Arquiteturas de Redes Neurais Convolucionais

Eduardo Adame

Redes Neurais

22 de outubro de 2025

- **Objetivo**: Compreender a evolução das arquiteturas CNN
- **Marcos históricos**: AlexNet, VGG, Inception, ResNet
- **Tendências atuais**: Vision Transformers e arquiteturas híbridas
- **Aplicações práticas**: Como escolher a arquitetura adequada

AlexNet (2012): O Marco Inicial

Contexto Histórico

- Criada para o ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- **Tarefa**: Classificar entre 1000 classes
- **Dataset**: Aproximadamente 1,2 milhão de imagens
- Considerada o "ponto de ignição"do deep learning moderno

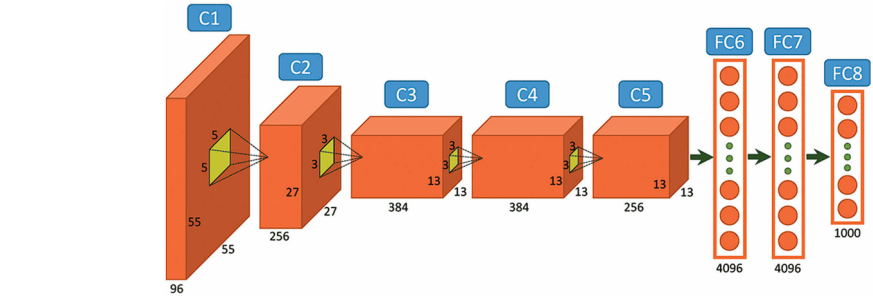
Resultados Revolucionários

- Top-5 error rate: 15,4%
- Segundo colocado: 26,2%
- **Diferença de mais de 10%** - resultado devastador

Por que foi tão importante?

- Primeira demonstração clara do poder das CNNs profundas
- Uso de GPUs para acelerar o treinamento
- Popularizou técnicas como ReLU e Dropout

Arquitetura da AlexNet



Características Principais

- **8 camadas**: 5 convolucionais + 3 totalmente conectadas
- **60 milhões de parâmetros**
- **Data augmentation**: Rotação, espelhamento, crops aleatórios
- **Dropout**: Regularização nas camadas FC
- **ReLU**: Primeira aplicação em larga escala

Template Básico da AlexNet

Listing 1: Implementação Simplificada

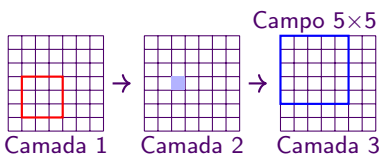
```
model = Sequential([
    Conv2D(96, (11,11), strides=4, activation='relu'),
    MaxPooling2D((3,3), strides=2),
    Conv2D(256, (5,5), activation='relu'),
    MaxPooling2D((3,3), strides=2),
    Conv2D(384, (3,3), activation='relu'),
    Conv2D(384, (3,3), activation='relu'),
    Conv2D(256, (3,3), activation='relu'),
    MaxPooling2D((3,3), strides=2),
    Flatten(),
    Dense(4096, activation='relu'),
    Dropout(0.5),
    Dense(4096, activation='relu'),
    Dropout(0.5),
    Dense(1000, activation='softmax')
])
```

VGG (2014): Simplicidade e Profundidade

Filosofia de Design

- **Simplificar a estrutura da rede**
- **Evitar escolhas manuais** do tamanho de convolução
- **Redes muito profundas** com convoluções 3×3
- Convoluções menores **"simulam"convoluções maiores**

Insight Principal: Campo Receptivo



Duas convoluções 3×3 consecutivas = Uma convolução 5×5

VGG: Vantagem dos Filtros Pequenos

Comparação de Parâmetros

Uma camada 7×7

$$\text{Parâmetros} = 7 \times 7 \times C \times C \quad (1)$$
$$= 49C^2 \quad (2)$$

Três camadas 3×3

$$\text{Parâmetros} = 3 \times (3 \times 3 \times C \times C) \quad (3)$$
$$= 27C^2 \quad (4)$$

