

# **Exercício Programa 02**

## **ANNs e Computer Vision**

---

Disciplina: PMR3508 (2024)

Monitor: Enzo Bustos

E-mail: [monitoria.pmr3508.2024@gmail.com](mailto:monitoria.pmr3508.2024@gmail.com)

# Objetivos do EP02

**01**

## Introdução CV

Conceitos gerais sobre Computacional e suas tarefas.

**02**

## MNIST

Primeira interação com o MNIST Dataset, contendo dígitos escritos à mão.

**03**

## Redes Neurais

Aplicação das ANNs no contexto do EP, primeiros passos do Pré-Processamento.

**04**

## Exercícios e Critérios

Esperado dos alunos nesse EP e nessa submissão do Kaggle.

# 01

# Visão

# Computacional

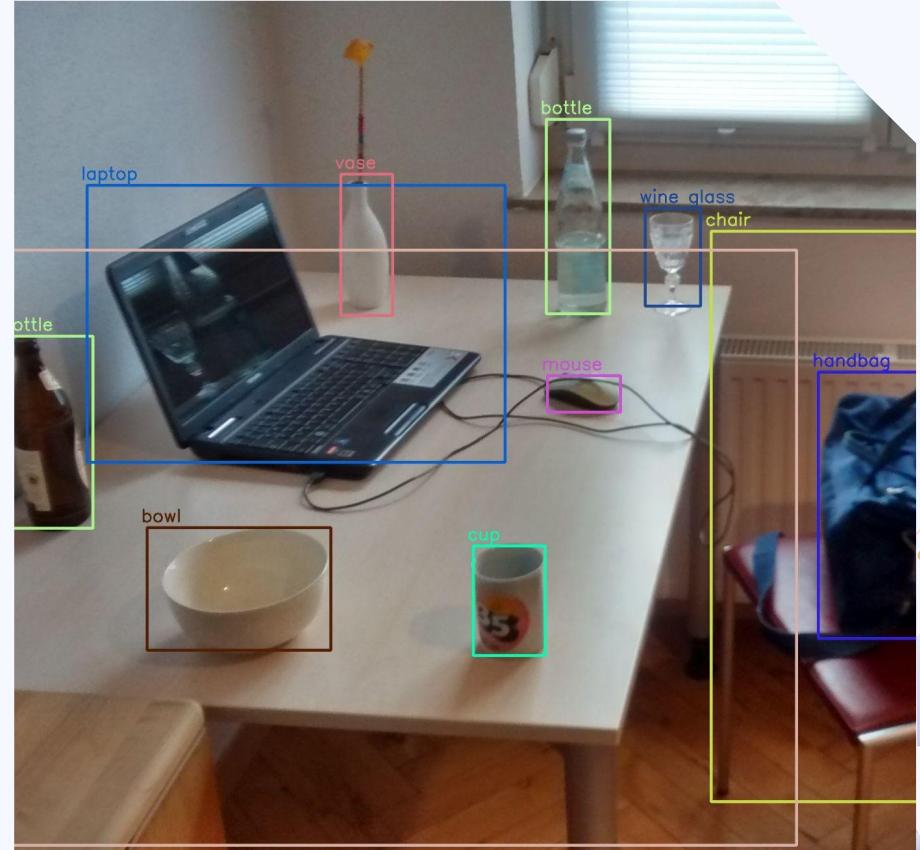


# Introdução a CV

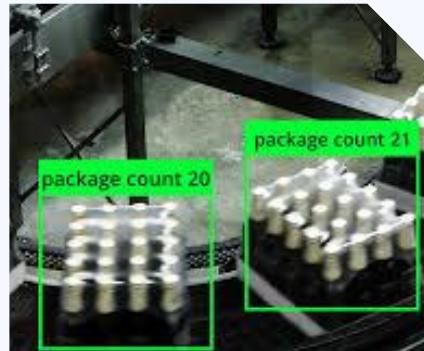
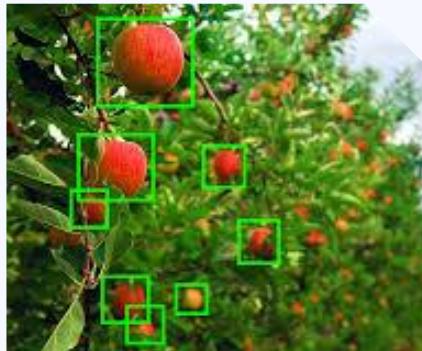
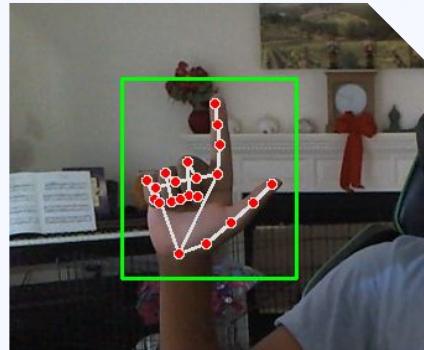
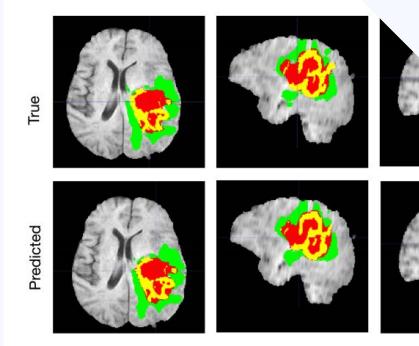
CV é um campo da IA que capacita máquinas a interpretar o mundo visualmente, por imagens ou vídeos.

Usando uma série de algoritmos para processar e analisar esses dados e extrair informações para tarefas específicas.

Ela combina conceitos de vistos como ANNs e Deep Learning para reconhecer padrões, identificar objetos e tomar decisões baseadas em dados visuais.



