



**UNICAMP**

Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

**IA048 – Aprendizado de Máquina**

29 de maio de 2024

Docentes: Levy Boccato & Romis Attux

Discente:

– Gabriel Toffanetto França da Rocha – 289320

## Atividade 3 – Redes Neurais

---

### Sumário

<b>1</b>	<b>Apresentação dos dados</b>	<b>2</b>
<b>2</b>	<b>MLP</b>	<b>4</b>
2.1	Busca do melhor modelo . . . . .	4
2.1.1	Número de neurônios . . . . .	4
2.1.2	Função de ativação . . . . .	7
2.1.3	<i>Dropout</i> . . . . .	8
2.1.4	Otimizador . . . . .	10
2.2	Análise do melhor modelo . . . . .	12
2.2.1	Análise dos erros . . . . .	13
<b>3</b>	<b>CNN Simples</b>	<b>16</b>
3.1	Busca do melhor modelo . . . . .	16
3.2	Análise do Melhor modelo . . . . .	18
3.2.1	Análise dos erros . . . . .	19
3.3	Aplicando uma entrada 64x64 à camada convolucional . . . . .	21
<b>4</b>	<b>CNN Profunda</b>	<b>23</b>
4.1	Análise do desempenho . . . . .	23
4.2	Análise de erros . . . . .	24
<b>5</b>	<b>Conclusão</b>	<b>28</b>
	<b>Referências</b>	<b>30</b>
	<b>Anexos</b>	<b>31</b>

---

# 1 Apresentação dos dados

A base de dados BloodMNIST é formada por um total de 15380 imagens de células sanguíneas divididas 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.

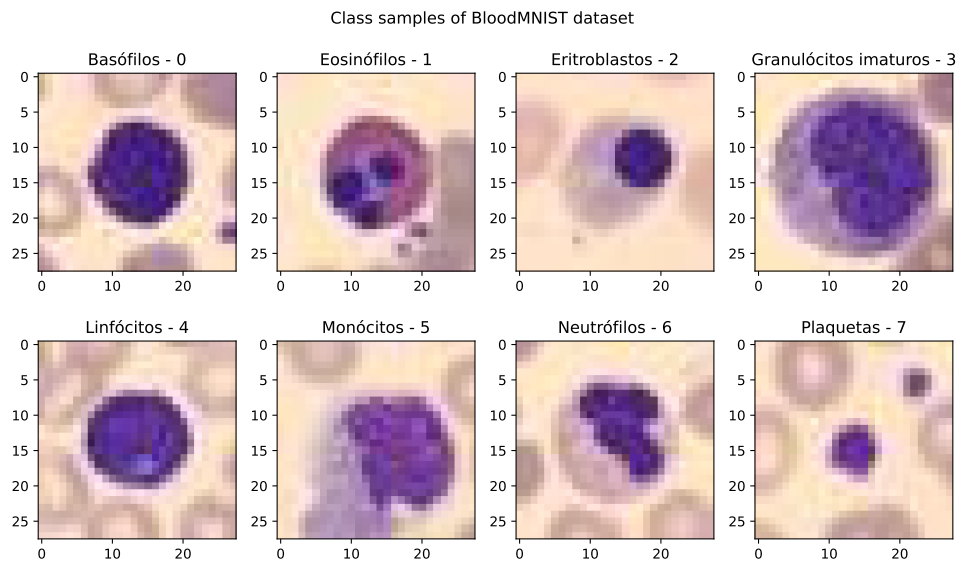


Figura 1: Amostras das classes do *dataset* BloodMNIST.

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 amostras 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 um número maior de 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*.

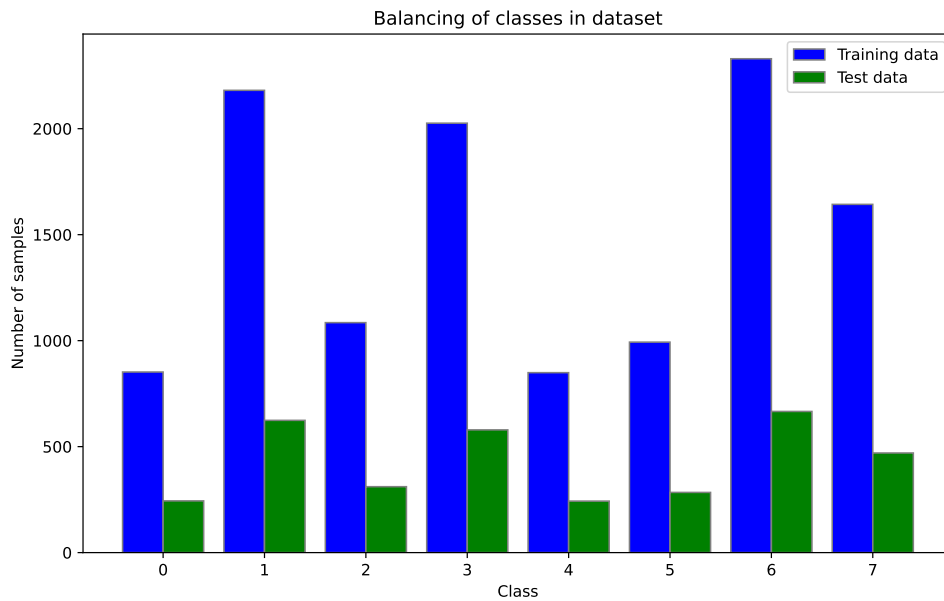


Figura 2: Balanço das classes nos *datasets* de treinamento e teste.

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.

Os dados de treinamento, validação e teste foram normalizados em uma escala de 0 a 1, de forma a facilitar o processo de ajuste de pesos via gradiente.

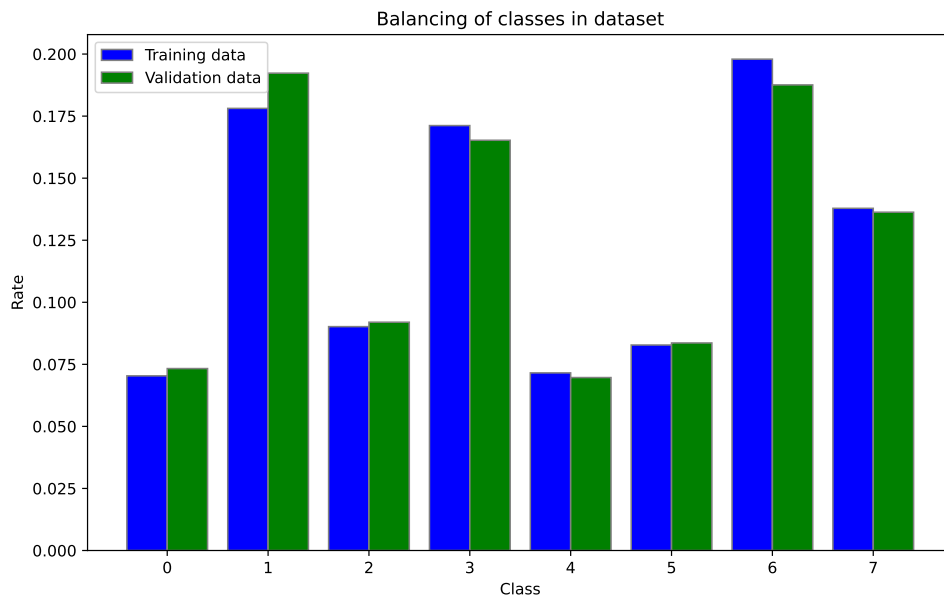


Figura 3: Balanço das classes nos conjuntos de dados de treinamento e validação cruzada do tipo *holdout*.

## 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 (??). 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*, e foi utilizada a entropia cruzada como função de custo, acompanhando também a métrica de acurácia durante o treinamento.

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 a quantidade que obteve maior acurácia, testou-se as funções de ativações candidatas 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.

Conjunto de hiper-parâmetros variados durante a busca:

- Número de neurônios: [ 256; 512; 1024; 2048; 4096 ]
- Função de ativação: [ ReLU; Sigmoid ]
- *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 acurácia, 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.

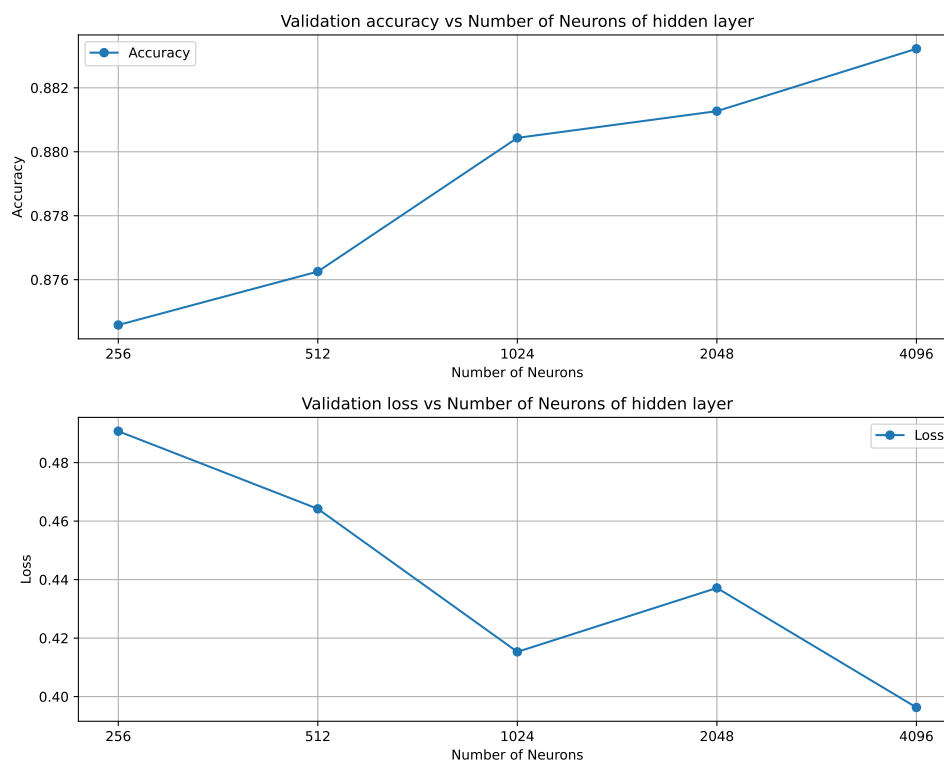
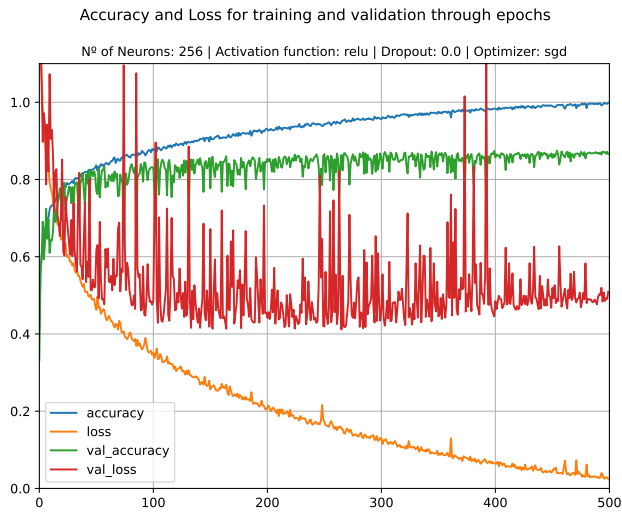
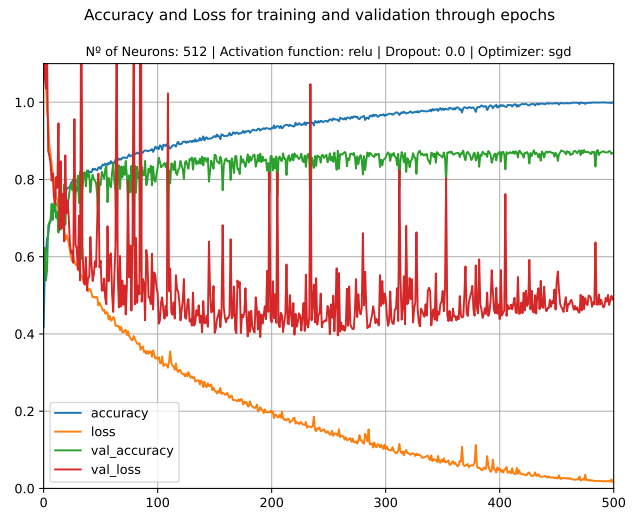


