UdeA

PROYECTO_ENTREGA2

Luis Mateo Hincapié Martínez
lmateo.hincapie@udea.edu.co
1038417207

Clasificación de tipos de células sanguíneas periféricas a partir de imágenes

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.

> 00 - 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

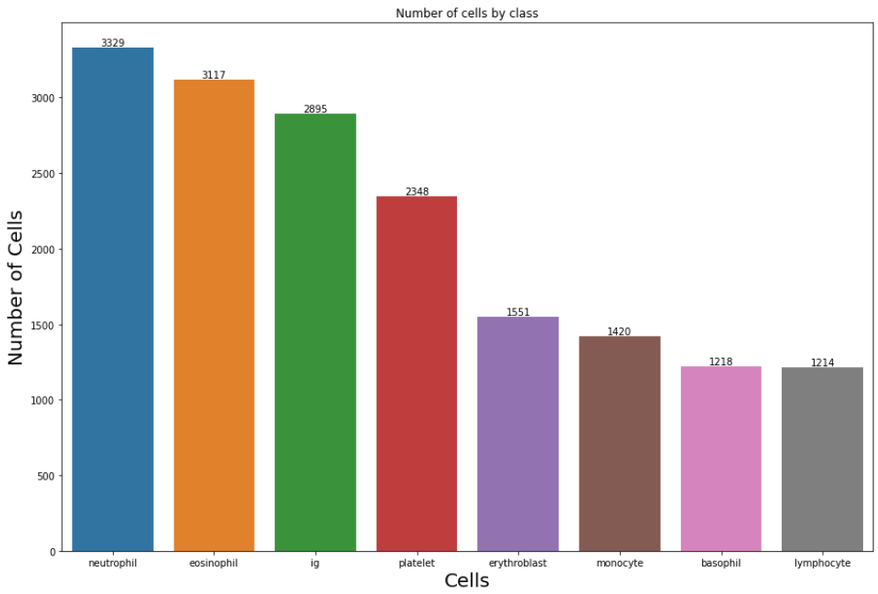
Cuando se trabaja con una gran cantidad de datos de imágenes del mundo real, las imágenes corruptas son comunes. Filtramos las imágenes mal codificadas que no tengan la cadena "JFIF" en su cabecera.

