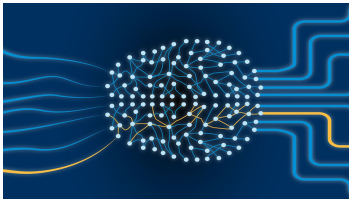
Parte IV: Aprendizado de Máquina.

Prof. Fabiano Araujo Soares, Dr. / FGA 0221 - Inteligência Artificial

Universidade de Brasília

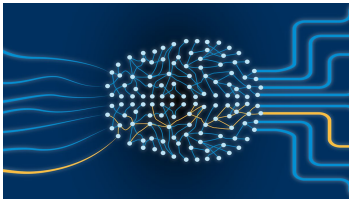
2025

Aprendizado Supervisionado: Algoritmos



Por que *Deep Learning* revolucionou diversas áreas do conhecimento na última década?

Aprendizado Supervisionado: Algoritmos



Por que *Deep Learning* revolucionou diversas áreas do conhecimento na última década?

Devido à capacidade das redes profundas de aprender representações complexas e detectar padrões ocultos diretamente a partir dos dados, superando métodos tradicionais em tarefas de visão, linguagem, medicina e muito mais.

O que é Deep Learning?

- **Deep Learning** é um subcampo do Machine Learning que utiliza redes neurais artificiais com múltiplas camadas ocultas

O que é Deep Learning?

- **Deep Learning** é um subcampo do Machine Learning que utiliza redes neurais artificiais com múltiplas camadas ocultas
- Inspirado na estrutura e funcionamento do cérebro humano

O que é Deep Learning?

- **Deep Learning** é um subcampo do Machine Learning que utiliza redes neurais artificiais com múltiplas camadas ocultas
- Inspirado na estrutura e funcionamento do cérebro humano
- Capaz de aprender representações hierárquicas e abstratas dos dados

O que é Deep Learning?

- **Deep Learning** é um subcampo do Machine Learning que utiliza redes neurais artificiais com múltiplas camadas ocultas
- Inspirado na estrutura e funcionamento do cérebro humano
- Capaz de aprender representações hierárquicas e abstratas dos dados
- Transformou aplicações em visão computacional, processamento de linguagem natural, reconhecimento de fala, etc.

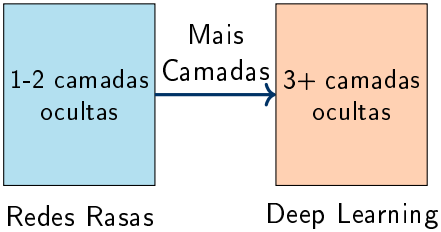
O que é Deep Learning?

- **Deep Learning** é um subcampo do Machine Learning que utiliza redes neurais artificiais com múltiplas camadas ocultas
- Inspirado na estrutura e funcionamento do cérebro humano
- Capaz de aprender representações hierárquicas e abstratas dos dados
- Transformou aplicações em visão computacional, processamento de linguagem natural, reconhecimento de fala, etc.

Características Principais

- Múltiplas camadas de abstração
- Aprendizado de representações automáticas
- Escalabilidade com GPUs
- Grande volume de dados necessários

Evolução: De Machine Learning a Deep Learning



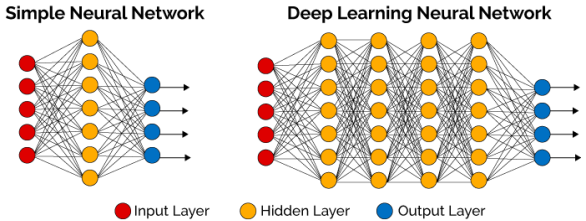
Machine Learning Tradicional:

- Engenharia manual de características
- Modelos simples
- Requer menos dados

Deep Learning:

- Aprendizado automático de características
- Modelos complexos
- Requer grandes volumes

Arquitetura: Profundidade vs Largura



Redes Rasas:

- 1-2 camadas ocultas
- Melhor para dados simples
- Treinamento rápido
- Menor poder representativo

Redes Profundas:

- 3+ camadas ocultas
- Dados complexos e não-lineares
- Aprendizado de abstrações
- Maior poder representativo

Vantagens das Redes Profundas

1. Representação Hierárquica

Cada camada aprende representações cada vez mais abstratas:

- Camadas inferiores: features simples (bordas, texturas)
- Camadas intermediárias: combinações (formas, objetos)
- Camadas superiores: conceitos abstratos (semântica)

2. Eficiência de Representação

Muitos conceitos complexos requerem exponencialmente menos neurônios em redes profundas do que em redes rasas (“maldição da dimensionalidade”).

