



Convolutional Neural Networks

CPE 727 - Aprendizado Profundo

Brenno Rodrigues de Carvalho da Silva, Gabriel Guimarães, Lucas

Alexandre, Raphael Nagao (brennorcs@poli.ufrj.br,
guimaraes921@poli.ufrj.br, lucas.alexandre@coppe.ufrj.br,
rnanagao@cos.ufrj.br)

7 de dezembro de 2025

Sumário

1 Revisão: Redes Convolucionais (CNNs)

- ▶ Revisão: Redes Convolucionais (CNNs)
- ▶ A matemática por trás
- ▶ Aritmética do Pooling
- ▶ Convolução Transposta
- ▶ Modelos Alternativos
- ▶ Conclusões
- ▶ Referências Bibliográficas

O que são CNNs?

1 Revisão: Redes Convolucionais (CNNs)

- CNNs (Convolutional Neural Networks) são redes neurais projetadas para processar dados com estrutura espacial, como imagens.

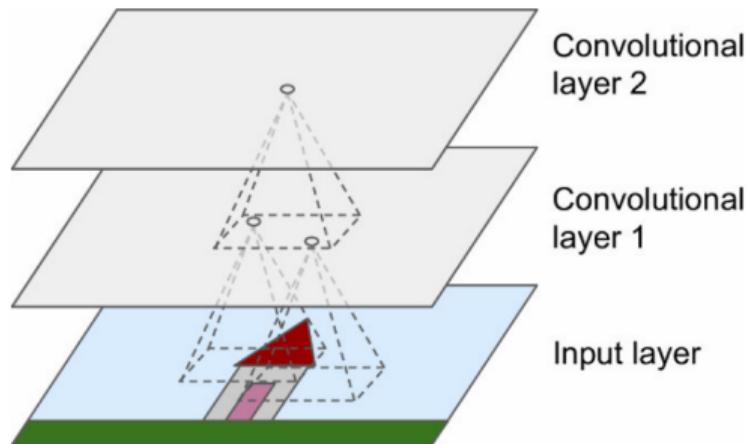


Figura: Camadas das CNN [1].

O que são CNNs?

1 Revisão: Redes Convolucionais (CNNs)

- Compostas por:
 - Camadas **convolucionais** (extração de padrões locais)
 - Camadas de **pooling** (redução de dimensionalidade)
 - Camadas densas (classificação)

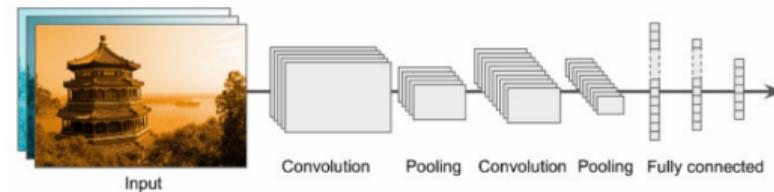


Figura: Arquitetura típica de CNN [1].

Camadas Convolucionais

1 Revisão: Redes Convolucionais (CNNs)

- Aplicam **filtros** (kernels) sobre a imagem para gerar mapas de ativação.
- Cada filtro aprende a detectar um padrão específico (bordas, texturas, etc).
- Resulta em mapas de ativação (feature maps).

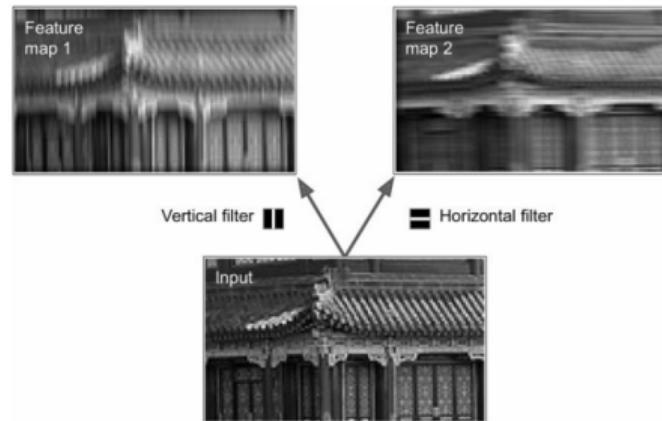
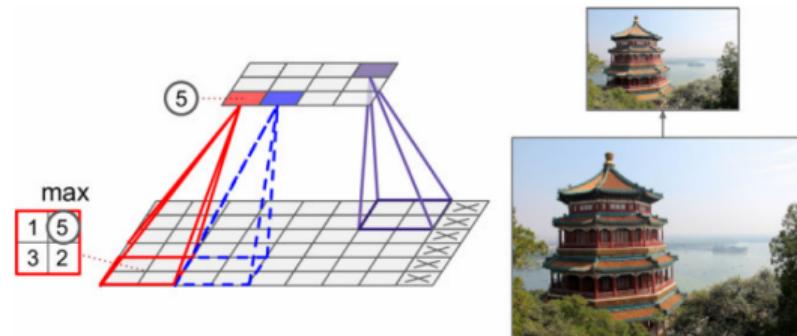


Figura: Aplicando 2 filtros diferentes para obter 2 mapas de características. [1].

Pooling e suas limitações

1 Revisão: Redes Convolucionais (CNNs)

- Reduz as dimensões dos mapas de ativação
- Os dois tipos mais comuns:
 - Max pooling: obtém o valor máximo da região.
 - Average Pooling: obtém a média dos valores da região.



Problema:
perde
informações
espaciais
importantes.

Figura: Camada de *max pooling* (*kernel pooling* de 2×2 , *stride* de 2, sem uso de [1]).

Sumário

2 A matemática por trás

- ▶ Revisão: Redes Convolucionais (CNNs)
- ▶ A matemática por trás
- ▶ Aritmética do Pooling
- ▶ Convolução Transposta
- ▶ Modelos Alternativos
- ▶ Conclusões
- ▶ Referências Bibliográficas



O que é a Convolução Discreta

2 A matemática por trás

Em redes neurais tradicionais (totalmente conectadas), geralmente "achatamos" os dados de entrada, como uma imagem, em um único vetor longo. Isso ignora uma propriedade fundamental desses dados: a estrutura intrínseca. Imagens, por exemplo, têm eixos (altura, largura) onde a ordem importa, e canais (como R, G, B) que representam diferentes "visões" dos dados.



O que é a Convolução Discreta

2 A matemática por trás

A convolução discreta é uma transformação linear projetada especificamente para preservar e explorar essa noção de ordem.

Diferente de uma transformação afim (multiplicação por matriz densa), a convolução é:

- **Esparsa:** Apenas algumas unidades de entrada contribuem para uma determinada unidade de saída.
- **Reutiliza parâmetros:** Os mesmos pesos (o "kernel") são aplicados em múltiplos locais da entrada.



