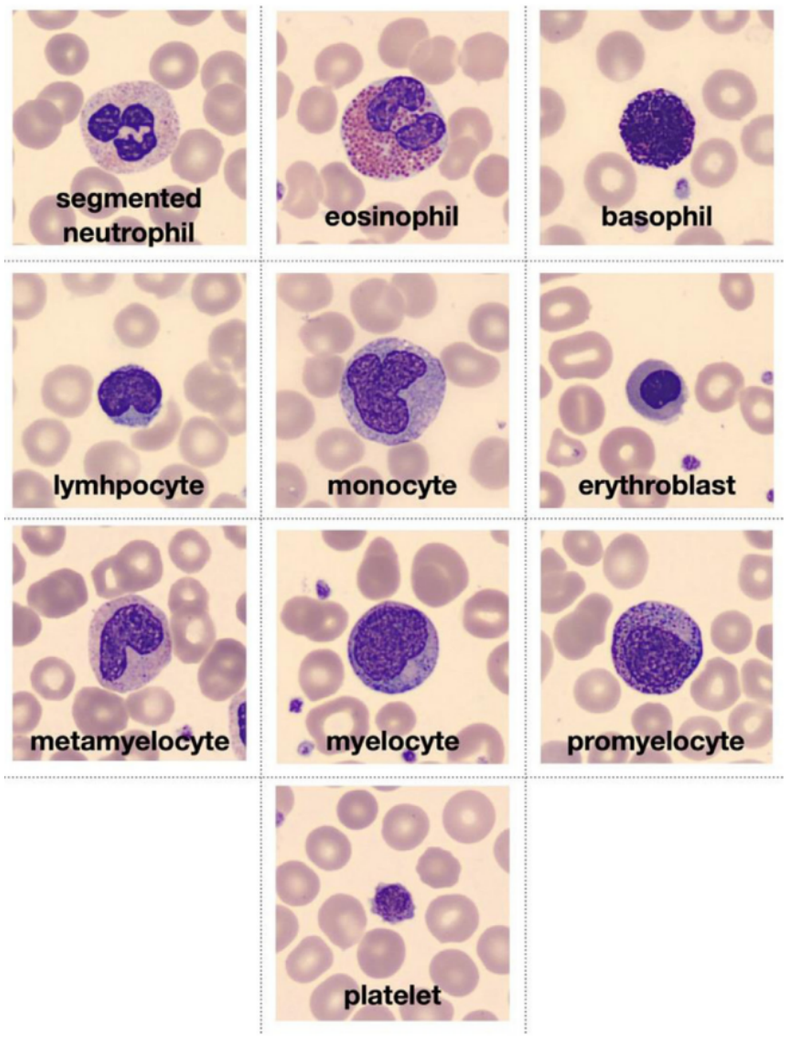


Figura 1. Ejemplo de los diferentes tipos de células.

El conjunto de datos que se encuentra disponible en https://data.mendeley.com/datasets/snkd93bnjr/1 fue compartido por el Hospital Clínico y Provincial de Barcelona desde el 8 de abril de 2020.

Métricas de desempeño

La métrica principal que se desea optimizar será la precisión (*accuracy*) global del modelo, que se analizará en conjunto a la gráfica de curva ROC. En caso de obtener un puntaje considerable de *accuray* con determinado modelo se usará el puntaje F1 o la matriz de confusión para un análisis más profundo.

En este caso no se está determinando una patología como tal, lo que se quiere es clasificar tipos de células, para luego ser contadas y determinar posibles patologías. Existen dilemas éticos y morales que hacen difícil dar un estimado de cual sería la precisión adecuada, por lo que la elección se realizó de forma más o menos arbitraria.

Para el problema que se abordara en este proyecto se buscará un *accuracy* del 95% y un F1 lo más grande posible, partiendo del hecho que tanto la reducción como el aumento en el número de células puede ocasionar diferentes patologías, por lo que no podemos “sesgar” el modelo para evitar que no descarte pacientes enfermos así se equivoque con los sanos como se hace en otras circunstancias, ya que el problema del proyecto no es de este tipo.

Criterio de desempeño deseable en producción

En el ámbito medico se tienen ciertas tazas de errores que ya han sido ampliamente estudiadas y puestas a disposición de las entidades responsables de asegurar la calidad de mediciones cuantitativas en los laboratorios clínicos. Incluso cada grupo de células clasificadas tiene un porcentaje de tolerancia de error diferente. Actualmente desconozco los valores de cada uno de estos porcentajes, pero podría dar una

estimación de que si el modelo logra permanecer dentro del rango de estos porcentajes es viable que salga a producción.

Exploración y preprocesado

Una vez hemos re conocida la base de datos con la que trabajaremos debemos revisar y prepararla previo a su uso en el entrenamiento del modelo.

Nos vamos a encontrar varios tareas propias de preprocesado de datos, en este caso se definirán de una forma conveniente y no tradicional puesto que la base de datos se compone únicamente de imágenes.

> 01 - test_model

Carga de datos

Lo primero es descargar la base de datos desde el repositorio de Mendeley.

```
!curl -O https://md-datasets-cache-zipfiles-prod.s3.eu-west-1.amazonaws.com/snkd93bnjr-1.zip
```

Se debe descomprimir el archivo .zip hasta llegar a la carpeta raíz donde se encuentran las 8 carpetas correspondientes alas categorías.

```
../PBC_dataset_normal_DIB/
    ├── basophil/
    ├── erythroblast/
    ├── lymphocyte/
    ├── neutrophil/
    ├── eosinophil/
    ├── ig/
    ├── monocyte/
    └── platelet/
```

Filtrar imágenes corruptas

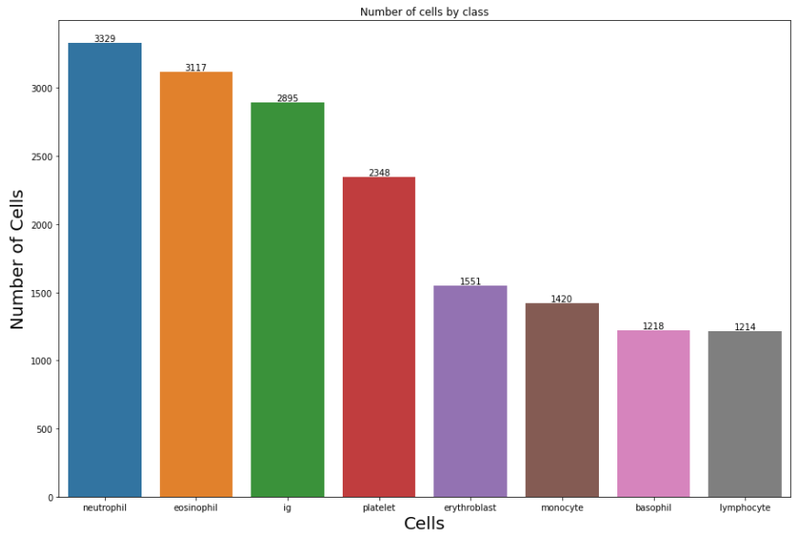
Cuando se trabaja con una gran cantidad de datos de imágenes del mundo real, las imágenes corruptas son comunes. Filtramos las imágenes mal codificadas que no tengan la cadena "JFIF" en su cabecera.