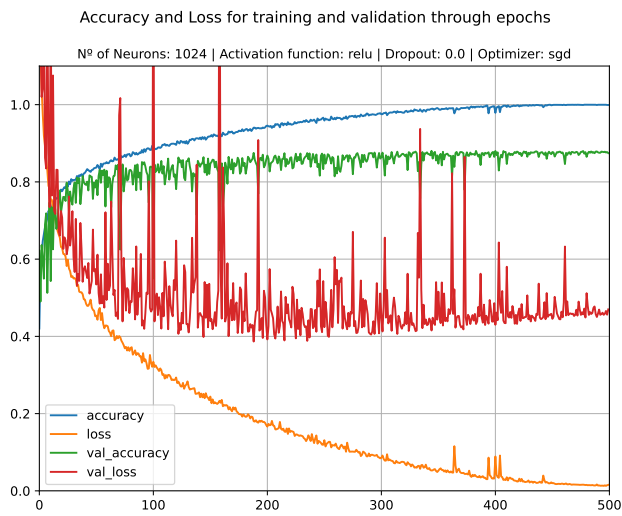
Figura 4: Acurácia e perda com a variação do número de neurônios da camada intermediária.



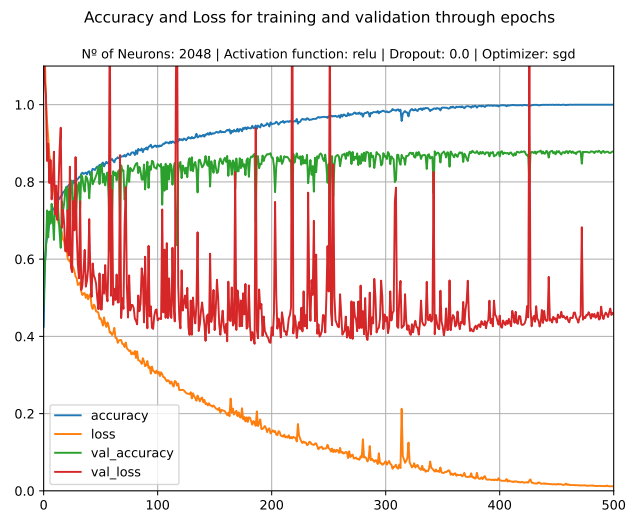
(a) 256 neurônios.



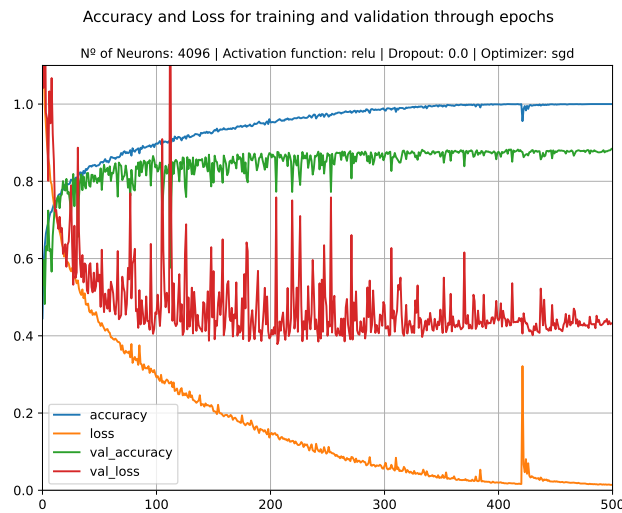
(b) 512 neurônios.



(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 sigmoide, 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 sigmoide 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.

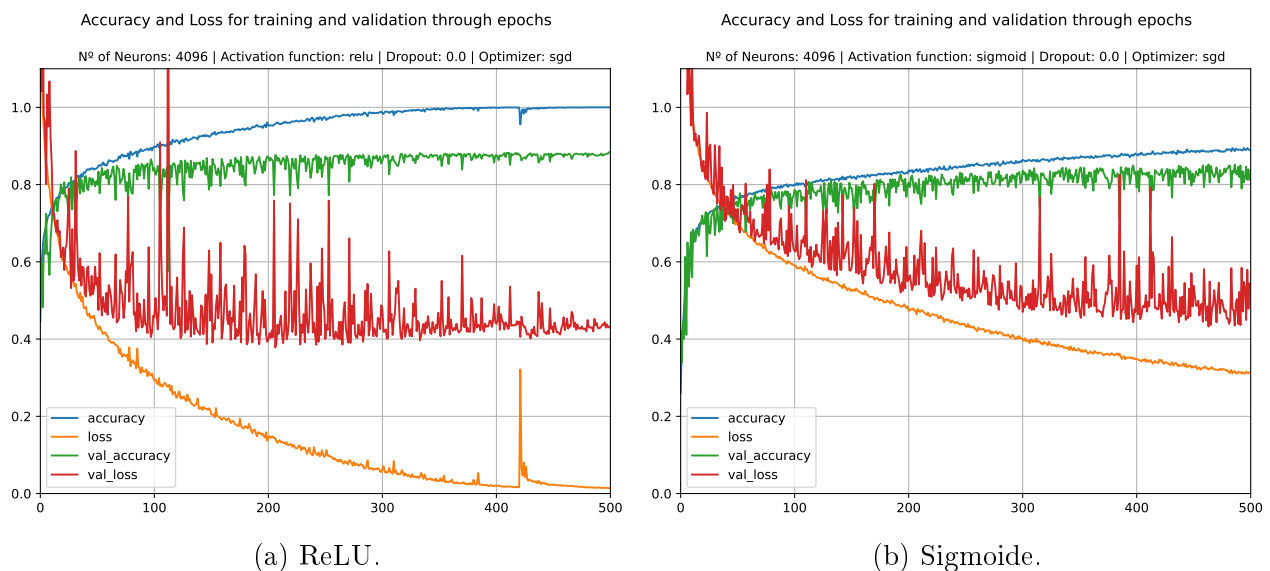


Figura 6: Evolução da perda e acurácia de treinamento e validação com as épocas de treinamento para cada função de ativação candidata.

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.

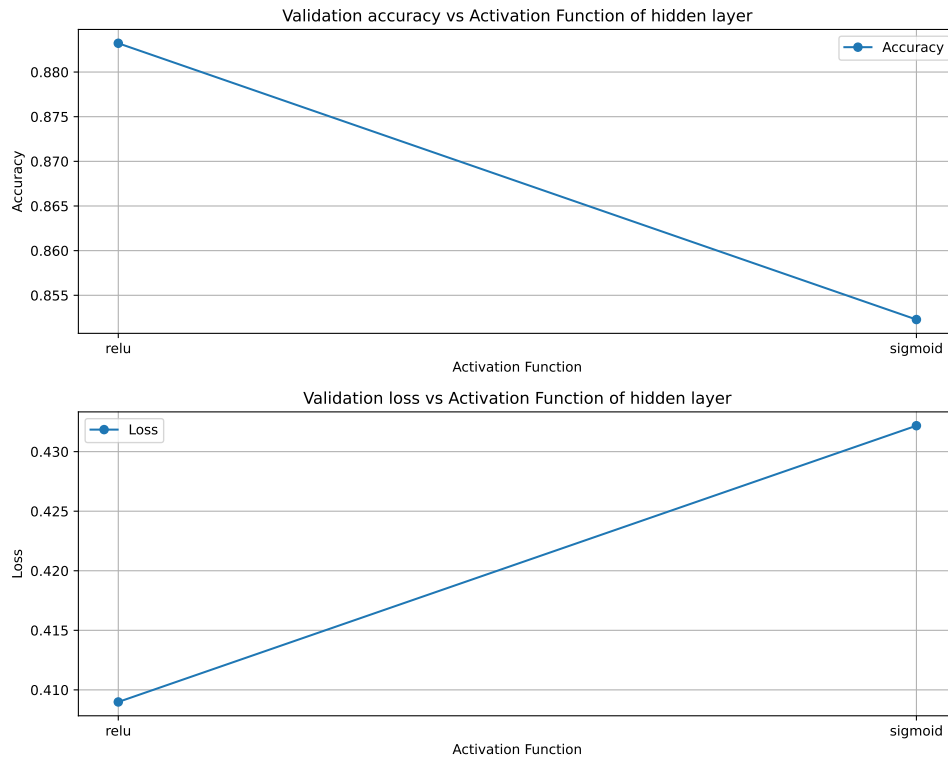


Figura 7: Acurácia e perda com a variação da função de ativação dos neurônios da camada intermediária.