3. Generalização

Com arquiteturas apropriadas, redes profundas generalizam melhor em dados não vistos.

Desafios no Treinamento de Redes Profundas

- **Vanishing/Exploding Gradients**: Gradientes podem diminuir ou explodir durante backpropagation

Desafios no Treinamento de Redes Profundas

- **Vanishing/Exploding Gradients**: Gradientes podem diminuir ou explodir durante backpropagation
- **Overfitting**: Com mais parâmetros, risco aumenta

Desafios no Treinamento de Redes Profundas

- **Vanishing/Exploding Gradients**: Gradientes podem diminuir ou explodir durante backpropagation
- **Overfitting**: Com mais parâmetros, risco aumenta
- **Custo Computacional**: Treino lento sem GPUs

Desafios no Treinamento de Redes Profundas

- **Vanishing/Exploding Gradients**: Gradientes podem diminuir ou explodir durante backpropagation
- **Overfitting**: Com mais parâmetros, risco aumenta
- **Custo Computacional**: Treino lento sem GPUs
- **Inicialização**: Peso inicial crítico para convergência

Desafios no Treinamento de Redes Profundas

- **Vanishing/Exploding Gradients**: Gradientes podem diminuir ou explodir durante backpropagation
- **Overfitting**: Com mais parâmetros, risco aumenta
- **Custo Computacional**: Treino lento sem GPUs
- **Inicialização**: Peso inicial crítico para convergência

Soluções

- ReLU e funções de ativação modernas
- Normalização em lote (Batch Normalization)
- Dropout e regularização L2 (penalização para pesos grandes)
- Otimizadores adaptativos (Adam, RMSprop – Taxas de aprendizado dinâmicas)

CNN: Motivação

- Redes totalmente conectadas não escalam bem para imagens grandes

CNN: Motivação

- Redes totalmente conectadas não escalam bem para imagens grandes
- Uma imagem $256 \times 256 \times 3 = 196.608$ pixels

CNN: Motivação

- Redes totalmente conectadas não escalam bem para imagens grandes
- Uma imagem $256 \times 256 \times 3 = 196.608$ pixels
- Cada neurônio teria $\sim 200k$ conexões (parâmetros!)

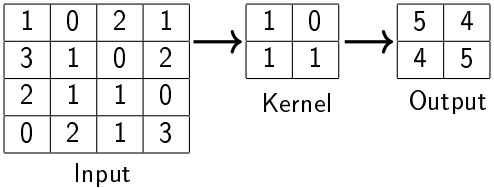
CNN: Motivação

- Redes totalmente conectadas não escalam bem para imagens grandes
- Uma imagem $256 \times 256 \times 3 = 196.608$ pixels
- Cada neurônio teria $\sim 200k$ conexões (parâmetros!)
- **Idea**: Usar **convolução** para extrair features localmente

CNN: Motivação

- Redes totalmente conectadas não escalam bem para imagens grandes
- Uma imagem $256 \times 256 \times 3 = 196.608$ pixels
- Cada neurônio teria $\sim 200k$ conexões (parâmetros!)
- **Idea**: Usar **convolução** para extrair features localmente
- Explorar propriedades de imagens:
  - Localidade: pixels vizinhos são correlacionados
  - Estacionariedade: características relevantes em qualquer posição
  - Composição hierárquica: características simples formam características complexas

Operação de Convolução



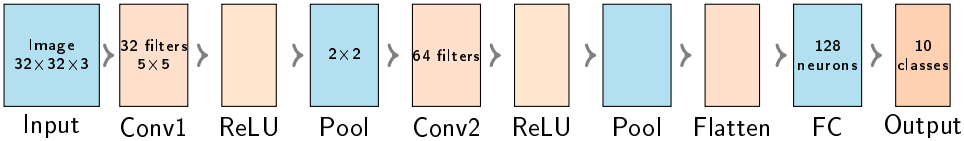
Operação matemática:

$$(I * K)[i,j] = \sum_m \sum_n I[i+m, j+n] \cdot K[m,n]$$

Características:

- Compartilhamento de pesos: mesmo kernel para toda imagem
- Redução de parâmetros: $k \times k$ (2×2) vs $W \times H$ (4×4)
- Detecção de features locais: bordas, texturas, padrões

Arquitetura CNN Típica

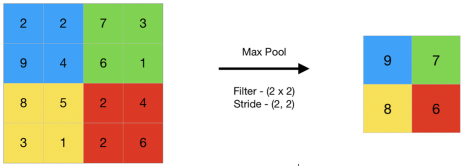


Camadas principais:

- **Convolução**: Extração de features
- **Ativação**: (ReLU) Não-linearidade
- **Pooling**: Redução de dimensionalidade e robustez
- **Fully Connected**: Classificação final

Operações em CNN

1. **Convolução:** $Y_{i,j} = \sum_{k,l} X_{i+k,j+l} \cdot W_{k,l} + b$
2. **Pooling (Max Pooling):** Seleciona valor máximo em janelas 2×2 ou 3×3



Benefícios do Pooling:

- Reduz dimensionalidade
- Torna features mais robustas a pequenas translações
- Reduz parâmetros e computação

RNN: Motivação

- Muitos problemas envolvem **sequências** (tempo, série temporal, linguagem)

RNN: Motivação

- Muitos problemas envolvem **sequências** (tempo, série temporal, linguagem)
- Dados anteriores são relevantes para prever o futuro

RNN: Motivação

- Muitos problemas envolvem **sequências** (tempo, série temporal, linguagem)
- Dados anteriores são relevantes para prever o futuro
- Redes feedforward padrão não mantêm "memória" de inputs anteriores

RNN: Motivação

- Muitos problemas envolvem **sequências** (tempo, série temporal, linguagem)
- Dados anteriores são relevantes para prever o futuro
- Redes feedforward padrão não mantêm "memória" de inputs anteriores
- **Solução**: Usar **conexões recorrentes** que permitem loops

RNN: Motivação

- Muitos problemas envolvem **sequências** (tempo, série temporal, linguagem)
- Dados anteriores são relevantes para prever o futuro
- Redes feedforward padrão não mantêm "memória" de inputs anteriores
- **Solução**: Usar **conexões recorrentes** que permitem loops
- O neurônio tem estado que depende do estado anterior

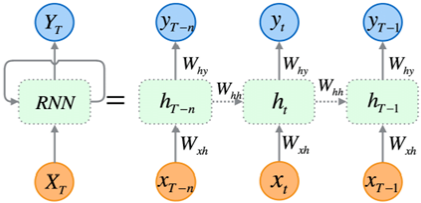
RNN: Motivação

- Muitos problemas envolvem **sequências** (tempo, série temporal, linguagem)
- Dados anteriores são relevantes para prever o futuro
- Redes feedforward padrão não mantêm "memória" de inputs anteriores
- **Solução**: Usar **conexões recorrentes** que permitem loops
- O neurônio tem estado que depende do estado anterior

Exemplos de Aplicações:

- Processamento de Linguagem Natural (NLP)
- Séries temporais e previsão
- Reconhecimento de fala
- Tradução automática

Arquitetura RNN



Equações RNN:

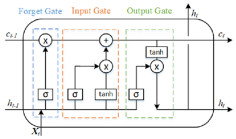
$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = W_{hy}h_t + b_y \quad (2)$$

onde h_t é o estado oculto no tempo t.

LSTM: Long Short-Term Memory

Problema com RNN Simples: Vanishing/Exploding Gradients em sequências longas
Solução - LSTM (1997, Hochreiter & Schmidhuber):



Componentes LSTM:

- **Forget Gate**: Decide o quanto do estado anterior manter
- **Input Gate**: Decide quais informações novas armazenar
- **Cell State**: Transporta informação através da sequência (caminho gradiente direto)
- **Output Gate**: Decide quais informações expor

Vantagens: Previne vanishing gradients, aprende dependências de longo prazo efetivamente.

