Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

IA048 – Aprendizado de Máquina 15 de maio de 2024

Docentes: Levy Boccato & Romis Attux

Discente:

– Gabriel Toffanetto França da Rocha – 289320

Atividade 3 – Redes Neurais

Sumário

1 Apresentação dos dados

A base de dados BloodMNIST é formada por um total de 15380 imagens de células sanguíneas divdidas em 8 classes, como mostrado na Figura 1. Essas amostras se dividem em conjunto de treinamento, com 11959 representantes e teste, com 3421 itens. As imagens estão disponíveis em diferentes resoluções, sendo *a priori*, escolhida a resolução de 28x28 para trazer uma maior eficiência computacional, principalmente para o caso das redes densas. **Porém, propõem-se o teste de amostras de maior resolução para a rede convolucional já treinada com as amostras 28x28, para validação da robustez ao tamanho da entrada.**

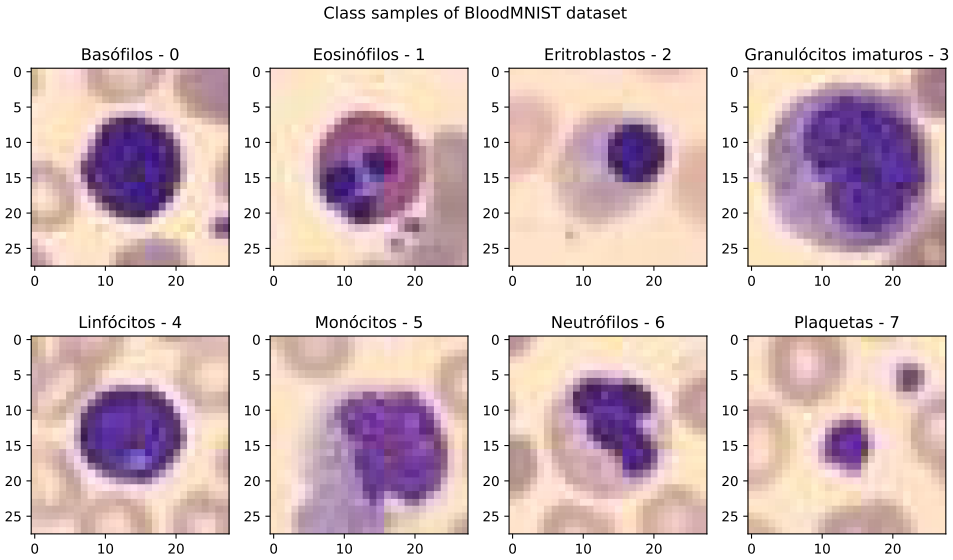


Figura 1: Amostras das classes do *dataset* BloodMNIST.

O *dataset* BloodMNIST já fornece de forma separada os dados utilizados para treinamento, e os que serão utilizados para teste. A Figura 2 mostra a distribuição de amostrar para cada classe, para os conjuntos de treinamento, em azul, e de teste, em verde. Observa-se que existem classes majoritárias, como a 1, 3, 6 e 7, que apresentam muito mais amostras que as demais. O perfil de amostras por classe é similar nos conjuntos de teste e treinamento, o que mostra que o impacto no mapeamento pelo desbalanceamento das classes irá afetar de forma igual ambos os *datasets*. **Tal disparidade pode ser tanto causada por um viés na coleta de dados, quanto também pela ocorrência real de mais representantes de uma classe do que das demais, o que modela de forma realista os dados.**

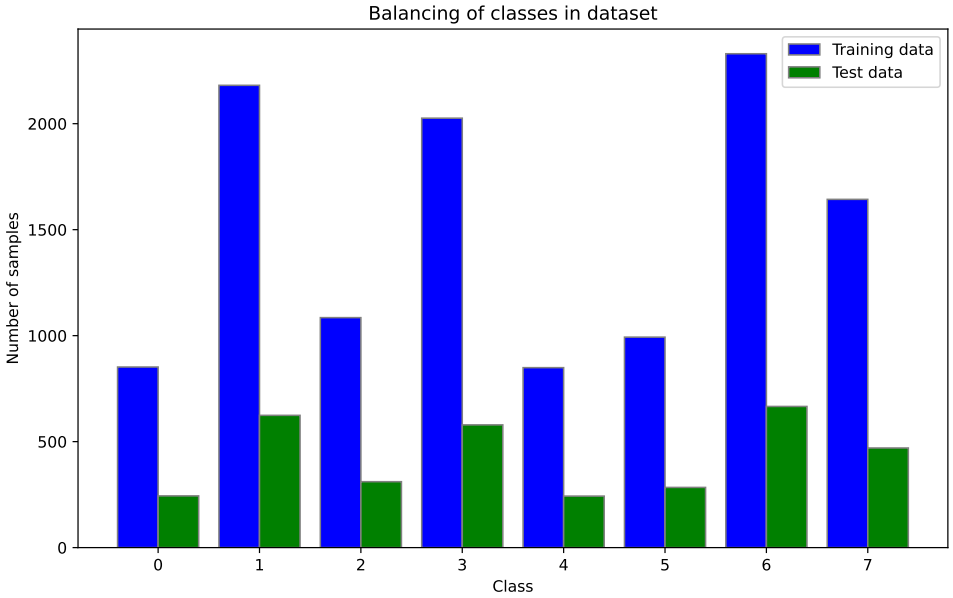


Figura 2: Balanço das classes nos *datasets* de treinamento e teste.

Para realizar o treinamento utilizando da ferramenta de validação cruzada, foi escolhida a validação do tipo *holdout*, por meio do particionamento do conjunto de dados de treinamento, obtendo um novo conjunto de dados de validação. O novo conjunto de treinamento é formado pelos primeiros 70% do conjunto original de treinamento, enquanto o de validação, os 30% restantes do final do *dataset* original. Para garantir a veracidade da validação, quer-se que ambos os conjuntos tenham a mesma representação de cada classe, o que pode se afirmar positivo de acordo com a Figura 3.

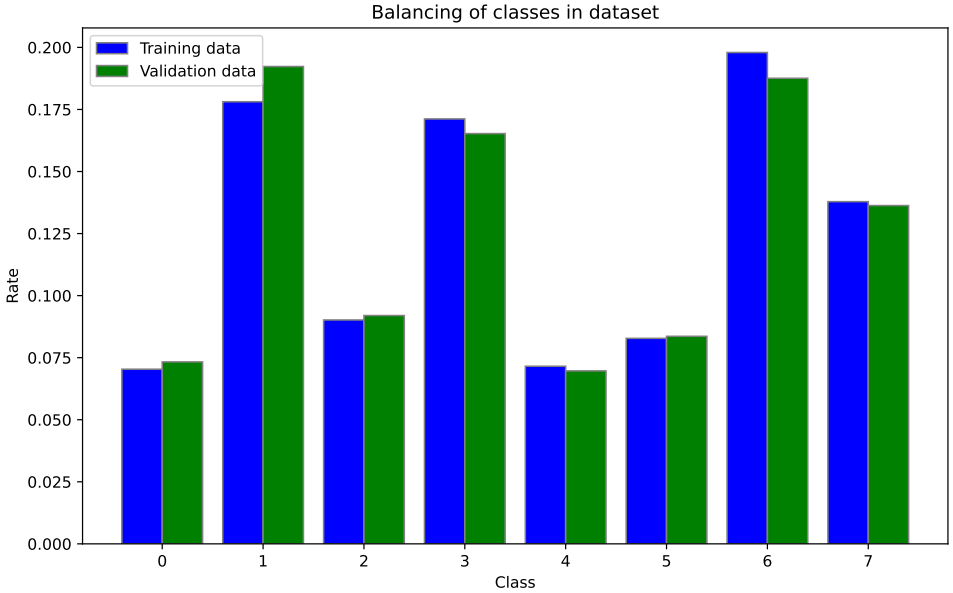


Figura 3: Balanço das classes nos conjuntos de dados de treinamento e validação cruzada do tipo *holdout*.

2 MLP

A implementação da rede MLP com uma camada intermediária se deu por meio do uso do *framework* TensorFlow, possuindo uma camada de entrada que sequencia os pixels da imagem em um vetor, uma camada de neurônios intermediária, e uma camada de saída com função de ativação *softmax* para geração do vetor *one-hot encoding* das probabilidades da entrada pertencer à cada uma das 8 classes.

2.1 Busca do melhor modelo

Para realizar a busca do melhor modelo da MLP, considerando que existem uma grande possibilidade de hiper-parâmetros, foi realizada uma busca exaustiva simplificada por etapas, baseada no funcionamento dos *wrappers*, onde dada uma configuração inicial de parâmetros, baseado no comumente visto na literatura (GÉRON, 2019). As variáveis selecionadas para a busca foram: Número de neurônios da camada intermediária, função de ativação dos neurônios da camada intermediária, taxa de *dropout* dos neurônios da camada intermediária e otimizador para ajuste dos pesos. Demais hiper-parâmetros como tamanho do *batch* e passo do algoritmo de otimização foram mantidos *default*.