En nuestro caso solo 1 imagen fue removida del dataset

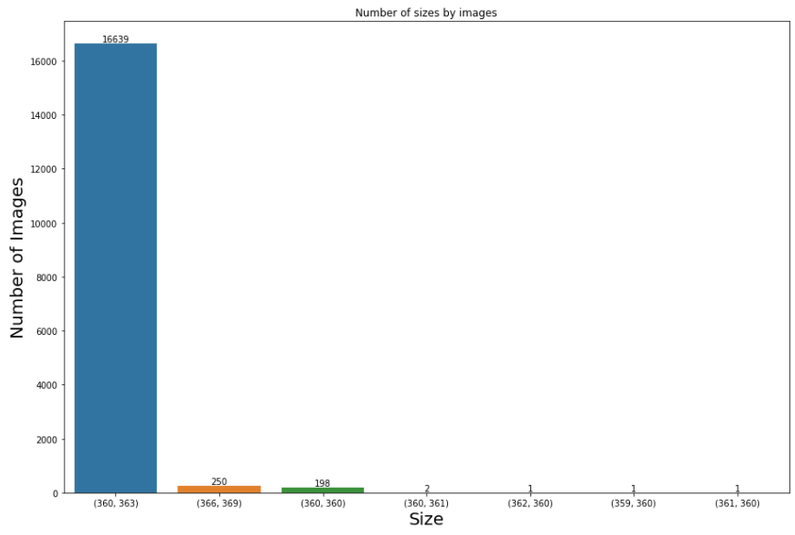
Exploración de los datos

Para revisar que datos tenemos y como se componen realizaremos un recuento de imágenes por categoría y las dimensiones de las imágenes.

```
Number of pictures: 17092
Number of different labels: 8
```



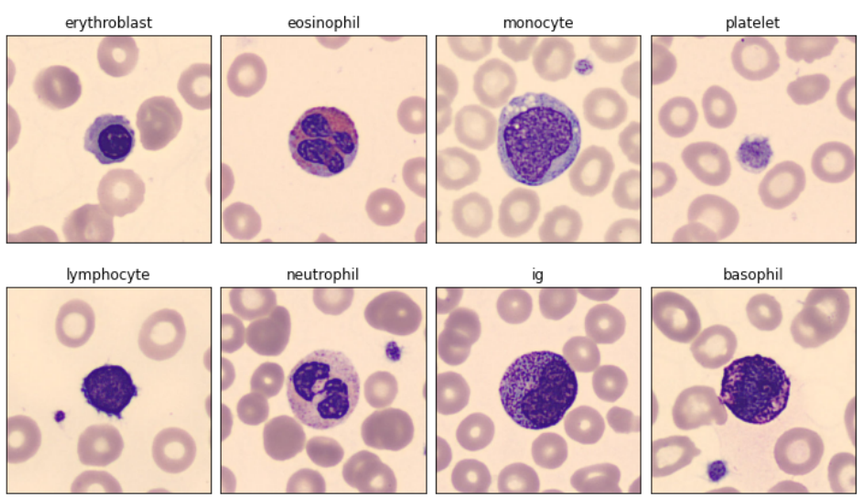
Del total de imágenes la categoría de *neutrophil* es la que mas imágenes tiene, mientras que la categoría de *lymphocyte* es la que menos tiene. Esto podría ser un problema pues lo ideal seria tener una cantidad homogénea de imágenes por cada categoría.



En cuanto las dimensiones de las imágenes, la gran mayoría tiene dimensiones de largo360 x 363 pixeles. Lo cual es ideal puesto que las imágenes las dimensiones de las imágenes no varia casi nada.

Visualización de los datos

Aquí están el ejemplo de cada imagen por categoría. Podemos darnos una idea de la aparencia y características visuales de cada categoría.



Generación del `Dataset`

Creamos primero un dataset con el 80 % de las imágenes para el entrenamiento, y el 20 % restante quedarán de validación, que a su vez partiremos para obtener un tercer conjunto de datos de test. quedando entonces con 13674 imágenes para entrenamiento, 3528 imágenes para validación y 160 imágenes para pruebas.

```
Firt Dataset partition:
Using 13674 files for training.
Using 3418 files for validation.

Second Dataset partition:
Train dataset: 13674 images.
Validation dataset: 3258 images.
Test dataset: 160 images.
```

Aumento de datos

Cuando no tiene un conjunto de datos de imagen grande, es una buena práctica introducir la diversidad de la muestra mediante la aplicación de transformaciones aleatorias pero realistas a las imágenes de entrenamiento, como volteo horizontal aleatorio o pequeñas rotaciones aleatorias. Esto ayuda a exponer el modelo a diferentes aspectos de los datos de entrenamiento mientras se disminuye sobreajuste.

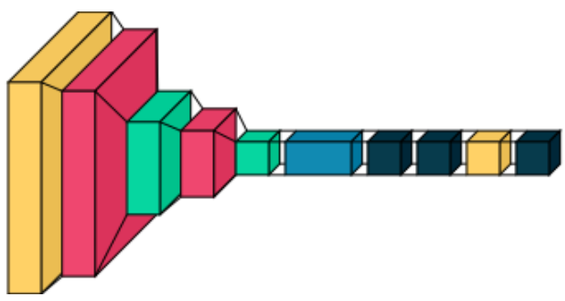
Configurar el conjunto de datos para mejorar el rendimiento

Por último antes de empezar la construcción del modelo, nos aseguramos de usar la captación previa almacenada en búfer para que podamos obtener datos del disco sin tener bloqueos.