O Mecanismo da Convolução

2 A matemática por trás

O processo funciona da seguinte maneira:

- 1. Mapa de Características de Entrada (Input):** A entrada da camada (por exemplo, uma imagem).
- 2. Kernel (Filtro):** Um pequeno conjunto de pesos (ex: 3x3) que "desliza" sobre o mapa de entrada.
- 3. Operação:** Em cada posição, calculamos o produto elemento a elemento entre o kernel e a porção da entrada que ele sobrepõe.
- 4. Soma:** Os resultados desses produtos são somados para gerar um único valor de saída.
- 5. Mapa de Características de Saída (Output):** Esse valor se torna um "pixel" no mapa de saída. O processo se repete até o kernel percorrer toda a entrada, formando um novo mapa.

O Mecanismo da Convolução

2 A matemática por trás

0	1	2
2	2	0
0	1	2

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₂	0 ₁	1 ₀	3 ₁	1 ₀
3 ₀	1 ₁	2 ₂	2 ₃	3 ₄
2 ₀	0 ₀	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	1 ₁	2 ₂	3 ₃	1 ₄
3 ₂	1 ₁	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	1 ₁	2 ₂	3 ₃	1 ₄
3 ₀	1 ₁	2 ₂	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

Figura: Kernel [2]

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	0 ₁	1 ₂	3 ₃	1 ₄
3 ₁	1 ₂	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	0 ₁	1 ₂	3 ₃	1 ₄
3 ₁	1 ₂	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	1 ₁	2 ₂	3 ₃	1 ₄
3 ₁	1 ₂	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	0 ₁	1 ₂	3 ₃	1 ₄
3 ₁	1 ₂	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	0 ₁	1 ₂	3 ₃	1 ₄
3 ₁	1 ₂	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	1 ₁	2 ₂	3 ₃	1 ₄
3 ₁	1 ₂	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

3 ₀	3 ₁	2 ₂	1 ₃	0 ₄
0 ₀	1 ₁	2 ₂	3 ₃	1 ₄
3 ₁	1 ₂	2 ₀	2 ₃	3 ₄
2 ₀	0 ₁	0 ₂	2 ₂	2 ₄
2 ₀	0 ₀	0 ₀	0 ₁	1 ₄

Figura: Convolução 2D [2]



A Aritmética da Convolução

2 A matemática por trás

Como prever o tamanho da saída de uma camada convolucional?

O tamanho da saída (o) é determinado por quatro hiperparâmetros:

- i (Input Size): O tamanho (altura ou largura) da entrada;
- k (Kernel Size): O tamanho (altura ou largura) do kernel;
- s (Stride): O "passo", ou seja, a distância que o kernel se move entre operações;
- p (Zero Padding): O número de zeros adicionados nas bordas da entrada.



Caso 1: Sem Padding ($p = 0$), Strides Unitários ($s = 1$)

2 A matemática por trás

Este é o caso mais básico. O kernel (k) desliza pela entrada (i) um pixel de cada vez.

$$o = (i - k) + 1 \quad (1)$$

Exemplo (Baseado na Fig 7):

- Input $i = 4$ (uma matriz 4×4)
- Kernel $k = 3$ (uma matriz 3×3)
- Stride $s = 1$,
- Padding $p = 0$
- Saída $o = (4 - 3) + 1 = 2$.

Caso 1: Sem Padding ($p = 0$), Strides Unitários ($s = 1$)

2 A matemática por trás

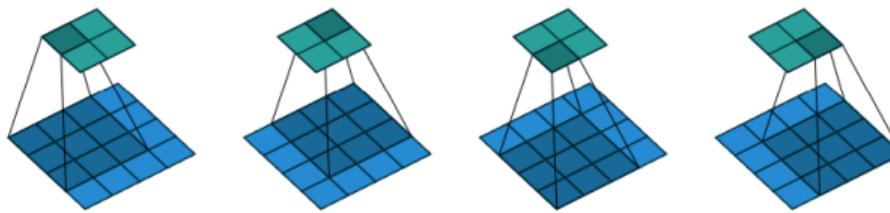


Figura: Convolução Kernel 3x3 com uma entrada 4x4 sem padding [2]



Caso 2: Padding ($p > 0$), Strides Unitários ($s = 1$)

2 A matemática por trás

O padding adiciona zeros ao redor da entrada. Se adicionarmos p zeros de cada lado, o "tamanho efetivo" da entrada se torna $i + 2p$. A fórmula geral é ajustada:

$$o = (i + 2p - k) + 1 \quad (2)$$

Exemplo (Baseado na Fig 8):

- Input $i = 5$ (uma matriz 5×5)
- Kernel $k = 4$ (uma matriz 4×4)
- Stride $s = 1$,
- Padding $p = 2$
- Saída $o = (5 + 2 \cdot 2 - 4) + 1 = 6$.

$$o = (i + 2p - k) + 1 \quad (3)$$

Caso 2: Padding ($p = 2$), Strides Unitários ($s = 1$)

2 A matemática por trás

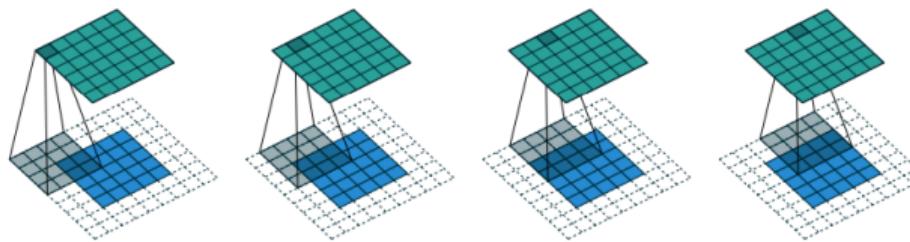


Figura: Convolução Kernel 4x4 com uma entrada 5x5 com padding = 2 [2]



Caso 3: Half padding Strides Unitários ($s = 1$)

2 A matemática por trás

Uma propriedade muito desejável é fazer com que a saída tenha o mesmo tamanho da entrada ($o = i$). Isso é comum em arquiteturas modernas (como ResNets) para manter as dimensões espaciais. Isso é alcançado (quando $s = 1$ e k é ímpar ($k = 2n + 1$)) definindo o padding como: $p = \lfloor k/2 \rfloor$

$$o = i + 2(k/2) - (k - 1) = i + 2n - 2n = i \quad (4)$$

Caso 3: Half padding Strides Unitários ($s = 1$)

2 A matemática por trás

Exemplo (Baseado na Fig 10):

- Input $i = 5$ (uma matriz 5×5)
- Kernel $k = 3$ (uma matriz 3×3)
- Stride $s = 1$,
- Padding $p = \lfloor 3/2 \rfloor = 1$.

