Classificar Formas Geométricas, uma análise de configurações e desempenho.

“Redes Neuronais – Feedforward”

Conhecimento e Raciocínio 2024-2025

* António Pedroso 2021132042
* Bruno Correia 2015015880

# Objetivo de Estudo Proposto

* Foi-nos proposto, no âmbito da unidade curricular de Conhecimento e Raciocínio do ano letivo 2024/2025, o desenvolvimento de redes neuronais feedforward – da configuração dos seus parâmetros e hiperparâmetros e análise do seu desempenho na classificação de um dataset de imagens de figuras geométricas variadas.
* Neste relatório vão ser descritos os passos e justificadas as opções tomadas na resolução das alíneas do enunciado, destacando as caraterísticas principais de cada rede e implicações das mesmas nos resultados obtidos.
* Todo o código descrito e explicado no relatório e trabalho prático foi desenvolvido utilizando o ambiente de programação MATLAB (2024).

## Dataset:

* Os dados usados para treinar, validar e testar o desempenho das redes desenvolvidas, além dos poucos criados através de desenho digital, são provenientes da pasta fornecida e são compostos por 6 classes diferentes relativas a figuras geométricas descritas abaixo:
  1. Classe ***‘circle’ –*** figura geométrica representativa de um círculo;
  2. Classe***‘kite’ -*** figura geométrica representativa de um quadrilátero;
  3. Classe ***‘parallelogram’ -*** figura geométrica representativa de um paralelograma;
  4. Classe ***‘square’ -*** figura geométrica representativa de um quadrado;
  5. Classe***‘trapezoid’ -*** figura geométrica representativa de um trapézio

As figuras estão proporcionalmente distribuídas por pastas com o nome da classe que as identifica, dentro de outras 3 pastas ***start****,* ***test*** *e* ***train*** com respetivamente 5, 10 e 50 exemplos diferentes de cada, que perfazem um *dataset* total de 390 imagens (224\*224px).

# Descrição das Metodologias Implementadas

## 1. Pré-processamento dos Dados

* As imagens do dataset foram **transformadas** com o intuito de reduzir o tempo computacional significativamente, sem perder as principais caraterísticas que as distinguem.
* Foi aplicada a função **rbg2gray** para reduzir 3 vezes a dimensão dos dados, de RBG para escalas de cinza, na condição if size(img, 3) == 3, que apenas executa a função caso a imagem tenha 3 canais (RGB), visto que as figuras estão a preto e branco.
* Foi aplicada outra função de transformação – **imresize**, redimensionando novamente as imagens, desta vez para [28 x 28] pixéis.
* Finalmente aplicou-se a função **imbinarize**, convertendo os valores de intensidade (0-255) em 0s e 1s – focando no contraste entre fundo e objeto.

## 2. Estruturação dos Dados

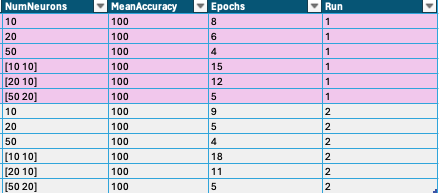
* Estrutura de dados **shapes**: é um array de strings que contém a lista com os nomes das classes, usada no pré-processamento de dados;
* Estrutura de dados **labels**: array que armazena os rótulos de todas as imagens, indicando a classe a que cada uma pertence – é o output esperado (alvo) durante o treino da rede;
* Estrutura de dados **binaryImages**: matriz que armazena todas as imagens como vetores de 784 elementos binários (28x28) onde cada coluna representa uma imagem e cada linha um pixel binarizado.

## 3. Processo de Treino

De seguida descrevem-se os processos de treino das diferentes redes, associados à alínea do enunciado e explicados detalhadamente:

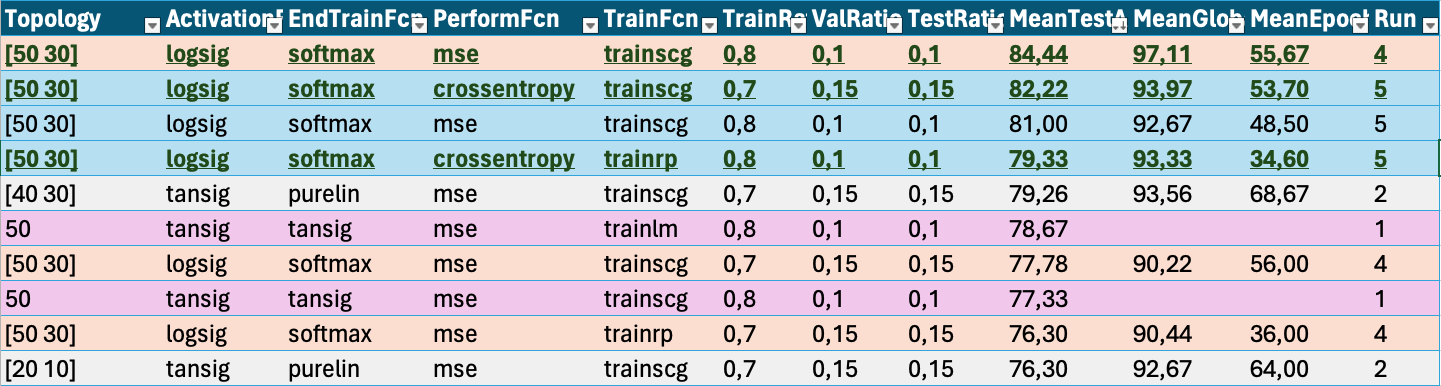
1. ***net = feedforwardnet(x);*** (x: nº neurónios/camada oculta)Uma rede *feedforward*, ou *multilayer perceptron* (**MLP**), é um tipo de rede neuronal artificial composta por neurónios organizados em camadas (entrada: 784, oculta(s): x, saída: 6) onde há um ajuste dos vetores de pesos, **w,** durante o treino para minimizar uma função de perda que calcula a diferença entre previsões (output da rede) e alvos (labels) – sendo usada para problemas de regressão e de **classificação**.   
   Primeiramente treinámos a rede com a totalidade das amostras da pasta *start* até chegar a uma taxa de acurácia de 100% - utilizámos os parâmetros **default** do MATLAB para redes neuronais, fazendo variar unicamente o número de camadas ocultas e o seu número de neurónios associados.  
    Parâmetros por *default* em redes neuronais no MATLAB:
   1. **tansig:** tangente sigmoide – comprime os valores entre [-1, 1] introduzindo não linearidade à rede, na camada oculta.  
      **f ’(x) = 1 − f(x)2**
   2. **purelin:** produz uma saída linear contínua, sem transformação.  
      **f(x) = x , f ′(x)=1**
   3. **trainlm:** combina Método de Gauss-Newton (para convergência rápida) e Gradiente Descendente (para estabilidade, quando Gauss-Newton falha), ajustando iterativamente os pesos dos neurónios para minimizar a função de perda, produzindo um valor não negativo que diminui com a precisão da rede.  
      ***Δw = −(J^T.J + μI)^(−1).JT^(e) (***Δ*w: variação dos pesos)*
   4. **mse**: função de perda que calcula o erro quadrático médio entre os **outputs** e os **alvos** da rede, onde **N** é o número de amostras – amplamente usada em regressão e, por default, no MATLAB para redes neuronais. ***MSE = 1/N∑ N (i=1)​ (ti−yi ​ )^2***
   5. **train split:** divisão das amostras entre treino, validação e teste.  
      **net.divideFcn: ‘dividerand’:**   
      **net.divideParam.trainRatio:** 1.00 (70% para treino p/default), **net.divideParam.valRatio:** 0.00 (15% para validação p/default), **net.divideParam.testRatio:** 0.00 (15% para teste p/default).
   6. **goal:** 0 (erro alvo mínimo p/default) – parâmetro de convergência: o treino continua a correr até que a função de perda atinja 0 (o que na prática acontece raramente, sendo frequentemente ajustado para valores positivos pertos de 0).

Com estes valores padrão do MATLAB para redes neuronais e no contexto da alínea a) do enunciado, fomos treinar as nossas redes com a **totalidade das amostras** da pasta *train,* obtendo precisões globais de 100% para todas as topologias testadas, variando apenas o número de Epochs que a rede demorou a chegar ao *goal* definido – resultado espectável uma vez que se usaram todos os exemplos no treino (a rede não classificou nada que não tivesse visto antes).

