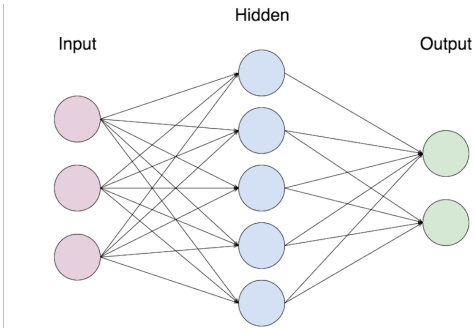
Instituto Superior de Engenharia

Politécnico de Coimbra

Conhecimento e Raciocínio

2022 – 2023



Redes Neuronais

Reconhecimento de Números

Trabalho realizado por:

Rúben Santos – 2019116244

Pedro Brás – 2017008938

Índice

Conhecimento e Raciocínio *1*

2022 – 2023 ..... *1*

Redes Neuronais *1*

Reconhecimento de Números *1*

Introdução *3*

Decisões tomadas *4*

Tratamento das Imagens **4**

Targets e Inputs **4**

Guardar Net's **4**

Treino e Estudo Estatístico *6*

Alínea a) **6**

Alínea b) **8**

Uma só Net 8

Nets Separadas 10

Alínea c) **12**

Observação Geral *14*

Aplicação *15*

Conclusões *17*

Bibliografia *18*

Introdução

Este trabalho foi realizado no âmbito da Unidade Curricular Conhecimento e Raciocínio, tendo como objetivo o desenvolvimento e estudo estatístico de redes neuronais para identificação de imagens de dígitos de 0 a 9 e operadores de adição, subtração, multiplicação e divisão.

A linguagem e tecnologia utilizada para o desenvolvimento do trabalho prático é o MATLAB e o estudo é feito recorrendo a redes neuronais do tipo feedforward com alteração de diversos parâmetros para efeitos de comparação e teste.

Adicionalmente, o trabalho prático inclui uma aplicação gráfica que permite criar, treinar, simular redes e desenhar dígitos ou operadores para serem reconhecidos por uma rede.

Decisões tomadas

Tratamento das Imagens

Para proceder ao tratamento das imagens primeiro tivemos de definir os *paths* dos *datasets* onde se encontram as imagens que se encontram fornecidas.

Para este projeto existem 3 *datasets*, sendo eles **start**, **train** e **custom**. Cada *dataset* tem sempre 14 pastas dentro, que são os dígitos de 0 a 9 e os operadores de adicação, subtração, multiplicação e divisão.

Sabendo que o número de pastas é fixo, depois de obter o *path* do *dataset*, pecorremos todas as 14 pastas, entrando por ordem em cada uma delas.

Dentro das pastas, percorremos todas as imagens e para cada uma, a fim de otimizar a leitura das imagens definimos um tamanho de 25x25 (original é 150x150). De seguida são colocadas em matrizes binárias e convertidas em uma única coluna, pois, no MATLAB, as redes neuronais recebem os seus *inputs* na forma de colunas de uma matriz.

Todos os ficheiros além do **trainFunction_d**, que corresponde à alínea d), recorrem a dois *for loops* para percorrer todas as pastas e os ficheiros dentro delas. Para a alínea d) como recorre a uma funcionalidade que apenas requer a leitura de uma imagem separada, criamos outro ficheiro para retornar a matriz binaria e o *target* de apenas essa imagem em específico.

Targets e Inputs

No fim de ter obtido todas as matrizes binarias das imagens, são gerados 14 vetores com uma dimensão igual ao número de imagens que se encontram dentro da pasta do *dataset* lido. Esta atribuição é feita pela ordem de leitura das pastas, ou seja, são 14 vetores que correspondem às pastas dos dígitos de 0 a 9, add, div, mul e sub.

Guardar Net's

Para cada alínea a rede é treinada 10 vezes para que se possa obter uma média do valor global de precisão e do valor de precisão teste. Com esses valores

comparamos as redes ao longo das execuções e vamos sempre guardando a rede com melhores valores para depois ser guardada dentro da pasta ‘networks’.