$$o = (5 + 2 \cdot 1 - 3) + 1 = (5 + 2 - 3) + 1 = 5 \quad (5)$$

Caso 3: Half padding Strides Unitários ($s = 1$)

2 A matemática por trás

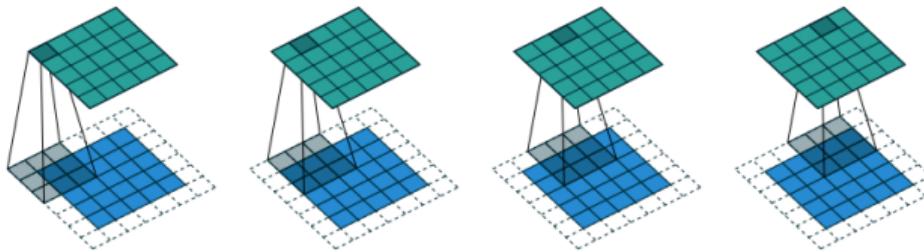


Figura: Convolução Kernel 3x3 com uma entrada 5x5 com stride = 1 e half padding [2]

Caso 4: Sem Padding ($p = 0$), Strides Não Unitários ($s > 1$)

2 A matemática por trás

Quando o stride (passo) é maior que 1 ($s > 1$), o kernel "pula" pela entrada. Isso funciona como uma forma de subamostragem (downsampling). Como o kernel dá passos de tamanho s , o número de posições possíveis diminui.

$$o = \left\lfloor \frac{i - k}{s} \right\rfloor + 1 \quad (6)$$

Caso 4: Sem Padding ($p = 0$), Strides Não Unitários ($s > 1$)

2 A matemática por trás

Exemplo (Baseado na Fig 10):

- Input $i = 5$ (uma matriz 5×5)
- Kernel $k = 3$ (uma matriz 3×3)
- Stride $s = 2$,
- Padding $p = 0$.
- $\lfloor \cdot \rfloor$ representa a função piso.

$$o = \left\lfloor \frac{5 - 3}{2} \right\rfloor + 1 = 2 \quad (7)$$

Caso 4: Sem Padding ($p = 0$), Strides Não Unitários ($s > 1$)

2 A matemática por trás

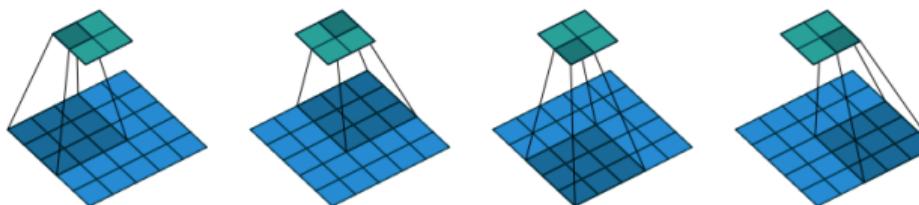


Figura: Convolução Kernel 3x3 com uma entrada 5x5 com stride = 2 e padding = 0 [2]



Caso Geral

2 A matemática por trás

Finalmente, podemos combinar todos os parâmetros (padding e strides não unitários) para obter a relação mais geral.

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1 \quad (8)$$

Sumário

3 Aritmética do Pooling

- ▶ Revisão: Redes Convolucionais (CNNs)
- ▶ A matemática por trás
- ▶ Aritmética do Pooling
- ▶ Convolução Transposta
- ▶ Modelos Alternativos
- ▶ Conclusões
- ▶ Referências Bibliográficas

Pooling

3 Aritmética do Pooling

O *pooling* é uma operação utilizada para reduzir a dimensionalidade espacial dos mapas de ativação, com objetivo de diminuir o custo computacional.

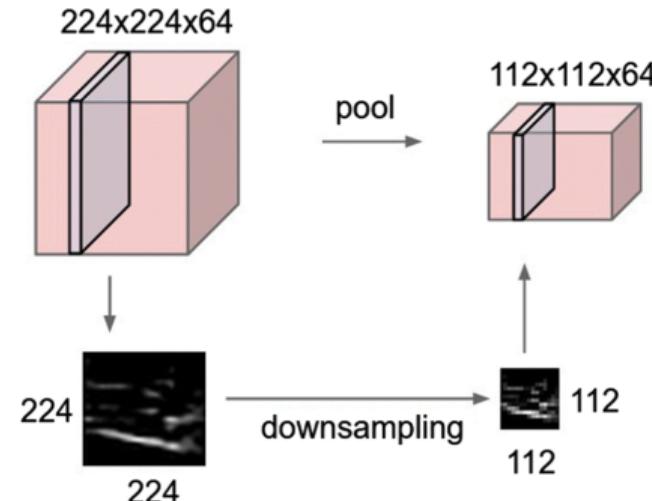


Figura: Exemplo de camada de *pooling*



Tipos de Pooling

3 Aritmética do Pooling

As camadas de pooling resumem as informações presentes em pequenas regiões da imagem.

Os tipos mais comuns são:

- 1. Max Pooling:** Seleciona o maior valor dentro da região analisada. Preserva as características mais intensas (bordas, texturas fortes).
- 2. Average Pooling:** Calcula a média dos valores dentro da região. Produz uma representação mais suave, útil quando se deseja manter uma noção geral da região.



Parâmetros que determinam o tamanho da saída

3 Aritmética do Pooling

O tamanho da saída de uma camada de *pooling* depende de três valores:

- **Input Size**: tamanho (altura ou largura) da entrada;
- **(Kernel Size)**: tamanho da janela de pooling;
- **(Stride)**: distância que a janela se desloca entre cada operação.

Consideraremos **padding = 0** (sem adição de bordas).



Fórmula

3 Aritmética do Pooling

A dimensão de saída (o) é dada por:

$$o = \left\lfloor \frac{i - k}{s} \right\rfloor + 1 \quad (9)$$

Onde:

- i é o tamanho da entrada;
- k é o tamanho da janela;
- s é o passo de deslizamento (*stride*);
- $\lfloor \cdot \rfloor$ representa a função piso.

Essa fórmula é aplicada separadamente para altura e largura.

Exemplos

3 Aritmética do Pooling

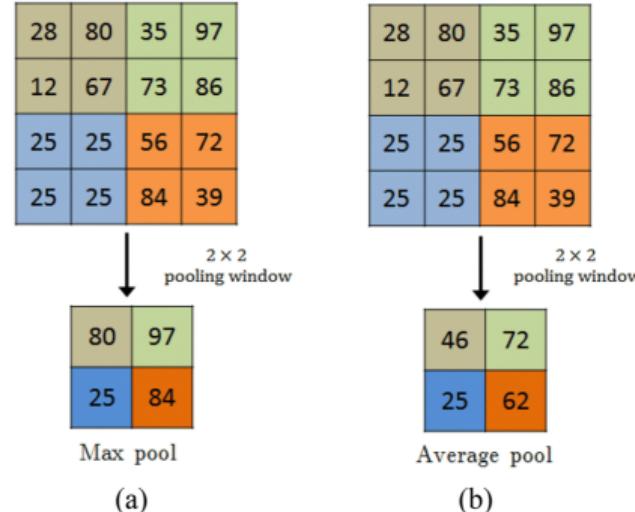


Figura: Exemplos de Max Pool e Average Pool

[?]

