Classificação de Imagens com Redes Neurais Artificiais*

Tiago C A Amorim (RA: 100675)[a], Taylon L C Martins (RA: 177379)[b]

[a] *Doutorando no Departamento de Engenharia de Petróleo da Faculdade de Engenharia Mecânica, UNICAMP, Campinas, SP, Brasil*
[b] *Aluno especial, UNICAMP, Campinas, SP, Brasil*

Keywords: Classificação, Redes Neurais Artificiais, Rede Neurais Convolucionais, ResNet

1. Introdução

Este relatório apresenta as principais atividades realizadas no desenvolvimento das atividades propostas na Lista 03 da disciplina IA048: Aprendizado de Máquina, primeiro semestre de 2024. O foco deste exercício é de construir e avaliar o desempenho de redes neurais artificiais – MLP (densa de uma camada intermediária) e CNN (convolucionais *rasa* e *profunda*) – na classificação de imagens de células sanguíneas periféricas.

2. Tarefa Proposta

Nesta atividade, vamos abordar o problema de reconhecimento de células sanguíneas periféricas utilizando a base de dados BloodMNIST [1, 2, 3] (https://medmnist.com/), a qual possui 17.092 imagens microscópicas coloridas (3 canais de cor). O mapeamento entre os identificadores das classes e os rótulos está indicado na Tabela 1.

Id	Rótulo
0	Basófilos
1	Eosinófilos
2	Eritroblastos
3	Granulócitos imaturos
4	Linfócitos
5	Monócitos
6	Neutrófilos
7	Plaquetas

Tabela 1: Correspondência entre os identificadores numéricos das classes e os tipos de células sanguíneas.

(a) Aplique uma rede MLP com uma camada intermediária e analise (1) a acurácia e (2) a matriz de confusão para os dados de teste obtidas pela melhor versão desta rede. Descreva a metodologia e a arquitetura empregada, bem como todas as escolhas feitas.

(b) Monte uma CNN simples contendo:

- Uma camada convolucional com função de ativação não-linear.
- Uma camada de *pooling*.
- Uma camada de saída do tipo *softmax*.

Avalie a progressão da acurácia junto aos dados de validação em função:

- Da quantidade de *kernels* utilizados na camada convolucional;
- Do tamanho do *kernel* de convolução.

(c) Escolhendo, então, a melhor configuração para a CNN simples, refaça o treinamento do modelo e apresente:

- A matriz de confusão para os dados de teste;
- A acurácia global;
- Cinco padrões de teste que foram classificados incorretamente, indicando a classe esperada e as probabilidades estimadas pela rede.

Discuta os resultados obtidos.

(d) Explore, agora, uma CNN um pouco mais profunda. Descreva a arquitetura utilizada e apresente os mesmos resultados solicitados no item (c) para o conjunto de teste. Por fim, faça uma breve comparação entre os modelos estudados neste exercício.

3. Aplicação

A tarefa proposta foi desenvolvida em três *notebooks* Jupyter, em Python. Foi gerado um *notebook* para cada uma das arquiteturas de rede neural utilizadas: Rede MLP, Rede Convolucional *Simples* e Rede Convolucional *Profunda*. Foi feito o uso das bibliotecas *TensorFlow* [4] para montar as redes neurais e *Scikit-learn* [5] para realizar a otimização dos hiperparâmetros.

O código pode ser encontrado em https://github.com/TiagoCAAmorim/machine_learning.

*Relatório número 03 como parte dos requisitos da disciplina IA048: Aprendizado de Máquina.

3.1. Base de Dados

A base *BloodMNIST* foi construída com imagens de diferentes resoluções. Para este exercício foi escolhida a base de menor resolução: (28, 28). As imagens são classificadas em 8 classes (Figura 1). A base de dados é composta por 17 092 amostras, divididas em treino (11 959), validação (1 712) e teste (3 421).

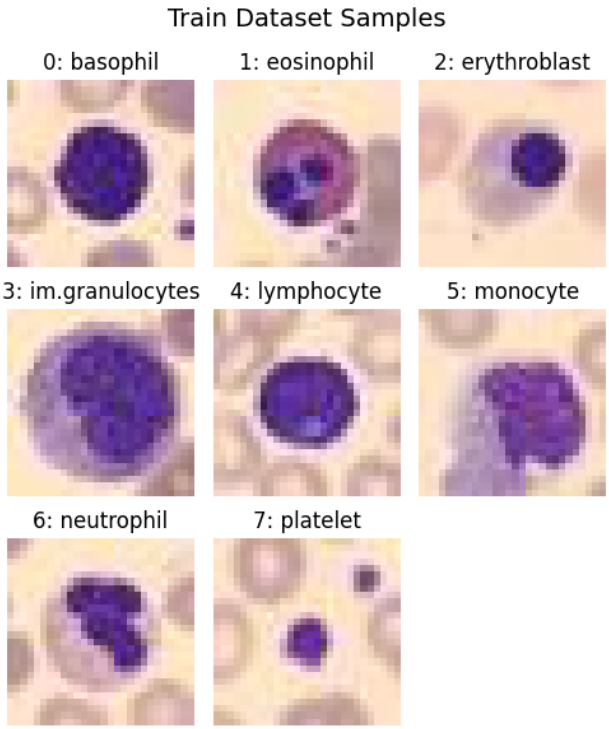

Figura 1: Exemplos de imagens por classe.

Os conjuntos de dados de treino e validação não são uniformemente distribuídos entre as classes (Figura 2). Algumas classes tem mais que o dobro de imagens que outras. Prevendo um possível efeito negativo no treinamento, foi proposto utilizar pesos por classe. A proposta foi definir os pesos proporcionais ao inverso do número de classes (Figura 3). O impacto do uso de pesos por classe será avaliado para cada classificador.

