

PALESTRA

Inteligência Artificial na Prática

dos conceitos ao primeiro modelo

com foco em visão computacional: IA aplicada a imagens

Matheus Arroyo de Lima

Engenheiro de Visão Computacional e Inteligência Artificial
Mestrando em Computação Aplicada na FFCLRP/USP

Ao fundo: dígitos reais do MNIST, o dataset do nosso primeiro exemplo.



Matheus Arroyo de Lima

FORMAÇÃO

Graduado em Ciência da Computação pelo Centro Universitário Barão de Mauá

PESQUISA

Mestrando em Computação Aplicada pela Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da Universidade de São Paulo (FFCLRP-USP)

MERCADO

Atua na área de Inteligência Artificial desde 2021; hoje é Engenheiro de Visão Computacional e Inteligência Artificial na IA SENSE

NO DIA A DIA

Visão computacional

Deteccção e contagem

Segmentação

Modelos na borda e na nuvem

Agro, indústria e laboratório

Da pesquisa acadêmica ao modelo rodando em campo.

O caminho que vamos percorrer

O recorte de hoje: IA aplicada a imagens. No fim, treinamos dois modelos de verdade: dígitos escritos à mão (Exemplo 1) e animais em fotos (Exemplo 2).

1

O que é IA

Definição e onde ela já está

2

Como a máquina aprende

Tipos de aprendizado · generativo × não generativo

3

A imagem

Do olho à câmera · pixel · RGB

4

Redes neurais

Do neurônio do cérebro ao artificial · função de perda · arquitetura

5

Visão computacional

Por que a CNN resolve o que a rede densa não consegue

6

Dados

Anotação · treino, validação e teste · aumento de dados

7

Ferramentas

TensorFlow × PyTorch e YOLO

8

Mão na massa

Dois exemplos completos: do dataset ao modelo treinado

9

Colocar para rodar

Onde o modelo roda: nuvem, servidor ou borda

10

IA na prática

Produtos reais que usam essas mesmas ideias

BLOCO 1

O que é Inteligência Artificial

Antes do código: o que a máquina faz de verdade quando dizemos que ela “aprende”.

1

Você provavelmente já usou IA hoje

Desbloqueio pelo rosto

Compara o rosto na câmera com o cadastrado: é você ou não?

Filtro de spam

Olha o e-mail e decide: spam ou não spam

Mapa e trânsito

Estima quantos minutos a viagem vai levar agora

Recomendações

Estima o quanto você vai gostar de um filme ou produto

Tradução automática

Gera a frase equivalente em outra língua

Assistentes de conversa

Geram uma resposta nova a partir do seu pedido

Produtos muito diferentes, feitos por empresas diferentes. Por baixo de todos eles há um punhado pequeno de ideias em comum.

Afinal, o que é Inteligência Artificial?

Sistemas que executam tarefas que, feitas por pessoas, exigiriam inteligência: reconhecer, prever, decidir, criar. Hoje, quase sempre aprendendo a partir de dados.

E o que é “aprender”? A definição clássica de aprendizado de máquina:

“Um programa aprende com a experiência E , em relação a uma tarefa T e a uma medida de desempenho P , se o seu desempenho em T , medido por P , melhora com E .”

MITCHELL, 1997 (tradução livre)

UM EXEMPLO CONCRETO



Tarefa

reconhecer dígitos escritos à mão



Desempenho

acurácia: % de dígitos acertados



Experiência

60.000 imagens com a resposta certa

A virada: em vez de escrever as regras, mostramos exemplos

PROGRAMAÇÃO TRADICIONAL



APRENDIZADO DE MÁQUINA



Tente escrever regras para reconhecer um “7” escrito à mão: traço horizontal? inclinação? e o 7 cortado? e o que parece um 1?

Mostrar 6.265 exemplos de “7” (é quantos o MNIST tem) é muito mais fácil. O treino descobre as regras sozinho.

Nem toda IA aprende, e nem todo aprendizado é deep learning

INTELIGÊNCIA ARTIFICIAL

também regras, busca e lógica escritas à mão

APRENDIZADO DE MÁQUINA

aprende com dados: árvores, SVM, k-vizinhos...

REDES NEURAIS

camadas de neurônios artificiais

DEEP LEARNING

muitas camadas: CNNs, Transformers

IA GENERATIVA

cria texto, imagem, áudio

Como ler o mapa

Cada caixa está dentro da anterior.

Deep learning é um tipo de rede neural, que é um tipo de aprendizado de máquina, que é uma forma de fazer IA.

Nem todo problema precisa de deep learning.

Em tabelas pequenas, árvores de decisão costumam ganhar.

O recorte de hoje:

redes profundas para visão computacional, ou seja, IA aplicada a imagens. Quase tudo que vamos ver vale para os outros ramos, mas os exemplos são de imagem.

BLOCO 2

Como a máquina aprende

Três formas de aprender e dois grandes tipos de problema. Com isso, dá para classificar quase qualquer aplicação de IA.

2

Três formas de aprender

Supervisionado

Aprende com exemplos que já trazem a resposta certa (o rótulo).

EXEMPLOS

foto → “zebra”
e-mail → “spam”
casa → R\$ 480 mil

Qual é a resposta para um exemplo novo?

Não supervisionado

Recebe dados sem rótulo e encontra estrutura sozinho.

EXEMPLOS

agrupar clientes parecidos
segmentar talhões
detectar comportamento anômalo

Que grupos ou padrões existem aqui?

Por reforço

Um agente age num ambiente e aprende por tentativa e erro, guiado por recompensas.

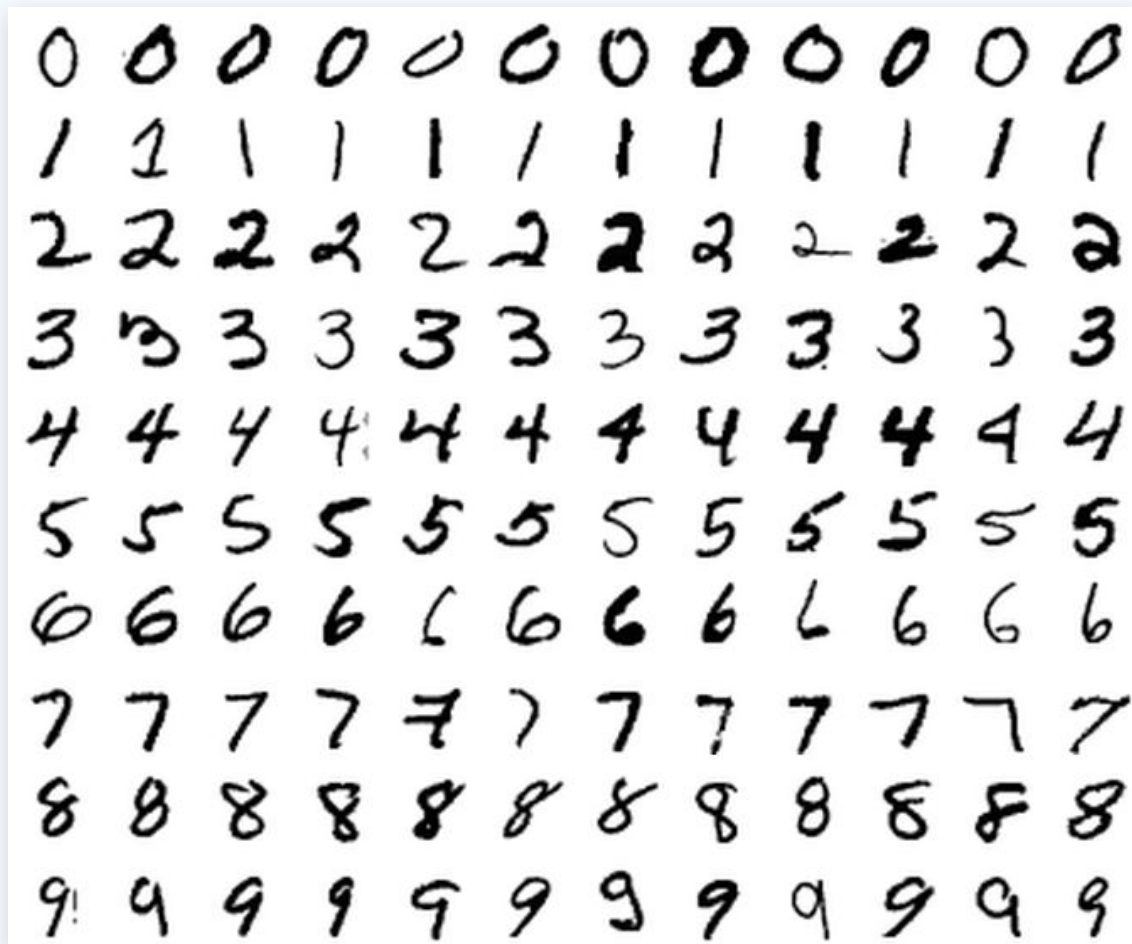
EXEMPLOS

jogos
robôs e controle
estratégias de decisão

Qual ação rende mais recompensa ao longo do tempo?

Hoje ficamos no supervisionado: nos dois exemplos, cada imagem vem com a resposta certa.

Aprender com gabarito



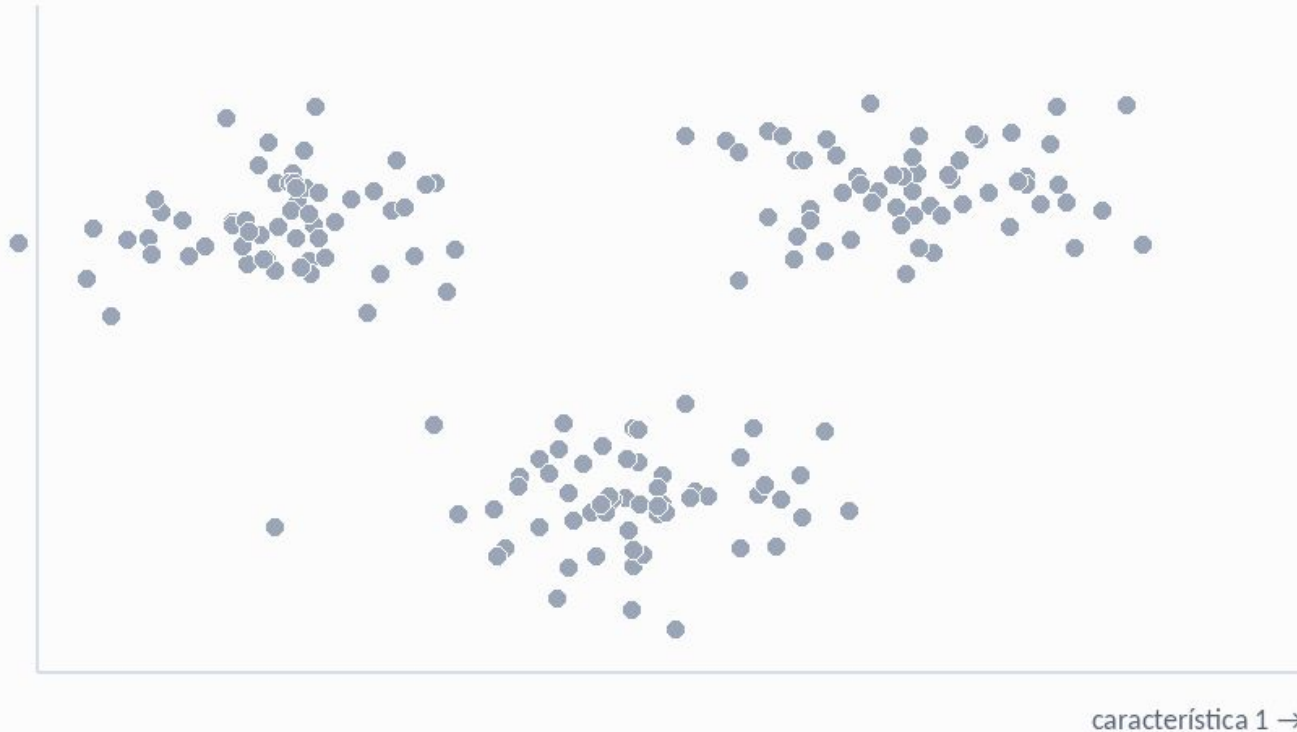
O MNIST

- 70.000 dígitos escritos à mão, 28×28 pixels, em tons de cinza
- 60.000 para treinar e 10.000 para testar
- Cada imagem vem com o rótulo certo: 0, 1, 2... 9
- É o “hello world” da visão computacional desde 1998

Repare na variedade à esquerda: 12 exemplos de cada dígito, todos reais. É essa variedade que o modelo precisa aprender a generalizar.

