

Conhecimento e Raciocínio - 2022/2023

Tema 1 - Redes Neurais

Reconhecimento de dígitos e símbolos
matemáticos

Índice

A) Funções de manipulação de imagem	2
Tratamento prévio das imagens	2
Resultados de treino	3
Plotconfusion dos resultados	4
Análise e conclusões	5
B) Testes de arquiteturas e parametrizações	6
Resultados de treino	6
O número e dimensão das camadas escondidas influencia o desempenho?	7
A função de treino influencia o desempenho?	8
As funções de ativação influenciam o desempenho?	9
A divisão dos exemplos pelos conjuntos influencia o desempenho?	10
A função de saída influencia o desempenho?	11
As funções de aprendizagem influenciam o desempenho?	12
Resultados dos tipos de treino com diferentes funções de treino	13
Combinação dos melhores resultados e testes	14
Análise e conclusões obtidas	16
C) Dataset manual	17
Criação do dataset	17
Resultados de classificações	18
Resultados do melhor modelo de números	18
Resultados do melhor modelo de operadores	18
Resultados do melhor modelo mix	18
Análise e conclusões relevantes	19
D) Aplicação gráfica	20
Apresentação da interface	20
Identificação de um número/operador	21
Deteção das componentes da expressão	22
Resultados obtidos	24
Conclusão	26

A) Funções de manipulação de imagem

Tratamento prévio das imagens

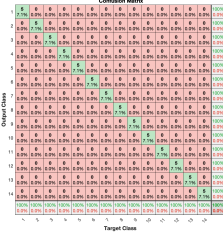
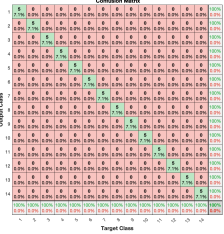
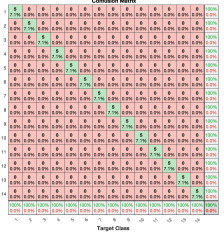
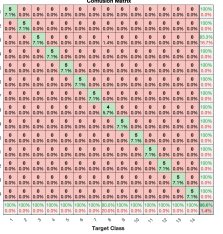
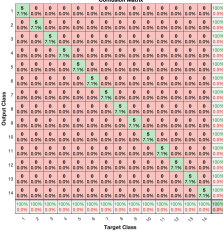
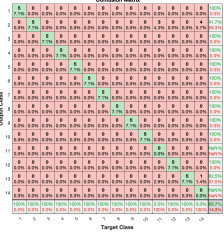
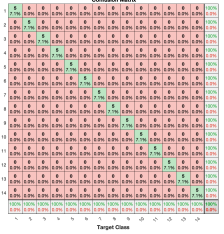
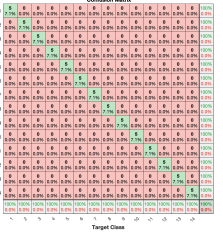
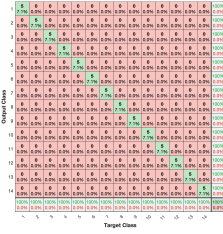
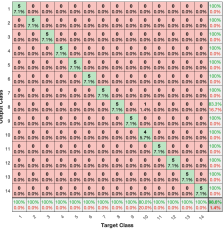
Ao iniciar o projeto, tentamos utilizar as imagens fornecidas, que estavam no tamanho de 150x150 pixels. No entanto, percebemos que esse tamanho estava a resultar em tempos de treino excessivamente longos. Portanto, utilizamos as funções da toolbox de processamento de imagem do Matlab para redimensionar as imagens, e desta forma garantimos que o tamanho mínimo das imagens fosse de 25x25 pixels, evitando a perda significativa de informações.

```
% Resize the image to the desired size  
resizedImg = imresize(img, [imageSize imageSize]);
```

Resultados de treino

	Número de camadas escondidas	Número de neurónios	Funções de ativação
Configuração por defeito	1	10	tansig, tansig
TREINOS	Precisão Global	Epoch	Tempo de Execução
1	100%	1000	00:03
2	100%	1000	00:01
3	100%	1000	00:03
4	98.57%	1000	00:02
5	100%	1000	00:02
6	85.71%	1000	00:01
7	100%	1000	00:02
8	100%	1000	00:02
9	100%	1000	00:01
10	98.57%	1000	00:03

Plotconfusion dos resultados

Treino 1 (100%)	Treino 2 (100%)	Treino 3 (100%)	Treino 4 (98,57%)
			
Treino 5 (100%)	Treino 6 (85,71%)	Treino 7 (100%)	Treino 8 (100%)
			
Treino 9 (100%)	Treino 10 (98,57%)		
			

Análise e conclusões

Criamos um modelo base, com uma camada escondida contendo 10 neurónios, com a função de ativação "tansig" e utilizamos uma rede neuronal feedforward que foi submetida a 10 treinos independentes utilizando os caracteres da pasta "start".

É importante destacar que, para simplificar o processo de treino e análise, não foi realizada a divisão dos dados em conjuntos de treino e validação. Todos os dados disponíveis foram utilizados para o treino e teste da rede neural. Esta abordagem foi adotada com o intuito de obter uma visão inicial do desempenho do modelo.

Observamos que a precisão global variou entre 85,71% e 100%, evidenciando um desempenho geralmente satisfatório do modelo. A maioria dos treinos alcançou uma precisão de 100%, indicando que a rede neural foi capaz de classificar corretamente todos os dígitos e símbolos matemáticos nas imagens de teste.

Notamos que ao testar o modelo com as mesmas imagens usadas no treino, o modelo pode se tornar "viciado" e memorizar os padrões específicos dessas imagens, sem generalizar adequadamente para novos dados.

