Detecção de Células Sanguíneas a partir de Dados de Esfregaço de Sangue Periférico utilizando a Arquitetura YOLOv8

Giovanni O. de Sousa[1], José Elislande B. S. Linhares[1], Marcelo C. Machado[1], Joethe M. Carvalho[1], Amadeu Anderlin Neto[1], Lia Alessandra da S. Martins[1]
[1]Instituto Federal de Educação, Ciência e Tecnologia do Amazonas (IFAM), AM-Brasil
Emails: goliveiradesousa@gmail.com, {breno.linhares,marcelo.chamy,joethe,amadeu.neto,lia.martins}@ifam.edu.br

***Resumo*— Na área da biomédica, o exame de esfregaço de sangue periférico é amplamente utilizado para a detecção de doenças pela contagem de células sanguíneas. O procedimento é tradicionalmente realizado por contagem manual em laboratório, sendo repetitivo e sujeito a erro humano. Recentemente, o algoritmo YOLO vem sendo utilizado para a detecção automática de células sanguíneas, porém o estado da arte carece de pesquisas completas nesse âmbito, utilizando a versão mais atualizada e estável lançada. Neste trabalho, aplica-se o algoritmo YOLOv8 em duas diferentes bases de dados, BCD e BCCD, para a detecção de três classes de células sanguíneas: glóbulos vermelhos, glóbulos brancos e plaquetas. Como resultados obtidos, alcançou-se o desempenho, em mAP50, de até $94,5\%$, superando outros trabalhos similares com versões anteriores da YOLO. Portanto, o algoritmo YOLOv8 atingiu resultados promissores nas duas bases de dados, com uma melhora para detecção de células sanguíneas.**

***Palavras-Chave*— Biomedicina, Hemograma, Detecção de objetos, Detecção de células sanguíneas, YOLOv8.**

***Abstract*— In the biomedical field, peripheral blood smear examination is widely used to detect diseases by counting blood cells. The procedure is traditionally performed by manual counting in the laboratory, being repetitive and subject to human error. Recently, the YOLO algorithm has been used for the automatic detection of blood cells, but the state of the art lacks complete research in this area, using the most updated and stable version released. In this work, the YOLOv8 algorithm is applied in two different databases, BCD and BCCD, for the detection of three classes of blood cells: red blood cells, white blood cells and platelets. As results obtained, a performance, in mAP50, of up to 94.5% was achieved, surpassing other similar works with previous versions of YOLO. Therefore, the YOLOv8 algorithm achieved promising results in both databases, with an improvement in blood cell detection.**

***Keywords*— *Biomedicine*, *Hemogram*, *Object detection*, *Blood cell detection*, *YOLOv8*.**

I. Introdução

O hemograma é um exame laboratorial utilizado para avaliar diferentes aspectos do sangue do paciente a partir da contagem de três tipos de células sanguíneas. Os três tipos de células sanguíneas são: i) glóbulos vermelhos ou hemácias que têm como principal função a entrega de oxigênio e a retirada de gás carbônico dos tecidos; ii) glóbulos brancos ou leucócitos que são partes importantes do sistema imunológico para a defesa do corpo humano; e iii) plaquetas que ajudam na coagulação do sangue para melhor recuperação de ferimentos [1]. Geralmente, a contagem é feita por meio de coloração, por *softwares* específicos ou por contagem manual utilizando um microscópio. A partir de níveis considerados anormais nas quantidades presentes dessas células no sangue, é possível detectar algumas doenças, tornando-se, assim, um exame muito importante para diagnósticos e tratamentos precoces de enfermidades [2].

