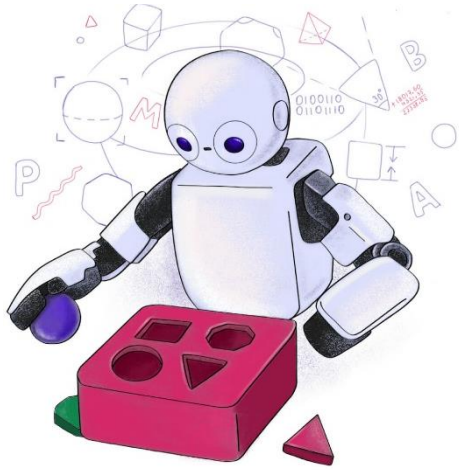
TP558 - Tópicos avançados em aprendizado de máquina: *MobileNets*

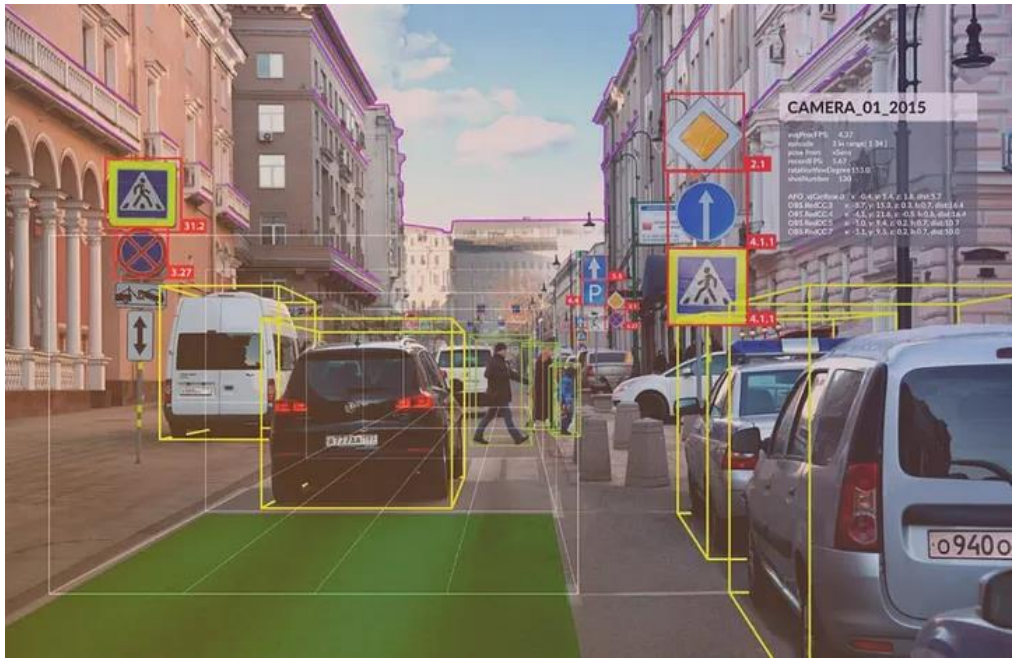


Inatel

Adrian A. C. Alanes
adrian@mtel.inatel.br

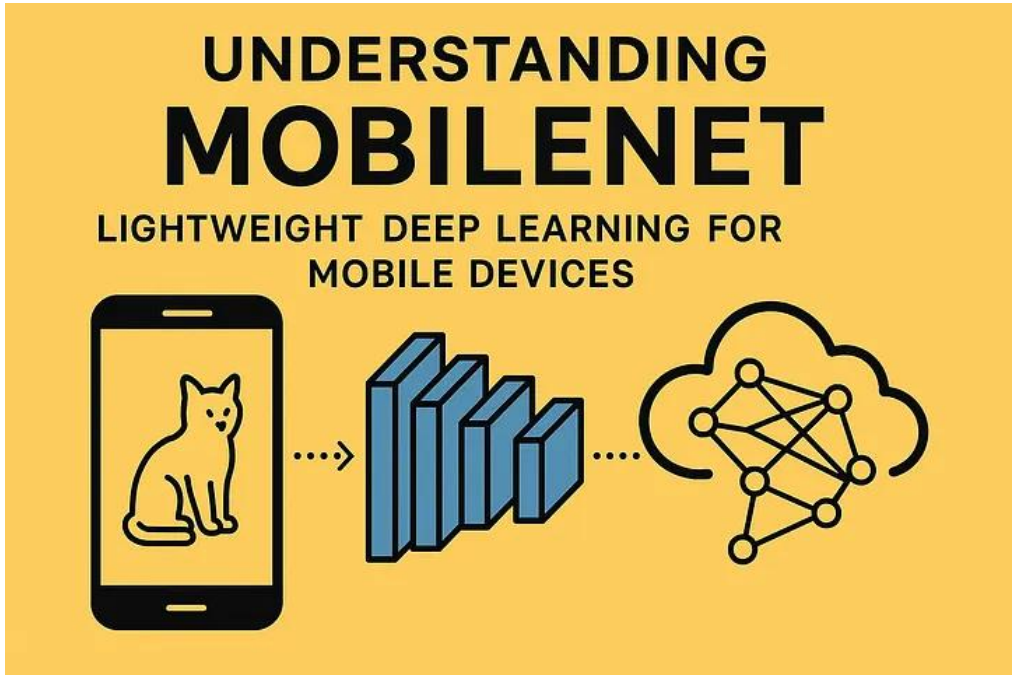
Introdução

- Desde a AlexNet (2012), as CNNs se tornaram o padrão em visão computacional. A tendência era tornar as redes mais profundas e obter mais precisão.
- VGGNet (2014): 16-19 camadas e 138 milhões de parâmetros; muito caro em termos de memória e computação.
- ResNet (2015): até 152 camadas; conexões residuais que permitem o treinamento de redes ultraprofundas com alta demanda computacional.