3.2. Rede MLP

O primeiro classificador construído é uma rede neural de uma camada intermediária (**MLP**). A rede é composta por uma camada de entrada, uma camada intermediária (com função de ativação não-linear) e uma camada de saída (com função de ativação *softmax*).

Apesar de ser uma rede simples, diferentes hiperparâmetros foram avaliados para tentar encontrar um classificador mais eficiente. A avaliação dos hiperparâmetros foi feita com busca em grade (*GridSearch*). Como o número de possíveis combinações é alto, a busca em grade foi feita por subconjuntos de hiperparâmetros.

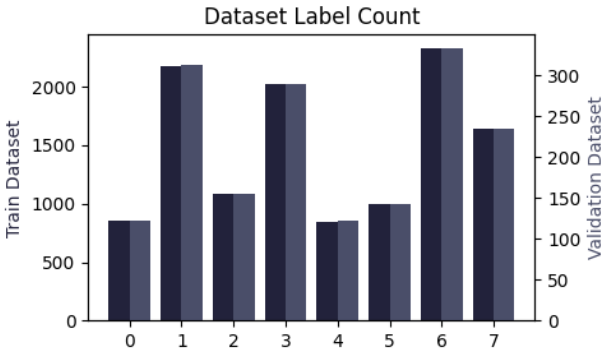

Figura 2: Número de imagens por classe.

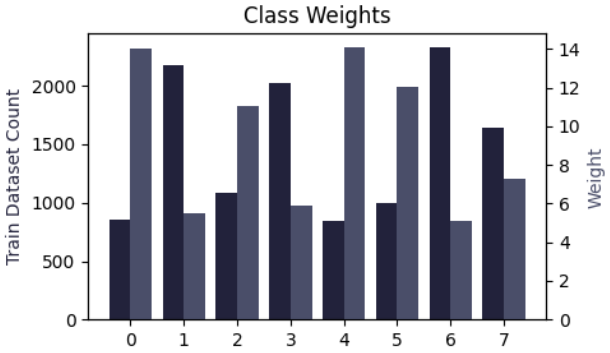

Figura 3: Pesos por classe.

Um conjunto de hiperparâmetros *ótimos* é inicialmente imposto. A partir deste conjunto *ótimo* é feita uma busca em grande com um subconjunto dos hiperparâmetros. Ao final desta busca os valores do conjunto de hiperparâmetros *ótimo* é atualizado com os valores encontrados que maximizam a acurácia do conjunto de validação. O processo é repetido para cada subconjunto de hiperparâmetros.

A Tabela 2 mostra os subconjuntos de hiperparâmetros (separados por linhas horizontais). A ordem dos subconjuntos na tabela coincide com a ordem de otimização dos hiperparâmetros.

A etapa de *data augmentation* testada consistiu de espelhamento (vertical e/ou horizontal) e rotação aleatórios. A descrição da base de dados cita que as imagens estão centradas, de forma que assumiu-se, para este exercício, não ser necessário aplicar a translação aleatória das imagens.

O ajuste de cada classificador foi feito por até 50 épocas, com parada antecipada (*early stopping*) caso a acurácia[1] dos dados de validação não melhore após 10 épocas (com um mínimo de 20 épocas). O classificador ajustado utiliza os pesos que deram a maior acurácia com os dados de validação durante o processo de ajuste.

[1]Como existe certo desbalanceamento entre as classes, a acurácia balanceada possivelmente seria uma métrica melhor, mas esta opção não está facilmente disponível no *TensorFlow*.

Hiperparâmetro	Opções
Usar *data augmentation*	Sim, Não
Usar pesos por classe	Sim, Não
Número de neurônios	64, 128, 256, 512, 1024
Otimizador	SGD, RMSprop, Adam
Função de ativação	relu, tanh
Tamanho do *batch*	16, 32, 64, 128

Tabela 2: Hiperparâmetros da rede MLP (parâmetros ótimos sublinhados).

Na busca em grade foi aplicada validação cruzada estratificada em 3 pastas (*StratifiedKFold*) com os dados de treino, com entropia cruzada como função objetivo. A métrica de definição do melhor classificador é a média da acurácia com os dados de validação. Para adequar o custo computacional ao *hardware* disponível, a busca em grade foi limitada a 40% dos dados de treino e validação.

A Figura 4 mostra o impacto do número de neurônios da camada intermediária na acurácia média com os dados de validação. Existe inicialmente um impacto positivo significativo em aumentar o número de neurônios (*underfitting*). O contínuo incremento leva a uma redução gradativa na qualidade do classificador (*overfitting*).

A otimização dos hiperparâmetros não foi exaustiva. Outros valores para os hiperparâmetros avaliados, além de outros hiperparâmetros, poderiam ter sido testados, possivelmente encontrando classificadores melhores. Avaliou-se que a otimização feita atende os objetivos do exercício.

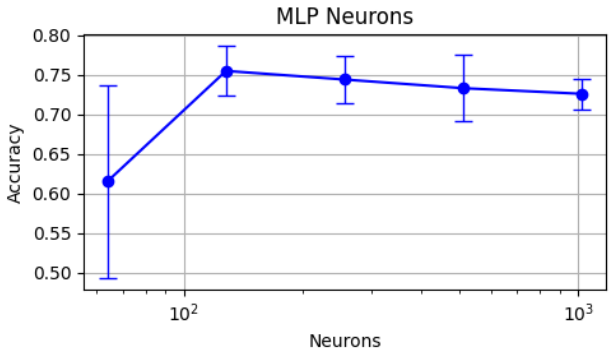

Figura 4: Efeito médio do número de neurônios da camada intermediária na acurácia da rede MLP (barras de erro são iguais a duas vezes o desvio padrão estimado com validação cruzada).

Uma nova rede MLP foi treinada com os hiperparâmetros ótimos (Figura A.15)[2], com um limite 200 épocas. É feita uma parada antecipada se a acurácia dos dados de validação não melhorar após 20 épocas. A Figura 5 mostra que a rede foi ajustada até a 71ª época. Esta rede tem 302 216 parâmetros treináveis.