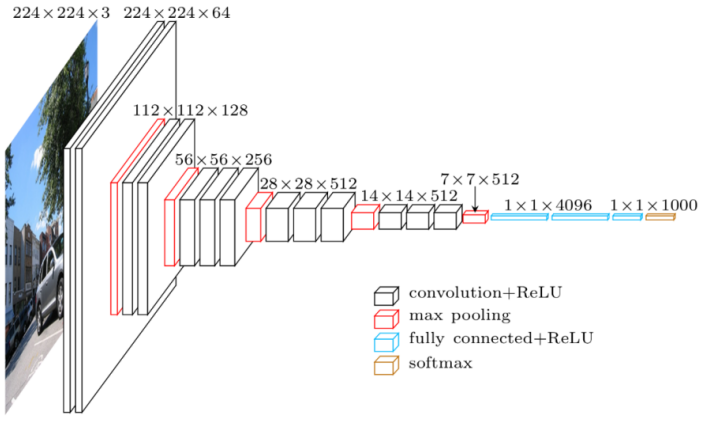
Redução de Parâmetros

$$49C^2 \rightarrow 27C^2 \approx \textbf{45\% de redução!}$$

Vantagens Adicionais

- **Mais não-linearidades**: 3 ReLUs vs 1 ReLU
- **Maior expressividade**: Função mais complexa
- **Melhor gradiente**: Propagação mais eficiente

Arquitetura VGG16



Padrão de Arquitetura

- **Duplicação progressiva**: 64 → 128 → 256 → 512 → 512
- **Redução espacial**: MaxPooling 2×2 após cada bloco
- **16 camadas com pesos**: 13 conv + 3 FC

Legado da VGG

Contribuições Principais

- **Primeira arquitetura** a experimentar com muitas camadas
- **Filosofia**: "Mais profundo é melhor"
- **Padronização**: Uso consistente de filtros 3×3
- **Modelo base**: Serviu como backbone para trabalhos futuros

Limitações

- **Muito pesada**: 138M parâmetros, arquivo de 500MB+
- **Lenta**: Muitas operações nas camadas FC
- **Consumo de memória**: Mapas de características grandes

Influência Atual

- Backbone em detecção de objetos (Faster R-CNN, etc.)
- Base para transfer learning
- Inspiração para arquiteturas modernas

Inception (GoogLeNet, 2014): Pensando Diferente

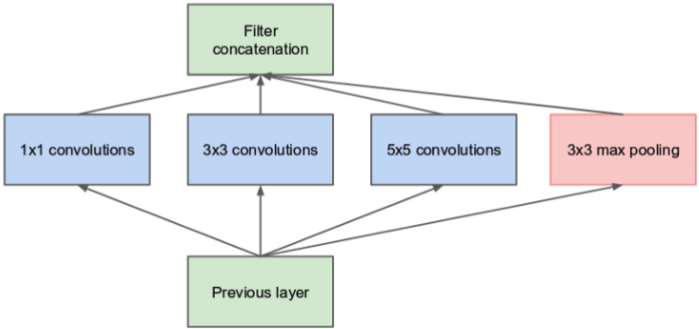
Motivação

- **Problema**: Qual tamanho de filtro usar? 1×1, 3×3, 5×5?
- **Insight**: Por que não usar *todos*?
- **Eficiência computacional**: Como manter baixo custo?
- **Principio Hebbiano**: "Neurônios que disparam juntos, se conectam"

Solução: Módulos Inception

- **Ramificação paralela**: Diferentes operações simultâneas
- **Concatenação**: Combinar todas as saídas
- **1×1 convoluções**: Redução de dimensionalidade
- **Especialização**: Cada ramo processa uma porção do trabalho

