CURSO SUPERIOR DE ADS

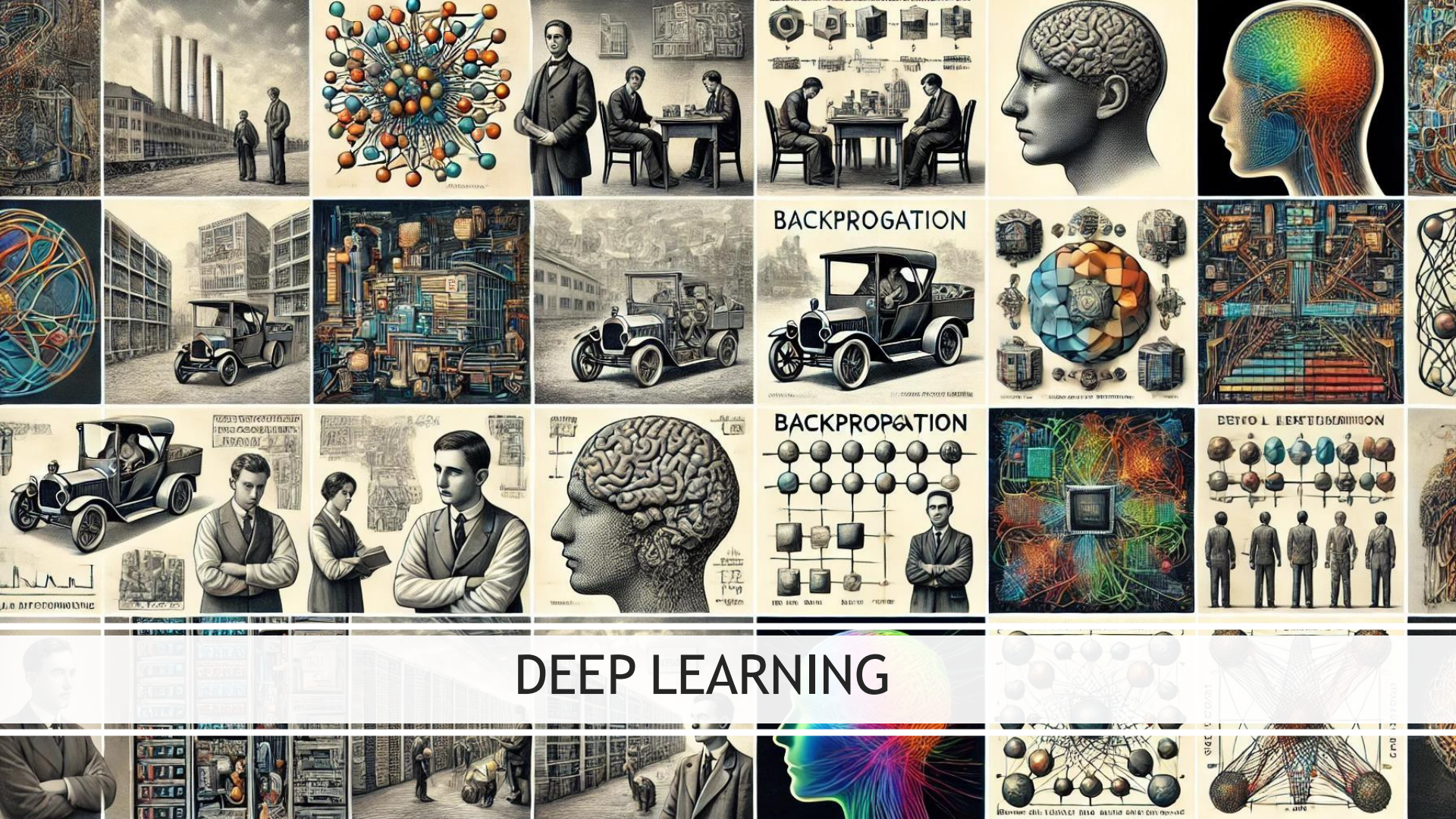
Deep Learning II

Prof. Fernando Marlon Soares Figueiredo

Disciplina: Ciência de Dados e Bigdata

SECTRAS

ADS



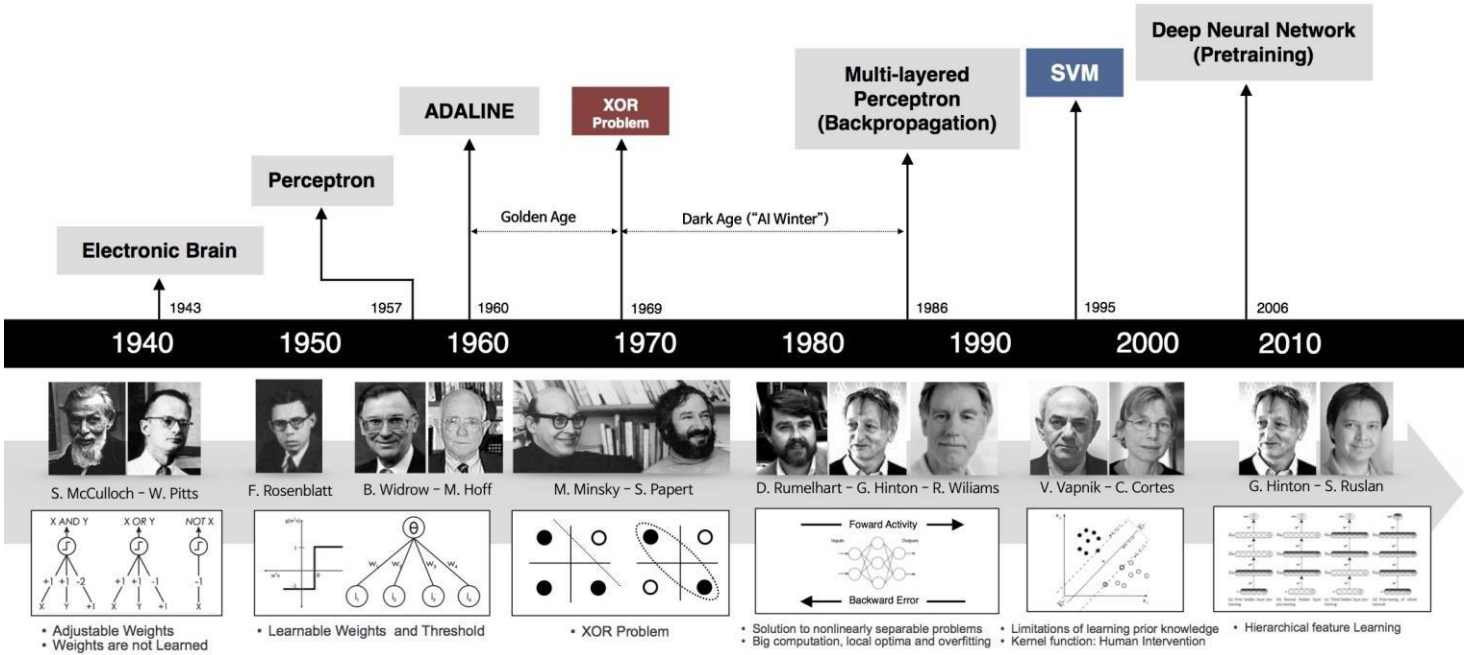
DEEP LEARNING

SITES ÚTEIS

- https://thispersondoesnotexist.com/

- https://theresanaiforthat.com/
- Capítulo 2 - Uma Breve História das Redes Neurais Artificiais - Deep Learning Book
- https://en.wikipedia.org/wiki/Lenna
- https://www.alura.com.br/conteudo/cnn-redes-neurais-convolucionais-deep-learning-pytorch?srsltid=AfmBOoqSGacjeh9s79i-xmsI_LgbQu5Gs6PZNSVP2dD-o8w7EBcVPuag

