Processamento de Linguagem Natural e Imagens



Aula 9

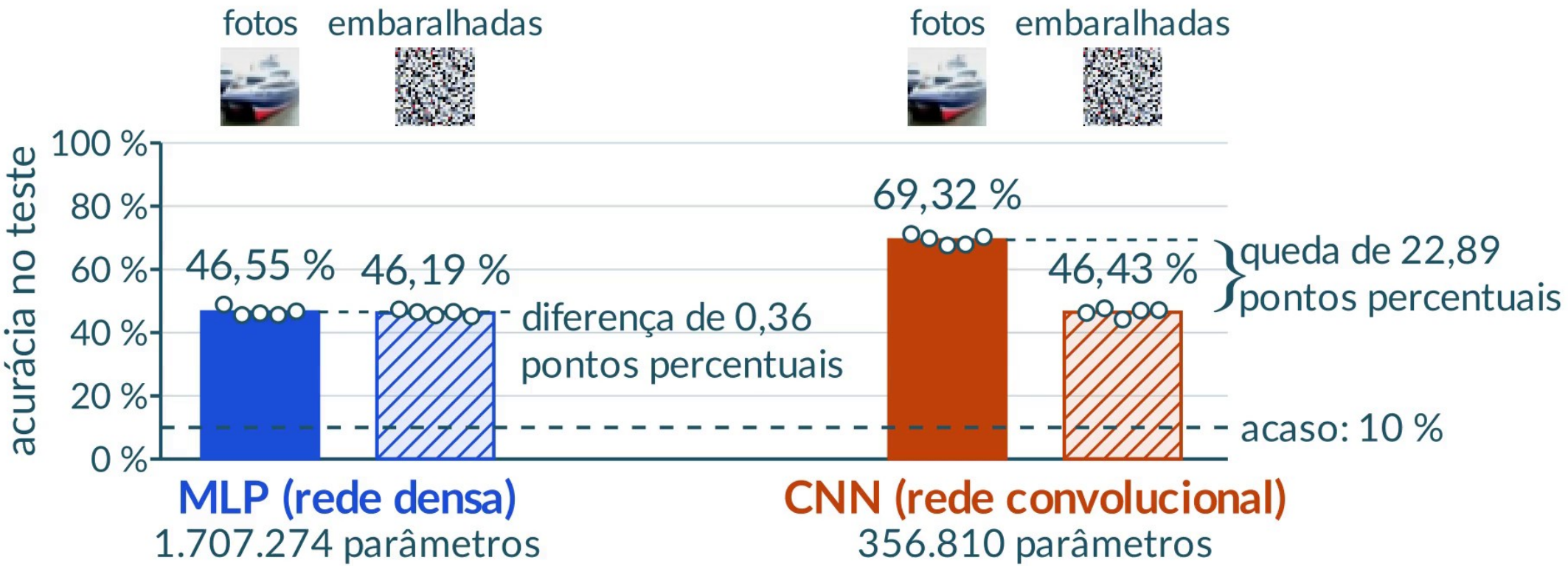
James Moraes de Almeida

Conteúdo do curso

- Qual o plano?
  - Processamento de imagens.
    - Aula 7 — imagem digital, filtros e Fourier.
    - Aula 8 — segmentação e quantificação de microscopia.
    - Aula 9 — redes neurais convolucionais.
- Material principal: Python for Microscopists (DigitalSreeni).

As regras do enigma

- As fotos: o CIFAR-10, 60.000 fotos de 32 × 32 pixels em 10 classes.
  - Treino: 45.000 fotos; ajustam os pesos.
  - Validação: 5.000, tiradas do treino; escolhem a época.
  - Teste: 10.000; a rede nunca as vê, só medem no fim.
- Cada rede é treinada do zero duas vezes: nas fotos e nas embaralhadas.
- Acurácia: a fração das 10.000 fotos de teste com a classe certa.
  - Ao acaso, 10 %.



Por que só a CNN perde com o embaralhamento?