En nuestro caso solo 1 imagen fue removida del dataset

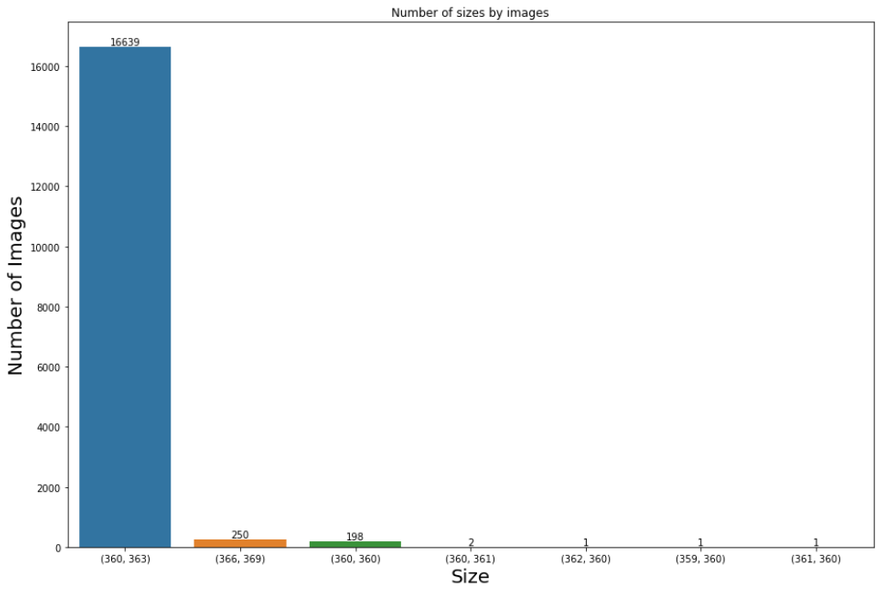
Exploración de los datos

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
```



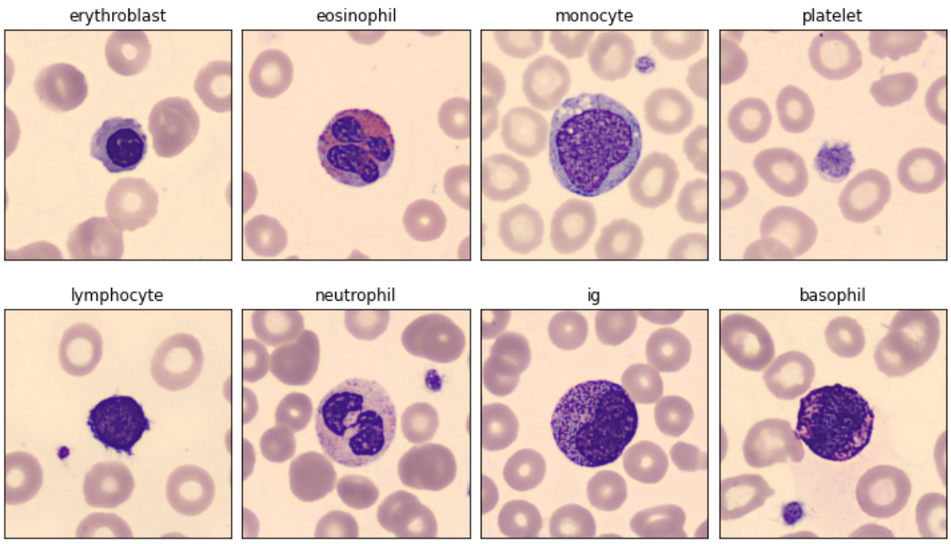
Del total de imágenes la categoría de *neutrophil* es la que mas imágenes tiene, mientras que la categoría de *lymphocyte* es la que menos tiene. Esto podría ser un problema pues lo ideal seria tener una cantidad homogénea de imágenes por cada categoría.



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.



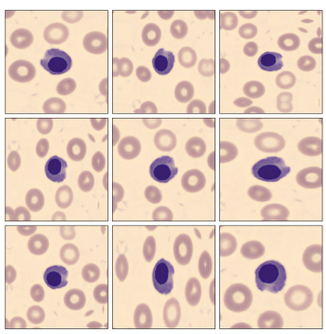
Generación del `Dataset`

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 imagenes 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.



Para nuestro caso, se debe tener en cuenta que la máquina que recolecta estas imágenes es bastante precisa y lo que nos garantiza la calidad de las imágenes, por lo que no habrá problemas por ejemplo con la luminosidad o resolución de estas. Por este motivo solo realizamos transformaciones de darle vuelta a las imágenes, rotarlas y aplicar zoom para aumentar el conjunto de datos.

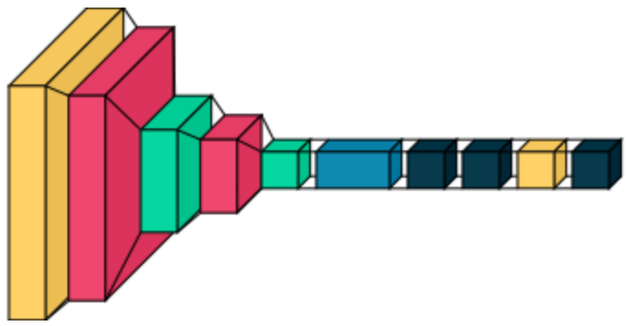
Configurar el conjunto de datos para mejorar el rendimiento

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.

Construir el modelo

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 confusion, resumenes de precisión, recuperación y F1-score y graficas de precisión y función de perdida de cada epoca 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.

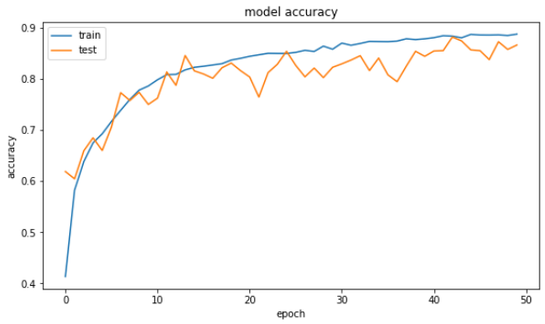
```
Model: "LeNet-5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_2 (Rescaling)     (None, 32, 32, 3)         0

