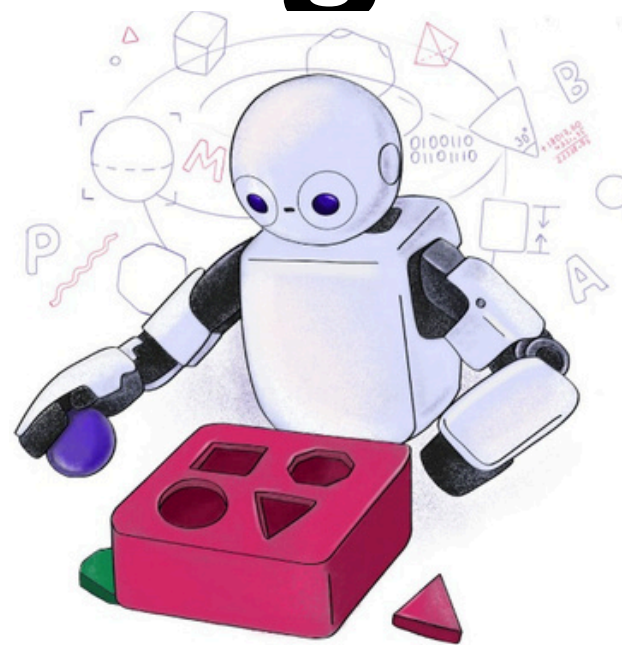


TP558 - Tópicos avançados em Machine Learning: **Siamese Neural Networks for One-shot Image Recognition**



Inatel

Ana Cecília Silveira
Fernandes
ana.fernandes@mtel.inatel.br


Introdução



Em aprendizado de máquina, treinar bons modelos geralmente requer muito dados e alto poder computacional. Mas e quando temos apenas *um único* exemplo de cada classe?



Em aprendizado de máquina, treinar bons modelos geralmente requer muito dados e alto poder computacional. Mas e quando temos apenas *um único* exemplo de cada classe?