Recentemente, diversas aplicações de contagem de células sanguíneas utilizando aprendizado profundo surgiram na literatura, demonstrando a viabilidade de diferentes métodos. Nesse âmbito, duas abordagens para detecção são empregadas: algoritmos de dois estágios (ADE) e algoritmos de um único estágio (AUE). A primeira abordagem utiliza uma rede de proposta de regiões para extrair uma região de interesse e depois classificá-la, enquanto que a segunda abordagem considera a localização e classificação como problemas de regressão. Alguns exemplos de ADE incluem as redes R-CNN e *Faster* R-CNN, enquanto que as redes SSD e YOLO (*You Only Look Once*) representam AUE. A diferença entre as abordagens reside na menor acurácia para os AUE, porém são mais rápidos e eficientes [1], [3]. As redes YOLO têm demonstrado uma facilidade de utilização, alto desempenho e bons resultados em várias aplicações, além de inovações entre cada versão disponibilizada da rede [4]–[6]. Muitas dessas versões são customizadas para servir a diversas aplicações, com melhorias no desempenho geral ou específico de um problema. Nesse sentido, é interessante avaliar diferentes versões da rede para comparações em uma determinada solução, como foi utilizado por [2], [3], [7], além de também empregar de melhorias nativas da versão ou outros algoritmos relacionados, podendo ter sistemas que beneficiem à sociedade.

A. Contribuições do artigo

A principal contribuição deste artigo é a avaliação comparativa de sistemas nativos e sem modificações com sistemas aprimorados para a tarefa de detecção de células sanguíneas em modelos YOLO. Justifica-se em decorrência da ausência de pesquisas completas focadas na versão 8 da YOLO em aplicações de detecção das três classes de células sanguíneas, especialmente, utilizando a sua configuração padrão. Para isso, aplicam-se duas diferentes base de dados amplamente utilizadas em trabalhos anteriores para apresentação dos resultados.

II. Trabalhos Relacionados

No trabalho proposto por [8], apresenta-se um sistema para detecção de células sanguíneas utilizando uma versão melhorada do YOLOv5, chamada de AYOLOv5. Como resultados obtidos, foi alcançada uma precisão média de 93*;* 3% para tarefa de detecção de células, utilizando o conjunto de dados *Blood Cell Count and Detection* (BCCD) [9] com o método de *data augmentation* do tipo mosaico.

Na pesquisa conduzida por [1], emprega-se uma variante do modelo de *deep learning* YOLOv5s, chamada YOLOv5s-TRBC, para detecção de células sanguíneas. Para avaliação do desempenho do modelo, obteve-se 93*;* 5% de precisão média no *dataset* BCCD e a métrica GFLOPs (do inglês, *Giga Floating-point Operations Per Second*) foi 6 vezes menor do que a YOLOv5.

No estudo proposto por [7], apresenta-se o sistema SDE-YOLO para detecção de células sanguíneas, a partir de melhorias da rede YOLOv5s. Como resultados obtidos, teve-se uma precisão média de 96% no *dataset* BCCD, considerado um bom desempenho, e levando vantagem em relação a outros modelos similares para detecção em tempo real de células sanguíneas.

De maneira geral, os trabalhos relacionados apresentam algum tipo de modelo baseado em redes YOLOv5 ou YOLOv5s, com diferentes modificações em partes específicas do processo, utilizando o *dataset* BCCD para avaliação de desempenho, a partir de métricas como, por exemplo, a precisão média. Baseado nas experiências anteriores, este trabalho se diferencia em propor a utilização da versão mais recente e estável da arquitetura YOLO, chamada de YOLOv8 [10], [11] para a tarefa de detecção de células sanguíneas, obtendo vantagens de diferentes configurações nativas da versão da rede, empregando os *datasets* BCCD e *Blood Cell Detection* (BCD) [12].

III. Metodologia Proposta

A. Introdução

Nesta seção, apresenta-se a metodologia empregada para a detecção de células sanguíneas a partir de dados de esfregaço de sangue periférico, conforme representado no diagrama em blocos da Figura 1. O sistema proposto é composto por duas principais etapas: treinamento e predição. Cada um dos blocos que compõem estas etapas serão descritos a seguir.