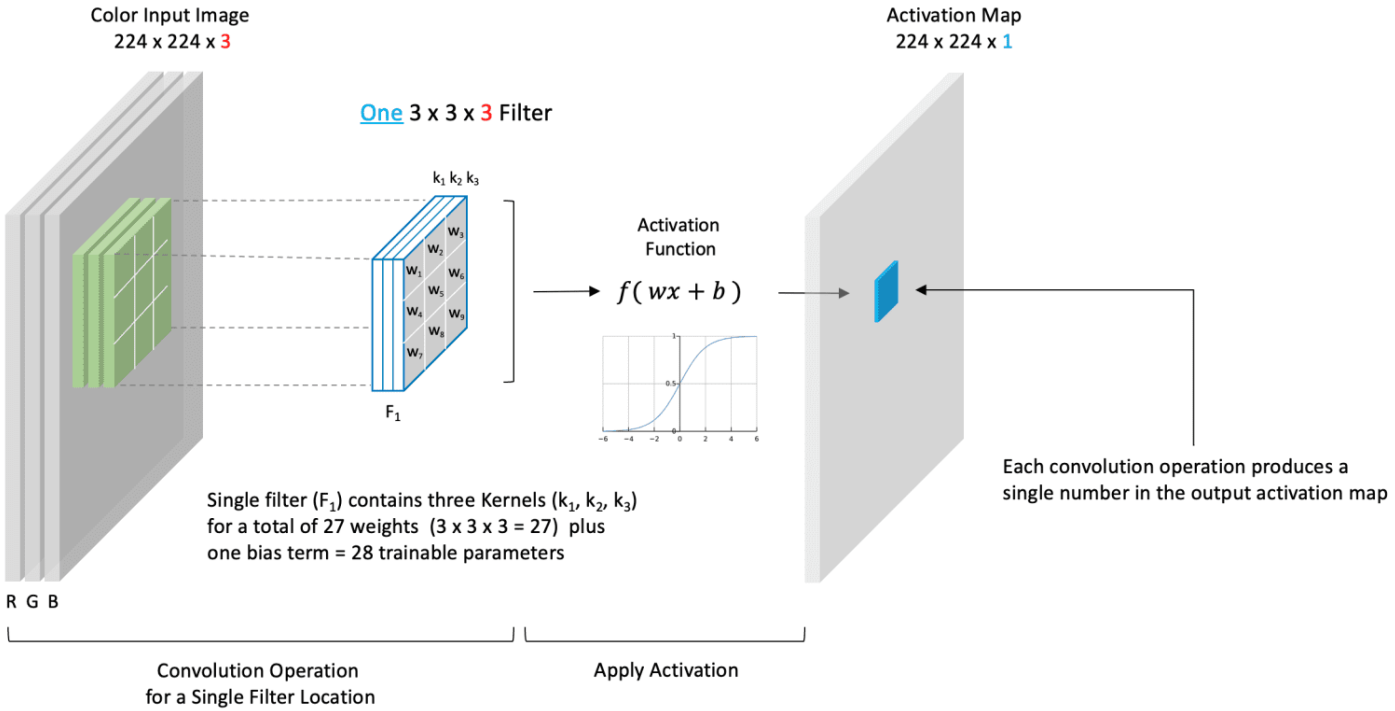
Introdução

- O MobileNet (2017) aprende com essas arquiteturas, mas se concentra em outro objetivo: eficiência em dispositivos móveis.
- Enquanto o AlexNet, o VGG e o ResNet buscavam mais precisão a qualquer custo, o MobileNet aplica convoluções separáveis por profundidade para reduzir drasticamente os parâmetros e as operações e, ao mesmo tempo, manter uma boa precisão.