Achar grupos sem ninguém dizer quais são

Dados sem rótulo: ninguém disse quais são os grupos

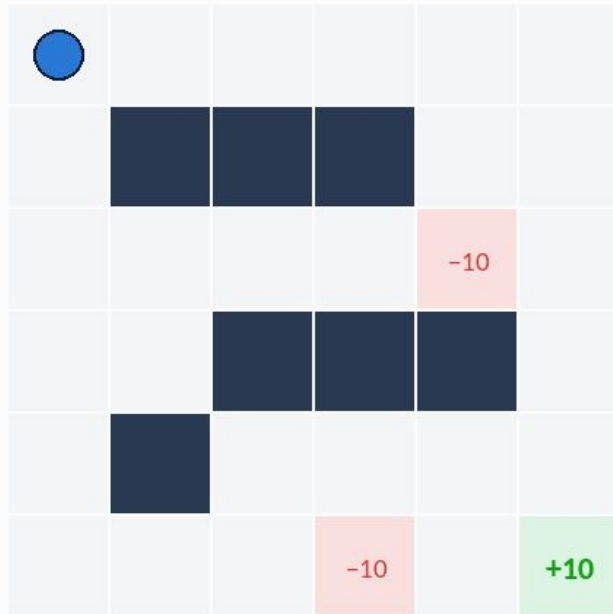


O que acontece na animação

- Os pontos chegam sem cor: não há rótulo
- O algoritmo (k-means) chuta 3 centros
- Cada ponto vai para o centro mais próximo
- Cada centro vai para o meio do seu grupo
- Repete até estabilizar

Na prática: agrupar talhões com comportamento parecido, perfis de consumo, células em uma lâmina.

Aprender por tentativa, erro e recompensa



Episódio 1

passos: 0

Agente: só vê onde está

Ação: cima, baixo, esq., dir.

Recompensa: +10 na meta,
-10 na armadilha, -0,1 por passo

Agente

quem decide (o círculo azul)

Ambiente

o labirinto

Ação

cima, baixo, esquerda, direita

Recompensa

+10 na meta, -10 na armadilha

Política

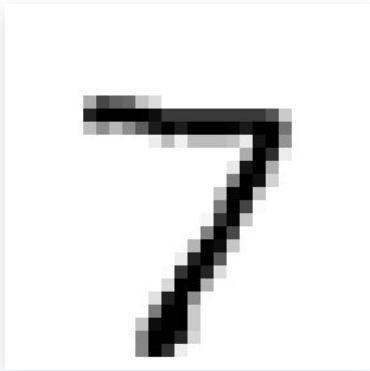
a regra que o agente aprendeu

Episódios 1, 8, 40 e 300 de um agente treinado aqui (Q-learning): cai na armadilha, anda em círculos e, no fim, faz o caminho de 10 passos.

Generativo × não generativo

NÃO GENERATIVO

Olha a entrada e responde algo sobre ela



→ “é um 7”

Também chamado de discriminativo. Classifica, estima, detecta. É o que vamos treinar hoje.

GENERATIVO

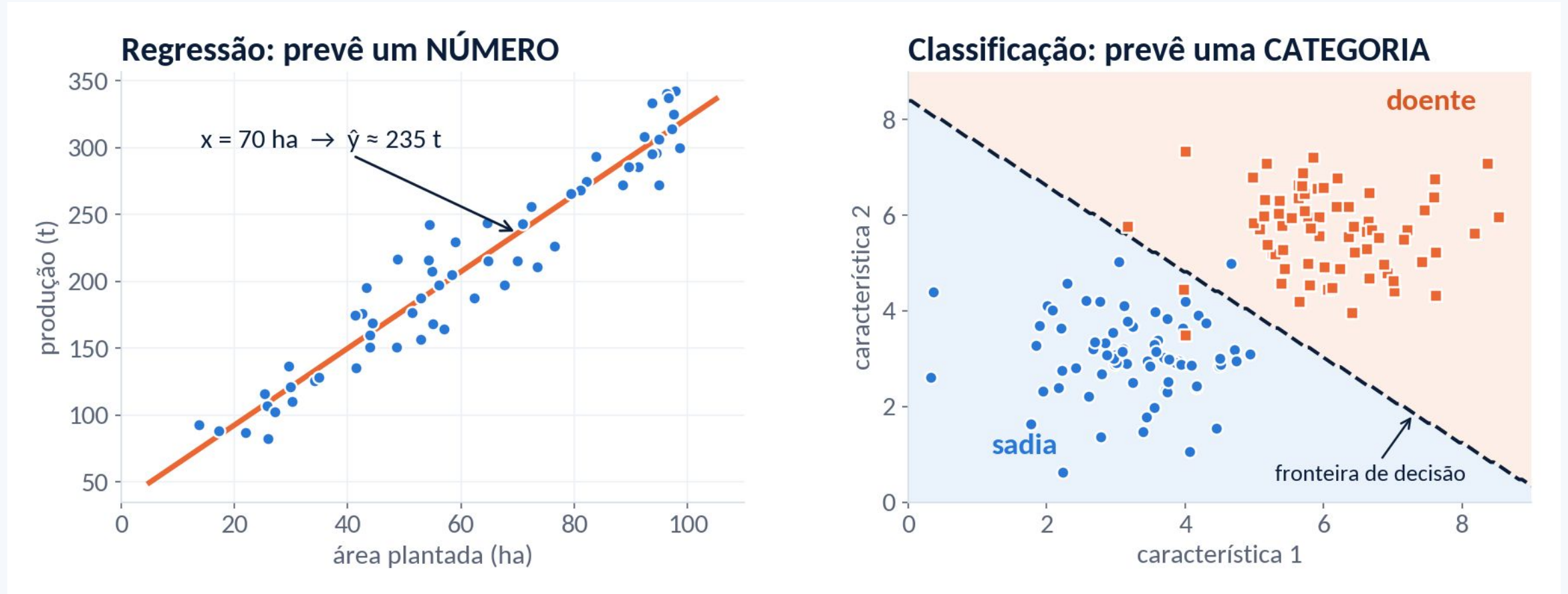
Aprende como os dados são e cria exemplos novos



Nenhum desses dígitos existe no MNIST: foram criados por um modelo generativo simples treinado com ele aqui, para esta palestra.

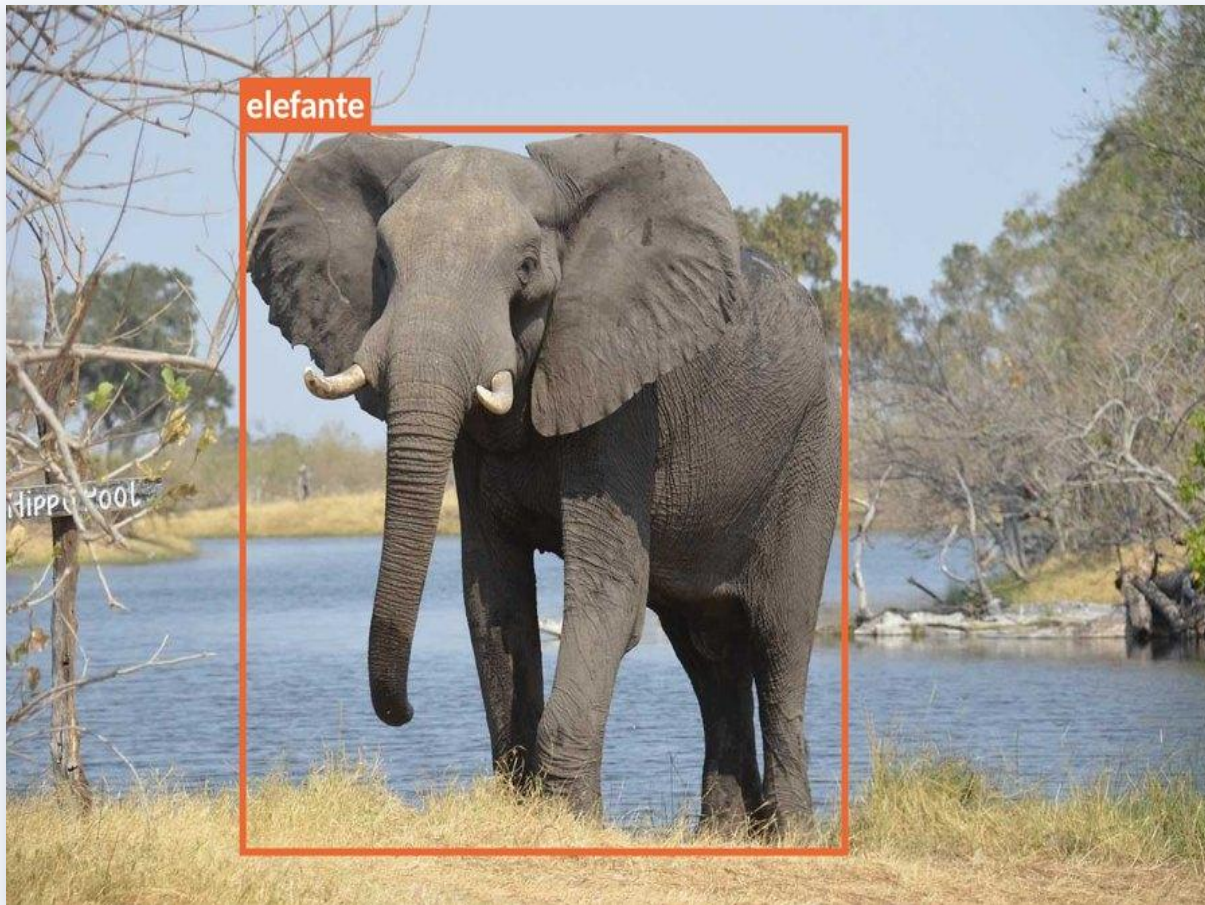
Em escala: texto (modelos de linguagem), imagem, áudio, código.

No não generativo, quase tudo é um de dois problemas



Regressão: produtividade, preço, temperatura, tempo de viagem. Classificação: doente ou sadia, dígito 0–9, espécie, spam.

E a detecção de objetos? É os dois ao mesmo tempo



Onde está?

regressão

4 números: centro x, centro y, largura, altura

O que é?

classificação

uma das classes: búfalo, elefante, rinoceronte, zebra

Com que certeza?

probabilidade

um número de 0 a 1: a confiança

Segmentação: classificar cada pixel.
Pose: regressão de pontos do corpo.

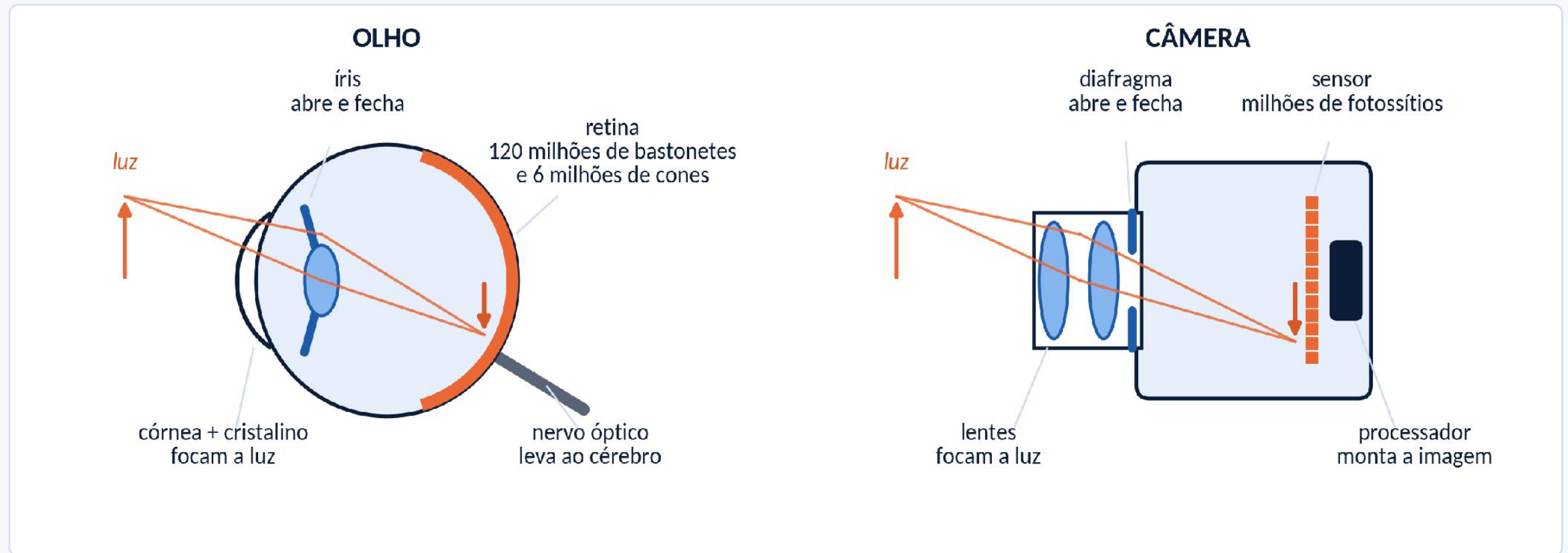
BLOCO 3

A imagem

Antes de ensinar o computador a ver, vale entender o que é uma imagem para ele.

