Deep Learning

Aplicações de CNN em Visão Computacional

Lucas Silveira Kupssinskü

Machine Learning Theory and Applications Laboratory
PUCRS

agosto de 2026

Visão Geral

Visão Geral

VGG

- Arquitetura de CNN proposta em 2014, publicada no ICLR 2015[1].
- Continua sendo usada em muitos cenários, especialmente como *backbone* pré-treinado.
- No ILSVRC-2014 (*ImageNet Large Scale Visual Recognition Challenge*) ficou em **1º na localização** e **2º na classificação** (vencida pelo GoogLeNet).
- Taxa de erro no ImageNet (ensemble multi-crop reportado no paper):
  - *Top-5*: 6.8%.
  - *Top-1*: 23.7%.
  - VGG-16 *single-model* fica em torno de 7.3% *top-5* e 24.7% *top-1*.
- $\sim$ 140M parâmetros treináveis.

[1] Karen Simonyan e Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". Em: arXiv preprint arXiv:1409.1556 (2014).

Definição do Problema

- ImageNet contém imagens RGB na resolução $224 \times 224 \times 3$.
  - As entradas são tensores $(224, 224, 3)$.
- O objetivo é classificar cada imagem em uma das 1000 classes pré-definidas.
  - Função de custo: entropia cruzada com os *logits*.
- Precisamos transformar um tensor $(224, 224, 3)$ em um vetor $(1, 1, 1000)$.

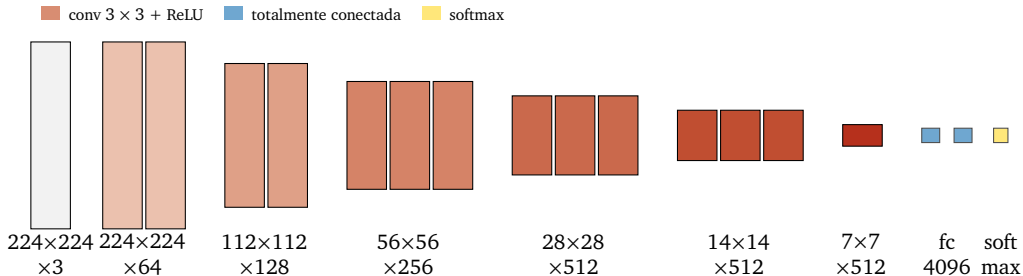
Blocos de Construção

- A VGG combina três tipos de operações:
  - **Convolução + ReLU**: aumenta o número de canais e extrai características espaciais.
  - ***Pooling***: subamostra a dimensão espacial (largura e altura).
  - **Camadas densas**: realizam a classificação final, partindo do *feature map* achatado.
- Toda a parte convolucional pode ser vista como um **extrator de *features***.
- As camadas densas no fim convertem essas *features* em probabilidades de classe.

VGG16

- VGG16 é a implementação mais comum, equivalente à variante “D” descrita no artigo.
- Possui 16 camadas com parâmetros treináveis: **13 convolucionais** e **3 densas**.
- Todas as convoluções são 3×3 com *stride* 1 e *padding* 1 (mantêm a dimensão espacial).
- Cada *max pooling* usa janela 2×2 com *stride* 2 (reduz pela metade a largura e altura).

Arquitetura da VGG16



Cada *max pooling* reduz a dimensão espacial pela metade e dobra (ou estabiliza) o número de canais.

Contando Parâmetros: Primeiro Bloco

- Bloco inicial: duas convoluções 3×3 produzindo 64 canais.
  - Cada filtro convolucional tem $K_h \cdot K_w \cdot C_{in}$ pesos, e há C_{out} filtros.
  - Primeira: kernel 3×3, $C_{in} = 3$ (RGB), $C_{out} = 64$, portanto $3 \cdot 3 \cdot 3 \cdot 64 = 9 \cdot 192 = 1.728$ pesos.
  - Segunda: kernel 3×3, $C_{in} = 64$ (saída da primeira), $C_{out} = 64$, portanto $3 \cdot 3 \cdot 64 \cdot 64 = 9 \cdot 4.096 = 36.864$ pesos.
- Total parcial: $1.728 + 36.864 = 38.592$ parâmetros.
- O *max pooling* não adiciona parâmetros.