[2]As figuras maiores foram concentradas no apêndice para facilitar a leitura.

A acurácia da rede MLP com os dados de teste ficou em 0.7992. A matriz de confusão (Figura 6) e os resultados por classe (Tabela 3) mostram que a distinção de algumas classes foi mais fácil (e.g.: Plaquetas), enquanto que a classificação de outras classes teve desempenho pior (e.g.: Monócitos).

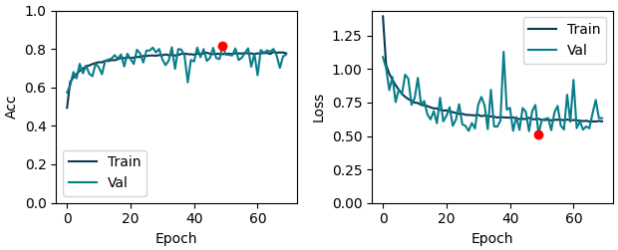

Figura 5: Histórico de ajuste da rede MLP.

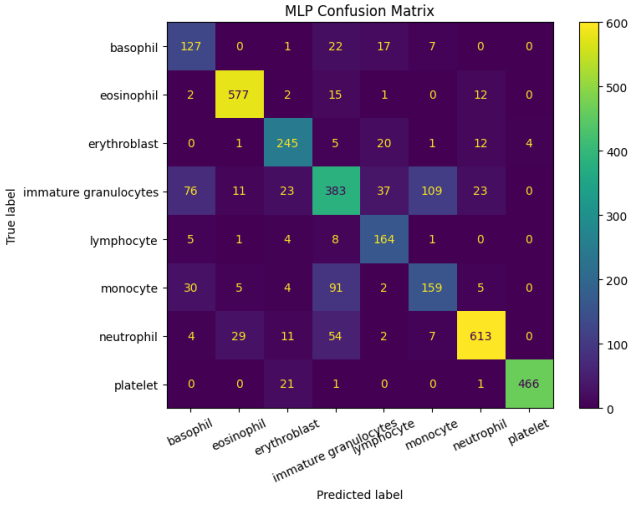

Figura 6: Matrix de confusão da rede MLP.

A Figura B.19 exemplifica alguns dos erros de classificação cometidos pelo modelo MLP. Na maioria das classificações errôneas a ordem associada à classe verdadeira é 2 (Figura 7)[3].

3.3. Rede Convolucional Simples

O segundo classificador construído é uma rede neural com uma camada convolucional (**CNN**). A rede é composta por uma camada de entrada, uma camada convolucional (com função de ativação não-linear), uma camada de *pooling* e uma camada de saída (com função de ativação *softmax*).

Os critérios (função objetivo, métricas, parada antecipada etc.) e o processo de otimização dos hiperparâmetros

[3]Cálculo da frequência relativa inclui as classificações corretas (ordem=1), que não é apresentada no gráfico por ter valor muito superior às demais barras.

Classe	Acurácia
Todas	0.7992
Basófilos	0.5205
Eosinófilos	0.9247
Eritroblastos	0.7878
Granulócitos imaturos	0.6615
Linfócitos	0.6749
Monócitos	0.5599
Neutrófilos	0.9204
Plaquetas	0.9915

Tabela 3: Resultados da rede MLP com os dados de teste.

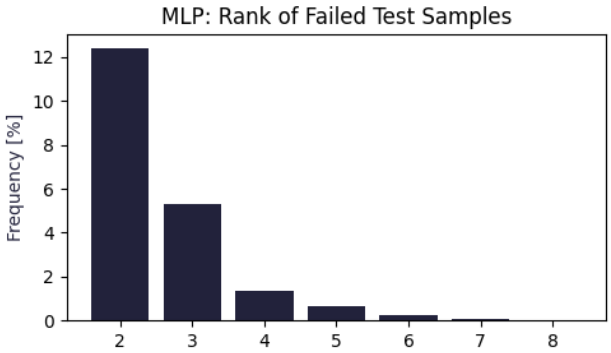

Figura 7: Histograma da ordem associada à classe verdadeira para as classificações errôneas com a rede MLP.

e de treinamento da rede CNN foi igual ao aplicado para a rede MLP. A Tabela 4 mostra os hiperparâmetros avaliados, os agrupamentos feitos e os valores ótimos.

Hiperparâmetro	Opções
Usar *data augmentation*	Sim, Não
Usar pesos por classe	Sim, Não
Tamanho dos filtros	3x3, 5x5, 7x7
Número de filtros	8, 16, 32, 64, 128
Tamanho do *pooling*	2, 3, 4, 5, toda figura[4]
Tipo do *pooling*	máximo, média
Otimizador	SGD, RMSprop, Adam
Função de ativação	relu, tanh
Tamanho do *batch*	16, 32, 64, 128

Tabela 4: Hiperparâmetros da rede CNN (parâmetros ótimos sublinhados).

A Figura 8 mostra o impacto do tamanho dos filtros (*kernel*) e do número de filtros da camada convolucional na acurácia média com os dados de validação. Observa-se que o comportamento não é linear, com o tamanho ótimo do filtro mudando em função do número de filtros.

[4]Equivale a usar as camadas *GlobalMax* e *GlobalAverage* do Keras/Tensorflow.

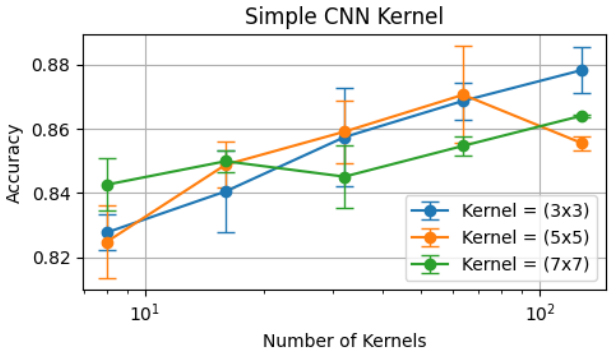

Figura 8: Efeito médio do tamanho dos filtros e do número de filtros da camada convolucional na acurácia da rede CNN.