Para uma avaliação mais precisa e confiável do desempenho do modelo, notamos que é melhor dividir os dados em conjuntos separados de treino, validação e teste. O conjunto de treino é utilizado para treinar o modelo, o conjunto de validação é usado para ajustar os hiperparâmetros do modelo durante o treino, e o conjunto de teste é reservado exclusivamente para avaliar o desempenho final do modelo após o treino. Esta abordagem permite verificar se o modelo é capaz de generalizar corretamente para os novos dados, não vistos durante o treino.

B) Testes de arquiteturas e parametrizações

Resultados de treino

Para obter os melhores resultados de desempenho foram testadas várias combinações de uma rede neuronal feedforward em que na qual foram alteradas parâmetros tais como número de camadas escondidas, número de neurónios, funções de ativação, funções de treino, divisão de exemplos, funções de aprendizagem, taxas de aprendizagem e em alguns casos número de repetições e epoch.

O objetivo destas alterações de parâmetros foi melhorar a precisão global e a precisão de testes da rede.

O treino foi dividido em oito quadros, em que em sete foi alterado somente um parâmetro da rede, e o restante foi um combinar de alterações que achamos mais impactantes a fim de obter o melhor resultado. Cada quadro de testes contém dez alterações e cada uma foi repetida 10x para obter uma média.

O exemplo base que a partir do qual utilizamos para fazer as diferentes combinações foi o seguinte:

- N.º de camadas escondidas: **10**;
- Funções de ativação: **tansig** e **purelin**;
- Função de treino: **trainlm**;
- Divisão de exemplo: **dividerand** com **70%** para treino, **15%** para validação e **15%** para testes;
- Função de aprendizagem: **learngdm**;
- N.º de repetições; **10x**;

Depois da repetição sucessiva das redes 10x conseguimos obter uma média de precisão global de 64.85% e uma média de precisão de testes de 16.66%;

O número e dimensão das camadas escondidas influencia o desempenho?

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
1	2	5, 5	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	30,87	7,04	-	1000	10	MIX
2	2	10,1	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	14,34	6,38	-	1000	10	MIX
3	2	1, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	19,43	7,71	-	1000	10	MIX
4	2	10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	58,64	11,71	-	1000	10	MIX
5	2	20, 20	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	81,43	29,33	-	1000	10	MIX
6	3	05, 5, 5	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	24	9	-	1000	10	MIX
7	3	10, 1, 1	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	13,32	8,38	-	1000	10	MIX
8	3	01, 10, 1	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	10, 05	7, 5228	-	1000	10	MIX
9	3	10, 10, 10	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	57,75	14,1	-	1000	10	MIX
10	3	10, 5, 10	tansig, tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	50,22	17,14	-	1000	10	MIX

Para uma variação do número de camadas entre 2 e 3 e o número de neurónios em que o menor foi 1 e maior foi 20, podemos confirmar que a melhor configuração conseguida foi com 2 camadas e com 20 neurónios em cada uma delas com 81,43% precisão global e 29,33% precisão de teste.

A função de treino influencia o desempenho?

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
11	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	learngdm	10	12,33	7,43	-	1000	10	MIX
12	1	10	tansig, purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}	learngdm	10	12	9	-	1000	10	MIX
13	1	10	tansig, purelin	trainbr	dividerand = {0.7, 0.15, 0.15}	learngdm	10	11.51	8.95	-	1000	10	MIX
14	1	10	tansig, purelin	trainc	dividerand = {0.7, 0.15, 0.15}	learngdm	10	58,2	30,81	-	1000	10	MIX
15	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}	learngdm	10	22,63	6,86	-	1000	10	MIX
16	1	10	tansig, purelin	trainrp	dividerand = {0.7, 0.15, 0.15}	learngdm	10	15,457	9,52	-	1000	10	MIX
17	1	10	tansig, purelin	traingda	dividerand = {0.7, 0.15, 0.15}	learngdm	10	27,757	8,76	-	1000	10	MIX
18	1	10	tansig, purelin	traingdx	dividerand = {0.7, 0.15, 0.15}	learngdm	10	35,3	9,33	-	1000	10	MIX
19	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}	learngdm	10	12,98	7,71	-	1000	10	MIX
20	1	10	tansig, purelin	traincgb	dividerand = {0.7, 0.15, 0.15}	learngdm	10	23,87	10	-	1000	10	MIX

Das funções de treino utilizadas no quadro acima a que se destacou mais foi a “trainc” obtendo 58,2 de precisão global e 30,81 de precisão de teste. Claramente foi um destaque de todas as funções, contudo, podemos assumir que existe um efeito de overfitting e é necessário um elevado período para o modelo.

As funções de ativação influenciam o desempenho?

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
21	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	58,64	16,57	-	1000	10	MIX
22	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	59,78	16,09	-	1000	10	MIX
23	1	10	logsig, tansig	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	56,54	16	-	1000	10	MIX
24	1	10	compet, hardlim	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	17,12	6,47	-	1000	10	MIX
25	1	10	logsig, netinv	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	59,01	15,33	-	1000	10	MIX
26	1	10	poslin, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	41,52	14,38	-	1000	10	MIX
27	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	67,57	18,47	-	1000	10	MIX
28	1	10	hardlim, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	65,95	17,71	-	1000	10	MIX
29	1	10	hardlims, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	25,81	7,9	-	1000	10	MIX
30	1	10	radbas, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	learngdm	10	56,95	12,57	-	1000	10	MIX

Pegando no modelo base e alterando a função de ativação podemos concluir que as diferentes combinações de funções de ativação na camada escondida não tiveram um impacto significativo na precisão global e na precisão de teste. Ainda assim, o melhor resultado foi obtido através da função “trainlm”, utilizada na configuração base com 67,57% precisão global e 18,47% precisão de teste.

