Rede Neuronal Convolucional: Explorando identificação de dígitos manuscritos

Luiz Gustavo Rodrigues Martins e Rodrigo Ferreira Guimarães
Departamento de Ciência de Computação e Faculdade de Tecnologia
Universidade de Brasília, Brasília
E-mails: luiz.rodrigues@aluno.unb.br e rodrigofegui@aluno.unb.br
Matrículas: 13/0123293 e 14/0170740

I. Resumo

A inteligência artificial dispõe de diversos algoritmos para a solução de inúmeros problemas, mas um dos desdobramentos mais conhecidos é a *deep learning*, com a utilização de redes neuronais convolucionais. Com isso em mente, este projeto visa a exploração de tais redes na identificação de todos dígitos decimais manuscritos da base MNIST, com implementação em Python e fazendo uso do *framework* Keras.

II. Embasamento Teórico

Uma rede neuronal convolucional, também conhecida pelo acrônimo CNN, é um modelo de rede projetada para trabalhar com dados de imagem bidimensionais. Porém, também pode ser utilizada para trabalhar com dados uni e tridimensionais.

O núcleo desse tipo de rede neuronal é a camada convolucional, responsável por realizar operações de convolução sobre os dados que fluem pela rede.

A. Camadas Convolucionais

As camadas de convolução são os blocos que dão forma a uma rede neuronal convolucional. São responsáveis por aplicar convoluções sobre os dados de entrada.

Na operação de convolução, um filtro é aplicado sobre a entrada resultando em ativações. A aplicação sucessiva desse filtro sobre uma dada entrada irá gerar um mapa de ativações, chamado de mapa de características (*feature map*). Esse mapa mostram locais onde ocorre e também a força de uma determinada característica detectada nos dados de entrada, geralmente uma imagem.

Durando o treinamento de uma CNN, um grande número de filtros são aprendidos com base nos dados de treinamento utilizados. Esses recursos aprendidos podem ser utilizados para encontrar características em imagens e, de uma forma preditiva, classificar os dados de entrada em grupos.

B. Pooling

Os mapas de características obtidos na saída de uma camada convolucional, apresentam sensibilidade quanto à localização dessas características nos dados de entrada. Com isso, pequenos deslocamentos de uma característica na imagem de entrada irá originar um diferente mapa de característica. Para se reduzir essa dependência gerada pela sensibilidade, pode-se utilizar uma abordagem de subamostragem.

Camadas de *pooling* é um recurso muito utilizado para realizar essa subamostragem. Essa camada irá reduzir um mapa de características por um determinado fator de acordo com o tamanho do filtro utilizado para o *pooling*.

Essas camadas são úteis, pois pequenas mudanças na localização de uma *feature* detectada pela camada convolucional nos dados de entrada resultarão em um *feature map* agrupado com a *feature* no mesmo local.

Duas formas comuns de *pooling* são:

- Média (*average pooling*): calcula o valor médio para cada *patch* do mapa de características.
- Máximo (*max pooling*): calcula o valor máximo para cada *patch* do mapa de características.

III. Desafio

A base de dados MNIST (*"re-Mixed" National Institute of Standards and Technology*) armazena dígitos manuscritos, de 0 a 9 dentre seus 70.000 registros, por isso esta deve ser a base para uso e teste de uma rede neuronal convolucional. A implementação deverá ser feita no *framework* Keras [1]. Por ser um experimento comum para os iniciantes nesta aprendizagem, consulta a implementações existentes e divulgadas está permitido [2].

Nesta exploração, as seguintes questões devem ser respondidas:

1) A influência da normalização nos resultados;
2) Considerando validação cruzada com proporções distintas: $70/30$ e $90/10$;
3) A influência da quantidade de camadas convolucionais nos resultados, de 1 a 3 camadas.

IV. Implementação e Análise

A implementação foi realizada em Python, ver. 3.8.5, e está disponível online no Github [3].