# Exemplos de Aplicação



# Principais tarefas de CV



## Classificação

Identifica a classe de uma imagem.



## Localização

Indica a posição de um objeto.



## Detecção

Encontra e classifica vários objetos.



## Segmentação

Classifica os objetos em cada pixel.

### Classification



CAT

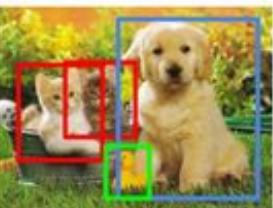
### Classification + Localization



CAT

Single object

### Object Detection



CAT, DOG, DUCK

### Instance Segmentation



CAT, DOG, DUCK

Multiple objects

02

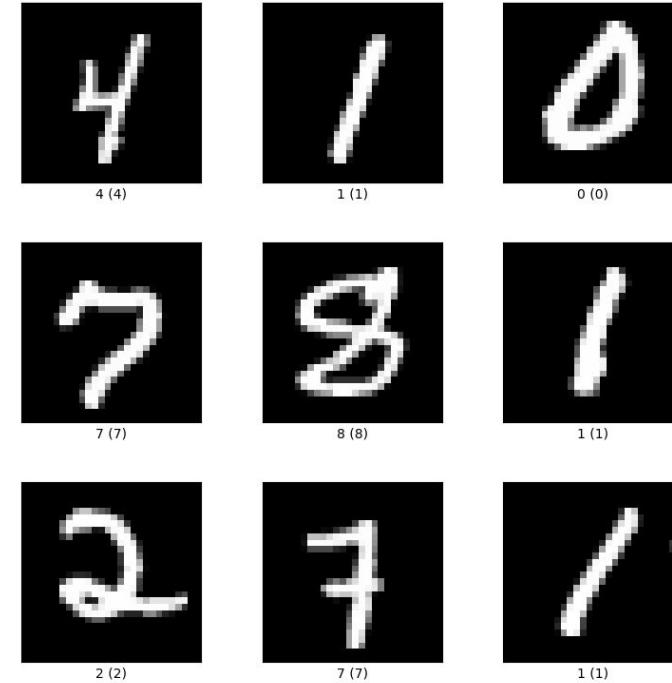
# Dataset MNIST

# História do Dataset

- O MNIST é um dos *datasets* mais utilizados no estudo de aprendizado de máquina e visão computacional.
- Foi criado a partir do *dataset* original do NIST, que continha dígitos manuscritos usados para reconhecimento de escrita
- Yann LeCun adaptou o MNIST para facilitar o treinamento de ANNs, com foco na classificação de dígitos.
- Por ser pequeno e relativamente simples, o MNIST tornou-se popular para testar algoritmos e novas arquiteturas de redes neurais.

# Sobre o Dataset

- 70k imagens de dígitos
  - ◆ 60k Treino
  - ◆ 10k de Teste
- Imagens de 28px×28px
- Rótulos de 0 a 9; 10 classes
- Vetor com 784 posições



# Sobre o Dataset

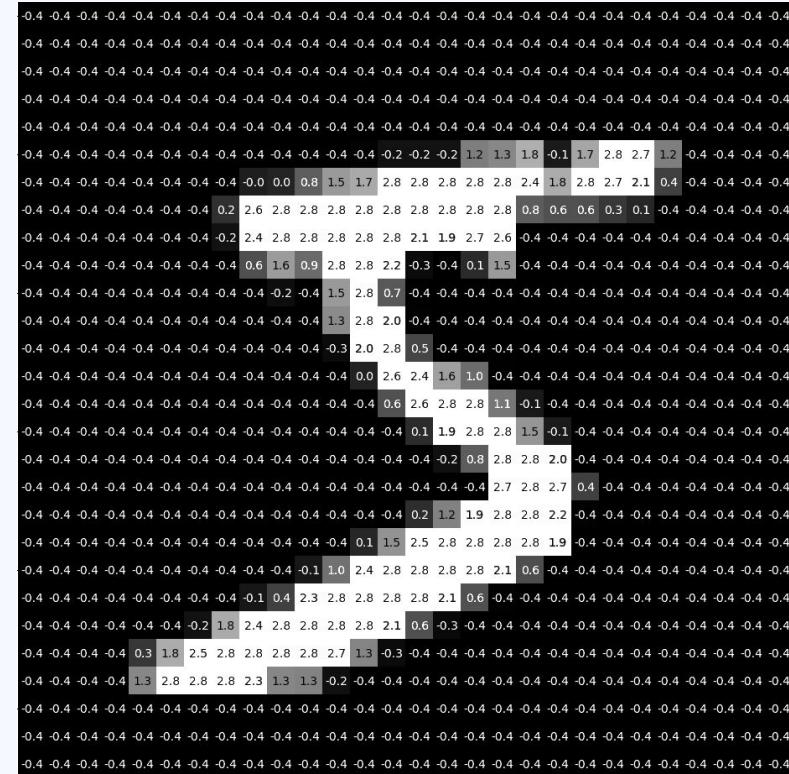
## Sobre as Imagens

As imagens são em escala de cinza e têm dimensões de  $28 \times 28$  pixels.

Cada pixel possui um valor entre 0 (preto) e 255 (branco), com tons de cinza intermediários.