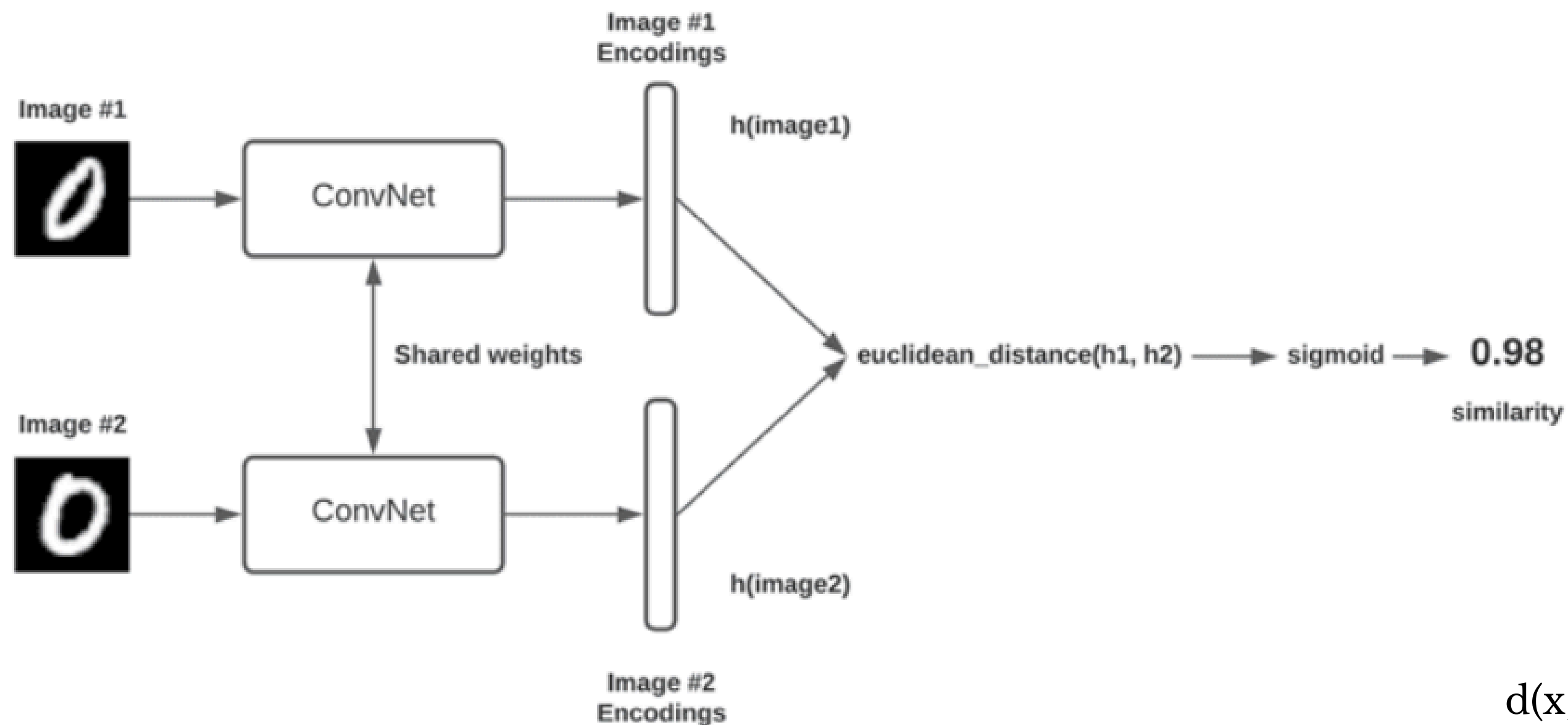
 **One-Shot Learning** 

Fundamentação Teórica/Arquitetura

	Zero-Shot Learning	One-Shot Learning
Definição	O modelo não recebe exemplos da classe alvo.	Recebe um único exemplo por classe para aprender.
Base de Conhecimento	Usa atributos, descrições semânticas ou embeddings para reconhecer classes novas.	Aprende a partir de exemplos reais (mesmo que apenas 1)
Objetivo	Transferir conhecimento para classes nunca vistas.	Generalizar a partir de poucos dados rotulados
Exemplo prático	Identificar um animal desconhecido com base em sua descrição textual.	Reconhecer uma pessoa com apenas uma foto de referência
Aplicações	Tradução automática para novos idiomas, classificação de imagens sem rótulos de treino.	Reconhecimento facial, diagnóstico de doenças raras, biometria etc.



Rede Neural Siamesa (SNN)