B. Base de Dados

Neste bloco, utilizaram-se duas bases de dados para a detecção de células sanguíneas: o BCD e o BCCD. O BCD *Dataset* é uma base de dados de baixa escala, com 364 imagens coloridas em extensão *.jpg* e resolução de 416×416 pixels, contendo objetos para três diferentes classes: células sanguíneas vermelhas (RBC, do inglês *Red Blood Cell*), células sanguíneas brancas (WBC, do inglês *White Blood Cell*) e plaquetas (*Platelets*). A base possui uma divisão, sendo 70% para treinamento, 10% para validação e 20% para teste, além de incluir as anotações das posições das caixas delimitadoras dos objetos no formato *.txt*. A base BCCD *Dataset* possui 410

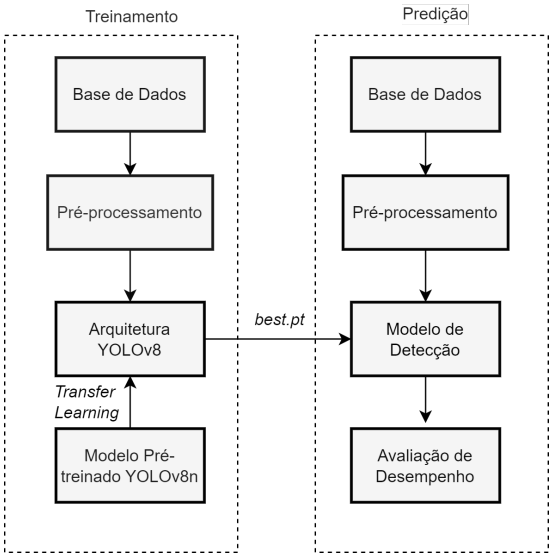

Fig. 1. Diagrama em blocos da metodologia proposta para detecção de células sanguíneas.

imagens coloridas com extensão *.jpeg* e as mesmas classes de objetos da base BCD. Além disso, as dimensões das imagens são de resolução 640×480 pixels. A anotação está em formato *.xml* e não possui divisão entre conjuntos de treinamento, teste e validação. As duas bases serão utilizadas para o treinamento do modelo YOLOv8, além da validação e teste final.

C. Pré-processamento

Neste bloco, realiza-se o pré-processamento da base BCCD, por possuir anotações no formato *.xml*, incompatível com o padrão YOLO. Foi utilizada a biblioteca Pylabel, na versão 0.1.55, para converter automaticamente as anotações em formato VOC (*.xml*) para YOLO (*.txt*). A base foi dividida em 80% para treinamento, 10% para validação e 10% para teste. A base BCD está organizada para um treinamento nas versões YOLOv5 e YOLOv8, não havendo necessidade para um pré-processamento dos dados.

D. Arquitetura YOLOv8

Neste bloco, utiliza-se a arquitetura YOLOv8, conforme apresentado por [6]. Empregam-se módulos complexos como o *C2f* ou gargalo parcial de estágio cruzado com duas convoluções que combina informação contextual com *features* de alto nível para melhorar a acurácia e uma rede CNN CSPDarknet23 modificada para extração de *features* no *backbone*. Além disso, possui camadas de convolução com função de ativação SiLU e *head* totalmente desacoplada do resto para realizar tarefas de regressão e classificação independentemente. A camada de *Spatial Pyramid Pooling Fast* (SPPF) acelera o processamento pelo *pooling* de *features* em mapa com tamanho fixo.

E. Modelos Pré-treinado YOLOv8n e de Detecção

Neste bloco, utiliza-se o modelo pré-treinado YOLOv8n para *transfer learning*, que possui a menor quantidade de

parâmetros (cerca de 3,2 milhões) em relação aos outros modelos (*small*, *medium*, *large*, *extra large*), sendo enquadrado na categoria *nano* [13]. Para treinamento, utilizou-se o valor *train* no parâmetro *mode* (*mode=train*), com 25 épocas, obtendo o modelo treinado nativamente sem *fine tuning* com nome padrão *best.pt*, ou seja, o modelo continua com a nomenclatura padrão do *framework*, para cada uma das duas bases de dados. Além disso, o *framework* também possui aplicação automática no conjunto de dados de validação enquanto o treinamento acontece. Os testes finais foram realizados com o modelo treinado em conjunto de dados específico para teste. Como resultados, obtevem-se diferentes métricas importantes para medir o desempenho.

