Sistemas Autónomos - CNNs vs MLPs na classificação de imagens

Diogo F. Braga
University of Minho
Department of Informatics
4710-057 Braga, Portugal
Email: a82547@alunos.uminho.pt

Resumo. *Uma pessoa pode descrever o conteúdo de uma fotografia que viu uma vez, pode resumir um vídeo que viu uma vez, pode reconhecer um rosto que viu uma vez, assim como pode realizar outras tantas ações processadas pela visão humana. Estes processamentos são indispensáveis para a evolução humana e, devido a tal, têm sido transferidos para a área da Informática no sentido de potenciar a evolução. Este processamento é realizado informaticamente através da Visão Computacional, sendo o principal objetivo destes problemas usar os dados de qualquer imagem observada para inferir algo sobre o mundo.*

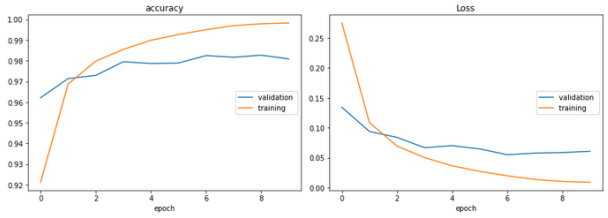
1 Introdução

A classificação de imagens é um processo de extração e interpretação de informação a partir de dados digitais. Este processo inclui-se na área científico-tecnológica da Visão Computacional, uma das principais sub-áreas da Inteligência Artificial. O principal objetivo deste processo é emular a visão humana e, assim, ter capacidade de interpretar imagens. Este fenómeno pode ser realizado através da utilização de redes neuronais artificiais sendo que, neste documento vão ser apresentados e comparados resultados de dois tipos de rede, são eles: *MultiLayer Perceptron (MLP)* e *Convolutional Neural Networks (CNN)*.

O *benchmark* é realizado sobre quatro diferentes *datasets*, iniciando num mais simples (*Mnist*), depois passando pelo *CIFAR10* e pelo *Mnist Fashion*, até utilizar no final um mais complexo, o (*CINIC-10*). Neste acrescento de complexidade compreendem-se características como a inserção de cores e a quantidade e variedade de imagens. Para cada uma das fases serão apresentados os gráficos de aprendizagem e os resultados obtidos, sendo através destes que se irão comparar a eficiência dos modelos utilizados. Estas fases serão todas testadas igualmente com 10 épocas, de forma à comparação ser justa. Todas as arquiteturas utilizadas em cada rede encontram-se anexadas na pasta principal deste projeto, sendo que neste documento vão ser, principalmente, discutidas as arquiteturas e hiperparâmetros dos últimos *datasets*. Durante o documento serão tidas em conta métricas como a *accuracy*, a *loss* e o tempo de processamento de cada modelo.

2 Mnist utilizando MLPs

Inicialmente num *dataset* simples (*Mnist*), em que o objetivo é processar imagens com números a preto e branco, é realizada uma classificação utilizando MLPs. Estas redes não são as ideais para processamento de imagem, no entanto neste exercício conseguiram atingir uma *accuracy* de **98.09%**, num tempo de processamento de **55.8 segundos**, tal como mostrado na figura 1. Neste gráfico é possível visualizar os gráficos de aprendizagem e a sua eficiente subida. Importante referir um pequeno *overfitting* aos dados de treino que, ainda que pouco substancial, poderia ser esvaecido utilizando uma camada de *dropout* na constituição da rede.