- Assim, ele representa uma evolução paralela à "corrida de profundidade", mas com foco na computação leve.



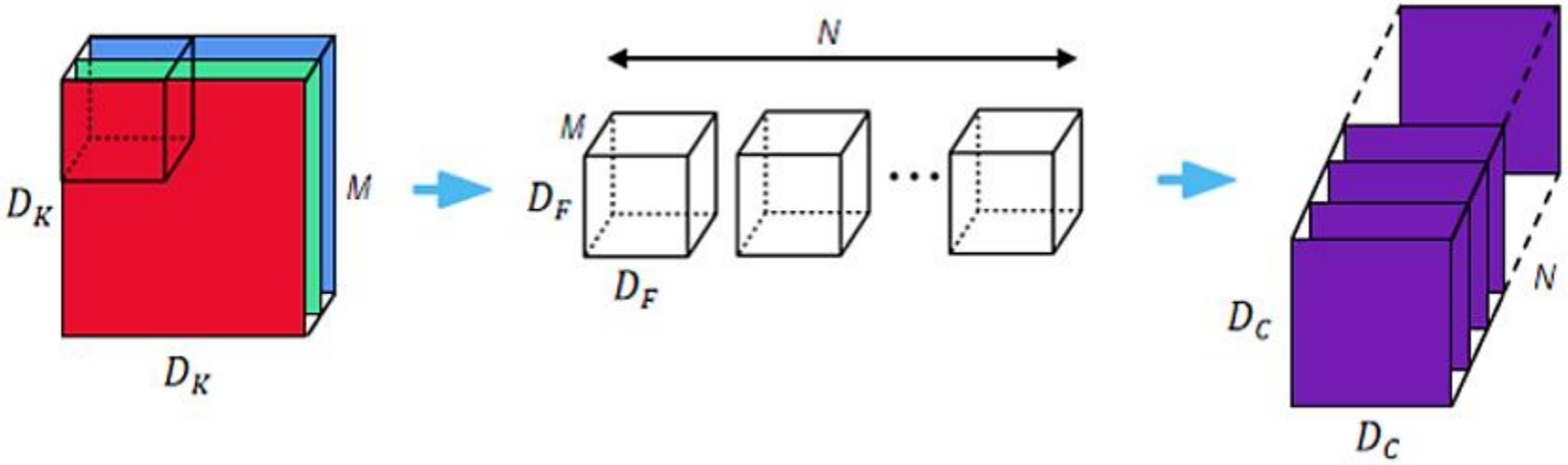
Fundamentação teórica

Em uma **CNN padrão**, cada filtro aplica operações (convolução) em todos os canais de entrada - como R, G, B - simultaneamente:



Fundamentos teóricos

Essa operação é poderosa e envolve mais cálculos e maior complexidade, além de ser altamente dispendiosa em termos de recursos computacionais.