### 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.



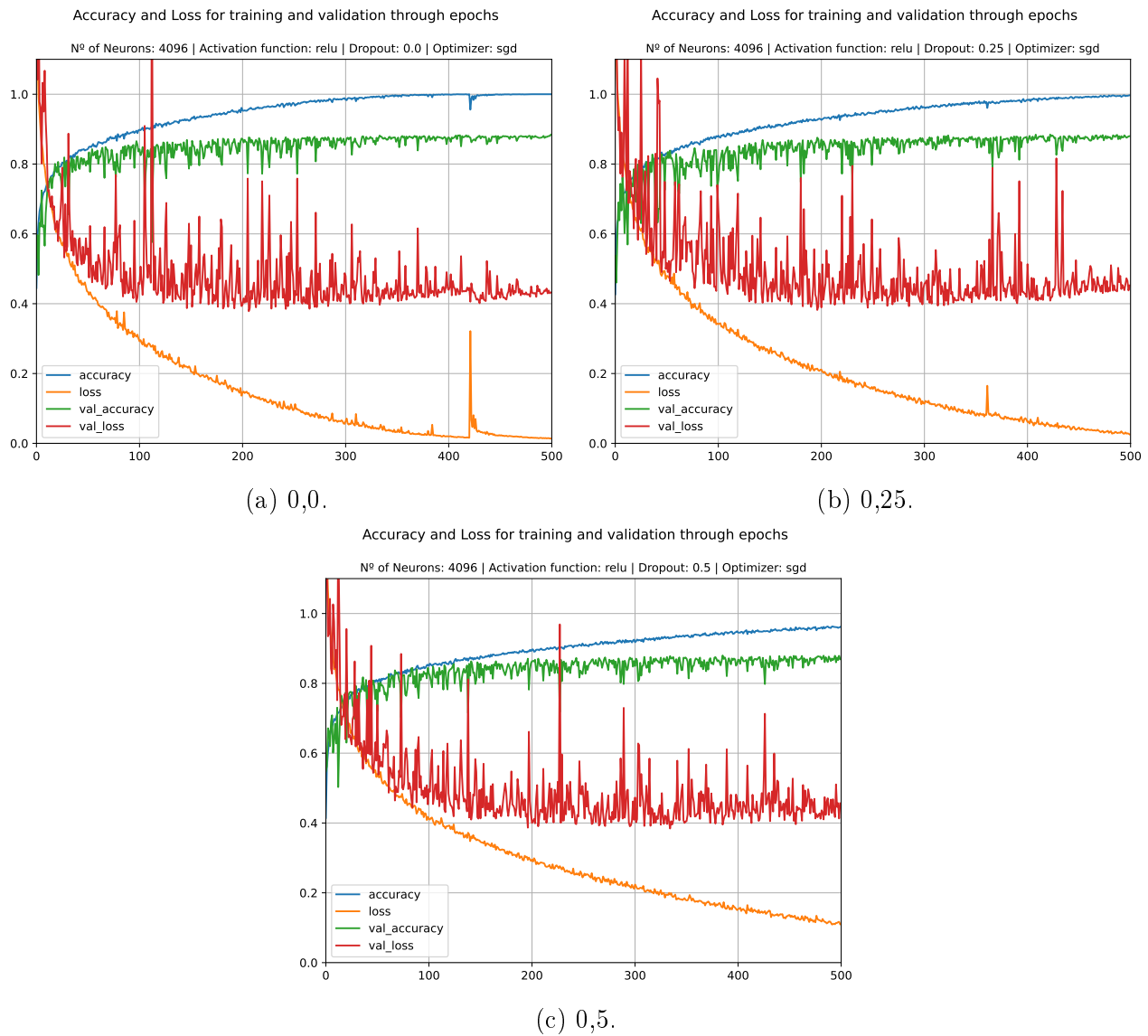


Figura 8: Evolução da perda e acurácia de treinamento e validação com as épocas de treinamento para diferentes taxas de *dropout*.

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

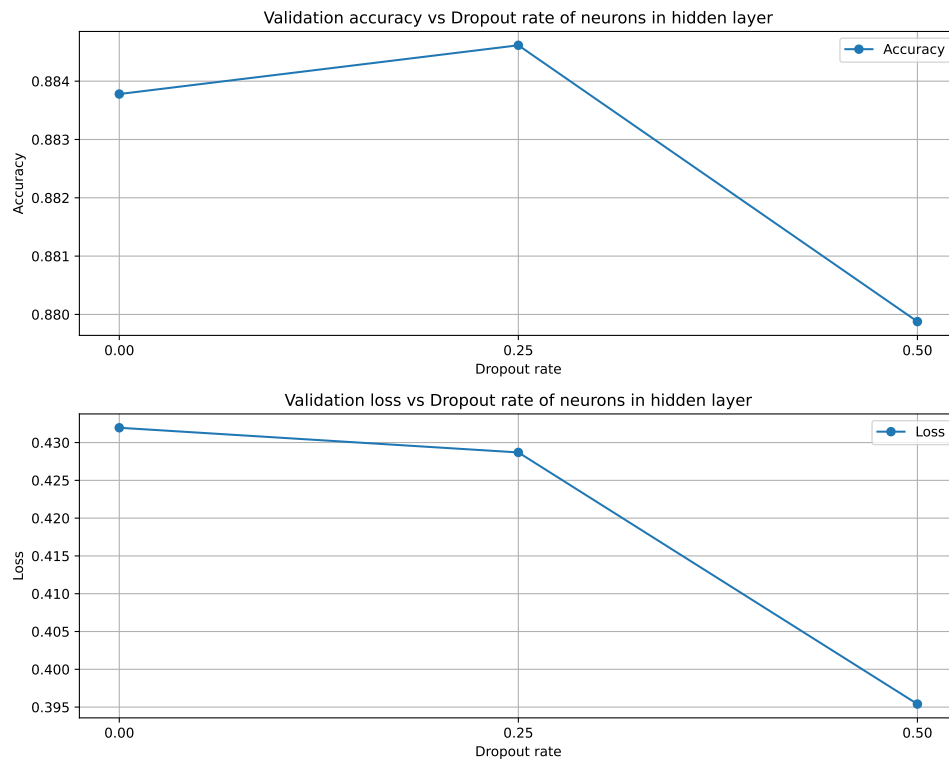


Figura 9: Acurácia e perda com a variação da taxa de *dropout* dos neurônios da camada intermediária.

#### 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.

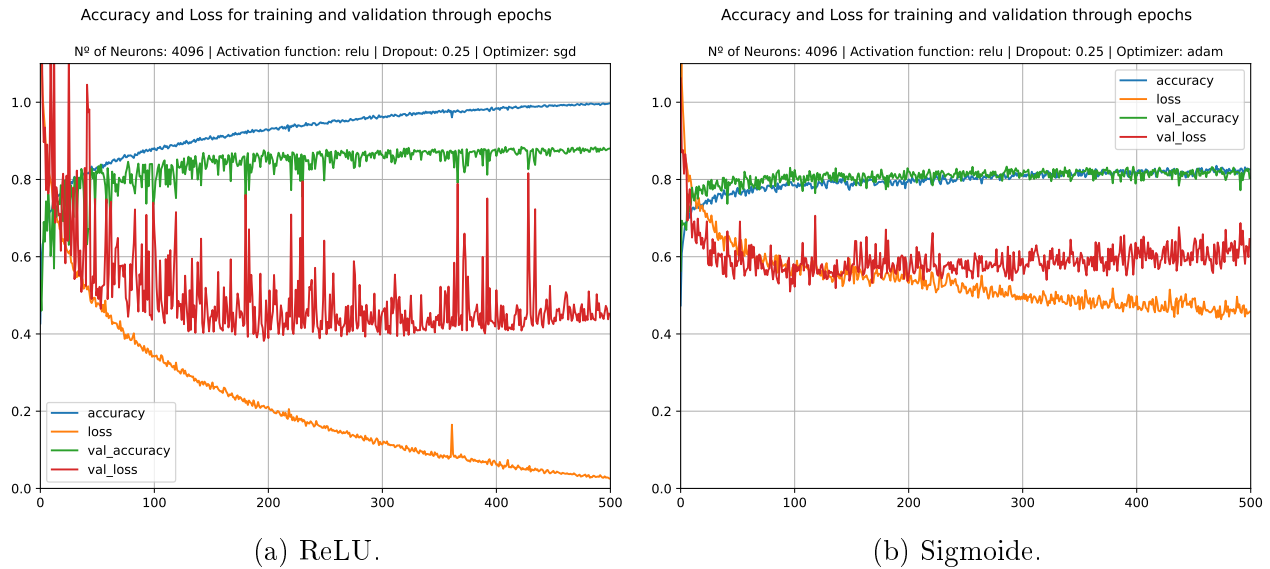


Figura 10: Evolução da perda e acurácia de treinamento e validação com as épocas de treinamento para os otimizadores analisados.

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 (??), 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.

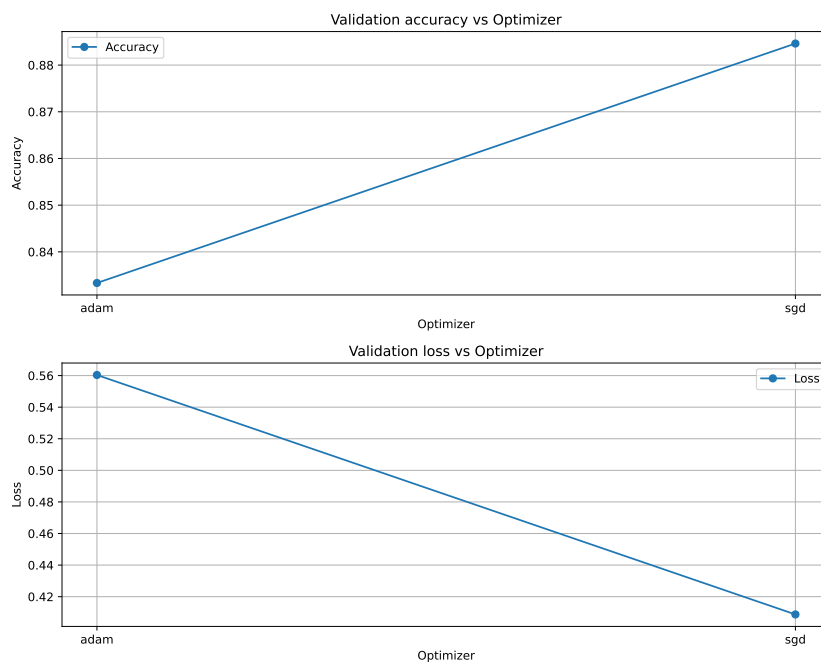


Figura 11: Acurácia e perda com a variação do algoritmo de otimização.

## 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:

$$Accuracy = 0.8705 \quad (1)$$

$$BA = 0.8437 \quad (2)$$

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.