GRU: Gated Recurrent Unit - Alternativa mais leve ao LSTM

Simplificação do LSTM com:

- 2 gates (Reset + Update) vs 3 em LSTM
- Menor número de parâmetros e Treinamento ligeiramente mais rápido
- Performance comparável em muitas tarefas

Equações GRU

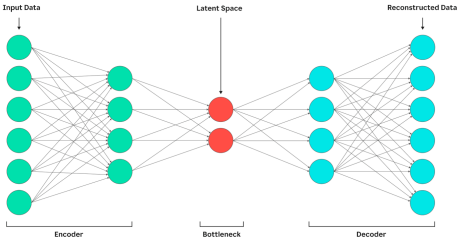
$$r_t = \sigma(W_{ir}x_t + W_{hr}h_{t-1}) \quad \text{(Reset Gate)} \tag{3}$$

$$z_t = \sigma(W_{iz}x_t + W_{hz}h_{t-1}) \quad \text{(Update Gate)} \tag{4}$$

$$\tilde{h}_t = \tanh(W_{ih}x_t + W_{hh}(r_t \odot h_{t-1})) \quad \text{(Candidate)} \tag{5}$$

$$h_t = (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{t-1} \tag{6}$$

Autoencoders



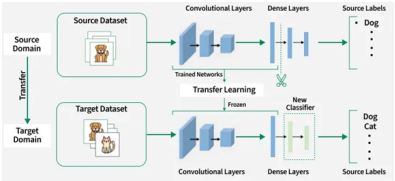
Características:

- Aprende representação comprimida (latent space) z
- Encoder: $x \rightarrow z$ (reduz dimensionalidade)
- Decoder: $z \rightarrow \hat{x}$ (reconstrói)
- Objetivo: minimizar $||x - \hat{x}||^2$

Aplicações:

- Redução de dimensionalidade não-linear
- Detecção de anomalias
- Pré-treinamento para transfer learning

Transfer Learning



Estratégias:

1. **Feature Extraction**: Congelar pesos iniciais, treinar últimas camadas
2. **Fine-tuning**: Descongelar e treinar todo modelo com taxa baixa

Vantagens:

- Requer menos dados para nova tarefa
- Converge mais rapidamente
- Melhor performance em datasets pequenos

O que é Dropout?

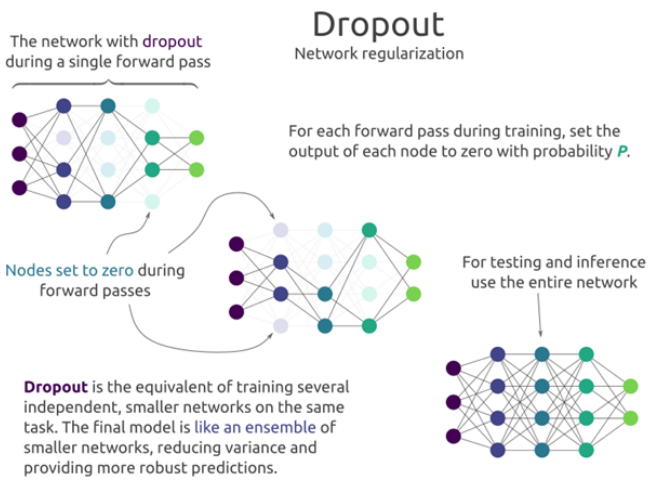
Dropout é uma técnica de regularização que desliga aleatoriamente neurônios durante o treinamento de uma rede neural.

- Durante o **treino**: desligar aleatoriamente 20-50% dos neurônios em cada iteração
- Força a rede a aprender representações **robustas e generalizáveis**
- Previne a **co-adaptação** entre neurônios
- Durante o **teste**: todos os neurônios ficam ativos

Analogia

Como um ensemble de redes: treina múltiplas sub-redes diferentes ao mesmo tempo, melhorando a generalização no final.

Como Funciona Visualmente



Por que Dropout Funciona?

- **Reduz Overfitting**: Impede que o modelo memorize dados de treino

Por que Dropout Funciona?

- **Reduz Overfitting**: Impede que o modelo memorize dados de treino
- **Ensemble Implícito**: Treina múltiplas sub-redes diferentes em paralelo

Por que Dropout Funciona?

- **Reduz Overfitting**: Impede que o modelo memorize dados de treino
- **Ensemble Implícito**: Treina múltiplas sub-redes diferentes em paralelo
- **Aprendizado Robusto**: Força neurônios a aprender características independentes e generalizáveis

Por que Dropout Funciona?