A rede densa (MLP)

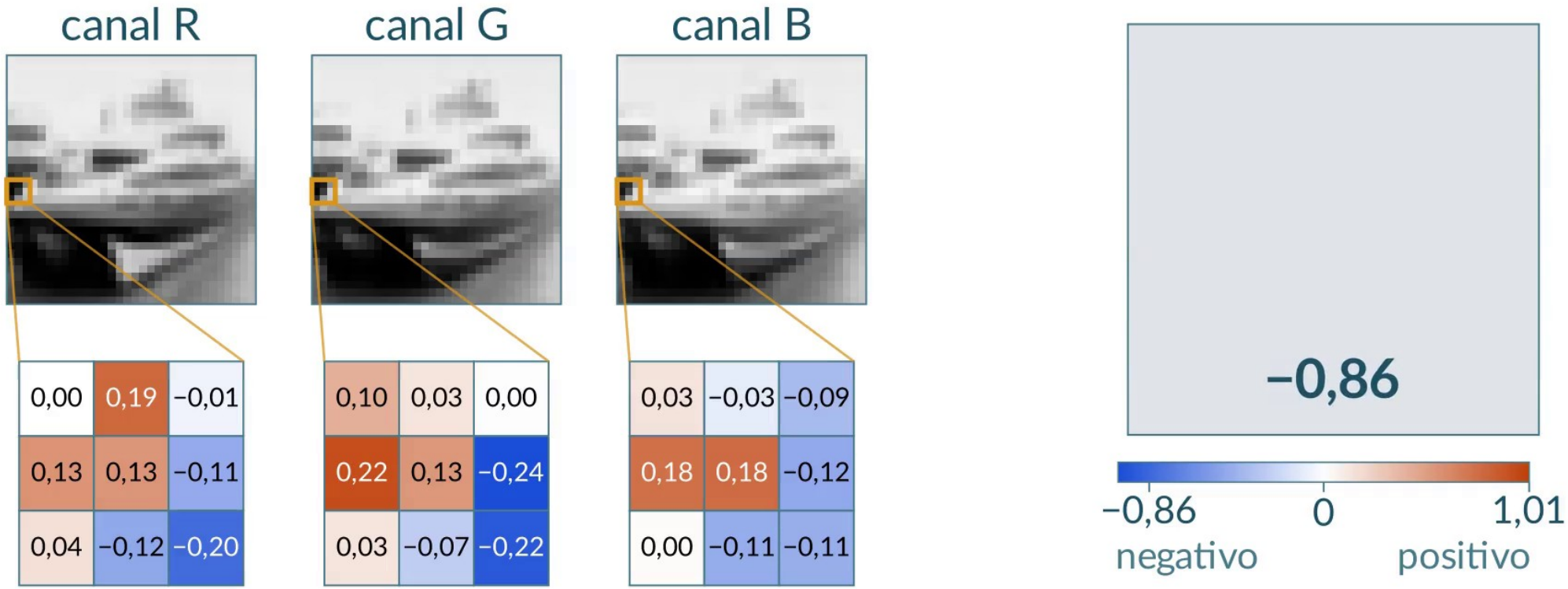
- Cada neurônio soma as entradas, cada uma com o seu peso, mais um viés.
- Camadas de 512, 256 e 10 neurônios: um número por classe na saída.
- Parâmetros = pesos + vieses.



Um neurônio da 1ª camada tem 3.072 pesos, um para cada número.

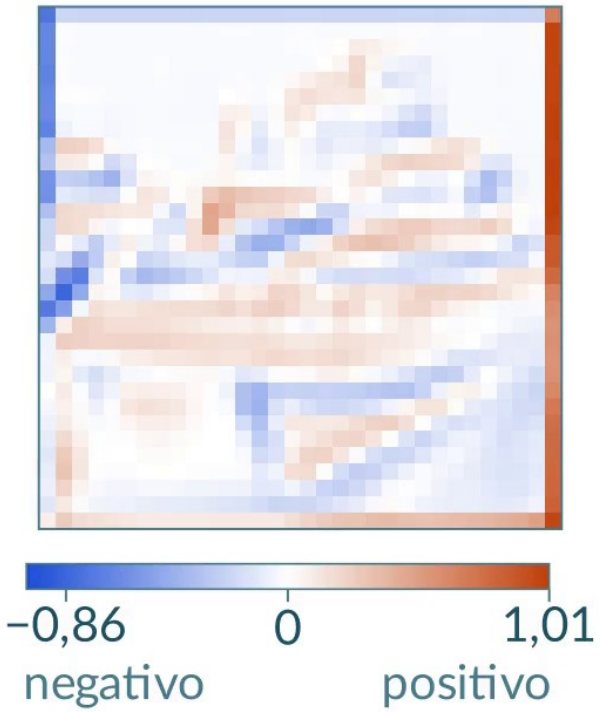
Por que outra rede?