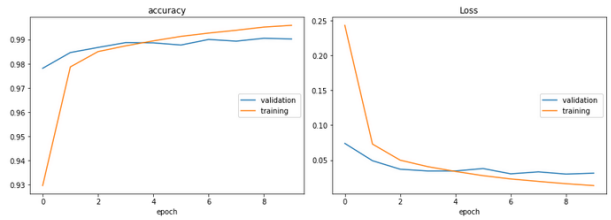

```
accuracy
        validation              (min:    0.962, max:    0.983, cur:    0.981)
        training                (min:    0.921, max:    0.998, cur:    0.998)
Loss
        validation              (min:    0.055, max:    0.134, cur:    0.061)
        training                (min:    0.009, max:    0.275, cur:    0.009)
Accuracy modelo MLP mnist: 98.09%
Erro modelo MLP mnist: 1.91%
Tempo processamento: 55.852 segundos
```

Fig. 1. Evolução da Accuracy e Loss referentes à classificação de imagens utilizando MLPs

3 Mnist utilizando CNNs

De forma a poder comparar a eficiência das CNNs em relação às MLPs, estas são agora também testadas no mesmo *dataset*. Para as CNNs foram testadas duas arquiteturas, uma mais simples e outra mais complexa. Neste exercício

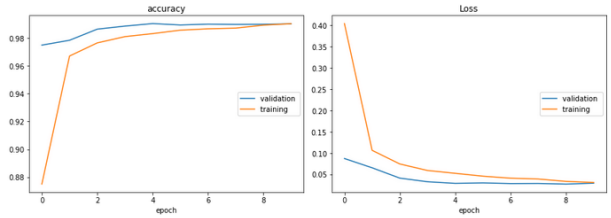
a solução simples atingiu uma *accuracy* de **99.02%**, num tempo de processamento de **257.0 segundos (4.3 min)**, tal como mostrado na figura 2. Naturalmente, devido à complexidade destas redes que estão melhor preparadas para processamento de imagens, estas obtiveram um melhor resultado. Tal acontece devido à utilização de uma camada convolucional e de *pooling*. No entanto, a diferença entre os modelos não é grande considerando a grande diferença no tempo de processamento, pelo que para este *dataset* ainda se mostra pouco eficiente a utilização das CNNs. De notar ainda a diminuição do *overfitting*, que foi esvaecido em parte pela utilização duma camada de *dropout*.




```
accuracy
        validation              (min:    0.978, max:    0.990, cur:    0.990)
        training                (min:    0.930, max:    0.996, cur:    0.996)
Loss
        validation              (min:    0.030, max:    0.074, cur:    0.031)
        training                (min:    0.013, max:    0.243, cur:    0.013)
Accuracy modelo CNN mnist simples: 99.02%
Erro modelo CNN mnist simples: 0.98%
Tempo processamento: 257.014 segundos
```

Fig. 2. Evolução da Accuracy e Loss referentes à classificação de imagens utilizando CNNs com uma arquitetura simples

Nesta arquitetura foi ainda introduzida mais complexidade, com especial foco na utilização de mais uma camada convolucional e de *pooling*, levando assim a uma maior redução nas *features* realçando as de maior importância. Neste exercício a solução complexa atingiu uma *accuracy* de **99.03%**, num tempo de processamento de **332.7 segundos (5.5 min)**, tal como mostrado na figura 3. Estes valores levam a concluir que, apesar de se tratarem de técnicas importantes, neste *dataset* simples estas não apresentam muita influência. Importante referir o esvaecimento quase total do *overfitting* aos dados de treino, tal justificável pelo maior número de camadas aliado à diminuição da diversidade de *features* utilizadas, através das camadas de *pooling*.

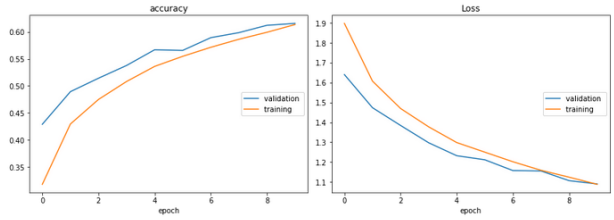



```
accuracy
        validation              (min:    0.975, max:    0.990, cur:    0.990)
        training                (min:    0.875, max:    0.990, cur:    0.990)
Loss
        validation              (min:    0.027, max:    0.087, cur:    0.029)
        training                (min:    0.031, max:    0.404, cur:    0.031)
Accuracy modelo CNN mnist plus: 99.03%
Erro modelo CNN mnist plus: 0.97%
Tempo processamento: 332.744 segundos
```