A rede CNN com hiperparâmetros otimizados tem um total de 176 648 parâmetros treináveis (Figura A.16). A rede foi ajustada por 70 épocas (Figura 5), seguindo a mesma parametrização utilizada para a rede MLP com hiperparâmetros otimizados.

A acurácia do rede CNN com os dados de teste ficou em 0.9073. A acurácia teve um significativo incremento comparado com a rede MLP, apesar da rede CNN ter menos parâmetros. A classe Plaquetas teve apenas uma imagem incorretamente classificada (Figura 10). Em comparação com a rede MLP, a rede CNN tem melhor acurácia para todas as classes (Tabela 5).

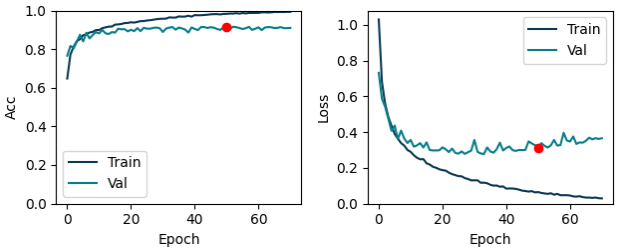

Figura 9: Histórico de ajuste da rede CNN.

Classe	Acurácia
Todas	0.9073
Basófilos	0.7992
Eosinófilos	0.9792
Eritroblastos	0.9293
Granulócitos imaturos	0.8238
Linfócitos	0.8848
Monócitos	0.7923
Neutrófilos	0.9339
Plaquetas	1.0000

Tabela 5: Resultados da rede CNN com os dados de teste.

Além de ter um número menor de classificações errôneas, o histograma da ordem associada à classe verdadeira

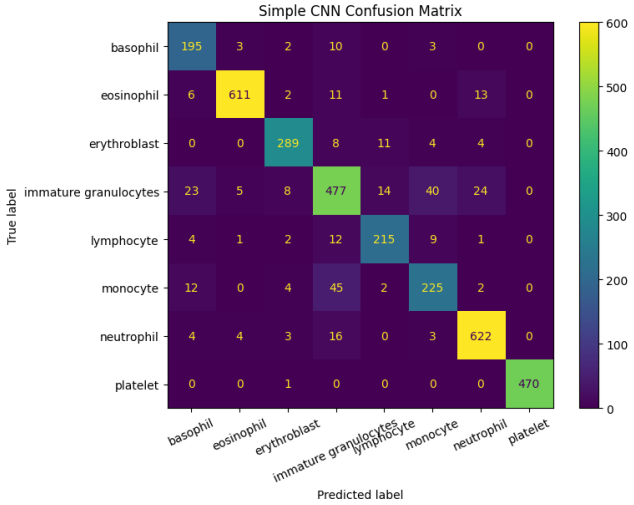

Figura 10: Matrix de confusão da rede CNN.

dos dados de teste para a rede CNN (Figura 11) se mostra um pouco mais concentrado nos valores menores que o histograma construído com a rede MLP.

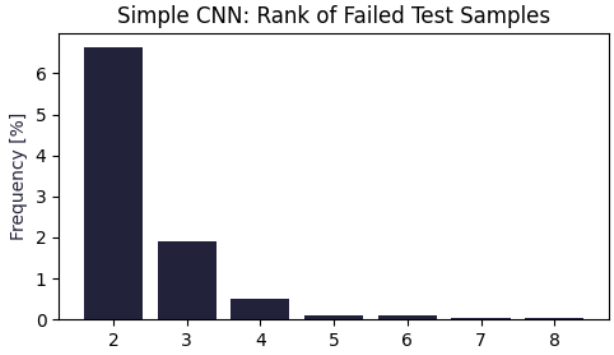

Figura 11: Histograma da ordem associada à classe verdadeira para as classificações errôneas com a rede CNN.

3.4. Rede Convolucional Profunda

A rede convolucional *Profunda* construída se baseou na arquitetura ***ResNet*** [6]. A resolução das imagens utilizadas nesta avaliação (28x28) é menor que a das imagens de entrada utilizadas na proposta original das ResNet (224x224). Em função desta diferença no tamanho das imagens, e para evitar uma *explosão* no número de parâmetros, foi construída uma versão simplificada dos blocos propostos para a ResNet-18.

Dado o alto custo computacional de treinar a rede neural proposta, não foi feita uma otimização dos hiperparâmetros. Os hiperparâmetros foram definidos de forma a limitar o número de parâmetros treináveis. A rede *ResNet* construída consiste de (hiperparâmetros utilizados entre parênteses):

1. Camada convolucional inicial (64 filtros 3x3).
2. Camada de *max pooling* com janela 3x3 (opção não utilizada).
3. Blocos residuais (2 blocos):
   (a) Duplica o número de filtros.
   (b) Camada convolucional inicial com *stride* igual a 2 (3x3).
   (c) Camadas convolucionais (2 camadas 3x3).
   (d) Camada convolucional 1x1 com *stride* igual a 2 aplicada no dado de entrada do bloco residual (*skip connection*).
   (e) Soma das saídas das camadas 3x3 com as saídas da camada 1x1.
   (f) Aplicada função de ativação (relu).
4. Camada de *Global Average Pooling*.
5. Camada densa (ativação relu).
6. Camada densa de saída.