28px

28px

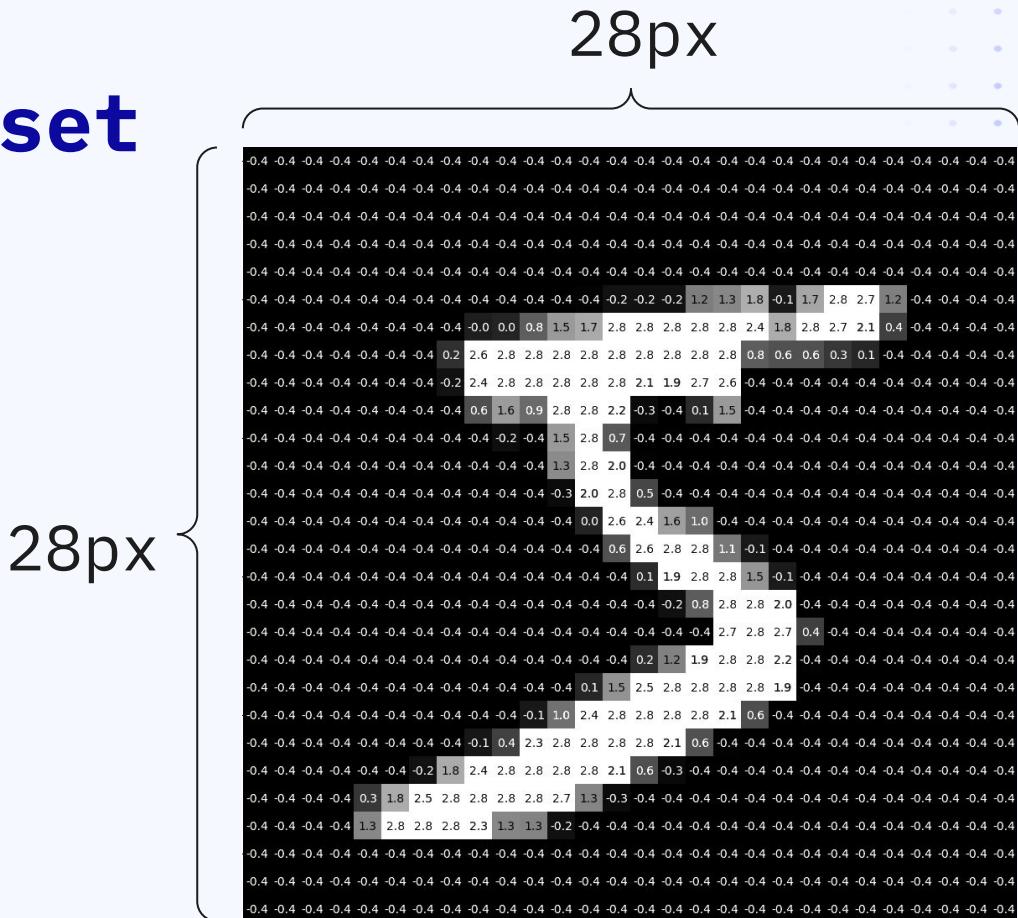


## Sobre o Dataset

# Sobre as Imagens

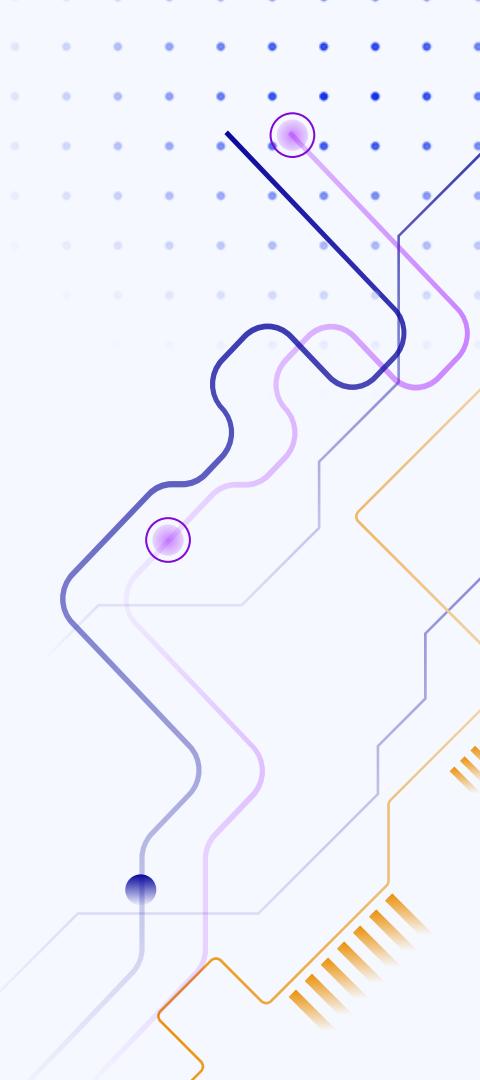
Cada uma tem rótulo de 0 a 9 (10 classes), que representam o dígito manuscrito.

Cada imagem pode ser representada como um vetor de 784 elementos.