Sumário

4 Convolução Transposta

- ▶ Revisão: Redes Convolucionais (CNNs)
- ▶ A matemática por trás
- ▶ Aritmética do Pooling
- ▶ **Convolução Transposta**
- ▶ Modelos Alternativos
- ▶ Conclusões
- ▶ Referências Bibliográficas

O que é uma Convolução Transposta?

4 Convolução Transposta

A convolução transposta (ou DCNN) é a operação inversa que permite que as redes neurais aumentem o tamanho dos dados, transformando uma entrada de baixa dimensão em uma saída de alta dimensão, o oposto do que uma convolução padrão faz.[2]

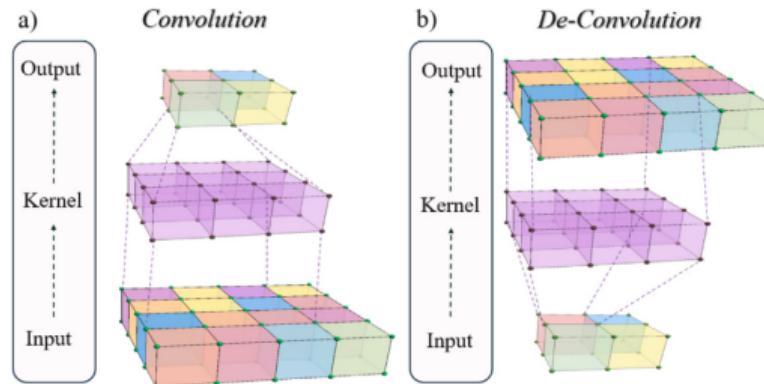


Figura: Exemplo de convolução transposta [4]

Objetivo

4 Convolução Transposta

- Reverter a Convolução
- Projeção para Maior Dimensão
- Decodificação e Reconstrução
- Resolução de Tarefas de Baixo Nível
- Expansão da Dimensão de Características

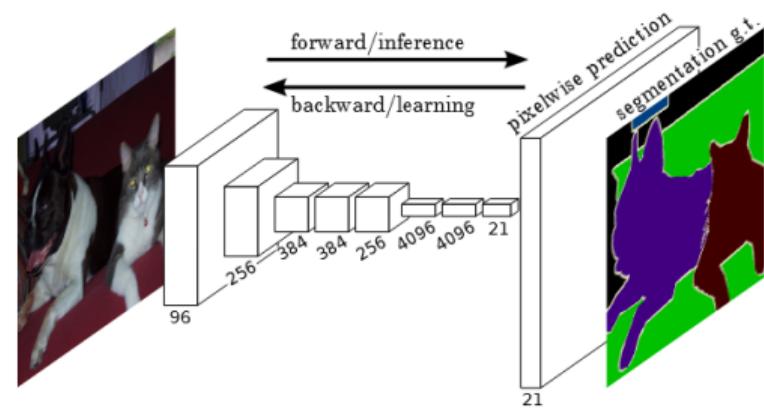


Figura: Exemplo de deconvolução numa imagem real [?]



Matemática da Deconvolução

4 Convolução Transposta

Dimensões de saída [5]

$$H_{out} = (H_{in} - 1) \times s[0] - 2 \times p[0] + d[0] \times (k[0] - 1) + p'[0] + 1 \quad (10)$$

$$W_{out} = (W_{in} - 1) \times s[1] - 2 \times p[1] + d[1] \times (k[1] - 1) + p'[1] + 1 \quad (11)$$

Algorítimo

4 Convolução Transposta

- A **Conv2D** aplica um filtro (kernel) sobre a imagem, extraíndo padrões locais.
- O **stride** e o **padding** controlam o tamanho da saída.
- O **ConvTranspose2D** faz a operação inversa, reconstruindo a forma espacial original.
- Ambas usam o mesmo kernel no exemplo ao lado.

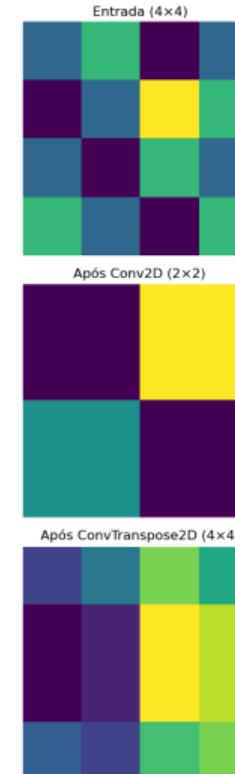


Figura: Convolução transposta gerada no python

No zero padding, unit strides, transposed

4 Convolução Transposta

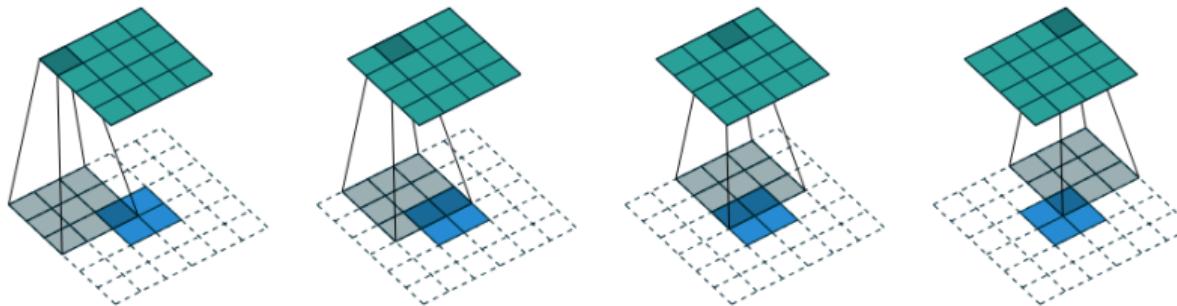


Figura: Convolução transposta ($i' = 2, k' = 3, s' = 1, p' = 2$)[2]