Construcción del modelo de prueba

Lo que se quiere con este trabajo es poner a prueba diferentes modelos ya conocidos como LeNet-5, AlexNet, VGG-16, ResNet, ect. Para esto se empezara con el mas sencillo LeNet-5 para probar como es la construcción y la evaluación mediante el uso de la matriz de confusión, resúmenes de precisión, recuperación y F1-score y graficas de precisión y función de perdida de cada época del entrenamiento del modelo.

LeNet-5



La arquitectura LeNet-5 es quizás la arquitectura CNN más conocida. Fue creado por Yann LeCun en 1998 y ampliamente utilizado para el reconocimiento de dígitos escritos (MNIST).

Para nosotros construir este modelo utilizamos *Keras.*

A continuación vemos un resumen del modelo.

```
Model: "LeNet-5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_2 (Rescaling)     (None, 32, 32, 3)         0

 conv2d_4 (Conv2D)           (None, 28, 28, 6)         456
```

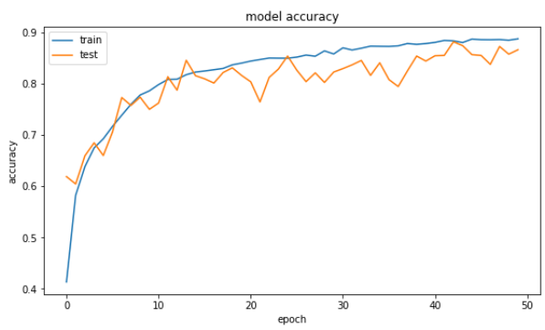
```
 max_pooling2d_4 (MaxPooling  (None, 14, 14, 6)        0
 2D)

 conv2d_5 (Conv2D)           (None, 10, 10, 16)        2416

 max_pooling2d_5 (MaxPooling  (None, 5, 5, 16)         0
 2D)

 flatten_2 (Flatten)         (None, 400)               0

 dense_6 (Dense)             (None, 120)               48120

 dense_7 (Dense)             (None, 84)                10164

 dropout_2 (Dropout)         (None, 84)                0

 dense_8 (Dense)             (None, 8)                 680

=================================================================
Total params: 61,836
Trainable params: 61,836
Non-trainable params: 0
_________________________________________________________________
```

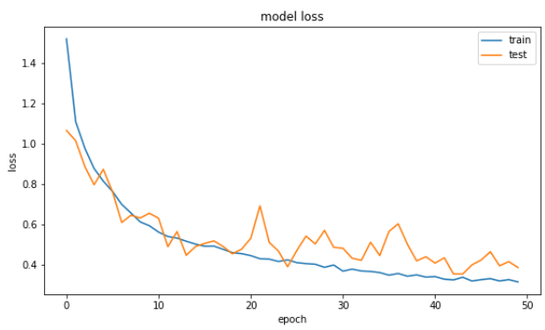
Se entreno el modelo con 50 épocas y se obtienen los siguientes resultados.

```
loss: 0.3872 - accuracy: 0.8646 - val_loss: 0.5702 - val_accuracy: 0.8029
```

Métricas

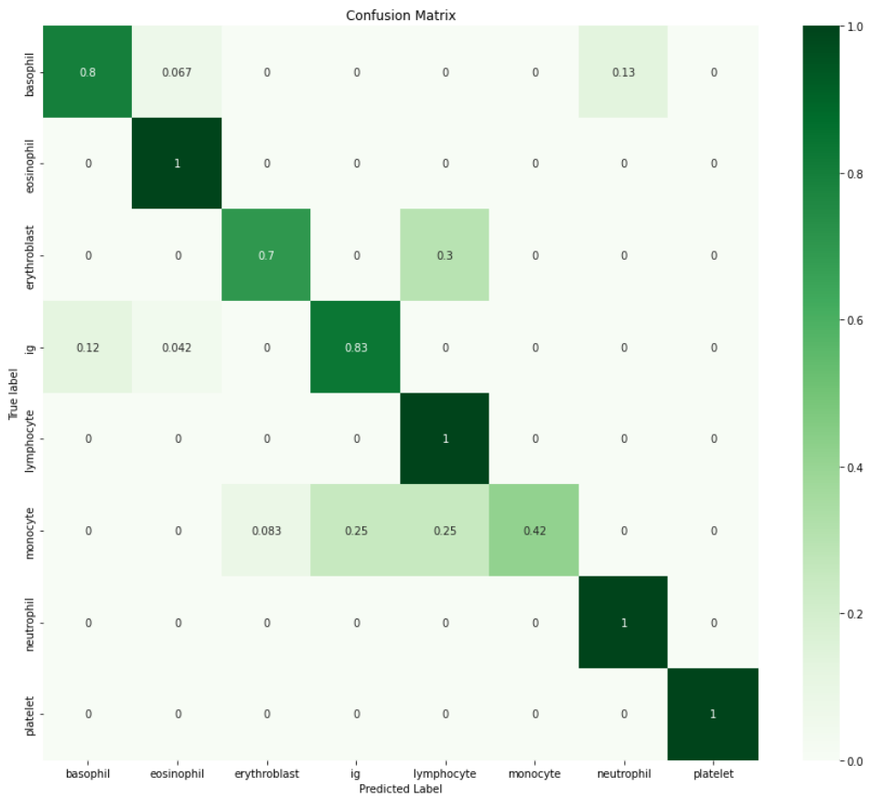
Resumen de precisión y función de perdida del modelo.





Matriz de confusión y reporte de clasificación

Se clasifican nuevos datos con dataset de pruebas y se obtiene la matriz de confusión.