3

Do olho à câmera: a mesma receita



lente

córnea e cristalino · vidro da objetiva

abertura

íris · diafragma

detector

retina · sensor

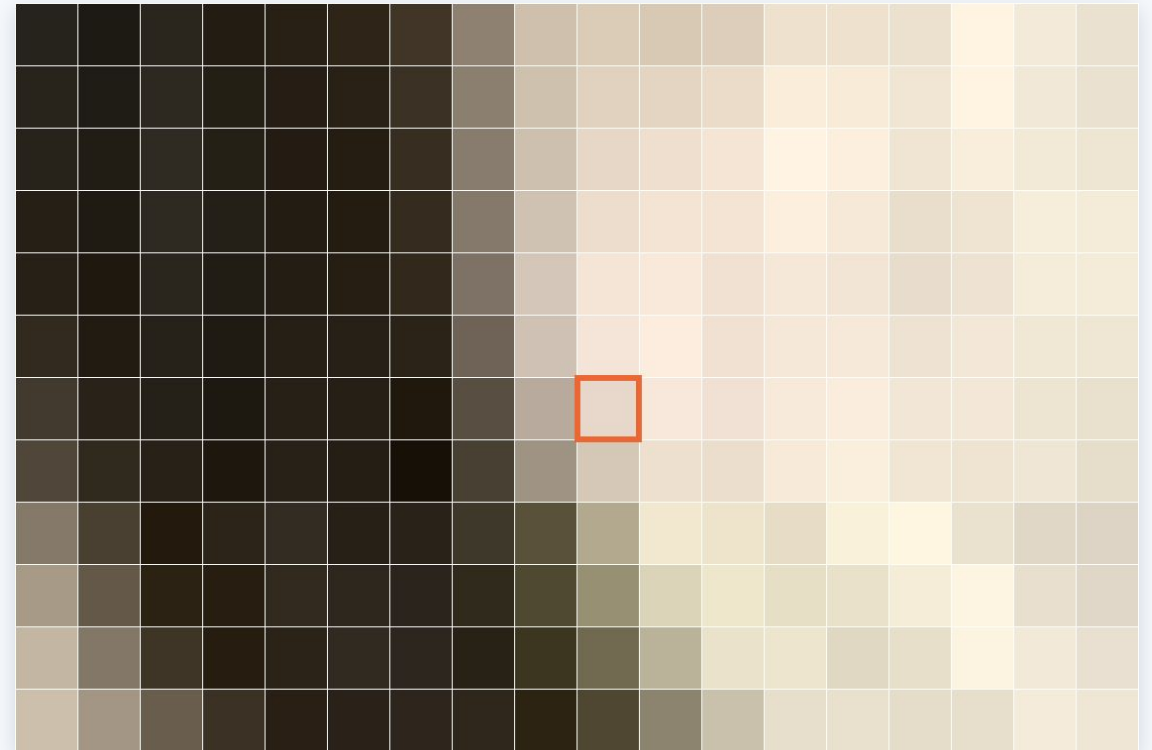
intérprete

cérebro · processador

Pixel: a menor peça da imagem



A foto inteira: $960 \times 640 = 614.400$ pixels



Um pedaço de 18×12 pixels, ampliado. O pixel em destaque é (232, 216, 203).

Pixel vem de picture element. A imagem é uma grade de pixels, cada um com uma cor só. Resolução é o tamanho dessa grade: amplie o suficiente e a foto sempre vira quadradinhos.

RGB: três números para cada pixel



a foto



só o vermelho (R)



só o verde (G)



só o azul (B)

R

205	206	206	207	209	208
206	207	207	208	210	209
207	207	208	208	210	210
208	208	209	209	211	211
208	209	209	210	212	211
208	207	210	208	212	210

G

177	178	178	179	181	180
178	179	179	180	182	181
179	179	180	180	182	182
180	180	181	181	183	183
180	181	181	182	184	183
180	182	182	183	184	185

B

137	138	138	139	141	140
138	139	139	140	142	141
139	139	140	140	142	142
140	140	141	141	143	143
140	141	141	142	144	143
140	141	142	142	144	144

Recorte de 6×6 pixels do capim, número por número.

CADA PIXEL, TRÊS NÚMEROS

0 a 255 em cada canal

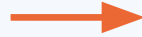
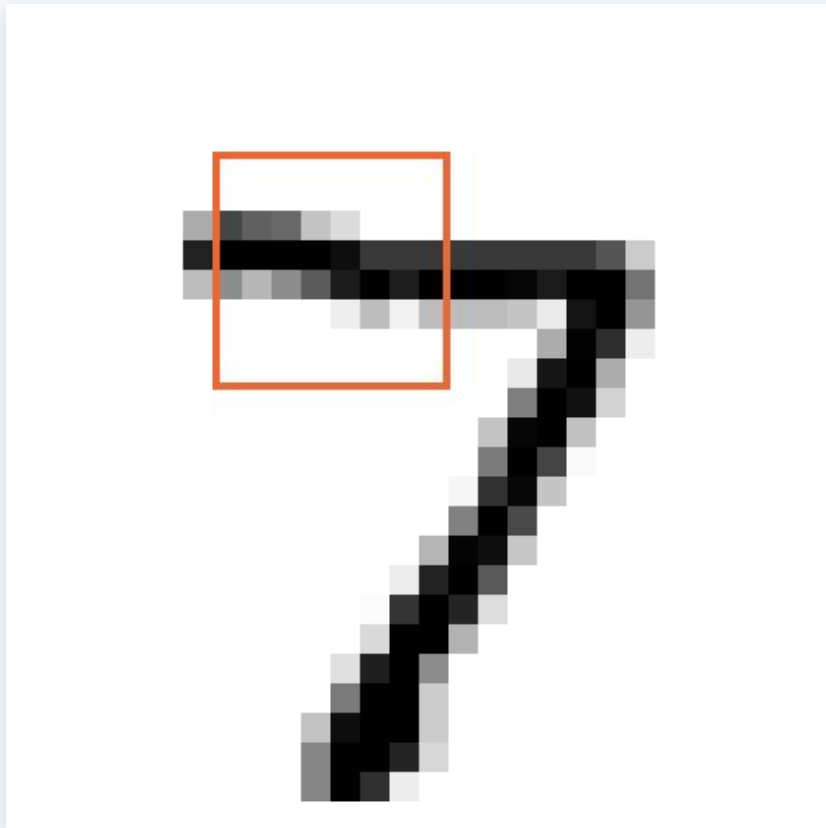
vermelho, verde e azul

16,7 milhões de cores

$256 \times 256 \times 256$ combinações

O capim é muito vermelho, verde médio e pouco azul. A tela faz o caminho de volta: mistura os três e você vê a cor.

Para o computador, imagem é uma tabela de números



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
185	159	151	60	36	0	0	0
254	254	254	254	241	198	198	198
114	72	114	163	227	254	225	254
0	0	0	0	17	66	14	67
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tons de cinza

$28 \times 28 = 784$ números de 0 a 255

Colorida

3 tabelas: vermelho, verde e azul

Uma foto 640×640

1.228.800 números

O “7” do teste do MNIST, ampliado. À direita, os valores reais da região marcada: 0 é fundo, 255 é tinta.

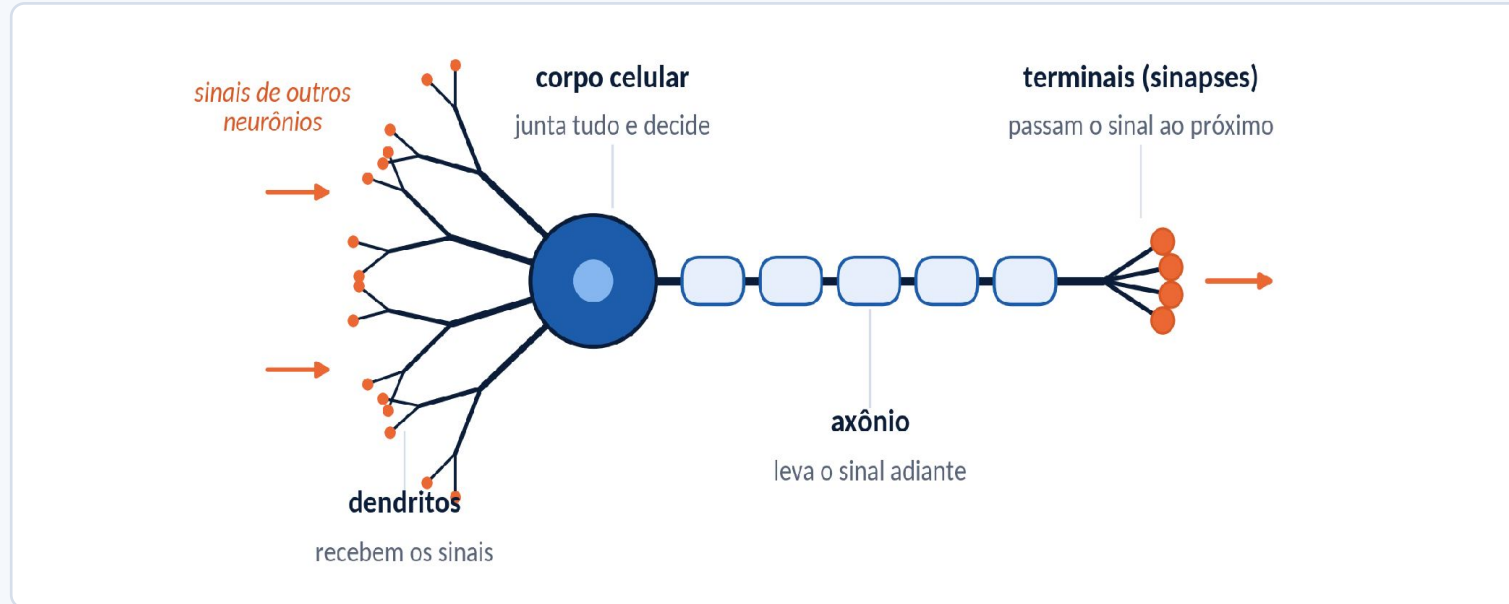
BLOCO 4

Redes neurais

Uma função enorme, feita de peças muito simples e ajustada com dados.



A inspiração: o neurônio do cérebro



Cada neurônio recebe sinais de milhares de outros. Se a soma passar de um limite, ele dispara e avisa os seguintes; se não passar, fica quieto.

86 bilhões

neurônios no cérebro humano

milhares

de conexões por neurônio

42

neurônios na rede que veremos daqui a pouco

O QUE O MODELO COPIOU

dendritos

↓ **entradas (x)**

força da sinapse

↓ **peso (w)**

juntar os sinais

↓ **soma ponderada + viés**

disparar ou não

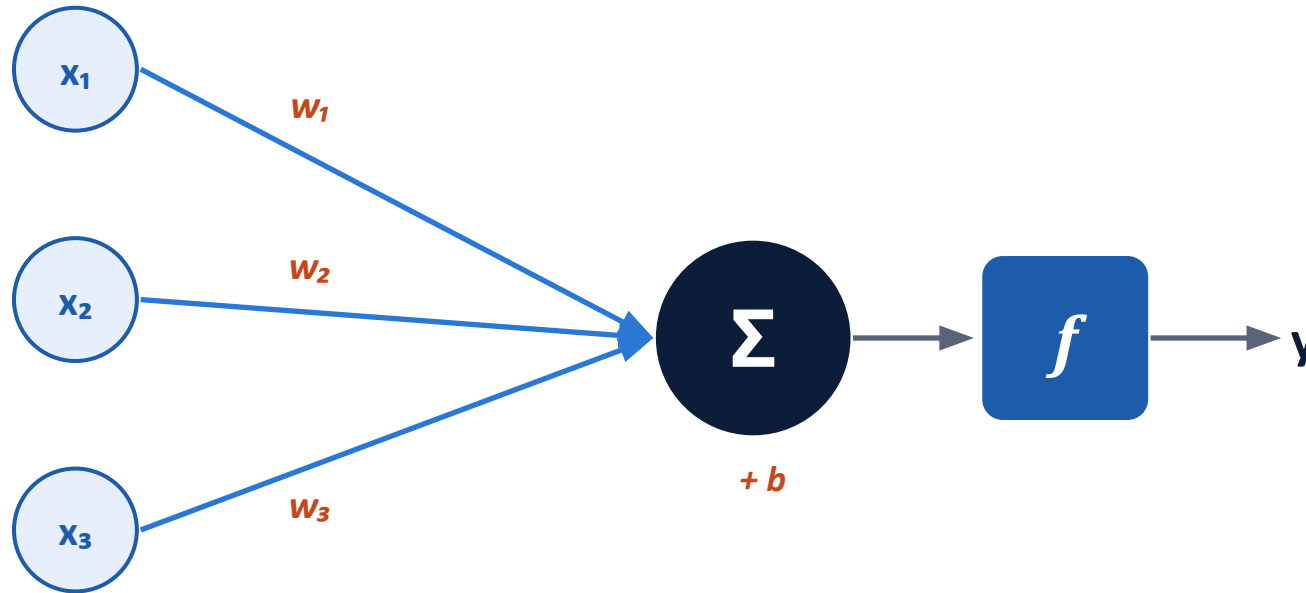
↓ **função de ativação**

axônio

↓ **saída (y)**

A cópia para por aí: o neurônio de verdade é muito mais complexo, dispara em pulsos e muda com o tempo.