Tendo como base um artigo da Medium [4], a implementação possui as seguintes etapas:

1) **Aquisição dos dados:** A partir do padrão disponibilizado pelo Keras, as imagens são transformadas em volumes, com a possibilidade de normalização dos *pixels*, e seus identificadores são codificados em *One-hot*, para futura classificação;
2) **Embaralhamento dos dados de testes:** Visto que as redes se ajustam aos dados para melhorar seus resultados, a ordem dos dados podem interferir no

desempenho do treinamento, por isso existe essa possibilidade;

3) **Separação em validação cruzada:** A divisão dos dados de treinamento e validação cruzada é controlada por porcentagem;
4) **Desenvolvimento do modelo:** Com três esqueletos disponíveis (1, 2 e 3 camadas convolucionais), têm-se a seleção de um deles para o treinamento e ponderações. Além disso, alguns ajustes estruturais dos modelos foram explorados, para entender sua influência sobre o desempenho dos modelos e para poder contestar os valores escolhidos no código-exemplo, sendo eles: **a)** quantidade de filtros na primeira camada convolucional, **b)** tamanho dos filtros, **c)** quantidade de filtros para eventuais demais camadas convolucionais, **d)** tamanho da primeira camada densa, **e)** modo de ativação do compilador, **f)** tipo de otimizador do compilador e **g)** tamanho dos lotes de treinamento;
5) **Análise de desempenho:** Plots da evolução do modelo ao longo das gerações são disponibilizados, assim como as taxas de erros e de acurácia e o valor absoluto dos erros do modelo frente os dados de testes.

A seguir será demonstrado o que foi encontrado durantes as explorações sobre este fluxo.

V. Resultados

Considerando o aspecto exploratório deste projeto sobre as CNN, os parâmetros estruturais dos modelos representam as primeiras explorações. Para tanto, foi considerada a rede com duas camadas convolucionais, por ser a mais simples dentre as multi-camadas, considerando como parametrização inicial os adotados no código-exemplo. Dessa forma, foram adotadas três gerações para todas as explorações e os resultados foram normalizados, para combinação de escalas distintas numa mesma análise, sendo analisados como segue:

a) Quantidade de filtros na primeira camada convolucional: Após a volumetrização das imagens, estas são submetidas à análise por $\mathcal{N}$ filtros convolucionais, sendo este o ponto de exploração. Para tanto, considerou-se $\mathcal{N} \in [1, 40]$, conforme demonstrado na Figura 1, e os melhores resultados foram obtidos com 27 filtros. Além disso, percebe-se que esta variação tem pouco influência sobre a acurácia a partir do 10 filtro, por outro lado os erros tendem a uma estabilidade com o acréscimo de filtros;
b) Tamanho dos filtros: Além da quantidade dos filtros, outro fator importante está sobre seu tamanho, $\mathcal{N} \times \mathcal{N}$. Portanto, considerou-se $\mathcal{N} \in [1, 10]$, conforme demonstrado na Figura 2, e os melhores resultados foram obtidos com tamanho 6. Além disso, percebe-se que uma certa estabilidade a partir do tamanho 3;
c) Quantidade de filtros para eventuais demais camadas convolucionais: Com o acréscimo de camadas convolucionais há a maior chance de especialização de alguns ramos a determinadas características, mas isso não necessariamente implica em aumento da quantidade de filtros, portanto considerou-se mantendo a quantidade inicial de filtros e até quadruplicando-a. Conforme demonstrado na Figura 3, percebe-se uma grande instabilidade sobre esta variação, tendo pouco impacto sobre os resultados, e o fator de $2.4x$ obteve os melhores resultados;