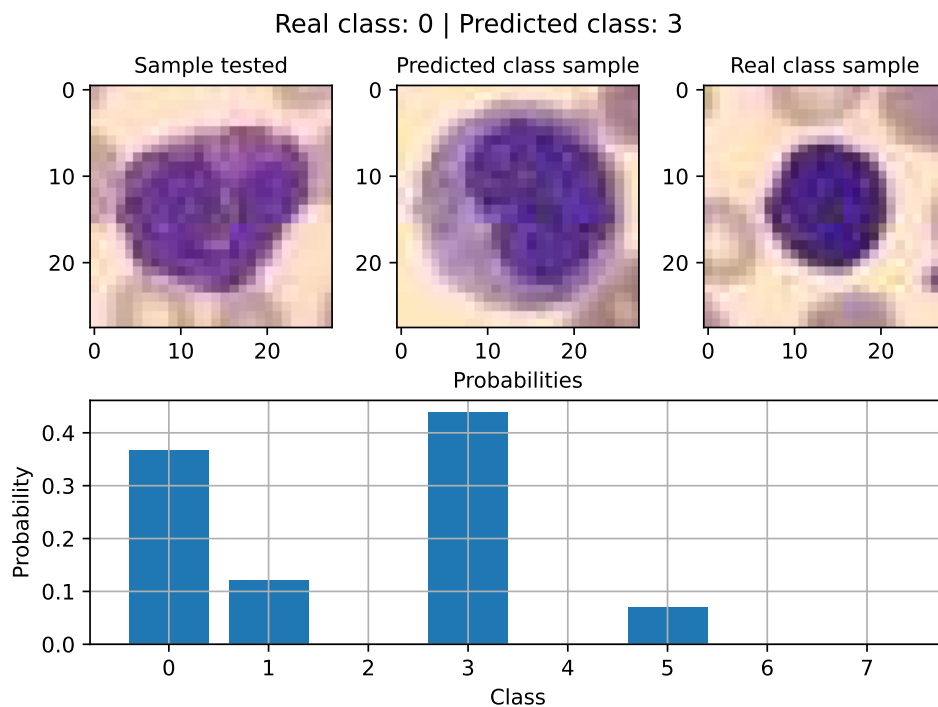


Figura 12: Análise de um caso de erro de classificação.

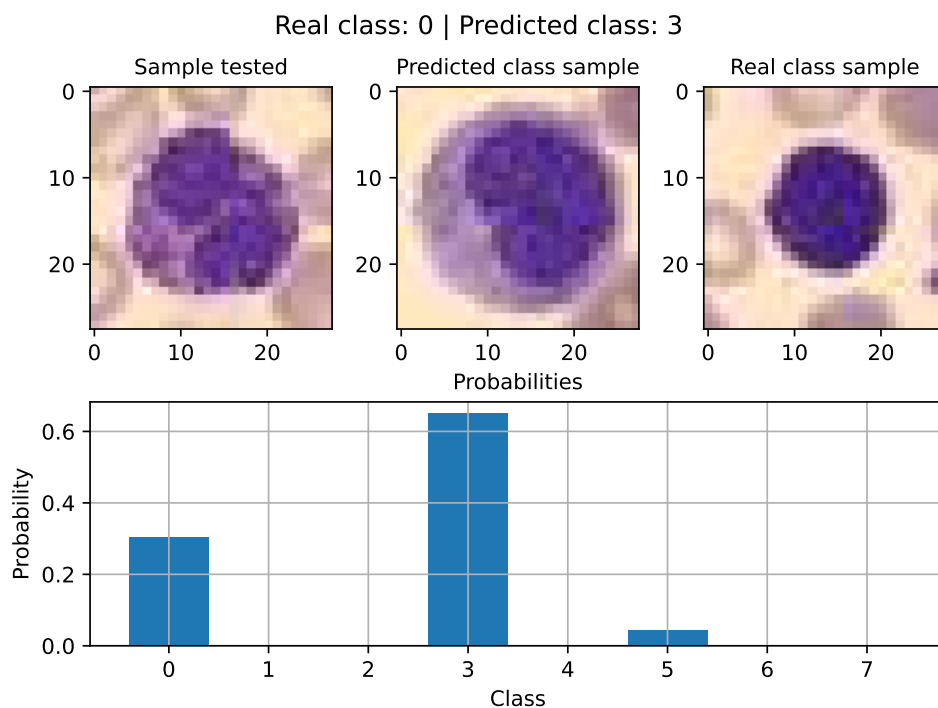


Figura 13: Análise de um caso de erro de classificação.

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 2<sup>a</sup> mais provável.

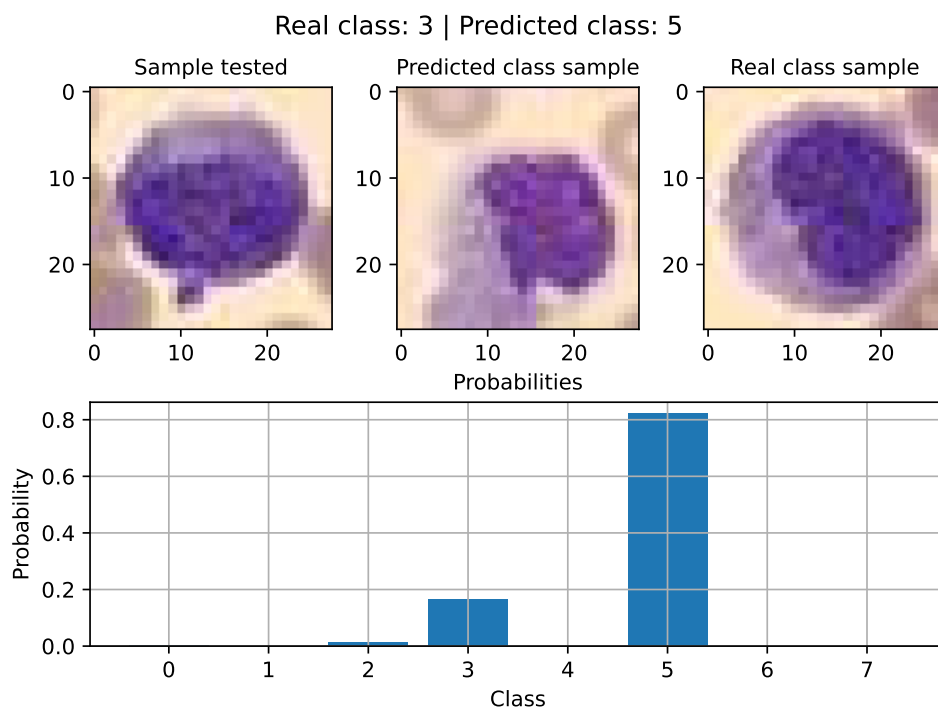


Figura 14: Análise de um caso de erro de classificação.

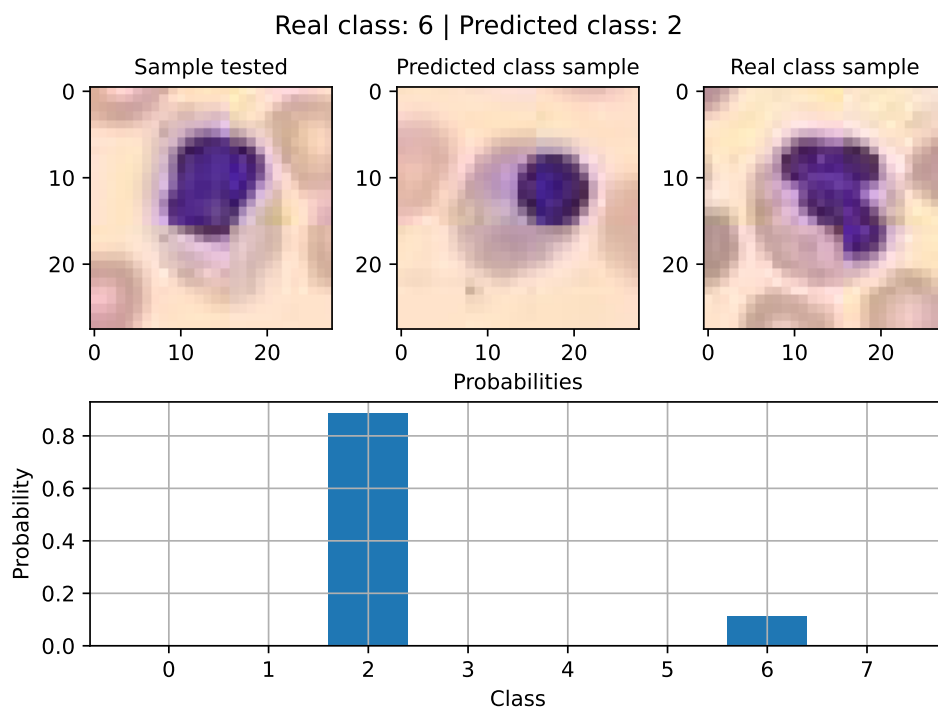


Figura 15: Análise de um caso de erro de classificação.

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.

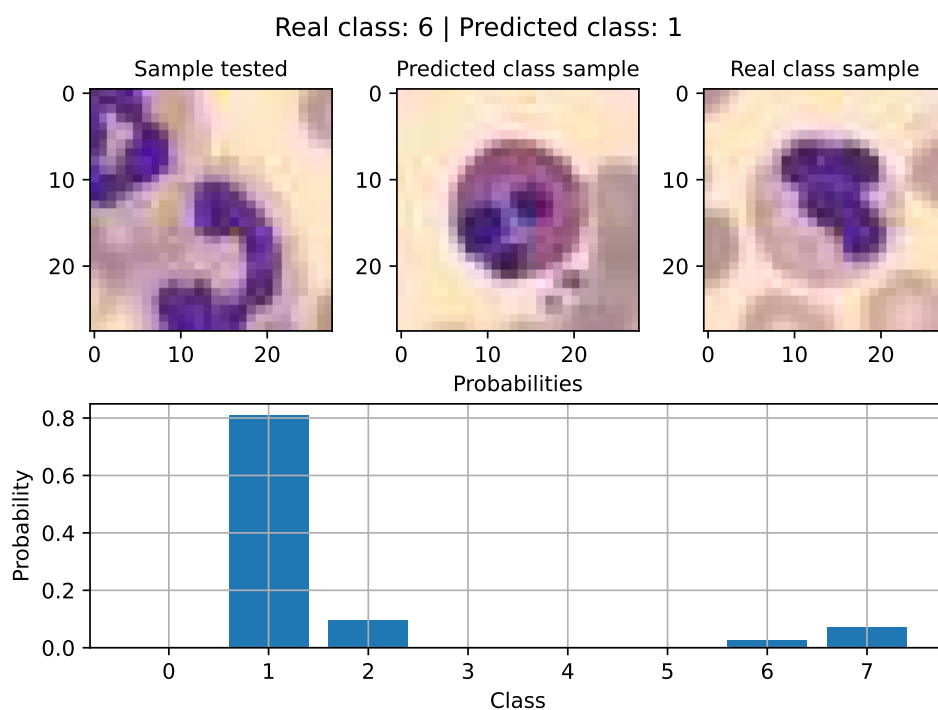


Figura 16: Análise de um caso de erro de classificação.