LINHA DO TEMPO

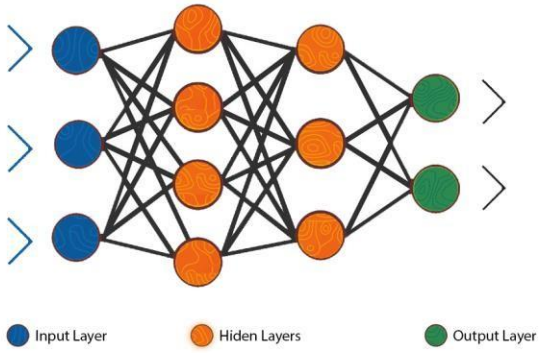


EVENTOS IMPORTANTES

1. **Workshop NIPS 200G**: Marcou um ponto de virada no aprendizado profundo, com pesquisadores apresentando avanços significativos na área e aumentando o interesse pelo tema.
2. **2012 - AlexNet**: A arquitetura de rede neural AlexNet, desenvolvida por Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton, revolucionou a visão computacional ao vencer a competição ImageNet, mostrando o potencial das redes profundas.
3. **2014-2015**:
    1. **Google DeepMind**: Avanços em inteligência artificial com redes neurais profundas, especialmente para jogos e reconhecimento de padrões complexos.
    2. **Facebook DeepFace**: Desenvolveu um sistema de reconhecimento facial com precisão próxima à humana, demonstrando o impacto das redes profundas em aplicações comerciais.
4. **2016 - AlphaGo**: O sistema de inteligência artificial AlphaGo, desenvolvido pela DeepMind, derrotou o campeão mundial Lee Sedol no jogo de tabuleiro Go, um marco na IA e no aprendizado profundo, mostrando a capacidade das máquinas de dominar jogos complexos.

ARQUITETURA DE REDES NEURAIS PROFUNDAS

- **Rede Neural Multicamadas (Multilayer Perceptron - MLP)**, composta por:
- **Camada de Entrada (Input Layer)**: Recebe os dados de entrada para serem processados.
- **Camadas Ocultas (Hidden Layers)**: Múltiplas camadas intermediárias onde ocorre o processamento e aprendizado dos padrões, com pelo menos duas camadas ocultas nessa rede. Essas camadas ajudam a modelar relações complexas nos dados.
- **Camada de Saída (Output Layer)**: Fornece o resultado final do processamento.



REDES CONVOLUCIONAIS

- As Redes Convolucionais são compostas por **unidades convolucionais**, que operam em **campos receptivos locais** para extrair características de imagens, como bordas e texturas. Essas redes são amplamente usadas para tarefas de visão computacional.
- **Arquiteturas conhecidas**: Incluem LeNet, AlexNet, ResNet, entre outras, que aplicam convoluções seguidas de camadas de pooling e conectam as características extraídas a camadas totalmente conectadas para classificação.
- **Funcionamento**:
  - **Camadas de Convolução**: Detectam padrões locais nas imagens.
  - **Camadas de Pooling (Max Pooling)**: Reduzem a dimensionalidade, preservando informações essenciais.
  - **Camadas totalmente conectadas**: Integram as características para classificação final, por exemplo, identificando objetos específicos em uma imagem.

- MAS ANTES, O QUE É CONVOLUÇÃO ?

- Para explicar **convolução** de forma bem didática, imagine o seguinte:
- Convolução é como um processo de "escaneamento" ou "filtro" aplicado em uma imagem para encontrar padrões específicos. Vamos pensar em como isso funciona passo a passo, usando uma analogia simples:

Analogia com uma Janela Móvel

- Imagine que você tem uma foto e coloca uma **janela pequena** sobre essa imagem. Essa janela vai cobrir uma parte pequena da foto. Agora, você olha através dessa janela e analisa apenas o que está dentro dela.

1. **Movimento da Janela**: Agora, você começa a mover essa janela para a direita, para baixo, para a esquerda, cobrindo diferentes partes da imagem. Cada vez que move a janela, você faz uma pequena análise do que está vendo naquela área.
2. **Detecção de Padrões**: Dentro dessa janela, você pode tentar encontrar certos padrões – como linhas horizontais, verticais, bordas, ou manchas escuras e claras. É como procurar algo específico em cada pedacinho da imagem.
3. **Criação de uma Nova Imagem Simplificada**: Quando termina de "escanear" toda a imagem com essa janela, você tem uma nova versão da imagem. Mas essa nova imagem não tem todos os detalhes; ela contém apenas os padrões que você encontrou com a janela.

Como a Convolução Ajuda

- A convolução, então, é esse processo de **aplicar um filtro ou janela** para encontrar padrões básicos em uma imagem. Na prática, é muito útil para identificar características visuais importantes, como os contornos de um rosto, as bordas de objetos, e até mesmo para distinguir entre diferentes objetos.

Exemplo na Vida Real

- Imagine que você está olhando para uma foto de uma zebra. Com uma janela de convolução, você poderia procurar padrões de listras. Cada vez que a janela se move, ela verifica se há listras na área que cobre. No final, o sistema terá uma "ideia" de onde estão as listras na imagem.

Resumo

- **Convolução** é como "escanear" uma imagem com uma janela pequena.
- Ela ajuda a encontrar **padrões específicos**, como bordas ou texturas.
- Ao juntar todos esses padrões, a rede neural consegue identificar objetos ou características na imagem completa.

REDES CONVOLUCIONAIS

- As Redes Convolucionais são compostas por **unidades convolucionais**, que operam em **campos receptivos locais** para extrair características de imagens, como bordas e texturas. Essas redes são amplamente usadas para tarefas de visão computacional.
- **Arquiteturas conhecidas**: Incluem LeNet, AlexNet, ResNet, entre outras, que aplicam convoluções seguidas de camadas de pooling e conectam as características extraídas a camadas totalmente conectadas para classificação.
- **Funcionamento**:
  - **Camadas de Convolução**: Detectam padrões locais nas imagens.
  - **Camadas de Pooling (Max Pooling)**: Reduzem a dimensionalidade, preservando informações essenciais.
  - **Camadas totalmente conectadas**: Integram as características para classificação final, por exemplo, identificando objetos específicos em uma imagem.