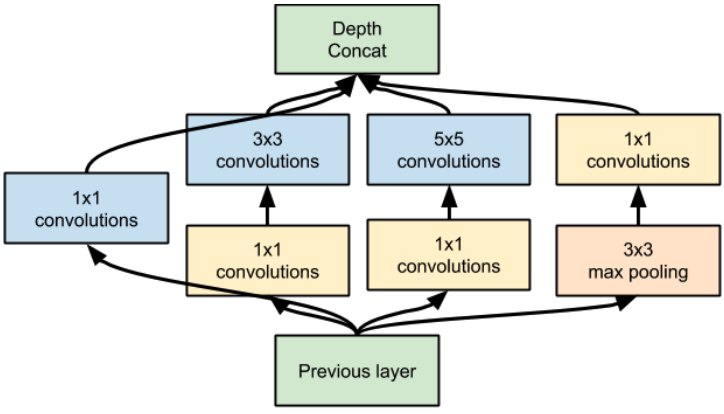
Módulo Inception: Versão Ingênua



Problema: Muitas operações caras!

- Convoluções 3×3 e 5×5 são computacionalmente intensivas
- Número de canais cresce rapidamente
- Explosão de parâmetros

Inception com Gargalos 1×1



Vantagens dos Gargalos 1×1

- **Redução de canais**: Antes das operações caras
- **Menos parâmetros**: Significativa economia computacional
- **Não-linearidade extra**: ReLU após cada 1×1

Evolução do Inception

Inception v1 (GoogLeNet, 2014)

- Módulos básicos com gargalos 1×1
- 22 camadas, apenas 5M parâmetros
- Saídas auxiliares para treinamento

Inception v2/v3 (2015-2016)

- **Factorização**: $5\times5 \rightarrow$ duas 3×3; $3\times3 \rightarrow 1\times3 + 3\times1$
- **Batch Normalization**: Acelera convergência
- **Label smoothing**: Regularização no treinamento

Inception v4 e Inception-ResNet (2016)

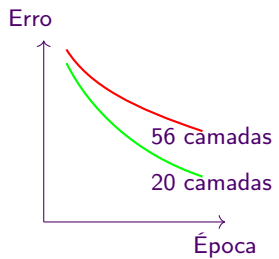
- Arquitetura mais uniforme e simples
- Combinação com conexões residuais
- Estado da arte até 2017

Xception (2017)

- Separable convolutions: Espacial + Canal
- Ainda mais eficiente que Inception v3

ResNet (2015): Resolvendo o Problema da Profundidade

O Problema Observado



Observação Surpreendente

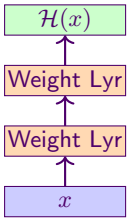
- Redes mais profundas performam *pior*
- Tanto em treino quanto em teste
- Não é overfitting!
- Problema de otimização

Diagnóstico

- **Desvanecimento de gradiente**: Camadas iniciais param de aprender
- **Degradação**: Performance satura e depois decai
- **Dificuldade de otimização**: Funções de identidade são difíceis de aprender

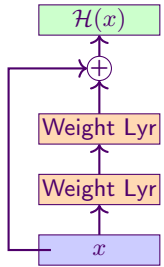
A Solução ResNet: Conexões Residuais

Bloco Tradicional



Aprende: $\mathcal{H}(x)$

Bloco Residual

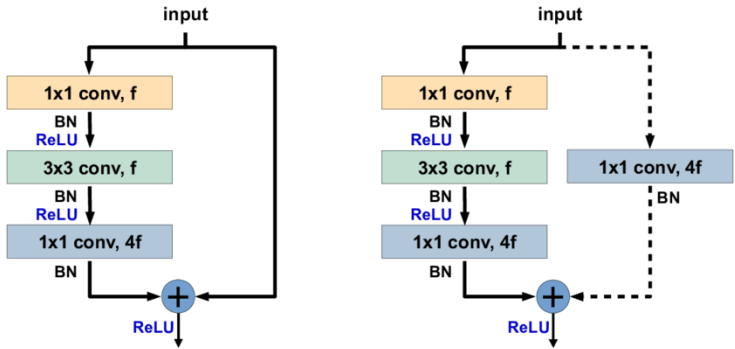


Aprende: $\mathcal{F}(x) = \mathcal{H}(x) - x$

Insight Chave

- **Função residual**: $\mathcal{F}(x)$ é mais fácil de aprender que $\mathcal{H}(x)$
- **Identidade**: Se a melhor função é identidade, $\mathcal{F}(x) = 0$
- **Gradientes**: Fluxo direto através das skip connections