### 3 CNN Simples

Para a implementação a rede neural convolucional (CNN) com arquitetura rasa (*shallow*), foi utilizada uma rede possuindo apenas uma camada convolucional, uma camada de *pooling* e uma camada de saída com função de ativação *softmax*.

Com base no visto anteriormente, a função de ativação foi mantida como ReLU, o otimizador como SGD, entropia cruzada como função de custo e não foi empregado o uso de *dropout*, devido ao tamanho da rede. A camada de *pooling* escolhida utiliza a operação de máximo, operação selecionada a partir de um teste prévio, que mostrou maior acurácia da rede utilizando a operação de máximo frente à operação de média. O *kernel* de *pooling* foi mantido em um tamanho pequeno, de 2x2, pois como a rede é rasa, houve a intenção de preservar a maior quantidade de dados possível, em vista que não há expansão do campo receptivo.

#### 3.1 Busca do melhor modelo

Para realizar a busca do melhor modelo de CNN, foram treinadas redes para os números de *kernels* e tamanhos de *kernels* mostrados a seguir:

- Número de *kernels*: [ 8; 16; 32; 64; 128 ]
- Tamanho dos *kernels*: [ 3x3; 5x5; 7x7; 9x9 ]

Diferentemente da MLP, devido à variação de somente dois hiper-parâmetros, foi realizada a busca exaustiva completa sobre tais variáveis no intervalo estipulado, e o mesmo foi expresso em um gráfico de temperatura, mostrado na Figura 17. Observa-se que a acurácia aumenta com o aumento do número de *kernels*, e também com o aumento do tamanho do *kernel*, mas principalmente com o primeiro. Como a rede convolucional possui apenas uma camada convolucional, não existe a expansão do campo receptivo dos neurônios sob o dado de entrada através das camadas, logo, um *kernel* maior consegue captar mais padrões, e dessa forma, chegar à um desempenho melhor. Já em relação ao número de *kernels*, é esperado que possuindo mais canais, cada canal possa detectar um atributo diferente e dessa forma generalizar melhor.

Devido ao pouco ganho de acurácia da configuração com *kernel* 9x9 para 64 ou 128 *kernels*, escolheu-se a opção com menos *kernels* para simplicidade da CNN, obtendo assim o melhor configuração da CNN *Shallow*:

- Número de *kernels*: [ 64 ]
- Tamanho dos *kernels*: [ 9x9 ]



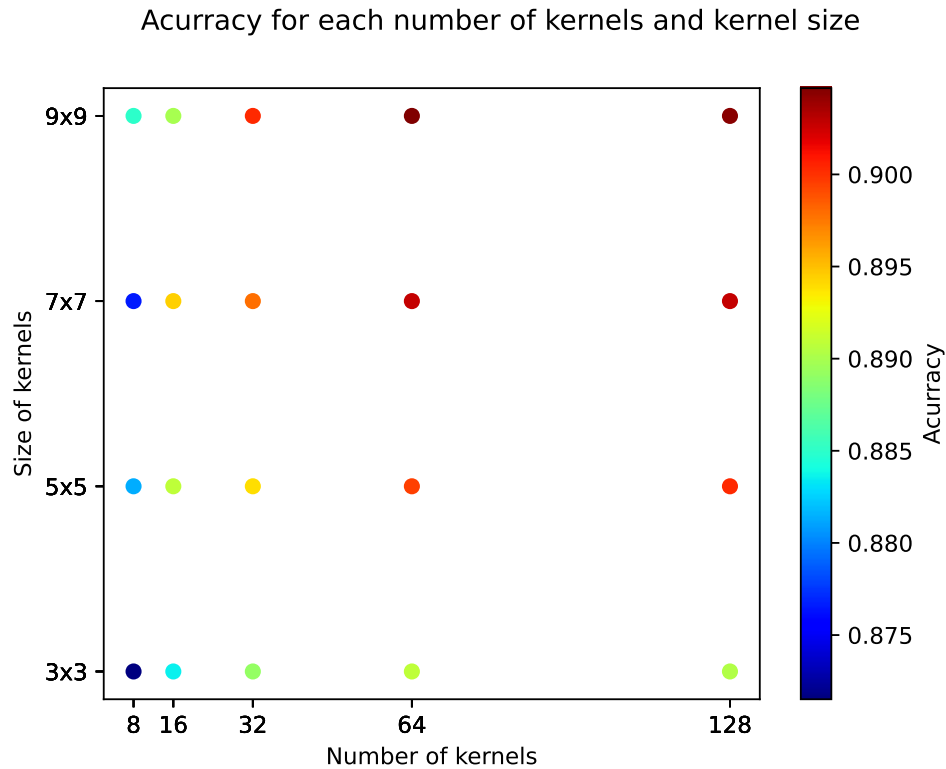


Figura 17: Acurácia do classificador de acordo com o número e tamanho dos *kernels*.

A Figura 18 mostra o treinamento da rede com melhor configuração por 200 épocas. Observa-se que a perda sob os dados de validação é bem menos errática, e não se percebe *overfitting*. Devido à saturação da acurácia, se fez satisfatório o treinamento em apenas 200 épocas, adotando de *early stopping* para concluir o treinamento da rede.



Figura 18: Acurácia e perda durante as épocas de treinamento do melhor modelo CNN *Shallow*.

### 3.2 Análise do Melhor modelo

Avaliando o desempenho do modelo com melhor configuração, o primeiro contraste que se explicita é que ele possui apenas 115,976 pesos treináveis, uma quantidade drasticamente inferior à da rede com camada densa. A acurácia e acurácia balanceada do classificador se veem em (3) e (4), apresentando um desempenho superior ao da MLP, com uma quantidade 842 vezes menor de parâmetros.

$$Accuracy = 0.9088 \quad (3)$$

$$BA = 0.8905 \quad (4)$$

A matriz de confusão do classificador baseado em CNN *Shallow* se encontra na Tabela 3. Observa-se que o perfil de classificação das classes se mantém o mesmo, porém com menos erros. A classe 3 continua sendo a mais desafiadora, porém a classe 5 passa a ter a menor precisão, sendo constantemente confundida com a classe 3. Dessa forma, se vê que o ganho de acurácia não é igual para todas classes, como visto na Tabela 4, onde algumas apresentaram um grande aumento de precisão e *recall*, já outras não possuíram grandes ganhos, como a classe 0 e 5.

	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	<i>Recall</i>
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

Analizando alguns casos de erro de classificação da rede neural convolucional, observa-se na Figura 19 um caso onde houve dúvida entre três classes, sendo a verdadeira a segunda maior probabilidade. Observando visualmente, vê-se que a amostra a ser rotulada apresenta uma disparidade com sua representante de classe, o que leva a dificuldade de se encontrar padrões.

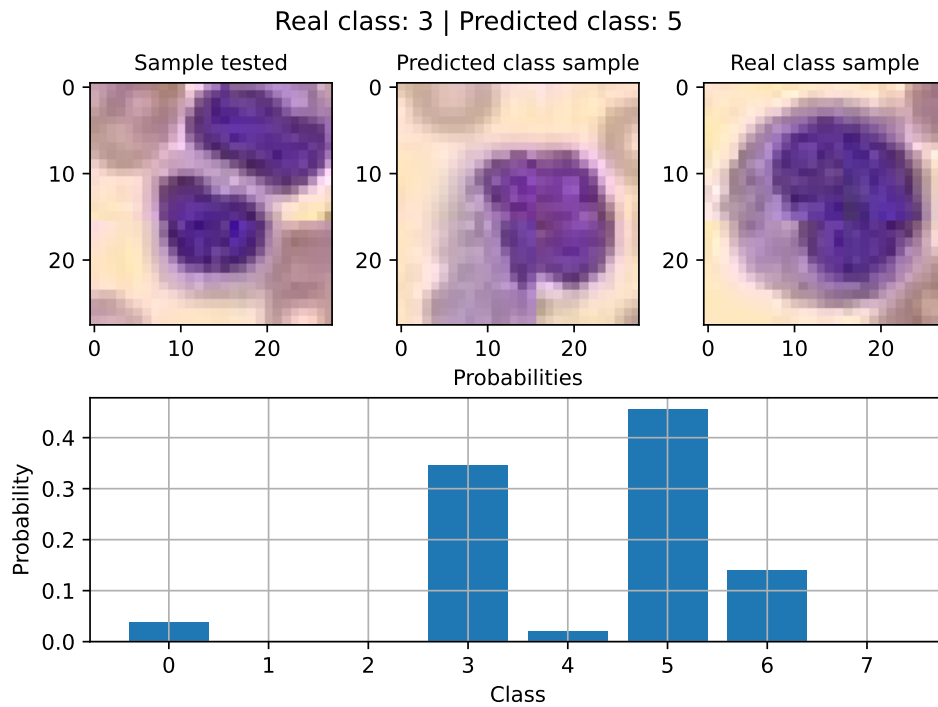


Figura 19: Análise de um caso de erro de classificação.

Já as amostras das Figuras 20 e 21, evidenciam casos muito desafiadores, onde houve dúvida entre 4 classes, e a classe verdadeira se encontrou como a última opção nos dois casos. Como no caso anterior, também se tratam de amostras que visualmente possuem um padrão diferente das suas representantes de classe, podendo se tratar de *outliers* ou amostras de baixa qualidade, onde o classificador demonstra mais dificuldade.

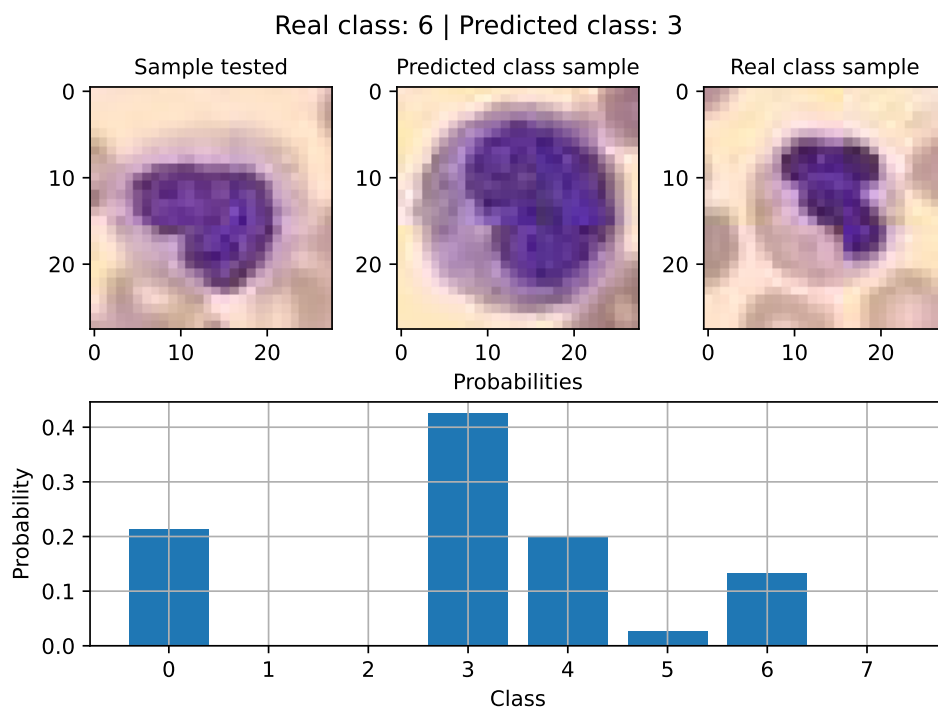


Figura 20: Análise de um caso de erro de classificação.

