Licenciatura em Engenharia Informática

Conhecimento e Raciocínio 2022/2023

Trabalho Prático / Tema 1 – Redes Neuronais

Reconhecimento de dígitos e símbolos matemáticos

Realizado por:

Diogo Baptista Pinto / Nº 2020133653

Hugo José Neves da Costa / Nº 2016017562

Instituto Superior
de Engenharia
Politécnico de Coimbra

Índice

Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Conhecimento e Raciocínio, escolhemos o tema redes neuronais por ser uma área fascinante que busca replicar o cérebro humano através de algoritmos computacionais.

Usámos o Matlab para implementar redes neuronais e realizar experimentos práticos.

Ao longo do trabalho, exploraremos os conceitos básicos das redes neuronais com a arquitetura feedforward, como camadas, neurónios, funções de ativação e treino.

Implementámos redes neuronais com diferentes cenários, ao longo do trabalho iremos visualizar quais as diferenças admitem.

Programa

Alínea – A

Nesta alínea utilizamos a pasta "*start*" que contém 5 imagens de cada dígito/operador.

Tratamento de imagens

Como primeira alínea, por questões de otimização, optamos por redimensionar as imagens em uma resolução de 25x25 e convertemos em matrizes binárias.

Targets e Inputs

Criámos manualmente uma matriz binária de dimensões 14*(14*5).

```
%% targets:
target = zeros(14, 14*5);
starterVar = 0;
for s=1:14 %digitos e operadores
    for i = 1:5
        target(s, (starterVar+1):s*5) = 1;
        starterVar = 5 * s;
    end
end
```

A matriz binária é constituída nas 10 primeiras linhas pelos números e do 11 ao 14 pelos operadores.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Nas posições 1:1 ao 1:5, representa os dígitos “0” das 5 imagens reconhecidas

Treino

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: Divisão aleatória

	camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Precisão
Teste 1	1	10	tansig, purelin	trainlm	divisão aleatória	1
Teste 2	1	10	tansig, purelin	trainlm	divisão aleatória	1
Teste 3	1	10	tansig, purelin	trainlm	divisão aleatória	0,971
Teste 4	1	10	tansig, purelin	trainlm	divisão aleatória	1
Teste 5	1	10	tansig, purelin	trainlm	divisão aleatória	1
Teste 6	1	10	tansig, purelin	trainlm	divisão aleatória	0,929
Teste 7	1	10	tansig, purelin	trainlm	divisão aleatória	1
Teste 8	1	10	tansig, purelin	trainlm	divisão aleatória	1
Teste 9	1	10	tansig, purelin	trainlm	divisão aleatória	0,957
Teste 10	1	10	tansig, purelin	trainlm	divisão aleatória	0,971
					Precisão Média	0,9828

Estudo

Realizando os 10 testes, visualizamos que os resultados desta alínea foram bastante positivos, acertando praticamente sempre os dígitos com uma média de 98%.

Alínea – B com 1 rede neuronal

Nesta alínea utilizamos a pasta "*train1*" que contém 50 imagens de cada dígito/operador.

Tratamento de imagens

Optámos por repetir o redimensionamento das imagens da alínea anterior em uma resolução de 25x25 e convertemos em matrizes binárias.

Treino 1 (configuração por defeito)

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos		Precisão Global	Precisão Teste
1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		41.428571	18.181818

Estudo

Realizando o primeiro teste, visualizamos que o resultado foi bastante negativo, dando uma precisão global de 41% e 18% de teste. É normal o resultado visto que apenas se usou uma rede de uma camada de 10 neurónios.

Treino com diferentes dimensões das camadas escondidas

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 2
- Número de neurónios: (5,5), (10,10) e (15,15)
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

O número e dimensão das camadas encondidas influencia o desempenho?								
Conf1	2	5, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		37,2857144	9,090909
Conf2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		61,8588715	35,5238095
Conf3	2	15-15	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		70,2857143	29,0909089

Estudo

Realizando os 10 testes com 1 rede de 2 camadas com nº de neurónios diferentes, visualizamos as suas médias.