$Costo\ Computacional$:

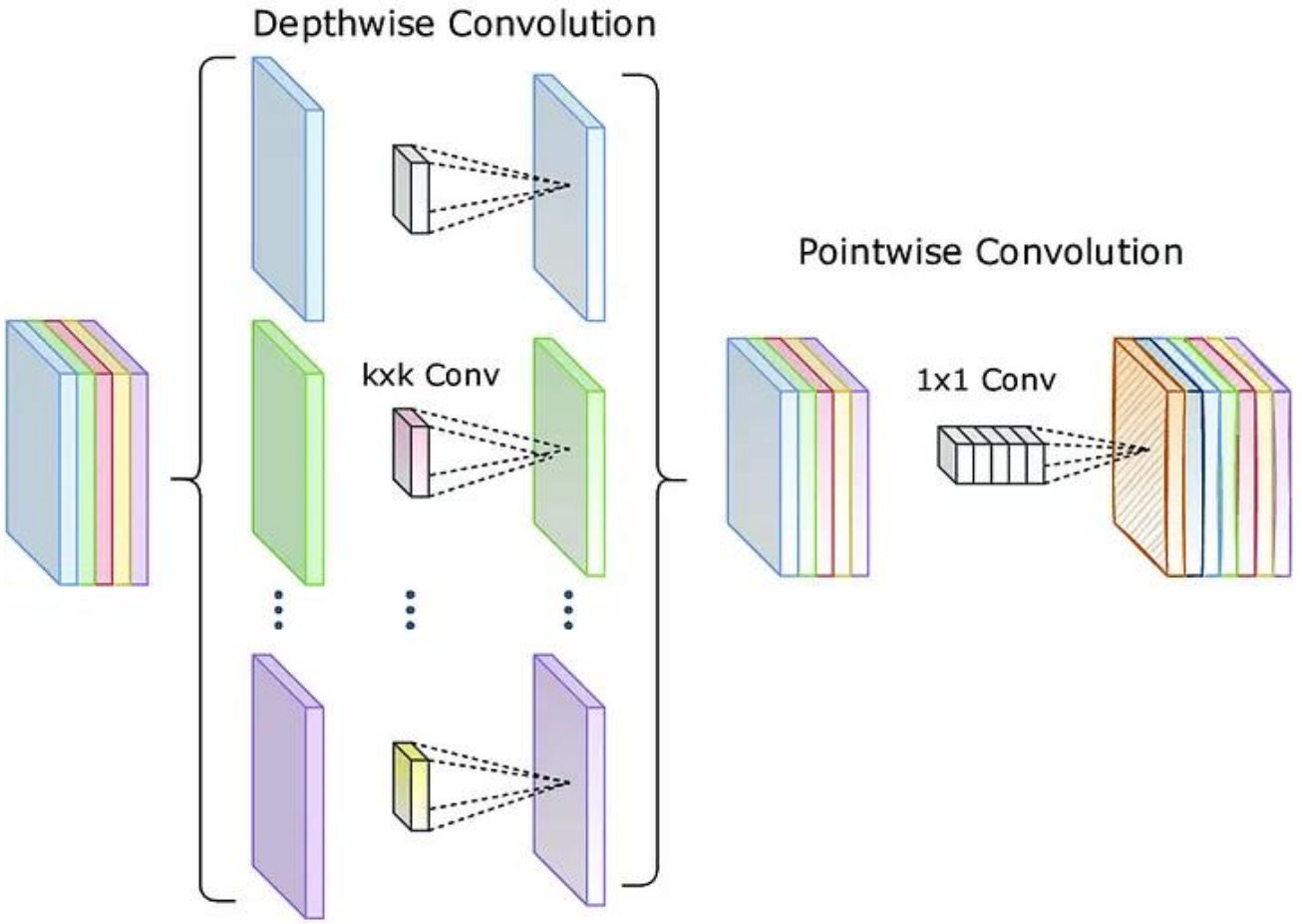
$$(D_K * D_K * M) * (D_F * D_F) * N = D_K^2 * M * N * D_F^2$$

Contexto teórico

No **MobileNet**, essa operação é dividida em duas etapas:

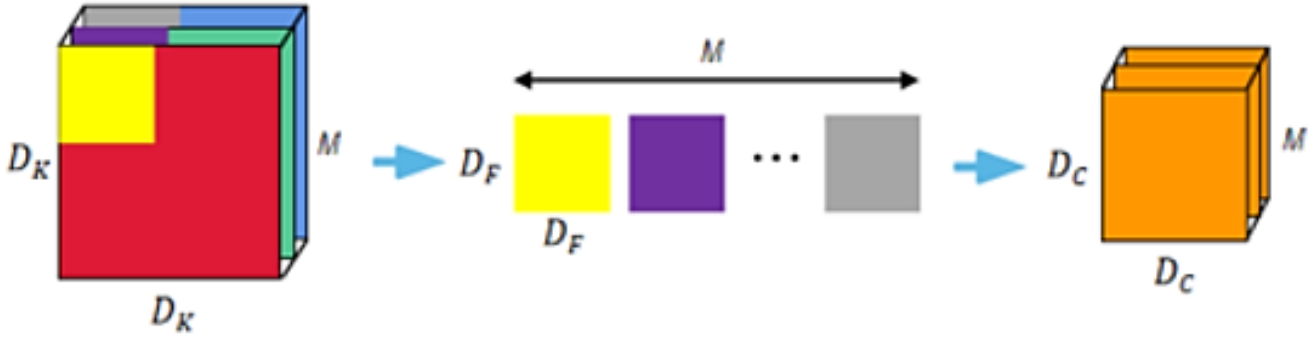
- **Em profundidade:** um filtro por canal → filtro.
- **Pointwise (1×1):** combinar os canais → mix.

Essa separação reduz drasticamente o custo computacional.



Contexto teórico

Convolução em profundidade: aplica um kernel independente em cada canal e apenas filtra, não combina informações entre canais.

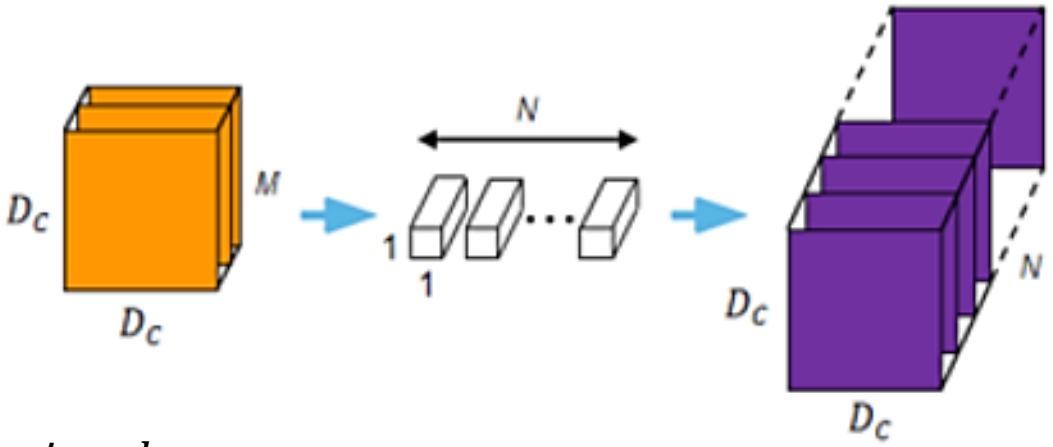


$Costo\ Computacional$:

$$D_K * D_K * M * D_F * D_F$$

Contexto teórico

Convolução pontual (1×1): usa kernels 1×1 para combinar os M canais e produzir N saídas. É a etapa que mistura as informações de todos os canais.



$Costo\ Computacional$:

$$M * N * D_F * D_F$$

Fundamentos teóricos

A **combinação** da convolução em profundidade e da convolução pontual é chamada de **convolução separável em profundidade**.