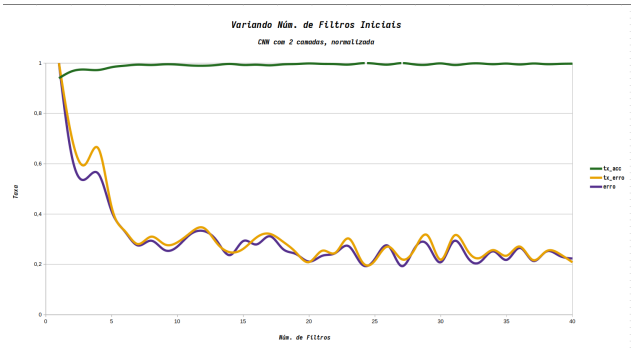

Figura 1: Análise da variação da quantidade de filtros iniciais. Os resultados desta análise foram normalizados

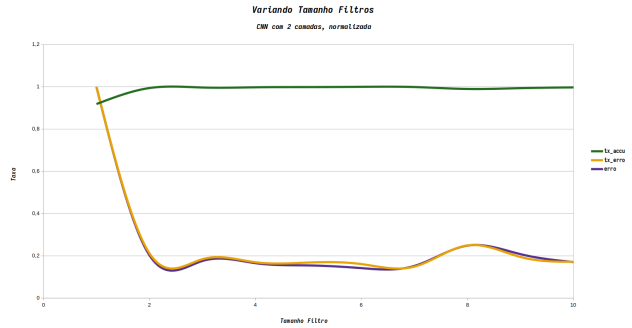

Figura 2: Análise da variação do tamanho dos filtros iniciais. Os resultados desta análise foram normalizados

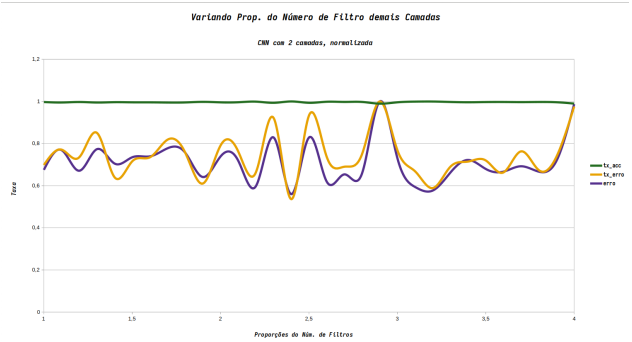

Figura 3: Análise da variação da proporção da quantidade de filtros para as camadas subsequentes. Os resultados desta análise foram normalizados

d) Tamanho da primeira camada densa: Após a vetorização da última camada convolucional, são conectados $\mathcal{N}$ à esta vetorização. Portanto, considerou-se $\mathcal{N} \in [100, 210]$, conforme demonstrado na Figura 4, e os melhores resultados foram obtidos com 210 neurônios. Além disso, percebe-se que a acurácia permaneceu alta, enquanto que houve bastante flutuações quanto aos erros, mas com uma tendência de queda para o aumento da quantidade de neurônios;
e) Modo de ativação do compilador: A transmissão de

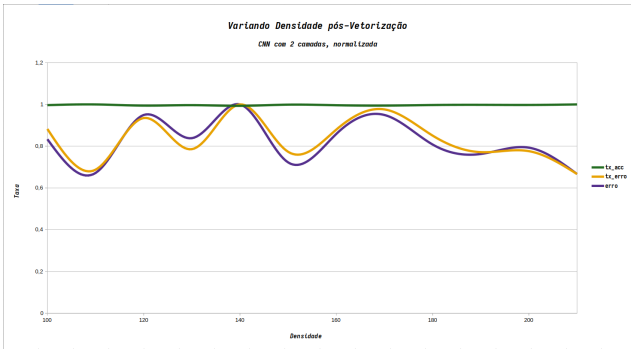

Figura 4: Análise da variação da densidade dos neurônios pós-vetorização. Os resultados desta análise foram normalizados