Os resultados desta alínea foram muito normais, visto que é lógico o número de neurónios ser proporcional ao aumento de precisão no resultado, mas nem sempre pode acontecer, visto que a precisão de teste (a amarelo) na configuração 3 obteve menos que na configuração 2.

Isto acontece por ter havido sorte ao acertar, visto que é aleatório a inicialização dos pesos, mas também por terem sido poucos testes a serem feitos, neste caso foi uma média de 10 testes o resultado.

Treino com diferentes funções de treino

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'traingd','trainscg','traingda','trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

A função de treino influencia o desempenho?								
Conf1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}		9,7142856	8,1818181
Conf2	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}		35,714286	9,090909
Conf3	1	10	tansig, purelin	traingda	dividerand = {0.7, 0.15, 0.15}		19,4285714	5,4545454
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		65,4285715	15,4545453

Estudo

Realizando os 10 testes com 1 rede de 1 camadas com 10 neurónios e funções de treino diferentes, visualizamos as suas médias.

Com grande variedade de valores, destaca-se a configuração 4 com a função de treino "trainln" que melhor resultado oferece, 65% de precisão global (a azul) e 15.5% de precisão de teste (a amarelo).

Treino com diferentes funções de ativação

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'logsig','tansig','satlin','hardlim'*
- Função de ativação da camada de saída: *'purelin', 'logsig'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

As funções de ativação influenciam o desempenho?								
Conf1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		69,957143	20,5714286
Conf2	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}		28,9714286	23,0476191
Conf3	1	10	satlin,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		59,5285715	28,3809525
Conf4	1	10	hardlim, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		25,8142856	20,1904762

Estudo

Realizando os 10 testes com 1 rede de 1 camada com 10 neurónios e funções de ativação diferentes, visualizamos as suas médias.

Como resultado das 4 configurações, há 2 que se destacam pela positiva, não obtendo grandes valores, mas ainda assim positivos.

A configuração 1 com a função de ativação de entrada “logsig” e de saída “purelin” obtiveram os resultados de 70% na precisão global (a azul) e 20.6% na precisão de teste (a amarelo).

A configuração 3 com a função de ativação de entrada “satlin” e de saída “purelin” obtiveram os resultados de 59.5% na precisão global (a azul) e 28% na precisão de teste (a amarelo).

Treino com diferentes divisões

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.33, 0.33, 0.33}, dividerand = {0.9, 0.05, 0.05}, dividerand = {0.2, 0.4, 0.4}, dividerand = {0.4, 0.3, 0.3}, dividerand = {0.8, 0.1, 0.1}

A divisão de exemplos pelos conjuntos influencia o desempenho?							
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.33, 0.33, 0.33}	41,2857142	16,0869565
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05, 0.05}	78,8571428	27,5
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.2, 0.4, 0.4}	27,5714285	11,7857143
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.4, 0.3, 0.3}	49,5714285	20,4761905
Conf5	1	10	tansig, purelin	trainlm	dividerand = {0.8, 0.1, 0.1}	66,5714286	22,2222222

Estudo

Realizando os 10 testes com 1 rede de 1 camada com 10 neurónios e diferentes divisões no treino, visualizamos as suas médias.

A configuração 2 foi o resultado com melhor destaque, com o parâmetro de divisão no treino a 90%, na validação a 5% e de teste 5%,com os resultados de 79% na precisão global (a azul) e 27.5% na precisão de teste (a amarelo).

Alínea – B com 2 redes neuronais

Nesta alínea utilizamos novamente a pasta "*train1*" que contém 50 imagens de cada dígito/operador.

Tratamento de imagens

Optámos por repetir outra vez o redimensionamento das imagens da alínea anterior em uma resolução de 25x25 e convertemos em matrizes binárias.