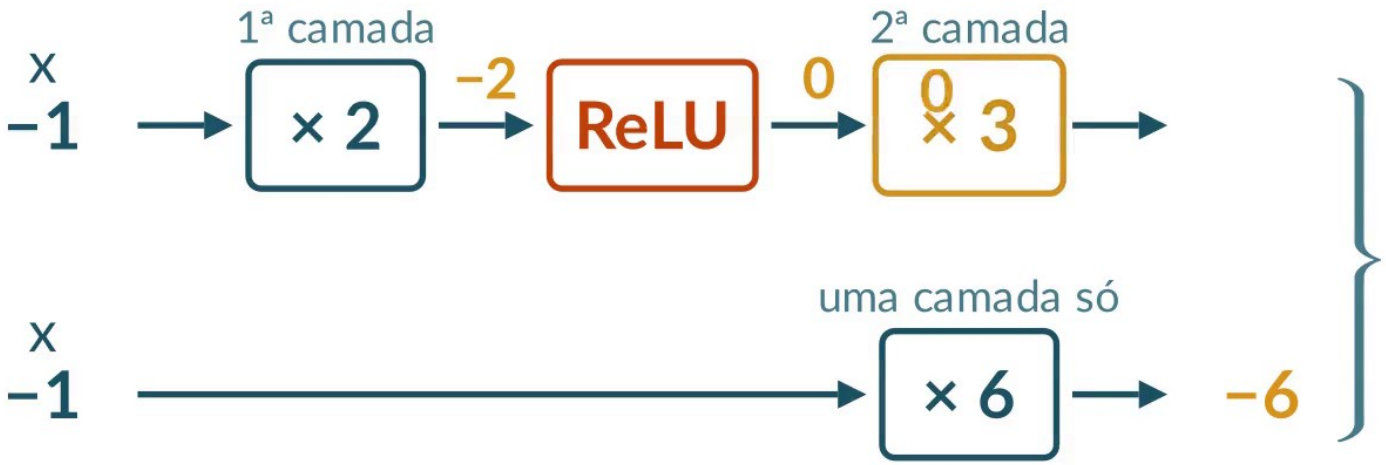
- A MLP não sabe quem é vizinho de quem.
- E paga um peso por número: 1.573.376 parâmetros só na 1ª camada.
- A ideia: cada saída olha só uma janela 3 × 3 da foto.
- E os mesmos pesos valem na foto toda.
- É o kernel da Aula 7, mas agora os seus valores saem do treino.



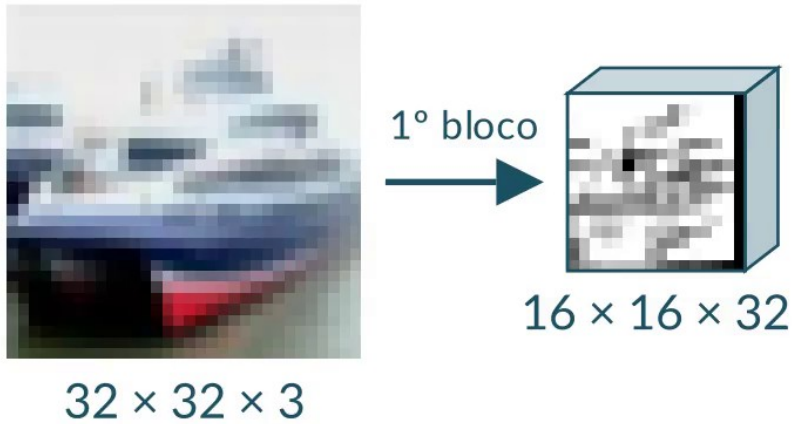
Em cada posição, um número: forma-se um mapa de 32 × 32 valores.




−0,86
0
1,01
negativo
positivo



Com a ReLU no meio, não: para x negativo, sai 0.


1° bloco
16 × 16 × 32
32 × 32 × 3

A CNN da aula em Keras

- 1ª camada: na MLP, 1.573.376 parâmetros; na CNN, 896:
  - 32 kernels de 3 × 3 × 3 pesos, mais 32 vieses.
- 73,5 % dos parâmetros estão na 1ª camada densa.
- No Keras, um array desses se chama tensor (daí o nome TensorFlow),
  - mas não é o tensor da matemática.

```
modelo = keras.Sequential([
    keras.Input(shape=(32, 32, 3)),
    layers.Conv2D(32, 3, padding="same"),
    layers.ReLU(), layers.MaxPooling2D(2),
    layers.Conv2D(64, 3, padding="same"),
    layers.ReLU(), layers.MaxPooling2D(2),
    layers.Conv2D(128, 3, padding="same"),
    layers.ReLU(), layers.MaxPooling2D(2),
    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dense(10, activation="softmax"),
])
```

	parâmetros
	896
	18.496
	73.856
	262.272
	1.290
total	356.810

Softmax e perda

- Perda 0: probabilidade 1 na classe certa.
- Chute, 0,1 em cada classe: perda −ln 0,1 = 2,303. O treino começa aí.
- O navio: p(navio) = 0,98, perda 0,02.
- Caminhão lido como automóvel: p = 0,22, perda 1,51: abaixo do chute.
- O treino baixa a perda média.