Treino e Estudo Estatístico

Alínea a)

Nesta alínea foram feitos testes o *dataset* da pasta 'start'. Este *dataset* é composto por 5 imagens de cada dígito e operador.

Após fazer o teste com a configuração base com uma camada escondida com 10 neurónios, percebemos que há um aumento significativo quando é aumentado o número de camadas escondidas assim como o número de neurónios por camada.

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}		60	56.36
Conf2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		77.14	68.18
Conf3	3	5, 10, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		53.57	43.64
Conf4	3	10, 10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		75.14	67.27

No próximo teste, foram testadas várias parametrizações dos parâmetros de divisão de treino, sendo que neste teste foi possível obter melhores resultados do que no teste anterior em que se alterou a dimensão das camadas escondidas e número de neurónios por camada.

Podemos reparar que ao utilizar um valor elevado para o parâmetro de treino em relação ao teste base, gerou sempre melhores resultados comparando com os outros testes em que esse parâmetro estava com um valor próximo aos parâmetros de teste e validação.

A divisão de exemplos pelos conjuntos influencia o desempenho?								
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.33, 0.33, 0.33}		68.86	64.35
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05, 0.05}		85.71	80
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.80, 0.10, 0.10}		82.29	67.1
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.50, 0.25, 0.25}		52.86	45
Conf5	1	10	tansig, purelin	trainlm	dividerand = {0.60, 0.20, 0.20}		71.71	73.57
Conf6	1	10	tansig, purelin	trainlm	dividerand = {0.40, 0.30, 0.30}		64	56.19

Agora onde obtivemos de facto o melhor resultado, tanto para a precisão de teste como para a precisão de treino nos testes das funções de ativação.

Aqui podemos notar com grande distinção que a combinação da função logsig e purelin foi a única que se destacou, gerando um resultado muito bom comparado com os restantes testes.

As funções de ativação influenciam o desempenho?								
Conf1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		90	85.45
Conf2	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}		5.71	7.27
Conf3	1	10	compet, elliotsig	trainlm	dividerand = {0.7, 0.15, 0.15}		23	22.73
Conf4	1	10	elliotsig, compet	trainlm	dividerand = {0.7, 0.15, 0.15}		7.14	10.91
Conf5	1	10	hardlim, hardlims	trainlm	dividerand = {0.7, 0.15, 0.15}		7.14	10
Conf6	1	10	hardlims, hardlim	trainlm	dividerand = {0.7, 0.15, 0.15}		7.14	9.09

Ao alterar as funções de treino foi onde obtivemos os piores tempos de execução, sendo extremamente demorados e a atingirem quase sempre 1000 épocas nas primeiras iterações para as funções **traingd** e **trainbfg**. Para além do tempo de execução elevado os resultados foram muito baixos comparando com a função **trainlm** que demonstrou uma melhor aprendizagem.

A função de treino influencia o desempenho?								
Conf1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}		66.71	57.27
Conf2	1	10	tansig, purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}		10.57	6.35
Conf3	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}		22.86	30
Conf4	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}		11.71	11.82

No entanto apesar de se ter obtido valores bons em geral, não é ideal efetuar um estudo apenas a este *dataset* devido ao baixo número de imagens que cada dígito tem, sendo que deve ser tomado apenas como comparação para *dataset* mais extensos.

Alínea b)

Uma só Net

Nesta alínea foram feitos testes o *dataset* da pasta 'train'. Este *dataset* é composto por 50 imagens de cada dígito e operador.

Como se pode ver, à semelhança da alínea anterior o número de neurónios e camadas escondidas tem um papel importante neste teste, visto que com o aumento dos mesmos, os valores de precisão também tiveram um aumento.

Porém com o aumento de neurónios e camadas escondidas o tempo de execução do teste e a memória associada também sofrem um aumento.