$$d(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

⌋ Pequena → Semelhantes
⌋ Grande → Diferentes

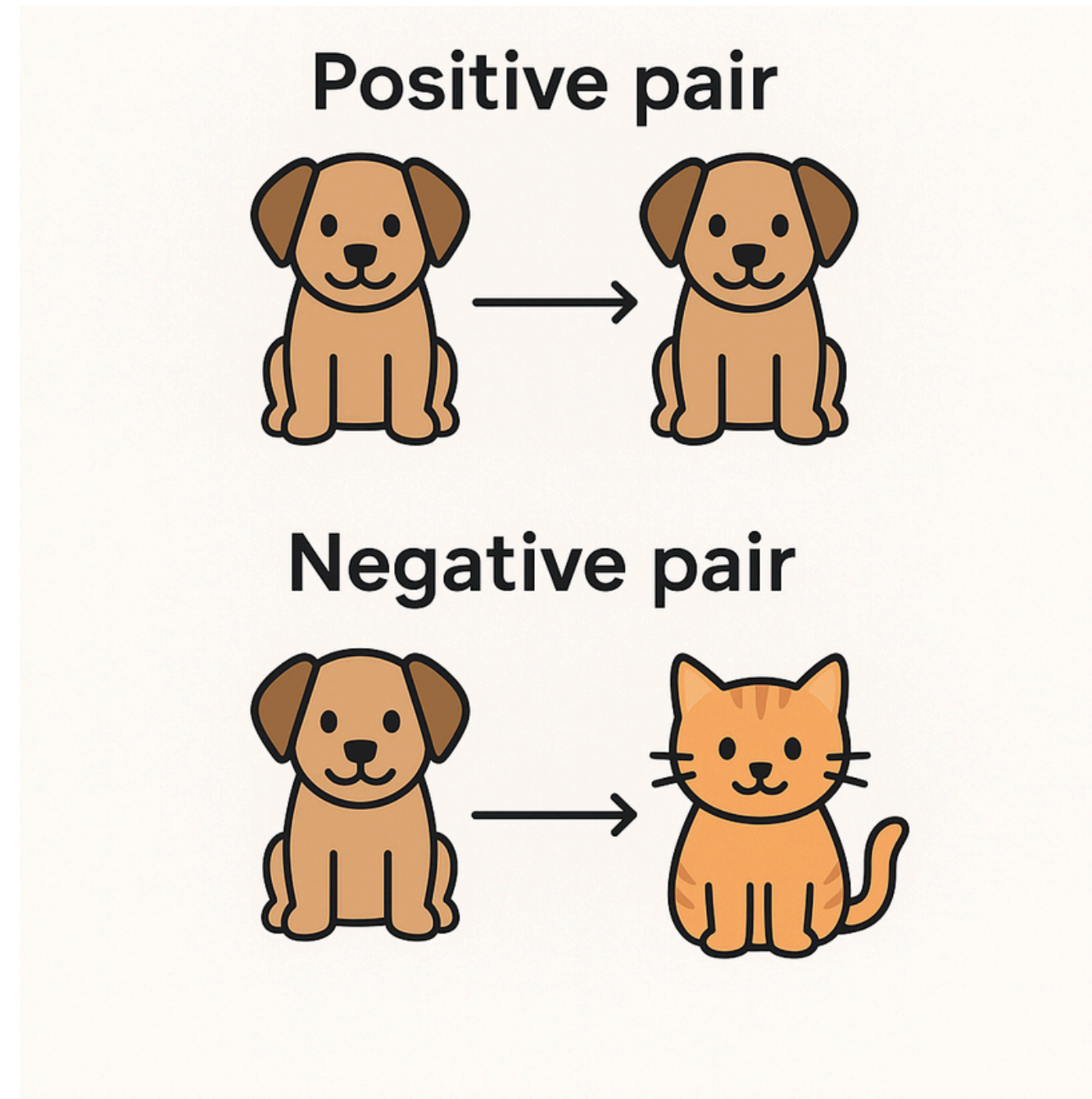
Rede Neural Siamesa (SNN)

C

Treinamento e otimização

Treinamento é baseado em pares de entrada:

- Pares positivos (mesma classe/mesmo objeto)
- Pares Negativos (classes diferentes)



Loss Function:

Objetivo: minimizar a distância entre embeddings de pares positivos e maximizar a distância entre embeddings de pares negativos, respeitando uma margem definida.

Contrastive Loss:

- Uma das mais comuns em SNNs.
- Trabalha com pares positivos (mesma classe) e pares negativos (classes diferentes).
- Objetivo: minimizar a distância entre embeddings de pares positivos e maximizar a distância entre embeddings de pares negativos, respeitando uma margem definida.
- Vantagem: intuitiva e diretamente voltada para medir similaridade.

Triplet Loss:

- Usa trios de exemplos: âncora, positivo (mesma classe da âncora) e negativo (classe diferente).
- Objetivo: garantir que a distância entre âncora e positivo seja sempre menor que a distância entre âncora e negativo, dentro de uma margem.
- Vantagem: mais robusta em problemas com grande variabilidade intra-classe.

Binary Cross-Entropy Loss (BCE):

- Trata a tarefa como um problema de classificação binária: o modelo prevê a probabilidade de dois inputs pertencerem à mesma classe.
- Muito usada quando a saída da rede é uma camada sigmoideal que indica "similar" (1) ou "não similar" (0).
- Pode ser combinada com regularização L2 ou dropout para reduzir overfitting.

Em resumo:

- Contrastive Loss \rightarrow foca em pares.
- Triplet Loss \rightarrow foca em trios, mais discriminativa.
- Binary Cross-Entropy \rightarrow foca em probabilidade, mais próxima da classificação binária

Otimização:

- Durante o treino, os pesos da rede são ajustados por backpropagation e algoritmos de otimização (ex.: Adam, SGD). O objetivo é minimizar a perda, ou seja, melhorar a capacidade da rede em mapear pares semelhantes próximos e pares diferentes distantes no espaço de embeddings.