 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
_________________________________________________________________
```

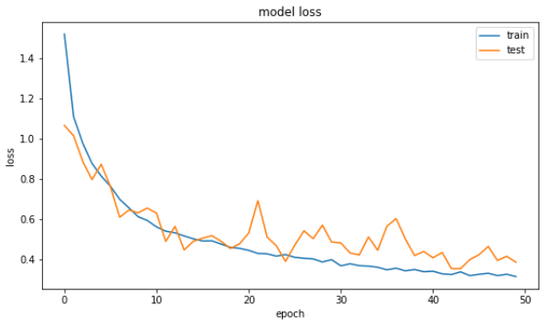
Se entreno el modelo con 50 épocas y se obtienen los siguientes resultados.

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

Metricas

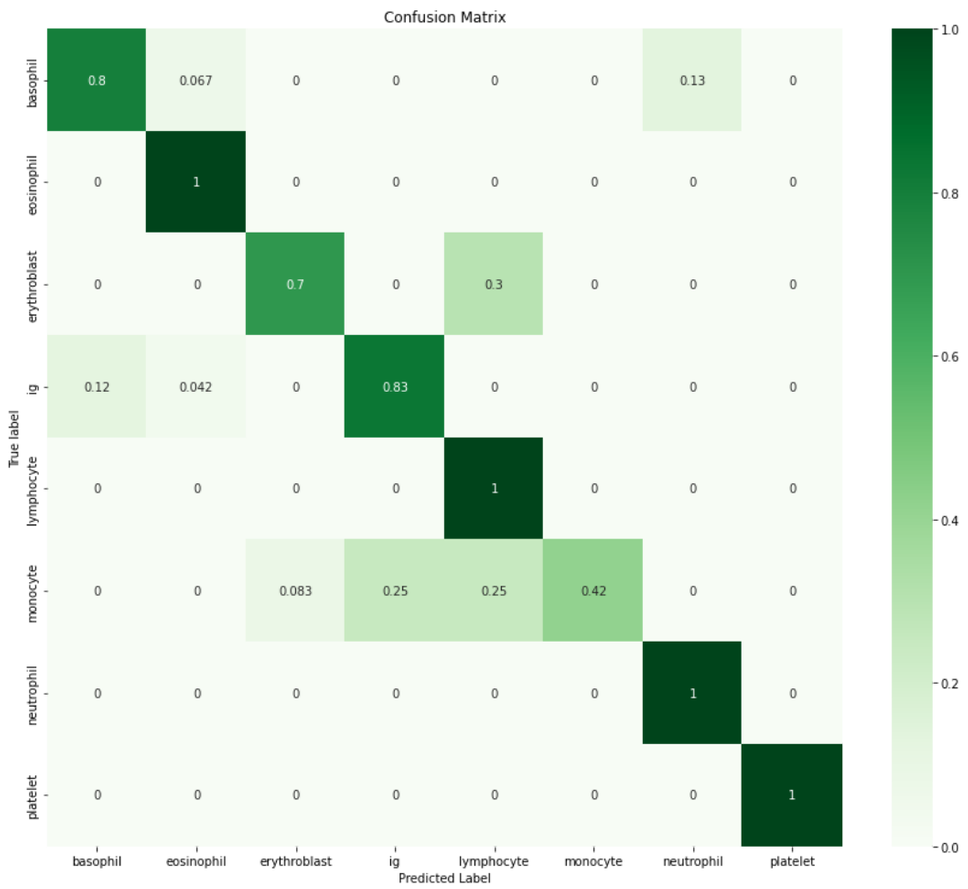
Resumen de precisión y función de perdida del modelo.





Matriz de confusión y reporte de clasificación

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



Tambien 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 sera 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 servira de estructura para entrenar los futuros modelos y analizar los resultados siguiendo un mismo flujo de trabajo evaluarlos de la misma forma.

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].