A busca por etapas se deu da seguinte forma: Dada a condição inicial, uma MLP de 256 neurônios na camada intermediária com função de ativação ReLU, sem *dropout* e ajustada por *Stochastic Gradient Descendent* (SGD), testou-se a rede alterando primeiramente o número de neurônios. Uma vez conhecido o valor que obteve maior acurácia, testou-se as funções de ativações canditadas para esse número de neurônios, buscando a combinação que desse a melhor acurácia. O processo se repete para o *dropout* e o algoritmo de otimização, onde ao final se obtém a combinação que agrega o melhor número de neurônios visto, melhor função de ativação para tal conjunto de neurônios, melhor taxa de *dropout* e o melhor otimizador.

Hiper-parâmetros variados durante a busca:

- Número de neurônios: [`256`; `512`; `1024`; `2048`; `4096`]
- Função de ativação: [`ReLU`; `Sigmoide`]
- *Dropout*: [`0,0`; `0,25`; `0,5`]
- Otimizador: [`SGD`; `ADAM`]

Para todos os casos, foram treinadas 500 épocas, com *mini-batch* de 32 amostras, utilizando uma função de *callback* para salvar os parâmetros da melhor época, evitando assim preocupações com *overfitting*.

2.1.1 Número de neurônios

Com intuito de definir a quantidade de neurônios que irão compor a camada intermediária da MLP, treinou-se a rede para camadas intermediárias com 256, 512, 1024, 2048 e 4096 neurônios,

monitorando a perda e a acurácia para os dados de treinamento e de validação. Os gráficos da Figura 5, mostram a evolução dessas medidas ao longo das épocas. Observa-se que para todos os casos, ocorre *overfitting* no treinamento, onde a perda de validação, em vermelho, começa a apresentar uma elevação na sua curva ruidosa. Porém, como o treinamento foi realizado salvando o conjunto de pesos da melhor época, não se faz um problema treinar o modelo mais épocas do que o necessário.

Ao analisar a acurácia e perda de validação para cada um dos modelos treinados com uma quantidade diferente de neurônios na camada intermediária, obtém-se a Figura 4. Como esperado, ao aumentar o número de neurônios se aumenta a acurácia e reduz a perda, uma vez que a rede se torna mais flexível e consegue aproximar mais o mapeamento alvo do treinamento. Porém, com o aumento da flexibilidade da máquina, o treinamento também se torna mais desafiador, sendo necessários uma grande quantidade de dados para maximizar a generalização do modelo.

Considerando todos os números de neurônios utilizados, não há um aumento estrondoso da acurácia, mostrando que para todos eles, o mapeamento pode ser bem aproximado, porém, a redução da perda é considerável considerando a rede com camada intermediária de 256 neurônios e a de 4096 neurônios. Observando a forma da curva de acuária, é perceptível que a taxa de crescimento apresenta uma saturação, ou seja, o valor máximo da acurácia que pode ser atingido por essa arquitetura de rede neural tende a atingir um valor máximo.

Com isso, a camada intermediária de 4096 neurônios foi escolhida para a rede final, e será utilizada nas buscas dos hiper-parâmetros subsequentes.

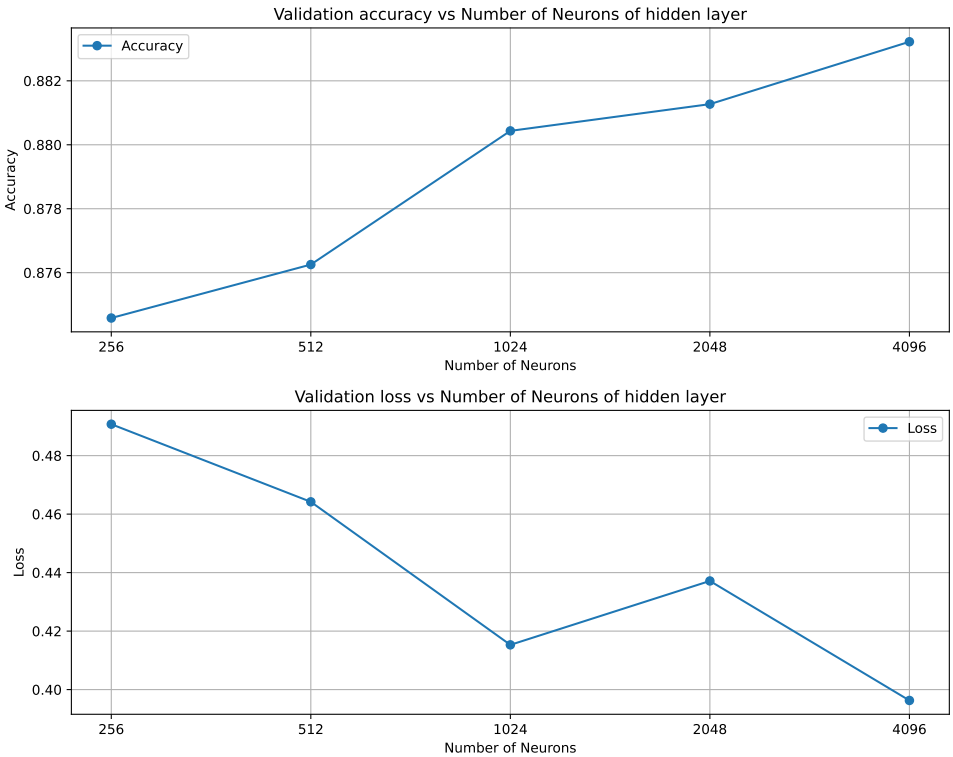
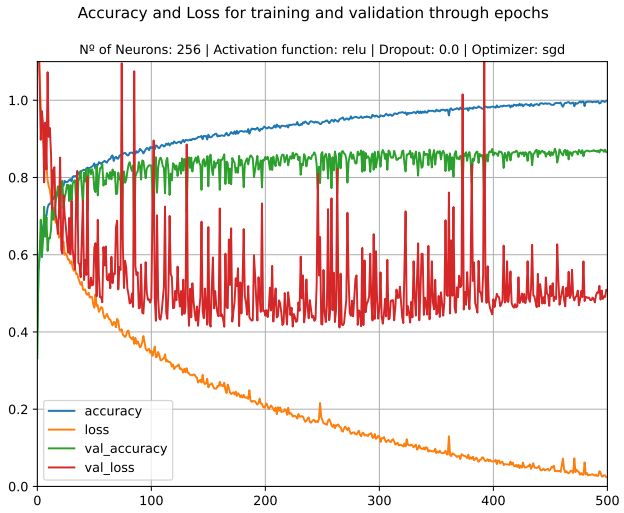
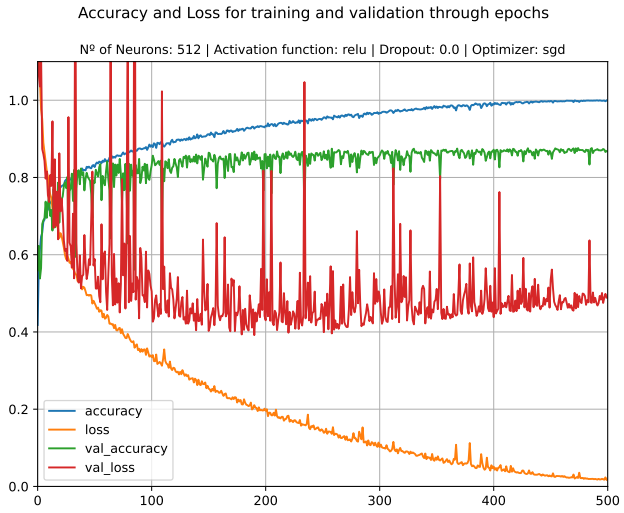


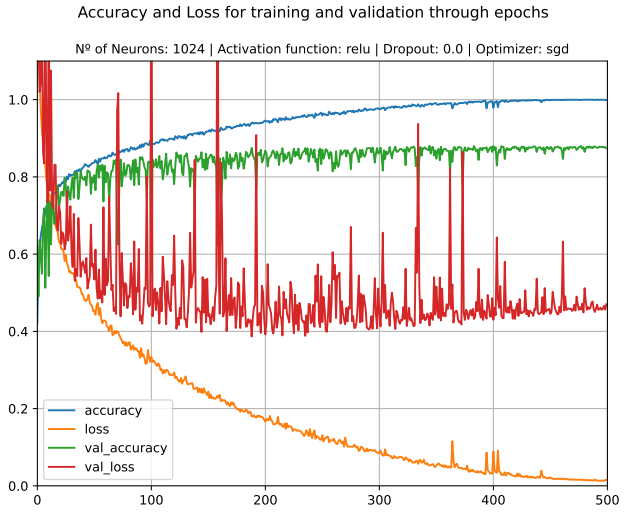
Figura 4: Acurácia e perda com a variação do número de neurônios da camada intermediária.