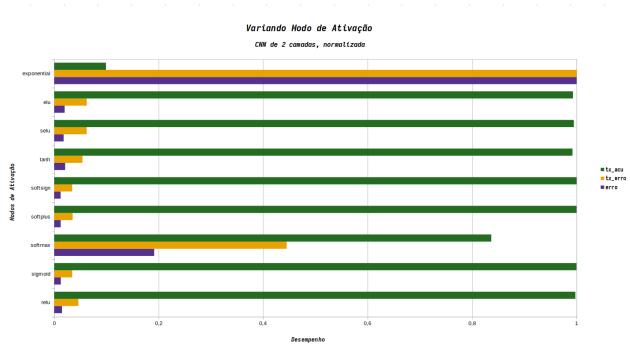

Figura 5: Análise da variação do modo de ativação dos filtros convolucionais e neurônios. Os resultados desta análise foram normalizados

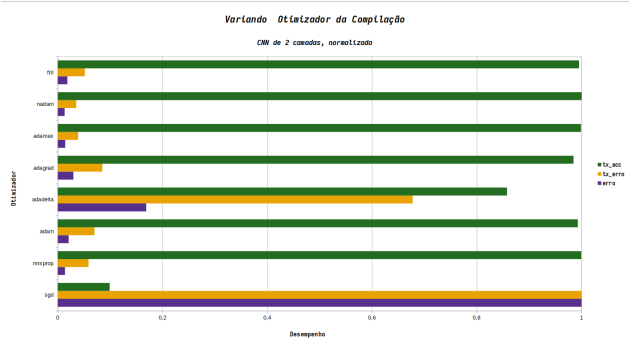

Figura 6: Análise da variação do otimizador do compilador. Os resultados desta análise foram normalizados

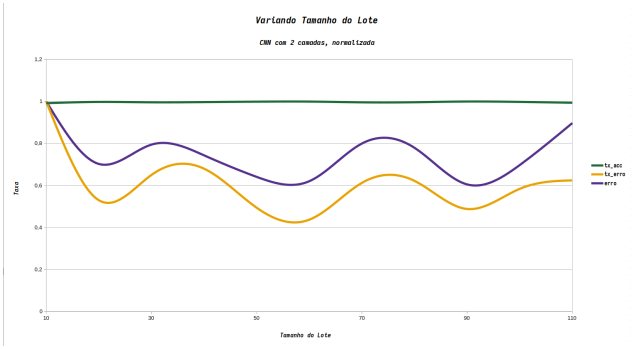

Figura 7: Análise da variação do tamanho dos lotes de treinamento. Os resultados desta análise foram normalizados

informação ocorre por meio da ativação ou não dos nós e a ferramenta utilizada disponibiliza diversas opções: "softplus" ($softplus(x) = log(exp(x)+1)$) foi a que apresentou os melhores resultados, conforme demonstrado na Figura 5. Além disso, como os dados não estavam normalizados, as opções que utilizam a exponencial obtiveram os piores resultados;

f) Tipo de otimizador do compilador: A ferramenta utilizada disponibiliza diversas opções: "nadam" foi a que apresentou os melhores resultados, conforme demonstrado na Figura 6;

g) Tamanho dos lotes de treinamento: Os dados de treinamento são divididos em sub-rodadas de treinamento, de tamanho $\mathcal{N}$. Portanto, considerando-se $\mathcal{N} \in [10, 110]$, conforme demonstrado na Figura 7, e os melhores resultados foram obtidos com grupos de 90. Além disso, percebe-se um grande instabilidade sobre esta variação.

Com estes parâmetros definidos restaria definir a quantidade de época de treinamento, mas como estará presente indiretamente nas análises das questões específicas, a mesma não foi realizada. Dessa forma, para todas as questões, foram consideradas até 15 gerações e foram obtidas as seguintes respostas:

1) A influência da normalização nos resultados;
   Resp.: A partir de uma CNN com duas camadas convolucionais, ao utilizar os dados originais não-normalizados as seguintes estatísticas foram obtidas: $0,0777$ de taxa de erro, $0,9853$ de taxa de acurácia e um total de 147 predições erradas, frente aos 10.000 dados de teste; enquanto que para com os dados normalizados foram: $0,0631$ ($\downarrow$ $18,8\%$), $0,9876$ ($\uparrow$ $0,23\%$) e 124 ($\downarrow$ $15,6\%$), respectivamente. As Figuras 8 e 9 demonstram a evolução do modelo frente esses dados ao longo das gerações e é de fácil percepção uma evolução suave com os dados normalizados. Por fim, os melhores resultados foram encontrados com os dados normalizados.
2) Considerando validação cruzada com proporções distintas: $70/30$ e $90/10$;
   Resp.: Considerando a análise anterior, os dados foram normalizados para esta análise. Além disso, a análise anterior considerava a proporção $90/10$, sendo necessário, portanto, trazer as estatísticas sobre a proporção $70/30$: $0,0528$ ($\downarrow$ $16,3\%$) de taxa de erro, $0,9866$ ($\downarrow$ $0,10\%$) de taxa de acurácia e um total de 134 ($\uparrow$ $8,06\%$) predições erradas. As Figuras 9 e 10 demonstram a evolução do modelo frente às proporções de validação cruzada. Por fim, os melhores resultados foram encontrados com a proporção $90/10$.
3) A influência da quantidade de camadas convolucionais nos resultados, de 1 a 3 camadas.
   Resp.: Considerando a análise anterior, os modelos tiveram validação cruzada com 10% dos dados de treinamento. Dessa forma, as estatísticas para 1 camada convolucional (com 1.114.342 parâmetros treináveis) foram: $0,0460$ ($\downarrow$ $27,1\%$) de taxa de erro, $0,9845$ ($\downarrow$ $0,31\%$) de taxa de acurácia e um

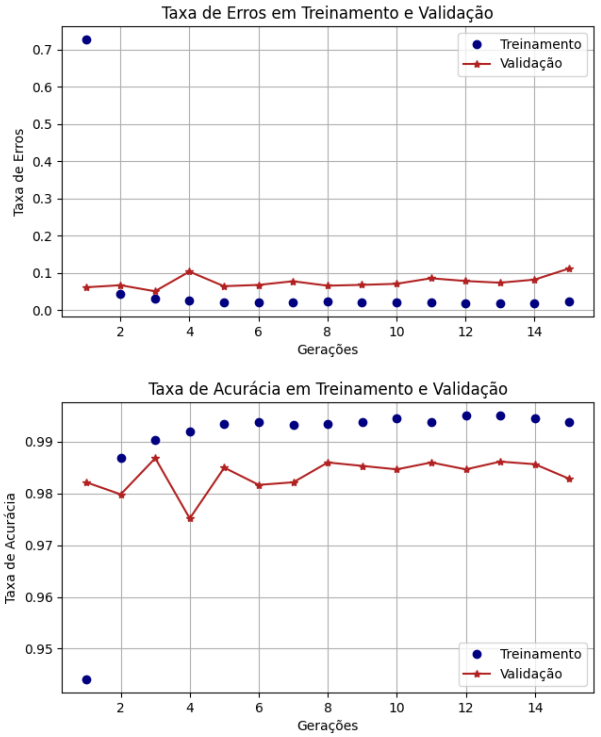

Figura 8: Análise da CNN com 2 camadas com os dados não-normalizados.