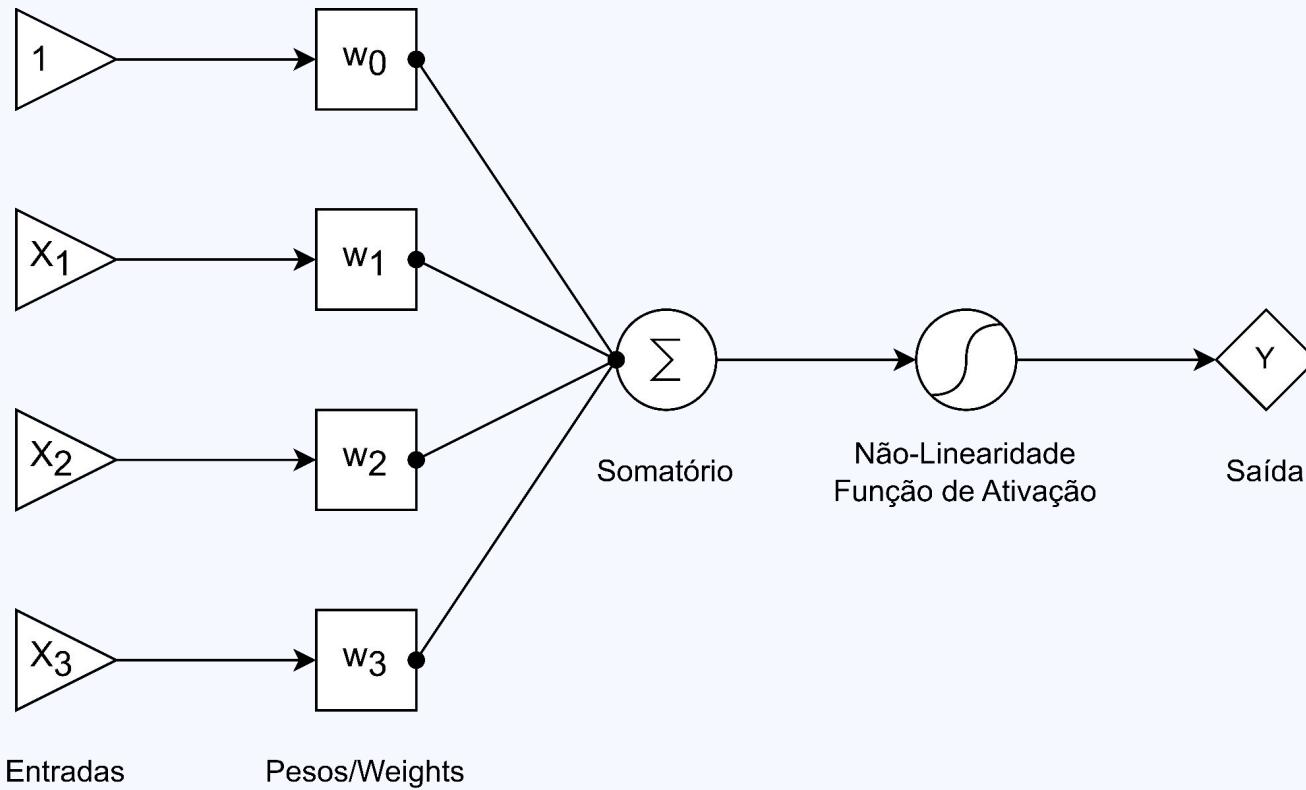


03

# Redes Neurais



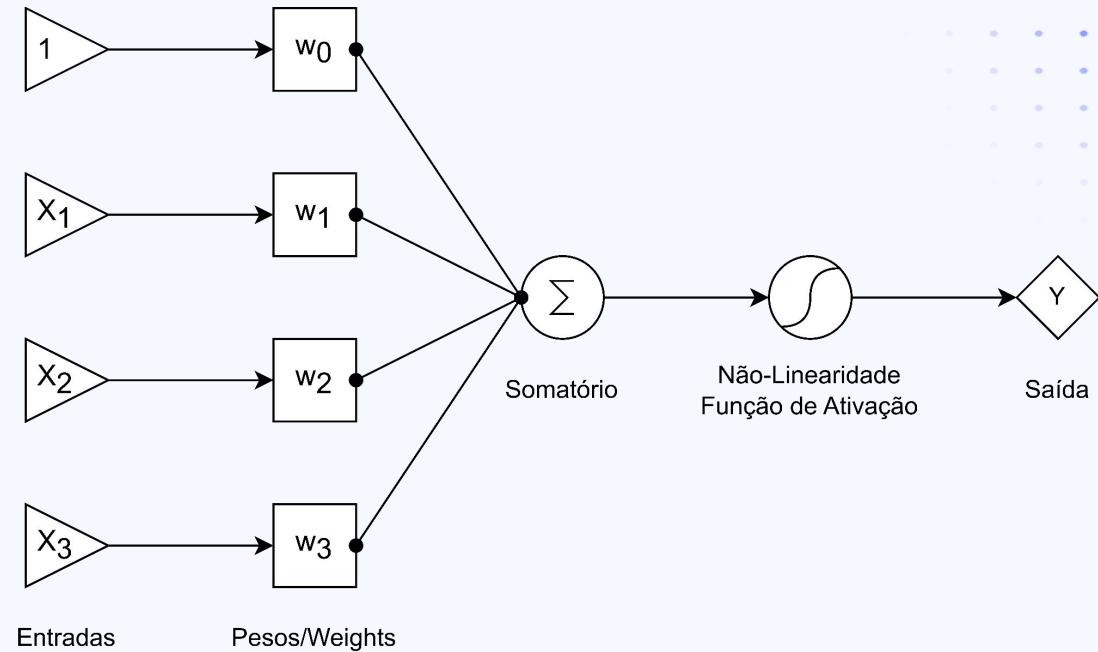
# Perceptron



# Perceptron

O Perceptron é a primeira tentativa de um modelo de um único neurônio artificial.

Desenvolvido por Frank Rosenblatt em 1958, conseguindo resolver problemas de classificação linearmente separáveis.