Treino 1 (configuração por defeito)

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos		Precisão Global	Precisão Teste
Configuração por defeito	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		85,2	50,666667

Estudo

Realizando o primeiro teste, visualizamos que o resultado foi bastante diferente de 1 para 2 redes neuronais, dando uma precisão global de 85% e 50.6% de teste comparando na alínea anterior a precisão global de 41% e 18% de teste.

Treino com diferentes dimensões das camadas escondidas

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 2
- Número de neurónios: (5,5), (10,10) e (15,15)
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

O número e dimensão das camadas encondidas influencia o desempenho?							
Conf1	2	5, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	72,005	53,1333332
Conf2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	84,57	67,0333333
Conf3	2	15-15	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	89,975	68,3333334

Estudo

Realizando os 10 testes com 2 redes de 2 camadas com nº de neurónios diferentes, visualizamos as suas médias.

Os resultados desta alínea foram normais, visto que é lógico que a precisão no resultado seja maior com o aumento do número de neurónios, obtendo 90% na precisão global (a azul) e 68% na precisão de teste (a amarelo) na configuração 3.

Treino com diferentes funções de treino

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'traingd','trainscg','traingda','trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

A função de treino influencia o desempenho?							
Conf1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	55,83	48,3333333
Conf2	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}	67,65	53,5333334
Conf3	1	10	tansig, purelin	traingda	dividerand = {0.7, 0.15, 0.15}	57,09	46,4666667
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	88,56	66,4666667

Estudo

Realizando os 10 testes com 2 redes de 1 camadas com 10 neurónios e funções de treino diferentes, visualizamos as suas médias.

Com grande variedade de valores, destaca-se a configuração 4, tal como já tínhamos visto, com a função de treino “trainln” os resultados são melhores e mais demorados oferecendo, 88.56% de precisão global (a azul) e 66.5% de precisão de teste (a amarelo).

Treino com diferentes funções de ativação

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'logsig','tansig','satlin','hardlim'*
- Função de ativação da camada de saída: *'purelin', 'logsig'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.7, 0.15, 0.15}

As funções de ativação influenciam o desempenho?							
Conf1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	85,575	67,0666667
Conf2	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}	62,4	54,2666666
Conf3	1	10	satlin,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	85,52	61,6333334
Conf4	1	10	hardlim, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	41,72	36,1666667

Estudo

Realizando os 10 testes com 2 redes de 1 camada com 10 neurónios e funções de ativação diferentes, visualizamos as suas médias.

Como resultado das 4 configurações, há as mesmas 2 que se destacam pela positiva, obtendo valores parecidos e positivos.

A configuração 1 com a função de ativação de entrada “logsig” e de saída “purelin” obtiveram os resultados de 85.6% na precisão global (a azul) e 67% na precisão de teste (a amarelo).

A configuração 3 com a função de ativação de entrada “satlin” e de saída “purelin” obtiveram os resultados de 85.5% na precisão global (a azul) e 61.6% na precisão de teste (a amarelo).

Treino com diferentes divisões

A nossa rede neuronal de arquitetura *feedforwardne*t como as seguintes características:

- Número de camadas escondida: 1
- Número de neurónios: 10
- Função de ativação da camada escondida: *'tansig'*
- Função de ativação da camada de saída: *'purelin'*
- Função de treino: *'trainlm'*
- Número de épocas: 1000
- Parâmetros de divisão: dividerand = {0.33, 0.33, 0.33}, dividerand = {0.9, 0.05, 0.05}, dividerand = {0.2, 0.4, 0.4}, dividerand = {0.4, 0.3, 0.3}, dividerand = {0.8, 0.1, 0.1}

A divisão de exemplos pelos conjuntos influencia o desempenho?							
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.33, 0.33, 0.33}	74,15	62,3152202
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05, 0.05}	93,885	61,6
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.2, 0.4, 0.4}	65,875	58,575
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.4, 0.3, 0.3}	78,06	63,0833334
Conf5	1	10	tansig, purelin	trainlm	dividerand = {0.8, 0.1, 0.1}	83,305	62,1166666