O neurônio artificial: multiplica, soma e decide



$$y = f(w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + b)$$

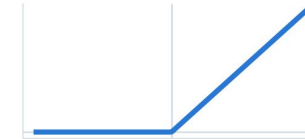
x entradas (ex.: pixels)

w pesos: o quanto cada entrada importa

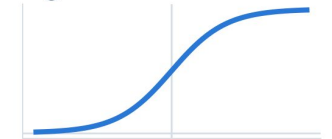
b viés: um ajuste fino

f ativação: deixa a conta não linear

ReLU



Sigmoide



Treinar = encontrar os valores de w e b . Sem f , empilhar mil camadas equivaleria a uma única conta linear.

A função de perda: o erro em um número só



A mesma rede 784-16-16-10 em duas imagens de teste: quando ela dá 99% para a resposta certa, a perda é quase zero; quando dá 16%, a perda sobe para 1,86.

previsão

o que o modelo respondeu

gabarito

o que era para responder

perda

um número: o tamanho do erro

O QUE PRECISA FICAR

Uma perda por exemplo

a perda da época é a média de todas

Treinar é minimizar

todo o ajuste dos pesos existe para fazer esse número cair

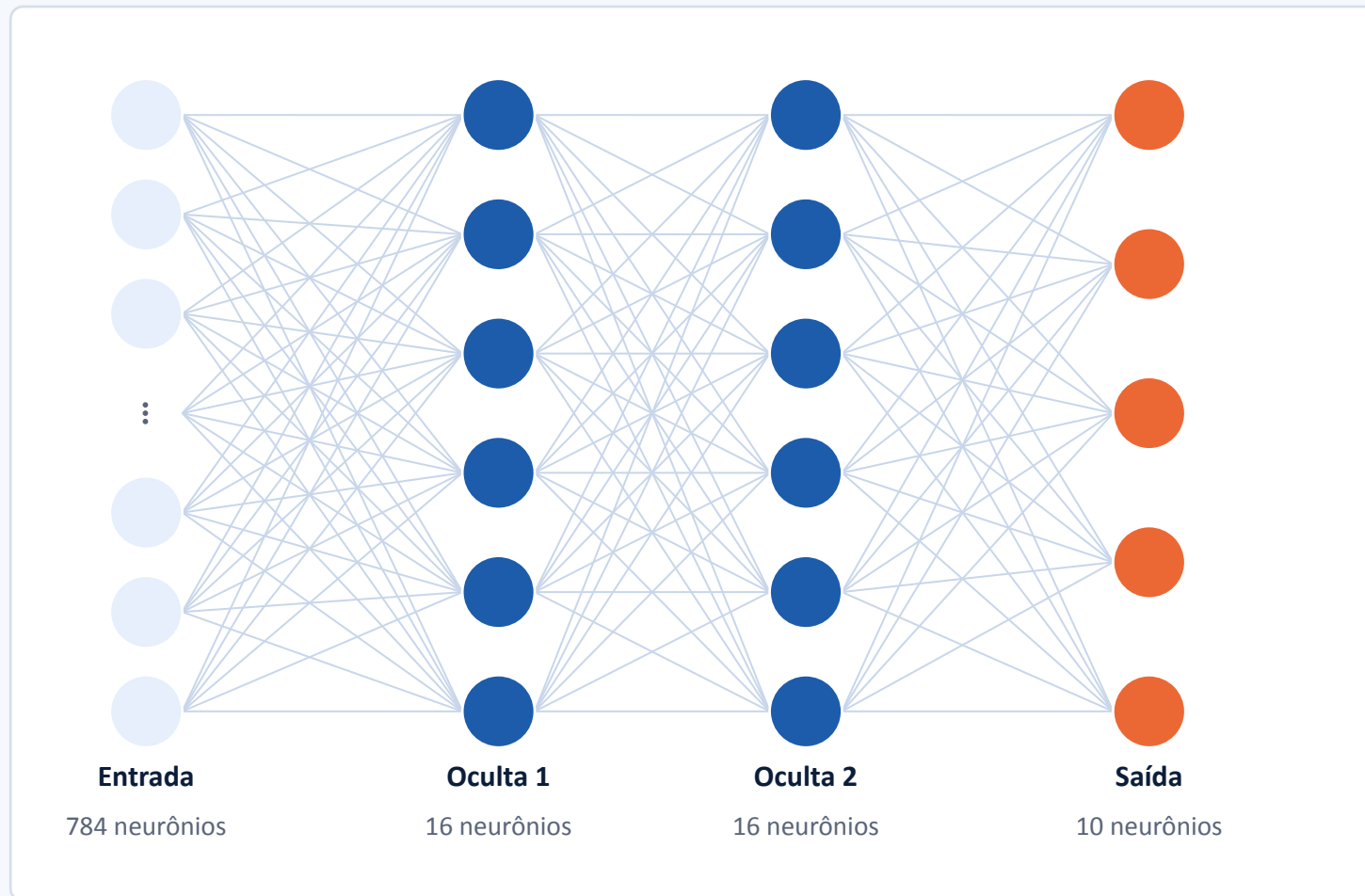
Perda não é métrica

a perda guia o treino; acurácia e mAP medem o resultado

Ela precisa ser suave

é a derivada da perda que diz para que lado mexer cada peso

Arquitetura: como os neurônios se organizam



QUANTOS PESOS?

$$784 \times 16 + 16 = 12.560$$

$$16 \times 16 + 16 = 272$$

$$16 \times 10 + 10 = 170$$

13.002

parâmetros ajustados no treino

Treinada aqui no MNIST: 95,5% de acerto no teste.

Largura = neurônios por camada · profundidade = número de camadas · “deep” = muitas camadas ocultas.

A rede em ação: um dígito atravessando as camadas



Brilho do círculo

o quanto aquele neurônio “acendeu”
(ativação)

Linha azul

contribuição que empurra para cima

Linha laranja

contribuição que empurra para baixo

Barras da saída

probabilidade de cada dígito

*No último dígito a rede hesita entre 4 e 9,
e mostra isso nas probabilidades.*

De 13 mil a milhões de parâmetros: redes de verdade

Rede densa do MNIST

nossa rede 784-16-16-10: 3 camadas



13 mil de parâmetros

YOLO26n-cls

classificador de dígitos do Exemplo 1: 86 camadas



1,5 milhão de parâmetros

YOLO26n

detector de animais do Exemplo 2: 260 camadas



2,5 milhões de parâmetros

Mesmas peças do neurônio simples, empilhadas em muito mais camadas e trocando o “denso” pela convolução.

BLOCO 5

Visão computacional

Por que a rede convolucional resolve o que a rede densa não consegue quando a entrada é uma imagem.

5

A rede densa “achata” a imagem numa fila de 784 números



o mesmo 3: no lugar · 4 px à direita · 4 px à esquerda e para baixo

Cada entrada é “o pixel da linha 10, coluna 7”. Se o dígito se mexe, os números caem em outras entradas e, para a rede densa, vira outra imagem.

Como a rede densa vê o 3 no lugar (784 entradas):



Como vê o mesmo 3, só que 4 pixels à direita:

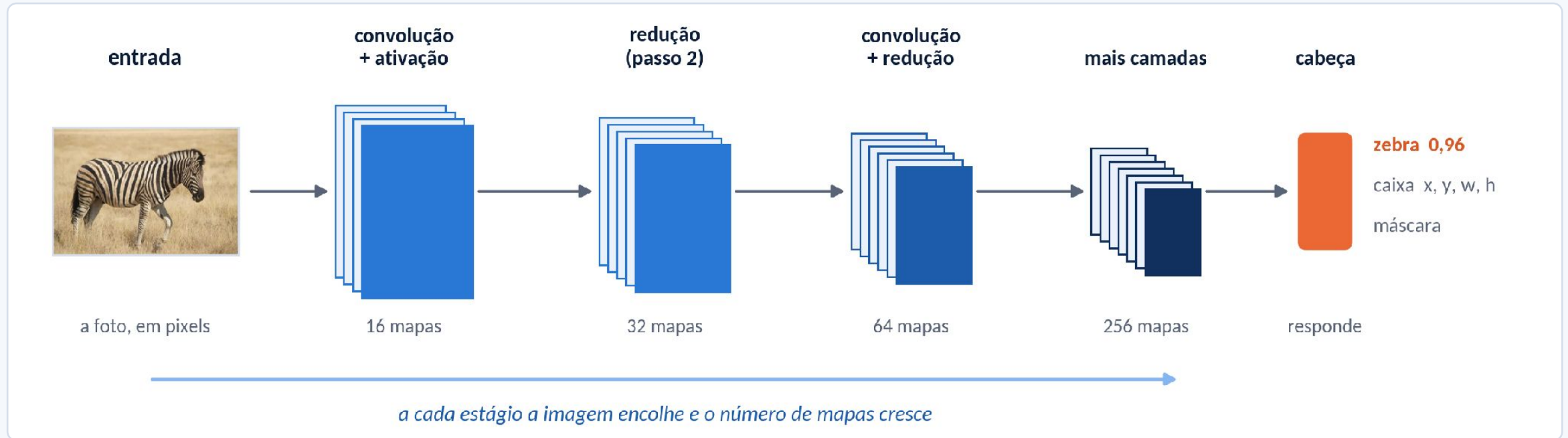


Entradas que mudaram de valor: 197 das 216 que têm tinta



A rede densa não sabe que o pixel 350 é vizinho do 351. Ela perde a noção de espaço.

A CNN: a rede desenhada para imagens



Convolução

um filtro pequeno passeia pela imagem inteira procurando o mesmo padrão em todo lugar

Redução

de tempos em tempos a imagem encolhe: sobra o que é forte e a conta fica mais leve

Profundidade

cada camada combina o que a anterior achou: bordas, depois texturas, depois partes

Cabeça

no fim, poucas camadas respondem o que se pediu: classe, caixa ou máscara

CNN é o apelido de rede neural convolucional. A rede densa continua ali no fim, só que agora ela recebe características, e não pixels crus.

BLOCO 6

Dados

O modelo é tão bom quanto os dados que ele vê. Aqui mora a maior parte do trabalho.

6

Garbage in, garbage out.

Entra lixo, sai lixo.

Algoritmo bom se baixa pronto. Dado bom e bem rotulado, não.

O MODELO COPIA O QUE VÊ

1

Rótulo errado

ele aprende o erro com convicção

2

Só fotos de dia

ele não conhece a noite

3

Uma classe rara

ele aprende a ignorá-la

4

Teste parecido demais com o treino

a métrica mente

Anotar: de onde vêm as respostas certas

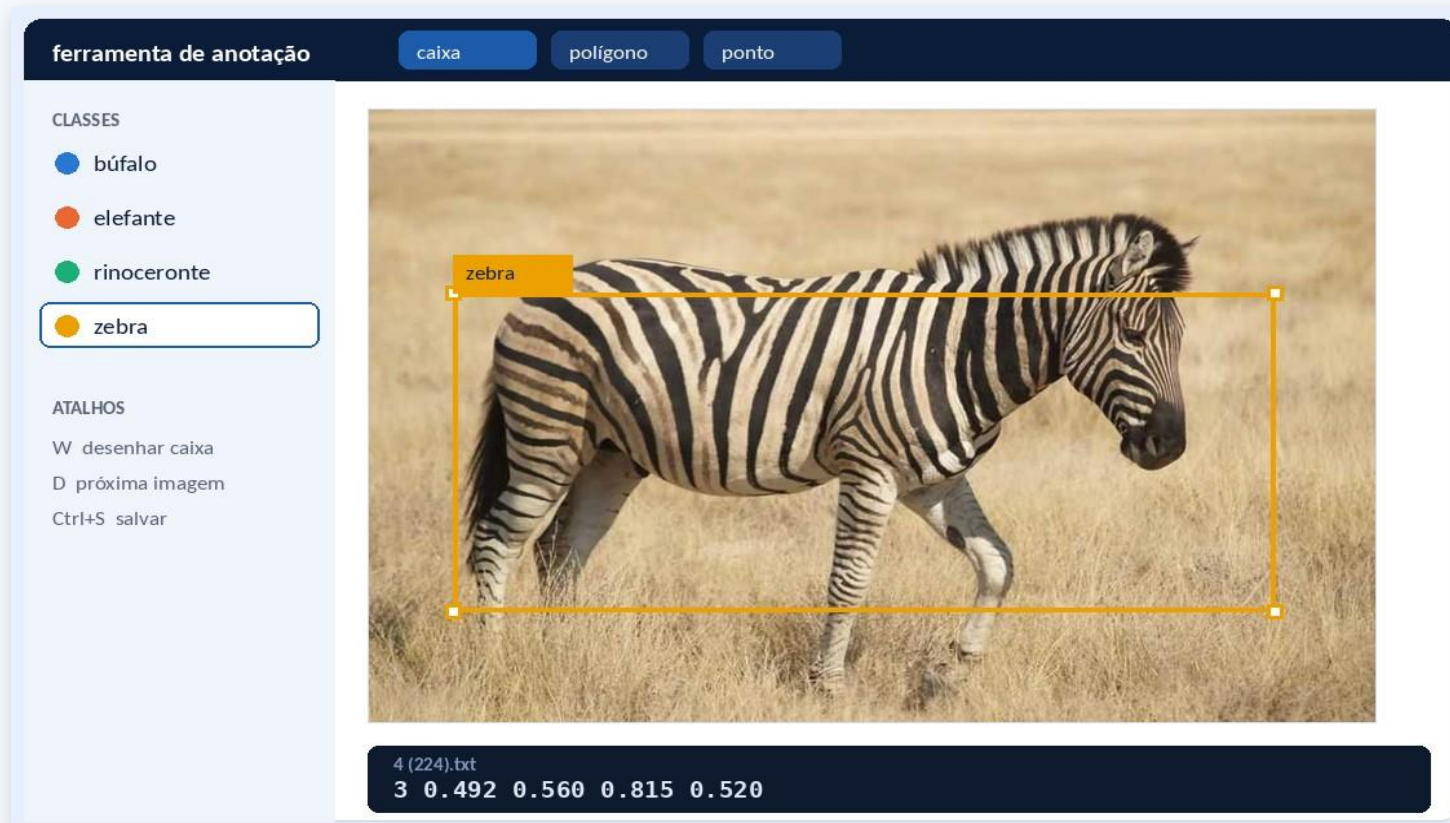


Ilustração de uma ferramenta de anotação de imagens: escolhe a classe, desenha a caixa, salva. Cada caixa vira uma linha de texto.