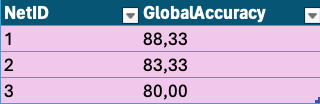
 Desempenhos das redes na alínea a):

**(...)**

1. Nesta alínea do enunciado expandimos o nosso conhecimento de redes neuronais, explorando uma série novos de parâmetros e hiperparâmetros que iremos explicar seguidamente:
   1. **poslin:** mais conhecida com ReLU (rectified linear unit) é uma função de ativação linear que preserva valores positivos, colocando os negativos a 0.  
      ***f(x) = max(0, x)***
   2. **logsig:** sigmoide logística – função de ativação aplicada nas camadas ocultas - comprime a entrada para o intervalo [0, 1] introduzindo uma não-linearidade suave (pouco eficaz em redes profundas).  
      ***f(x) = 1 / (1+ e^(-x))***
   3. **hardlim:** função de ativação binária. Backpropagation requer derivadas da função de ativação para ajustar os pesos – a derivada de hardlim é 0 em todos os pontos o que impossibilita o treino da rede.  
      ***f(x) = 1 se x >= 0, f(x) = 0 se x < 0***
   4. **purelin:** tentativa de usar como função de ativação – passa a entrada diretamente como saída sem transformação, não introduz não-linearidade o que nos fez obter resultados muito baixos semelhantes a iii.  
      **f(x) = x**
   5. **softmax:** converte os valores da camada final em probabilidades normalizadas que somam 1, ideal para classificação multiclasse – permitiu-nos obter valores altos de acurácia usando esta função de saída.  
      ***f(xi) = e^x \*i / ∑K(j=1) e^x \*j*** , K = nº classes
   6. **crossentropy:** a função de perda entropia cruzada mede a divergência entre a distribuição de probabilidades prevista pela rede e os alvos reais – otimizada para classificação multiclasse e emparelha bem com **softmax.  
      *CE = − 1/N ​\*∑N i=1 ∑K c=1 t\*i,c ​\*log(y\*i,c).*** , K = nº classes
   7. **trainscg:**(Scaled Conjugate Gradient) função de treino que utiliza gradientes conjugados para ajustar os pesos de forma eficiente, sendo **mais leve** que **trainlm** é ideal para datasets médios, como no nosso caso (390 amostras) com convergências rápidas que permitem efetuar as iterações em menos tempo com resultados satisfatórios.  
       wk+1 = wk + α\*k\*p\*k , p\*k é a direção de busca, a\*k é o passo ajustado
   8. **trainrp**:(Resilient Backpropagation) ajusta os pesos com base no sinal do gradiente, ignorando a magnitude – o que o torna robusto a ruídos e eficiente em redes com menos camadas, mas pode ser lento em convergência. ***Δwij = −sign( ∂E / ∂wij ) \* Δwij*** , onde Δwij é o passo adaptativo
   9. **trains:**(Sequential Order Incremental Training) treina a rede numa ordem sequencial, sendo útil na adaptação a padrões locais mas pode exigir mais iterações. ***wk+1 = wk −*** ***η ( ∂E / ∂ wk )*** , onde η é a taxa de aprendizagem
   10. **traingd:**(Gradient Descent) ajusta os pesos com base no gradiente descendente simples, sendo o método mais básico utilizado – lento.  
       ***wk+1 = wk −*** ***η ( ∂E / ∂ wk )***

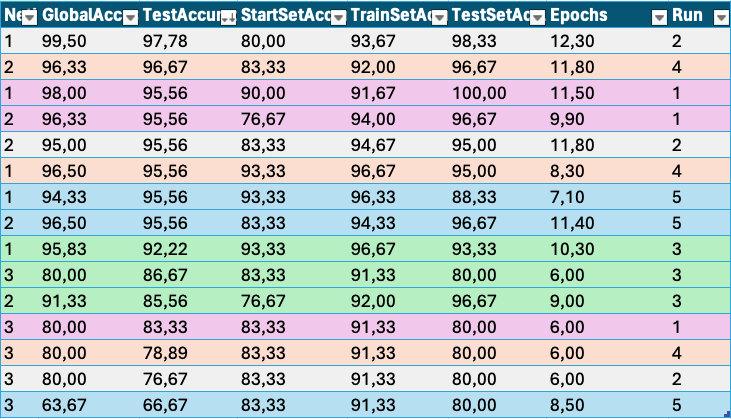
Desempenhos das melhores redes na alínea b):