	Adam	SGD (puro ou com momentum)
Velocidade de convergência	Rápida nos estágios iniciais.	Mais lenta, mas pode alcançar melhor generalização.
Estabilidade	Mais estável em datasets ruidosos (como pares positivos/negativos de SNN).	Pode oscilar bastante sem ajustes de taxa de aprendizado.
Dependência do LR	Menos sensível, valores padrões funcionam bem.	Muito sensível → escolher LR errado pode travar o treino.
Generalização	Pode convergir para mínimos locais bons, mas às vezes “superajusta”.	Geralmente generaliza melhor a longo prazo, especialmente com momentum.
Uso típico em SNN	Muito comum em implementações rápidas e práticas (MNIST, Omniglot, few-shot learning).	Usado quando se busca performance máxima em benchmarks e com mais tempo para tuning.

Generalização:

- Depois de treinada, a SNN consegue comparar novas classes nunca vistas, porque ela não aprendeu rótulos diretamente, mas sim a medir similaridade entre representações.

Vantagens e desvantagens

✅ Vantagens

- Generalizam para novas classes sem precisar re-treinar totalmente
- Precisam de poucos dados por classe (one-shot, few-shot)
- Embeddings podem ser reutilizados para busca e verificação
- Funcionam em diversos tipos de dados (imagem, texto, áudio, biometria)
- Menor necessidade de re-treinar quando surgem novas classes