$$o' = i' + (k - 1) \tag{12}$$

Zero padding, unit strides, transposed

4 Convolução Transposta

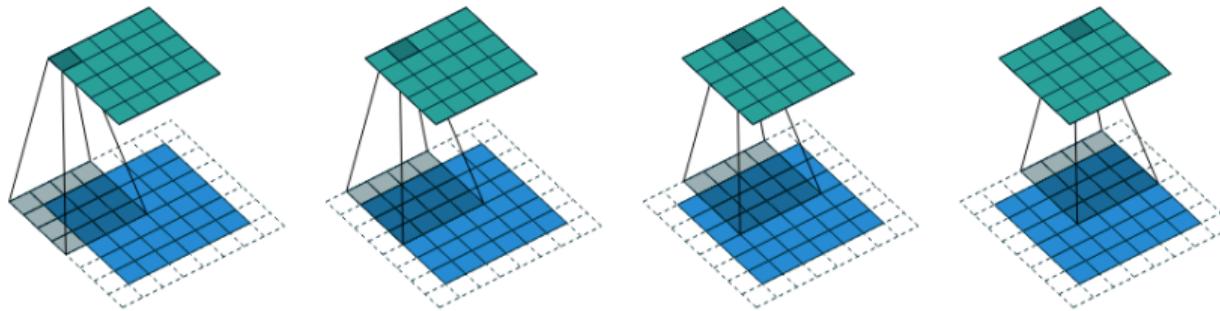


Figura: Convolução transposta ($i' = 6, k' = 4, s' = 1, p' = 1$)[2]

$$o' = i' + (k - 1) - 2p \quad (13)$$

No zero padding, non-unit strides, transposed

4 Convolução Transposta

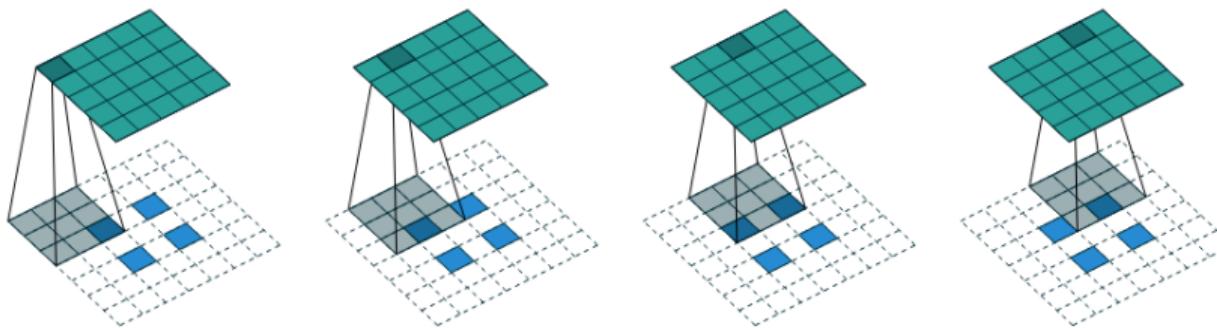


Figura: Convolução transposta ($i' = 2, \tilde{i}' = 3, k' = 3, s' = 1, p' = 2$)[2]

$$o' = s(i' - 1) + k \quad (14)$$

Zero padding, non-unit strides, transposed

4 Convolução Transposta

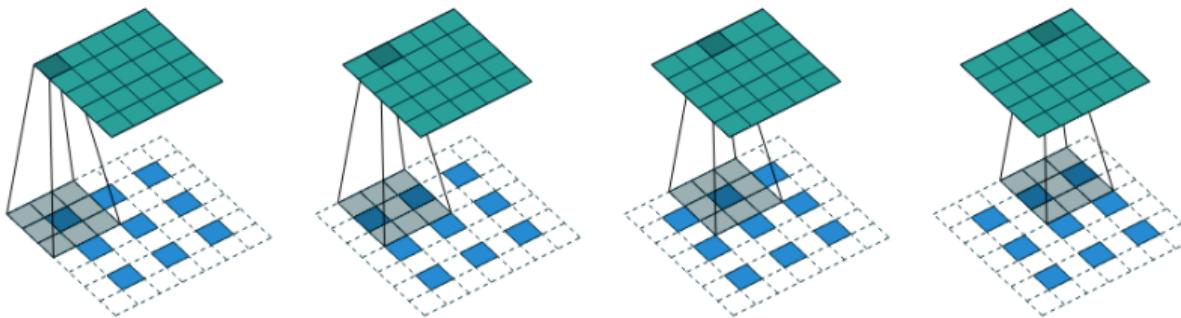


Figura: Convolução transposta ($i' = 3$, $\tilde{i}' = 5$, $k' = 3$, $s' = 1$, $p' = 1$)[2]

$$o' = s(i' - 1) + k - 2p \quad (15)$$

Sumário

5 Modelos Alternativos

- ▶ Revisão: Redes Convolucionais (CNNs)
- ▶ A matemática por trás
- ▶ Aritmética do Pooling
- ▶ Convolução Transposta
- ▶ Modelos Alternativos
- ▶ Conclusões
- ▶ Referências Bibliográficas

LeNet-5 (1998)

5 Modelos Alternativos

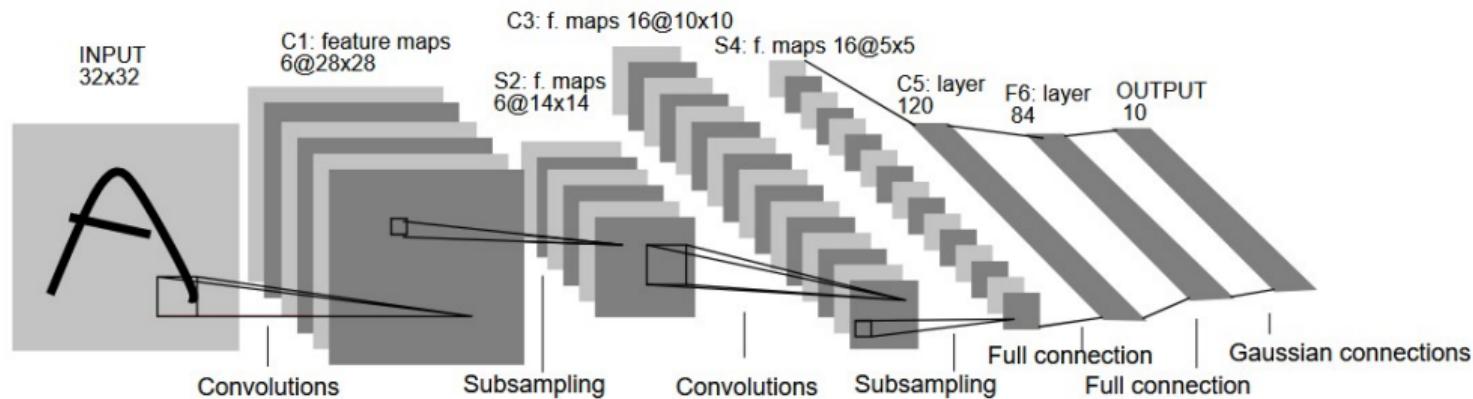


Figura: Arquitetura da LeNet-5 [6].