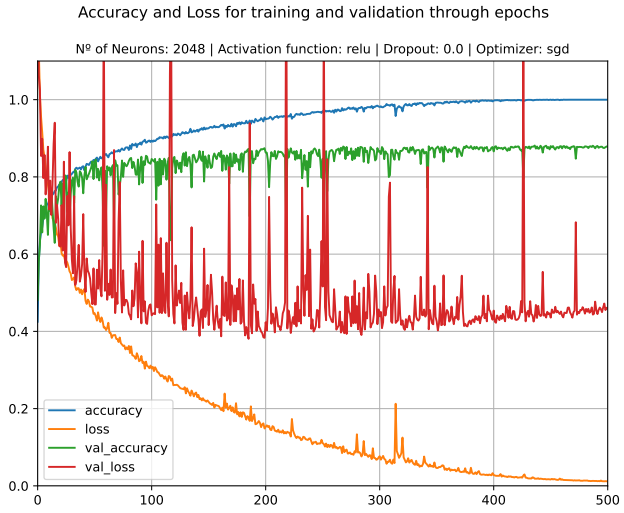
(a) 256 neurônios.



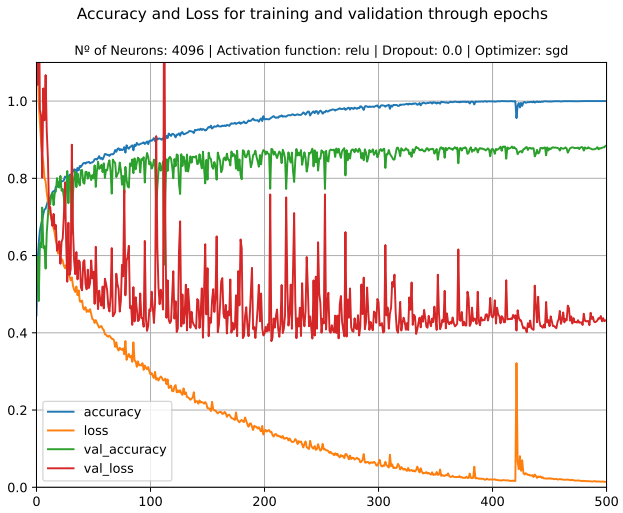
(b) 512 neurônios.



(c) 1024 neurônios.



(d) 2048 neurônios.



(e) 4096 neurônios.

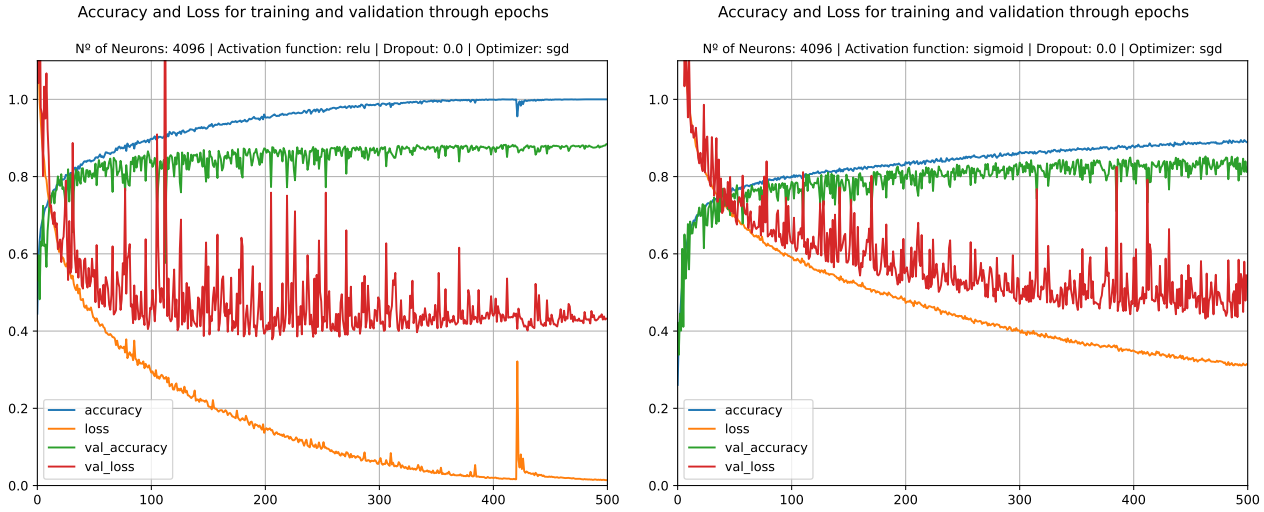
Figura 5: Evolução da perda e acurácia de treinamento e validação com as épocas de treinamento para cada número de neurônios avaliados.

2.1.2 Função de ativação

A função de ativação é a responsável por dar um mapeamento não-linear para a MLP, dessa forma, sua escolha é crucial para viabilizar o treinamento da rede, assim como para garantir a capacidade de generalização da mesma. Foram escolhidas duas funções de ativação com características diferentes, a ReLU, que executa a operação de um retificador linear, e a sigmoide, que varia seu valor de 0 a 1 na forma de uma função logística.

A Figura 6a mostra o desenvolvimento do treinamento para a rede com função de ativação ReLU. Observa-se que ocorre um leve *overfitting*, e que é possível se aproximar de forma considerável de um bom mínimo local da superfície de erro. Já para a função sigmoidal, Figura 6b, o treinamento ocorre de forma muito mais lenta, não se aproximando tanto do mínimo local. Mesmo assim, se observa a saturação da acurácia de validação em um valor menor.

Como a função sigmoidal apresenta saturação, seus valores de ativação tendem a ser menores que os valores de saída de neurônios com ReLU, o que faz com que o gradiente seja limitado e os passos de treinamento sejam menores, justificando um treinamento mais lento, e a acomodação na bacia de atração de um mínimo local de pior qualidade.



(a) ReLU. (b) Sigmoide.

Figura 6: Evolução da perda e acurácia de treinamento e validação com as épocas de treinamento para cada função de ativação candidata.

Como previsto pelos dados de treinamento, a acurácia para a rede com função de ativação logística é menor, como visto na Figura 7. Dessa forma, a função ReLU foi mantida como função de ativação do modelo final, e continuará sendo usada nas próximas buscas.

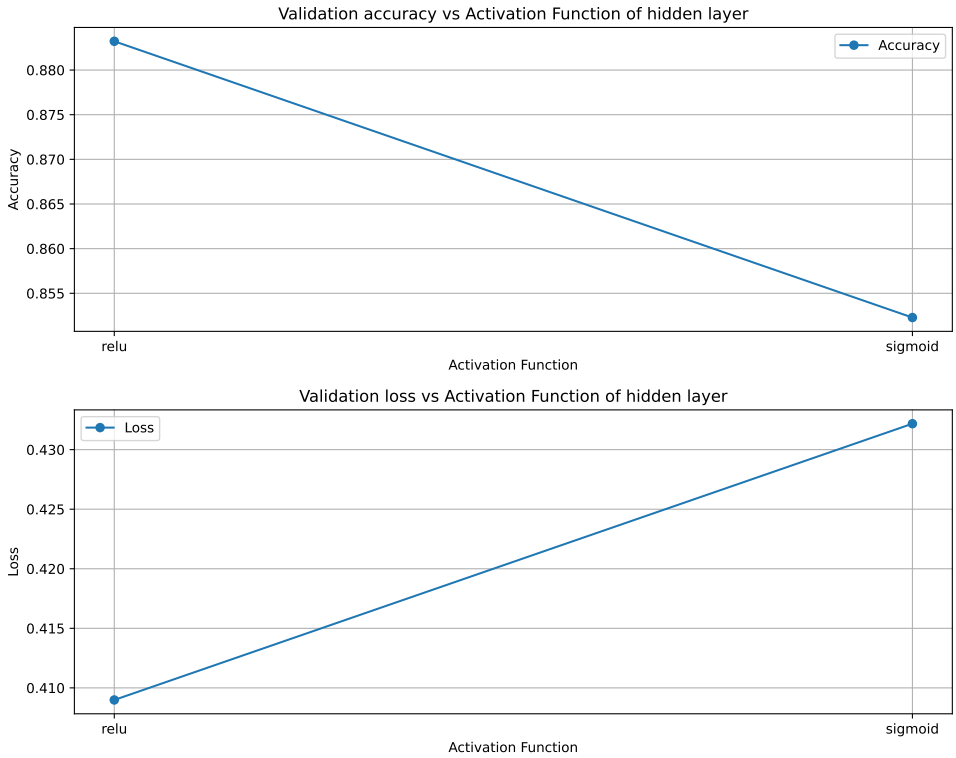
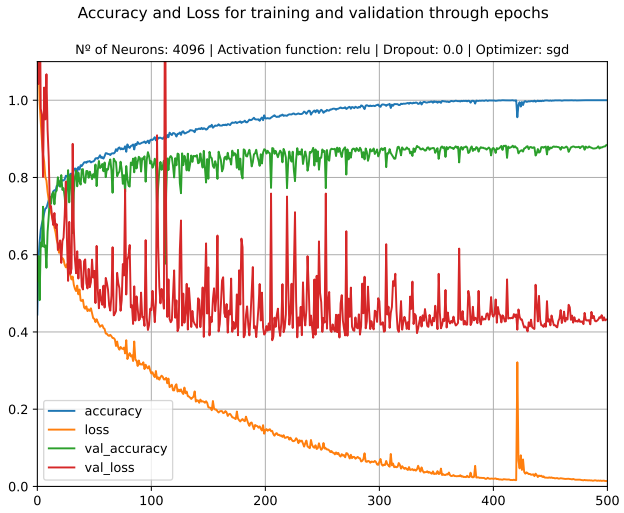


Figura 7: Acurácia e perda com a variação da função de ativação dos neurônios da camada intermediária.