Fig. 3. Evolução da Accuracy e Loss referentes à classificação de imagens utilizando CNNs com uma arquitetura complexa

4 CIFAR10 (cores) utilizando CNNs

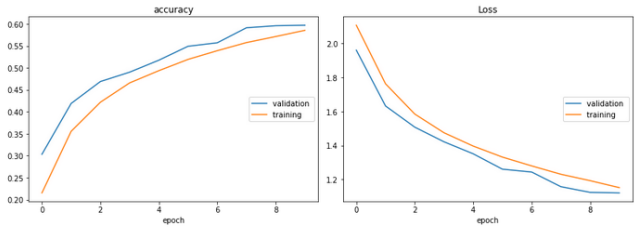
De forma a introduzir mais complexidade nos dados, o novo *dataset* utilizado possui as *features* relacionadas com as cores, mais especificamente 3 canais de cores, levando assim a um aumento significativo na quantidade de dados. Neste exercício a solução atingiu uma *accuracy* de **61.57%**, num tempo de processamento de **1240.4 segundos (20 min)**, tal como mostrado na figura 4. Nesta rede, as camadas significativas são duas convolucionais e uma de *pooling*. Devido ao aumento significativo de dados, a *accuracy* apresentada é um valor dentro do esperado, sendo importante reparar na evolução contínua das curvas, o que deixa a concluir que com mais épocas de treino o resultado atingiria certamente valores mais elevados.




```
accuracy
        validation              (min:    0.429, max:    0.616, cur:    0.616)
        training                (min:    0.318, max:    0.614, cur:    0.614)
Loss
        validation              (min:    1.090, max:    1.640, cur:    1.090)
        training                (min:    1.088, max:    1.897, cur:    1.088)
Accuracy modelo CNN cifar10 simples: 61.57%
Erro modelo CNN cifar10 simples: 38.43%
Tempo processamento: 1240.384 segundos
```

Fig. 4. Evolução da Accuracy e Loss referentes à classificação de imagens com cores utilizando CNNs com uma arquitetura simples

Nesta arquitetura foi também introduzida mais complexidade, com especial foco na utilização de mais duas séries de camadas convolucionais e *pooling*. Em 10 épocas, o resultado da *accuracy* foi bastante semelhante ao de complexidade simples, mas é expectável que para um número superior de épocas este modelo obtivesse melhores resultados, tendo em conta a grande complexidade introduzida com os 3 canais de cores.




```
accuracy
        validation              (min:    0.303, max:    0.597, cur:    0.597)
        training                (min:    0.215, max:    0.586, cur:    0.586)
Loss
        validation              (min:    1.121, max:    1.961, cur:    1.121)
        training                (min:    1.152, max:    2.108, cur:    1.152)
Accuracy modelo CNN cifar10 plus: 59.71%
Erro modelo CNN cifar10 plus: 40.29%
Tempo processamento: 2786.663 segundos
```

Fig. 5. Evolução da Accuracy e Loss referentes à classificação de imagens com cores utilizando CNNs com uma arquitetura complexa

5 Mnist Fashion utilizando MLPs e CNNs

O *Mnist Fashion* é um *dataset* semelhante ao *Mnist*, mas com bastante mais complexidade através de mais variedade de dados. Possui apenas a cor preta e branca, mas devido à sua maior complexidade é um bom *dataset* para testar a eficiência das redes criadas. Neste exercício, para as MLPs a solução atingiu uma *accuracy* de **88.5%**, num tempo de processamento de **61.7 segundos**, tal como mostrado na figura 6. O valor mais baixo da *accuracy*, em relação ao modelo testado na secção 2, deve-se à maior abundância e complexidade dos dados, sendo que tal era previsível tendo em conta a utilização da mesma arquitetura da rede. O *overfitting* mantêm-se presente pelas mesmas razões apresentadas.