F. Avaliação de Desempenho

Neste bloco, apresentam-se os resultados obtidos pelo modelo treinado para cada base de dados na etapa de validação e teste. O desempenho para detecção de objetos no YOLOv8 é baseado nas métricas mAP50 e mAP50-95, isto é, na média da precisão média (mAP, do inglês *Mean Average Precision*), considerando uma porcentagem de 50% e entre 50 a 95%, respectivamente. Esta porcentagem indica o valor de *Intersection over Union* (IoU), ou seja, o grau de sobreposição entre a caixa de detecção predita e anotação verdadeira (*ground truth*) [14]. Conforme apresentado na Equação 2, para calcular o mAP, deve-se obter inicialmente a precisão de detecção na imagem (ver Equação 1), em que TP representa a detecção correta da anotação verdadeira e o FP a detecção incorreta de um objeto inexistente. Em seguida, calcula-se a média da precisão de todas as amostras (N) para uma determinada classe (AP) e depois a média incluindo todas as classes.

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2}$$

Na Figura 2, apresenta-se um exemplo de detecção com a marcação de caixas delimitadoras nas células vermelhas, células brancas e plaquetas presentes nas imagens.

IV. Resultados e Discussão

A. Setup

O treinamento, validação e teste do modelo para detecção de células sanguíneas foram realizados no serviço em nuvem Google Colaboratory que permitiu a utilização, na versão gratuita, da GPU Nvidia Tesla T4, com 16 GB GDDR5. O ambiente de desenvolvimento foi integrado ao Google Drive para armazenamento persistente de dados gerados, assim como os *logs* de treinamento, validação e teste. Na Tabela I a seguir, apresenta-se o *setup* utilizado, com as especificações de *hardware* e *software*, contendo informações acerca de velocidades, capacidades de processamento e versões de bibliotecas, por exemplo. O *framework* Ultralytics utilizado possui funções para treinamento da arquitetura YOLO, versão 8, na linguagem Python e a biblioteca PyLabel foi importante para a conversão das anotações do *dataset* BCCD para o formato empregado na YOLO.

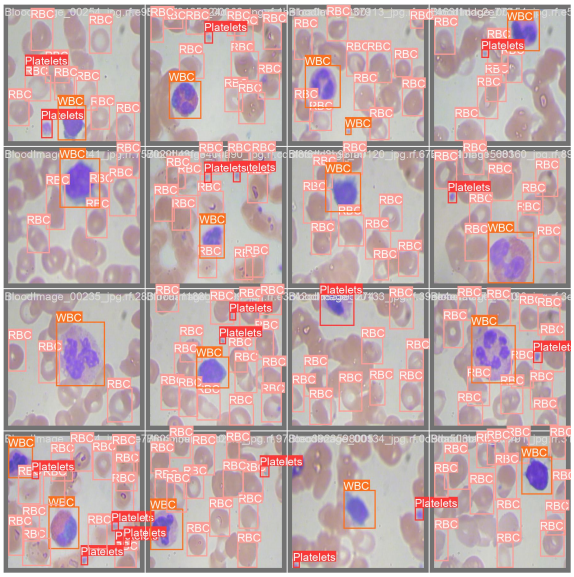

Fig. 2. Amostras de detecção utilizadas na validação do modelo treinado utilizando o *dataset* BCD.

TABELA I

Recursos de hardware e software utilizados nos experimentos.