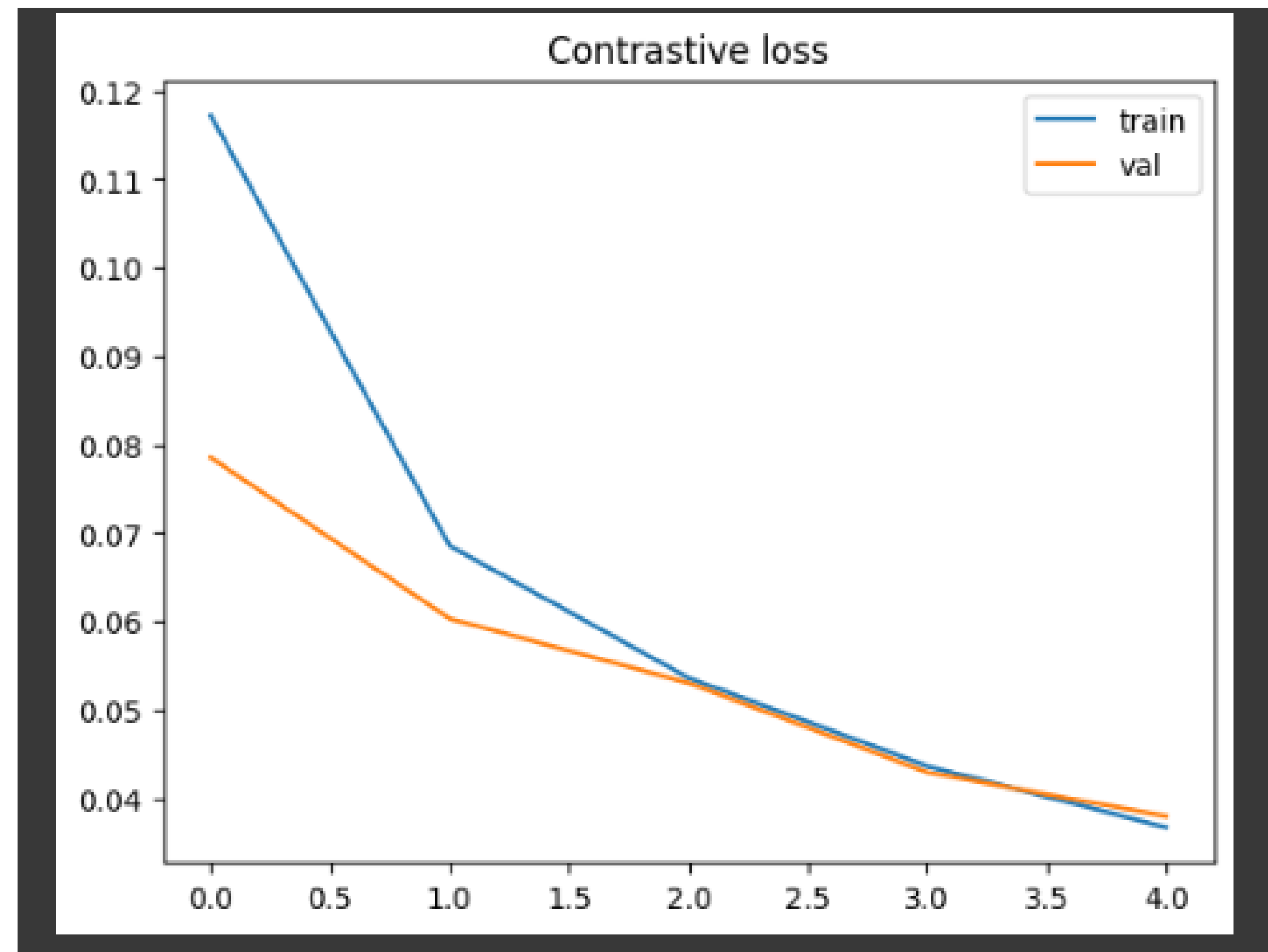
⚠️ Desvantagens

- Exigem muitos pares positivos/negativos → treinamento pode ser caro
- Número de pares cresce muito em bases grandes (problema de escalabilidade)
- É necessário definir um threshold de similaridade (pode variar por dataset)
- Dependem fortemente da qualidade dos embeddings aprendidos
- Comparação par-a-par pode ser lenta em sistemas de grande escala

Exemplo(s) de aplicação



Img A	Img B	Distance	Pred same?	True
		0.154	1	1
		0.275	1	1
		0.095	1	1
		0.115	1	1
		1.265	0	0



- A contrastive loss cai de forma estável em treino e validação, indicando que o modelo aproxima positivos e afasta negativos além da margem. A curva de validação levemente abaixo da de treino sugere boa generalização, sem sinais de overfitting.



Pipeline da CNN (camada a camada)

- Conv32 (3×3 , ReLU) \rightarrow capta bordas/traços finos dos dígitos.
- MaxPool (2×2) \rightarrow reduz resolução, mantém o que é mais saliente.
- Conv64 (3×3 , ReLU) \rightarrow combina traços em partes maiores (curvas, cantos).
- MaxPool (2×2) \rightarrow nova redução + invariância leve a deslocamentos.
- Conv128 (3×3 , ReLU) \rightarrow captura padrões mais abstratos do dígito.
- GAP (Global Average Pooling) \rightarrow transforma mapas em 1 vetor por canal, reduz parâmetros e evita overfitting.
- Dense(64) \rightarrow comprime tudo num vetor de 64 dimensões = embedding.

Matriz de Confusão e métricas.



Matriz de confusão:

```
[[3962  48]
```

```
[ 100 3908]]
```

TN=3962 FP=48 FN=100 TP=3908

Acurácia=0.982 Precisão=0.988 Recall=0.975 F1=0.981 AUC=0.998

- A rede aprendeu um espaço de embeddings com separação quase perfeita entre pares iguais e diferentes (AUC = 0.998).
- Com o limiar escolhido, obteve Acurácia = 98.2%, Precisão = 98.8%, Recall = 97.5% e F1 = 98.1%.
- Erros estão baixos e bem distribuídos: FAR \approx 1.2% (falsas aceitações) e FRR \approx 2.5% (falsas rejeições).
- As curvas de loss de treino/validação caem juntas → boa generalização, sem overfitting visível.