- **Reduz Overfitting**: Impede que o modelo memorize dados de treino
- **Ensemble Implícito**: Treina múltiplas sub-redes diferentes em paralelo
- **Aprendizado Robusto**: Força neurônios a aprender características independentes e generalizáveis
- **Simples de Implementar**: Basta adicionar uma linha de código na maioria dos frameworks

Por que Dropout Funciona?

- **Reduz Overfitting**: Impede que o modelo memorize dados de treino
- **Ensemble Implícito**: Treina múltiplas sub-redes diferentes em paralelo
- **Aprendizado Robusto**: Força neurônios a aprender características independentes e generalizáveis
- **Simples de Implementar**: Basta adicionar uma linha de código na maioria dos frameworks

Resultado

O modelo generaliza melhor para dados nunca vistos durante o treinamento.

Evolução dos Otimizadores

- **SGD (Stochastic Gradient Descent)**:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

Simples, mas pode ser lento ou instável.

Evolução dos Otimizadores

- **SGD (Stochastic Gradient Descent)**:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

  Simples, mas pode ser lento ou instável.

- **Momentum**: Acumula direção do gradiente

$$v_t = \beta v_{t-1} + \nabla J(\theta_t), \quad \theta_{t+1} = \theta_t - \eta v_t$$

Evolução dos Otimizadores

- **SGD (Stochastic Gradient Descent)**:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

  Simples, mas pode ser lento ou instável.

- **Momentum**: Acumula direção do gradiente

$$v_t = \beta v_{t-1} + \nabla J(\theta_t), \quad \theta_{t+1} = \theta_t - \eta v_t$$

- **RMSprop**: Adapta taxa por parâmetro (componentes maiores $\rightarrow$ menor taxa)

Evolução dos Otimizadores

- **SGD (Stochastic Gradient Descent)**:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

  Simples, mas pode ser lento ou instável.

- **Momentum**: Acumula direção do gradiente

$$v_t = \beta v_{t-1} + \nabla J(\theta_t), \quad \theta_{t+1} = \theta_t - \eta v_t$$

- **RMSprop**: Adapta taxa por parâmetro (componentes maiores $\rightarrow$ menor taxa)
- **Adam** (Adaptive Moment Estimation): Combina RMSprop + Momentum

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} m_t$$

Muito popular, funciona bem na maioria dos casos.

Dicas de Treinamento

Preparação dos Dados

- **Normalização**: Subtraia média, divida por desvio padrão
- **Embaralhamento**: Embaralhe dados entre épocas
- **augmentation**: Aumente dataset com transformações (rotação, translação, etc)

Hiperparâmetros

- **Taxa de aprendizado**: Comece com 1e-3 ou 1e-4, ajuste
- **Batch size**: Típico 32, 64, 128 (maior = mais estável mas atualização menos frequente)
- **Número de épocas**: Use *early stopping* com validação

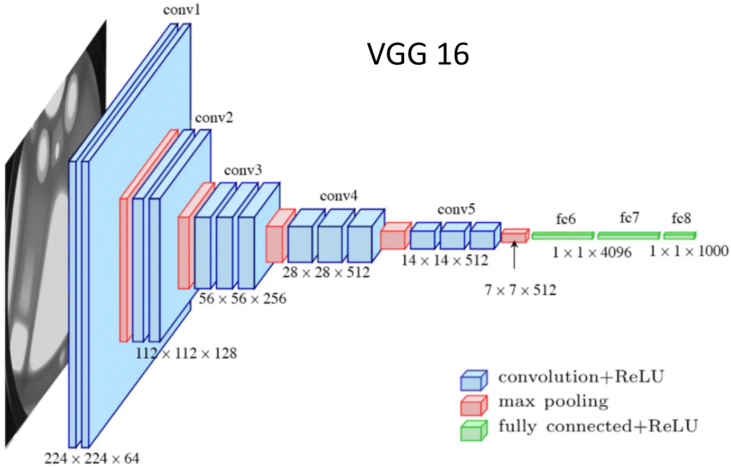
Monitoramento

- Monitore perda de treino e validação
- Se validação começa a piorar: reduz taxa ou usa early stopping
- Plote ativações para verificar dead neurons (ReLU)