Todas as camadas convolucionais e de *pooling* utilizam *padding* para buscar manter o tamanho das figuras. Onde não é explícito, foi utilizado *stride* igual a 1. Em todas as camadas a função de ativação é relu, exceto na camada de saída, que utiliza *softmax*. Ao final de cada camada convolucional é feito *batch normalization* antes de aplicar a função de ativação.

No início de cada bloco residual é duplicado o número de filtros, e o tamanho da figura é reduzido pela metade (*stride* 2 na primeira camada do bloco). Devido à mudança no tamanho da figura, a *skip connection* não utiliza a matriz identidade. É aplicada uma camada convolucional com filtro 1x1 e *stride* 2 para que as saídas tenham mesma dimensão e possam ser somadas.

Em face dos resultados anteriores, foi utilizado *data augmentation* nesta rede, e não foram utilizados pesos por classe. A rede é apresentada visualmente nas Figuras A.17 e A.18.

A rede tem 1 957 768 parâmetros treináveis. Foram utilizados os mesmos critérios de ajuste e parada antecipada das redes anteriores. A rede foi ajustada por 102 épocas (Figura 12).

A curva da acurácia com o conjunto de validação tem um comportamento mais *errático* que o observado nos gráficos das demais redes. Este comportamento indica que o critério de parada prematura não é adequado para esta rede mais complexa, e que possivelmente um número maior de épocas levaria a um resultado melhor.

A acurácia do rede *ResNet* com os dados de teste ficou em 0.9608. A acurácia teve um novo incremento significativo comparado com as demais redes. Apenas a acurácia da classe Plaquetas não melhorou com relação à rede CNN(Figura 13 e Tabela 6).

O histograma da ordem associada à classe verdadeira com os dados de teste (Figura 14) está mais concentrado nos valores 2 e 3 que os histogramas das demais redes. Este resultado indica que esta rede tem potencial para gerar melhores resultados com uma otimização dos hiperparâmetros e/ou um número maior de épocas no processo de ajuste.

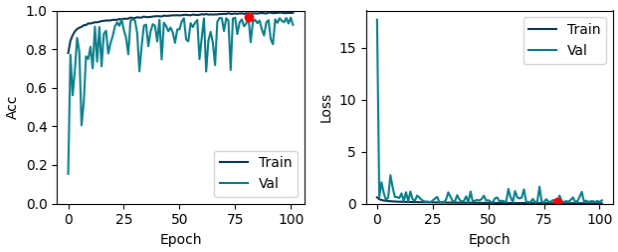

Figura 12: Histórico de ajuste da rede *ResNet*.

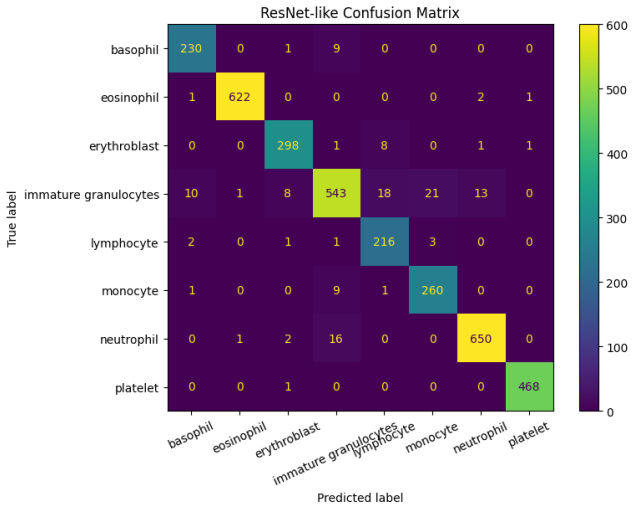

Figura 13: Matrix de confusão da rede *ResNet*.

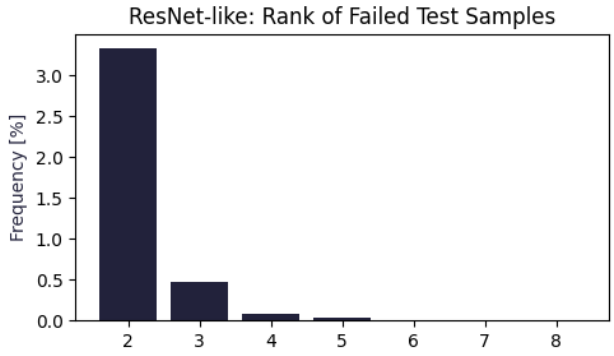

Figura 14: Histograma da ordem associada à classe verdadeira para as classificações errôneas com a rede *ResNet*.

4. Conclusão

A Tabela 7 resume os resultados principais. Fica claro que, para o problema proposto, é eficiente o uso de camadas convolucionais e de redes residuais profundas. A rede *ResNet* construída não passou por um processo de otimização dos hiperparâmetros ou fez uso de um grande número de camadas, mas a sua acurácia ficou comparável às reportadas para redes mais profundas.

Classe	**Acurácia**
Todas	0.9608
Basófilos	0.9426
Eosinófilos	0.9968
Eritroblastos	0.9582
Granulócitos imaturos	0.9378
Linfócitos	0.8889
Monócitos	0.9155
Neutrófilos	0.9760
Plaquetas	0.9957

Tabela 6: Resultados da rede *ResNet* com os dados de teste.

Rede	**Acurácia**
ResNet-18 (28)	0.958
ResNet-18 (224)	0.963
ResNet-50 (28)	0.956
ResNet-50 (224)	0.950
auto-sklearn	0.878
AutoKeras	0.961
Google AutoML Vision	0.966
MLP	0.799
CNN	0.907
ResNet	0.961

Tabela 7: Resultados reportados na publicação original [2] em comparação com as redes construídas nesta atividade.