AlexNet (2012)

5 Modelos Alternativos

- Considerada um marco no deep learning modernos;
- Utilização de grande volume de dados para o treinamento (ImageNet);
- Uso de GPU;
- Dropout;
- Data Augmentation;
- Substituição da $\tanh()$ pela $\text{ReLU}()$.

AlexNet (2012)

5 Modelos Alternativos

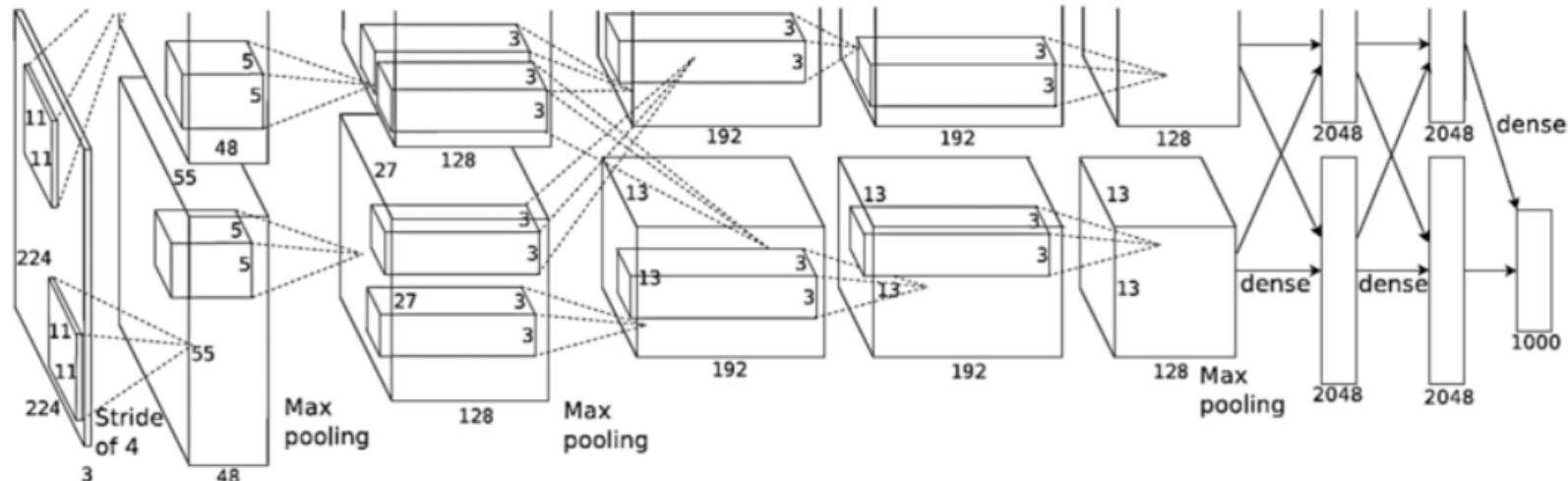


Figura: Arquitetura da LeNet-5 [7].



VGGNet (2014)

5 Modelos Alternativos

- Aumentar a performance, aumentando a profundidade da rede;
- Pode-se aproximar filtros largos como 5×5 e 7×7 utilizando filtros 3×3 :
 - um filtro 5×5 tem o mesmo campo receptivo de 2 filtros 3×3 ;
 - um filtro 7×7 tem o mesmo campo receptivo de 3 filtros 3×3 .
- Diminuição do número de parâmetros;
- Adiciona mais camadas não lineares ReLu com o uso de filtros menores.

VGGNet (2014)

5 Modelos Alternativos

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura: Família de modelos VGGNet [8].



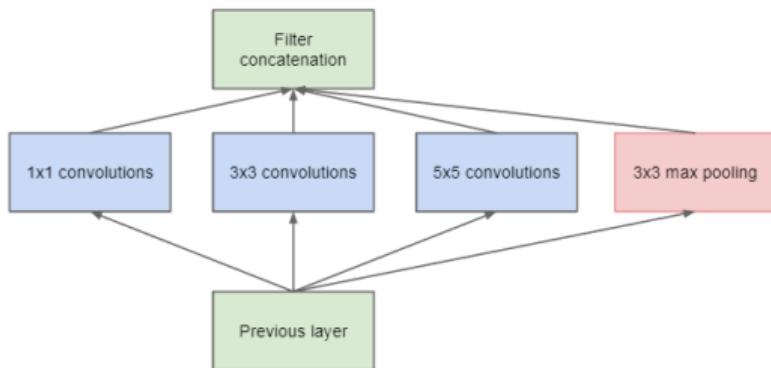
GoogLeNet / Inception (2012)

5 Modelos Alternativos

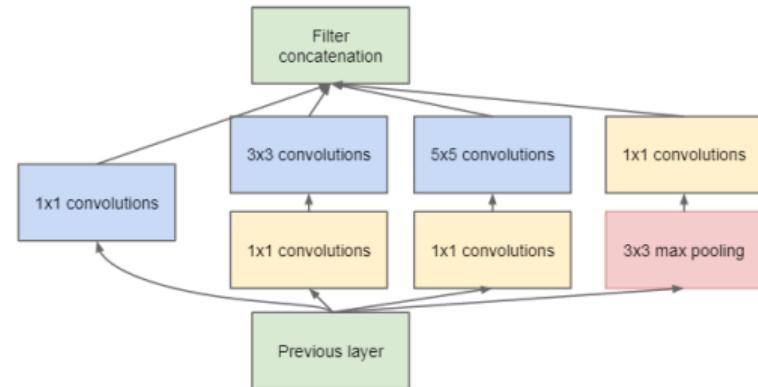
- Busca de aumentar o desempenho da AlexNet sem aumento de custo computacional;
- Criação do Módulo Inception;
- Introdução da camada de bottlenecks (convoluçãoes de 1x1).

GoogLeNet / Inception (2012)

5 Modelos Alternativos



(a) Inception module, naïve version



(b) Inception module with dimension reductions

Figura: Módulo Inception [9].