Visão Computacional

- **Classificação**: ImageNet, MNIST, CIFAR-10
- **Detecção**: YOLO, Faster R-CNN, SSD
- **Segmentação**: U-Net, DeepLab (aplicações médicas, autônomas)
- **Reconhecimento Facial**: Face ID, reconhecimento de expressões

Exemplos de redes: ResNet, VGG, Inception, EfficientNet


VGG 16
conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

- **Classificação de Texto**: Análise de sentimento, categorização
- **Tradução Automática**: Seq2Seq, Transformers (BERT, GPT)
- **Geração de Texto**: GPT-2, GPT-3 (language models)
- **Reconhecimento de Fala**: Wav2Vec, Whisper

Arquiteturas: RNN/LSTM, Attention, Transformer (revolucionou NLP)

Aplicações em Bioengenharia

- **Diagnóstico Médico**:
  - Detecção de glaucoma em fundos oculares
  - Diagnóstico de câncer em imagens médicas (CT, MRI, mammografia)
  - Classificação de arritmias cardíacas
- **Processamento de Sinais Bioelétricos**:
  - EEG: Detecção de epilepsia, classificação de estados
  - EMG: Reconhecimento de gestos, próteses inteligentes
  - ECG: Detecção de arritmias
- **Imagem Biomédica**:
  - Segmentação de tumores
  - Reconstrução de imagens (super-resolução, denoising)

Desafio: Dados médicos são limitados e requerem privacidade. Federated learning e synthetic data são soluções.

Desafios Atuais

1. Interpretabilidade (Explainability)

Redes profundas são "caixas pretas". Como entender decisões?

- Grad-CAM, SHAP, LIME para visualizar contribuições (XAI)
- Crítico em aplicações médicas

2. Eficiência Computacional

Modelos gigantes (GPT-3: 175B parâmetros) são caros de treinar e usar

- Model compression: Quantização, destilação
- Eficiência: MobileNet, EfficientNet para edge devices

Desafios Atuais (Continuação)

3. Robustez a Adversários

Exemplos adversariais: pequenas perturbações podem enganar rede

- Adversarial training
- Certified robustness

4. Requer Grandes Volumes de Dados

Redes profundas precisam de muitos dados rotulados

- Few-shot learning
- Self-supervised learning
- Data augmentation

Desafios Atuais (Continuação)

5. Considerações Éticas

- Bias nos dados $\rightarrow$ discriminação
- Privacidade: dados sensíveis em treinamento
- Responsabilidade em decisões críticas

6. Falta de Generalização

Modelos treinados em um domínio falham em outro (domain shift)

- Domain adaptation
- Meta-learning

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)
- **Few-Shot Learning**: Aprender com poucos exemplos como humanos

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)
- **Few-Shot Learning**: Aprender com poucos exemplos como humanos
- **Self-Supervised Learning**: Aprender sem rótulos (representações automáticas)

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)
- **Few-Shot Learning**: Aprender com poucos exemplos como humanos
- **Self-Supervised Learning**: Aprender sem rótulos (representações automáticas)
- **Neuromorphic Computing**: Hardware inspirado em cérebro (SpikeNN, Intel Loihi)

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)
- **Few-Shot Learning**: Aprender com poucos exemplos como humanos
- **Self-Supervised Learning**: Aprender sem rótulos (representações automáticas)
- **Neuromorphic Computing**: Hardware inspirado em cérebro (SpikeNN, Intel Loihi)
- **Federated Learning**: Treinar em dados distribuídos sem centralizar

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)
- **Few-Shot Learning**: Aprender com poucos exemplos como humanos
- **Self-Supervised Learning**: Aprender sem rótulos (representações automáticas)
- **Neuromorphic Computing**: Hardware inspirado em cérebro (SpikeNN, Intel Loihi)
- **Federated Learning**: Treinar em dados distribuídos sem centralizar
- **Quantum Machine Learning**: Explorar vantagens quânticas

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)
- **Few-Shot Learning**: Aprender com poucos exemplos como humanos
- **Self-Supervised Learning**: Aprender sem rótulos (representações automáticas)
- **Neuromorphic Computing**: Hardware inspirado em cérebro (SpikeNN, Intel Loihi)
- **Federated Learning**: Treinar em dados distribuídos sem centralizar
- **Quantum Machine Learning**: Explorar vantagens quânticas
- **Interpretabilidade**: Redes mais transparentes e compreensíveis