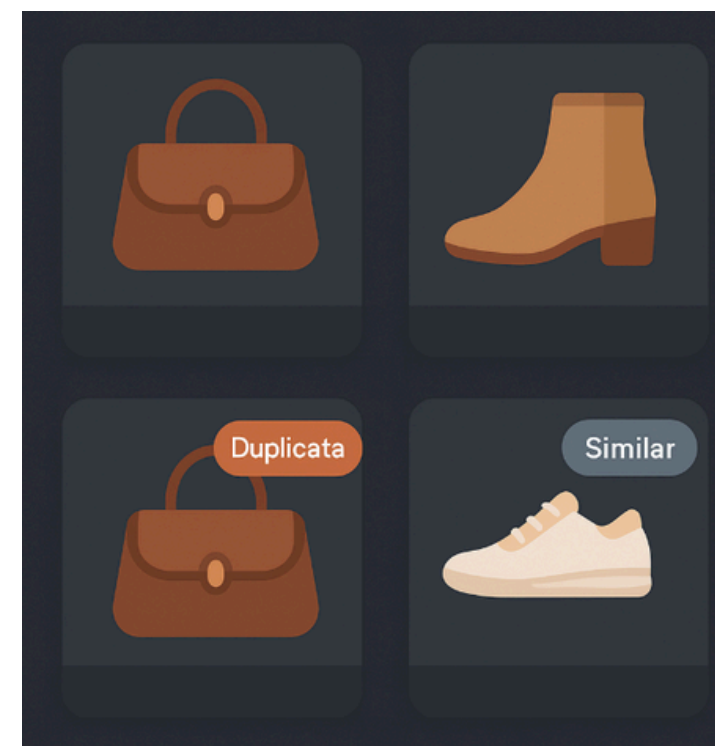
Casos de Sucesso



Verificação de assinatura



Verificação facial
(foto-a-foto)



Deduplicação de catálogo /
busca visual em e-commerce

Comparação com outros algoritmos

	SNN (Siamese Neural Network)	CNN Clássica	KNN (k-Nearest Neighbors)	SVM (Support Vector Machine)
Tipo de saída	Similaridade (0 ou 1 / distância)	Classe (ex: gato, cachorro)	Classe do vizinho mais próximo	Classe, com margem máxima
Quando funciona melhor	Poucos dados por classe (one-shot, few-shot learning)	Muitas amostras para treinar cada classe	Dados de baixa dimensão e bem separados	Dados com margens bem definidas entre
Reutilização	Pode generalizar para novas classes nunca vistas (só	Precisa re-treinar para novas classes	Só adiciona novos pontos no dataset	Precisa re-treinar para novas classes
Complexidade de treino	Alta (treina em pares, demanda redes profundas)	Alta (treino supervisionado com muitos dados)	Baixa (sem treino real, só comparação)	Média/Alta (ajuste de parâmetros e kernels)
Uso prático	Reconhecimento facial (FaceNet, Signature	Classificação de imagens, áudio, texto	Sistemas simples de recomendação e classificação	Classificação em bioinformática, NLP, visão
Vantagem principal	Generalização para novos cenários	Alta acurácia se houver muitos dados	Simples de implementar	Boa performance em espaços de alta dimensão
Desvantagem	Mais difícil de treinar e ajustar	Não lida bem com classes novas	Escalabilidade ruim (custo cresce com dataset)	Difícil escolher kernel e parâmetros

Conclusão

As Redes Neurais Siamesas são uma alternativa poderosa quando temos muitos rótulos possíveis, mas poucos exemplos por classe. Elas mudam o foco de classificação direta para medição de similaridade, oferecendo flexibilidade e generalização que outros algoritmos tradicionais dificilmente alcançam.

Perguntas?

Referências

- [1] PyImageSearch. Siamese Networks with Keras, TensorFlow, and Deep Learning. Disponível em: <https://pyimagesearch.com/2020/11/30/siamese-networks-with-keras-tensorflow-and-deep-learning/>
- [2] [Koch, G. et al. Siamese Neural Networks for One-shot Image Recognition. CMU. Disponível em: https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf](#)
- [3] [Schroff, F., Kalenichenko, D., & Philbin, J. \(2015\). FaceNet: A Unified Embedding for Face Recognition and Clustering. CVPR.](#)
- [5] [Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. S. \(2016\). Fully-Convolutional Siamese Networks for Object Tracking.](#)
- [6] <https://youtu.be/uPSAk27B-Xs>

Obrigada!

Formulário