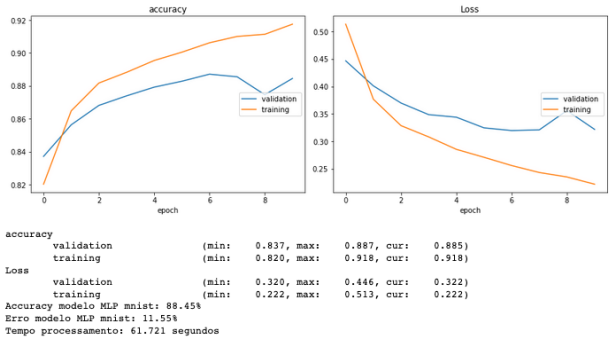


Fig. 6. Evolução da Accuracy e Loss referentes à classificação de imagens utilizando MLPs

Os acontecimentos visualizados na passagem das MLPs para as CNNs no *Mnist* verificam-se também com este novo *dataset*, pelo que a única diferença é a esperada redução da *accuracy*, relacionada com o aumento da complexidade dos dados. De notar, novamente, a maior eficiência na redução do *overfitting* na arquitetura complexa das CNNs, causada pelas camadas de *dropout* e *pooling*, que diminuem a complexidade das *features* nos dados. A solução simples atingiu uma *accuracy* de **91.1%** num tempo de processamento de **274.5 segundos (4.6 min)**, enquanto que a solução complexa atingiu uma *accuracy* de **89.6%** num tempo de processamento de **351.0 segundos (5.9 min)**.

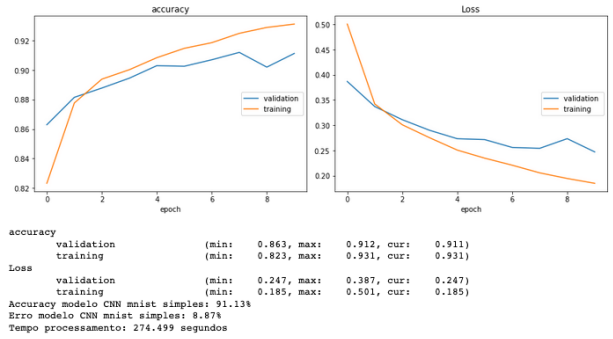


Fig. 7. Evolução da Accuracy e Loss referentes à classificação de imagens utilizando CNNs com uma arquitetura simples

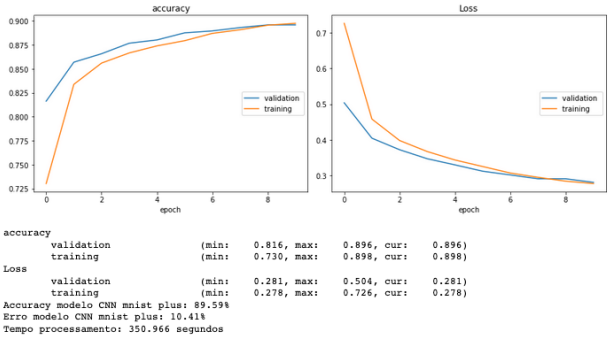


Fig. 8. Evolução da Accuracy e Loss referentes à classificação de imagens utilizando CNNs com uma arquitetura complexa