A ETAPA QUE NINGUÉM MOSTRA

Aprendizado supervisionado precisa de gabarito

sem rótulo não há o que comparar, e sem comparação não há perda nem treino

O rótulo depende da tarefa

classificação: a pasta ou uma coluna · detecção: caixa + classe · segmentação: o contorno

Quem anota precisa conhecer o problema

em laboratório e no campo, quem sabe diferenciar as classes é o especialista

É o maior custo do projeto

tempo, revisão e um guia escrito de como anotar os casos duvidosos

Com o que se anota, na prática

Tipo de dado	Ferramenta	Como é
Tabela (linhas e colunas)	planilha: Excel, Google Sheets, CSV	uma coluna a mais com a resposta: doente ou sadia, preço, classe
Imagem, caixas simples	LabelImg	programa leve de desktop; salva no formato da YOLO; não é mais mantido
Imagem e vídeo, em equipe	CVAT	aberto, roda no navegador, caixas, polígonos e pontos, revisão e pré-anotação
Vários tipos de dado	Label Studio	aberto, anota imagem, texto, áudio e séries temporais
Dataset + treino na nuvem	Roboflow e afins	serviço pago: anota, versiona e já treina, com plano gratuito limitado

- **Escreva o guia antes**

o que conta como objeto, o que fazer com o que está cortado ou escondido
- **Anote uma amostra primeiro**

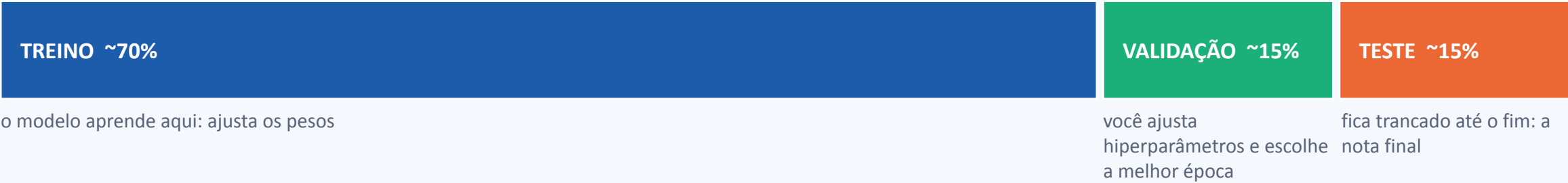
50 imagens já mostram se as classes fazem sentido
- **Reveja por amostragem**

duas pessoas anotando o mesmo lote revelam as regras que faltam
- **Deixe o modelo ajudar**

depois das primeiras centenas, ele pré-anota e você só corrige

LabelImg passou a fazer parte da comunidade do Label Studio e não recebe mais desenvolvimento ativo (github.com/HumanSignal/labelimg). CVAT: cvat.ai.

Treino, validação e teste: três conjuntos, três funções



	Treino	Validação	Teste
Exemplo 1 • MNIST	60.000	usa o teste*	10.000
Exemplo 2 • Animais	1.052	225	227

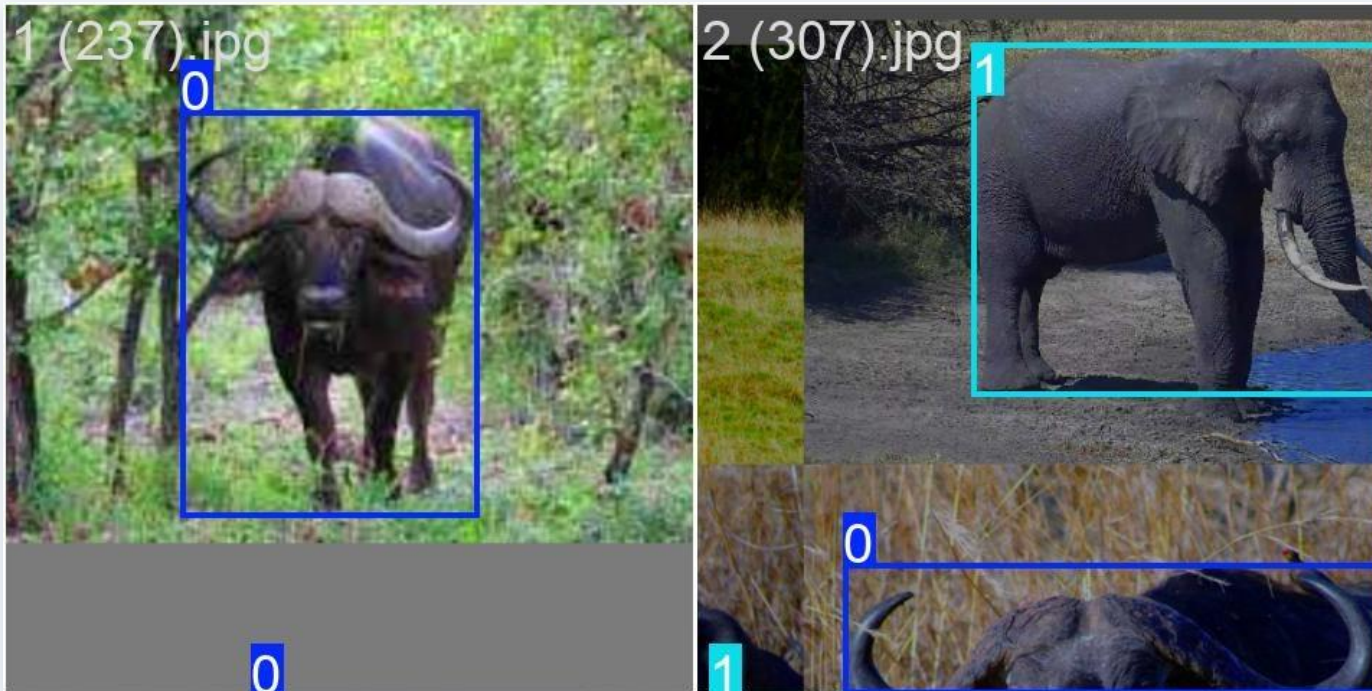
* O MNIST vem só com treino e teste; a YOLO usa o teste para validar durante o treino.

O ERRO QUE MAIS INFLA MÉTRICA

Vazamento de dados: fotos quase iguais (mesma planta, mesmo dia, mesmo paciente) em treino e teste. A nota sobe; o modelo não melhorou.

Divida por grupo (talhão, dia, animal), não por foto.

Aumento de dados: mais variedade sem fotografar de novo



Um lote real do treino do Exemplo 2: mosaico de 4 fotos, cortes, mudança de escala e de cor, gerados na hora pela YOLO.

O que a YOLO faz por padrão

- Mosaico de 4 imagens
- Espelhamento horizontal (50%)
- Variação de cor, brilho e saturação
- Escala e deslocamento

MAS CUIDADO

Espelhar uma zebra continua sendo zebra. Espelhar um “2” não é mais um “2”. No MNIST desligamos: `flipplr=0.0`.

BLOCO 7

Ferramentas

Quais bibliotecas fazem o trabalho pesado e por que escolhemos a YOLO para os exemplos.



Os dois frameworks que dominam o deep learning

TensorFlow

Google

- Nasceu como ferramenta interna do Google Brain, sucessor do sistema DistBelief
- Aberto ao público em 9 de novembro de 2015, com licença Apache 2.0
- Forte em produção e dispositivos: TF Lite (hoje LiteRT), TF.js; usa o Keras como API de alto nível

PyTorch

Meta (antigo Facebook)

- Criado no laboratório de IA do Facebook e lançado em 2016
- Estilo “Python puro”: você escreve e depura como um programa comum
- Desde 2022 é governado pela PyTorch Foundation (Linux Foundation); a Meta segue como grande mantenedora

A Ultralytics (YOLO) é construída sobre o PyTorch. Outras que você vai encontrar: Keras, JAX, scikit-learn (ML clássico) e Hugging Face (modelos prontos).

Por que escolhemos a YOLO (Ultralytics) para os exemplos

1

Uma biblioteca, várias tarefas

classificação, detecção, segmentação, pose, caixas giradas

2

Modelos pré-treinados

começa de uma rede que já sabe ver (ImageNet, COCO)

3

Código mínimo

carregar, treinar, avaliar e exportar: uma linha cada

4

Exporta para tudo

ONNX, TensorRT, OpenVINO, CoreML, TFLite...

5

Tempo real

“You Only Look Once”: detecta tudo numa única passada pela rede

```
from ultralytics import YOLO

modelo = YOLO("yolo26n.pt")
modelo.train(data="animais.yaml", epochs=50)
modelo.val(split="test")
modelo.export(format="onnx")
```

YOLO26 (jan/2026): versão atual. Detecção de ponta a ponta, sem a etapa extra que junta as caixas repetidas, e foco em rodar rápido na CPU e na borda.

Licença AGPL-3.0: livre para estudo e projetos abertos; uso comercial fechado exige licença paga.

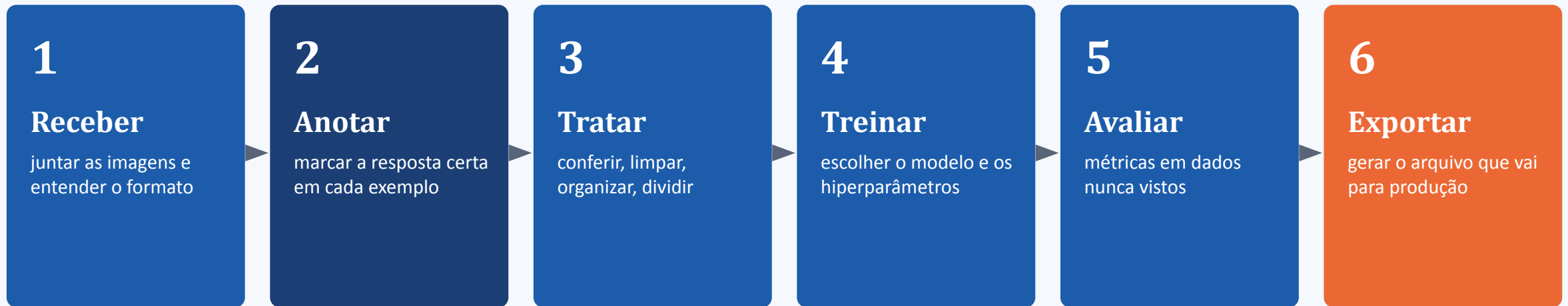
BLOCO 8

Mão na massa: do dataset ao modelo

Dois exemplos completos com YOLO, com o mínimo de código possível. Tudo o que aparece aqui rodou de verdade.



O fluxo é sempre o mesmo, em seis etapas



EXEMPLO 1 • CLASSIFICAÇÃO

MNIST em CSV → 70 mil PNGs em pastas → YOLO26n-cls → acurácia e análise de erros → .onnx

EXEMPLO 2 • DETECÇÃO

African Wildlife (imagens + .txt) → inspeção dos rótulos → YOLO26n → mAP no teste → .onnx

Nos nossos dois exemplos a etapa 2 já vinha pronta: alguém anotou antes, com as ferramentas que vimos no Bloco 6. No seu projeto, é ela que costuma tomar mais tempo. E depois da etapa 6 vem colocar o modelo para rodar, que é o Bloco 9.