Perspectivas Futuras

- **Transformers**: Revolucionaram NLP, agora aplicados a visão (Vision Transformer)
- **Few-Shot Learning**: Aprender com poucos exemplos como humanos
- **Self-Supervised Learning**: Aprender sem rótulos (representações automáticas)
- **Neuromorphic Computing**: Hardware inspirado em cérebro (SpikeNN, Intel Loihi)
- **Federated Learning**: Treinar em dados distribuídos sem centralizar
- **Quantum Machine Learning**: Explorar vantagens quânticas
- **Interpretabilidade**: Redes mais transparentes e compreensíveis
- **Multimodal Learning**: Integrar múltiplas modalidades (texto, imagem, áudio)

Resumo de Conceitos

Conceito	Quando Usar	Características
CNN	Imagens	Convolução, pooling
RNN/LSTM	Sequências	Recorrência, memória
Autoencoder	Redução dim.	Compressão, features
Transfer	Pouco dado	Reutiliza conhecimento
Dropout	Overfitting	Regularização

Fluxo de Desenvolvimento:

1. Escolha arquitetura apropriada (CNN/RNN/Transformer/etc)
2. Prepare e normalize dados
3. Implemente, compile, treine, monitore
4. Ajuste hiperparâmetros
5. Deploy com inferência otimizada

Resumo

- Deep Learning é **aprendizado hierárquico** de representações

Resumo

- Deep Learning é **aprendizado hierárquico** de representações
- **CNNs**: Excelente para visão computacional

Resumo

- Deep Learning é **aprendizado hierárquico** de representações
- **CNNs**: Excelente para visão computacional
- **RNNs/LSTMs**: Perfeito para sequências e séries temporais

Resumo

- Deep Learning é **aprendizado hierárquico** de representações
- **CNNs**: Excelente para visão computacional
- **RNNs/LSTMs**: Perfeito para sequências e séries temporais
- **Arquitetura correta é crucial**: escolha depende do problema

Resumo

- Deep Learning é **aprendizado hierárquico** de representações
- **CNNs**: Excelente para visão computacional
- **RNNs/LSTMs**: Perfeito para sequências e séries temporais
- **Arquitetura correta é crucial**: escolha depende do problema
- **Regularização é essencial**: Dropout, Batch Norm, L2

Resumo

- Deep Learning é **aprendizado hierárquico** de representações
- **CNNs**: Excelente para visão computacional
- **RNNs/LSTMs**: Perfeito para sequências e séries temporais
- **Arquitetura correta é crucial**: escolha depende do problema
- **Regularização é essencial**: Dropout, Batch Norm, L2
- **Transfer Learning**: Aproveitamente pré-treinamento

Resumo

- Deep Learning é **aprendizado hierárquico** de representações
- **CNNs**: Excelente para visão computacional
- **RNNs/LSTMs**: Perfeito para sequências e séries temporais
- **Arquitetura correta é crucial**: escolha depende do problema
- **Regularização é essencial**: Dropout, Batch Norm, L2
- **Transfer Learning**: Aproveitamente pré-treinamento
- **Dados são rei**: Qualidade > Quantidade

Resumo

- Deep Learning é **aprendizado hierárquico** de representações
- **CNNs**: Excelente para visão computacional
- **RNNs/LSTMs**: Perfeito para sequências e séries temporais
- **Arquitetura correta é crucial**: escolha depende do problema
- **Regularização é essencial**: Dropout, Batch Norm, L2
- **Transfer Learning**: Aproveitamente pré-treinamento
- **Dados são rei**: Qualidade > Quantidade
- **Interpretabilidade importa**: Especialmente em aplicações críticas

Referências

- Russell, S., Norvig, P., "Artificial Intelligence: A Modern Approach", 4th ed., Person, 2022.
- Muller, A. C., Guido, S., "Introduction to Machine Learning with Python A Guide for Data Scientists", O'Reilly, 2017.
- Hochreiter, S., Schmidhuber, J. "Long Short-Term Memory", Neural Computation, 1997.

Obrigado!

E-mail: fabianosoares@unb.br

LinkedIn: https://www.linkedin.com/in/fabiano-soares-06b6a821a/

Site do curso: https://www.fabianosoares.eng.br/fga0221-inteligência-artificial