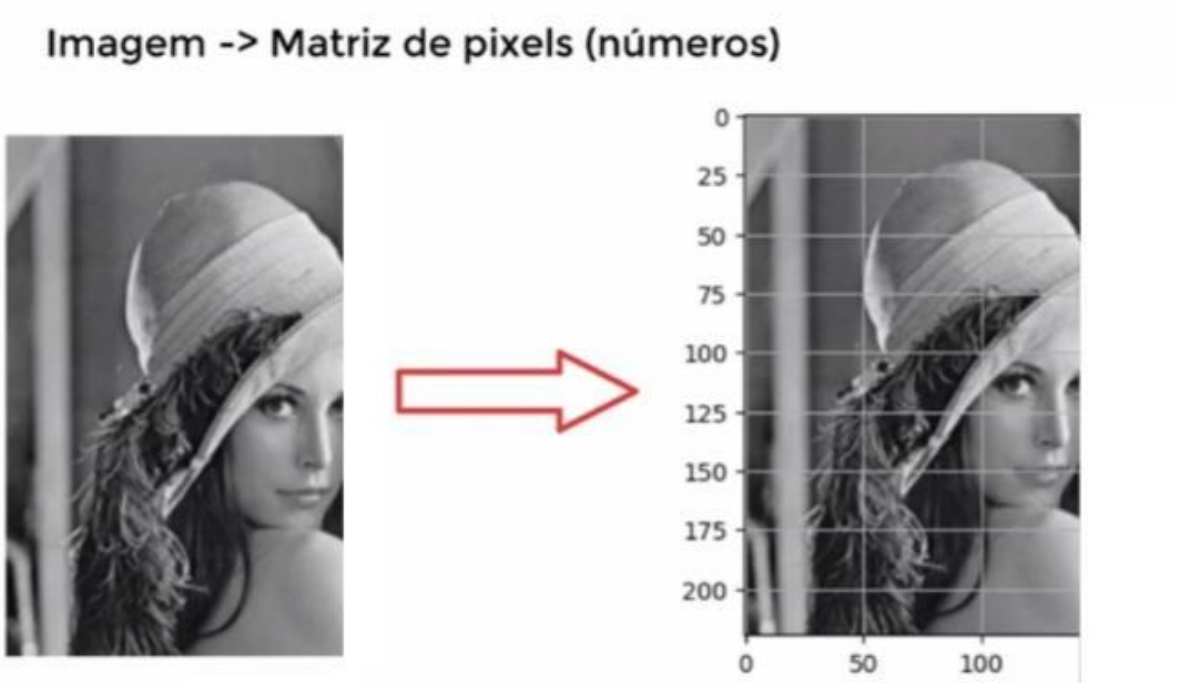
Redes Convolucionais (CNNs)

- Usam **campos receptivos locais**: cada neurônio se especializa em uma pequena região da entrada (ex: parte de uma imagem).
- **Não utilizam neurônios densos** nas camadas convolucionais; cada neurônio processa apenas uma área específica, não a imagem toda.
- Aplicam o mesmo neurônio em diferentes regiões, criando um mapeamento de padrões por toda a imagem.
- Estrutura típica: várias camadas convolucionais seguidas de camadas densas para classificação.
- Exemplos de arquiteturas: LeNet, AlexNet, ResNet, VGG.

Convolução:

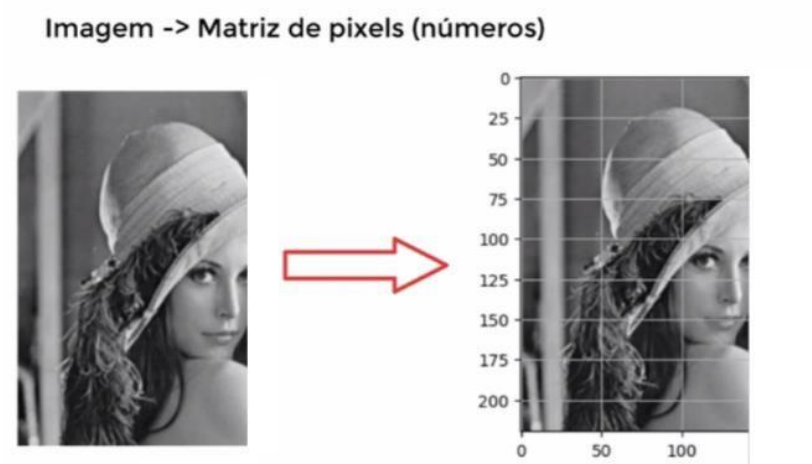
- Operação matemática que combina duas funções (f e g) para produzir uma terceira, frequentemente uma versão modificada de uma das funções originais.
- Em processamento de imagens, altera as intensidades dos pixels com base nos pixels ao redor, aplicando um filtro para criar efeitos como desfoque ou nitidez.

O QUE É UMA IMAGEM EM TERMOS COMPUTACIONAIS?



MATRIZ QUADRADA – EXEMPLO LENA

- Nesse caso, uma matriz quadrada de dimensões 256x256. que em outras palavras significa que são 256 números por 256 números, é uma matriz quadrada de números que nós vamos ver como isso se transforma numa imagem.



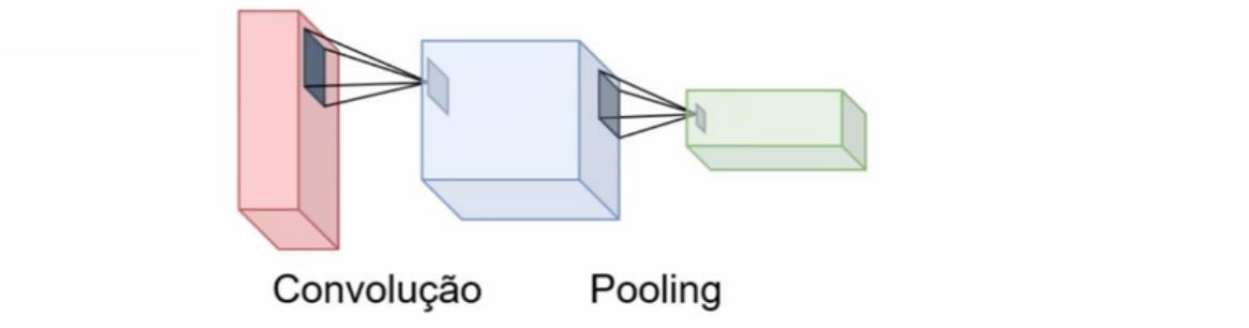
CADA PIXEL REPRESENTA UMA INTENSIDADE DE COR

- Então se nós pegarmos um trecho dessa imagem, o olho da Lena nós conseguimos ver nesse pedaço pequeno, os quadrados certos que são os pixels, que definem a imagem.