Ao concluir os testes podemos perceber que os valores são significativamente melhores do que os obtidos no teste da alínea anterior quando o número de neurónios e camadas escondidas é maior.

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}		40.27	37.81
Conf2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		81.16	76.19
Conf3	3	5, 10, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		53.86	48.67
Conf4	3	10, 10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		86.19	83.62

Para as funções de treino, os resultados obtidos foram semelhantes aos da alínea anterior sendo que o tempo de execução foi extremamente elevado para as funções **traingd** e **trainbfg** e, em geral, os resultados foram muito baixos comparando com a **trainlm** que demonstrou bons resultados.

A função de treino influencia o desempenho?								
Conf1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}		21.2	19.62
Conf2	1	10	tansig, purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}		7.43	8.19
Conf3	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}		29.41	30.48
Conf4	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}		7.19	6.29

Para testar a função **traingd** de outras formas, visto que os resultados tinham sido muito baixos e o tempo de execução extremamente alto, decidimos experimentar diferentes parâmetros de divisão de tempo, alterando o parâmetro de treino e teste, e deixando o parâmetro de validação a 0.

Com este teste podemos perceber que sem a existência da validação o tempo de execução do teste foi muito menor e os resultados foram relativamente melhores comparados com o teste anterior.

O parâmetro de validação influencia o desempenho?								
Conf1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0, 0.3}		45.59	40.33
Conf2	1	10	tansig, purelin	traingd	dividerand = {0.5, 0, 0.5}		37.46	27.91
Conf3	1	10	tansig, purelin	traingd	dividerand = {0.3, 0, 0.7}		34.86	31.24

Mais uma vez a combinação de funções de ativação que se destacou foi a logsig e purelin, que permitiram obter melhores precisões que as restantes testadas.

As funções de ativação influenciam o desempenho?								
Conf1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		84.4	80.29
Conf2	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}		43.39	41.33
Conf3	1	10	compet, elliotsig	trainlm	dividerand = {0.7, 0.15, 0.15}		12.46	11.71
Conf4	1	10	elliotsig, compet	trainlm	dividerand = {0.7, 0.15, 0.15}		7.14	7.71
Conf5	1	10	hardlim, hardlims	trainlm	dividerand = {0.7, 0.15, 0.15}		11	10.67
Conf6	1	10	hardlims, hardlim	trainlm	dividerand = {0.7, 0.15, 0.15}		9	8.75

Por fim, passando aos testes de feitos à divisão de treino podemos notar com grande distinção que a combinação da função logsig e purelin foi a única que se destacou, gerando um resultado muito bom comparado com os restantes testes.

A divisão de exemplos pelos conjuntos influencia o desempenho?							
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.33, 0.33, 0.33}	59.46	56.87
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05, 0.05}	85.14	80.86
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.80, 0.10, 0.10}	75.43	71.14
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.50, 0.25, 0.25}	76.13	71.54
Conf5	1	10	tansig, purelin	trainlm	dividerand = {0.60, 0.20, 0.20}	67.94	64.21
Conf6	1	10	tansig, purelin	trainlm	dividerand = {0.40, 0.30, 0.30}	64.53	61.38

Nets Separadas

Ao alterar o número de camadas escondidas e neurónios podemos perceber que o comportamento é semelhante aos testes anteriores, em que um número mais elevado de camadas escondidas e neurónios, permite atingir melhores resultados. Porém com isso acabamos por ser prejudicados no tempo de execução e memória que também sofrem um aumento significativo.

Comparando com os testes anteriores, os valores de precisão global para estes testes são muito próximos de 100% quando temos 3 camadas de escondidas com 10 neurónios em cada uma.

O número e dimensão das camadas encondidas influencia o						Digitos		Operadores	
Conf1	2	5, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	60.16	61.47	91	8.67
Conf2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	82.29	79.47	97.3	12.67
Conf3	3	5, 10, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	60.66	59.6	95.8	13.33
Conf4	3	10, 10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	97.5	16.67	97.5	23.33

Como nas alíneas anteriores, tempo de execução e precisão dos testes foram muito prejudicados para a rede dos dígitos. Para a rede dos operadores os resultados foram muito bons comparando com as redes das alíneas anteriores.