Blocos ResNet em Detalhes



Bottleneck Design

- $1\times1 \rightarrow 3\times3 \rightarrow 1\times1$: Reduz complexidade computacional
- **Menos parâmetros**: Para redes muito profundas (50+ camadas)
- **Eficiência**: Mantém expressividade com menor custo

Família ResNet e Impacto

Variantes Principais

- **ResNet-18/34**: Blocos básicos, até 34 camadas
- **ResNet-50/101/152**: Blocos bottleneck, até 152 camadas
- **ResNet-200+**: Experimentos com redes extremamente profundas

Resultados no ImageNet (2015)

- **ResNet-152**: 3.57% top-5 error (melhor que humanos!)
- **Primeiro a superar performance humana** em ImageNet
- **Vencedor**: ILSVRC 2015 Classification, Detection, Localization

Impacto Transformador

- **Arquitetura padrão**: Backbone em quase todos os modelos
- **Transfer learning**: Base para inúmeras aplicações
- **Conceito residual**: Adoptado em Transformers, GAN, etc.
- **Possibilitou redes profundas**: 1000+ camadas funcionais

Evoluções Pós-ResNet (2016-2018)

DenseNet (2017)

- **Conexões densas**: Cada camada conecta a todas as anteriores
- **Reutilização de features**: Máximo aproveitamento de informação
- **Menos parâmetros**: Mais eficiente que ResNet

MobileNets (2017-2019)

- **Depthwise Separable Convolutions**: Factorização espacial/canal
- **Dispositivos móveis**: Otimizado para baixo consumo
- **MobileNetV2**: Inverted residuals + linear bottlenecks
- **MobileNetV3**: Neural Architecture Search + squeeze-excitation

EfficientNet (2019)

- **Compound scaling**: Balancea width, depth, resolution
- **Neural Architecture Search**: Busca automatizada de arquiteturas
- **Estado da arte**: Melhor trade-off precisão/eficiência

Revolução dos Vision Transformers (2020-2025)

Vision Transformer (ViT) - 2021

- **Patches como tokens**: Divide imagem em patches 16×16
- **Self-attention**: Mecanismo dos Transformers para visão
- **Sem convoluções**: Primeira arquitetura puramente attention-based
- **Escalabilidade**: Performa melhor com datasets muito grandes

Arquiteturas Híbridas

- **ConvNet**: CNN early layers + Transformer later layers
- **Swin Transformer**: Attention hierárquico com janelas
- **CoAtNet**: Combinação otimizada de Conv + Attention

Estado Atual (2025)

- **ConvNext**: CNNs modernizadas competem com ViTs
- **Foundation models**: CLIP, DALL-E, modelos multimodais
- **Trend**: Arquiteturas que combinam o melhor dos dois mundos

Comparativo de Arquiteturas

Arquitetura	Ano	Parâmetros	ImageNet Top-1	Inovação Principal
AlexNet	2012	60M	42.9%	CNNs profundas + GPU
VGG-16	2014	138M	28.1%	Filtros 3×3 uniformes
Inception v3	2016	24M	21.2%	Módulos paralelos
ResNet-50	2015	26M	20.7%	Skip connections
ResNet-152	2015	60M	19.4%	Redes muito profundas
DenseNet-201	2017	20M	18.1%	Conexões densas
EfficientNet-B7	2019	66M	15.7%	Compound scaling
ViT-Large	2021	307M	16.4%	Pure attention
ConvNext-XL	2022	350M	13.9%	CNNs modernizadas

Observações

- **Evolução contínua**: Cada arquitetura trouxe insights únicos
- **Trade-offs**: Precisão vs eficiência vs interpretabilidade
- **Especialização**: Diferentes arquiteturas para diferentes tarefas

Como Escolher uma Arquitetura?