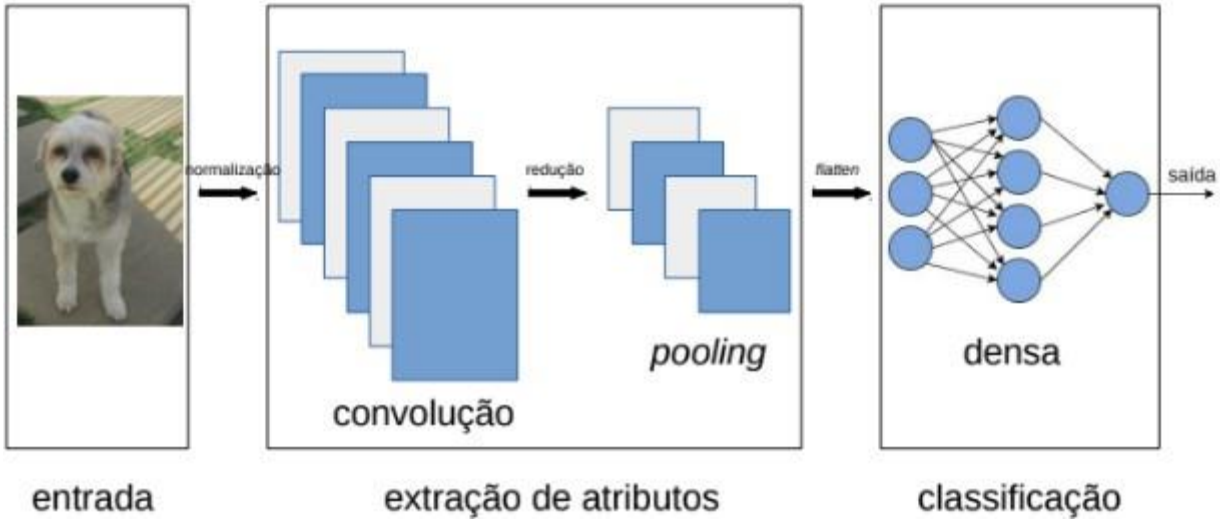
- E em termos simples esses pixels, esses quadrados são a intensidade da cor daquele elemento, daquele quadrados. Então se eu tenho uma matriz, nesse caso 15 por 15, que a região do olho em um determinado elemento tem valor de 0.1, está perto de 0 então representa algo mais próximo do preto, 0.9 está mais perto do 1 representa algo próximo do branco e 0.5, 0.6 está na faixa dos cinza.

Representação CNN

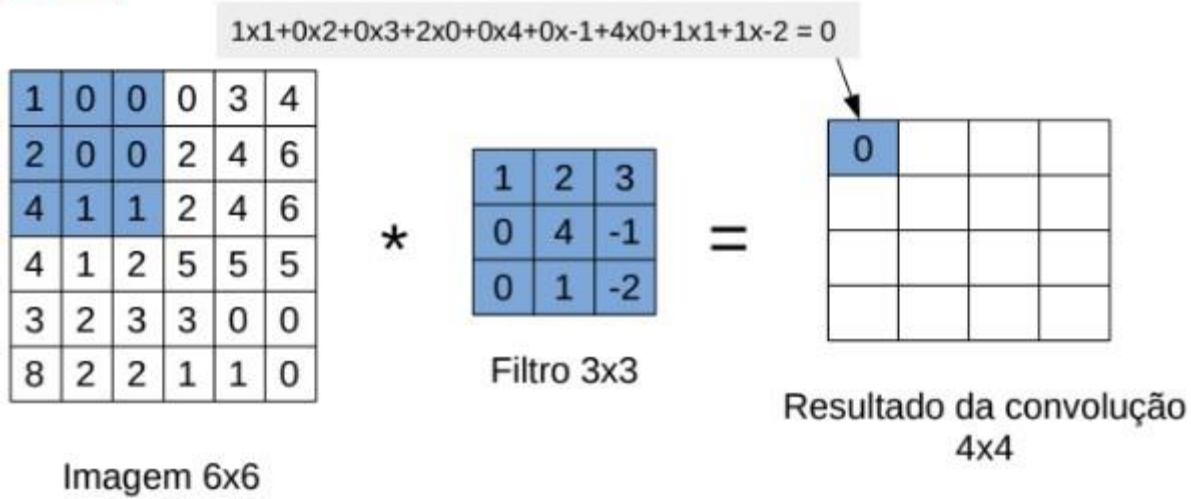
- Consegue trabalhar com volumes 3D (altura, largura e profundidade
  - Profundidade: canais de cor, mapa de características