A divisão dos exemplos pelos conjuntos influencia o desempenho?

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
31	1	650 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.33, 0.33, 0.33}	learnngdm	10	60,37	27,72	-	1000	10	MIX
32	1	651 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.9, 0.05, 0.05}	learnngdm	10	69,78	19,14	-	1000	10	MIX
33	1	652 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.5, 0.25, 0.25}	learnngdm	10	67,2	27,71	-	1000	10	MIX
34	1	653 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.1, 0.8, 0.1}	learnngdm	10	29,91	18,28	-	1000	10	MIX
35	1	654 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.1, 0.1, 0.9}	learnngdm	10	27,28	15,46	-	1000	10	MIX
36	1	655 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.25, 0.25, 0.5}	learnngdm	10	52,27	27,22	-	1000	10	MIX
37	1	656 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.8, 0.1, 0.1}	learnngdm	10	69,85	26,28	-	1000	10	MIX
38	1	657 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.70, 0.15, 0.15}	learnngdm	10	70,35	24,76	-	1000	10	MIX
39	1	658 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.25, 0.5, 0.25}	learnngdm	10	51,91	28,22	-	1000	10	MIX
40	1	659 200 100	tansig, tansig, tansig, purelin	traingd	dividerand = {0.25, 0.25, 0.5}	learnngdm	10	53,32	27,28	-	1000	10	MIX

Neste quadro foi alterado para além das divisões de exemplos também foi alterado o número de neurónios varia entre 100 e 659, as funções de ativação para “tansig” e “purelin” e a função de treino para “traingd”. Esta configuração foi mantida durante todos os testes deste quadro.

Ao analisar os resultados, é possível concluir que as configurações variam em termos de precisão global e precisão de teste. No entanto, não há uma configuração específica que se destaque consistentemente em termos de desempenho. Ainda assim, o melhor resultado foi obtido através da divisão de exemplos 80% para treino, 10% para validação e 10% para testes com 69,85% precisão global e 26,28% precisão de teste.

A função de saída influencia o desempenho?

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
41	1	10	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	23,143	11,429	-	1000	10	MIX
42	1	10	tansig, hadrlim	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	18,5	9,2	-	1000	10	MIX
43	1	10	tansig, logsig	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	20	7	-	1000	10	MIX
44	1	10	tansig, hadrlims	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	17,87	8,6	-	1000	10	MIX
45	1	10	tansig, poslin	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	17,75	8,6	-	1000	10	MIX
46	1	10	tansig, netinv	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	18,17	9,6	-	1000	10	MIX
47	1	10	tansig, radbas	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	17,52	8,8	-	1000	10	MIX
48	1	10	tansig, radbasn	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	18,17	9	-	1000	10	MIX
49	1	10	tansig, satlin	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	19,443	9,6	-	1000	10	MIX
50	1	10	tansig, satlins	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	18,52	8,9	-	1000	10	MIX

Neste quadro foi mantida a configuração base, a exceção da função de treino durante as 10 configurações.

Alterando as funções de saída podemos concluir que a melhor configuração foi a função de ativação "tansig" para a camada escondida e "softmax" para a camada de saída, teve a maior precisão global de 23,14% e a maior precisão de teste de 11,42%.

As funções de aprendizagem influenciam o desempenho?

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
51	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learncon	10	41,54	12,38	0.05%	1000	10	MIX
52	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnrd	10	36,19	7,62	0.05%	1000	10	MIX
53	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnp	10	41,17	9,81	0.05%	1000	10	MIX
54	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnh	10	43,9	11,81	0.05%	1000	10	MIX
55	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnhd	10	41	10,3	0.05%	1000	10	MIX
56	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnis	10	41	10,01	0.05%	1000	10	MIX
57	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnk	10	39,97	10,76	0.05%	1000	10	MIX
58	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnlv1	10	39,14	8,95	0.05%	1000	10	MIX
59	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnlv2	10	41,77	10,38	0.05%	1000	10	MIX
60	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnos	10	44,47	10,47	0.05%	1000	10	MIX

Utilizando neste quadro 100 neurónios e as funções de ativação que melhor resultado tiveram no quadro anterior, ou seja, “tansig” e “softmax” e variando a função de aprendizagem, podemos concluir que não existe nenhuma configuração que se destaque, contudo, a que obteve um valor mais alto foi a “learnos” com 44,47% de precisão global e 10,47% para a precisão de testes.

Resultados dos tipos de treino com diferentes funções de treino

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
802	1	100	tansig, softmax	trainscg	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	52,88	13	0.05%	1000	50	MIX
801	1	100	tansig, softmax	trainscg	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	89,92	56	0.05%	1000	50	OP
800	1	100	tansig, softmax	trainscg	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	78	36,667	0.05%	1000	50	NUM
803	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	38,04	10	0.05%	1000	50	MIX
804	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	93,35	74,33	0.05%	1000	50	OP
805	1	100	tansig, softmax	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	61,58	20,93	0.05%	1000	50	NUM
806	1	100	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	41,1	9,71	0.05%	1000	50	MIX
807	1	100	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	92,25	73	0.05%	1000	50	OP
808	1	100	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	64,16	21,2	0.05%	1000	50	NUM
802	1	100	tansig, softmax	trainscg	dividerand = {0.7, 0.15, 0.15}	learnngdm	10	52,88	13	0.05%	1000	50	MIX

No quadro acima, o objetivo foi testar as configurações para diferentes tipos de treino, ou seja, aplicar a configuração só a números (NUM), só a operadores (OP) ou a número e operadores (MIX) e perceber se existe alguma diferença entre treinar as imagens separadas.