Estudo

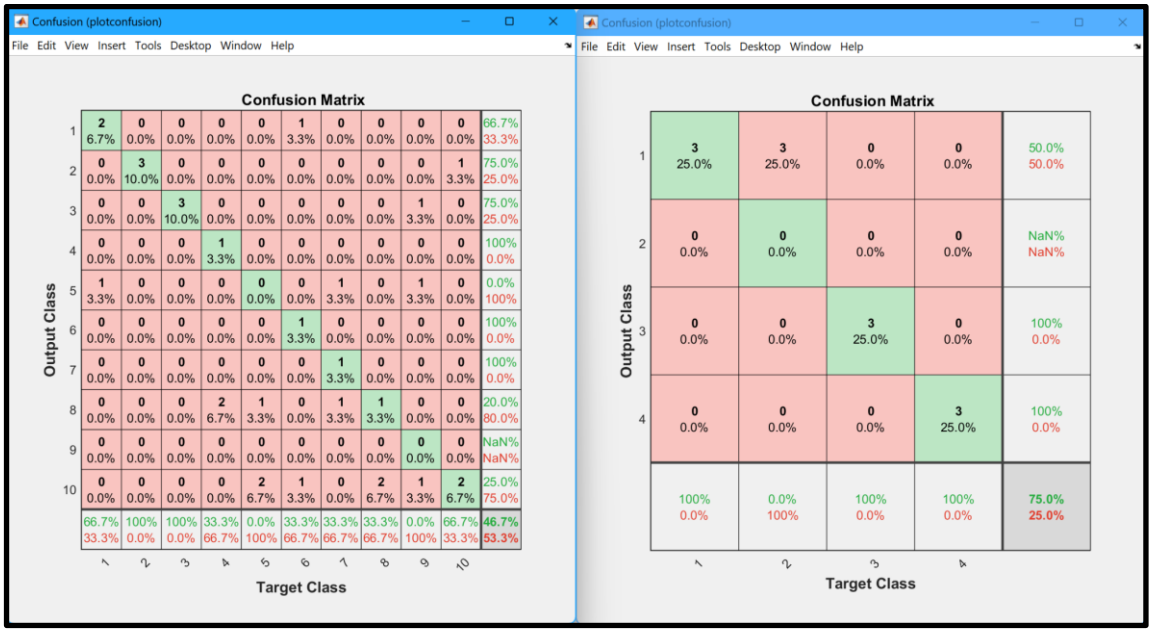
Realizando os 10 testes com 2 redes de 1 camada com 10 neurónios e diferentes divisões no treino, visualizamos as suas médias.

A configuração 2 foi o resultado como na alínea anterior, com o parâmetro de divisão no treino a 90%, na validação a 5% e de teste 5%,com os resultados de 94% na precisão global (a azul) e 61.6% na precisão de teste (a amarelo).

Alínea – C (dataset novo)

Nesta alínea foi realizado um novo dataset para realização de testes com a melhor rede obtida da alínea anterior.