También se obtiene el reporte de clasificación

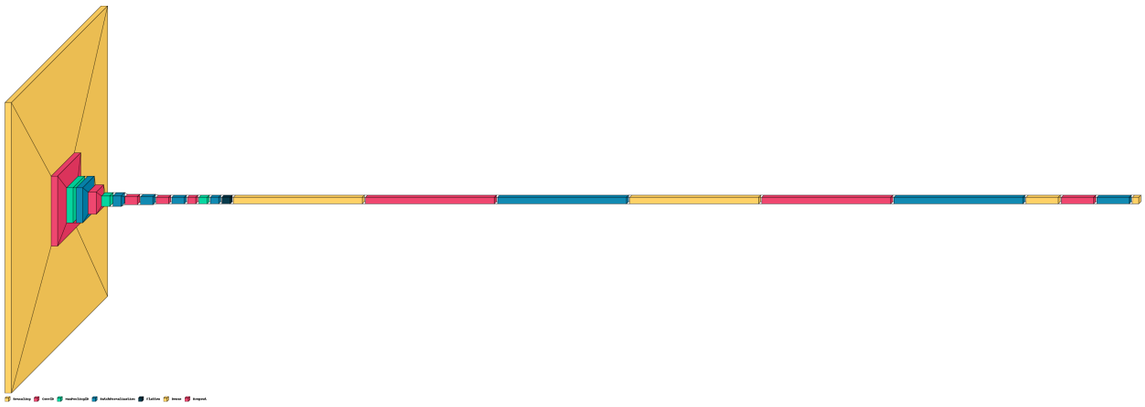
```
Classification Report

              precision    recall  f1-score   support

    basophil       0.80      0.80      0.80        15
  eosinophil       0.93      1.00      0.96        27
erythroblast       0.88      0.70      0.78        10
          ig       0.87      0.83      0.85        24
  lymphocyte       0.62      1.00      0.77        10
    monocyte       1.00      0.42      0.59        12
  neutrophil       0.95      1.00      0.97        35
    platelet       1.00      1.00      1.00        27

    accuracy                           0.89       160
   macro avg       0.88      0.84      0.84       160
weighted avg       0.91      0.89      0.89       160
```

Lo siguiente será construir y entrenar modelos mas avanzados, este primer modelo no se analizara a fondo puesto que es un modelo que no esta diseñado para este tipo de problemas, es bastante sencillo y aunque los primeros resultados no son malos incluso están bastante bien para un primera iteración, no podemos asegurar obtener una mejora significativa debido a las evidentes limitaciones como lo son el tamaño de entra de las imágenes que en este modelo es de 32x32, lo cual le resta bastante margen de mejora, además partiendo de modelos mas actuales se puede asegurar una mejor resultado en menos tiempo. Este primer modelo de prueba `test_model` nos servirá de estructura para entrenar los futuros modelos y analizar los resultados siguiendo un mismo flujo de trabajo evaluarlos de la misma forma.

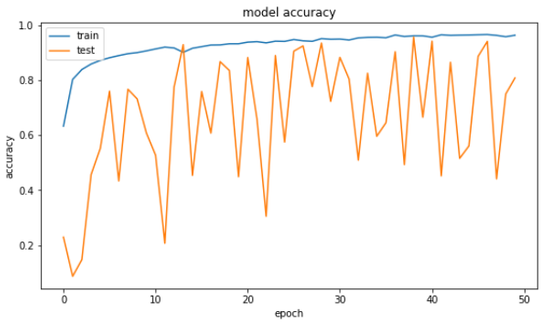
Modelos

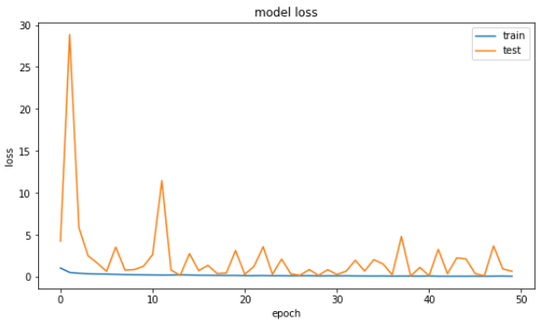
AlexNet

> 02 - AlexNet_model

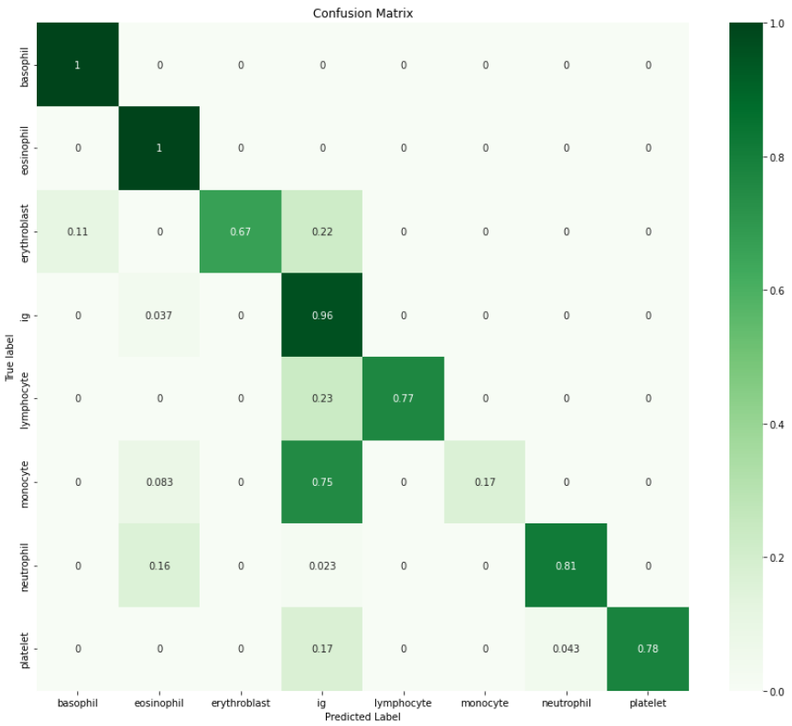


Los resultados del entrenamiento del modelo fueron los siguientes:





Se obtienen fluctuaciones en el conjunto de datos de pruebas.