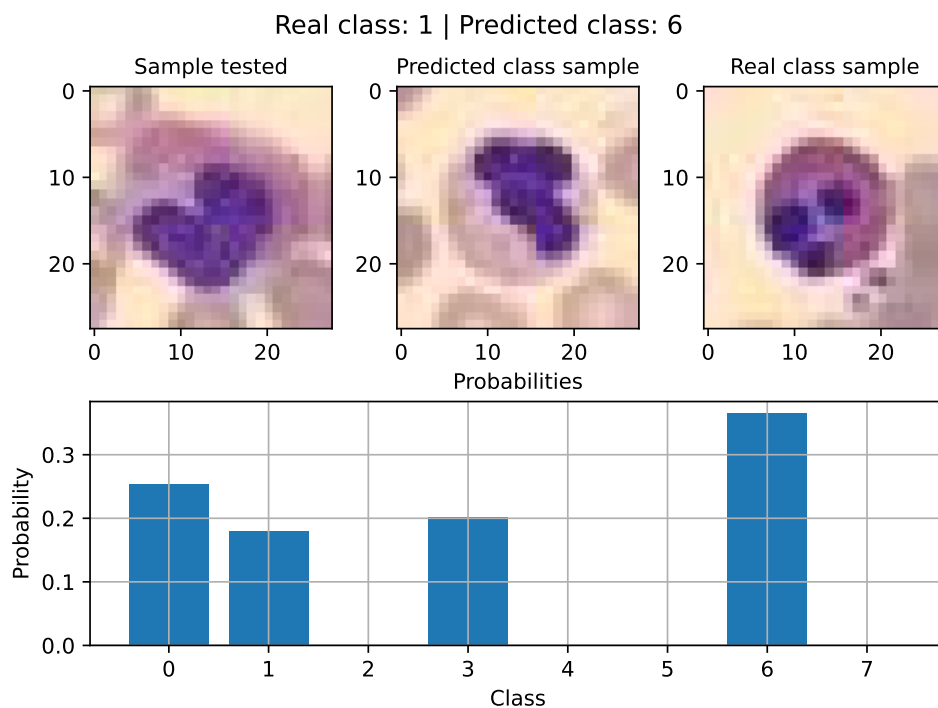


Figura 21: Análise de um caso de erro de classificação.

Como visto para a classificação utilizando a rede MLP, os dados das Figuras 22 e 23 também foram classificados erroneamente pela CNN *Shallow*, havendo uma certeza majoritária sobre a classe errada.

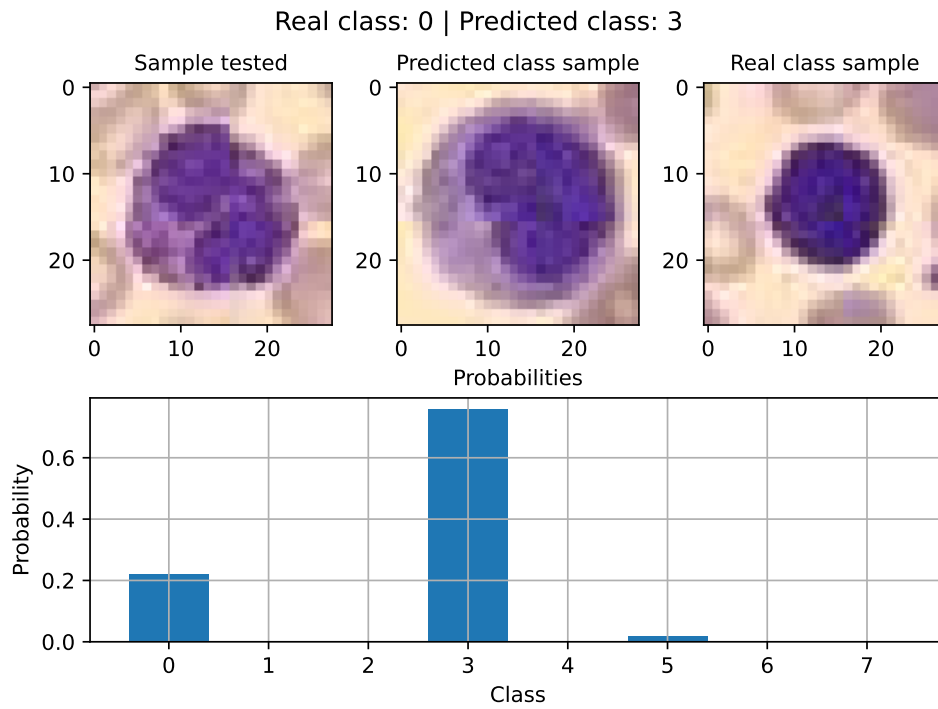


Figura 22: Análise de um caso de erro de classificação.

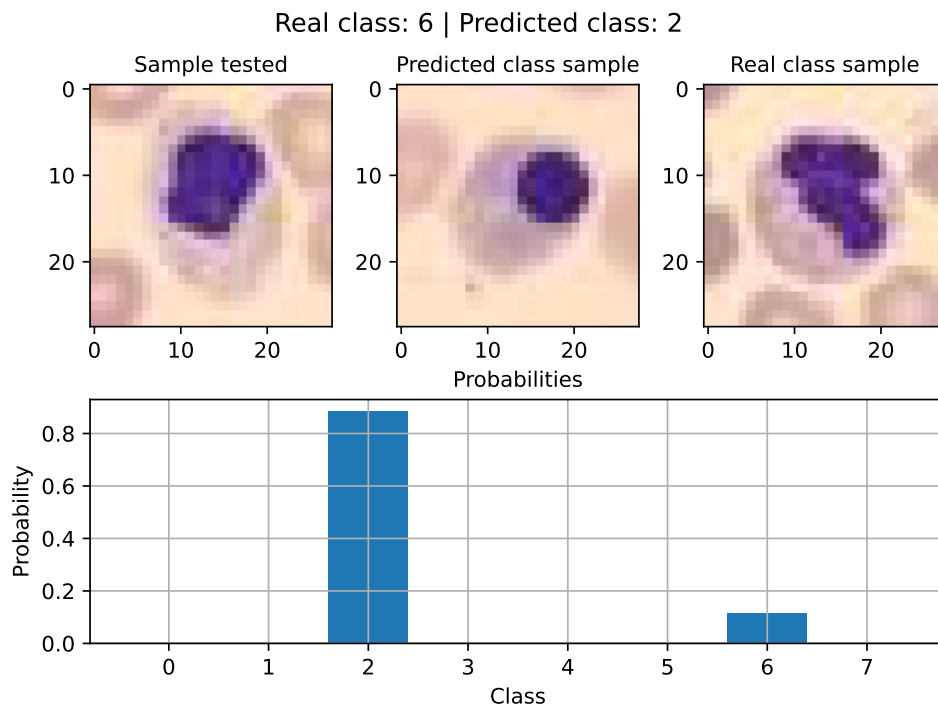


Figura 23: Análise de um caso de erro de classificação.

### 3.3 Aplicando uma entrada 64x64 à camada convolucional

Como se sabe, a camada convolucional é robusta à alteração da dimensão do dado de entrada, ou seja, a rede aqui treinada para dados 28x28, suporta o processamento de dados de dimensão, como será testado, 64x64. Porém, como uma entrada maior gera *feature maps* de

saída também maiores, a camada convolucional terá seus pesos congelados, e uma nova camada de saída *softmax* será treinada.

Ao treinar os 524,296 pesos resultantes da camada *softmax*, conservando os 15,616 pesos da camada convolucional, obteve-se o perfil de treinamento mostrado na Figura 24, e a acurácia sobre os dados de teste (5). Uma vez que apenas a camada de saída foi treinada, para um dado consideravelmente maior de entrada, a queda de acurácia não se faz tão grande, uma vez que para uma entrada com esse tamanho, além de pesos com diferença relevante, pode se fazer necessário o uso de *kernels* maiores.

$$Accuracy = 0.8620 \quad (5)$$

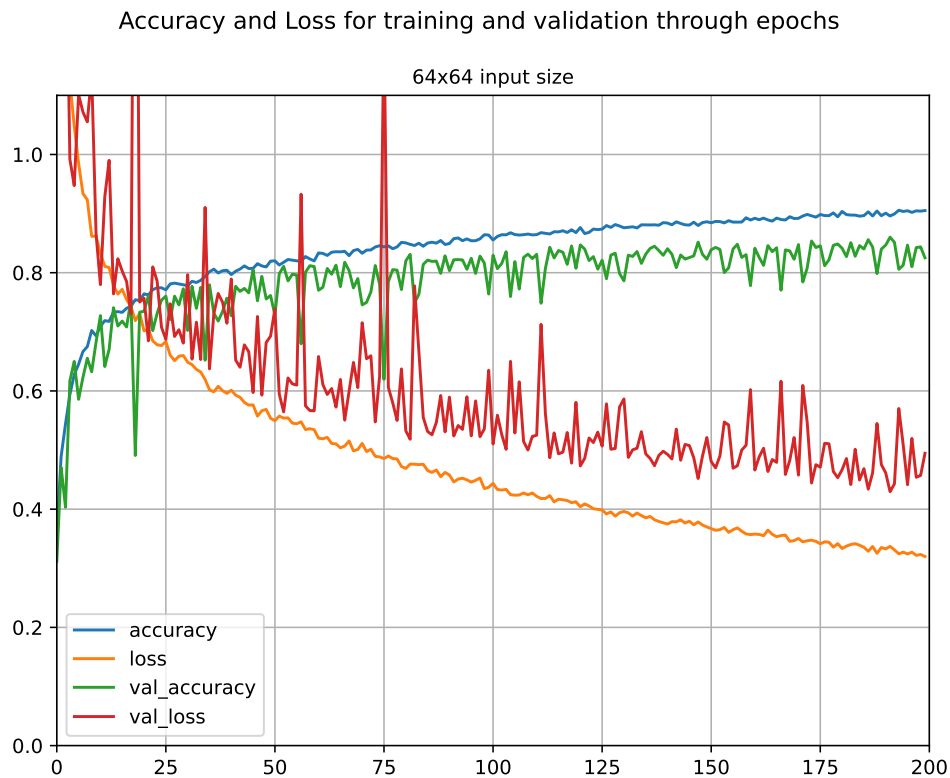


Figura 24: Acurácia e perda durante as épocas de treinamento apenas da camada de saída.