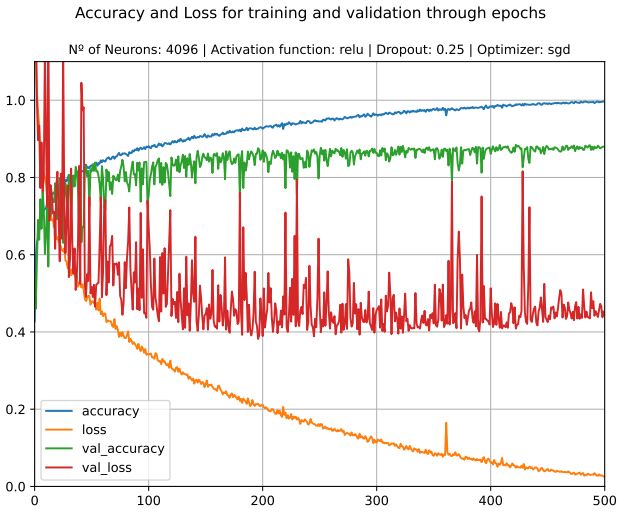
2.1.3 *Dropout*

Uma vez que se está sendo utilizada uma rede com 4096 neurônios na camada intermediária, o treinamento se torna desafiador, onde para maximizar a capacidade de generalização, se deseja que todos os neurônios apresentem uma função útil na camada. Para isso, buscou-se o uso do *dropout*, que foi testado com diferentes taxas.

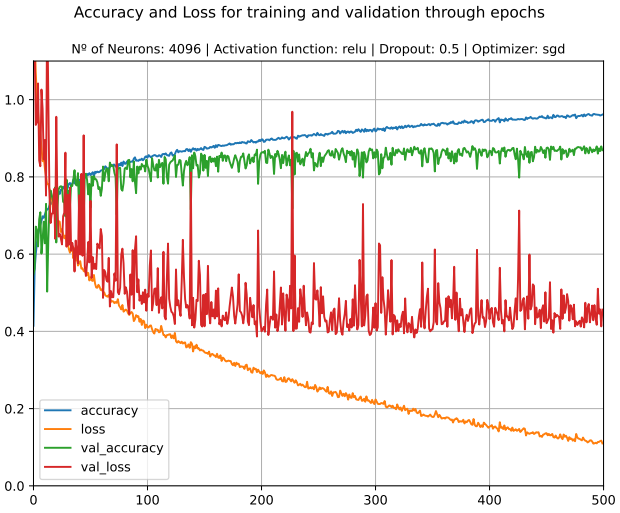
Os gráficos da Figura 8 mostram a evolução do treinamento da MLP sem *dropout*, e com taxas de *dropout* de 0,25 e 0,5. Observa-se primeiramente, que a inserção dessa técnica limitou o *overfitting*, e para o caso da Figura 8c, resultou em uma maior lentidão do treinamento da rede, ou seja, o desligamento dos neurônios se tornou forte o suficiente para diminuir a capacidade de aprendizado da rede.



(a) 0,0.



(b) 0,25.



(c) 0,5.

Figura 8: Evolução da perda e acurácia de treinamento e validação com as épocas de treinamento para diferentes taxas de *dropout*.

Dessa forma, a taxa de *dropout* de 0,25 foi incorporada à rede MLP, e será utilizada nas buscas posteriores.

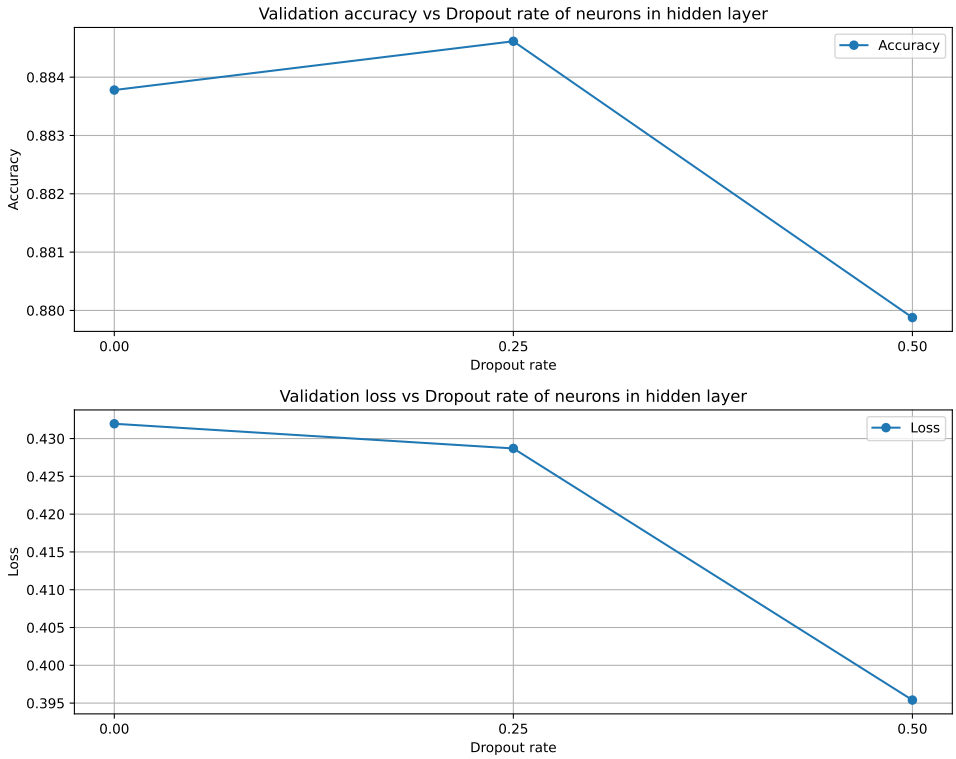
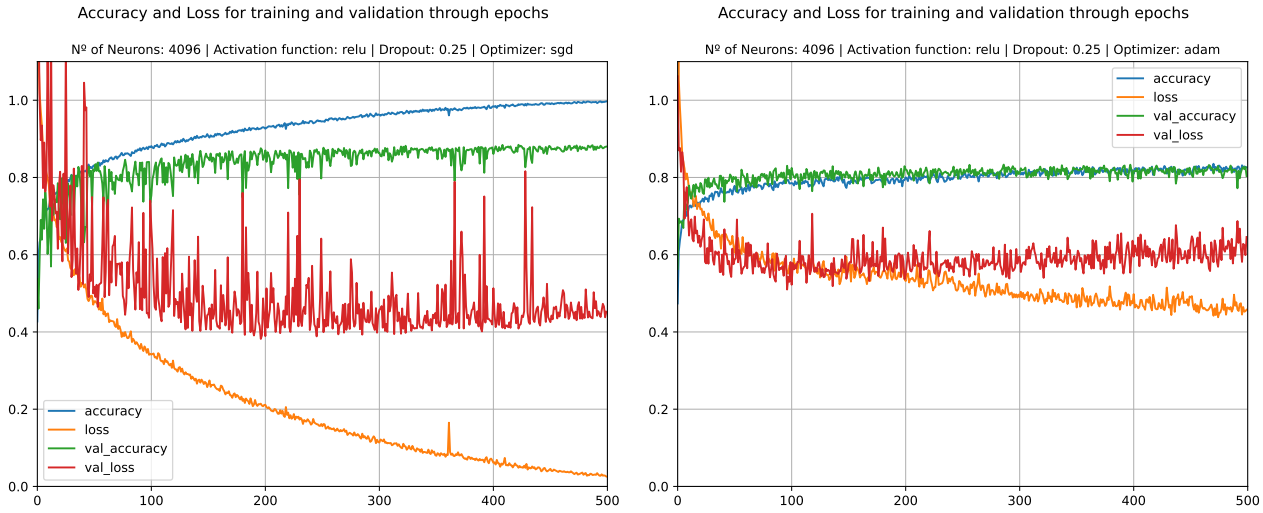


Figura 9: Acurácia e perda com a variação da taxe de *dropout* dos neurônios da camada intermediária.