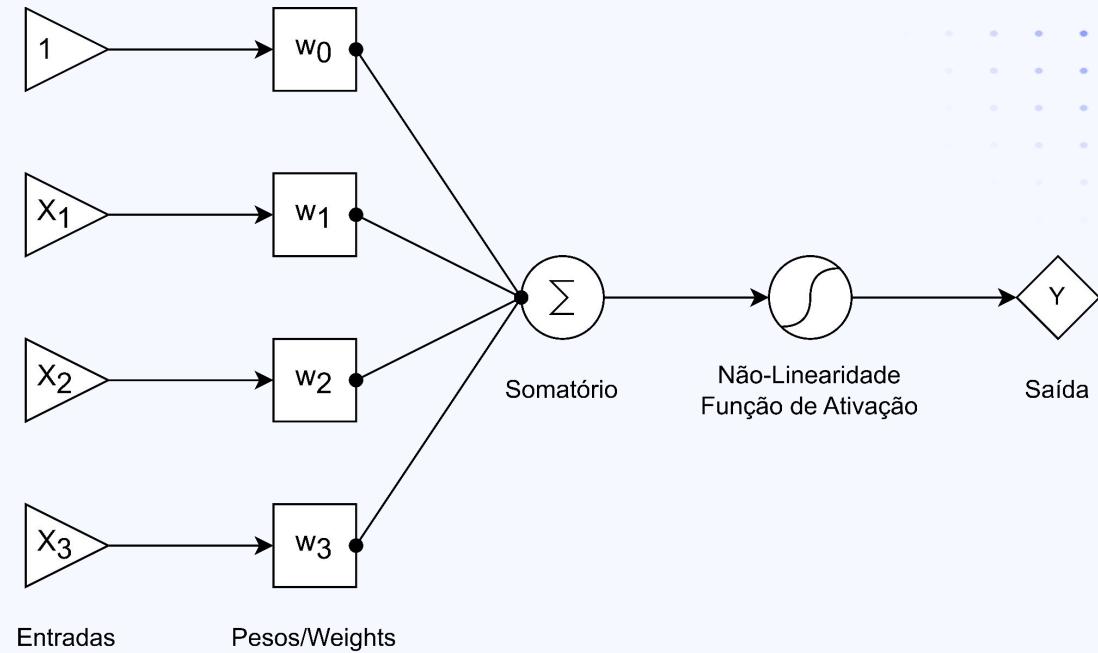


# Perceptron

Inputs: Recebe múltiplos inputs.

Pesos: Cada input é multiplicado por um peso ajustados durante o aprendizado.

Função de Ativação: Aplica uma função que decide a saída.

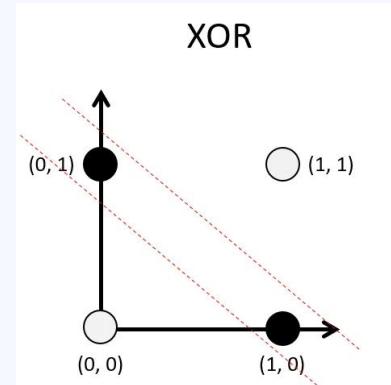
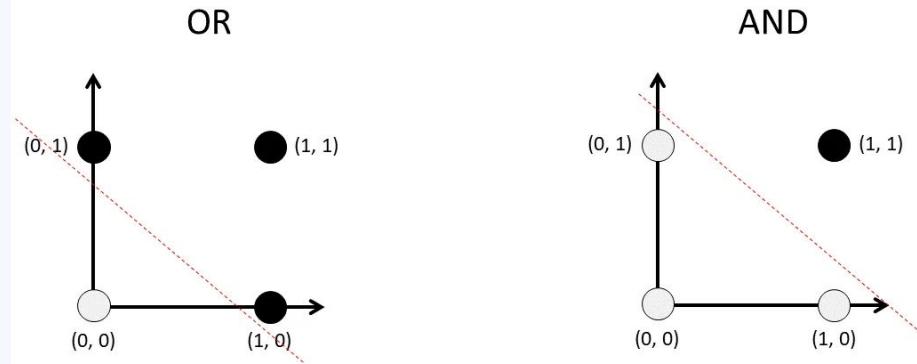


# Problemática - Linear

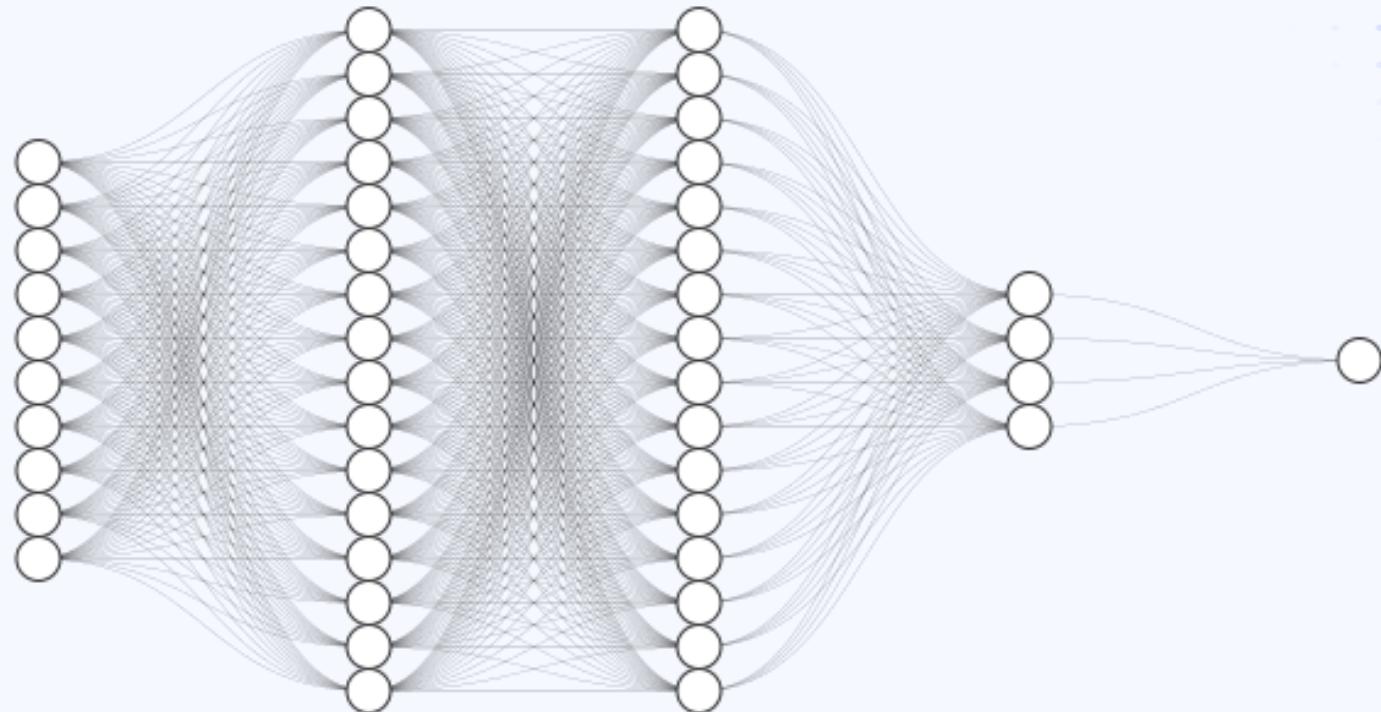
O Perceptron só pode resolver problemas que são linearmente separáveis, ou seja, classes podem ser separadas por linha reta.