6 CINIC-10 (cores) utilizando CNNs

De forma a introduzir mais complexidade nos dados em relação ao *CIFAR-10*, este novo *dataset (CINIC-10)* possui mais quantidade de dados, e possui igualmente cores associadas. Para este *dataset* foi necessário utilizar *data generators*, de modo a aceder aos dados de cada classe diretamente na diretoria em que estão presentes. Desta forma, foram também adaptadas as funções associadas ao modelo, com o intuito de utilizar os dados de treino e de teste diretamente da diretoria. Neste exercício a solução atingiu uma *accuracy* de **51.23%**, num tempo de processamento de **4360.1 segundos (1h 12 min)**, tal como mostrado na figura 9. Desta execução referir a imensidade de tempo para a realização das 10 épocas que, logicamente, provêm da grande densidade de dados em causa. Novamente, a utilização de 10 épocas é também uma limitação, mas pode-se observar a subida da curva, pelo que é natural que este resultado atinja melhores resultados ao fim de mais épocas. Este *dataset* atingiu resultados mais baixos que o *CIFAR-10*, expectável devido à maior quantidade de dados que possui.

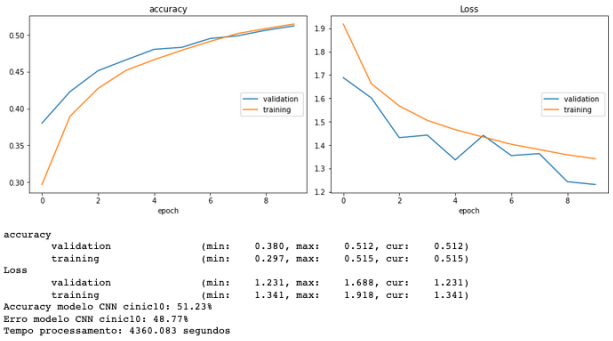


Fig. 9. Evolução da Accuracy e Loss referentes à classificação de imagens a cores utilizando CNNs

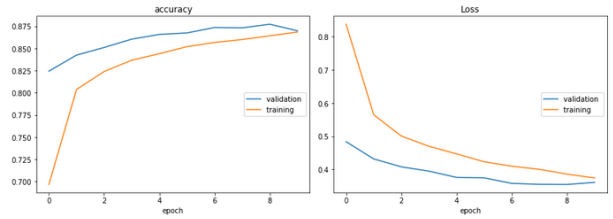
7 Evolução da arquitetura e hiperparâmetros

Até este momento foram justificadas as decisões em causa e resultados obtidos, como a introdução de camadas de *dropout* e de *pooling*. As primeiras com intenções de

diminuir o *overfitting* através do anulamento de alguns nodos, e as segundas com intenções de criar reduções nas *features* em causa realçando as de maior importância. Nesta secção vai ser introduzida um pouco mais de exploração.

7.1 Mnist Fashion

Em relação ao *Mnist Fashion*, nas MLPs foi adicionada uma camada regular intermédia de ativação e também uma de *dropout*, no sentido de diminuir o considerável *overfitting* apresentado no gráfico 6). O resultado foi o esperado e, apesar de o resultado ser mais baixo num ponto percentual, o *overfitting* aos dados de treino na 10ª época já se nota como maioritariamente esvaecido (figura 10).




```
accuracy
        validation              (min:    0.824, max:    0.877, cur:    0.870)
        training                (min:    0.697, max:    0.868, cur:    0.868)
Loss
        validation              (min:    0.355, max:    0.483, cur:    0.361)
        training                (min:    0.374, max:    0.838, cur:    0.374)
Accuracy modelo MLP fashion mnist: 86.98%
Erro modelo MLP fashion mnist: 13.02%
Tempo processamento: 77.544 segundos
```

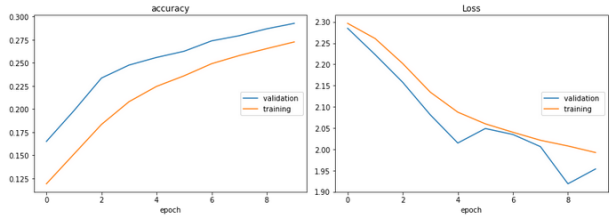
Fig. 10. Evolução da Accuracy e Loss referentes à classificação de imagens utilizando MLPs