2.1.4 Otimizador

Uma vez que a superfície de erro é desconhecida e comumente multimodal, o uso de um bom otimizador se faz crucial para garantir a convergência do modelo para um bom mínimo local, que garante uma melhor chance de maximização da capacidade de generalização do modelo. Para a busca, foram candidatos um algoritmo clássico, o *Stochastic Descendent Gradient* (SGD), e um algoritmo adaptativo, o ADAM.

O treinamento com cada um dos otimizadores se dá na Figura 10, onde pode-se observar que como esperado, o algoritmo adaptativo converge em menos épocas e de forma menos ruidosa, porém para um mínimo local de pior qualidade. Já o SGD, apresenta uma trajetória ruidosa, mas consegue alcançar um mínimo local de melhor qualidade, atingindo valores altos de acurácia e minimizando consideravelmente a perda.



(a) ReLU. (b) Sigmoide.

Figura 10: Evolução da perda e acurácia de treinamento e validação com as épocas de treinamento para os otimizadores analisados.

Como já se sabe, algoritmos adaptativos tendem a convergir em menos épocas, porém nada se garante sobre a qualidade do mínimo local de convergência. No trabalho realizado por Wilson et al. (2018), a principal conclusão obtida é que máquinas treinadas com algoritmos adaptativos tendem a generalizar pior do que máquinas treinadas com algoritmos de SGD. Dessa forma, o mesmo resultado pode ser visto na MLP treinada, onde pela Figura 11, observa-se que a rede neural treinada com SGD consegue uma acurácia consideravelmente maior, e minimiza mais a função de custo sobre os dados de validação.

Sendo assim, o algoritmo de otimização SGD foi definido para o modelo final da rede, encerrando a busca de hiper-parâmetros.

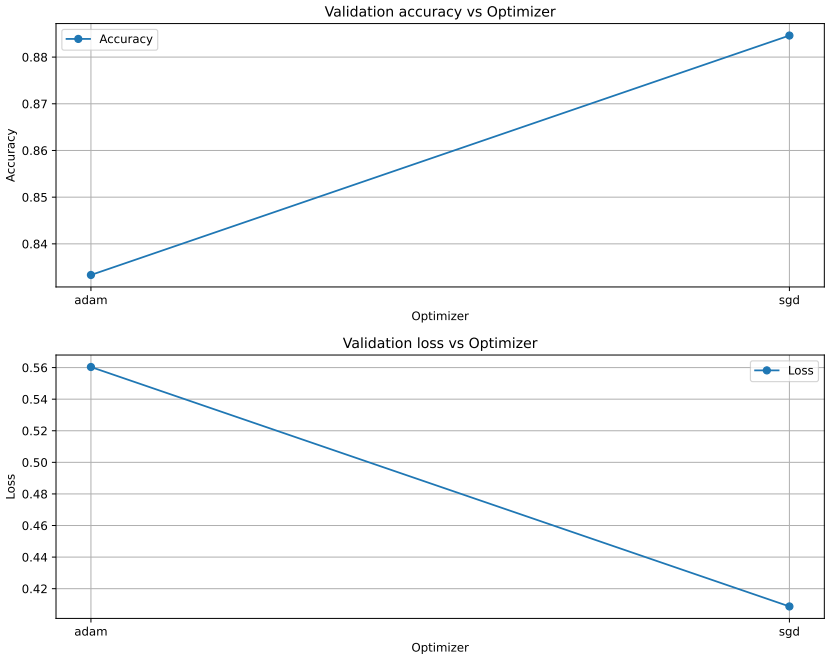


Figura 11: Acurácia e perda com a variação do algoritmo de otimização.

2.2 Análise do melhor modelo

Com a realização da busca exaustiva, foram obtidos como melhores hiper-parâmetros para a rede MLP:

- Número de neurônios: [`4096`]
- Função de ativação: [`ReLU`]
- *Dropout*: [`0,25`]
- Otimizador: [`SGD`]

Essa rede apresenta um total de 9,670,664 pesos treináveis, que uma vez ajustados durante o treinamento, apresentaram os seguintes valores de acurácia e acurácia balanceada frente aos dados de teste:

$$Accuracy = 0.8705 \tag{1}$$

$$BA = 0.8437 \tag{2}$$

A matriz de confusão do classificador é exibida na Tabela 1. Primeiramente, observa-se que a classe 3 é a mais desafiadora, uma vez que ela é fortemente confundida com as classes 0, 2, 4, 5 e 6, sendo a classe 7 a única que não se confunde com a 3. Pela Tabela 2, consegue-se ter um melhor panorama da precisão e do *recall* das classes, uma vez que as mesmas não são balanceadas. Como esperado, a classe 3 apresenta baixa precisão e o pior *recall*, mas a classe 0 é a que apresenta pior precisão, apresentando falsos positivos para a classe 3, 4 e 5. A classe 7 é a que apresenta a melhor classificação, com precisão quase unitária.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	177	3	0	43	4	17	0	0
1	1	613	0	3	1	0	6	0
2	4	1	264	16	6	3	11	6
3	34	26	3	447	5	32	32	0
4	13	0	11	31	182	0	6	0
5	10	2	1	48	3	215	5	0
6	0	22	5	22	2	2	611	2
7	0	0	1	0	0	0	0	469

Tabela 1: Matriz de confusão do classificador baseado em MLP.

Classe	Precisão	*Recall*
0	0.7254	0.7406
1	0.9824	0.9190
2	0.8489	0.9263
3	0.7720	0.7328
4	0.7490	0.8966
5	0.7570	0.7993
6	0.9174	0.9106
7	0.9979	0.9832

Tabela 2: Precisão e *recall* por classe do classificador baseado em MLP.

2.2.1 Análise dos erros

Foram selecionados alguns casos de erro, onde é exibido a amostra que foi classificada erroneamente, uma representante da classe do falso positivo, e uma representante da classe real daquela amostra, seguido pelas probabilidades daquele dado pertencer a cada uma das classes.

Nas Figuras 12 e 13, observa-se que houve grande desafio para classificar a classe da amostra, e a classe correta se encontra como a 2ª maior probabilidade, sem estar consideravelmente para trás.

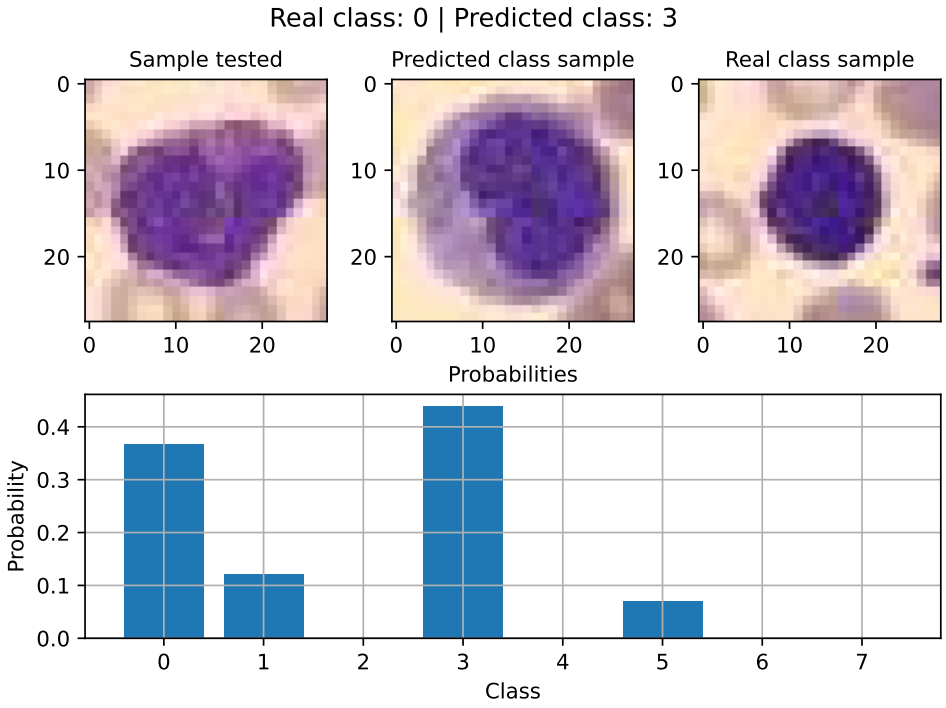


Figura 12: Análise de um caso de erro de classificação.