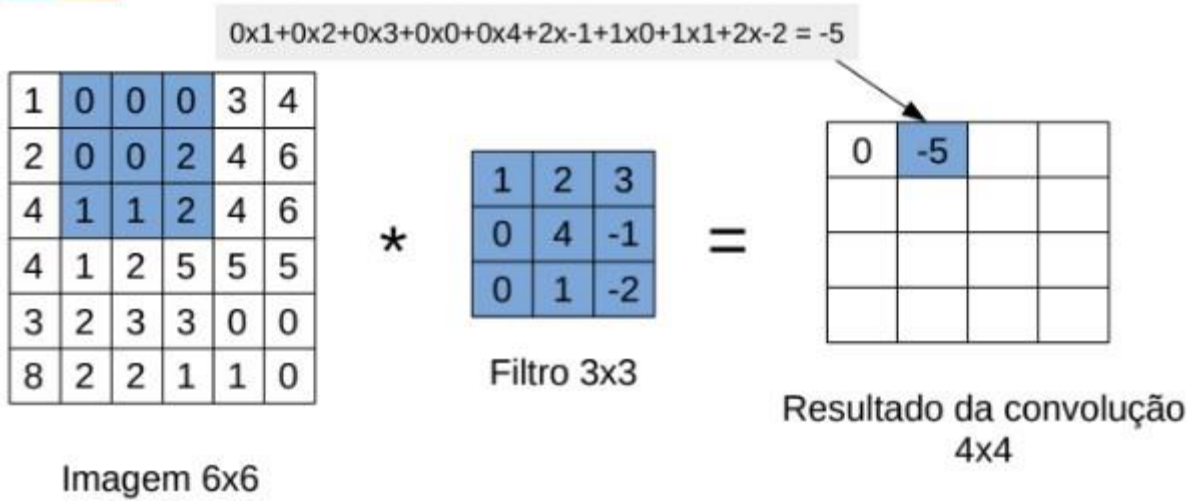
Camadas de uma CNN



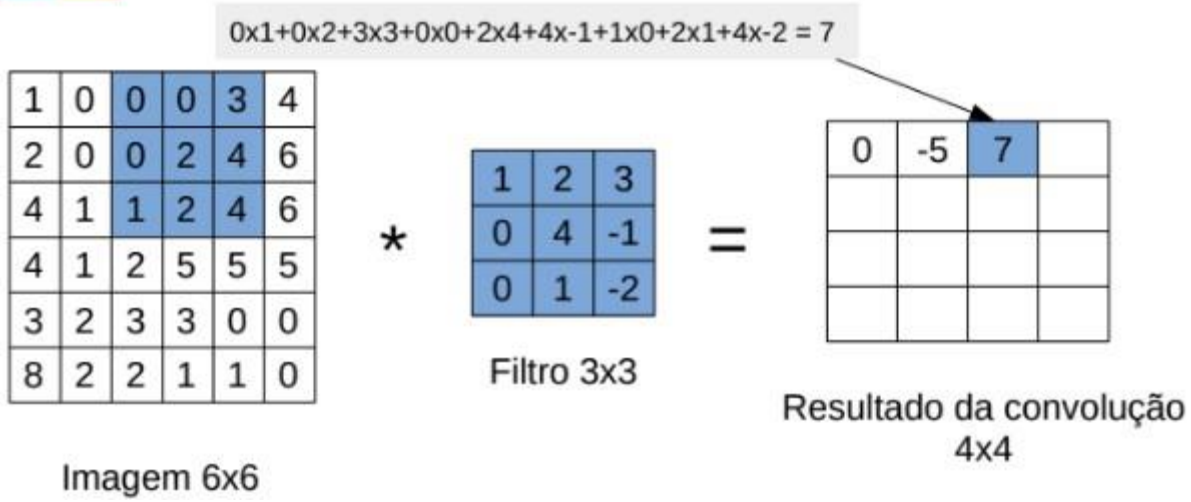
Operação de Convolução



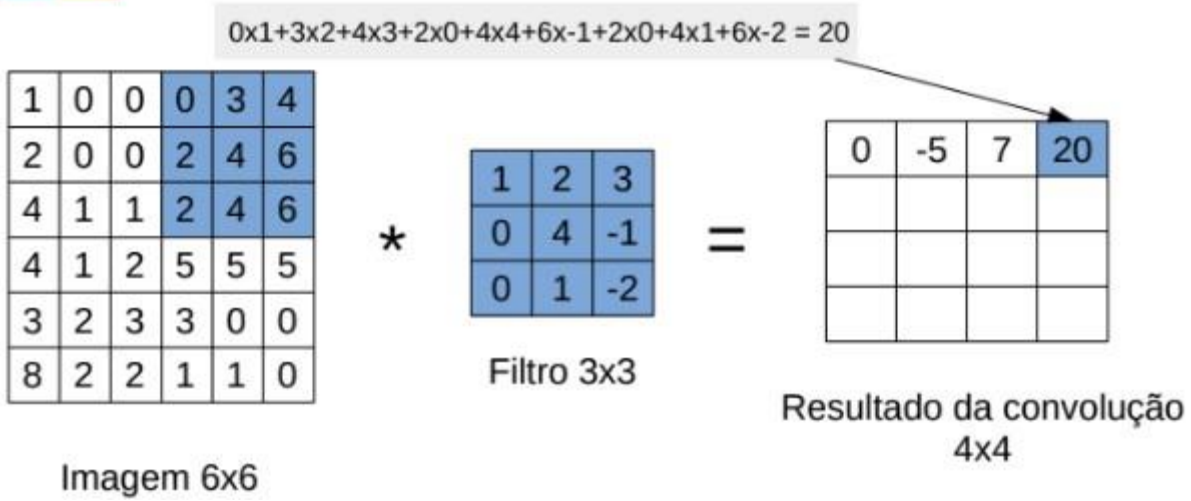
Operação de Convolução



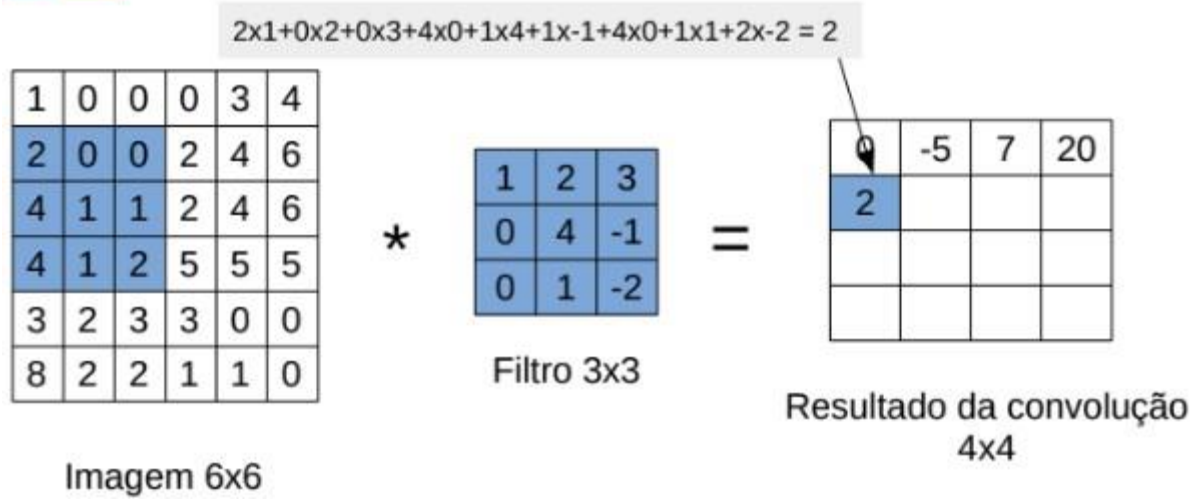
Operação de Convolução



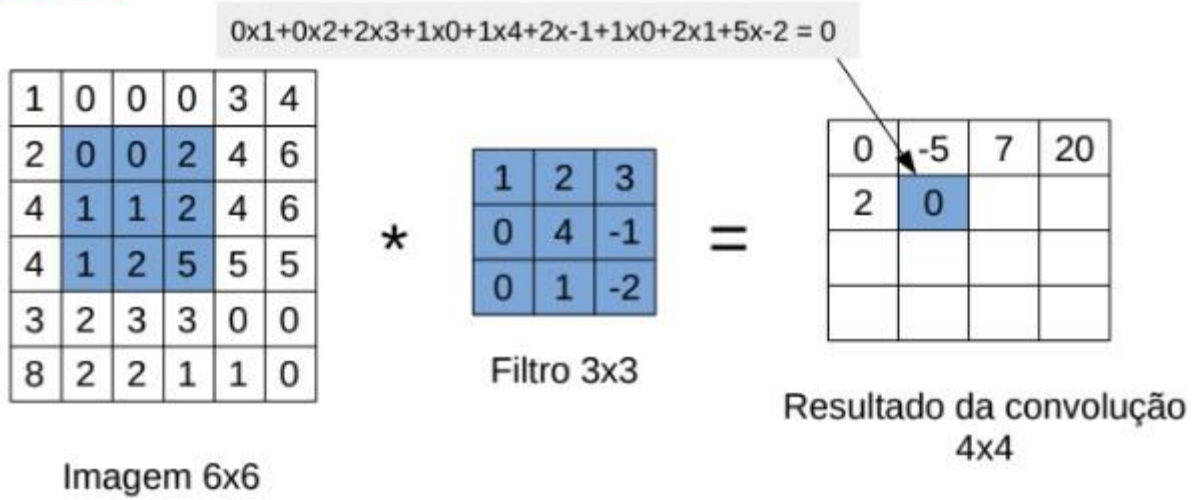
Operação de Convolução



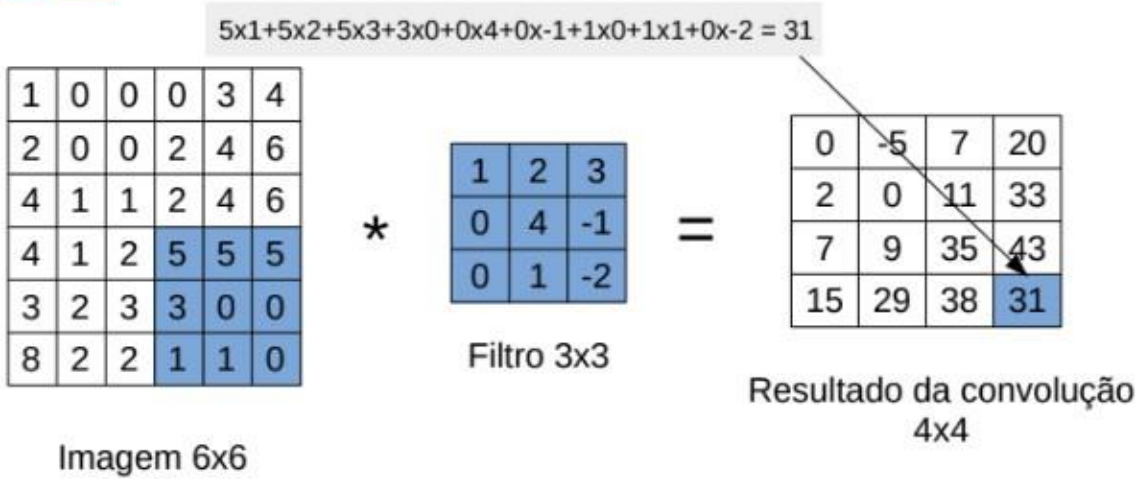
Operação de Convolução



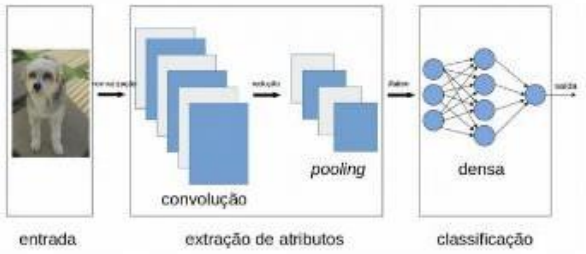
Operação de Convolução



Operação de Convolução



Detecção de Arestas



Aprendendo a Detectar Arestas

- Em imagens complexas, como especificar o melhor filtro?
- Deixe a rede neural descobrir os melhores parâmetros (pesos)

5	5	1	2	4	0
8	3	5	1	1	1
9	5	3	2	1	0
9	4	2	2	2	1
7	5	5	0	0	3
6	6	8	8	7	5

$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$=$

Operação de Convolução

1	0	0	0	3	4
2	0	0	2	4	6
4	1	1	2	4	6
4	1	2	5	5	5
3	2	3	3	0	0
8	2	2	1	1	0

Imagem 6x6

Imagem nxn

*

1	2	3
0	4	-1
0	1	-2

Filtro 3x3

Filtro fxf

=

0	-5	7	20
2	0	11	33
7	9	35	43
15	29	38	31

Resultado 4x4

n-f+1 x n-f+1

Problema: a imagem é reduzida

Operação de Convolução

1	0	0	0	3	4
2	0	0	2	4	6
4	1	1	2	4	6
4	1	2	5	5	5
3	2	3	3	0	0
8	2	2	1	1	0

*

1	2	3
0	4	-1
0	1	-2

=