## 4 CNN Profunda

Para a solução do problema de classificação das células sanguíneas, utilizando uma rede neural convolucional profunda, foi criada uma arquitetura inspirada nas *DenseNets* (??). A rede profunda modelo foi bastante simplificada, reduzindo o número de camadas, ao utilizar apenas um *Dense Block*, e também considerando camadas convolucionais e de *pooling* com *stride* unitário, com intuito de preservar melhor a dimensão dos dados, em vista que a entrada utilizada possui tamanho 28x28. Em vista disso, o tamanho do *kernel* da camada de *pooling* também foi alterado para 2x2, conservando um canal de saída maior, e a camada de transição não foi utilizada devido ao fato de não haver a concatenação de *dense blocks*. A arquitetura da rede neural implementada é mostrada na Tabela 5.

Camada	Dimensão de saída	Descrição
Convolução	[28, 28, 3]	BN() + ReLU() + 128, $7 \times 7 + 1(S)$
<i>Pooling</i>	[14, 14, 64]	$2 \times 2 + 1(S)$ max pooling
<i>Dense Block</i>	[14, 14, 256]	BN() + ReLU() + $\begin{bmatrix} 32, 1 \times 1 + 1(S) \\ 32, 3 \times 3 + 1(S) \end{bmatrix} \times 6$
<i>Pooling</i>	[256]	<i>Global average pooling</i>
Saída	[8]	<i>softmax</i> ()

Tabela 5: Estrutura da CNN profunda baseada na *DenseNet*.

Como feito na versão original, para cada camada convolucional, é aplicado uma camada de *batch normalization*, função de ativação ReLU e então é feita a convolução dos dados. Isso se faz tanto na camada convolucional inicial, como nas que estão presentes dentro do bloco denso.

A rede possui um total de 351,048 parâmetros, sendo 347,656 deles treináveis. Observa-se que mesmo com a camada densa de saída, o número de parâmetros não aumenta expressivamente devido ao uso da camada de *global average pooling*, que compacta cada canal de saída do *dense block* em um único valor, reduzindo consideravelmente a quantidade de dados que serão enfileirados para gerar a saída pela camada *softmax*.

Assim como anteriormente, o tamanho do *mini-batch* e o passo do otimizador SGD foram mantidos como padrão, e foi utilizada a entropia cruzada como função de custo. O modelo foi treinado por 200 épocas, sendo sempre salvo o valor da melhor época por meio de validação cruzada, que foi restaurado ao final do treinamento.

### 4.1 Análise do desempenho

Ao classificar os dados de teste com a rede convolucional profunda já treinada, observa-se um ganho interessante de acurácia, como mostrado em (6) e (7). Com a inserção de mais camadas, o campo receptivo se amplia, permitindo o aprendizado de padrões mais complexos, e o aprendizado da representação, por meio da extração de atributos camada-a-camada.

$$Accuracy = 0.9497 \quad (6)$$

$$BA = 0.9444 \quad (7)$$

A matriz de confusão do classificador para os dados de teste é vista na Tabela 6, onde se percebe uma esparsidade maior, apresentando menos casos de erro. A classe mais desafiadora persiste sendo a 3, porém ao contrário do ocorrido com a CNN *Shallow*, a classe 5 apresentou uma grande ganho de precisão, como visto na Tabela 7. As classes 1 e 7 foram as que alcançaram maiores precisões e *recalls*, onde a última conseguiu atingir precisão unitária e apenas um caso de falso negativo.

	0	1	2	3	4	5	6	7
0	228	0	0	7	2	5	2	0
1	1	621	0	1	0	0	1	0
2	1	0	304	2	3	0	0	1
3	17	2	12	504	5	21	18	0
4	0	0	6	14	221	2	0	0
5	1	0	1	24	2	255	1	0
6	1	1	4	13	0	1	646	0
7	0	0	0	0	0	0	0	470

Tabela 6: Matriz de confusão do classificador baseado em CNN *Shallow*.

Classe	Precisão	<i>Recall</i>
0	0.9344	0.9157
1	0.9952	0.9952
2	0.9775	0.9297
3	0.8705	0.8920
4	0.9095	0.9485
5	0.8979	0.8979
6	0.9700	0.9671
7	1.0000	0.9979

Tabela 7: Matriz de confusão do classificador baseado em CNN *Shallow*.

## 4.2 Análise de erros

Analisando os erros da CNN *Deep*, observa-se principalmente um comportamento menos desafiador, onde a rede tende a errar menos, mas ao errar, o faz com certeza que está certa. Tal questão se mostra nas Figuras 25, 26 e 27, onde há grande probabilidade para a classe vencedora, e a classe verdadeira ocupa a segunda posição, com probabilidade consideravelmente menor.



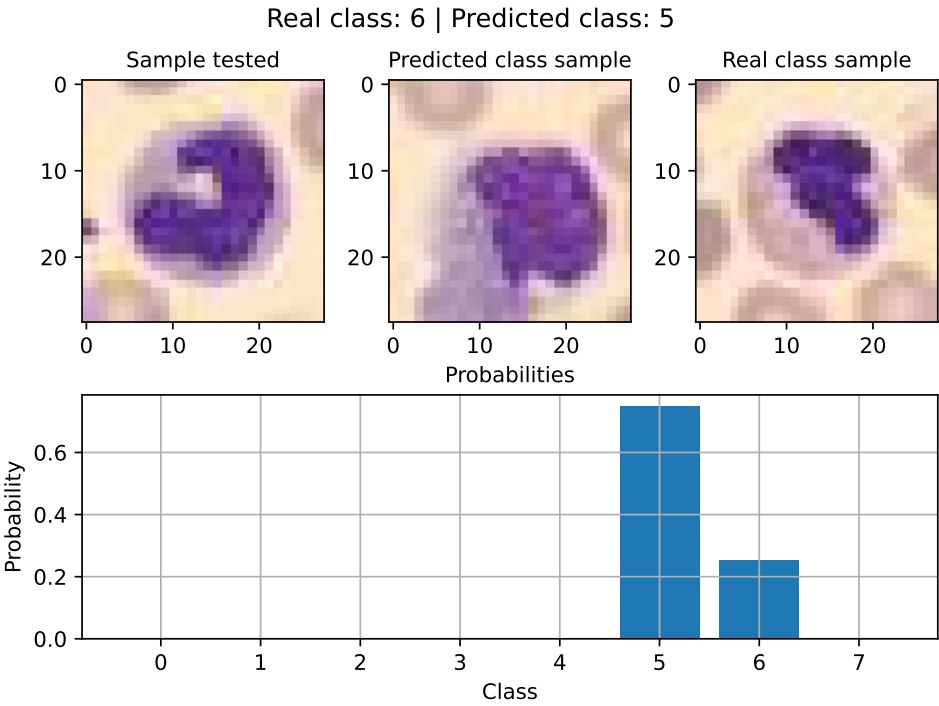


Figura 25: Análise de um caso de erro de classificação.

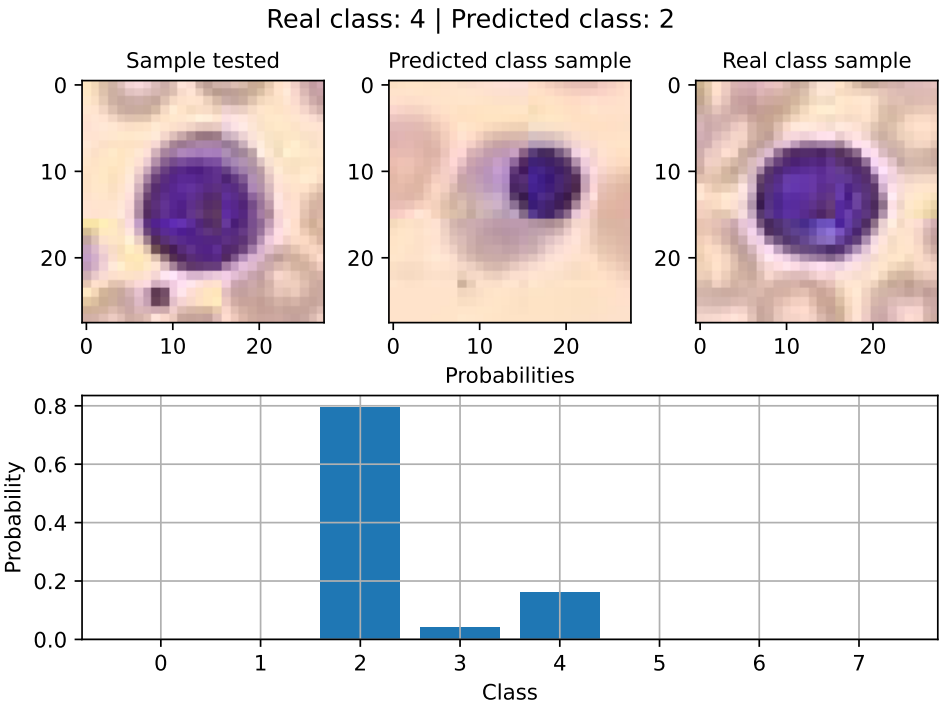


Figura 26: Análise de um caso de erro de classificação.