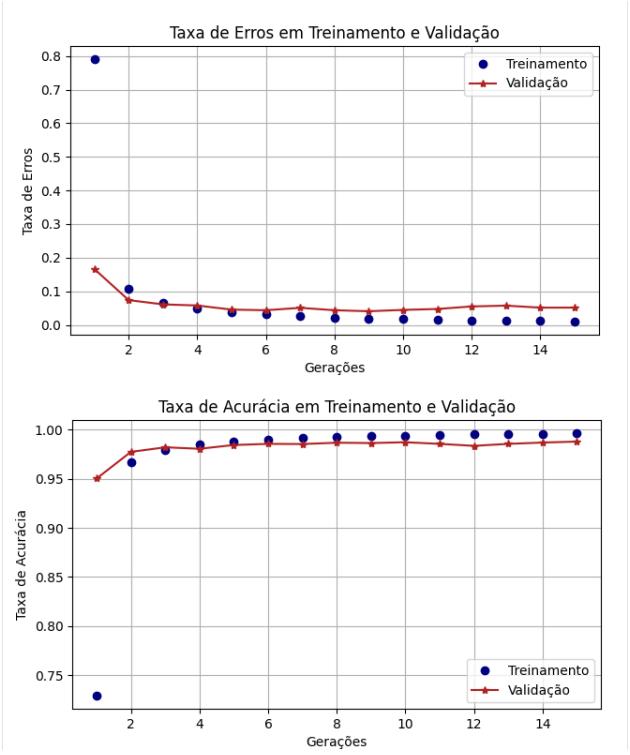

Figura 9: Análise da CNN com 2 camadas com os dados normalizados.

total de 155 ($\uparrow$ 25%) predições erradas; e para 3 camadas (269.710 parâmetros): $0,0597$ ($\downarrow$ $5,39\%$), $0,9888$ ($\uparrow$ $0,12\%$) e 112 ($\downarrow$ $9,67\%$), respectivamente. As Figuras 9, 11 e 12 demonstram as evoluções dos modelos. Por fim, os melhores resultados foram

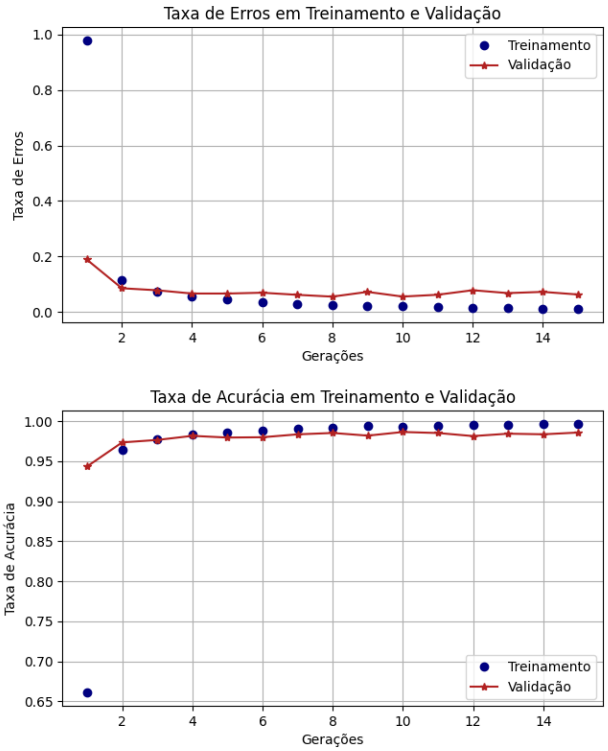

Figura 10: Análise da CNN com 2 camadas com os dados normalizados e proporção $70/30$ de validação cruzada.

encontrados com 3 camadas convolucionais.