0	-5	7	20
2	0	11	33
7	9	35	43
15	29	38	31

Problema: os pixels das bordas têm menos destaque

Padding

Preenchimento das bordas, nornalmente com valores iguais a zero

	1	0	0	0	3	4	
	2	0	0	2	4	6	
	4	1	1	2	4	6	
	4	1	2	5	5	5	
	3	2	3	3	0	0	
	8	2	2	1	1	0	

Imagem n x n = 6x6 ==> 8x8

*

1	2	3
0	4	-1
0	1	-2

Filtro f x f = 3x3

Padding p = 1

=

Preserva o tamanho original

	0	-5	7	20	
	2	0	11	33	
	7	9	35	43	
	15	29	38	31	

Resultado = 6 x 6

n+2p-f+1 x n+2p-f+1

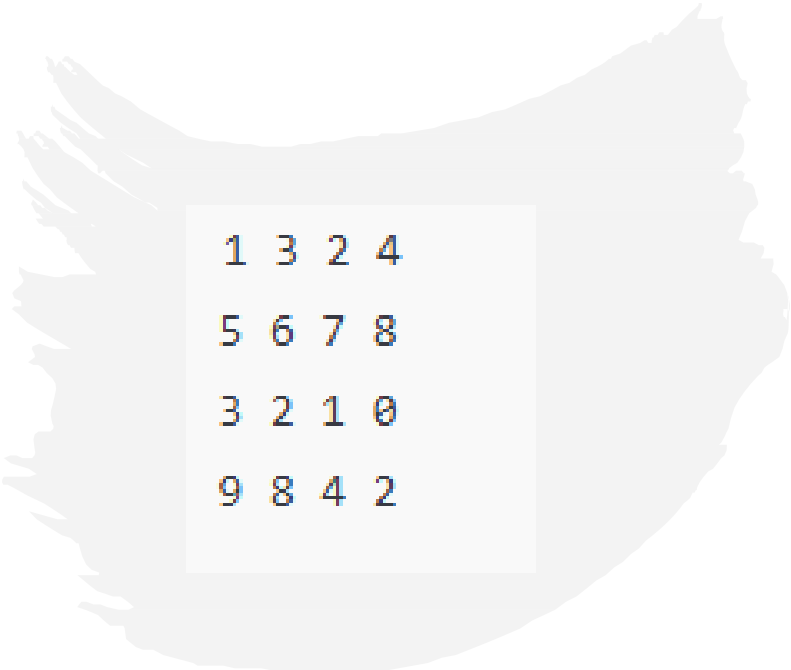
Pooling

- A camada de *pooling* em uma Rede Neural Convolucional (CNN) tem o objetivo de reduzir a dimensão espacial (altura e largura) dos dados, mantendo as informações mais importantes. Ela funciona como uma espécie de "resumo" das características extraídas pelas camadas convolucionais, permitindo que a rede aprenda de forma mais eficiente e com menos processamento.
- Aqui está uma explicação prática de como funciona o *pooling*:

1. **Redução de Dimensão**: O *pooling* é aplicado sobre uma pequena região da imagem (por exemplo, 2x2 pixels), movendo-se por toda a imagem. Para cada região, o *pooling* realiza uma operação simples, geralmente *máximo* ou *média*, para reduzir as informações.