A função de treino influencia o desempenho?						Digitos		Operadores	
Conf1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	54.24	53.2	89.1	4
Conf2	1	10	tansig, purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}	44.9	42.13	50.35	0.33
Conf3	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}	48.74	49.87	92.85	11
Conf4	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}	11.32	9.33	74.6	27.67

Ao alterar as funções de ativação foi novamente com a combinação da função logsig com a purelin que obtivemos os melhores resultados para ambas as redes neste teste.

As funções de ativação influenciam o desempenho?						Digitos		Operadores	
Conf1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	93.58	91.07	97.9	13.33
Conf2	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}	57.9	57.6	95.25	12.33
Conf3	1	10	compet, elliotsig	trainlm	dividerand = {0.7, 0.15, 0.15}	22.6	23.87	42.3	0
Conf4	1	10	elliotsig, compet	trainlm	dividerand = {0.7, 0.15, 0.15}	11.2	12.8	25	2
Conf5	1	10	hardlim, hardlims	trainlm	dividerand = {0.7, 0.15, 0.15}	9.8	9.33	27	16.33
Conf6	1	10	hardlims, hardlim	trainlm	dividerand = {0.7, 0.15, 0.15}	10	9.07	21.5	23.33

Para a alteração dos parâmetros de divisão de treino também obtivemos bons resultados em ambas as redes sendo que a rede dos dígitos obteve resultados muito bons em quase todos os testes.

A divisão de exemplos pelos conjuntos influencia o desempenho?						Digitos		Operadores	
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.33, 0.33, 0.33}	82.24	79.64	89	9.4
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05, 0.05}	91.68	87.2	99.2	8
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.80, 0.10, 0.10}	91.52	87	98.35	12
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.50, 0.25, 0.25}	77.68	72.4	93.2	15.8
Conf5	1	10	tansig, purelin	trainlm	dividerand = {0.60, 0.20, 0.20}	89.86	86.2	97.1	11.25
Conf6	1	10	tansig, purelin	trainlm	dividerand = {0.40, 0.30, 0.30}	81.44	75.87	92.25	8.33

Em geral, a rede dos operadores divido a ser uma rede com uma complexidade de aprendizagem menor, obteve muitos bons resultados em todos os testes, enquanto a rede dos dígitos devido a ter um maior número de imagens acabou por não ter um desempenho igual, mas comparando com o teste em que foi feito tudo na mesma rede, os resultados ao separar foram muito melhores.

Alínea c)

Nesta alínea foram feitos testes o *dataset* da pasta 'custom_draw'. Este *dataset* é composto por 3 imagens de cada dígito e operador.

Este *dataset* foi criado através da nossa GUI em que recorremos à funcionalidade de desenho para gerar as imagens dos dígitos e operadores.

Ao alterar o número de camadas escondidas e neurónios podemos perceber que o comportamento é semelhante aos testes anteriores, em que um número mais elevado de camadas escondidas e neurónios, permite atingir melhores resultados.

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}		40.71	35
Conf2	2	10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		83.10	68.33
Conf3	3	5, 10, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		48.1	40
Conf4	3	10, 10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		83.1	81.67

À semelhança das alíneas anterior, tempo de execução muito elevado para as funções **traingd** e **trainbfg**, porém para as restantes funções obtivemos melhores resultados do que nos outros *datasets*.

A função de treino influencia o desempenho?								
Conf1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}		42.14	38.33
Conf2	1	10	tansig, purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}		39.52	33.33
Conf3	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}		59.52	45
Conf4	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}		52.14	43.33

Os testes de função de ativação mostram um comportamento semelhante aos testes feitos com outros *datasets* em que a combinação da função logsig e purelin é a que demonstra melhores resultados.