Observação

Ignoramos o *bias* para simplificar a contagem. Na prática, cada filtro tem um *bias* associado.

Contando Parâmetros: Convoluções

- Crescimento por bloco convolucional:

Bloco	Convoluções	Parâmetros do bloco	Total acumulado
1	$2\times$ conv $3 \times 3 \times 64$	38.592	38.592
2	$2\times$ conv $3 \times 3 \times 128$	221.184	259.776
3	$3\times$ conv $3 \times 3 \times 256$	1.474.560	1.734.336
4	$3\times$ conv $3 \times 3 \times 512$	5.898.240	7.632.576
5	$3\times$ conv $3 \times 3 \times 512$	7.077.888	14.710.464

- Após cinco blocos convolucionais: $\sim$ 14.7M parâmetros.

Contando Parâmetros: Camadas Densas

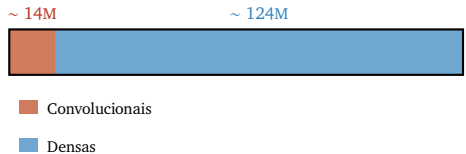
- Após o último *pooling*, achatamos um tensor $7 \times 7 \times 512 = 25.088$.

Camada	Cálculo	Parâmetros
fc1	25.088×4.096	102.760.448
fc2	4.096×4.096	16.777.216
fc3 (saída)	4.096×1.000	4.096.000
Subtotal denso		**123.633.664**

- Total da rede: $14.710.464 + 123.633.664 \approx 138.3$M parâmetros.

Onde Estão os Parâmetros?

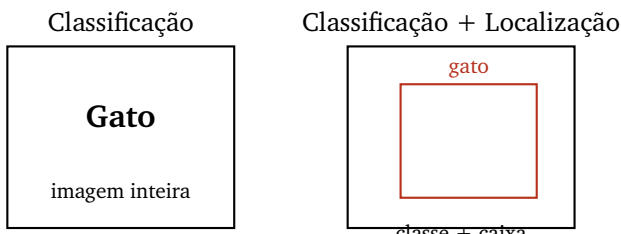
- A maioria dos parâmetros treináveis está nas **camadas densas**.
  - Convoluções: ~ 14M.
  - Densas: ~ 124M.
- Convoluções compartilham pesos, então usam ordens de magnitude menos parâmetros que camadas totalmente conectadas equivalentes.
- Por isso é comum trocar as densas por uma cabeça menor quando se reaproveita a VGG em outras tarefas.



Visão Geral

Classificação vs Classificação + Localização

- Além de dizer “há um gato na imagem”, queremos saber **onde** esse gato está.
- A saída adicional é uma *bounding box* delimitando o objeto.

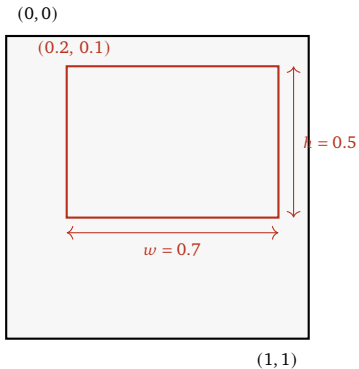


Bounding Box

- Para descrever uma *bounding box* precisamos de 4 valores:
  - (x, y): coordenadas de um ponto de referência (canto superior esquerdo ou centro).
  - h: altura da caixa.
  - w: largura da caixa.
- Podemos modelar esses 4 valores como um problema de **regressão multivariada**.
- Convenção comum: o canto superior esquerdo da imagem é $(0, 0)$ e o canto inferior direito é $(1, 1)$.

Exemplo de Bounding Box

- Alvo $\mathbf{y} = (x, y, h, w)$.
- Para o exemplo ao lado:
  - $(x, y) = (0.2,\ 0.1)$.
  - $h = 0.5$.
  - $w = 0.7$.
- Logo, $\mathbf{y} = (0.2,\ 0.1,\ 0.5,\ 0.7)$.