1. **Tipos de Pooling**:
    1. **Max Pooling**: Seleciona o valor máximo dentro da janela de 2x2 pixels (ou do tamanho escolhido). Isso ajuda a destacar as características mais fortes (ou seja, bordas ou texturas intensas) e a eliminar pequenos ruídos.
    2. **Average Pooling**: Calcula a média dos valores na janela. É menos comum em CNNs, mas ainda é útil em algumas situações onde a suavização é preferível.

- **Como o Max Pooling Funciona:**
- Imagine uma matriz de entrada como uma imagem de 4x4 pixels, e você aplica uma janela de *max pooling* 2x2 com um "stride" (passo) de 2 pixels:



Como o Max Pooling

- O *max pooling* 2x2 seleciona o valor máximo em cada região de 2x2:Primeira região (canto superior esquerdo): 1, 3, 5, 6 →Máximo =6
- Segunda região (canto superior direito): 2, 4, 7, 8 →Máximo =8
- Terceira região (canto inferior esquerdo): 3, 2, 9, 8 →Máximo =9
- Quarta região (canto inferior direito): 1, 0, 4, 2 →Máximo =4

```
1 3 2 4
5 6 7 8
3 2 1 0
9 8 4 2
```

```
6 8
9 4
```

Benefícios do Pooling:

- **Redução de Complexidade**: Ao diminuir as dimensões, a rede processa menos dados, acelerando o treino e reduzindo o uso de memória.
- **Invariância a Translações**: Como o *pooling* pega o valor máximo ou a média em uma região, a rede se torna mais resistente a pequenas mudanças na posição das características da imagem, como bordas ou texturas.
- **Prevenção de Overfitting**: Reduzindo a quantidade de parâmetros e focando nas características principais, o *pooling* ajuda a rede a generalizar melhor em novos dados.

Max Pooling

Hiperparâmetros:
f = 3
s = 2
~~p = 0~~
Nenhum parâmetro

Máximo (1,0,0,2,0,0,4,1,1) = 4

1	0	0	0	3	4	1
2	0	0	2	4	6	2
4	1	1	2	4	6	3
4	1	2	5	5	5	6
3	2	3	3	0	0	5
8	2	2	1	1	0	4
7	1	1	2	2	1	3

Imagem 7x7

Filtro 3x3
Stride = 2

4		

Resultado 3x3

$$\frac{n+2p-f}{s}+1$$

Diferença entre Convolução e Correlação Cruzada:

- **Convolução**: Usada para modificar ou extrair características da imagem através da resposta de um sinal a outro.
- **Correlação Cruzada**: Mede a correspondência entre sinais, verificando a similaridade entre duas sequências de dados.

- **Filtros (ou Kernels)**:
- Pequena matriz aplicada sobre a imagem para extrair padrões específicos.
- Passos para aplicar um filtro:
  - Multiplicar cada elemento do filtro pelo pixel correspondente na imagem.
  - Somar os resultados.
  - Normalizar (opcional), dividindo pelo total de elementos do filtro.

Exemplos comuns:

- Detecção de bordas (realça transições de intensidade).
- Desfoque (suaviza a imagem).

HISTÓRIA LENNA

- A imagem "Lenna" é uma foto icônica no campo do processamento de imagens digitais. Ela foi tirada da edição de novembro de 1972 da revista Playboy e representa a modelo sueca Lena Söderberg. Desde 1973, a imagem é amplamente usada para testar algoritmos de processamento de imagem, sendo conhecida por suas características técnicas adequadas, como detalhes, texturas e sombreamento.
- Ao longo dos anos, o uso dessa imagem gerou controvérsia devido ao seu vínculo com a Playboy, e várias instituições passaram a desencorajar ou proibir seu uso, considerando-a inapropriada. Lena Forsén, a modelo, também manifestou desejo de aposentar a imagem. Em 2019, ela afirmou no documentário "Losing Lena" que era hora de a tecnologia seguir em frente e se desvincular da sua imagem.
- Além das críticas éticas, a imagem Lenna também foi questionada pela sua qualidade como padrão, especialmente em tempos de tecnologias avançadas de alta resolução.
- Em 2024, o IEEE anunciou que não aceitará mais o uso da imagem em suas publicações.

Camada de Convolução em Redes Neurais

- Cada camada aplica múltiplos filtros para extrair características da imagem em diferentes níveis (bordas, texturas, etc.).
- A camada gera múltiplos "mapas de ativação", que são representações da imagem com diferentes características.

ReLU (Rectified Linear Unit):

- Função utilizada para introduzir não-linearidade na rede, definindo ReLU(x) =max(0, x).Converte valores negativos em zero, permitindo que a rede aprenda padrões mais complexos e evite problemas de
- retropropagação com valores negativos.

Como Funciona a ReLU

A função ReLU é simples:

- Para uma entrada x, a saída da ReLU é:

$$\text{ReLU}(x) = \max(0, x)$$

- Em outras palavras, se o valor de entrada é **positivo**, ele é mantido. Se é **negativo**, ele é convertido para **zero**.

1. **Exemplo com Números Positivos e Negativos:**
    - Suponha que temos uma lista de valores: `[-2, -1, 0, 1, 2, 3]`.
    - Aplicando a função ReLU:
        - `ReLU(-2) = 0`
        - `ReLU(-1) = 0`
        - `ReLU(0) = 0`
        - `ReLU(1) = 1`
        - `ReLU(2) = 2`
        - `ReLU(3) = 3`
    - Resultado: `[0, 0, 0, 1, 2, 3]`

2. **Por Que Usamos ReLU em Redes Neurais?**

   - Sem funções de ativação, uma rede neural seria equivalente a uma função linear (ou seja, muito limitada em sua capacidade de aprendizado).
   - ReLU introduz uma **não-linearidade** que permite à rede aprender padrões complexos.
   - Comparada com outras funções de ativação, como **sigmoid** e **tanh**, ReLU tem a vantagem de ser computacionalmente eficiente e não sofre tanto com o problema de **gradiente desvanecente** (o que facilita o treinamento de redes profundas).

3. **Aplicação da ReLU em Imagens**:
    - Suponha que aplicamos uma convolução em uma imagem e o resultado é uma matriz de valores que inclui números negativos (exemplo: `[-1, -0.5, 0, 1, 1.5]`).
    - Depois de aplicar ReLU, os valores negativos são removidos:
        - Exemplo: `[0, 0, 0, 1, 1.5]`
    - Esse processo ajuda a destacar somente as ativações importantes, eliminando ruído desnecessário na imagem.