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^{10} e^{z_j}}$$

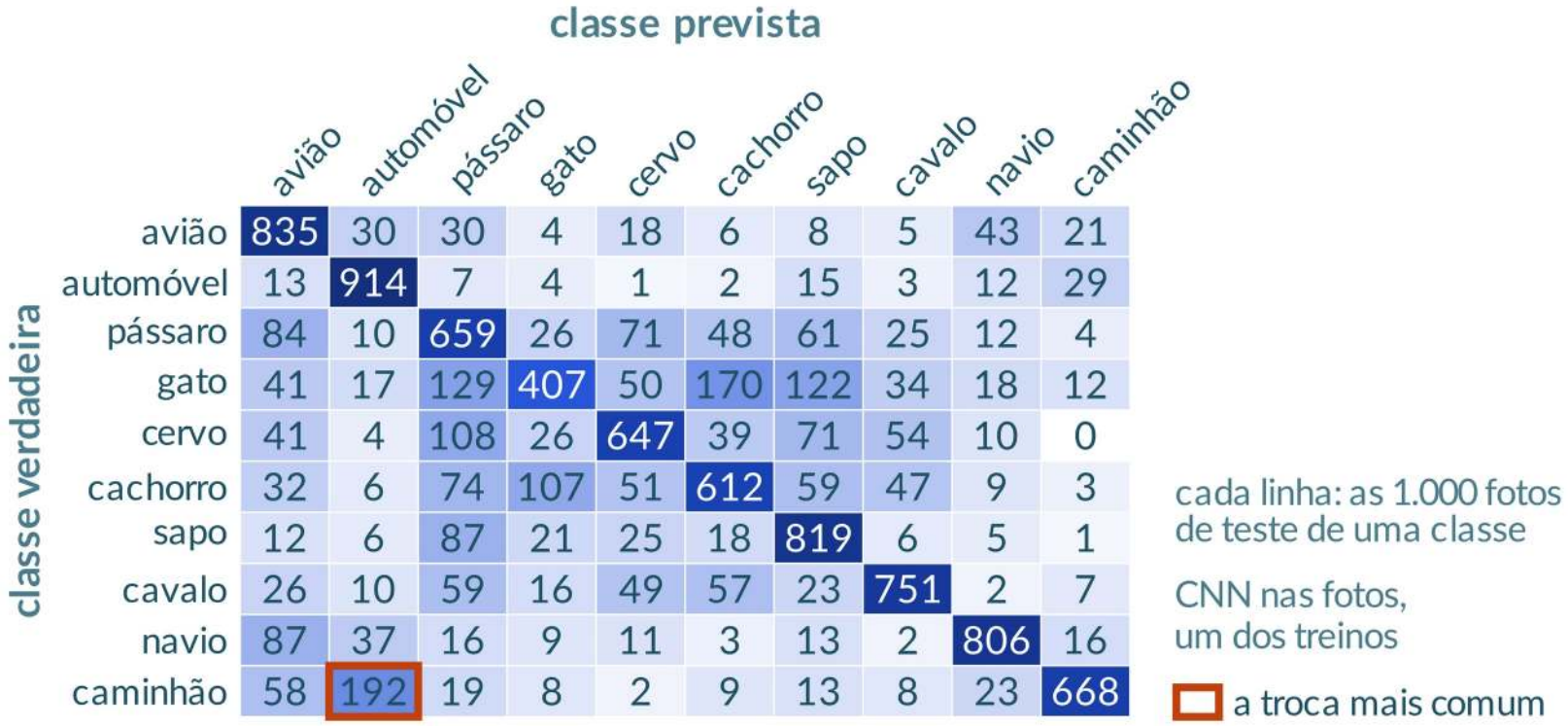
p_k: a probabilidade da classe k
z_k: o número da saída k, antes da softmax
j: percorre as 10 classes

$$L = -\ln p_{\text{classe certa}}$$

L: a perda de uma foto
$p_{\text{classe certa}}$: o que a rede dá à classe certa

A matriz de confusão

- Na diagonal, os acertos; fora dela, as trocas.
- Mais acertos: automóvel (91,4 %); menos: gato (40,7 %).
- A troca mais comum: caminhão lido como automóvel, 192 de 1.000.