Apêndice A. Redes Neurais Construídas

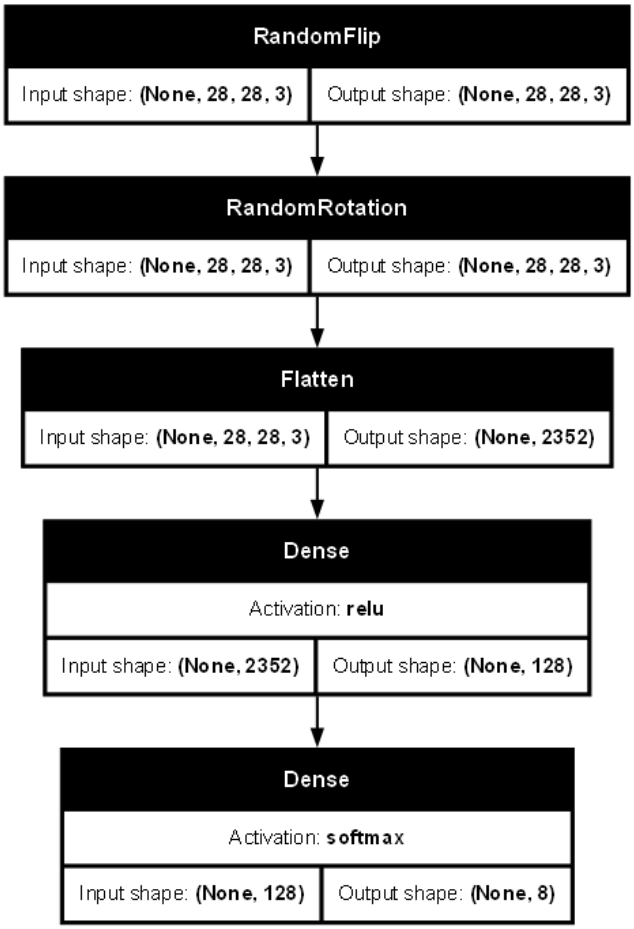

Figura A.15: Rede MLP.

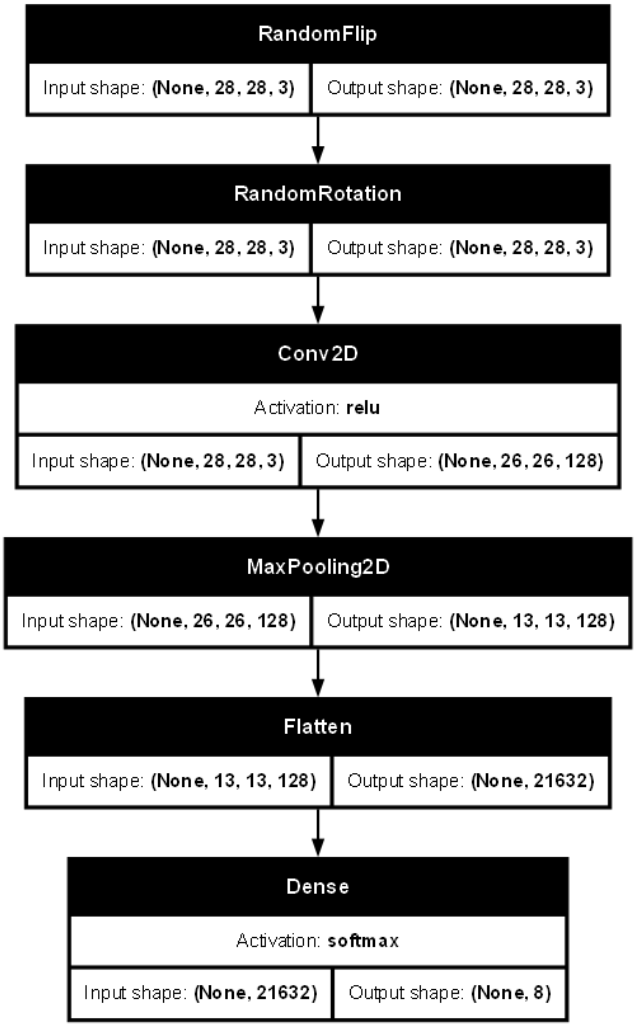

Figura A.16: Rede CNN.