Após analisado, verificamos que a rede consegue ter uma precisão muito melhor se for treinada de forma separada. Relativamente a uma configuração MIX, se a rede só treinar os números, tem um ganho médio na precisão de aproximadamente 15%. Já nos operadores o incremento é ainda mais e pode chegar até aos 30% a 40% dependendo dos casos. Em suma, uma rede treinada individualmente será sempre melhor.

Combinação dos melhores resultados e testes

Configuração	Número de camadas escondidas	Número de neurónios	Funções de ativação	Função de treino	Divisão dos exemplos	Função Aprendizagem	Nº Repetições?	Precisão Global	Precisão Teste	Taxa de aprendizagem	Epochs	Validation Checks	Tipo de treino
61	2	450, 250	{'tansig', 'tansig', 'softmax'}	trainscg	dividerand = {0,7, 0,15, 0,15}	'learngdm', 'learngdm'	100+	94	72	0,06%	1000	150	MIX
62	1	500	{'tansig', 'purelin'}	traingd	dividerand = {0,7, 0,15, 0,15}	learncon'	1	43,2	16,5	0,05%	2000	300	MIX
63	1	300	{'tansig', 'purelin'}	traingd	dividerand = {0,7, 0,15, 0,15}	learngd'	10	41	9,3	0,05%	2000	300	MIX
64	1	300	{'tansig', 'softmax'}	traingd	dividerand = {0,7, 0,15, 0,15}	learngd'	10	39,91	10,91	0,05%	2000	300	MIX
65	3	650 200 100	tansig, tansig, tansig, softmax	traingd	dividerand = {0,7, 0,15, 0,15}	'learngdm', 'learngdm', 'learngdm'	10	85,35	47,22	0,05%	2000	300	MIX
66	3	650 200 100	tansig, tansig, tansig, softmax	traingd	dividerand = {0,7, 0,15, 0,15}	'learngdm', 'learngdm', 'learngdm'	10	89	59,12	0,05%	4000	300	MIX
67	2	450, 50	{'tansig', 'hardlim', 'softmax'}	trainscg	dividerand = {0,7, 0,15, 0,15}	'learngdm', 'learngdm'	10	93	63	0,05%	2500	150	MIX
68	2	450, 50	{'tansig', 'tansig', 'softmax'}	trainscg	dividerand = {0,7, 0,15, 0,15}	'learngdm', 'learngdm'	10	90,95	61	0,05%	2000	150	MIX
69	2	400, 400	{'tansig', 'tansig', 'softmax'}	trainscg	dividerand = {0,7, 0,15, 0,15}	'learngdm', 'learngdm'	200+	95,39	77	0,07%	2000	150	MIX
809	1	100	tansig, softmax	traingd	dividerand = {0,7, 0,15, 0,15}	learngdm'	200+	96,5	86,66	0,05%	1000	50	OP
810	1	400	tansig, softmax	traingd	dividerand = {0,7, 0,15, 0,15}	learngdm'	30	24,36	23,89	0,05%	1000	50	OP
811	1	100	tansig, softmax	traingd	dividerand = {0,7, 0,15, 0,15}	learngdm'	200+	97	90	0,05%	1000	50	OP
812	1	75	tansig, softmax	traingd	dividerand = {0,7, 0,15, 0,15}	learngdm'	60+	80	36	0,05%	1000	50	NUM
813	2	100, 100	tansig, tansig, softmax	traingd	dividerand = {0,7, 0,15, 0,15}	learngdm', 'learngdm'	100+	90,2	70,66	0,05%	1000	50	NUM
814	5	100, 100, 75, 75, 75	tansig;'tansig'; tansig;'tansig'; tansig', 'softmax'	trainscg	dividerand = {0,7, 0,15, 0,15}	learngdm', 'learngdm', 'learngdm', 'learngdm', 'learngdm'	500+	96	88	0,05%	1000	50	NUM

815	5	100, 100, 75, 75, 75	tansig','tansig'; tansig','tansig'; tansig'; 'softmax'	trainscg	dividerand = {0.7, 0.15, 0.15}	learngdm'; 'learngdm','lea rngdm','learn gdm','learngdm'	200+	96,42	83,81	0.05%	1000	50	MIX
-----	---	----------------------	-----------------------------------------------------------------	----------	-----------------------------------	-------------------------------------------------------------------	------	--------------	--------------	-------	------	----	------------

Este quadro é a combinação dos melhores testes que executamos nos quadros anteriores justamente com algumas configurações soltas em que verificamos que tinham potencial para uma boa precisão global.

Os principais aspetos que verificamos que melhora a precisão são o número de neurónios, a função de saída “softmax” e as funções de treino “trainscg” e “traingd”. Partindo deste facto criamos várias configurações em torno deste princípio aplicadas a cada tipo de treino tendo conseguido os seguintes resultados por diferente tipo de operador:

- Configuração para **Operadores (OP)**: 97% Precisão Global e 90% de Precisão de teste;
- Configuração para **Números (NUM)**: 96% Precisão Global e 88% de Precisão de teste;
- Configuração para **Operadores e Números (MIX)**: 96,42% Precisão Global e 83,81% de Precisão de teste;

Análise e conclusões obtidas

De modo geral, podemos concluir que não existe uma configuração genérica que possa ser usada para todas as situações, visto que, o que resulta para um exemplo pode não resultar para outro, dado que depende muito das imagens que estão a ser tratadas e o contexto das mesmas, basta verificar que com as mesmas configurações aplicadas a tipo de treino diferente dão resultados diferentes.