Recurso	Nome	Especificação/Versão	Observação
Processador (CPU)	Intel Core i7-8565U	4,2 GHz	*Hardware* do servidor
Memória RAM	DDR4	20 GB	*Hardware* do servidor
Placa gráfica (GPU)	Nvidia Tesla T4	16 GB GDDR6	GPU do servidor
Interpretador	Python	3.10.12	-
Ambiente de Desenvolvimento	Google Colaboratory	-	-
Framework	Ultralytics	8.2.2	Treinamento, validação e teste da YOLOv8
Biblioteca	PyLabel	0.1.55	Conversão de anotação

B. Análise de Resultados

Realizado o treinamento, diferentes resultados foram gerados pelo *framework* Ultralytics. Porém, selecionaram-se os mais importantes para verificar o desempenho e robustez do modelo. Nas Tabelas II e III, apresentam-se os resultados da validação do treinamento e teste para os conjuntos BCD e BCCD, respectivamente. É possível verificar o desbalanceamento de classes nos dois *datasets*, com a classe *RBC* possuindo pelo menos 10 vezes mais instâncias do que as demais. Porém, é razoável este desequilíbrio entre classes no domínio do problema, por existir mais glóbulos vermelhos do que glóbulos brancos ou plaquetas. O *precision* e *recall* são métricas que relacionam, respectivamente, a precisão das caixas delimitadoras e a detecção de objetos, enquanto que o mAP50 e mAP50-95 são as precisões médias para um IoU específico. Os valores de mAP50 foram, em média, bons para todas as classes, porém, houve um pior desempenho para a detecção de *RBC*. Os modelos obtiveram resultados safisfatórios para a deteção da classe *WBC*, nas duas métricas, enquanto que a detecção de plaquetas teve um resultado inferior no treinamento e teste para a métrica mAP50-95.

Analisando por *dataset*, ao utilizar o BCD, o modelo obteve bons resultados na validação do treinamento, alcançando uma precisão geral de 83:5%, mas com um decréscimo no teste, com uma precisão geral de 78:2%. Para o BCCD, os resultados foram superiores ao *dataset* BCD, com uma precisão geral de 86:5% para validação do treinamento e 86:4% para teste.

TABELA II

DESEMPENHO DO MODELO UTILIZANDO O DATASET BCD NAS ETAPAS DE VALIDAÇÃO DO TREINAMENTO E TESTE.

Validação do Treinamento						
Classe	**Imagens**	**Instâncias**	***Precision***	***Recall***	**mAP50**	**mAP50-95**
Todas	73	967	0,835	0,939	0,925	0,647
RBC	73	819	0,689	0,929	0,889	0,631
WBC	73	72	0,969	1	0,974	0,802
Plaquetas	73	76	0,839	0,89	0,912	0,507
Teste						
Todas	36	471	0,782	0,915	0,876	0,609
RBC	36	398	0,653	0,93	0,869	0,628
WBC	36	37	0,959	0,93	0,964	0,779
Plaquetas	36	36	0,734	0,844	0,796	0,419

TABELA III

DESEMPENHO DO MODELO UTILIZANDO O DATASET BCCD NAS ETAPAS DE VALIDAÇÃO DO TREINAMENTO E TESTE.

Validação do Treinamento						
Classe	**Imagens**	**Instâncias**	***Precision***	***Recall***	**mAP50**	**mAP50-95**
Todas	20	275	0,865	0,926	0,938	0,648
RBC	20	234	0,797	0,839	0,895	0,632
WBC	20	21	0,975	1	0,995	0,821
Plaquetas	20	20	0,825	0,941	0,924	0,49
Teste						
Todas	68	915	0,864	0,926	0,945	0,677
RBC	68	789	0,789	0,849	0,903	0,668
WBC	68	69	0,99	1	0,995	0,824
Plaquetas	68	57	0,812	0,93	0,938	0,541

A detecção de objetos é uma tarefa similar à classificação, portanto, a matriz de confusão é uma importante ferramenta para avaliar os acertos e erros do modelo. Nas Figuras 3 e 4, apresentam-se as matrizes de confusão resultantes da etapa de teste utilizando os *datasets* BCD e BCCD, respectivamente. Ao analisar os resultados obtidos no conjunto BCD, verifica-se que o modelo classificou corretamente a maioria dos dados na classe *RBC*. Porém, houve muitas detecções incorretas no *background*, explicando, assim, o baixo valor de *precision* e um alto *recall*. Além disso, não houve detecção de *WBC* como *blackground* e somente uma instância foi detectada incorretamente, justificando o alto valor de *precision* e *recall*. O *recall* na classe *Platelets* é o mais baixo, pois o modelo detectou incorretamente *Platelets* como as demais classes.