Fatores de Decisão

Recursos Disponíveis

- Poder computacional
- Memória GPU/RAM
- Tempo de treinamento
- Latência de inferência

Natureza dos Dados

- Tamanho do dataset
- Resolução das imagens
- Número de classes
- Complexidade visual

Requisitos da Aplicação

- Precisão necessária
- Tempo real vs batch
- Dispositivo alvo
- Interpretabilidade

Estratégia de Desenvolvimento

- Transfer learning
- Fine-tuning
- Treino do zero
- Ensemble methods

Recomendações por Cenário

Projetos Iniciantes / Prototipagem

- **ResNet-18/34**: Equilibrio simplicidade/performance
- **MobileNetV2**: Para dispositivos com restrições
- **EfficientNet-B0**: Boa baseline moderna

Aplicações de Produção

- **ResNet-50**: Padrão da indústria, muitos modelos pré-treinados
- **EfficientNet-B3/B4**: Melhor custo-benefício
- **ConvNext**: Para máxima precisão

Pesquisa / Estado da Arte

- **ViT-Large**: Para datasets muito grandes
- **Swin Transformer**: Tarefas hierárquicas (detecção, segmentação)
- **Foundation models**: CLIP para multimodalidade

Implementação Prática: ResNet Moderno

Listing 2: ResNet-18 com Melhorias 2025

```python
class BasicBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride=1):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels, out_channels, 3,
                               stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.conv2 = nn.Conv2d(out_channels, out_channels, 3,
                               padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channels)
        self.skip = nn.Identity()
        if stride != 1 or in_channels != out_channels:
            self.skip = nn.Sequential(
                nn.Conv2d(in_channels, out_channels, 1,
                          stride=stride, bias=False),
                nn.BatchNorm2d(out_channels)
            )
        self.relu = nn.ReLU(inplace=True)
```

Implementação Prática: ResNet Moderno

Listing 3: ResNet-18 com Melhorias 2025

```
def forward(self, x):
    identity = self.skip(x)

    out = self.relu(self.bn1(self.conv1(x)))
    out = self.bn2(self.conv2(out))

    out += identity
    return self.relu(out)
```

Tendências Futuras em Arquiteturas CNN

Direções de Pesquisa 2025+

- **Arquiteturas Neurais Buscadas (NAS)**
  - Busca automatizada de arquiteturas otimizadas
  - Especialização para hardware específico
  - Consideração de multiple objetivos (precisão, latência, energia)
- **Modelos Foundation Multimodais**
  - Integração visão + linguagem + audio
  - Transfer learning entre modalidades
  - Zero-shot learning aprimorado
- **Eficiência Energética**
  - Green AI: redução da pegada de carbono
  - Quantização e pruning automáticos
  - Edge computing otimizado
- **Interpretabilidade**
  - Arquiteturas inherentemente interpretáveis
  - Attention visualization melhorada
  - Debugging automático de modelos

Resumo e Conclusões

Evolução das CNNs: Lições Aprendidas

- **AlexNet (2012)**: Provou o poder das CNNs profundas
- **VGG (2014)**: Simplicidade e padronização são valiosas
- **Inception (2014)**: Paralelização e eficiência computacional
- **ResNet (2015)**: Skip connections resolvem problemas de otimização
- **Era moderna**: Híbridos CNN+Transformer dominam

Princípios Fundamentais

- **Não existe "melhor" arquitetura**: Depende do contexto
- **Trade-offs são inevitáveis**: Precisão vs eficiência
- **Transfer learning é essencial**: Aproveite modelos pré-treinados
- **Experimente sistematicamente**: Dados empíricos são cruciais

Próxima Aula

- **Tópico**: Técnicas Avançadas de Treinamento
- **Conteúdo**: Otimizadores, regularização, data augmentation