Os padrões encontrados em que se verifica que realmente existe um incremento de precisão são o número de neurónios, a função de saída “softmax” e as funções de treino “trainscg” e “traingd”. Esta configuração aliada ao aumento das epochs e ajustes nas funções de aprendizagem e ajustes na sua taxa, conferem um aumento significativo na precisão global e de testes.

O principal problema que nos deparamos foi o overfitting, ou seja, o modelo ajusta-se demasiado bem ao treino. Conseguimos identificar a existência de uma grande diferença entre as precisões de treino e teste.

C) Dataset manual

Criação do dataset

Criamos um pequeno dataset contendo desenhos de dígitos e símbolos feitos manualmente com o objetivo de apresentarem semelhanças com os exemplos usados no treino da rede neural. O dataset foi elaborado com cuidado, garantindo que houvesse pelo menos 3 imagens para cada dígito e operador, tendo desta forma uma diversidade de amostras que permitiu uma avaliação mais abrangente do desempenho da rede neural na classificação dos caracteres.

Exemplo 1 (1)	Exemplo 2 (div)	Exemplo 3 (3)
		
Exemplo 4 (8)	Exemplo 5 (2)	Exemplo 6 (mul)
		

Resultados de classificações

Executamos uma série de testes através de um script projetado especificamente para avaliar a eficácia dos modelos que desenvolvemos.

Este procedimento permitiu uma avaliação mais abrangente e rigorosa, uma vez que empregamos dados que se assemelham mais de perto àqueles encontrados em cenários do mundo real.

O nosso principal objetivo era compreender com precisão a taxa de acerto dos modelos, fornecendo uma visão mais clara do desempenho quando expostos a novos dados.

Dado isto, decidimos criar um novo dataset denominado 'custom', e obtivemos os seguintes resultados dos seguintes datasets:

Resultados do melhor modelo de números

Modelo	Tipo do Modelo	Taxa de acerto	Dataset
model_814_96_88_NUM.mat	NUM	36%	custom
model_814_96_88_NUM.mat	NUM	94%	start
model_814_96_88_NUM.mat	NUM	96%	trainl

Resultados do melhor modelo de operadores

Modelo	Tipo do Modelo	Taxa de acerto	Dataset
model_810_97_90_OP.mat	OP	40%	custom
model_810_97_90_OP.mat	OP	70%	start
model_810_97_90_OP.mat	OP	97%	trainl

Resultados do melhor modelo mix

Modelo	Tipo do Modelo	Taxa de acerto	Dataset
model_815_96_83_MIX.mat	MIX	41,4286%	custom
model_815_96_83_MIX.mat	MIX	90%	start
model_815_96_83_MIX.mat	MIX	96,4286%	trainl

Análise e conclusões relevantes




Os resultados obtidos fornecem informações valiosas sobre a capacidade da(s) rede(s) neural(is) em generalizar e reconhecer desenhos de dígitos e símbolos que apresentam semelhanças com os exemplos utilizados no treino. Estas conclusões são fundamentais para aprimorar e otimizar a aplicação da rede neural em casos reais, onde a classificação de desenhos manuais pode desempenhar um papel importante.

Dado que o modelo foi treinado com os dados presentes apenas da pasta train1, podemos comprovar que com imagens aproximadas o modelo mostra-se bastante efetivo, tanto é, que as taxas de acerto na pasta start mostram-se bastante ambiciosas com resultados 94%, 70% e 90%.

Por outro lado, o nosso dataset customizado com 5 imagens de cada dígito, mostrou taxas de acerto muito baixas ao que era expectável, tendo os valores 36%, 40% e 41.42%.

Esta taxa bastante baixa pode ser derivada da baixa quantidade de imagens utilizadas como treino para os nossos modelos, comprovando a importância das quantidades de dados utilizadas para treinar os modelos.

Observe a diferença dos dígitos presentes nas três pastas:

custom	start	train1
		

Com estes resultados concluímos que a importância de um dataset grande ajuda bastante na detecção de todos os elementos.