Nas Figuras 5 e 6, apresentam-se as curvas de *precision-recall* para analisar a capacidade de detecção dos modelos e o *trade-off* entre as referidas métricas. Na Figura 5, apresenta-se a curva de *precision-recall* para cada classe no *dataset* BCD. É possível verificar que *WBC* atinge o maior mAP50 entre as classes, por apresentar as métricas *precision* e *recall* próximas de 1. Além disso, verifica-se que a classe *Platelets* possui o pior resultado, considerando que a curva decai rapidamente.

Na Figura 6 apresenta-se a curva de *precision-recall* para

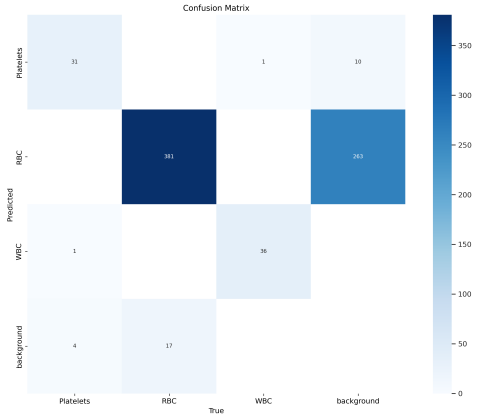

Fig. 3. Matriz de confusão da etapa de teste utilizando o *dataset* BCD.

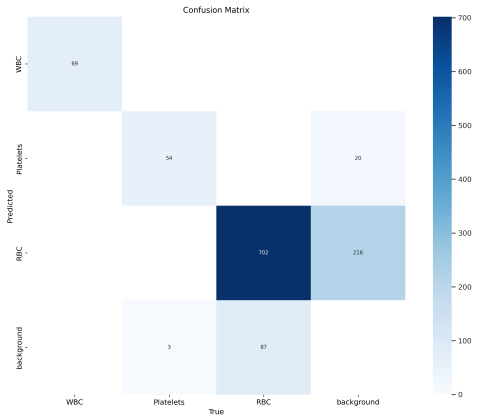

Fig. 4. Matriz de confusão da etapa de teste utilizando o *dataset* BCCD.

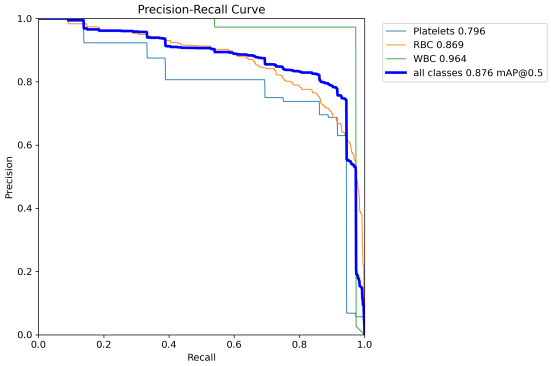

Fig. 5. Curva de *precision-recall* na etapa de teste utilizando o *dataset* BCD.

cada classe do *dataset* BCCD. Observa-se que a curva de *WBC* não está visível, mas está localizada próxima de 1 para a métrica *precision*, com um mAP50 de 0:995. Além disso, é possível verificar que a classe *Platelets* obteve um desempenho superior ao da *RBC*. Porém, todas as classes conseguiram ser detectadas com maior precisão, quando comparado com o *dataset* BCD.

Em relação à comparação com outras pesquisas, verifica-se a superioridade dos resultados, conforme apresentado na Tabela IV. Em termos da métrica mAP50, nesta pesquisa,