GoogLeNet / Inception (2012)

5 Modelos Alternativos

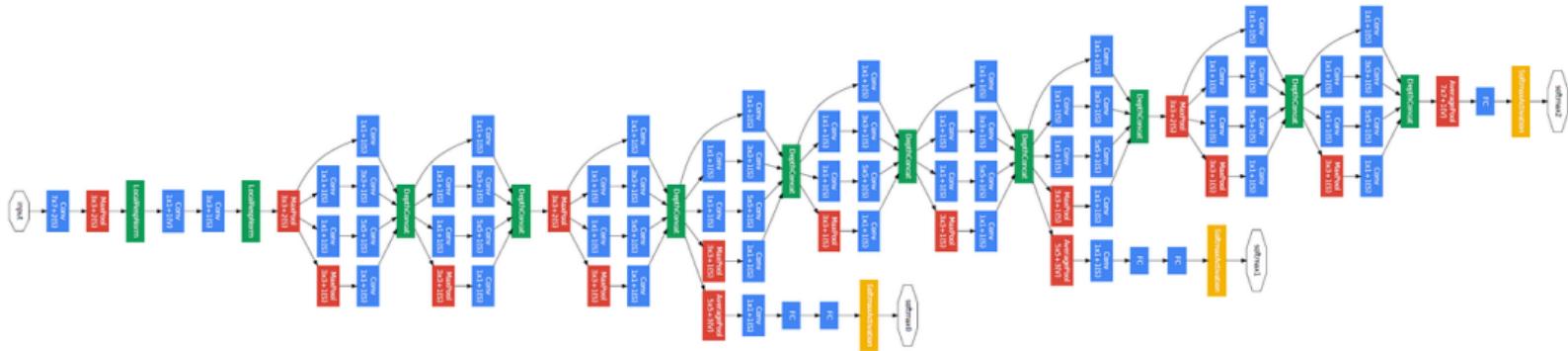


Figura: Arquitetura completa da GoogLeNet [9].

ResNet (2015)

5 Modelos Alternativos

- Problema de degradação para redes muito profundas;
- Prova experimental, utilizando camadas de identidade, que redes mais profundas devem ser pelo menos tão boas quanto suas versões menores;
- Criação de blocos de resíduos (shortcut connection).
 - Algoritmos de resíduos são mais performáticos em visão computacional (VLAD, Fisher Vectors, Multigrid)
 - Resíduo converge mais rápido e é mais fácil de otimizar

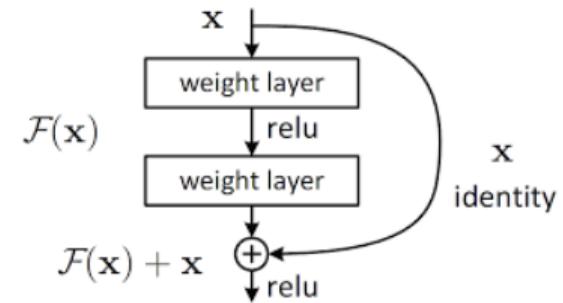


Figura: Bloco de Resíduo (shortcut connection) [10].

ResNet (2015)

5 Modelos Alternativos

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
conv2_x	56×56			3×3 max pool, stride 2		
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura: Arquitetura família ResNet [10].



DenseNet (2017)

5 Modelos Alternativos

- Busca superar a limitação da ResNet de overfit para mais de centenas de camadas;
- Shortcut connection entre todas as camadas, com concatenação dos resultados dos mapas de características ao invés de somar;
- A preservação das características aprendidas faz com que haja um melhor fluxo das informações e do gradiente no treinamento.

DenseNet (2017)

5 Modelos Alternativos

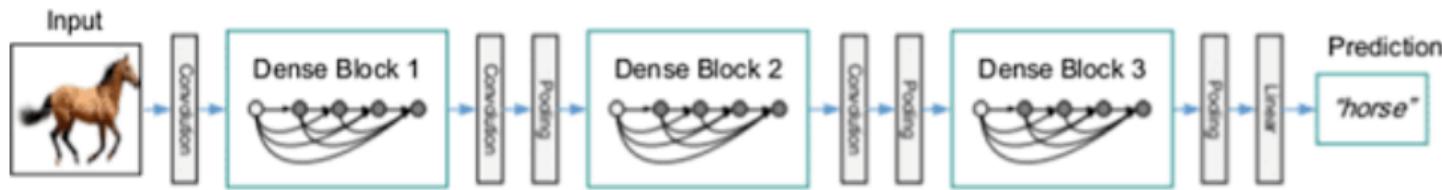


Figura: Arquitetura DenseNet [11].

CapsNet (2017)

5 Modelos Alternativos

Uma cápsula é um grupo de neurônios cuja saída é um vetor:

- **Comprimento do vetor:** probabilidade de existência.
- **Direção do vetor:** propriedades da entidade (posição, orientação, etc).

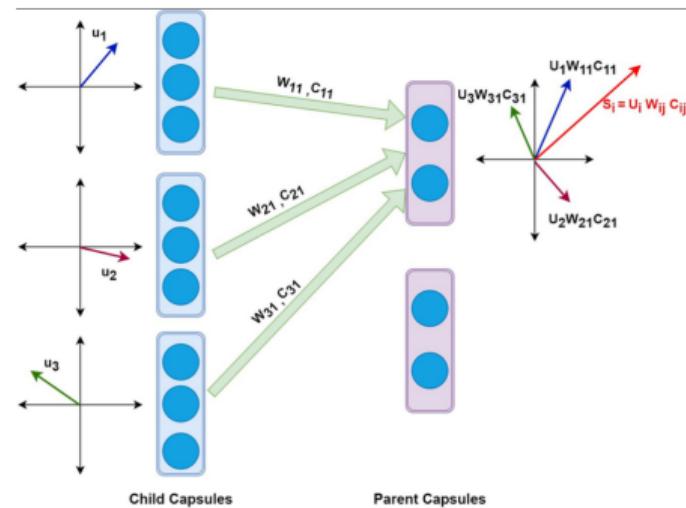


Figura: dinâmica de cápsulas “filho” e “pai”, com vetores de ativação e previsão (c-vetores) conectando-os.

CapsNet (2017)

5 Modelos Alternativos

- Camada convolucional inicial.
- **Primary Capsules** (lower-level): vetores derivados de filtros convolucionais.
- **Digit Capsules** (higher-level): uma por classe (ex: dígitos 0-9).
- Classificação baseada na magnitude do vetor de saída.