D) Aplicação gráfica

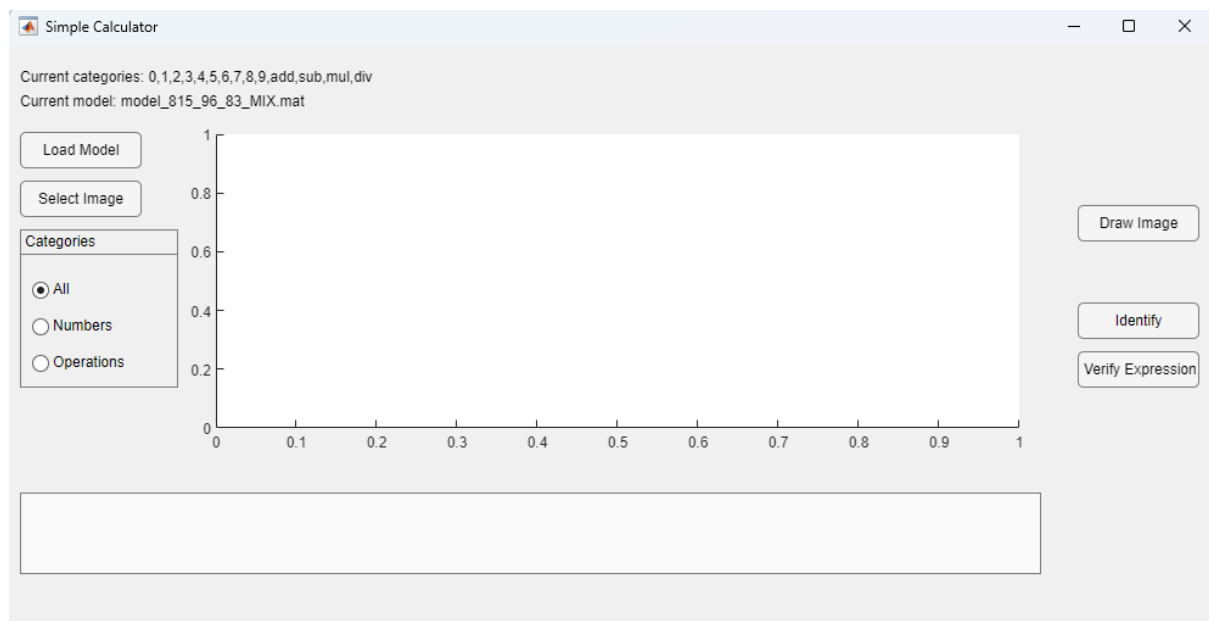
Apresentação da interface

Para a execução e testes dos nossos modelos, foi criada uma interface que permitisse realizar estas respectivas operações.

Esta interface permite carregar novos modelos, selecionar imagens, e efetuar testes sobre os diferentes modelos dado que existem modelos apenas de números, outros de operações e os modelos MIX.

Nesta interface é nos apresentado por defeito todas as categorias, isto significa que o modelo carregado ou o que deverá ser utilizado é o MIX.

Observe a seguinte imagem em baixo:



Caso fosse necessário apenas testar um determinado modelo de números ou operadores deve ser selecionado o respectivo tipo de categoria necessário, caso contrário irá fornecer a informação errada do que o modelo está realmente a identificar.

Identificação de um número/operador

Para identificar um único número ou operador presente na imagem, o utilizador precisa selecionar o tipo de modelo desejado, escolher a imagem e, de seguida, clicar no botão 'Identificar'.

Ao acionar este botão, a imagem será convertida para preto e branco caso, caso ainda não esteja, e redimensionada para o tamanho 25x25. Após isto, a imagem é convertida para uma matriz coluna dado que é o tipo de dados de entrada que os nossos modelos foram treinados.

Em baixo podemos observar a identificação de um número:

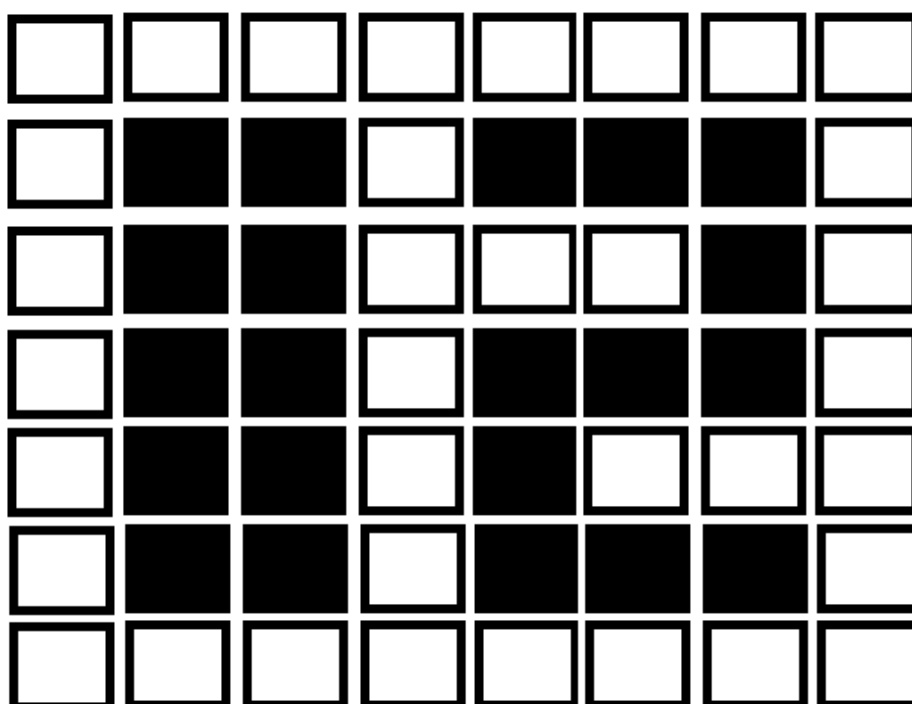


Deteção das componentes da expressão

Para identificar elementos individuais numa expressão, quer sejam números ou operadores, tivemos que adaptar a nossa abordagem. Isto deve-se ao facto de as nossas redes neuronais esperarem receber apenas um caractere de cada vez.

Assim, criámos um novo botão, denominado 'Verify Expression'. Este botão analisa a imagem carregada pelo utilizador, detecta números e operadores, divide a imagem em partes individuais, redimensiona cada parte para 25x25 pixels e, finalmente, converte cada parte numa matriz binária e numa matriz de colunas.