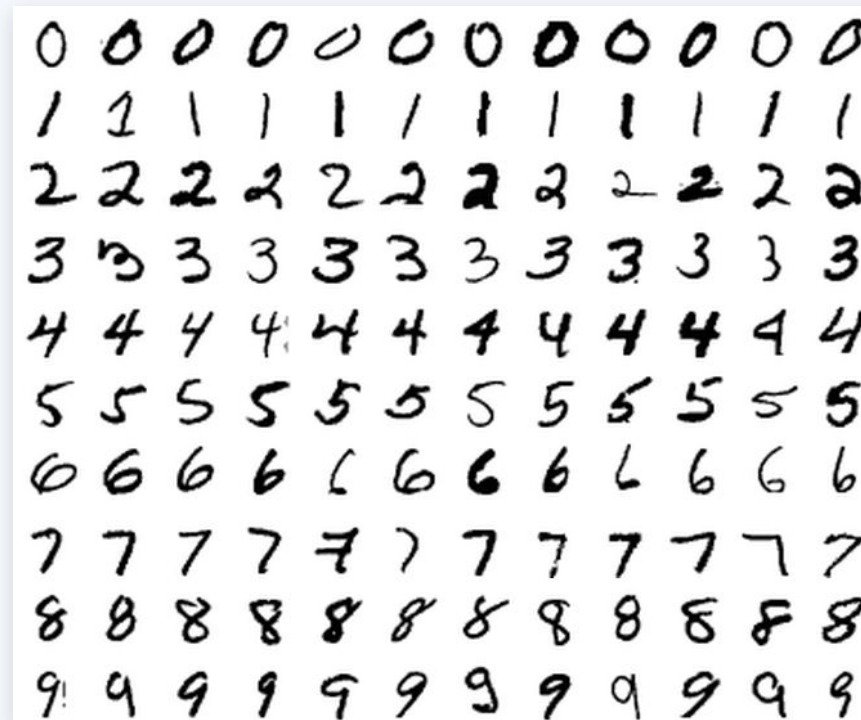
Transformar cada linha numa imagem, na pasta da sua classe

```
exemplo1_mnist.py
```

```
def tratar(arquivo_csv, destino):
    tabela = pd.read_csv(arquivo_csv, header=None).to_numpy()
    for i, linha in enumerate(tabela):
        rotulo, pixels = linha[0], linha[1:]
        pasta = Path("dataset_mnist", destino, str(rotulo))
        pasta.mkdir(parents=True, exist_ok=True)
        imagem = pixels.reshape(28, 28).astype(np.uint8)
        Image.fromarray(imagem).save(pasta / f"{i:05d}.png")
```

```
dataset_mnist/
  train/0/ ... 9/
  test/0/ ... 9/
```

É o formato que a YOLO espera para classificação:
uma pasta por classe.



A linha vira matriz 28×28 com reshape, a mesma ideia do “imagem é matriz”.

Poucas linhas para treinar, avaliar e exportar

```
exemplo1_mnist.py  
modelo = YOLO("yolo26n-cls.pt")          # pré-treinado  
modelo.train(data="dataset_mnist", epochs=5, imgsz=32,  
              batch=256, flip1r=0.0)  
metricas = modelo.val()  
modelo.export(format="onnx")
```

yolo26n-cls.pt

parte de uma rede que já aprendeu a ver no ImageNet (aprendizado por transferência)

epochs=5

MNIST é fácil: 5 passadas bastam

imgsz=32

as imagens têm 28×28; a rede pede múltiplos de 32

flip1r=0.0

sem espelhar: um “2” espelhado não é um “2”

NO NOTEBOOK DO PALESTRANTE

1.543.914

parâmetros no modelo

7 min

de treino na CPU (80 s por época)

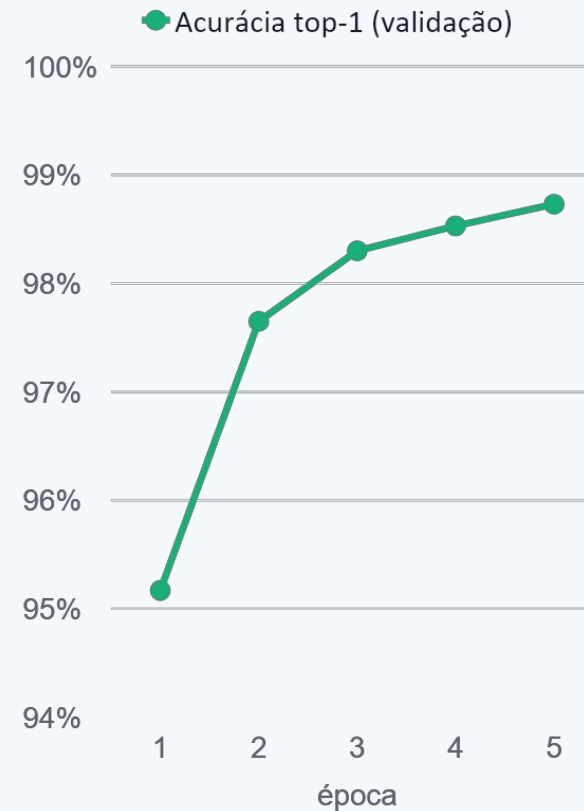
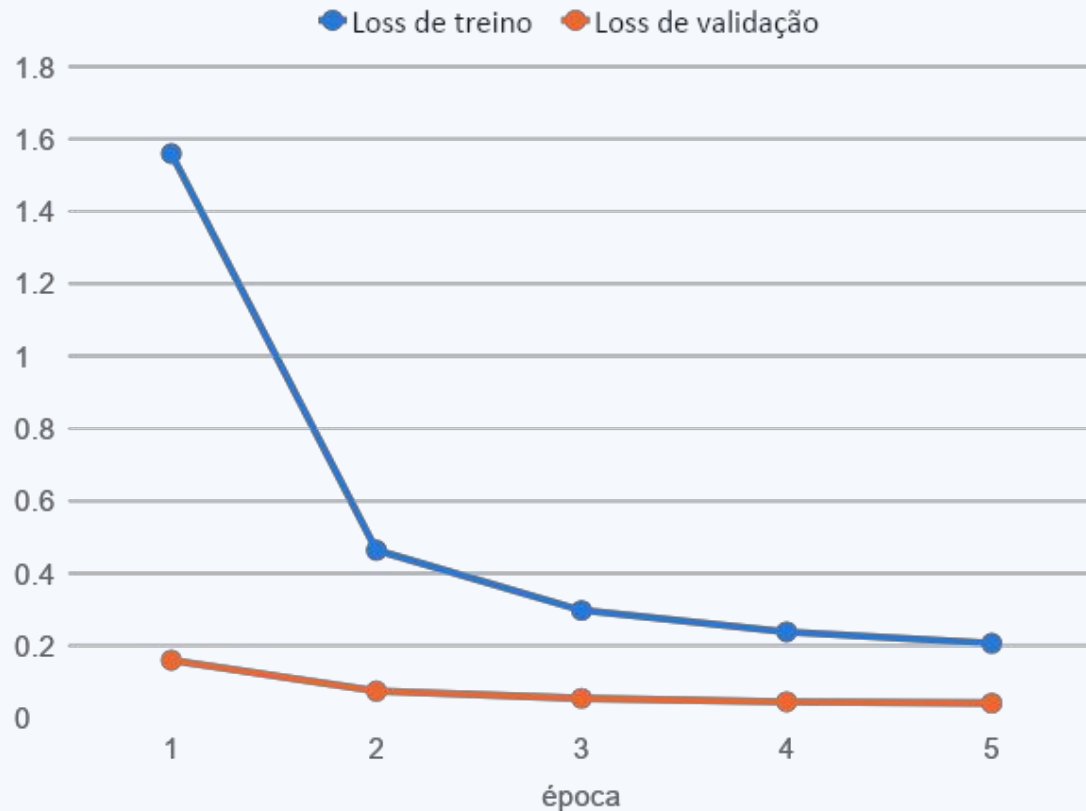
235

iterações por época

6,2 MB

arquivo .onnx exportado

O resultado, época por época



98,74%

de acerto nas 10.000 imagens de teste

126

erros no total

loss de treino > validação? O aumento de dados deixa o treino mais difícil de propósito.

Números reais do treino (results.csv). Depois de exportado, o .onnx foi testado de novo, fora da Ultralytics: mesmos 98,74%.

Olhar os erros ensina mais que olhar a média

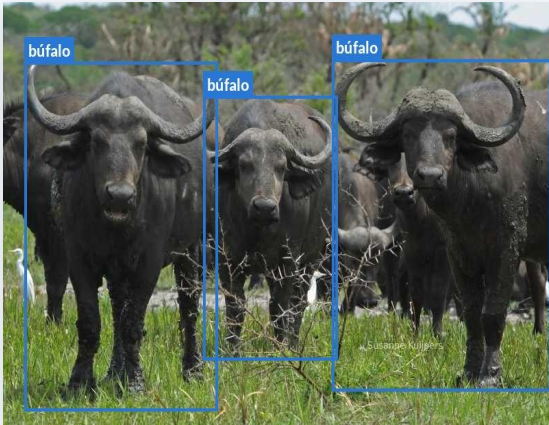


Alguns erros são até compreensíveis: um 4 que parece 9, um 7 com perna de 2. Outros revelam dígitos ambíguos ou mal escritos: há rótulos discutíveis até no MNIST.

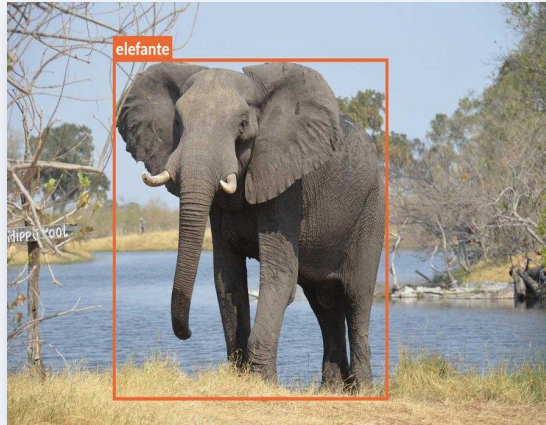
Erros reais do modelo do Exemplo 1 no conjunto de teste. “é 4, disse 9” = o rótulo diz 4, o modelo previu 9.

Próximo passo real: juntar mais exemplos dos pares que mais se confundem ($9 \leftrightarrow 7$, $4 \leftrightarrow 9$) e revisar rótulos suspeitos.

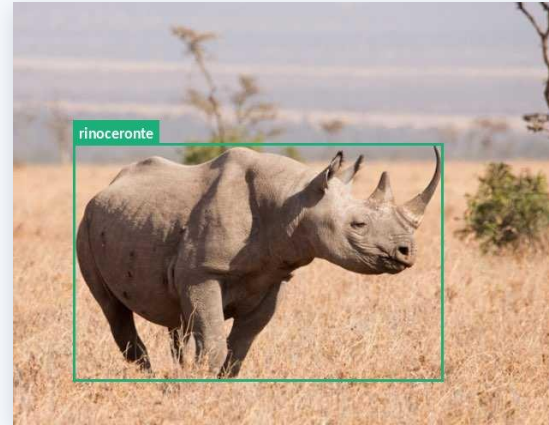
African Wildlife: 1.504 fotos, 4 animais



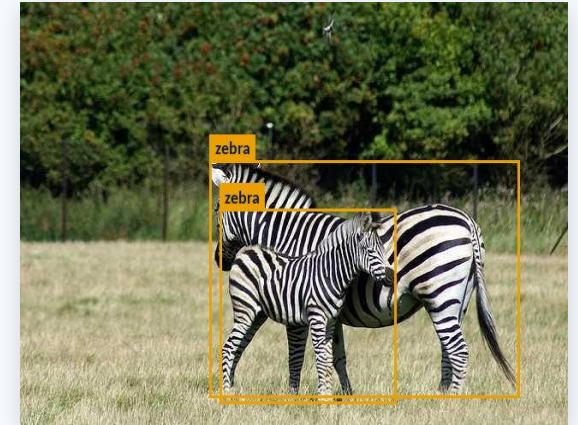
búfalo



elefante



rinoceronte



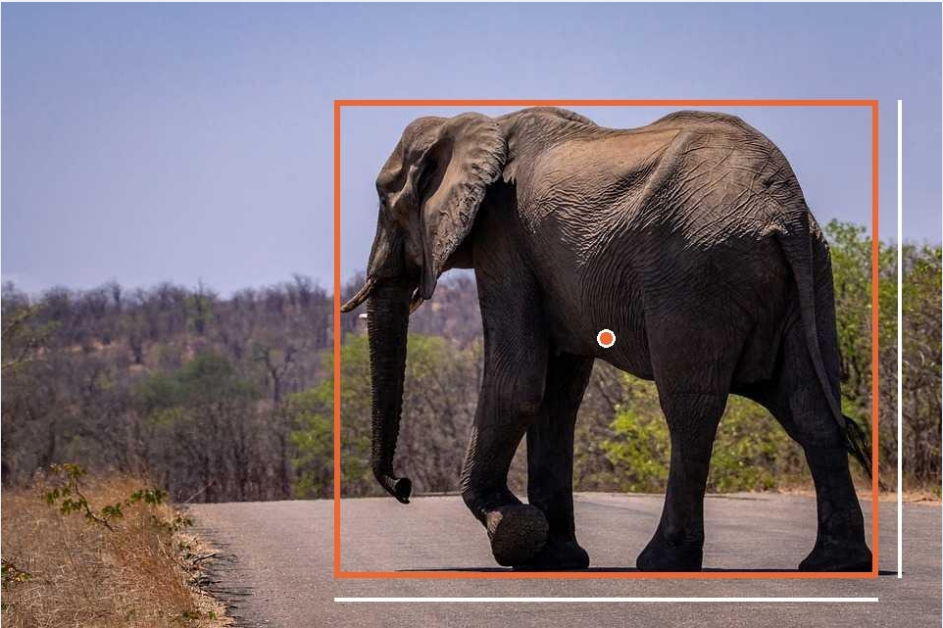
zebra

```
exemplo2_parte1_receber_tratar.py (resumido)
URL = ("https://github.com/ultralytics/assets/releases/"
       "download/v0.0.0/african-wildlife.zip")
download(URL, dir=".")
```

Dataset público do Kaggle, adaptado pela Ultralytics. As caixas coloridas acima são os rótulos reais.

```
african-wildlife/
  images/train val test
  labels/train val test
    L um .txt por imagem
```

Entender o rótulo e inspecionar antes de treinar



1 0.643 0.539 0.577 0.761

classe · centro x · centro y · largura · altura, divididos pelo tamanho da foto (0 a 1)

	Imagens	Caixas	Sem rótulo	Inválidas
Treino	1.052	1.931	0	0
Validação	225	379	0	0
Teste	227	375	0	0

Caixas por classe no treino



Tudo limpo: nenhuma imagem sem rótulo, nenhuma caixa fora da foto. Mas as fotos têm 782 tamanhos diferentes, e a YOLO redimensiona para 640 sem distorcer.