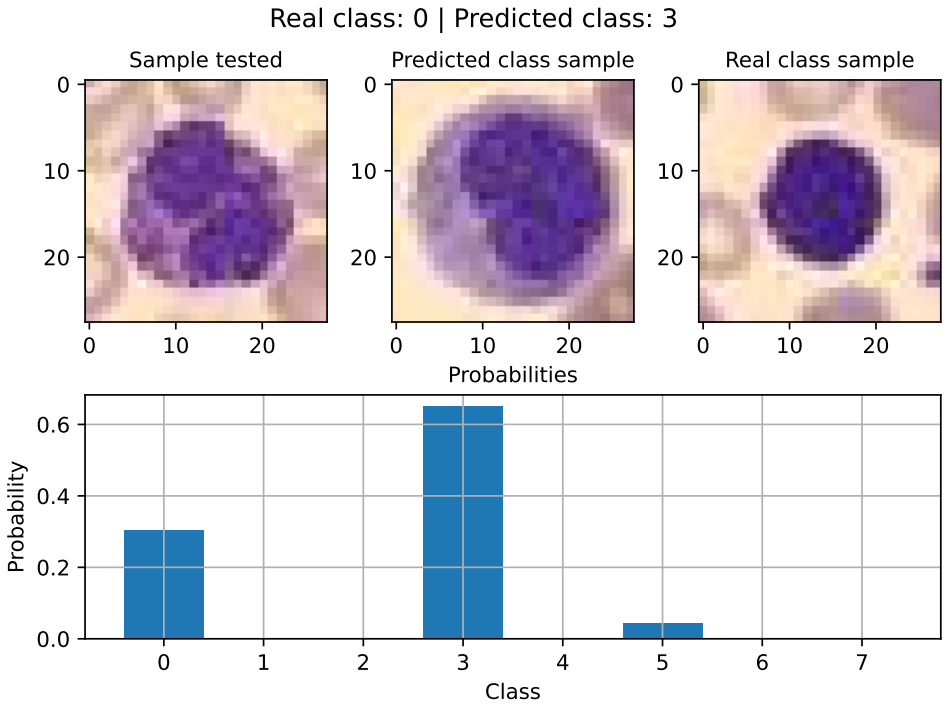


Figura 13: Análise de um caso de erro de classificação.

Já para as amostras das Figuras 14 e 15, o classificador errou com grande certeza, apresentando mais de 0,8 de probabilidade de ser a classe errada, mostrando uma grande falha na classificação. Porém, a classe correta continuou sendo a 2ª mais provável.

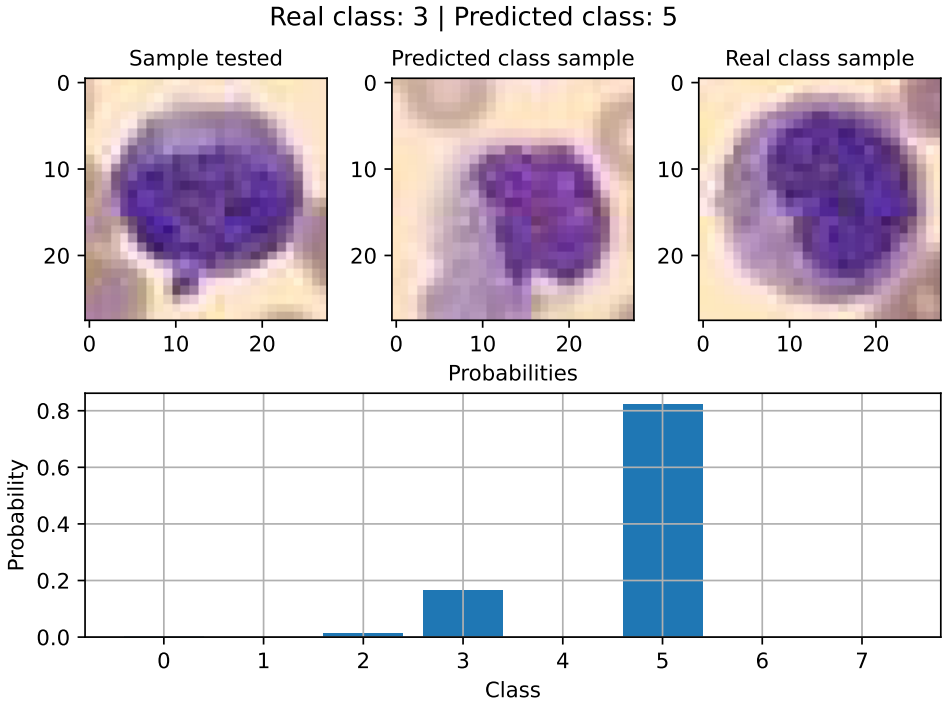


Figura 14: Análise de um caso de erro de classificação.

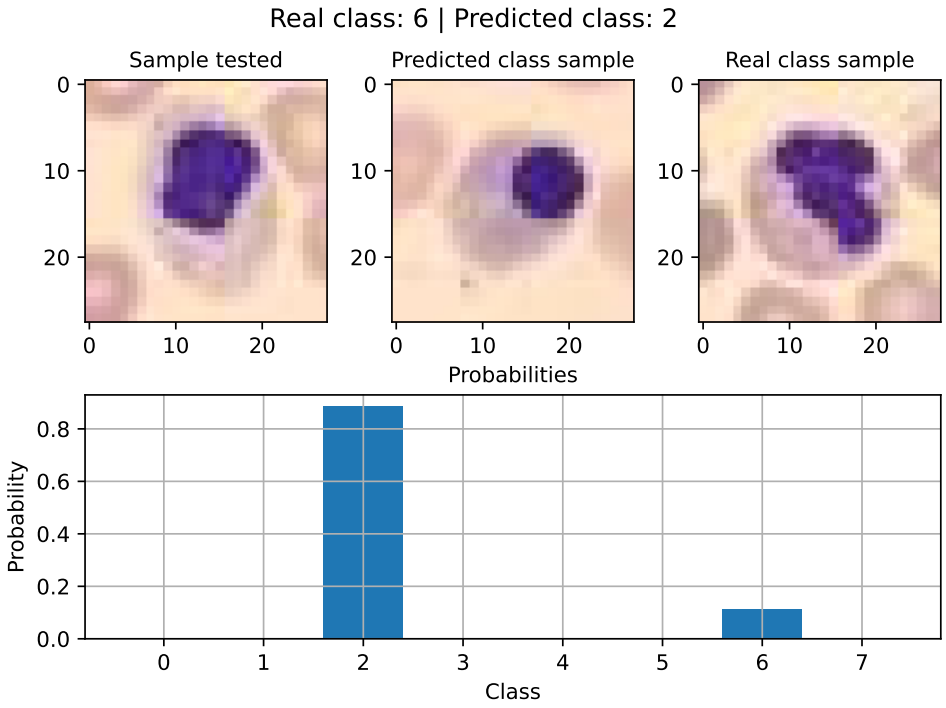


Figura 15: Análise de um caso de erro de classificação.

Já para a classificação do dado da Figura 16, houve grande certeza que a amostra era da classe 1, apresentando na sequência maiores probabilidades de pertencer à classe 2 e 7, respectivamente. Porém, a classe correta, 6, foi a 4ª mais provável, se mostrando como uma amostra muito desafiadora, o que pode ser percebido visualmente, por apresentar um padrão muito diferente do visto para a classe 6 na Figura 1.

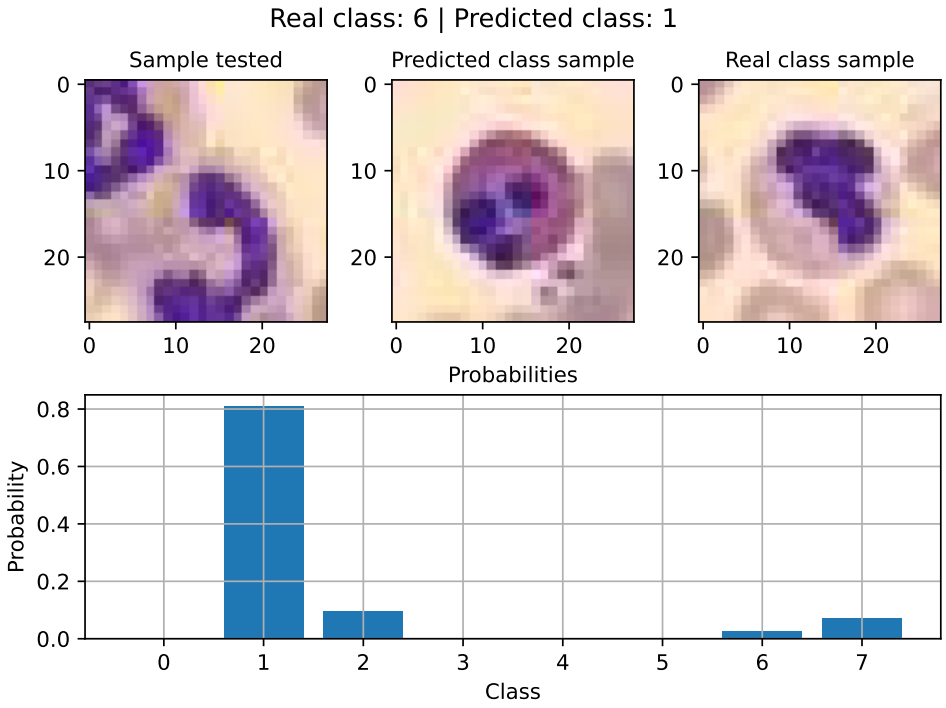


Figura 16: Análise de um caso de erro de classificação.