Como a rede aprende

- A cada lote de 64 fotos, a rede calcula a perda.
- Os pesos passam por todas as camadas até p: a perda depende deles.
- A derivada da perda diz para que lado mexer cada peso.
- Cada peso dá um passo morro abaixo; o Keras calcula as derivadas.
- Cada lote é um passo; uma época são 704 passos.

$$L(w_1, \ldots, w_N) = -\frac{1}{B} \sum_{b=1}^{B} \ln p_{y_b}$$

L: a perda média do lote
B: fotos por lote
y_b: a classe certa da foto b
p_{y_b}: a probabilidade que a rede dá a ela; depende de todos os w_i

$$w_{i,\text{novo}} = w_i - \eta \frac{\partial L}{\partial w_i}$$

w_i: o parâmetro i (peso ou viés), antes do passo
$w_{i,\text{novo}}$: o mesmo, depois do passo
N: os parâmetros da rede (356.810 na CNN)
i: de 1 a N; todos dão o passo juntos
η: taxa de aprendizado, o tamanho do passo
$\partial L / \partial w_i$: a inclinação da perda

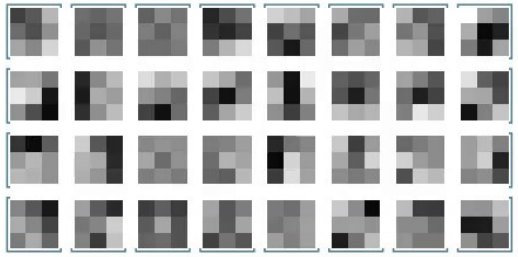
os 32 kernels da 1ª convolução

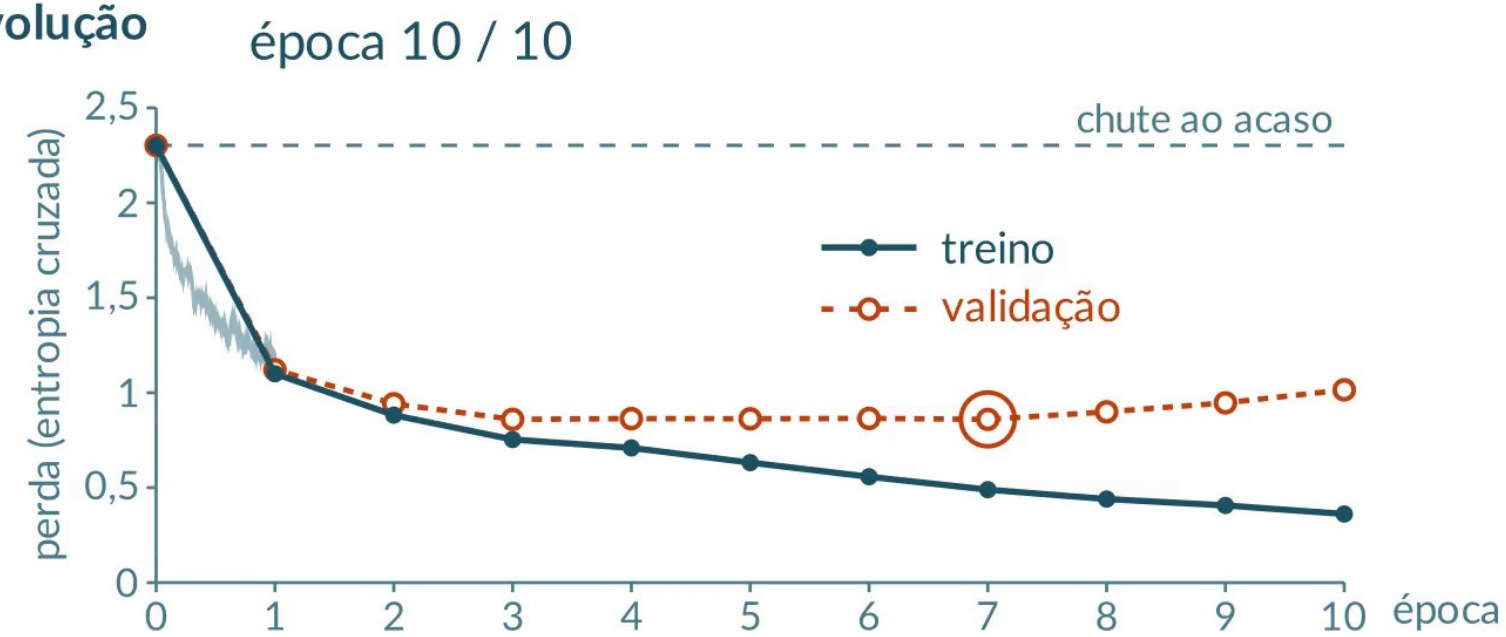
pesos no fim da época 10

em cores (R, G, B)