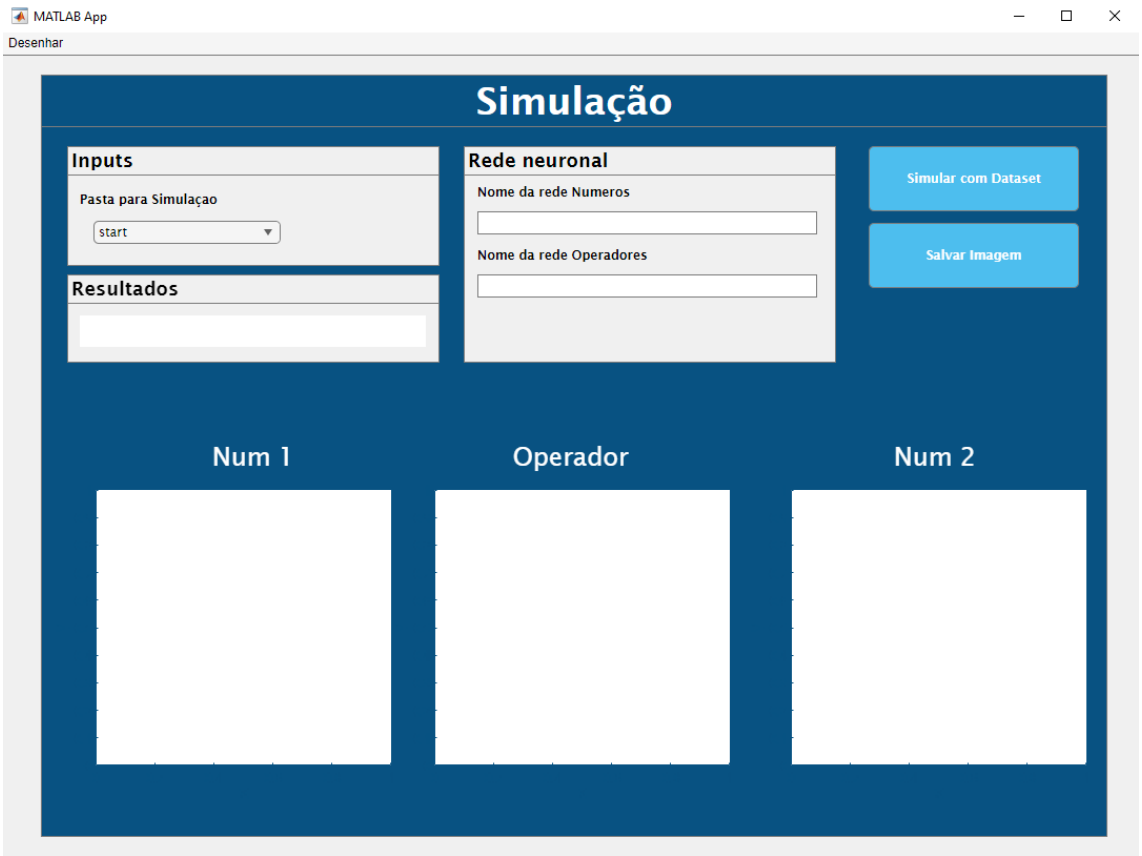
Obtemos uma precisão de 46.7% na classificação dos dígitos e 75.0% nos operadores.



Estudo

Tivemos mau desempenho ao classificar os dígitos, provavelmente deverá ser do facto de redimensionarmos as imagens para um tamanho muito pequeno (devido a limitações quer de tempo quer de hardware). Nos operadores tivemos bons resultados na maior parte deles exceto no de divisão que muitas vezes foi confundido com o operador de soma, mais uma vez poderá ser explicado por causa do redimensionamento.

APP (apenas parte gráfica)



Não implementámos código por falta de tempo e falta de experiência para a parte do desenho.

Conclusão

De forma geral, ficamos muito satisfeitos com o resultado final.

Observamos que a parametrização das redes é crucial no seu desempenho, sendo esta dependente do problema em questão.

O tratamento prévio das imagens é de alta importância, permite à rede focar-se nas características que são mais relevantes ao problema como, neste caso, a forma dos dígitos.

Foi um projeto com altos e baixos, que com esforço, pesquisa e muito trabalho foi-se orientando aos poucos.

Concluímos este trabalho sobre redes neuronais, que consideramos ter sido um sucesso,

tanto em termos de aprendizagem e aplicação de conceitos como em termos de desenvolvimento de redes neuronais com desempenhos aceitáveis.

Referências bibliográficas

MathWorks. "Deep Learning Toolbox Documentation". MathWorks. Disponível em:

[https://www.mathworks.com/help/deeplearning/index.html]

MathWorks. "Matlab Documentation". MathWorks. Disponível em:

[https://www.mathworks.com/help/matlab/]