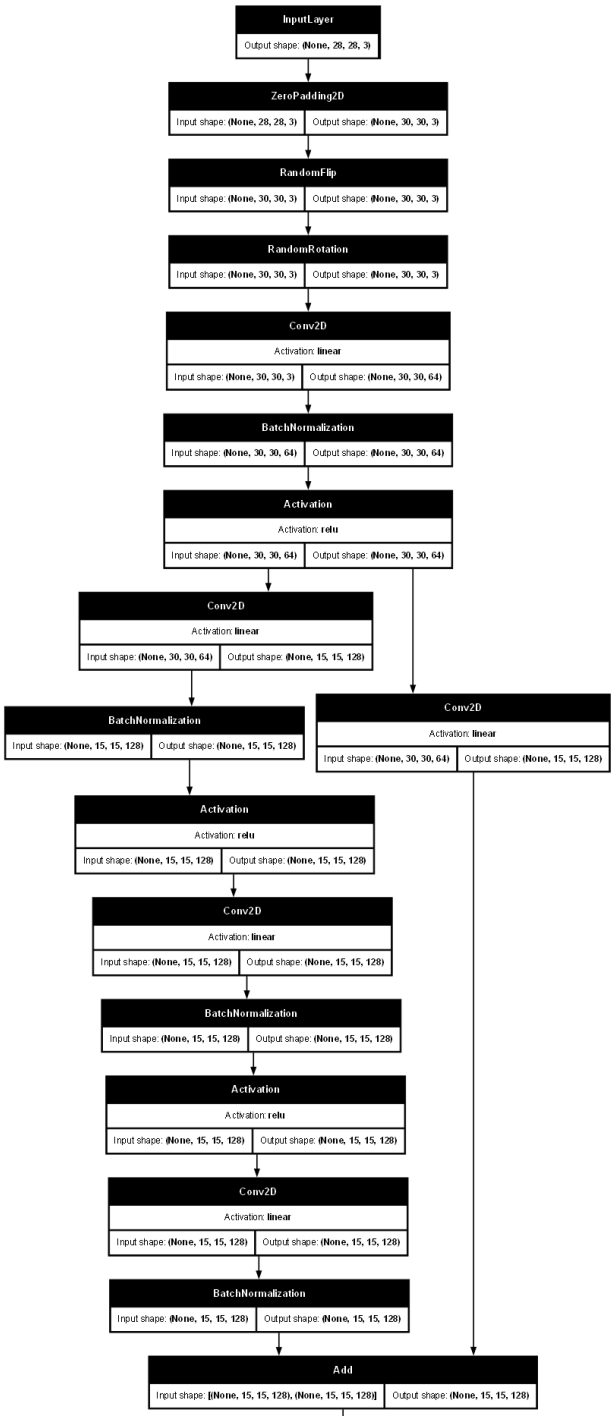

Figura A.17: Início e primeiro bloco residual da rede *ResNet*.

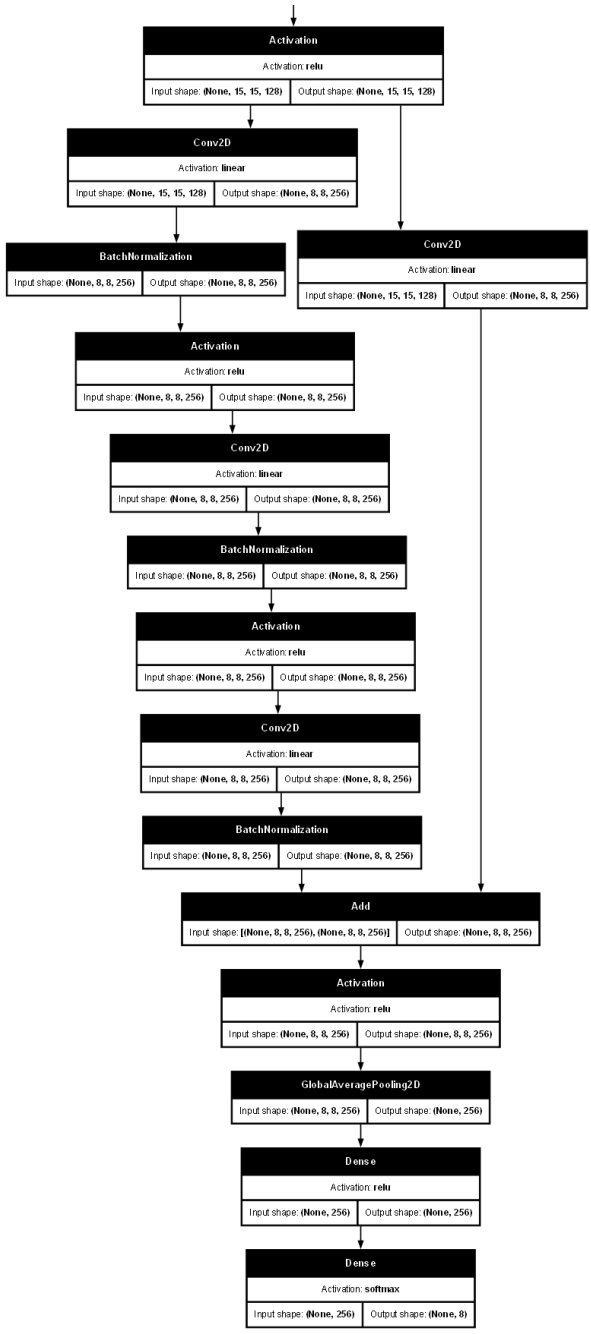

Figura A.18: Segundo bloco residual e final da rede *ResNet*.

Apêndice B. Exemplos de Classificações Errôneas

À esquerda é apresentada uma figura do conjunto de testes, junto com a sua classificação correta. À direita de cada imagem é apresentada a saída do classificador (probabilidades associadas a cada classe), com a indicação da ordem (*rank*) associada à classe verdadeira.

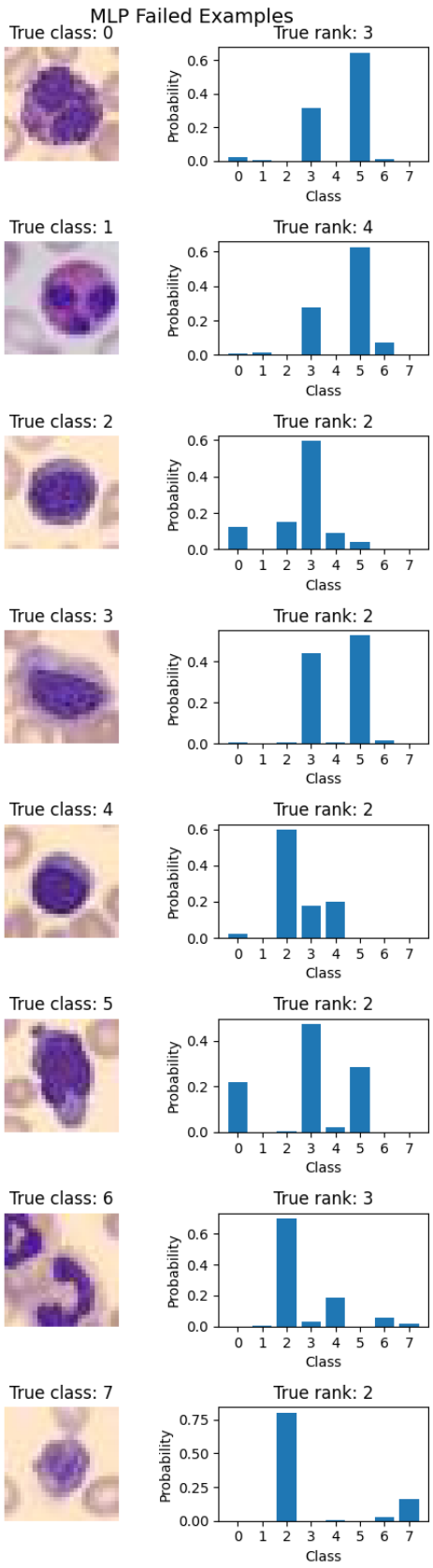

Figura B.19: Exemplos de classificações errôneas com a rede MLP.