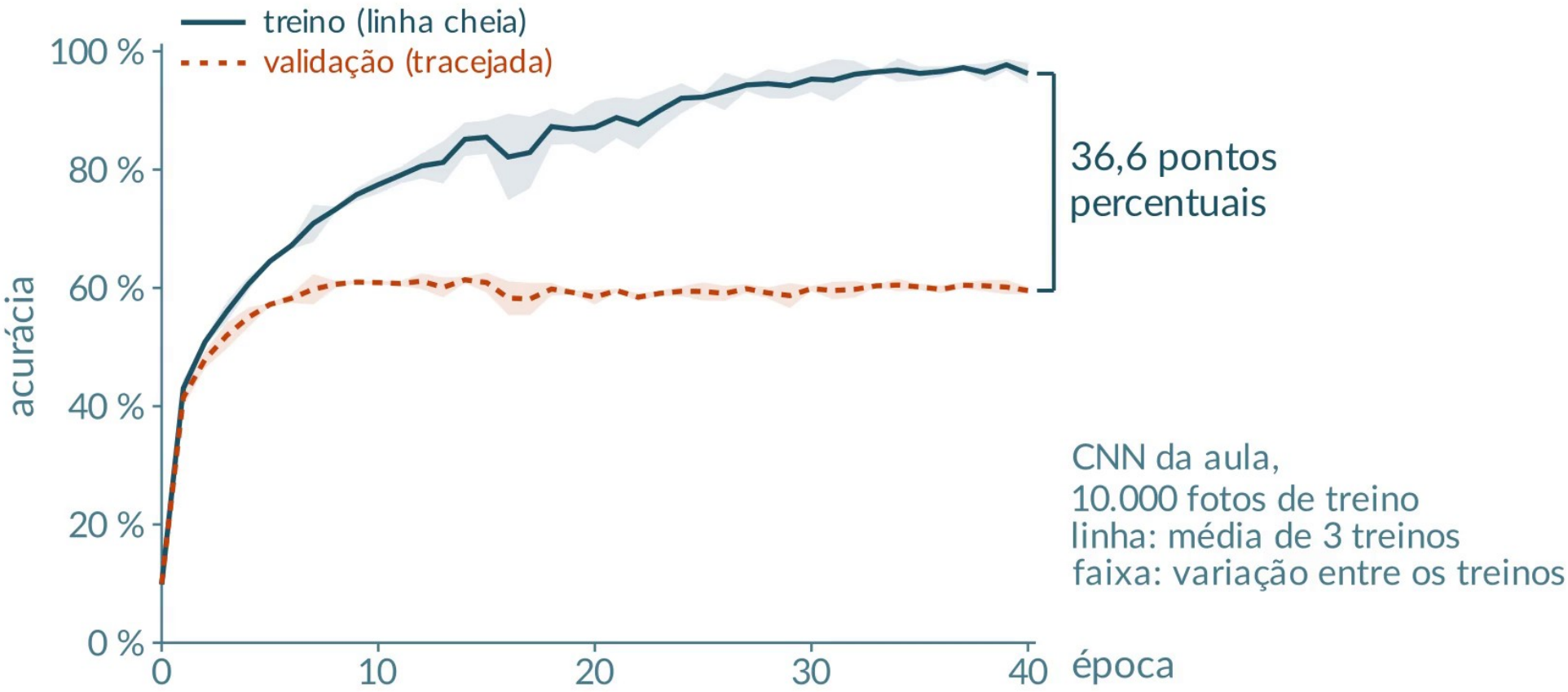


em tons de cinza





Época a época, a perda de treino cai; a de validação sobe após a época 7.



Sobreajuste: a rede acerta muito mais as fotos do treino que as novas.

Regularização no Keras

- O aumento de dados vira camadas do modelo, ativas só no treino.
- EarlyStopping: para quando a validação para de melhorar.
- ModelCheckpoint: guarda os pesos da melhor época.
- Os dois olham a VALIDAÇÃO; o teste fica para a medida final.