Não é de espantar que as melhores redes treinadas têm todas a mesma topologia **[50 30]** pois 2 camadas ocultas representam um bom equilíbrio entre capacidade de generalização e redução de overfitting, assim como o número de neurónios que é suficiente para a complexidade das amostras testadas. Outra caraterística que as une é a aplicação da **sigmoide logística** nas camadas ocultas que lhes confere uma não-linearidade suave que é ideal para redes pouco profundas. Confirmou-se também a hipótese acima referida de que a função de saída **softmax** iria emparelhar bem com a função de perda **crossentropy** devido ao facto de que ambas são indicadas para problemas de classificação multiclasse (6) onde a softmax gera probabilidades normalizadas e a crossentropy penaliza as discrepâncias entre essas previsões e os outputs alvo. Outra especificidade que une as nossas 3 melhores redes é a função de treino **trainscg** que é ideal para topologias médias, como [50 30], acelerando a convergência tendo em consideração as melhores direções de busca por pesos que conferem melhores desempenhos na rede.Para efeitos de diversidade nas redes e para conseguir extrair conclusões mais ricas, optámos por escolher a quarta melhor, em que foi aplicada a função de treino **trainrp –** apesar de nos ter conferido uma performance inferior de teste, devido ao facto de que utiliza somente o sinal do gradiente para atualizar os pesos, consideramos que é satisfatória em comparação com as centenas de outras redes testadas. Algo inesperado e talvez inexplicável é o facto da combinação softmax + **mse** ter sido a nossa melhor rede, já que teoricamente o mse é mais adequado para regressão pois calcula o erro quadrático médio entre previsões e alvos, minimizando diferenças numéricas. Talvez a binarização das imagens possa ter reduzido a variabilidade dos erros e tornado o mse relativamente melhor, e o trainscg pode ter ajustado os pesos de forma a compensar essa limitação.

1. Resultados dos testes das 3 melhores redes (treinadas somente com imagens da pasta **train**) com as amostras da pasta **test:**

​

Verifica-se que as redes estão com uma boa capacidade de generalização, mas ainda há bastante margem para melhorias. Resultados satisfatórios mas espera-se ainda um aumento de performance.

​ Desempenho das redes após treino com os dados da pasta **test:**

## 

## Verifica-se o espectável: as redes aprenderam com os treinos de um conjunto maior de imagens, o que lhes vai conferir uma capacidade ainda maior de generalização.

## Desempenho das redes após treino com a totalidade do dataset (pastas **train, start** e **test,** 390 amostras)**:**

Todas as redes aumentaram no seu desempenho – novamente solidifica a hipótese de que aumentar o dataset de treino vai resultar num aumento proporcional na capacidade de classificação de imagens de figuras geométricas como é o nosso caso. Algumas das redes inclusivamente atingiram uma precisão global de 100% o que, contrariamente ao que possa parecer, é algo indesejado pois reflete um **overfitting** aos dados onde as redes estão a classificar no teste amostras que já viram no treino.

## Resultados da performance das redes na classificação de imagens desenhadas (custom images):

## Verifica-se uma redução muito significativa na acurácia das redes – isto deve-se principalmente a dois fatores:

## a qualidade dos desenhos – as redes tentaram classificar imagens não só nunca antes vistas mas ligeiramente diferentes do que estão treinadas para identificar;

1. confirma-se o **overfitting** aos dados do treino – as redes estão com uma baixa capacidade de generalização.

## 5. Matrizes de Confusão

## 

## Podemos observar as matrizes de confusão relativas às redes 1 e 3 após o treino com a pasta de teste (alínea c) ii.) e treino com a totalidade do dataset (alínea c) iii.) – comprova-se que a capacidade das redes classificarem os exemplos das pastas vai aumentando em par com o número de amostras treinadas.

## 6. Classes com melhor Predição

Das centenas de redes que foram desenvolvidas ao longo deste trabalho prático, consegue-se depreender que as classes **circle, triangle,** **square** e **kite** são as que somam predições verdadeiras positivas em maior quantidade – são as mais fáceis de identificar. Por outro lado as classes **parallelogram** e **trapezoid** foram as que atingiram valores mais baixos (cerca de 96% nas matrizes de confusão finais relativas às melhores redes) por terem caraterísticas mais complexas.

# Conclusões

O desenvolvimento deste trabalho prático permitiu-nos testar os nossos conhecimentos teóricos adquiridos ao longo do semestre. Conseguimos progredir com as melhores redes ao longo do mesmo e as hipóteses inicialmente lançadas foram-se verificando, pelo menos na sua maioria. É com satisfação que concluímos este relatório pois é o reflexo do trabalho árduo não apenas de escrita de código, mas de tratamento e análise de dados.

# Bibliografia

Forums MATLAB;

Duckduckgo;

Moodle – Ficheiros de apoio às aulas.