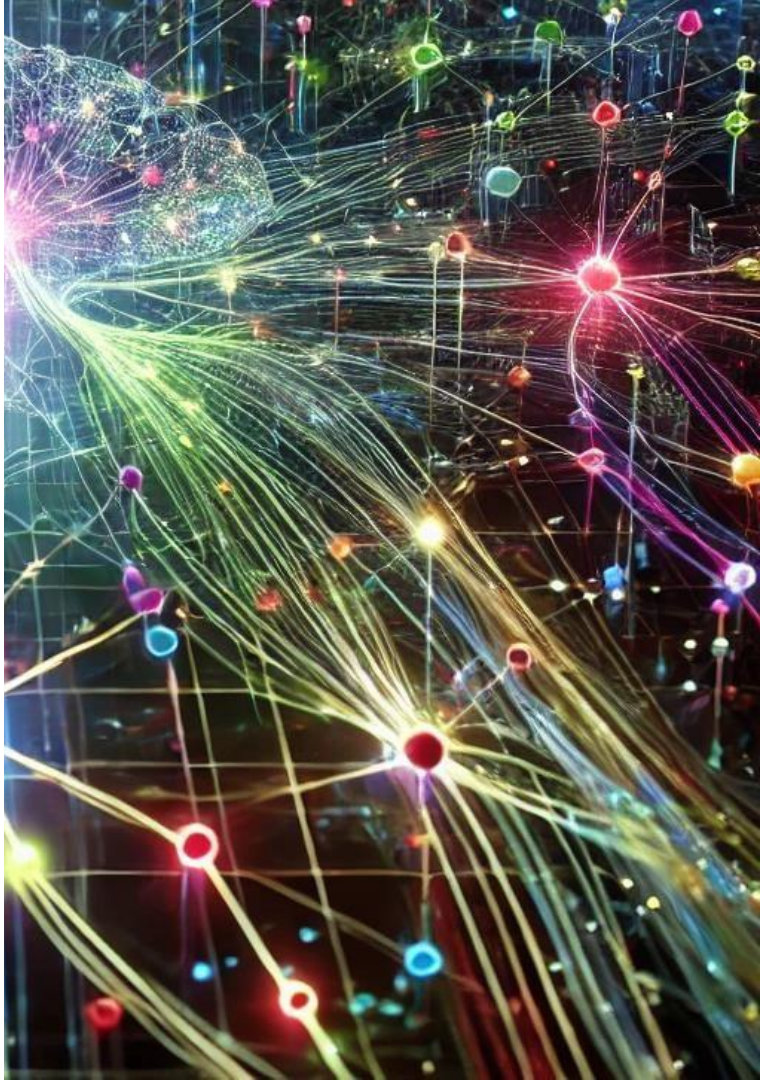
Vantagens da ReLU

1. **Simplicidade e Eficiência**: A ReLU é muito rápida e fácil de calcular.
2. **Mitigação do Problema de Gradiente Desvanecente**: Em redes profundas, funções como sigmoid e tanh podem reduzir o gradiente a valores muito pequenos, dificultando o treinamento. A ReLU mantém o gradiente mais forte para valores positivos.
3. **Capacidade de Representação**: A ReLU permite que a rede represente padrões complexos, pois cada camada pode aprender algo novo e não-linear a partir da camada anterior.

Desvantagens e Soluções

1. **Morte de Neurônios**: Alguns neurônios podem "morrer" durante o treinamento (ficam presos em zero) se receberem constantemente entradas negativas. Para resolver isso, surgiram variações como **Leaky ReLU** e **Parametric ReLU** que permitem valores negativos levemente atenuados, em vez de zeros.

OUTRAS REDES



Redes Recorrentes (RNNs)

- Criam uma **memória** ao longo do tempo: o estado de cada neurônio depende do estado anterior e das entradas atuais.
- Usadas para dados sequenciais, onde o contexto do passado influencia a interpretação do presente.
- **Memórias de curto prazo** em RNNs padrão; para memórias de longo prazo, utilizam-se células LSTM e GRU.
- Permitem aprendizado de **sequências longas**, associando partes iniciais de uma sequência a partes posteriores.

Deep Belief Networks (DBNs)

- Introduzidas por Geoffrey Hinton (2006).
- Baseadas em redes RBF (Redes de Funções de Base Radial) treinadas camada por camada e depois concatenadas para formar uma rede profunda.
- Treinamento em duas etapas: cada camada é treinada individualmente e, depois, ajusta-se a rede completa com retropropagação.
- Eficazes para **mapear representações complexas**, superando métodos tradicionais da época, como Análise de Componentes Principais (PCA).

Autoencoders

- Estrutura composta por duas partes: **codificador** (encoder) e **decodificador** (decoder).
- O codificador mapeia a entrada para um **espaço latente** (representação compacta).
- O decodificador reconstrói a saída a partir da representação latente; em alguns casos, a saída é a própria entrada (modelo autossupervisionado).
- Usados em tarefas de **extração de características** e como base para modelos generativos (ex: geração de texto e imagens).

Redes Adversariais Generativas (GANs)

- Compostas por duas redes: um **gerador** e um **discriminador**.
- O gerador tenta criar dados que pareçam reais, enquanto o discriminador tenta diferenciar entre dados reais e gerados.
- Treinamento adversarial: as duas redes competem entre si, o que melhora a capacidade do gerador de criar dados realistas.
- Amplamente utilizadas em **geração de conteúdo**, como imagens realistas, síntese de voz e criação de arte.

PARA MELHOR VISUALIZAR REDES

- a possibilidade de "percorrer" cada neurônio e camada:

1. **TensorFlow Playground**
    1. **URL**: playground.tensorflow.org
    2. **Descrição**: Esta ferramenta interativa permite explorar redes neurais simples com até duas camadas escondidas. É possível ajustar a estrutura da rede, selecionar hiperparâmetros e observar como o modelo aprende com diferentes conjuntos de dados em tempo real. Embora limitada a redes pequenas, é ótima para visualização de como cada camada afeta o aprendizado.
2. **Netron**
    1. **URL**: Netro netron.app
    2. **Descrição**: Netron é uma ferramenta de visualização para modelos de redes neurais que suporta vários formatos de arquivos (ONNX, TensorFlow, Keras, PyTorch, etc.). Com ele, é possível carregar e visualizar o modelo como um gráfico de blocos, expandir cada camada e observar os detalhes de cada parâmetro. Não tem animação, mas é excelente para explorar estruturas complexas camada por camada.
3. **Deep Playground (Deep Visualization Toolbox)**

PARA MELHOR VISUALIZAR REDES

1. **URL**: Deep Visualization Toolbox
2. **Descrição**: Esta ferramenta oferece uma visualização detalhada das ativações em redes neurais, incluindo o efeito de cada camada. É particularmente interessante para CNNs, pois permite ver o que cada filtro convolucional está aprendendo. Oferece animações e permite percorrer a rede, observando o que cada neurônio responde.

1. **LSTM Vis (LSTM Neural Network Visualizer)**
    1. **URL**: LSTM Vis
    2. **Descrição**: Esta ferramenta é ideal para redes neurais recorrentes (RNNs) e LSTMs. Ela permite observar as ativações e estados ocultos em cada célula LSTM, acompanhando como a rede processa sequências temporais. É interativa, então você pode explorar o funcionamento camada a camada.

2. **Google Colab com TensorBoard**
    1. **URL**: Google Colab
    2. **Descrição**: Com o TensorBoard integrado ao Google Colab, é possível visualizar redes neurais complexas com gráficos interativos e explorar camadas individualmente. TensorBoard permite visualizar tanto a estrutura quanto o treinamento da rede, incluindo pesos, gradientes e ativações de camadas em modelos profundos.