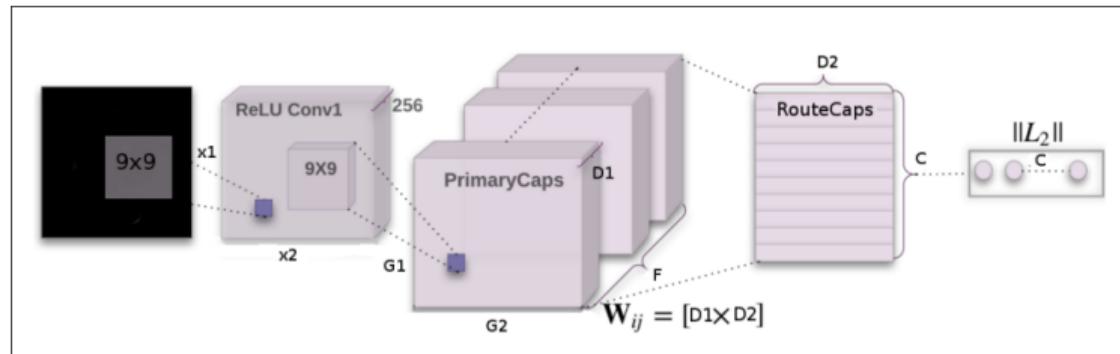


Figura: Arquitetura da CapsNet [12]

EfficientNet (2019)

5 Modelos Alternativos

- Ao contrário dos outros modelos até então a EfficientNet não busca introduzir uma inovação na arquitetura das CNNs, mas sim numa maneira de escalar e crescer a rede;
- **Compound Scalling (depth, width, resolution):** ao invés de escalar a rede arbitrariamente em uma das 3 dimensões, a EfficientNet propõe um método para escalar uniformemente largura, profundidade e resolução da rede utilizando um conjunto fixo de coeficientes de escalonamento.

$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

Figura: Coeficientes de escalonamento do compound scalling [13].

EfficientNet (2019)

5 Modelos Alternativos

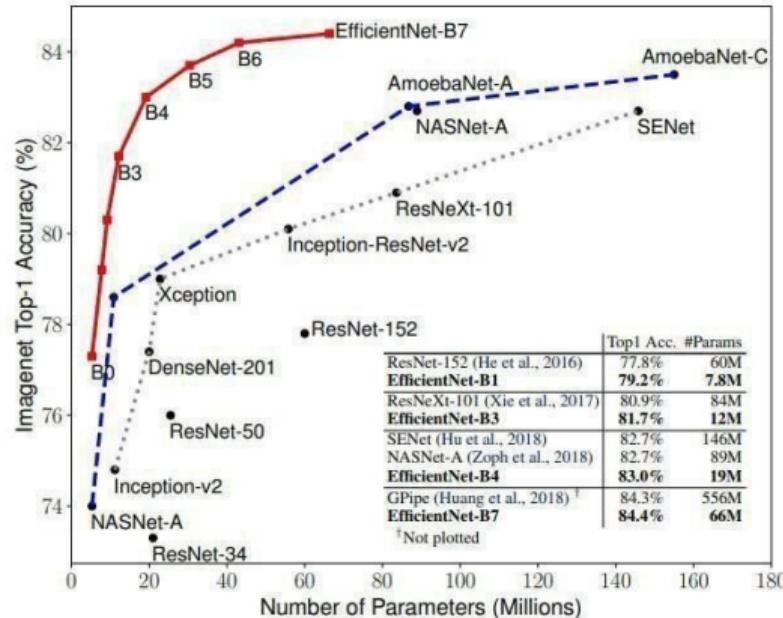


Figura: Comparativo tamanho e precisão entre modelos no ImageNet [13].

Sumário

6 Conclusões

- ▶ Revisão: Redes Convolucionais (CNNs)
- ▶ A matemática por trás
- ▶ Aritmética do Pooling
- ▶ Convolução Transposta
- ▶ Modelos Alternativos
- ▶ Conclusões
- ▶ Referências Bibliográficas

Vantagens das CNNs

6 Conclusões

- Redução de dimensionalidade.
- Prevenção de overfitting.
- Preservação das informações mais relevantes.
- Compatibilidade com arquiteturas profundas.

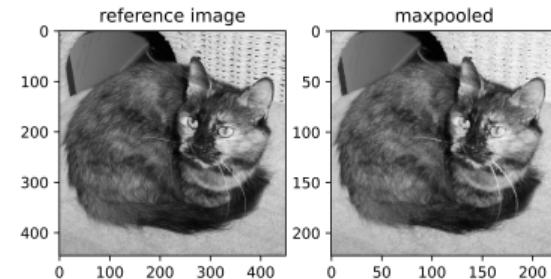


Figura: Redução da dimensionalidade.

Limitações das CNNs

6 Conclusões

- Podem não capturar relações espaciais entre partes do objeto.
- Sensíveis a rotações, escala e perspectiva ("Picasso problem").
- Dependem de grandes volumes de dados e *data augmentation*.
- Pooling remove informação estrutural importante.

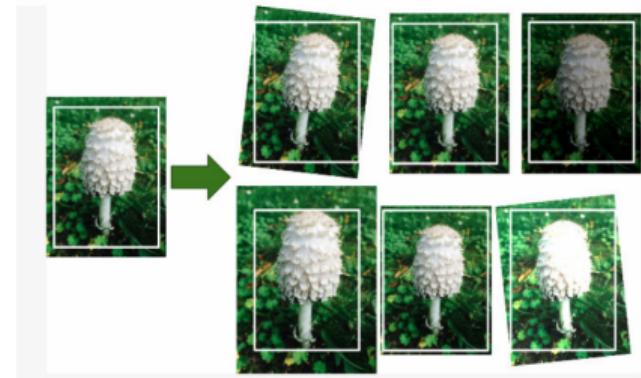


Figura: DATA AUGMENTATION - novas instâncias de treinamento a partir de instâncias existentes.
[1]

Sumário

7 Referências Bibliográficas

- ▶ Revisão: Redes Convolucionais (CNNs)
- ▶ A matemática por trás
- ▶ Aritmética do Pooling
- ▶ Convolução Transposta
- ▶ Modelos Alternativos
- ▶ Conclusões
- ▶ Referências Bibliográficas



Referências Bibliográficas

7 Referências Bibliográficas

- [1] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. "O'Reilly Media, Inc.", 2022.
- [2] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.
- [3] N. Akhtar and U. Ragavendran, "Interpretation of intelligence in cnn-pooling processes: a methodological survey," *Neural computing and applications*, vol. 32, no. 3, pp. 879–898, 2020.
- [4] F. Dos Reis and N. Karathanasopoulos, "Deep learning, deconvolutional neural network inverse design of strut-based lattice metamaterials," *Computational Materials Science*, vol. 244, p. 113258, 2024.



Referências Bibliográficas

7 Referências Bibliográficas

- [5] PyTorch Core Development Team, "nn.ConvTranspose2d," 2025.
- [6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.
- [8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.



Referências Bibliográficas

7 Referências Bibliográficas

- [9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [11] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [12] R. Mukhometzianov and J. Carrillo, "Capsnet comparative performance evaluation for image classification," *arXiv preprint arXiv:1805.11195*, 2018.



Referências Bibliográficas

7 Referências Bibliográficas

- [13] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.



Convolutional Neural Networks

*Obrigado pela Atenção!
Alguma Pergunta?*

brennorcs@poli.ufrj.br, guimaraes921@poli.ufrj.br, lucas.alexandre@coppe.ufrj.br,

rnagao@cos.ufrj.br