3 CNN Simples

3.1 Busca do melhor modelo

– Número de *kernels*: [`8`; `16`; `32`; `64`; `128`]

– Tamanho dos *kernels*: [`3x3`; `5x5`; `7x7`; `9x9`]

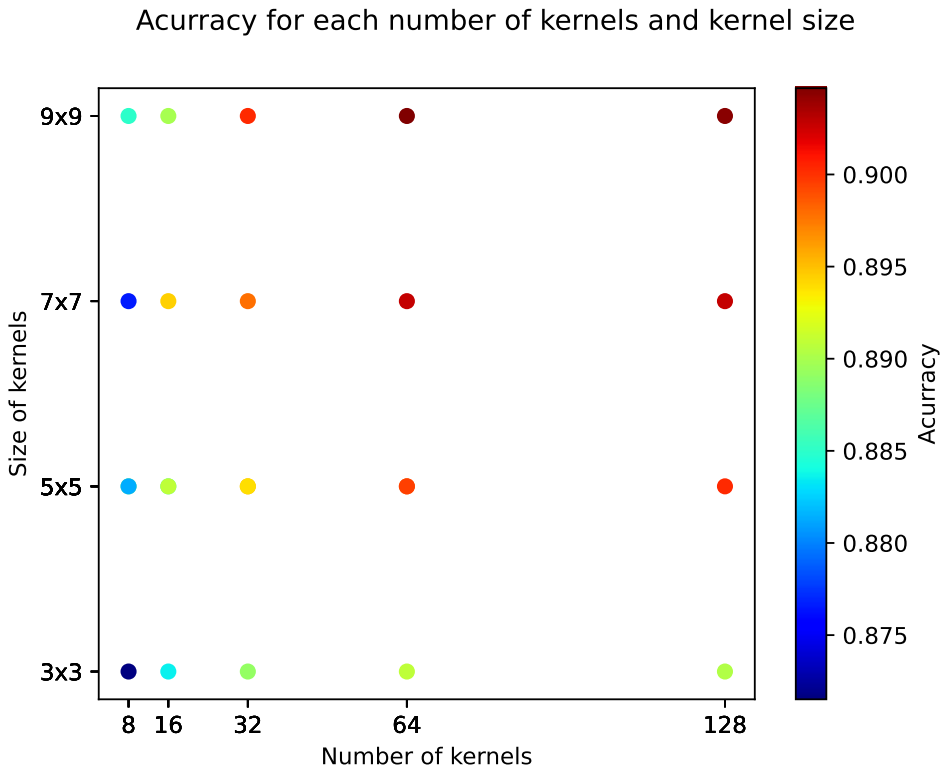


Figura 17: Acurácia e perda com a variação do algoritmo de otimização.

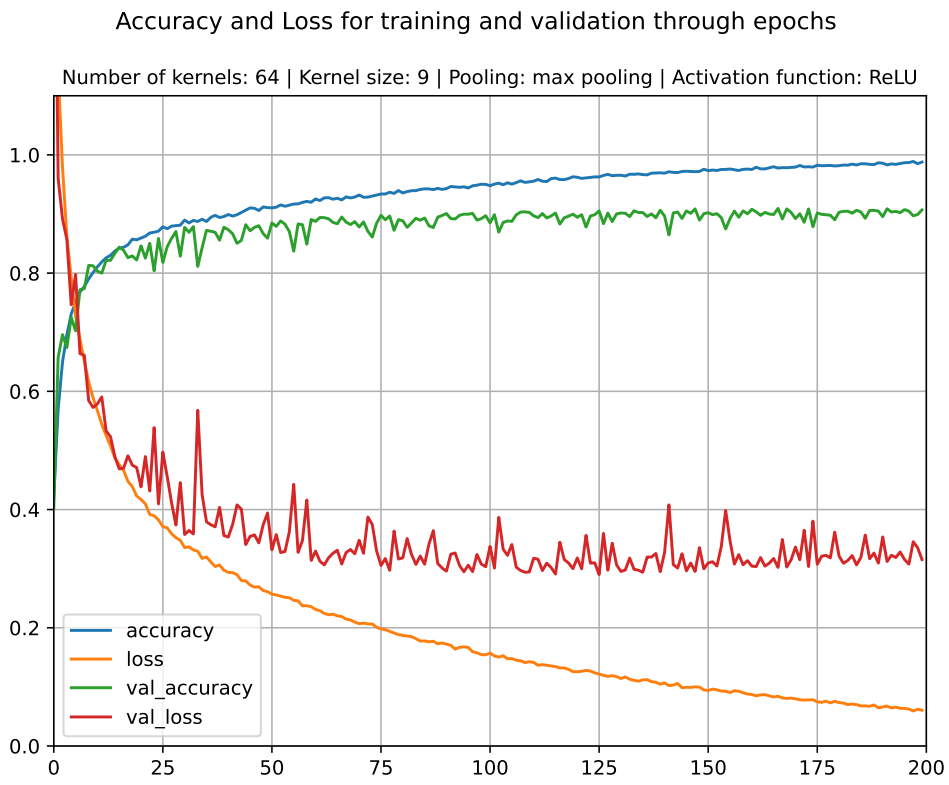


Figura 18: Acurácia e perda com a variação do algoritmo de otimização.

3.2 Melhor modelo

Quantidade de parâmetros: 115,976

$$Accuracy = 0.9088 \tag{3}$$

$$BA = 0.8905 \tag{4}$$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	193	2	1	34	4	10	0	0
1	2	610	0	4	1	1	6	0
2	3	2	286	9	2	3	4	2
3	23	12	14	478	13	21	18	0
4	9	0	6	12	214	1	1	0
5	4	0	4	50	6	219	1	0
6	1	7	2	14	2	0	640	0
7	0	0	0	0	0	0	1	469

Tabela 3: Matriz de confusão do classificador baseado em CNN *Shallow*.

Classe	Precisão	*Recall*
0	0.7910	0.8213
1	0.9776	0.9637
2	0.9196	0.9137
3	0.8256	0.7953
4	0.8807	0.8843
5	0.7711	0.8588
6	0.9610	0.9538
7	0.9979	0.9958

Tabela 4: Matriz de confusão do classificador baseado em CNN *Shallow*.

3.2.1 Análise dos erros

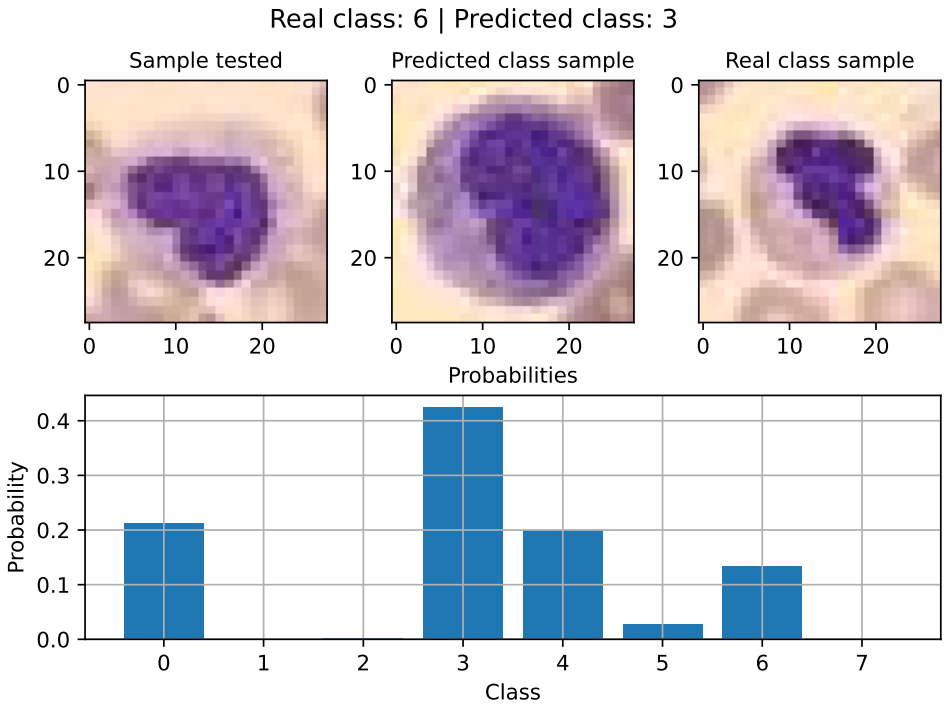


Figura 19: Análise de um caso de erro de classificação.