Exemplo: O Perceptron pode resolver o problema do OU (OR), mas não pode resolver o problema do XOR.

$$y = \sigma(w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n + b)$$



# Redes Neurais Artificiais (ANNs)



Input Layer  $\in \mathbb{R}^{10}$

Hidden Layer  $\in \mathbb{R}^{16}$

Hidden Layer  $\in \mathbb{R}^{16}$

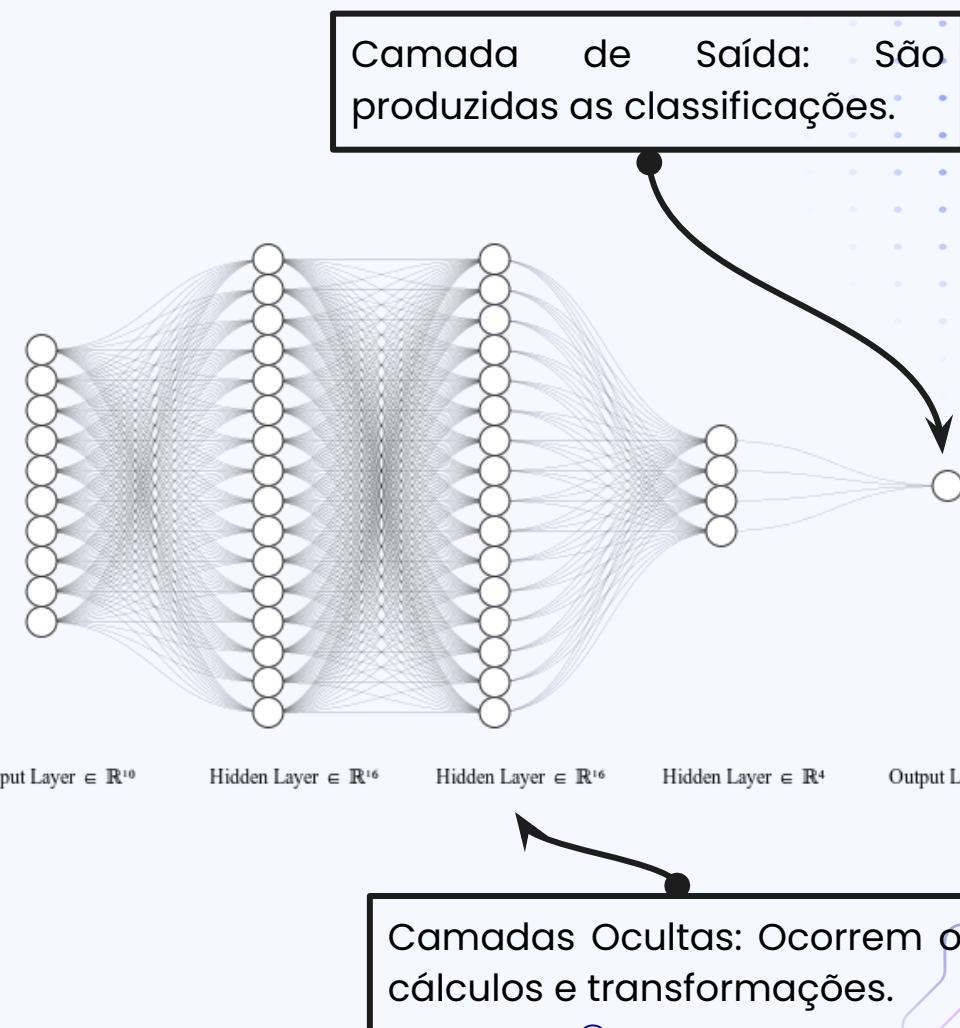
Hidden Layer  $\in \mathbb{R}^4$

Output Layer  $\in \mathbb{R}^1$

# ANNs ou MLPs

São modelos inspirados no funcionamento do cérebro. Formadas por neurônios artificiais organizados em camadas.

Camada de Entrada: dados são introduzidos no modelo.



# Funções de Ativação

$$\text{ReLU}(z) = R(z) = \max(0; z)$$

$$\text{sigmoid}(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\tanh(z) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

# Treinamento

Forward Pass:

$$z_1 = \text{ReLU}(b_0 + w_1 \cdot x_1 + w_2 \cdot x_2)$$

$$z_2 = \text{ReLU}(b_1 + w_3 \cdot x_1 + w_4 \cdot x_2)$$

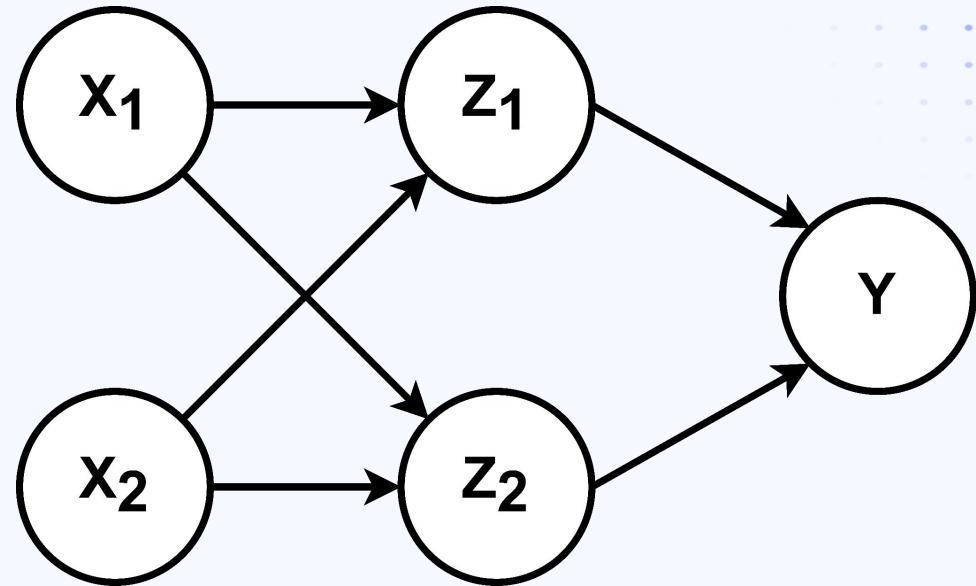
$$y = \text{ReLU}(b_2 + w_5 \cdot z_1 + w_6 \cdot z_2)$$

Loss Function:

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Backpropagation/Gradient Descent

$$w_i^* = w_i - \eta \cdot \frac{\partial L(w)}{\partial w_i}$$



04

# Exercício-Programa

# Passo-a-passo do EP:

**01**



**Probabilidade**

Questões sobre  
probabilidades  
de pixels

**02**



**EDA**

Análise  
exploratória  
dos Dados

**03**



**Modelos**

Teste de  
Modelos de  
ANNs ≠

**04**



**Resultado**

Comunicação  
visual e  
interpretação

# 1. Probabilidade e Estatística (2 pontos)



- a) Descubra o número do Dataset associado ao seu Hash.
- b) Calcule a probabilidade de um píxel ser claro na sua imagem obtida no item anterior.
- c) Qual é a probabilidade de um píxel ser claro dentre todos os píxeis que tem a mesma classe que a sua imagem obtida em a)?

## 2. EDA e Pré-processamento (2 pontos)



- a) Conte quantas vezes cada dígito (de 0 a 9) aparece. Todos os dígitos aparecem a mesma quantidade? Qual o valor médio dos píxeis dos dígitos?
- b) Faça um histograma que mostre a distribuição dos valores dos píxeis para cada dígito. Há muitos valores que são “apagados” (ou seja, com valor 0) ou a distribuição dos valores é mais equilibrada entre os dígitos?
- c) Crie uma imagem para cada dígito (de 0 a 9) em que cada píxel dessa nova imagem representa a média do valor dos píxeis para aquela classe. Você consegue reconhecer os dígitos nas imagens criadas?

### 3. Teste de ANNs (2 pontos)



- a) Treine a ANN1 com 784 entradas, 8 neurônios na 1<sup>a</sup> camada oculta, 8 neurônios na 2<sup>a</sup> camada oculta e 10 saídas. Utilize 5 épocas para o treinamento.
- b) Treine a ANN2 com 784 entradas, 256 neurônios na 1<sup>a</sup> camada oculta, 256 neurônios na 2<sup>a</sup> camada oculta, 256 neurônios na 3<sup>a</sup> camada oculta, 256 neurônios na 4<sup>a</sup> camada oculta e 10 saídas. Utilize 20 épocas dessa vez.

### 3. Teste de ANNs

#### (2 pontos)

- c. Crie 5 diferentes configurações de redes neurais e utilize o método **GridSearchCV()** para encontrar um classificador que seja melhor que os anteriores dos itens a) e b).
- d. Para os modelos treinados nas questões a) e b), além do classificador encontrado na questão 3, compare o desempenho dos modelos, analisando se apresentam *underfitting* ou *overfitting*. Justifique com gráficos e análises.

## 4. Resultados e Visualizações (2 pontos)



- a) Gere e apresente uma matriz de confusão que mostre a distribuição das previsões do melhor modelo. Quais as métricas de Acurácia, Precisão, Recall e F1-Score para esse modelo?
- b) Exiba gráficos que mostram a evolução da acurácia e da perda durante o treinamento e validação do melhor modelo.
- c) Escolha algumas imagens do conjunto de teste e mostre previsões do seu modelo, com acertos e erros. Discuta quais fatores podem ter contribuído para essas previsões corretas e incorretas.

## 5. Kaggle e Documentação (2 pontos)



- a) Publique seu trabalho no Kaggle e envie a versão final.
- b) Utilize o formato de nome do notebook como **PMR3508-2024-HASH-MNIST**.
- c) Assegure-se de que seu classificador esteja funcionando corretamente e consiga fazer previsões.
- d) Documente seu código, explicando suas etapas e compartilhe seus *insights*.