Aumento de dados, só no treino

```
modelo = keras.Sequential([
    keras.Input(shape=(32, 32, 3)),
    layers.RandomFlip("horizontal"),
    layers.RandomTranslation(0.1, 0.1,
        fill_mode="reflect"),
    # e as camadas da CNN da aula
])
```

No treino: validação e callbacks

```
modelo.fit(x_treino, y_treino, epochs=40,
    validation_data=(x_val, y_val),
    callbacks=[parada, melhor])
```

Parar quando a validação para de melhorar

```
parada = keras.callbacks.EarlyStopping(
    monitor="val_loss", patience=5,
    restore_best_weights=True)
```

Guardar o melhor modelo, pela validação

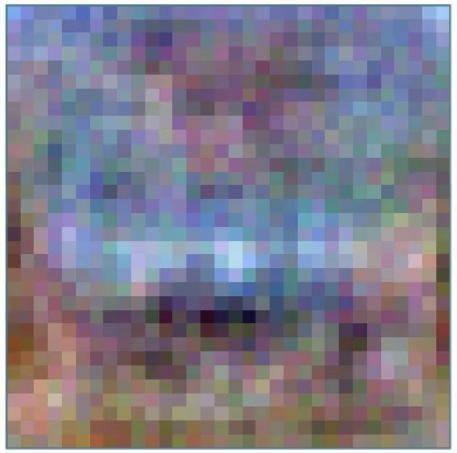
```
melhor = keras.callbacks.ModelCheckpoint(
    "melhor.keras", monitor="val_loss",
    save_best_only=True)
```

foto (32 × 32 px)