O mesmo padrão de código, agora para detecção

```
exemplo2_parte2_treinar.py  
  
modelo = YOLO("yolo26n.pt")          # pré-treinado no COCO  
modelo.train(data="animais.yaml", epochs=50, imgsz=640)  
  
metricas = modelo.val(data="animais.yaml", split="test")  
modelo.predict("african-wildlife/images/test", save=True)  
modelo.export(format="onnx")
```

```
animais.yaml  
  
path: african-wildlife  
train: images/train  val: images/val  test: images/test  
names: {0: búfalo, 1: elefante, 2: rinoceronte, 3: zebra}
```

O QUE ACONTECEU NO TREINO

2.505.360

parâmetros (260 camadas)

50 épocas

2 h 26 min na CPU do notebook

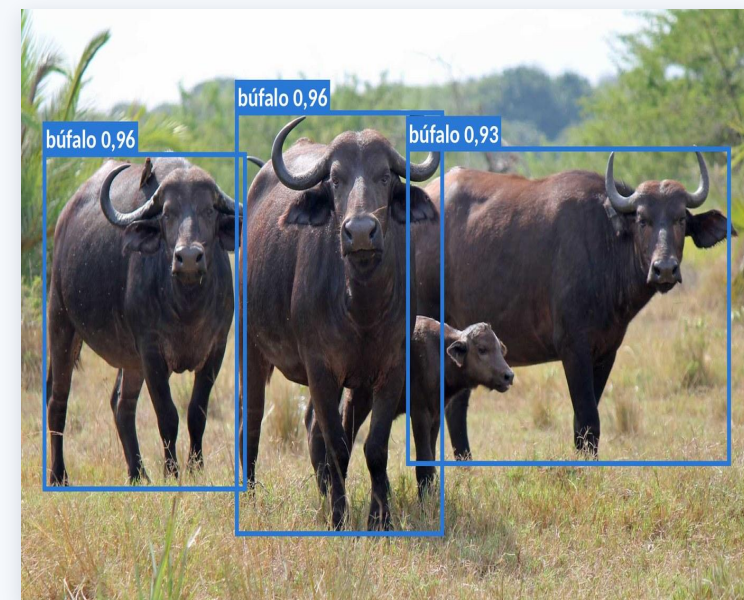
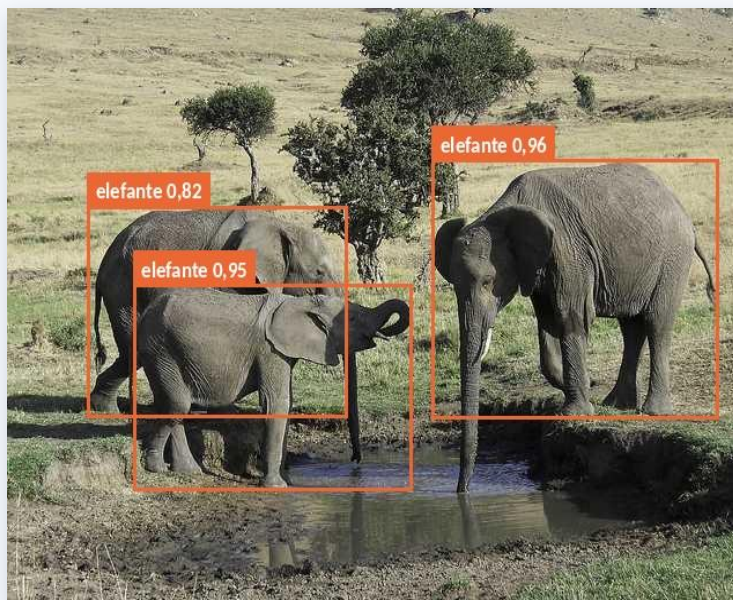
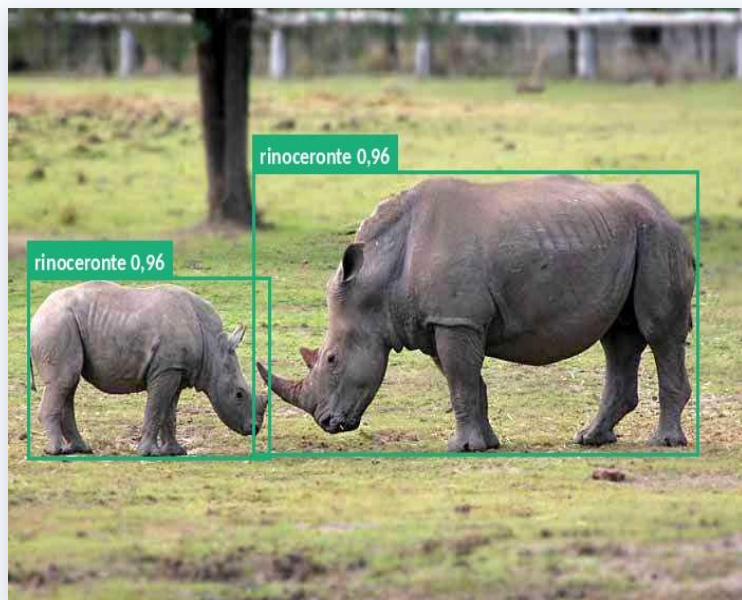
66

iterações por época

612 de 708

blocos de pesos reaproveitados do COCO

O detector em fotos que ele nunca viu



0,976

mAP50

0,850

mAP50-95

95,8%

Precisão

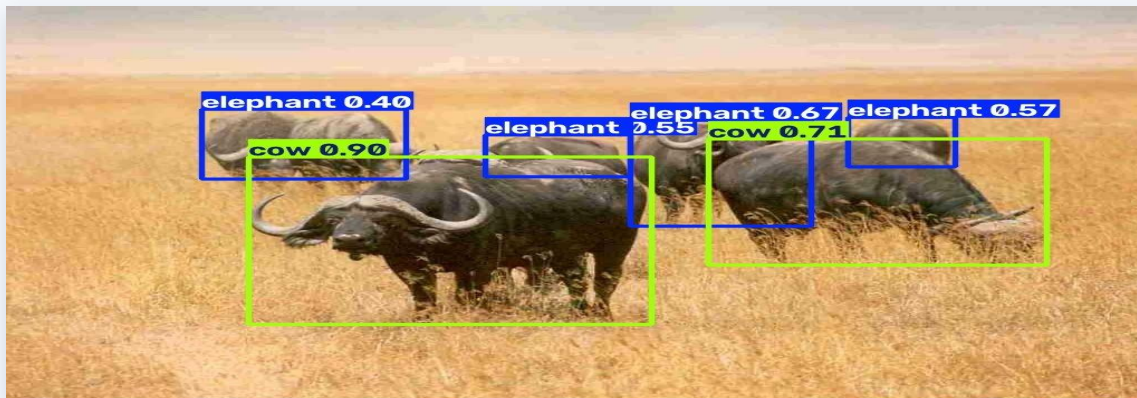
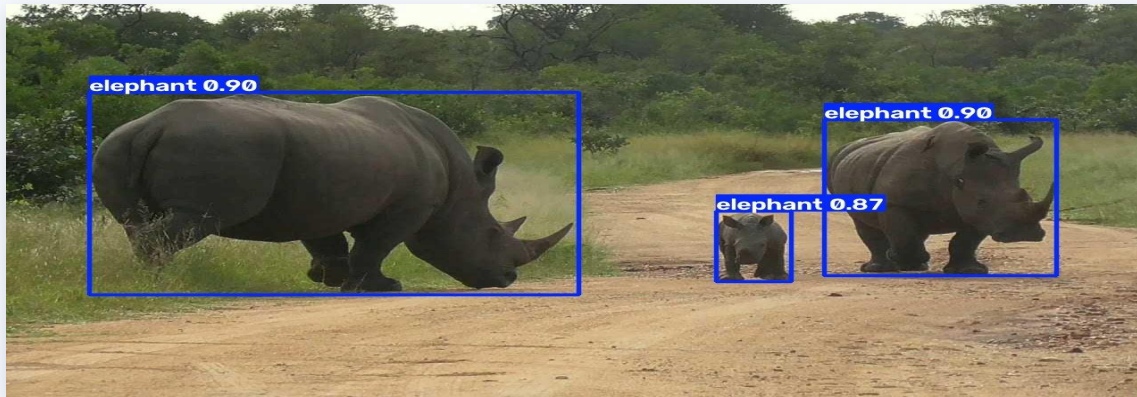
94,8%

Recall

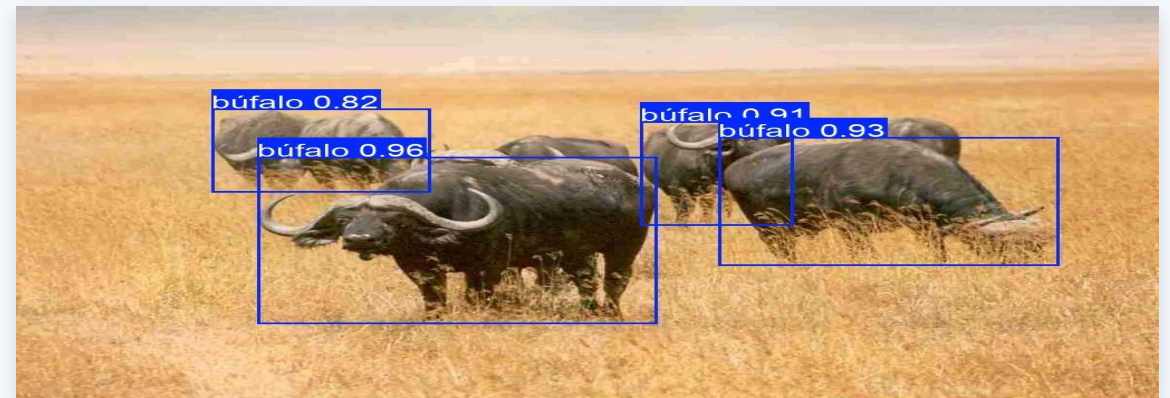
Predições reais do modelo treinado nas imagens de teste (227 fotos). Métricas calculadas nesse mesmo conjunto.

Antes e depois do ajuste fino

MODELO PRÉ-TREINADO (COCO, 80 CLASSES)



DEPOIS DE TREINAR COM AS 1.052 FOTOS



O COCO não tem rinoceronte nem búfalo: o modelo original chamou os rinocerontes de “elephant” e os búfalos de “cow” ou “elephant”. Depois do ajuste fino, acerta os dois.