# Organização

Três arquivos serializados:

- MNIST-images.pkl
- MNIST-labels.pkl
- MNIST-Validation-images.pkl

Uma pasta compactada:

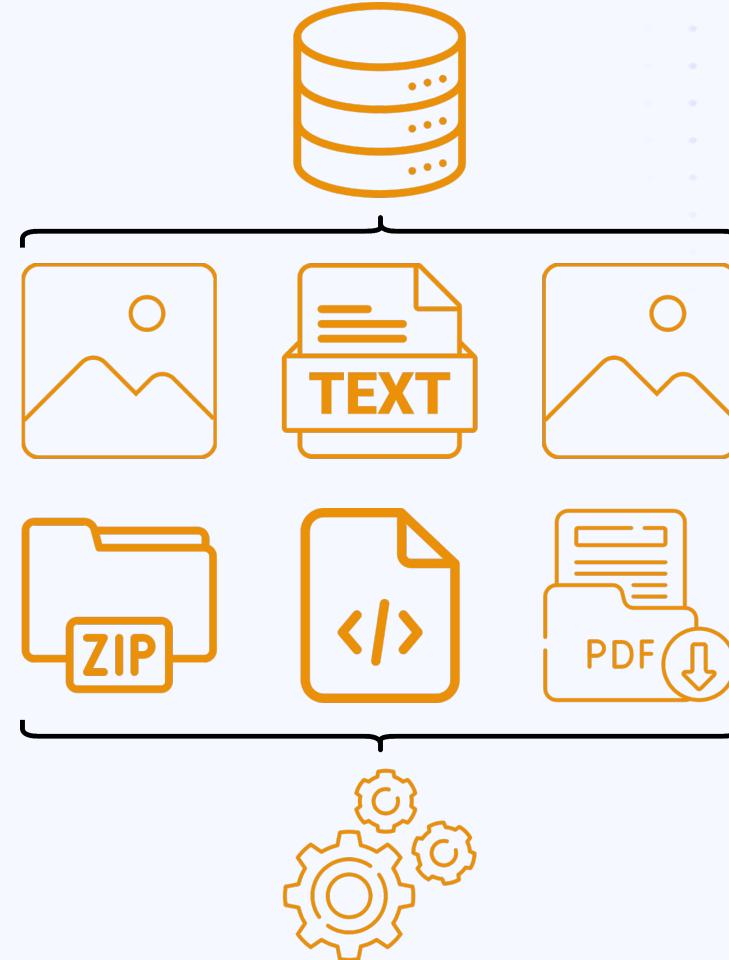
- MNIST-PMR3508

Um notebook de modelo:

- Template\_EP02.ipynb

Este PDF:

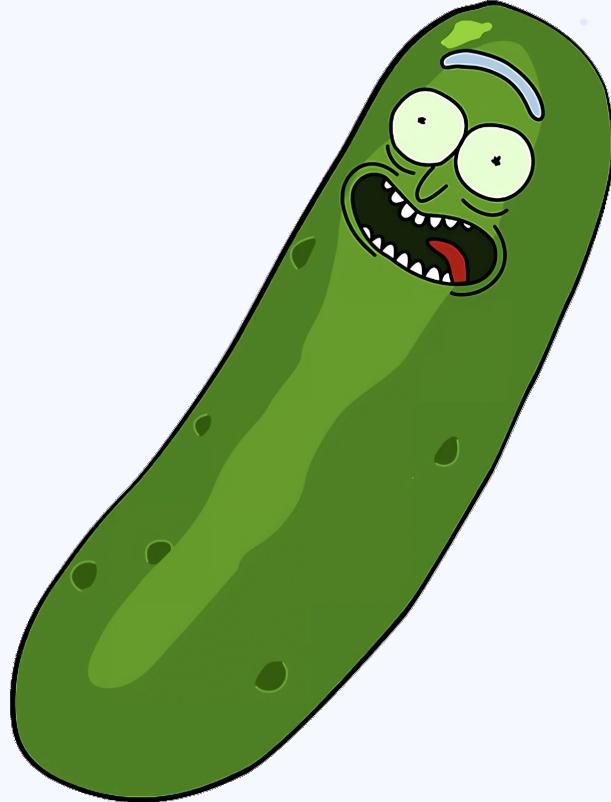
- Apresentação EP02.pdf



# Arquivo PKL

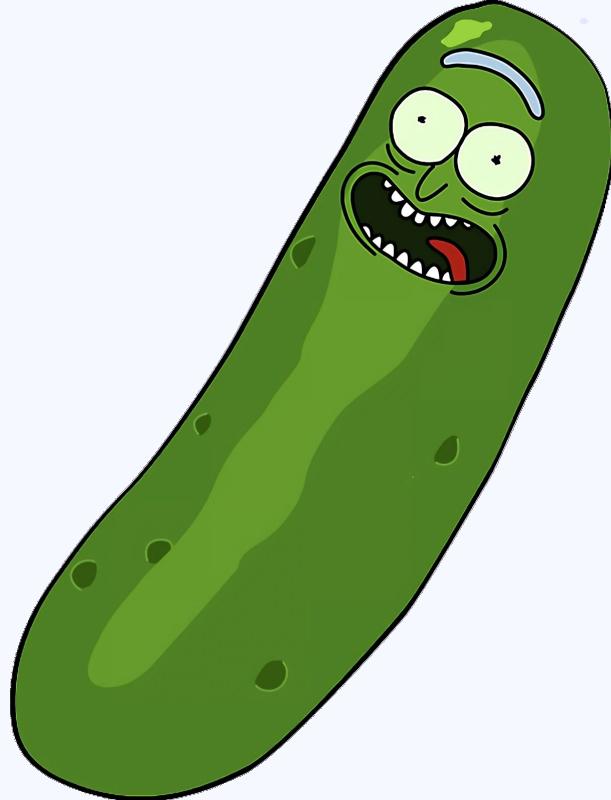
A biblioteca `pickle` do Python permite serializar objetos, convertendo-os em bytes para armazenamento ou transmissão, e depois desserializá-los, restaurando-os ao estado original. Isso é útil para salvar dados ou estados de programas, como modelos de Machine Learning, e reutilizá-los sem precisar recriar tudo do zero.

Neste EP usei para salvar alguns processamentos que fiz, como embaralhamento dos dados de validação.



# NÃO SE PREOCUPE

Os códigos para carregar os dados já foram feitos pelo monitor e podem ser acessados no modelo de código para ser usado, você apenas deve carregar os arquivos na sua sessão do Colab ou diretamente no Notebook do Kaggle



# Obrigado!



# Dúvidas? Questões?