Quanto às CNNs, possuindo como otimizador o *'adam'*, já tem automaticamente uma otimização ao nível do *learning rate*, sendo garantida por isso uma escolha satisfatória a esse nível. Além desta evolução implementada nos capítulos anteriores, nesta secção final foi testada a eficiência do modelo com a utilização de camadas *batch normalization*. Estas são utilizadas para normalizar as ativações da camada anterior em relação à camada atual, ou seja, aplicar transformações que mantenha a ativação média próxima a 0 e o desvio padrão da ativação próximo a 1. Desta forma, é também considerada uma camada de regularização que esvaece o *overfitting*. A comparação foi estabelecida com a arquitetura simples (figura 7), e os resultados mantiveram-se parecidos, pelo que a introdução de camadas *batch normalization* não se mostrou muito eficiente. Tal pode ser justificado pela presença das camadas *dropout*, que já esvaecem o *overfitting*.

7.2 CINIC-10

Em relação ao *CINIC-10*, foi realizada uma comparação de modelos com valores diferentes de *learning rate*, visto neste ser aplicado um otimizador *'SGD'*, sendo os resultados desse segundo modelo apresentados na imagem 11, e as comparações com o *learning rate* apresentados na tabela 1.

Desta tabela é possível constatar que a aprendizagem do segundo modelo está num nível mais baixo (como seria expectável devido ao *learning rate*), no entanto a sua inclinação faz prever um melhor resultado que o primeiro modelo pois o




```
accuracy
        validation              (min:    0.165, max:    0.293, cur:    0.293)
        training                (min:    0.119, max:    0.273, cur:    0.273)
Loss
        validation              (min:    1.919, max:    2.285, cur:    1.954)
        training                (min:    1.992, max:    2.296, cur:    1.992)
Accuracy modelo CNN cinic10: 29.28%
Erro modelo CNN cinic10: 70.72%
Tempo processamento: 5022.951 segundos
```

Fig. 11. Evolução da Accuracy e Loss referentes à classificação de imagens a cores utilizando CNNs

Table 1. Resultados associados ao CINIC-10 em CNNs com alteração do learning rate (ao fim de 10 épocas)

Learning rate	Accuracy	Tempo
0.01	51.23%	1h12min
0.0001	29.28%	1h23min

seu crescimento é mais constante. Tal acontece porque, possuindo uma taxa de aprendizagem mais pequena, esta é naturalmente mais cautelosa aprendendo em mais tempo, mas a partir de uma quantidade maior de dados, que por sua vez são também mais diversificados.

8 Conclusão

Deste documento e pesquisas relacionadas é importante, inicialmente, constatar a importância que o *hardware* possui na execução de CNNs, devido à grande quantidade de dados treinados e testados nas arquiteturas normalmente complexas neste tipo de rede. Este ponto foi uma limitação neste *benchmark* pois devido à falta de um GPU não foi possível aplicar outras bibliotecas com melhores rendimentos e, consequentemente, não foi possível gerar muitos testes para comparação nem conseguir visualizar o processo de treino com um bom ritmo.

Nas redes MLPs, os tipos de camadas normalmente utilizadas são as de ativação e também *dropout* e *batch normalization*, o que faz não existir tanta complexidade a este nível. Nas CNNs faz sentido realçar a utilização de mais tipos de camadas, como as de convolução e *pooling*, cada uma com funções e vantagens diferentes dentro da rede, já antes esclarecidas. Desta forma, foram abordadas as camadas que por norma as CNNs possuem na sua arquitetura e adquirido o conhecimento nestas envolvido.

Por fim, concluir que as CNNs apresentam, de facto, melhores resultados que as MLPs na classificação de imagens. Estas são mais complexas e realizam um processo mais lento, no entanto, os resultados são bastantes mais eficientes e, consequentemente, são bastante mais úteis na interpretação e classificação de imagens.