# Aplicando técnicas de Deep Learning para classificar imagens de células sanguíneas

## Introdução

Este relatório descreve o projeto de desenvolvimento em Python para a classificação de imagens de células sanguíneas em 4 classes distintas. O objetivo do projeto é construir um modelo de aprendizado de máquina capaz de classificar corretamente diferentes tipos de células sanguíneas com base em imagens digitais.

## Metodologia

O projeto foi desenvolvido seguindo as seguintes etapas:

**1. Coleta de Dados**

Um conjunto de dados foi coletado, consistindo de imagens de células sanguíneas rotuladas com as 4 classes: monocytes, lymphocytes, neutrophils, eosinophils. O conjunto de dados foi obtido do MNIST (Modified National Institute of Standards and Technology database) e contém um número suficiente de exemplos para cada classe. Para treino e para teste.

**2. Preparação dos Dados**

As imagens coletadas foram pré-processadas para prepará-las para o treinamento do modelo. As etapas de pré-processamento incluíram:

Redimensionamento das imagens para um tamanho adequado (64x64 pixels), Foram realizados testes com as imagens não redimensionadas em seu tamanho original de 240 por 320 pixels, porém o custo computacional e o tempo de treino estava muito elevado.

Foi realizado dois testes com a preparação dos dados, um inicial com um flatten manual e uma preparação utilizando a biblioteca ImageDataGenerator do keras. A forma do keras foi escolhida por se adaptar melhor com a rede utilizada no tensorflow. O flatten manual serviu bem para testes realizados com a rede da biblioteca de redes neurais do sklearn, porém devido a pouca personalização nos parâmetros da rede ela não se adequou ao problema.

Normalização dos valores dos pixels para garantir que todas as imagens tenham valores de pixel comparáveis.

Divisão do conjunto de dados em conjuntos de treinamento e teste. O conjunto de treinamento foi usado para treinar o modelo, enquanto o conjunto de teste foi usado para avaliar seu desempenho.

**3. Desenvolvimento do Modelo**

Um modelo de classificação foi desenvolvido usando a biblioteca TensorFlow em conjunto com a API Keras. O modelo adotado foi uma rede neural convolucional (CNN), conhecida por sua eficácia na classificação de imagens. A arquitetura da CNN incluiu camadas convolucionais, camadas de pooling e camadas totalmente conectadas.

Foram feitos vários testes com varias tentativas de treino.  
o resultado que mais convergiu em uma solução sem overfit e com uma taxa de acerto adequada foi a seguinte arquitetura de rede neural covolucional:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 62, 62, 32) 896

max\_pooling2d (MaxPooling2D (None, 31, 31, 32) 0

)

conv2d\_1 (Conv2D) (None, 29, 29, 32) 9248

max\_pooling2d\_1 (MaxPooling (None, 14, 14, 32) 0

2D)

conv2d\_2 (Conv2D) (None, 12, 12, 16) 4624

max\_pooling2d\_2 (MaxPooling (None, 6, 6, 16) 0

2D)

flatten (Flatten) (None, 576) 0

dense (Dense) (None, 4) 2308

=================================================================

Total params: 17,076

Trainable params: 17,076

Non-trainable params: 0

nessa arquitetura contém 3 camadas covolucionais intercaladas com camadas de maxpooling, uma camada para o flatten adaptar os dados para a camada sensa e ao fim uma camada densa com 4 neurônios para a saída das 4 classes.

Todos os testes foram realizados com ajustes de parâmetros manuais por motivos de acompanhamentos dos resultados de treino, para uma melhor aprendizagem pessoal sobre o comportamento da curva de gradiente do problema.

**4. Treinamento do Modelo**

O modelo foi treinado usando o conjunto de dados de treinamento preparado anteriormente. Durante o treinamento, a função de perda (loss function) e a métrica definida foi a acurácia. O modelo foi ajustado iterativamente aos dados de treinamento para minimizar a perda e melhorar seu desempenho. Normalmente quando a rede se adequava ao problema 50 Epochs eram mais do que suficientes.

**5. Avaliação do Modelo**

Após o treinamento, o modelo foi avaliado usando o conjunto de dados de teste separado. A acurácia, matriz de confusão e as métricas precision e recall também foram calculadas para medir o desempenho do modelo em relação à classificação correta das células sanguíneas nas 4 classes. Não houve grande desbalanceamento de classificações e o modelo após alguns treinos e testes conseguiu se adequar ao problema.

## Resultados

O modelo treinado alcançou uma acurácia de 82% no conjunto de teste. A matriz de confusão revelou que a classe monocytes foi classificada corretamente em 77% dos casos, a classe lymphocytes, em 99% dos casos, a classe neutrophils, em 75% dos casos e a classe eosinophils em 79% dos casos.

Vale ressaltar que não foram todos os treinos que convergiram na menor função de custo, diversos casos o modelo não se ajustava ou realizava pequenos ajustes até super ajustar os parâmetros. Por se tratar de um problema complexo e utilizar de inicialização aleatória de pesos na rede neural, foi necessário vários testes para conseguir uma modelo bom para o problema.

## Conclusão

O projeto de desenvolvimento em Python para a classificação de imagens de células sanguíneas em 4 classes obteve resultados promissores. O modelo de aprendizado de máquina desenvolvido foi capaz de classificar corretamente células.