Arquitetura para Localização

- Para regressão dos 4 valores da *bounding box*, a saída precisa de **4 unidades sigmoide** (em vez de *softmax* para 1000 classes).
  - A sigmoide garante saídas em $[0, 1]$, compatíveis com coordenadas normalizadas.
  - Hipótese implícita: assumimos que a caixa cabe inteiramente dentro da imagem, com $(x, y) \in [0, 1]^2$ e $h, w \in [0, 1]$.
  - Caixas que estouram a borda exigem outra parametrização (por exemplo, logits sem ativação, ou regressão de *offsets* em relação a uma *anchor*).
- A função de custo passa a ser o **erro quadrático médio** (MSE):

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} \left[\left(y_0^{(i)} - \hat{y}_0^{(i)}\right)^2 + \left(y_1^{(i)} - \hat{y}_1^{(i)}\right)^2 + \left(y_2^{(i)} - \hat{y}_2^{(i)}\right)^2 + \left(y_3^{(i)} - \hat{y}_3^{(i)}\right)^2 \right]$$

- As convoluções continuam funcionando como extrator de *features*.

Reaproveitando uma Rede Pré-treinada

- Imagine que já treinamos a rede para **classificar** as fotos.
  - Os parâmetros das convoluções já estão ajustados para extrair *features* úteis.
- Estratégia comum: **congelar** os pesos da convolução e treinar apenas a cabeça densa para a nova tarefa[2].
  - Reduz drasticamente o tempo de treinamento.
  - Funciona bem quando o conjunto de dados novo é pequeno.
- Alternativa: *fine-tuning* da rede inteira com *learning rate* baixa.

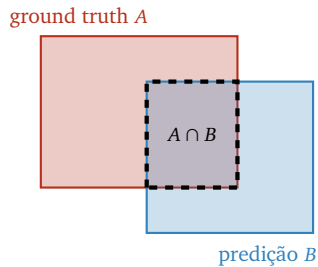
[2] Pierre Sermanet et al. "OverFeat: Integrated recognition, localization and detection using convolutional networks". Em: arXiv preprint arXiv:1312.6229 (2013).

Métricas de Localização: *Intersection over Union*

- Computa a interseção entre a *bounding box* prevista e o *ground truth*, sobre a união das duas áreas.

$$IoU = \frac{A \cap B}{A \cup B}$$

- Valor ideal: $IoU = 1$ (sobreposição perfeita).
- Ignora o tamanho absoluto das caixas, só importa a sobreposição relativa.

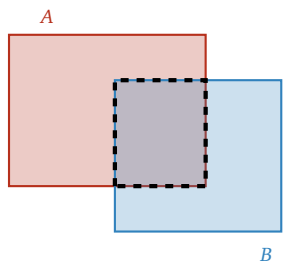


Métricas de Localização: *Dice Score*

- Computa o dobro da interseção sobre a soma das áreas.

$$Dice = \frac{2 \cdot (A \cap B)}{Area(A) + Area(B)}$$

- Valor ideal: $Dice = 1$.
- Penaliza menos diferenças de tamanho que o IoU.



IoU vs Dice: Exemplo Numérico

- Suponha duas caixas com:
  - $Area(A) = 10$.
  - $Area(B) = 10$.
  - $A \cap B = 5$.
  - $A \cup B = Area(A) + Area(B) - A \cap B = 15$.

$$IoU = \frac{5}{15} \approx 0.333 \qquad Dice = \frac{2 \cdot 5}{10 + 10} = 0.5$$

- Ambas as métricas medem a sobreposição entre caixas.
- Para a mesma sobreposição, o IoU produz valores menores que o *Dice*.
  - Pode ser interpretado como uma penalização “mais quadrática” do erro.

Localização: Resumo

- A *bounding box* é modelada como quatro saídas de regressão multivariada.
- Função de custo: MSE.
- Ativação na saída: sigmoide (4 unidades).
- Podemos:
  - Reaproveitar pesos de convolução já treinados.
  - Treinar apenas a nova cabeça densa.
  - Ou treinar a rede inteira (*fine-tuning*).
- Avaliamos com IoU ou *Dice Score*, comparando com o *ground truth*.