As funções de ativação influenciam o desempenho?								
Conf1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}		87.62	86.67
Conf2	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}		39.52	25
Conf3	1	10	compet, elliotsig	trainlm	dividerand = {0.7, 0.15, 0.15}		30.71	18.33
Conf4	1	10	elliotsig, compet	trainlm	dividerand = {0.7, 0.15, 0.15}		7.14	8.33
Conf5	1	10	hardlim, hardlims	trainlm	dividerand = {0.7, 0.15, 0.15}		4.76	3.33
Conf6	1	10	hardlims, hardlim	trainlm	dividerand = {0.7, 0.15, 0.15}		9.52	8.33

No teste de vários parâmetros de divisão de treino o que teste que se voltou a destacar foi o que teve um parâmetro de treino mais elevado.

A divisão de exemplos pelos conjuntos influencia o desempenho?								
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.33, 0.33, 0.33}		54.29	52.14
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.9, 0.05, 0.05}		89.05	75
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.80, 0.10, 0.10}		85	75
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.50, 0.25, 0.25}		62.14	62.73
Conf5	1	10	tansig, purelin	trainlm	dividerand = {0.60, 0.20, 0.20}		73.57	67.5
Conf6	1	10	tansig, purelin	trainlm	dividerand = {0.40, 0.30, 0.30}		70.24	57.69

Observação Geral

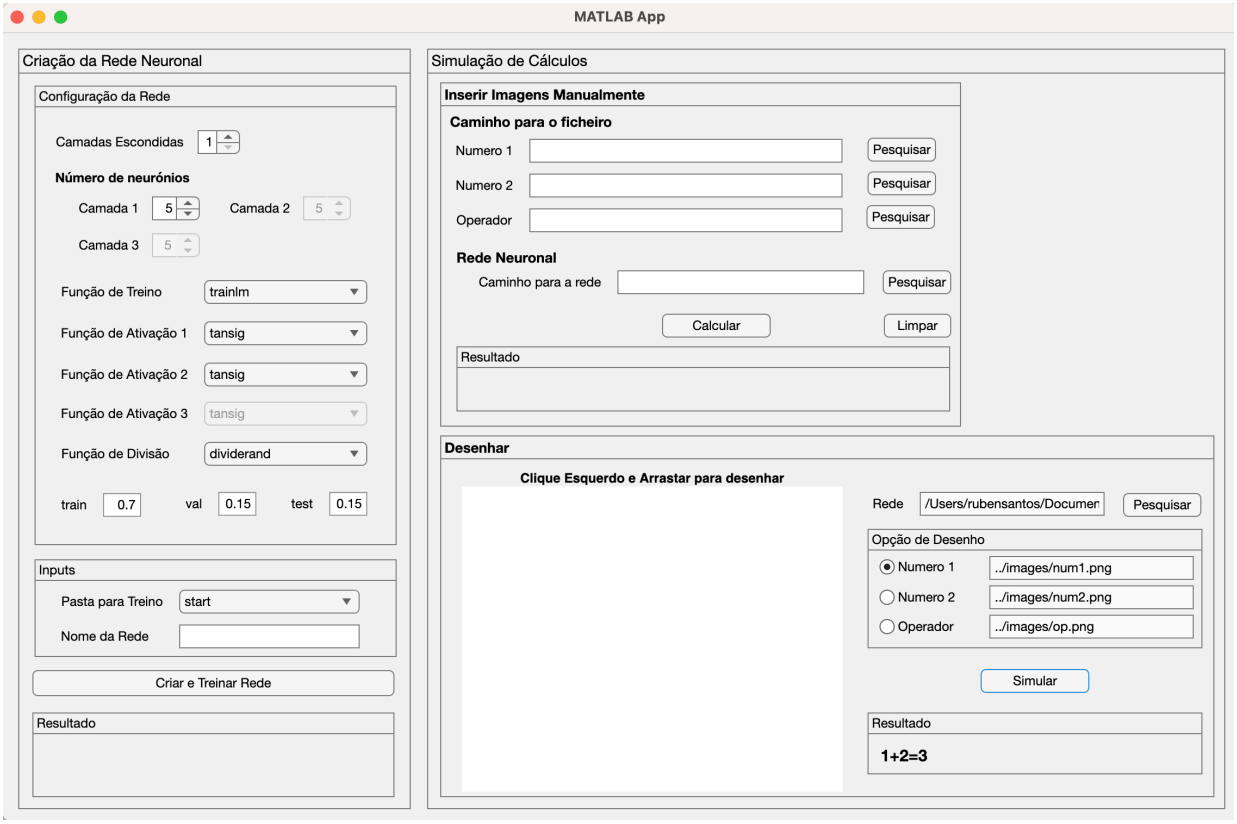
No fim de testar todos os *datasets* com diferentes parametrizações podemos concluir que os melhores resultados que obtivemos para a precisão global foram obtidos ao separar as redes, uma para dígitos e outra para operadores. Porém a precisão de teste ficou muito abaixo dos testes com apenas uma rede.