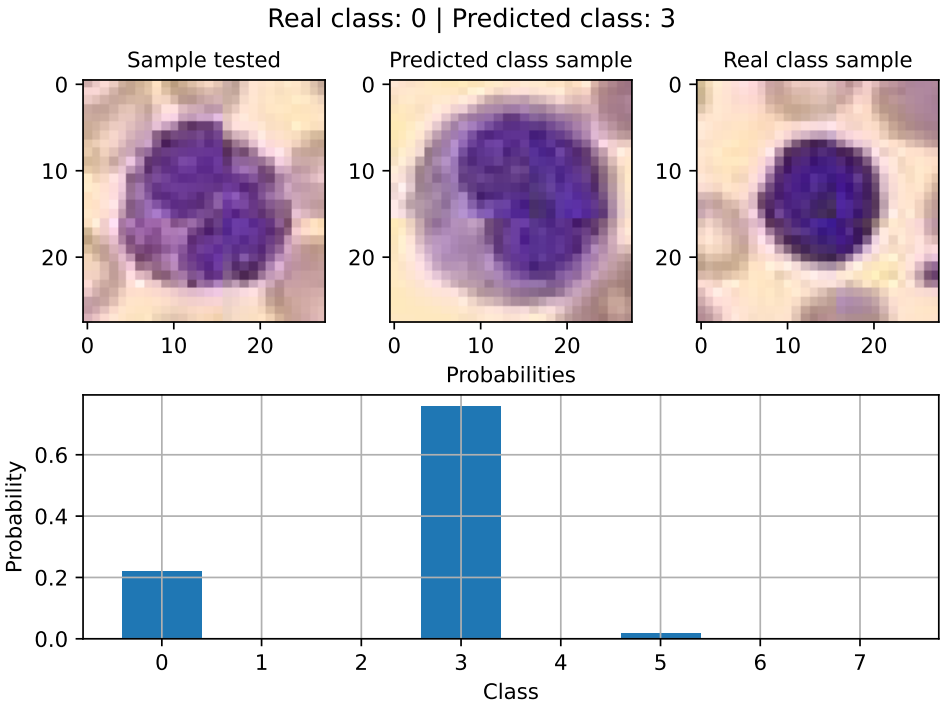


Figura 20: Análise de um caso de erro de classificação.

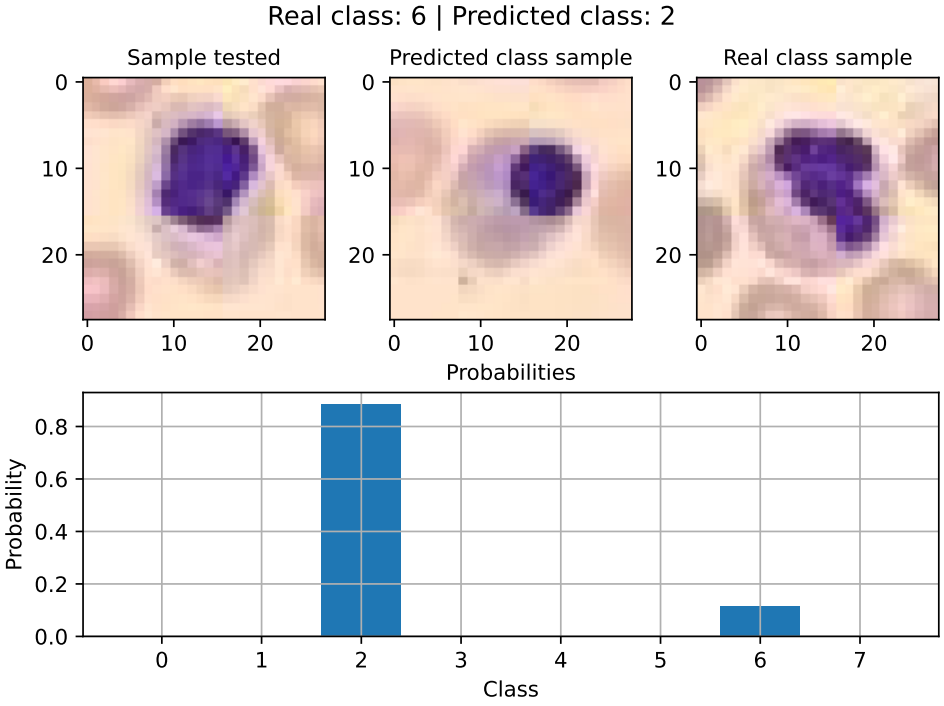


Figura 21: Análise de um caso de erro de classificação.

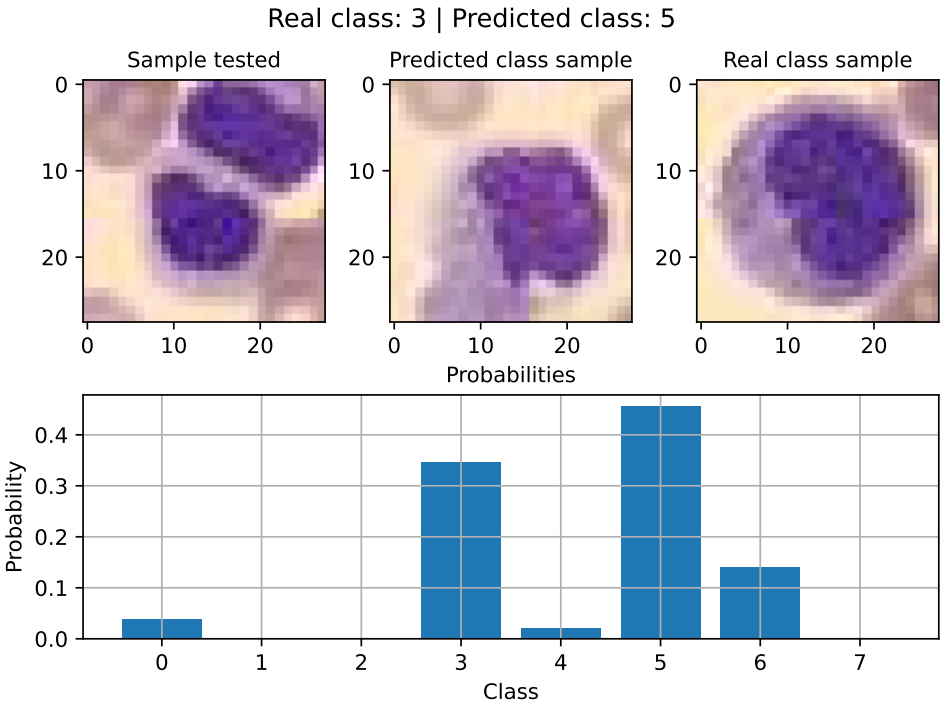


Figura 22: Análise de um caso de erro de classificação.

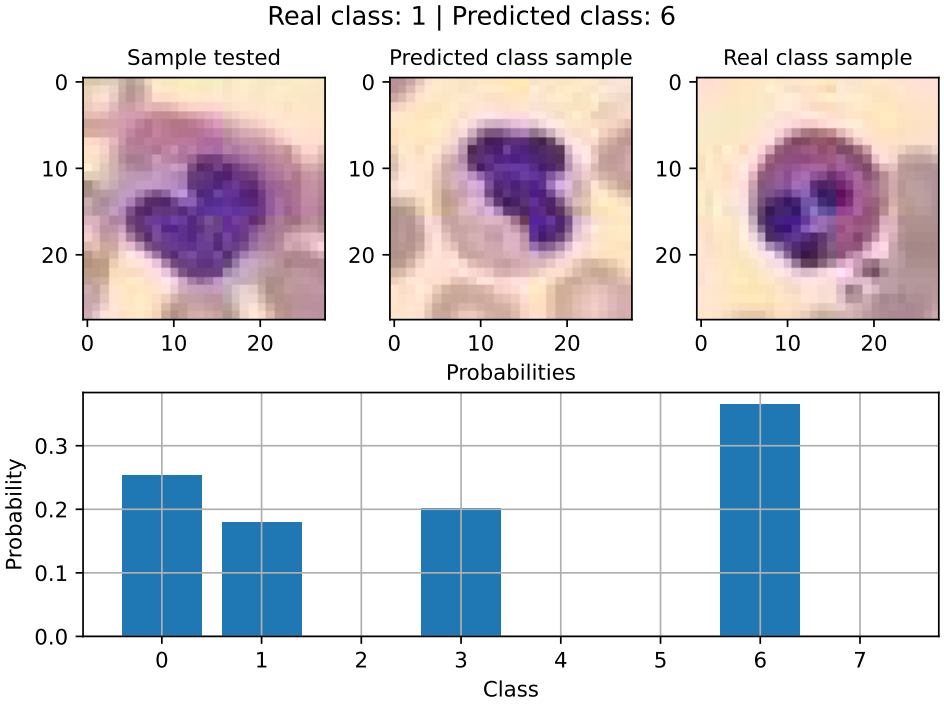


Figura 23: Análise de um caso de erro de classificação.

4 CNN Profunda

Referências

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. [S.l.]: O'Reilly Media, 2019. ISBN 9781492032618.

WILSON, A. C. et al. *The Marginal Value of Adaptive Gradient Methods in Machine Learning*. [S.l.]: arXiv: 1705.08292, 2018.

Anexos

Códigos fonte

Todos os códigos fonte e arquivos de dados utilizados para a elaboração deste documento podem ser encontrados no repositório do GitHub no link: github.com/toffanetto/ia048.