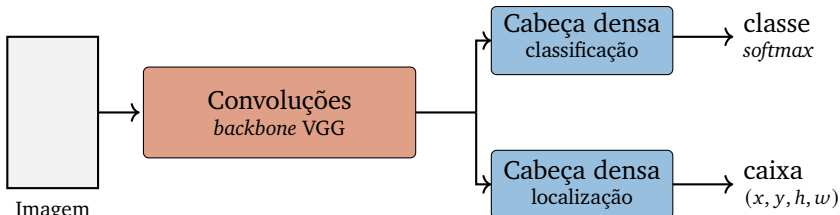
Visão Geral

Múltiplas Tarefas Simultâneas

- Podemos resolver classificação e localização **ao mesmo tempo**.
- A parte convolucional serve como **extrator de *features*** compartilhado.
- Adicionamos duas cabeças densas independentes:
  - Cabeça de classificação: *softmax* sobre as 1000 classes.
  - Cabeça de localização: 4 unidades sigmoide para a *bounding box*.
- Cada cabeça contribui com seu próprio termo na função de custo:

$$\mathcal{L} = \mathcal{L}_{\text{classif.}} + \lambda \, \mathcal{L}_{\text{loc.}}$$

Backbone como Extrator de Features



- Trocando a cabeça e mantendo o *backbone*, podemos resolver muitas tarefas distintas a partir das mesmas *features*.

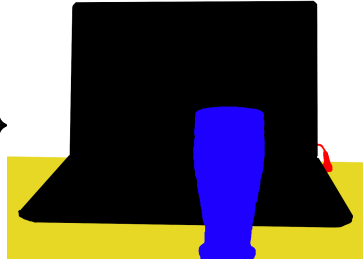
Visão Geral

Segmentação Semântica

- É basicamente uma classificação em **nível de pixel**.
- Cada pixel da imagem recebe uma probabilidade de pertencer a cada classe.
- Aplicações típicas: direção autônoma, imagens médicas, sensoriamento remoto.

Imagem de entrada

Máscara por classe



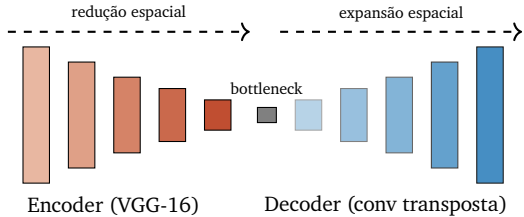
Cada pixel recebe uma classe (*notebook*, *copo*, *mesa*, fundo). Exemplo CC0 de Martin Thoma, *Wikimedia Commons*.

Primeira Ideia (Ingênua)

- Separar a imagem em *patches*.
- Para cada *patch*, classificar o pixel central com uma CNN.
- Repetir para todos os pixels da imagem.
- Problemas:
  - Extremamente **ineficiente**: requer milhares de *forwards* por imagem.
  - Muito redundante, pois *patches* vizinhos compartilham a maior parte do conteúdo.
- Precisamos de uma forma de classificar todos os pixels em um único *forward*.

Arquitetura Encoder-Decoder

- A CNN deve classificar todos os pixels em um único *forward*[3].
- **Encoder**: reduz a dimensão espacial via convoluções e *pooling*.
- **Decoder**: aumenta a dimensão espacial via *unpooling* e convoluções transpostas.



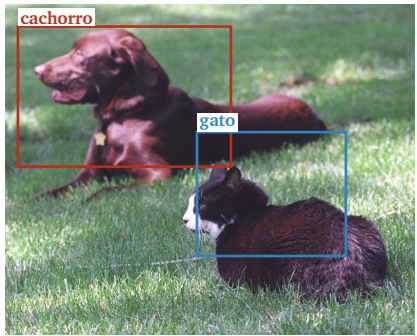
- Saída: máscara com a mesma resolução da entrada, com uma classe por pixel.

[3] Hyeonwoo Noh, Seunghoon Hong e Bohyung Han. "Learning deconvolution network for semantic segmentation". Em: ICCV. 2015, pp. 1520–1528.

Visão Geral

Detecção de Objetos

- Tarefa muito parecida com classificação + localização.
- Diferença: a imagem pode conter **vários objetos** simultaneamente.
  - Cada objeto detectado tem sua própria classe e *bounding box*.
- Desafio principal: **não sabemos de antemão quantos objetos** aparecem em cada imagem.



Cada objeto recebe classe + *bounding box*. Foto: *Socks e Buddy* (Casa Branca), domínio público.