Sendo assim a rede que demonstrou um valor mais alto em ambas as precisões foi o teste ao *dataset* 'custom_draw', seguido do *dataset* 'start' que também obteve boas precisões.

Aplicação

A nossa aplicação é distribuída em 3 partes, sendo elas:

- Criar uma rede neuronal
    - No painel 'Criação de Rede Neuronal' pode criar uma rede neuronal com os parâmetros que pretender, dar um nome à rede e selecionar qual *dataset* quer treinar (start, train ou custom_draw).
- Simular um cálculo de forma manual ao escolher as imagens
    - No painel 'Inserir Imagens Manualmente' pode pesquisar no seu computador, ou nos datasets fornecidos, uma imagem para o número 1, número 2 e operador. Depois basta selecionar qual a rede que pretende para avaliar as imagens.
    - O output será mostrado no painel 'Resultado', em que a ordem é: numero1 operador numero2 = resultado da operação.
- Simular um cálculo ao desenhar os dígitos
    - À semelhança de introduzir as imagens manualmente, no painel 'Desenhar' é o utilizador que desenha no quadrado branco ao pressionar o botão esquerdo do rato e arrastar para criar uma forma. Para escolher qual dígito se pretende desenhar basta selecionar o radio button correspondente (Número 1, Número 2 ou Operador). Também deve selecionar qual rede pretende usar.
    - O output será mostrado no painel 'Resultado', em que a ordem é: numero1 operador numero2 = resultado da operação.


MATLAB App
Criação da Rede Neuronal
Configuração da Rede
Camadas Escondidas
1
Número de neurónios
Camada 1
5
Camada 2
5
Camada 3
5
Função de Treino
trainlm
Função de Ativação 1
tansig
Função de Ativação 2
tansig
Função de Ativação 3
tansig
Função de Divisão
dividerand
train
0.7
val
0.15
test
0.15
Inputs
Pasta para Treino
start
Nome da Rede
Criar e Treinar Rede
Resultado
Simulação de Cálculos
Inserir Imagens Manualmente
Caminho para o ficheiro
Numero 1
Pesquisar
Numero 2
Pesquisar
Operador
Pesquisar
Rede Neuronal
Caminho para a rede
Pesquisar
Calcular
Limpar
Resultado
Desenhar
Clique Esquerdo e Arrastar para desenhar
Rede
/Users/rubensantos/Documen
Pesquisar
Opção de Desenho
Numero 1
../images/num1.png
Numero 2
../images/num2.png
Operador
../images/op.png
Simular
Resultado
1+2=3

Conclusões

Com este trabalho prático podemos concluir que não há apenas uma parametrização predefinida que obtenha os melhores resultados. Para determinados datasets o melhor é testar vários tipos de parametrização e tirar as conclusões a partir daí.

Concluímos que o trabalho sobre o tema redes neuronais foi terminado com sucesso ao conseguir executar todas as tarefas pedidas no enunciado e com isso evoluímos o nosso conhecimento sobre redes neuronais.

Bibliografia