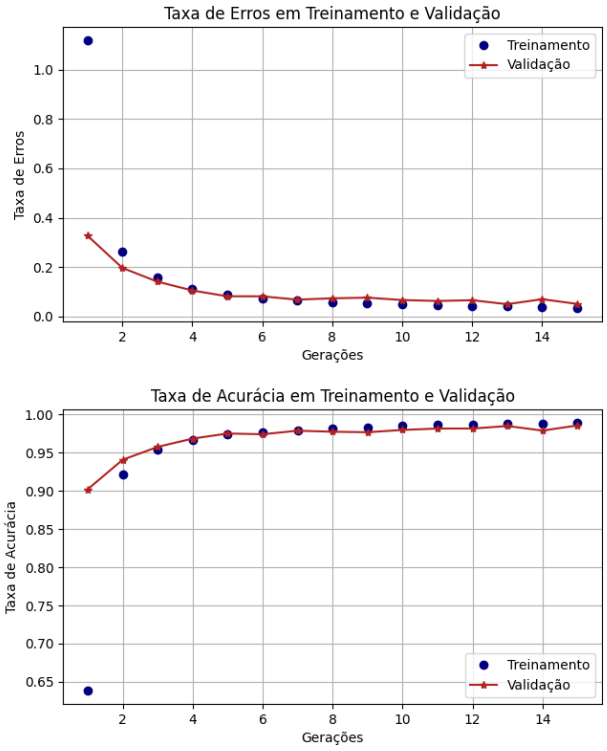

Figura 11: Análise da CNN com 1 camada convolucional.

VI. Conclusão

As redes neuronais convolucionais exploradas para a detecção de dígitos manuscritos trouxeram diversas dúvidas

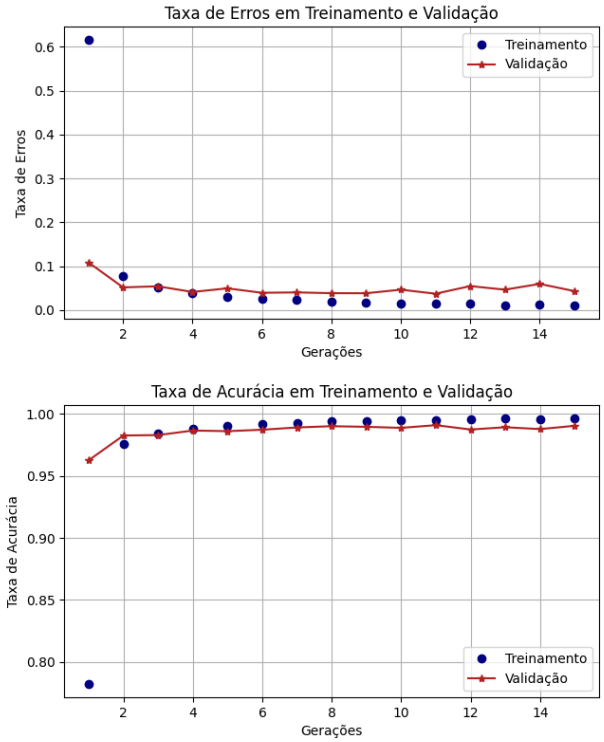

Figura 12: Análise da CNN com 3 camadas convolucionais.

e esclarecimentos sobre as parametrizações, visto que: ao aumentar a quantidade de filtros convolucionais implicaria no aprendizado de mais características, portanto quanto mais filtros melhor seria o modelo, mas o observado é que os resultados convergem e descarta a necessidade de muitos filtros; analogamente, a mesma análise vale para o tamanho quadrático desses filtros. Ao passo que para as demais parametrizações, precisam ser consideradas a cada contexto de desafio, portanto conhecê-las agora foi essencial.

A partir das perguntas iniciais, ficou possível observar que: a normalização dos dados de entrada melhora o desempenho geral das CNN; a validação cruzada é uma boa prática, principalmente se a base de dados for grande suficiente para ter bastante dados de treinamento, visto que ao utilizar somente 10% dos dados de treinamento foi a melhor seleção pelo maior volume de aprendizado; o aumento da quantidade de camadas convolucionais proporcionou melhores resultados, mas com uma convergncia e deve ser analisada por contexto.

Referências

[1] K. Team, "Keras: the python deep learning api," 2020. [Online]. Available: https://keras.io/

[2] D. L. Borges, "Projeto 4, Introdução à Inteligência Artificial, Turma A, 1/2020," Universidade de Brasília, Tech. Rep., 2020.

[3] A. própria, "Ia-proj4: Projeto 4 da disciplina intro a ia," 2020. [Online]. Available: https://github.com/rodrigofegui/IA-proj4/

[4] M. Bhobé, "Mnist — digits classification with keras," 2020. [Online]. Available: https://medium.com/@mjbhobe/mnist-digits-classification-with-keras-ed6c2374bd0e