Famílias de Detectores: Two-Stage

- **R-CNN**[4]: gera $\sim$ 2.000 *region proposals* (via *selective search*), recorta cada uma e roda a CNN separadamente. Lento, mas estabeleceu a base do paradigma.
- **Fast R-CNN**: a CNN processa a imagem inteira uma só vez; *ROI pooling* extrai *features* de cada região a partir do mesmo *feature map*.
- **Faster R-CNN**[5]: substitui *selective search* por uma **Region Proposal Network** (RPN), também convolucional. Treino *end-to-end*.
- Característica comum, primeiro propõe regiões, depois classifica cada uma. Tipicamente mais precisos, porém mais lentos.

[4] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". Em: CVPR. 2014.

[5] Shaoqing Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks". Em: NeurIPS. 2015.

Famílias de Detectores: One-Stage e Transformers

- **YOLO**[6] (*You Only Look Once*): divide a imagem em uma grade $S \times S$ e prediz, em **uma única passada**, a classe e a *bounding box* para cada célula.
  - Muito mais rápido, opera em tempo real, ao custo de menor precisão em objetos pequenos.
  - Variantes posteriores (YOLOv3-v8) recuperam grande parte dessa diferença de precisão.
- **SSD** (*Single Shot MultiBox Detector*): também *one-stage*, prediz caixas em vários níveis de resolução simultaneamente.
- **DETR**[7]: formula detecção como predição direta de um conjunto de caixas usando um **Transformer**.
  - Elimina *anchors* e *non-maximum suppression*.
  - Treina com *Hungarian matching* entre predições e *ground truth*.

[6] Joseph Redmon et al. "You only look once: Unified, real-time object detection". Em: CVPR. 2016.

[7] Nicolas Carion et al. "End-to-end object detection with transformers". Em: ECCV. 2020.

Visão Geral

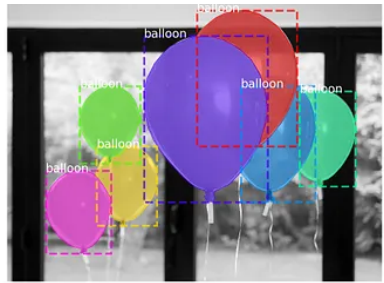
Segmentação de Instâncias

- Muito parecida com detecção de objetos:
  - Cada objeto recebe um valor por pixel, e não apenas uma *bounding box*.
  - Instâncias diferentes da mesma classe são **diferenciadas** entre si.
- Diferença em relação à segmentação semântica: dois gatos lado a lado aparecem como duas máscaras distintas, não como uma só.

Segmentação Semântica vs Instâncias

Segmentação Semântica

Segmentação de Instâncias



Na **semântica**, todos os objetos da classe “balão” recebem a **mesma** cor; na de **instâncias**, cada balão recebe uma cor **distinta**.

Figura: *Mask R-CNN*, *Wikimedia Commons* (CC BY-SA 4.0).

Visão Geral

Leituras Recomendadas

- Sugere-se **fortemente** a leitura do Capítulo 10 de *Understanding Deep Learning* [4].
  - Disponível em `https://udlbook.github.io/udlbook/`.
- Os artigos abaixo são acessíveis e fornecem a visão original dos autores sobre as ideias apresentadas nessa aula:
  - *Very deep convolutional networks for large-scale image recognition* [8].
  - *OverFeat: Integrated recognition, localization and detection using convolutional networks* [7].
  - *Learning deconvolution network for semantic segmentation* [3].

Referências

[1] Nicolas Carion et al. “End-to-end object detection with transformers”. Em: ECCV. 2020.

[2] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. Em: CVPR. 2014.

[3] Hyeonwoo Noh, Seunghoon Hong e Bohyung Han. “Learning deconvolution network for semantic segmentation”. Em: ICCV. 2015, pp. 1520–1528.

[4] Simon J. D. Prince. Understanding Deep Learning. MIT Press, 2023.

[5] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. Em: CVPR. 2016.

[6] Shaoqing Ren et al. “Faster R-CNN: Towards real-time object detection with region proposal networks”. Em: NeurIPS. 2015.

[7] Pierre Sermanet et al. “OverFeat: Integrated recognition, localization and detection using convolutional networks”. Em: arXiv preprint arXiv:1312.6229 (2013).

[8] Karen Simonyan e Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. Em: arXiv preprint arXiv:1409.1556 (2014).