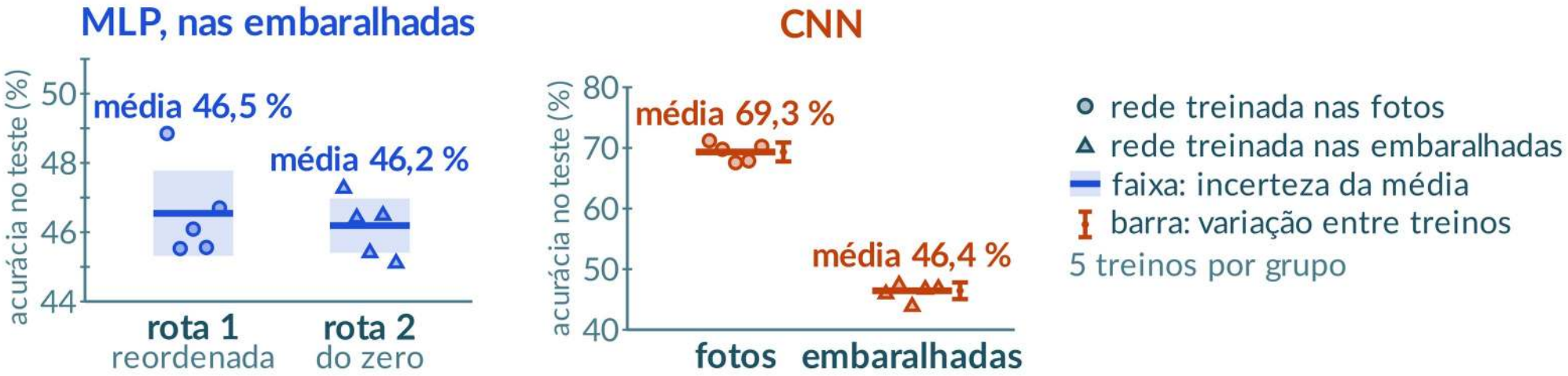
×

pesos de um neurônio



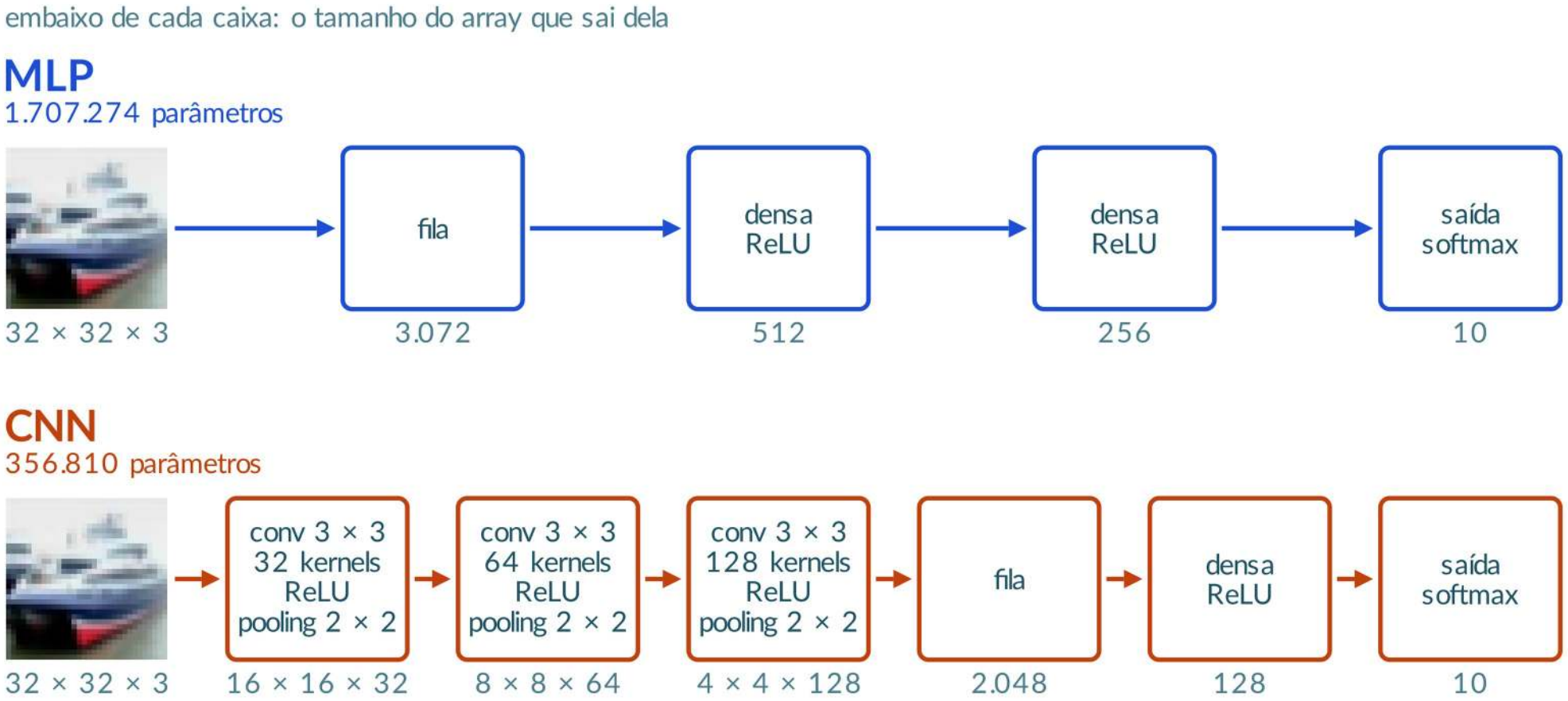
Duas rotas para o mesmo número

- Rota 1, sem treino novo: a MLP das fotos, pesos reordenados: 46,55 %.
- Ela dá, foto a foto, as mesmas 10.000 respostas que dava nas fotos normais.
- Rota 2, treinando do zero nas embaralhadas: 46,19 %.
- A diferença, 0,36 ponto, cabe na variação entre treinos (até 1,46 ponto).
- CNN: de 69,32 % para 46,43 %, uma queda de 22,89 pontos.



As duas redes, inteiras

- As duas recebem a mesma foto e dão as mesmas 10 probabilidades.
- Na CNN, cada bloco reduz o lado à metade e aumenta o número de mapas.



Vamos ao Python!

- O notebook da aula está no Teams da disciplina: Aula9_cnn.ipynb.
  - Modo rápido (10.000 fotos, 5 épocas) para o laptop; completo no Colab.
  - Seu número pode diferir um pouco do slide: cada treino dá um resultado.
- Ambiente img_proc da Aula 7, com o TensorFlow:
  - pip install tensorflow==2.21.0
- O CIFAR-10 (163 MB) vem pelo Teams; a seção 1 do notebook o instala.
- Apêndice opcional: uma rede pré-treinada no ImageNet (MobileNetV2) classifica uma foto sua.

Referências

- Material original: tutoriais 93 a 126 de DigitalSreeni,
  - github.com/bnsreenu/python_for_image_processing_APEER
- 3Blue1Brown, But what is a convolution? (vídeo)
- Stanford CS231n: Convolutional Neural Networks for Visual Recognition
- Documentação do Keras 3: keras.io
- CIFAR-10: A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, 2009

OBRIGADO

James Moraes de Almeida
james.almeida@ilum.cnpem.br

ilum.cnpem.br