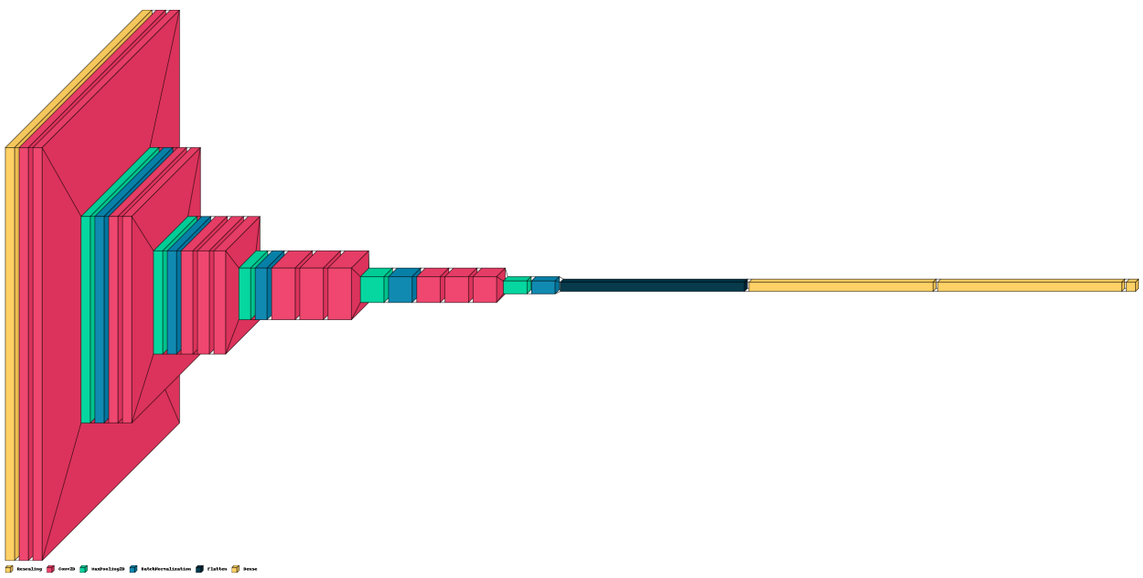

```
Classification Report

              precision    recall  f1-score   support

    basophil       0.89      1.00      0.94         8
  eosinophil       0.74      1.00      0.85        25
erythroblast       1.00      0.67      0.80         9
          ig       0.58      0.96      0.72        27
  lymphocyte       1.00      0.77      0.87        13
    monocyte       1.00      0.17      0.29        12
  neutrophil       0.97      0.81      0.89        43
    platelet       1.00      0.78      0.88        23
```

```
     accuracy                           0.81       160
    macro avg       0.90      0.77      0.78       160
weighted avg       0.87      0.81      0.80       160
```

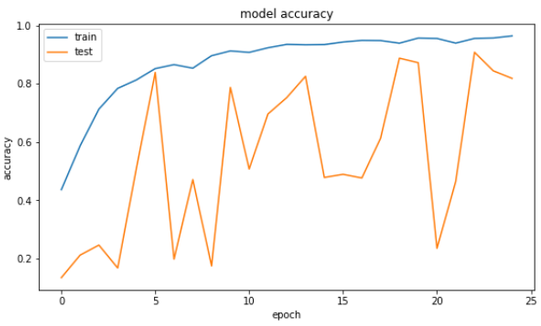
Los resultados de la matriz de confusión y el reporte clasificando datos nuevos son regulares, la categoría de *monocyte* tiene un f1 score muy bajo en comparación con el resto de categorías, se probabrá otro modelo antes de hacer cambios en el dataset o llegar a conclusiones apresuradas, se probaran modelos nuevos con menos épocas para tratar de optimizar el entrenamiento y el tiempo que se toma en obtener un resultado aceptable.

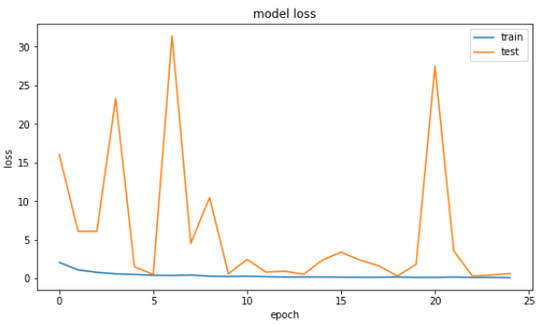
VGG16

> 03 - VGG16_model

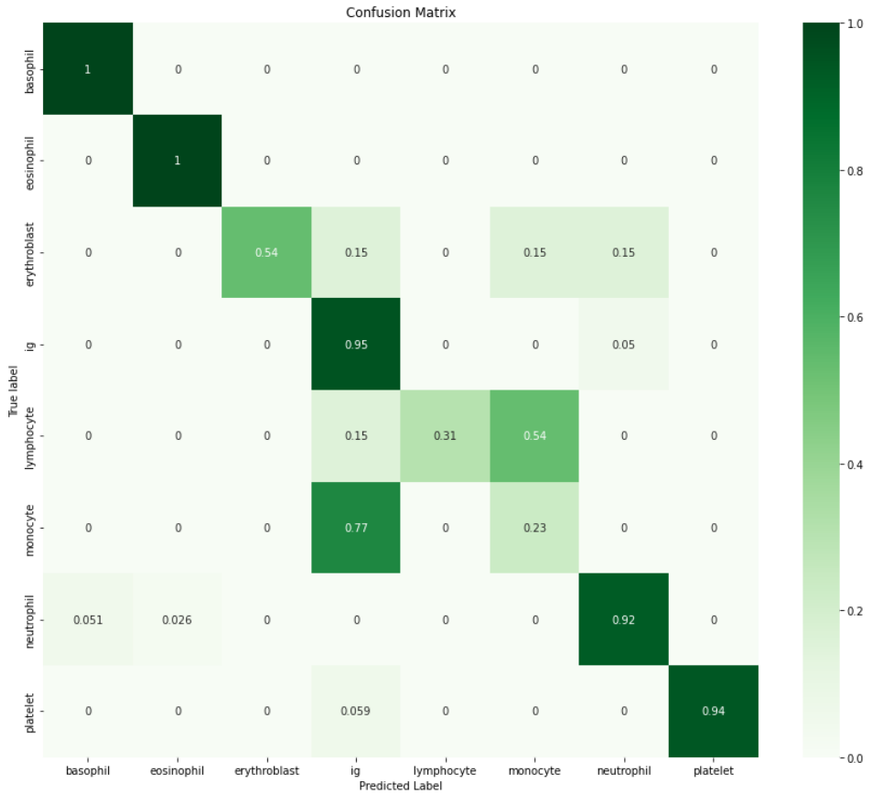


Los resultados del entrenamiento del modelo fueron los siguientes:





después de 25 épocas el modelo muestra un aprendizaje progresivo sobre el conjunto de entrenamiento, sin embargo, no es igual sobre conjunto de test, en el cual fluctúa sobre picos altos y bajos sobre todo en el *accuracy*.