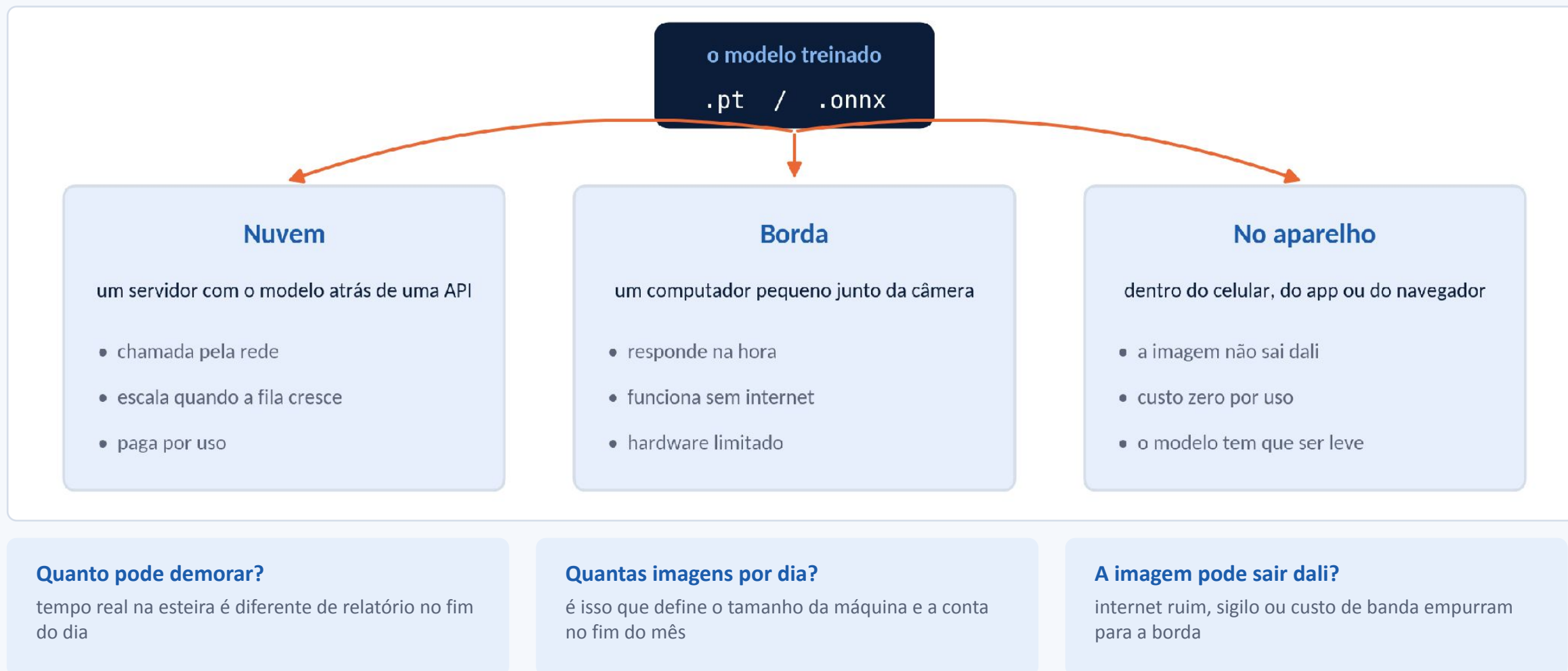
BLOCO 9

Colocar o modelo para rodar

O arquivo treinado não resolve nada sozinho: alguém precisa chamá-lo, medir e manter.



Treinou, exportou. E agora, onde isso roda?



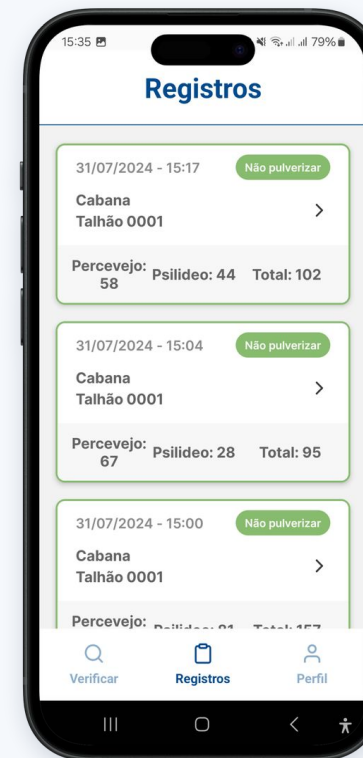
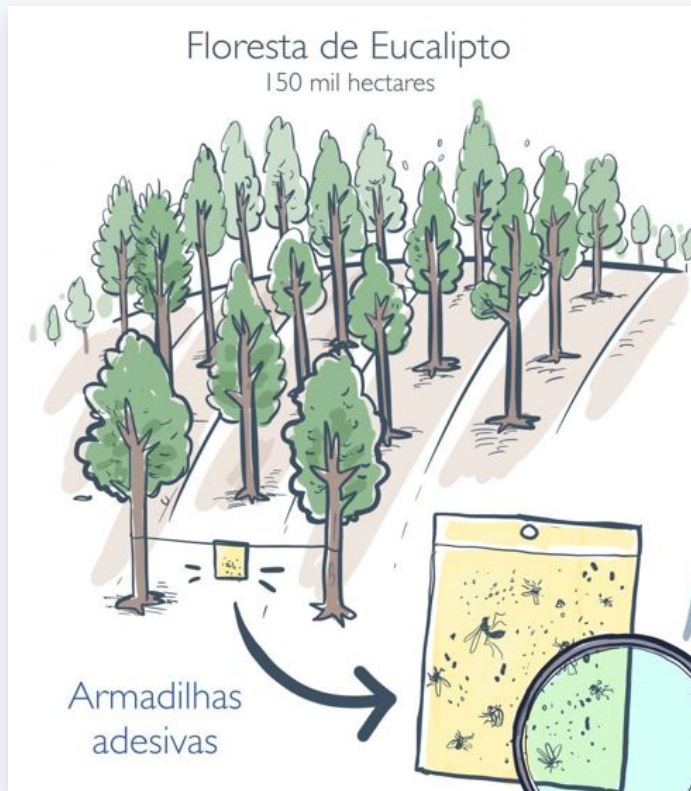
10

BLOCO 10

IA na prática

Produtos reais construídos com as mesmas ideias: detectar, segmentar, contar e integrar num sistema.

Contagem de pragas em floresta plantada



PROBLEMA

Contagem manual de psilídeo e percevejo-bronzeado em armadilhas, anotada em papel.

150 mil

hectares de eucalipto no escopo

45 dias

para o MVP: app + nuvem + dashboard

91%

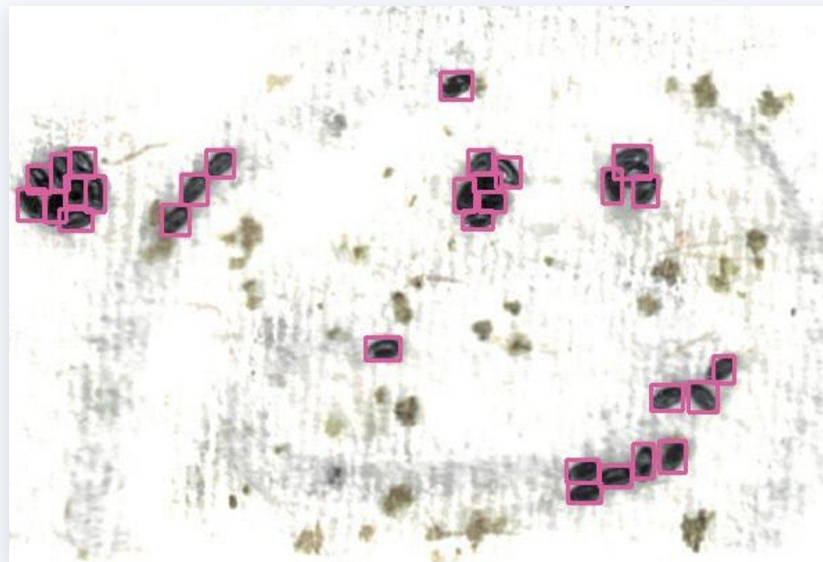
de precisão inicial do modelo

A armadilha adesiva é fotografada pelo celular no talhão; o modelo detecta e conta; o registro sobe georreferenciado para o dashboard.

Contagem de ovos em laboratório



ANTES: contagem inteiramente manual, sob microscópio.



DEPOIS: cada ovo detectado e contado em segundos.

98%

de precisão

45 dias

para o MVP em nuvem, com dashboard

- Imagens processadas na nuvem, ou na borda, se o cliente tiver hardware
- Contagem por fita, frente e verso, mais o total consolidado
- Imagens originais e processadas lado a lado, para auditoria

Contagem de toletes de cana no plantio



Máscara roxa

segmentação por instância: um contorno por tolete

Linha vermelha

a linha virtual que decide quando contar

CONTAGEM

sobe a cada tolete que cruza a linha

Imagens reais do sistema rodando em campo. Status: MVP em validação.

PROBLEMA

Aferir a densidade de plantio a olho e anotar à mão, dentro da plantadeira em movimento.

3 segundos

para a contagem por IA, direto do celular

- Segmentação de cada tolete + linha de contagem
- Histórico de registros em campo
- Dashboard web para gestão

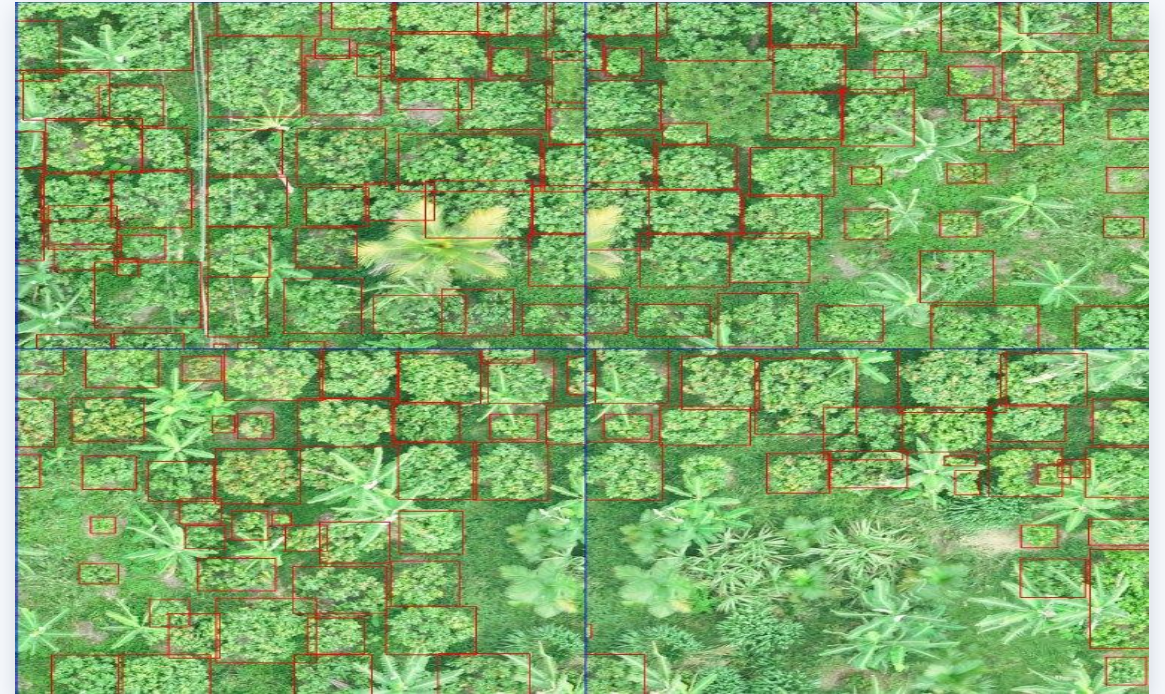
Imagem aérea e ortomosaico



Estimativa de produção de abacaxi

Ortomosaico gerado por imagens de drone.

LIMA, M. A. et al. XIX Workshop de Visão Computacional, 2024.



Detecção de árvores de cacau

Ortomosaico de satélite, com YOLOv8 + SAHI (fatiamento da imagem).

MARTINS, T. M. et al. XIX Workshop de Visão Computacional, 2024.

Classificar, inspecionar e contar com câmera fixa



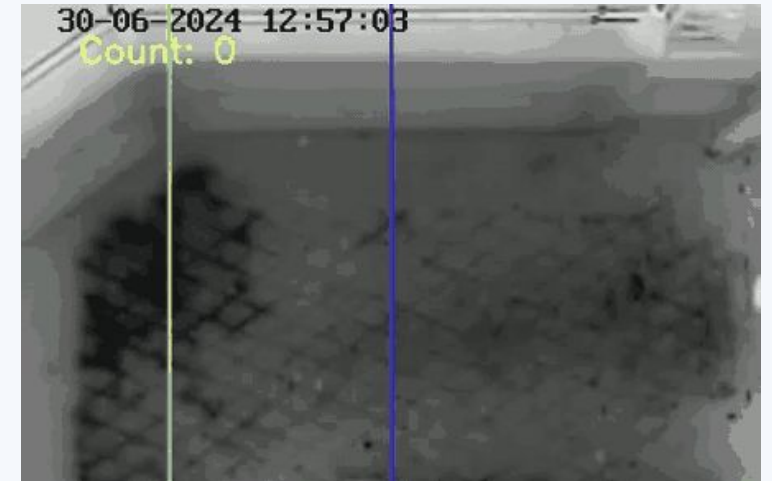
Classificação de grãos de café

o sistema segmenta cada grão e separa por tipo e defeito



Inspeção de frutas em esteira

segmentação em fluxo contínuo, fruta a fruta



Contagem de gado no desembarcador

cada animal ganha um número e é contado ao cruzar a linha

Com a câmera parada, a luz e o enquadramento deixam de variar. O que muda é o que passa na frente, e a resposta precisa acompanhar esse ritmo.

Seis ideias para levar para casa

1 Aprender é ajustar pesos com dados o modelo descobre as regras que não conseguiríamos escrever

2 Quase tudo é classificação, regressão ou geração e detecção é classificação + regressão

3 A CNN preserva o “onde” ela diz o que é e também onde está; a rede densa perde essa informação

4 Dado bom vale mais que modelo novo anotação cuidadosa, divisão honesta, variedade real

5 Modelo pronto só conhece o que já viu as classes do seu problema vêm do ajuste fino com os seus dados

6 Do dataset ao modelo rodando receber, anotar, tratar, treinar, avaliar, exportar e colocar para rodar

Obrigado!

Perguntas?

Matheus Arroyo de Lima

Engenheiro de Visão Computacional e Inteligência Artificial
Mestrando em Computação Aplicada na FFCLRP/USP