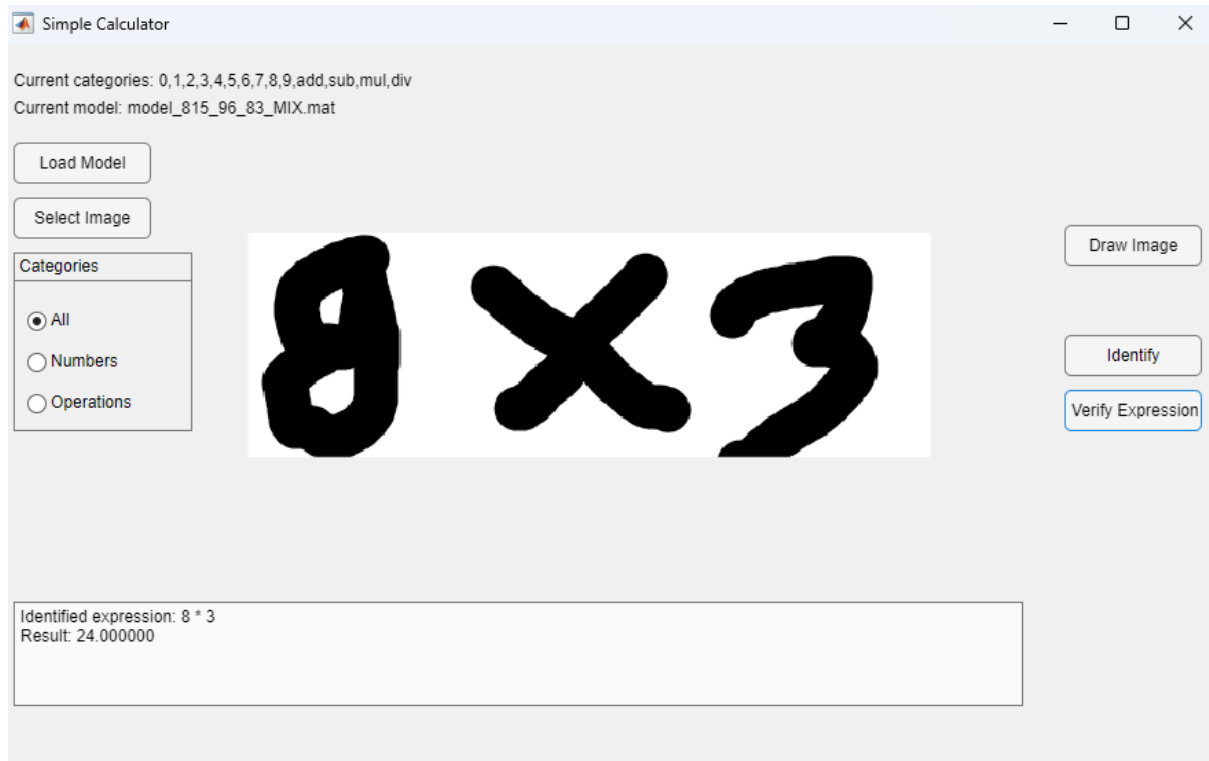
A detecção de números e operadores baseia-se nas cores presentes nas colunas, uma vez que as imagens são sempre a preto e branco. Observa a imagem abaixo para uma melhor compreensão:



Como podemos observar, na primeira coluna, todos os 'bits' da imagem são brancos, o que significa que nenhum número ou operador foi detectado. No entanto, na segunda coluna, detectamos um bit preto, o que marca o início da detecção do primeiro dígito. Na quarta coluna, todos os bits são brancos, indicando o fim da detecção daquela imagem, já que a imagem termina na terceira coluna.

Com esta abordagem, conseguimos detectar facilmente os dígitos presentes na imagem e redimensioná-los para o tamanho 25x25, onde de seguida são utilizados como entrada para o modelo que está carregado na interface.

Observe abaixo uma imagem, onde através de uma expressão matemática, conseguimos identificar corretamente os elementos presentes e o resultado da operação.