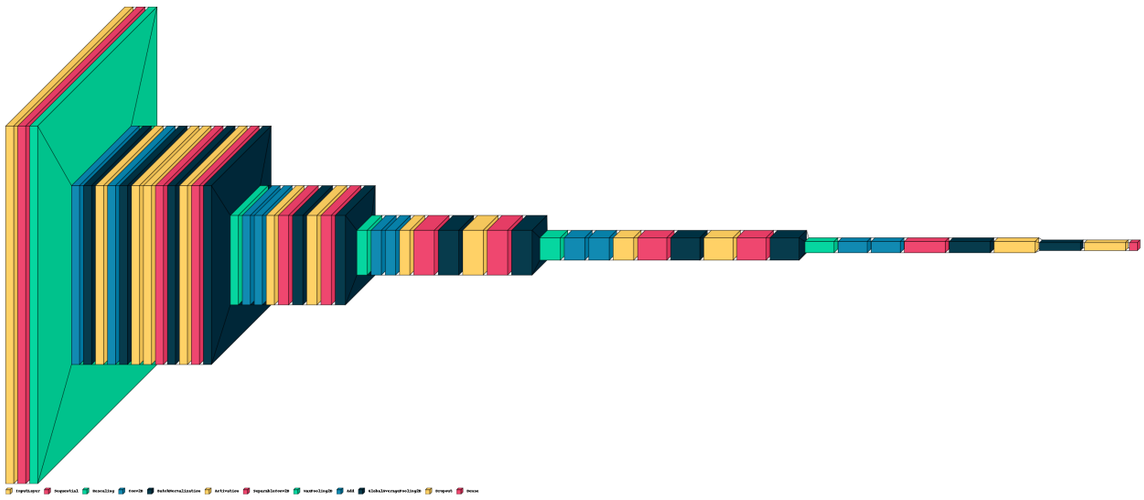



```
Classification Report

              precision    recall  f1-score   support

    basophil       0.89      1.00      0.94        16
  eosinophil       0.97      1.00      0.98        29
erythroblast       1.00      0.54      0.70        13
          ig       0.56      0.95      0.70        20
  lymphocyte       1.00      0.31      0.47        13
    monocyte       0.25      0.23      0.24        13
  neutrophil       0.92      0.92      0.92        39
    platelet       1.00      0.94      0.97        17

    accuracy                           0.81       160
   macro avg       0.82      0.74      0.74       160
weighted avg       0.85      0.81      0.80       160
```

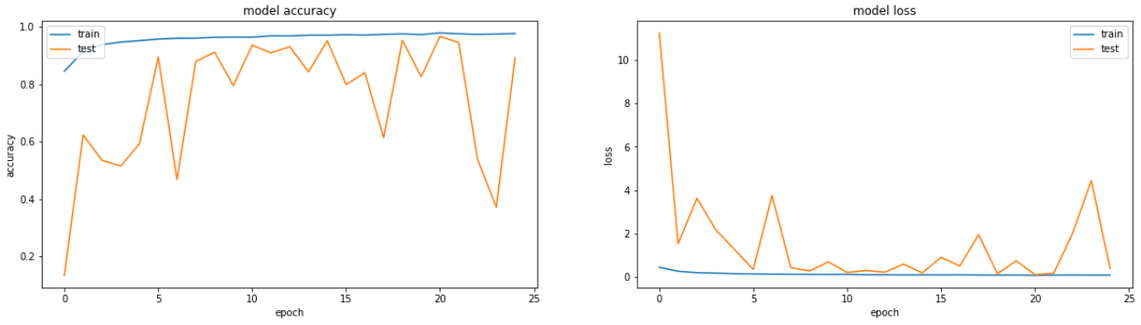
La matriz de confusión y el reporte de clasificación sobre inferencias en un tercer conjunto de datos nos muestra que el problema esta sobre todo en las categorías de *lymphocyte* y *monocyte*, aunque otras tiene comportamientos regulares como *erythroblast* y *ig*. Al tener un *accuracy* alto podemos plantear varios escenarios, que el modelo se esta aprendiendo la base de datos (overfitting), que las imágenes no son suficientes en alguna categorías, que el problema es más complejo de lo que parece y debemos agrupar las categorías problemáticas puesto que a primera vista son muy similares y hasta para un humano puede costar diferenciarlas, de forma que se puedan trabajar con un modelo independiente.

Inception v3

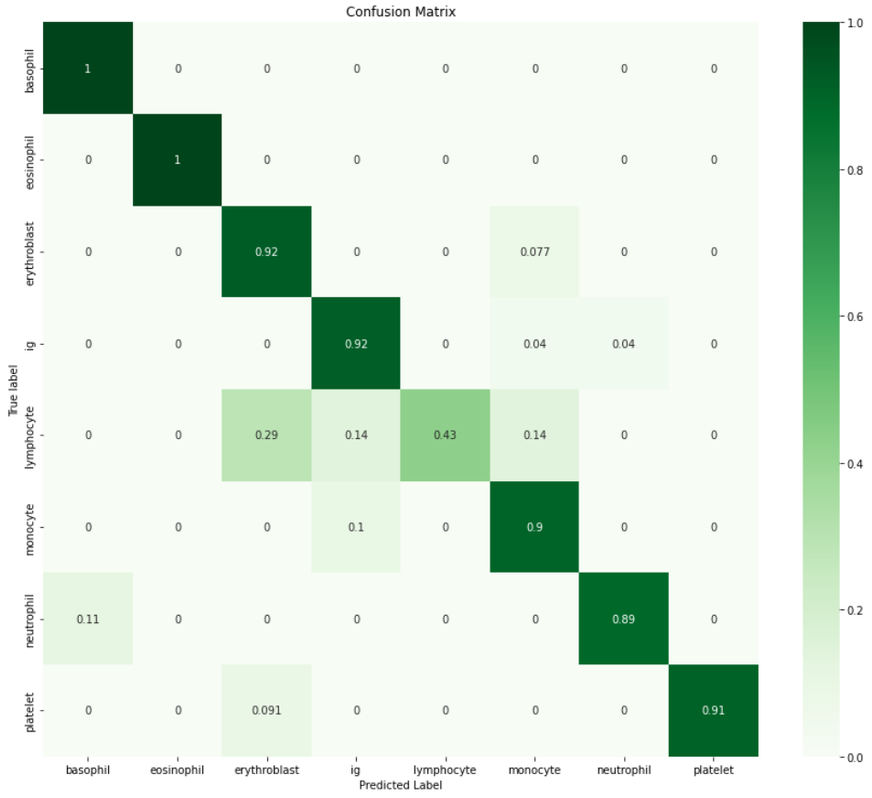
> 04 - Inception_model



Los resultados del entrenamiento del modelo fueron los siguientes:



De forma similar al modelo VGG16 tenemos fluctuaciones marcadas sobre el conjunto de pruebas.