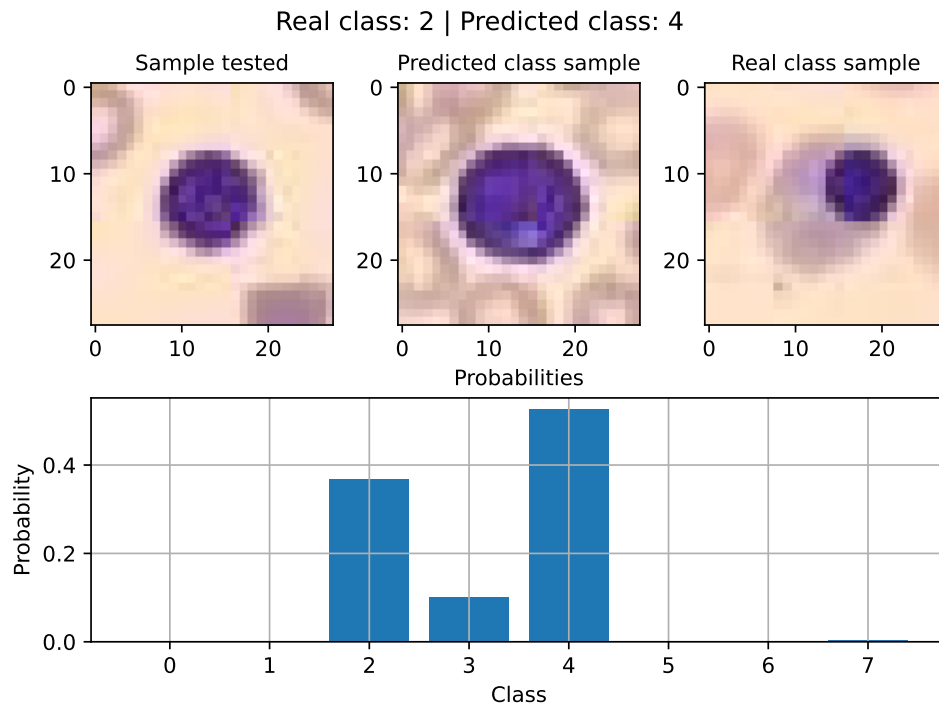


Figura 27: Análise de um caso de erro de classificação.

A amostra das Figuras 28 e 29 também apresentaram erro de classificação para a rede convolucional rasa, porém, ao comparar a Figura 20 e Figura 28, observa-se a mudança no panorama de probabilidades, onde a rede profunda passa a errar com certeza, onde para a rede rasa existia muita dúvida. Já ao comparar a Figura 15 e Figura 23 com a Figura 29 se vê o mesmo perfil de probabilidade, mas agora a rede profunda se confunde mais, colocando a probabilidade da classe real em posições inferiores, e aumentando a certeza sob a classe errada.

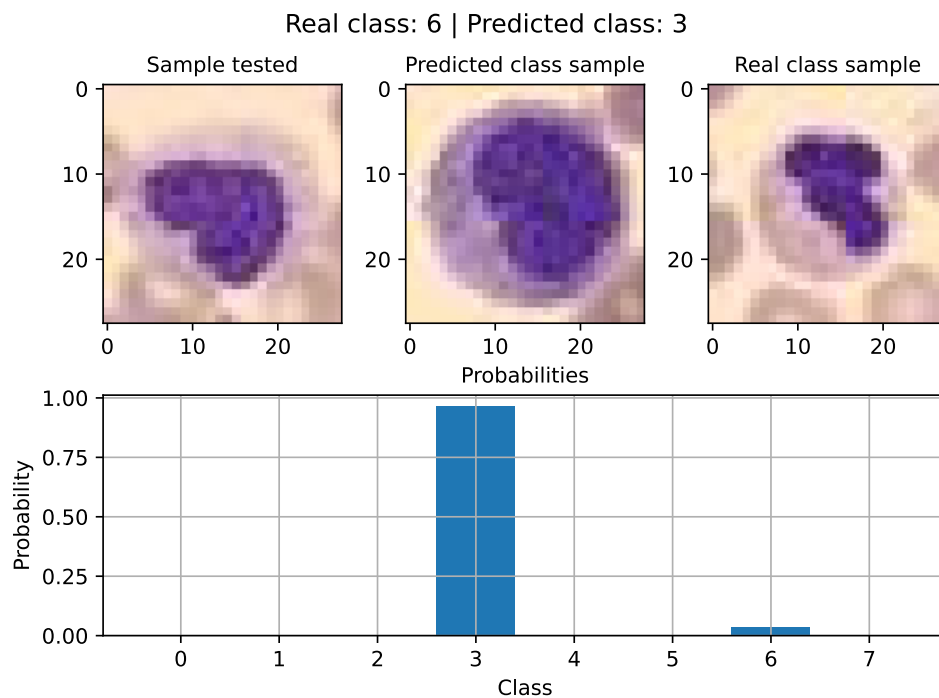


Figura 28: Análise de um caso de erro de classificação.

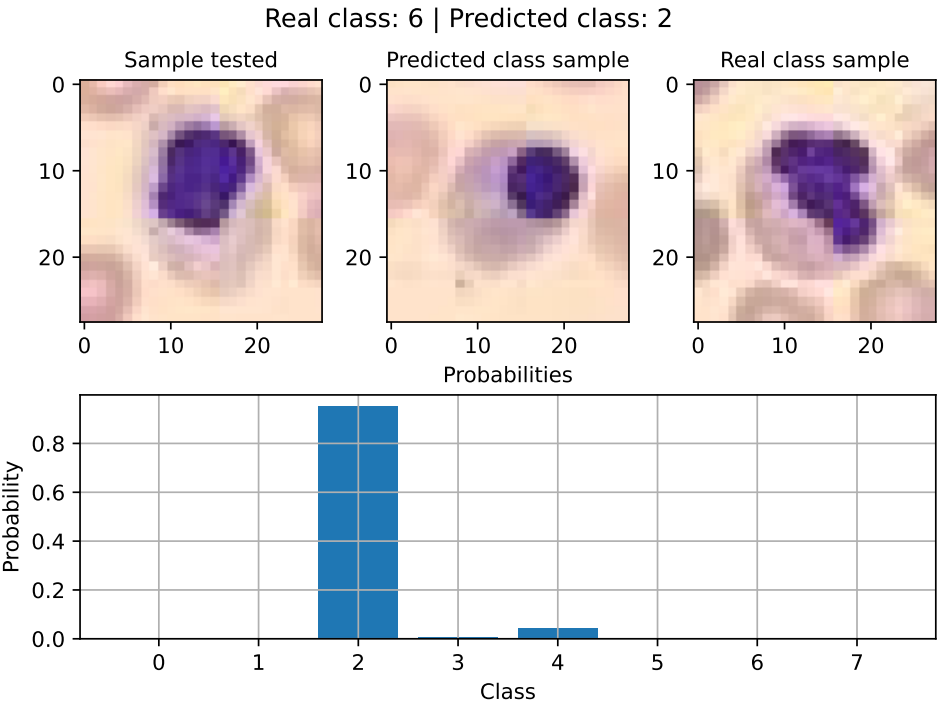


Figura 29: Análise de um caso de erro de classificação.

## 5 Conclusão

O problema solucionado com técnicas de redes neurais é desafiador, uma vez que têm-se que classificar em 8 classes tipos de células sanguíneas, que visualmente não possuem diferenças tão discrepantes, e que ainda porém apresentar variância no tamanho e posição na imagem. Com a utilização das amostras em tamanho 28x28, aceitou-se um pouco de perda de desempenho ao perder resolução, pelo ganho computacional para treinamento e classificação.

Para confecção do classificador, utilizou-se primeiramente redes neurais densas (MLP), que fazem o processamento do dado da imagem a partir do enfileiramento de todos os *pixels* dos três canais de cor em um grande vetor, que foi normalizado. Ao fazer isso, se perde a característica espacial do dado original da imagem, o que faz com seja uma solução menos robusta à variação dos dados de entrada. Para encontrar os hiper-parâmetros que maximizavam o desempenho da MLP, foi realizada uma busca exaustiva pelo número de neurônios da camada intermediária, função de ativação destes neurônios, taxa de *dropout* e algoritmo de otimização. Tal busca, apesar de trazer grande custo computacional, que impede a combinação de todas as possibilidades, trouxe um modelo com boa capacidade de generalização, em vista do uso de apenas uma camada.

O emprego de redes neurais com camadas convolucionais também foi realizado, implementando primeiramente uma CNN de apenas uma camada convolucional. A operação de convolução mantém a natureza espacial dos dados, dessa forma, cada conjunto de pesos aprende a detectar certos padrões, que podem ser encontrados independente da posição na imagem de entrada. Para essa rede, também se fez a busca pelos melhores valores de dois hiper-parâmetros, sendo eles: o de número de *kernels* e o tamanho desses *kernels*. Devido a grande redução do número de pesos, nesse caso pode-se fazer uma busca exaustiva completa entre os candidatos, que além dos valores ótimos, exibiu o perfil de comportamento da relação da acurácia com os parâmetros buscados. Como a camada convolucional é robusta ao tamanho do dado de entrada, testou-se também a aplicação de uma imagem de entrada 64x64, onde após treinar apenas a camada de saída, conseguiu-se manter uma acurácia elevada, o que é limitado pelo campo receptivo reduzido de uma rede rasa.

Já ao dar profundidade à rede convolucional, com emprego de diversas camadas montadas na forma de blocos densos, assim como na *DenseNet*, observou-se um grande ganho de acurácia, reduzindo o erro em casos desafiadores e dando uma capacidade de generalização ainda maior à rede. Com o emprego de rede convolucionais profundas, consegue-se ampliar o campo receptivo à cada camada, permitindo o uso de *kernels* menores, gerando mais canais de saída e extraíndo mais atributos, onde cada camada simplifica o problema, e tornando a classificação pela camada de saída mais simples.

Ao comparar as três redes, o que mais se acentua é a quantidade de pesos ajustáveis presente em cada uma, onde a rede MLP chega a ter mais de 800 vezes a quantidade de pesos da CNN *Shallow*, apresentando um desempenho inferior. Já a rede profunda, devido às suas muitas camadas, apresenta um grande volume de pesos, ainda bem menor que da MLP, mas muito reduzido devido à camada de saída que realiza a compactação de cada um dos 256 canais de

saída da rede convolucional em um escalar, reduzindo consideravelmente o número de pesos da camada densa de saída. Porém também é claro o ganho de desempenho ao se escolher redes convolucionais, e principalmente, dando profundidade às mesmas. Ao processar imagens, ser robusto ao tamanho da entrada e à translação dos padrões dentro da imagem trazem resultados positivos, e que são alcançados ao usar CNNs. Porém, as redes MLP também apresentam grande desempenho, e são constantemente utilizadas na literatura para processar os dados no espaço latente criado pelas camadas convolucionais, trazendo um grande desempenho para as redes convolucionais profundas.

Dessa forma, vê-se que para obter um bom desempenho para classificação de imagens utilizando redes neurais, é necessário a tomada de diversas decisões: desde à arquitetura que será utilizada, até os hiper-parâmetros que serão escolhidos para tal. É importante também casar o modelo que se deseja utilizar com o *hardware* disponível, para que seu treinamento ocorra em tempo hábil e a mesma também possa ser utilizada com facilidade. Cada aplicação em aprendizado de máquina terá suas peculiaridades, e cabe ao cientista fazer bom uso da literatura e da sua experiência para fazer boas escolhas e realizar diversos testes para a escolha de um modelo capaz de generalizar da melhor forma possível, evitando assim um modelo muito rígido, que não aproxima bem os dados, ou um modelo com *overfitting*, que trás grande erro de estimação à solução.

## Referências

A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning, 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](https://github.com/toffanetto/ia048).