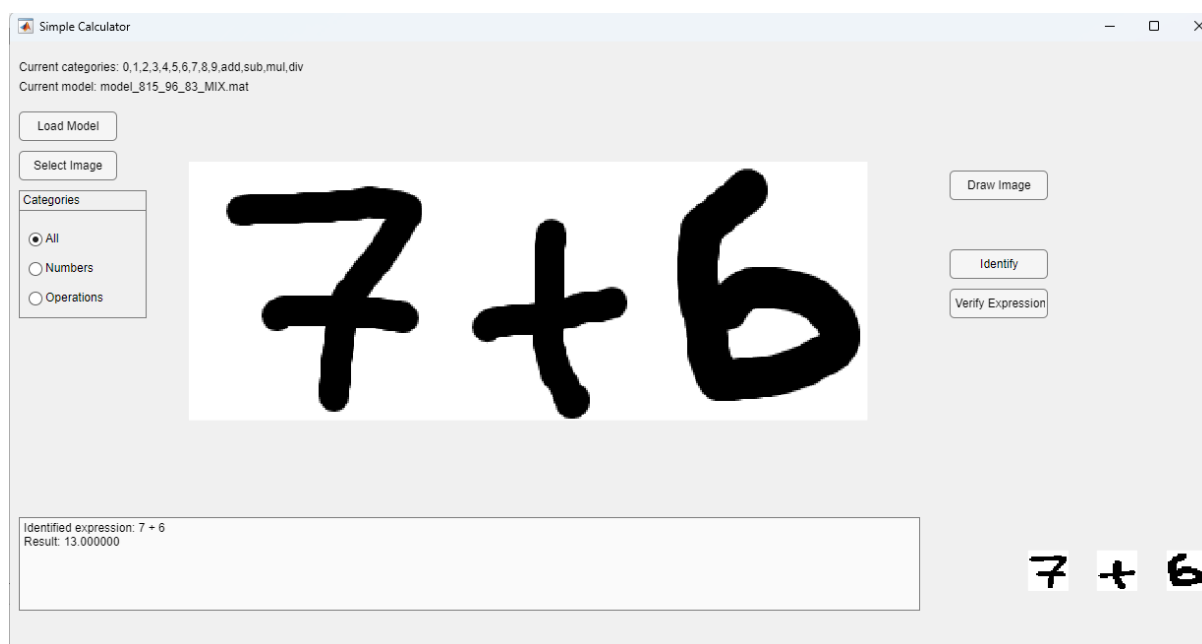


Resultados obtidos

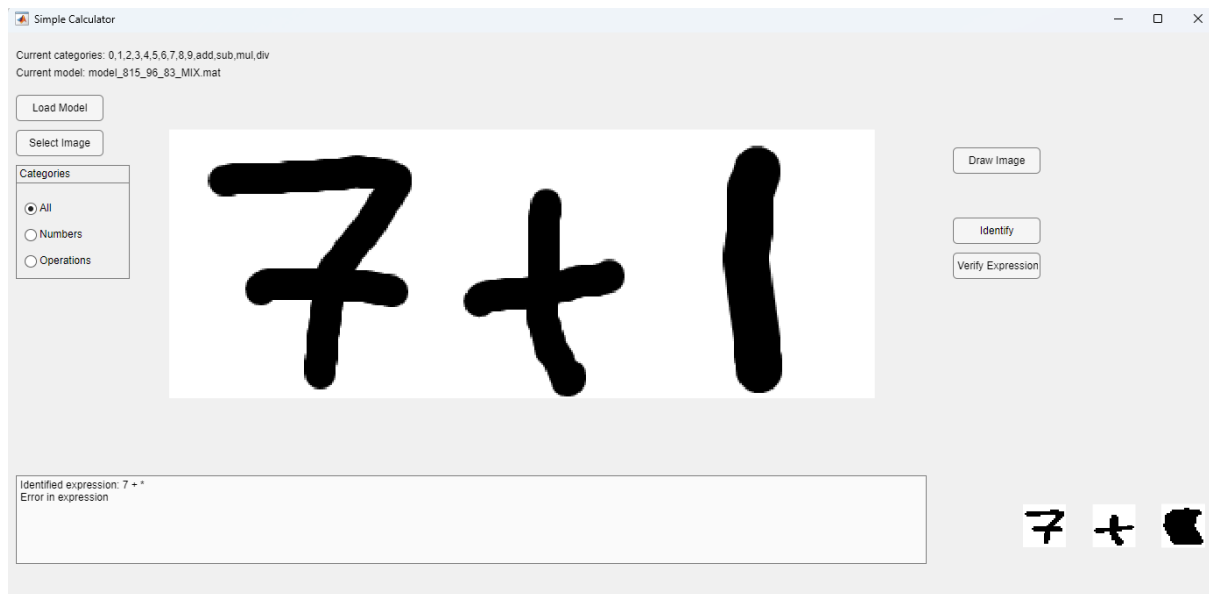
Os resultados individuais dado que é através apenas de uma imagem foram os expectáveis apresentando taxas de acerto entre 70-90%, por outro lado, a identificação dos elementos na expressão e o envio para o modelo apresentou taxas de acerto muito mais baixas. Acredita-se que esta situação se deva ao facto da forma como os dados são enviados para o modelo.

Um dos casos que reparamos que se revela problemático com a abordagem que seguimos foi a detecção de imagem com a presença do número 1. Como apenas obtemos o dígito a partir do momento que detectamos os pretos, com o resize do número o que acontece é que acaba por enviar para o modelo quase um quadrado todo preto.

Observe como as imagens são divididas dadas as detecções de imagem:



Como podemos observar nas imagens em pequeno, presentes do canto direito inferior na imagem, a nossa abordagem prova-se eficaz na detecção dos dígitos da expressão, mas agora observe o problema que observamos com a situação do número 1:



Note que os redimensionamos da imagem do número 1 para 25x25 são problemáticos, dado que a rede neuronal, não sabe se é o número 0,1, ou até mesmo o operador de multiplicação. Isto comprova que os dados enviados para o modelo e os dados de treino são muito importantes quando estamos a treinar uma rede neuronal.

Outra situação implementada foi a necessidade de em vez de fazer o resize para 25x25, acabamos por o fazer para 23x23 e adicionar uma borda branca de 2x2 em cada imagem, isto provou ajudar bastante na identificação dos elementos presentes na expressão aproximando-se mais dos dados de treino utilizados para a criação do modelo, mas mesmo assim, no caso do número 1 o problema manteve-se.

Conclusão

Dado agora finalizado o trabalho prático da cadeira de Conhecimento e Raciocínio, em que desenvolvemos uma ferramenta capaz de identificar os números e operadores presentes em expressões matemáticas, calcular e apresentar estes resultados, consideramos que fomos capazes de realizar o trabalho com sucesso dado que este era o objetivo principal.

Consideramos que este projeto foi desafiador, porém gratificante, dado que nos permitiu mergulhar em vários aspetos cruciais do treino de redes neurais e reconhecimento de padrões.

Aprendemos a importância do pré-processamento de dados, dado que é uma etapa fundamental para preparar as imagens para um treino eficaz.

Conseguimos apreciar também a importância dos ajustes de parâmetros das redes neuronais, incluindo a seleção do número de neurónios, a escolha da função de treino, etc..

Reconhecemos que a configuração do modelo não é uma ciência exata, mas sim um equilíbrio delicado entre a teoria e a prática, muitas vezes alcançado por meio de experimentação e ajuste iterativo.

Com a implementação da “calculadora”, fomos capazes de ver a aplicação da prática da teoria, ao transformar o conhecimento abstrato numa aplicação tangível e útil, onde apreciamos o valor e a relevância dos conceitos e técnicas aprendidos ao longo da disciplina.

Em conclusão, as competências e o conhecimento adquiridos durante as aulas e no desenvolvimento do trabalho, serão de imenso valor no nosso futuro académico e profissional.