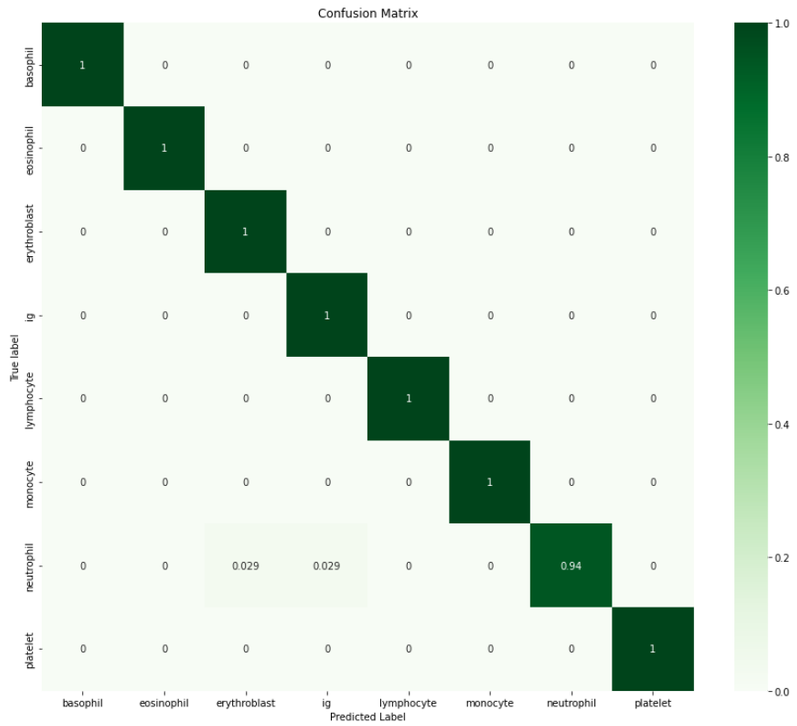



```
Classification Report

              precision    recall  f1-score   support

    basophil       0.89      1.00      0.94        16
  eosinophil       0.97      1.00      0.98        29
```

```
erythroblast       1.00      0.84      0.90        13
          ig       0.76      0.95      0.81        20
  lymphocyte       1.00      0.31      0.43        13
    monocyte       0.95      0.83      0.87        13
  neutrophil       0.92      0.92      0.92        39
    platelet       1.00      0.94      0.97        17

    accuracy                           0.81       160
   macro avg       0.82      0.74      0.74       160
weighted avg       0.85      0.81      0.80       160
```

Tras analizar los resultados podríamos plantear que el modelo es demasiado complejo y el problema es más simple, demasiados parámetros para este problema. En este caso solo tenemos el problema con los *lymphocyte*, categoría con pocas imágenes.

En este caso se tuvieron épocas con accuracy y val accuracy muy altos por lo que se saco el modelo de la mejor ´época y se realizo una inferencia sobre nuevos datos.

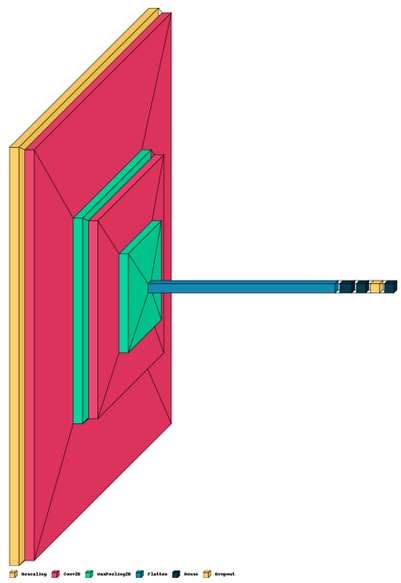



```
Classification Report

              precision    recall  f1-score   support

    basophil       1.00      1.00      1.00        17
  eosinophil       1.00      1.00      1.00        28
erythroblast       0.88      1.00      0.93         7
          ig       0.96      1.00      0.98        26
  lymphocyte       1.00      1.00      1.00        11
    monocyte       1.00      1.00      1.00        14
  neutrophil       1.00      0.94      0.97        35
    platelet       1.00      1.00      1.00        22

    accuracy                           0.99       160
   macro avg       0.98      0.99      0.99       160
weighted avg       0.99      0.99      0.99       160
```

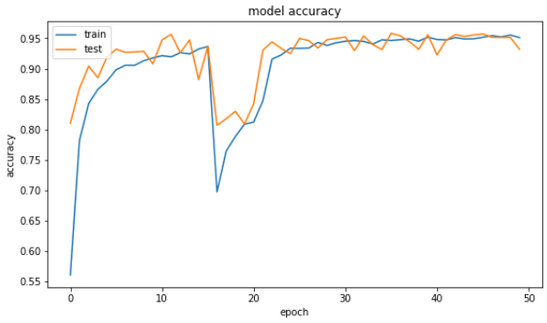
hasta ahora este es el mejor resultado que hemos obtenido en este caso se obtiene un resultado muy alto sobre nuevas clasificaciones con un macro avg del 99%. Tomar este modelo podría se una opción, no seria lo mas recomendable dado que el modelo se entreno con pocas épocas y se podría llegar a un mejor resultado con mas épocas. Sin embargo, dado la naturaleza de las imágenes y que las toma una maquina, probablemente se pueda usar este modelo con una alta fiabilidad en nuevas predicciones.