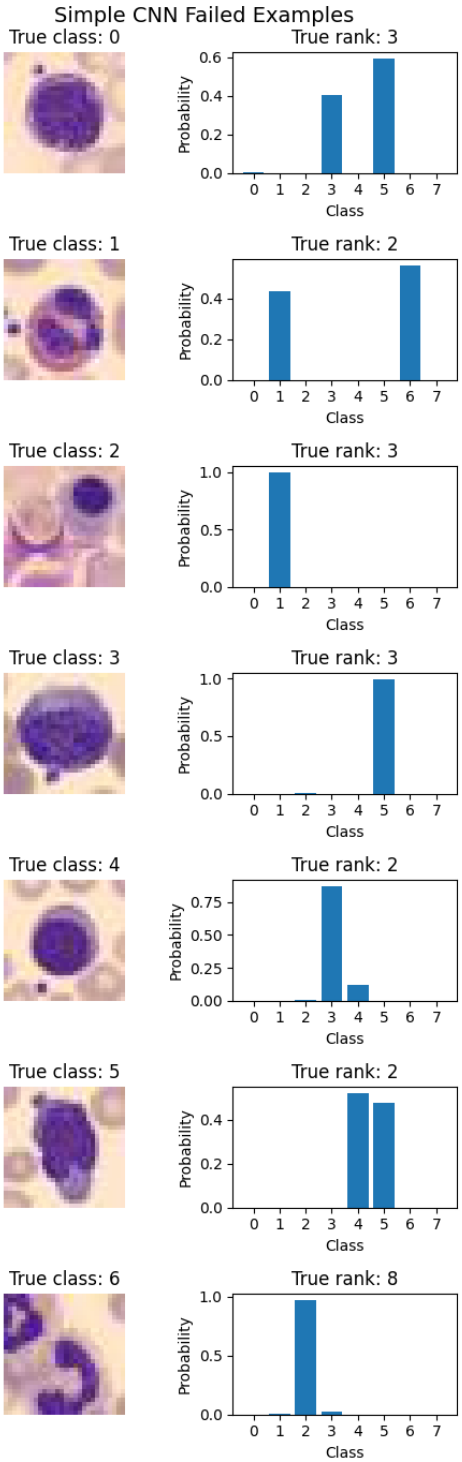

Figura B.20: Exemplos de classificações errôneas com a rede CNN.

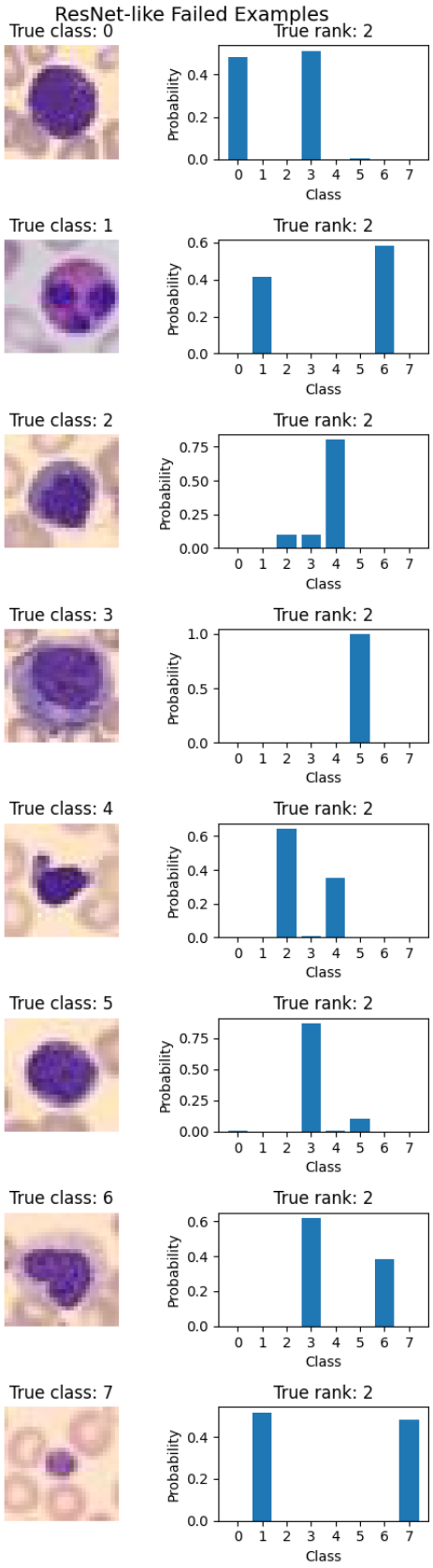

Figura B.21: Exemplos de classificações errôneas com a rede *ResNet*.

Referências

[1] J. Yang, R. Shi, B. Ni, Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis, in: IEEE 18th International Symposium on Biomedical Imaging (ISBI), 2021, pp. 191–195.

[2] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, B. Ni, Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification, Scientific Data 10 (1) (2023) 41.

[3] A. Acevedo, A. Merino González, E. S. Alférez Baquero, Á. Molina Borrás, L. Boldú Nebot, J. Rodellar Benedé, A dataset of microscopic peripheral blood cell images for development of automatic recognition systems, Data in brief 30 (article 105474) (2020).

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `https://www.tensorflow.org/`

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. corr abs/1512.03385 (2015) (2015).