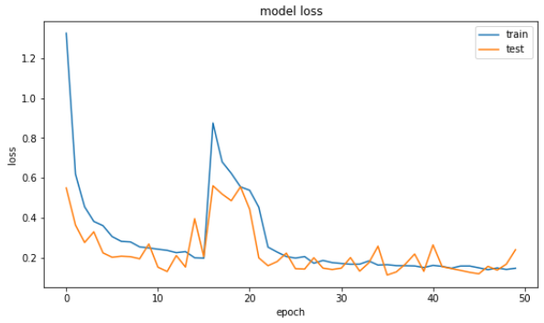
LetNet-5 Modificado

> 05 - LetNet_Mod_model



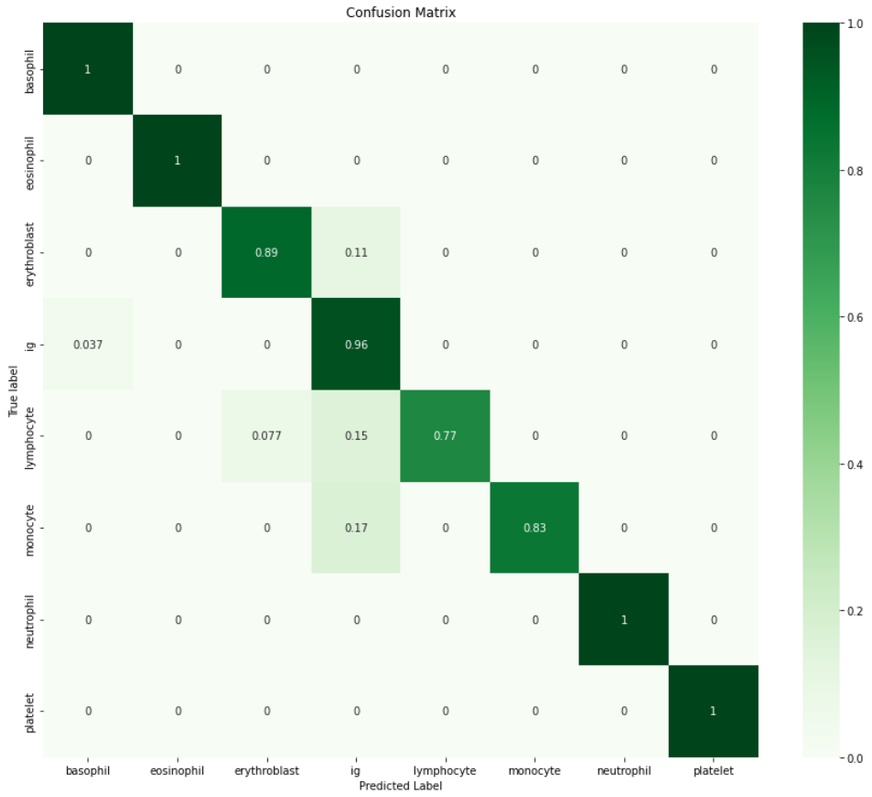
Los resultados del entrenamiento del modelo fueron los siguientes:





Teniendo en cuenta el resultado del modelo de prueba, LetNet-5 original, el cual no fue del todo malo, y analizando los resultados de modelos mas complejos. Se realizo una prueba modificando el modelo LetNet-5. Se cambio las el tamaño de la imagen de entrada a 224x224 y se cambiaron los valores de la cantidad de nodos de las capas intermedias aumentándolos. Esta versión es una prueba para tener un modelo mas simple y ver como se comporta, así tratar de reducir el overfitting.

Si no fuera por el desface las épocas 18 a 20, podría considerarse como el modelo con el entrenamiento más “limpio”, y menos fluctuaciones.




```
Classification Report

              precision    recall  f1-score   support

    basophil       0.80      0.80      0.80        15
  eosinophil       0.93      1.00      0.96        27
erythroblast       0.88      0.70      0.78        10
          ig       0.87      0.83      0.85        24
  lymphocyte       0.62      1.00      0.77        10
    monocyte       1.00      0.42      0.59        12
  neutrophil       0.95      1.00      0.97        35
    platelet       1.00      1.00      1.00        27

    accuracy                           0.89       160
   macro avg       0.88      0.84      0.84       160
weighted avg       0.91      0.89      0.89       160
```

el resultado con predicciones sobre nuevos datos es aceptable, no es mejor que otros modelos vistos anteriormente, sin embargo es el modelo que menos tiempo tomo entrenar y menos problemas de overfitting tuvo, se sigue con problemas sobre la clase *monocyte.*

Conclusiones

Como se pudo ver en los resultados el modelo mas complejo no es necesariamente el mejor modelo, en términos de machine learning y deep learning es complicado encontrar los parámetros adecuados para cada problema. Sin embargo con un poco te tiempo y análisis se pueden obtener resultados aceptables en tiempos ajustados.

La cantidad de muestras es un factor a tener en cuenta a la hora de resolver un problema el desbalance sobre las imágenes en este caso nos pudo haber supuesto un problema, y al se una fuente externa fuera de nuestro control no tenemos mas posibilidades que recortar el conjunto de datos, lo que supone una valiosa perdida de información, o se pueden aumentar artificialmente la cantidad de muestras pero esto tampoco va a suponer una mejora sobre muestras nuevas, mas bien nos ayuda en términos de imágenes de las mismas muestras pero en condiciones distintas con parámetros de toma de muestras distintos.

Referencias

Acevedo, Andrea; Merino, Anna; Alferez, Santiago; Molina, Ángel; Boldú, Laura; Rodellar, José (2020), “A dataset for microscopic peripheral blood cell images for development of automatic recognition systems”, Mendeley Data, V1, doi: 10.17632/snkd93bnjr.1

image_dataset_from_directory, G., R, J., R, J., Catalano, A., Yoshi, H. and karimi, A., 2022. *Get labels from dataset when using tensorflow image_dataset_from_directory*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/64687375/get-labels-from-dataset-when-using-tensorflow-image-dataset-from-directory>.

Amor, E., 2022. *4 CNN Networks Every Machine Learning Engineer Should Know*. [online] TOPBOTS. Available at: <https://www.topbots.com/important-cnn-architectures/> .

Medium. 2022. *Illustrated: 10 CNN Architectures*. [online] Available at: <https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>.

Medium. 2022. *Visualizing Data Augmentations from Keras Image Data Generator*. [online] Available at: <https://gac6.medium.com/visualizing-data-augmentations-from-keras-image-data-generator-44f040aa4c9f> .

Kaggle.com. 2022. *Fruit and Vegetable Classification*. [online] Available at: <https://www.kaggle.com/code/databeru/fruit-and-vegetable-classification>.

Kaggle.com. 2022. 🍇 *Image Classification Using CNN - Fruits* 🍒. [online] Available at: <https://www.kaggle.com/code/rafetcan/image-classification-using-cnn-fruits/notebook>.

Is it possible to split a tensorflow dataset into train, v. and kliachkin, a., 2022. *Is it possible to split a tensorflow dataset into train, validation AND test datasets when using image_dataset_from_directory?*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/71129505/is-it-possible-to-split-a-tensorflow-dataset-into-train-validation-and-test-dat>.

Team, K., 2022. *Keras documentation: Image classification from scratch*. [online] Keras.io. Available at: <https://keras.io/examples/vision/image_classification_from_scratch/> [Accessed 24 August 2022].