$Costo\ Computacional$:

$$D_K^2 * M * D_F^2 + M * N * D_F^2$$

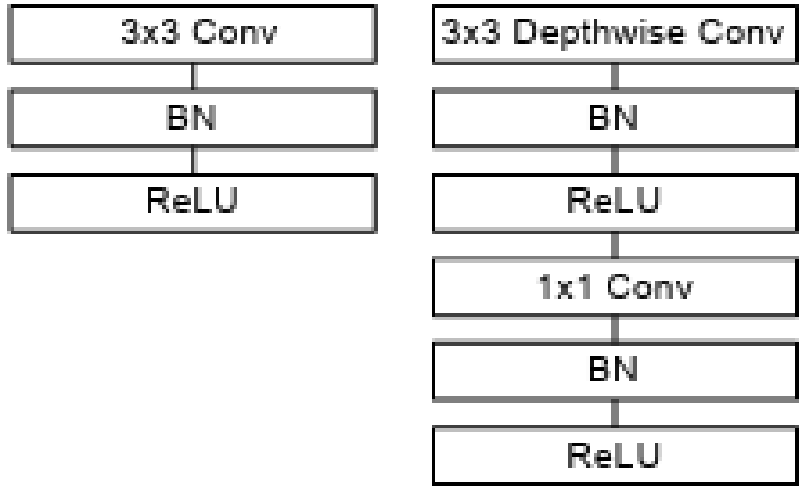
O MobileNet usa **a convolução separável em profundidade 3x3**, o que resulta em um custo computacional 8 a 9 vezes menor do que a convolução padrão, com uma pequena redução na precisão.

$Relacion\ Costo\ Computacional$:

$$\frac{Costo_{mobilenet}}{Costo_{estandar}} = \frac{1}{N} + \frac{1}{D_K^2}$$

Arquitetura e operação
Estrutura do MobileNet v1

- **CNN:** camadas convolucionais padrão seguidas por BatchNorm + ReLu
- **MobileNet:** a rede é baseada em convoluções separáveis em profundidade em todas as camadas, exceto na primeira (conv padrão 3×3). Cada camada é seguida por BatchNorm + ReLU.



Arquitetura e operação

A arquitetura inclui:

- 1 convolucional padrão inicial na inicialização:
- 13 blocos separáveis (cada um com depthwise + pointwise).
- Um pooling médio global.
- 1024 → 1000 classes totalmente conectadas.
- Softmax como classificador

Contando separadamente o depthwise e o pointwise, o MobileNetV1 tem **28 camadas** no total.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Evolução arquitetônica: v2, v3, v4

Versão	Inovações	Ativações	Aplicativos
V1 (2017)	Dividir as convoluções em etapas menores (*separáveis em profundidade*) → modelo muito mais leve.	ReLU6	Primeiros aplicativos móveis de visão computacional (classificação em fotos, reconhecimento básico).
V2 (2018)	Blocos mais eficientes (*resíduos invertidos*) → fazem melhor uso das informações e melhoram a precisão sem aumentar o tamanho.	ReLU6 + saída linear	Reconhecimento de objetos em tempo real (SSDLite), segmentação leve.
V3 (2019)	Ajustado automaticamente para celular (*NAS*), adiciona atenção ao canal (*SE*) e ativação mais eficiente (*h-swish*).	ReLU + h-swish	Uso estendido em aplicativos móveis (detecção de câmera, filtros de AR, aplicativos de saúde/fitness).
V4 (2024)	Blocos "universais" que funcionam bem em CPUs, GPUs e NPUs; adiciona atenção leve (*Mobile MQA*) e técnicas de aprendizado mais avançadas.	h-swish + ReLU	Modelos mais precisos para dispositivos modernos: assistentes inteligentes, visão de drones, IoT avançada.

Treinamento e otimização

O MobileNet foi treinado no TensorFlow, usando o otimizador RMSProp.

- **RMSProp:** adapta a taxa de aprendizado a cada parâmetro, permitindo um treinamento mais estável e mais rápido.

O procedimento seguiu a mesma estratégia do InceptionV3.

- **Descida de gradiente assíncrona:** vários processadores treinam em paralelo e atualizam os parâmetros sem esperar que todos eles terminem.

Todas as variantes do MobileNet podem ser treinadas com a configuração básica.

- Resultados diferentes ao alterar apenas os hiperparâmetros **(α e ρ)**.

Treinamento e otimização

Regularização na MobileNet	• Usa menos regularização do que as redes maiores. • Não usa técnicas adicionais, como suavização de rótulos ou cabeças laterais.
Aumento de dados	• Foram aplicadas transformações mais simples: recorte moderado, sem distorções agressivas.
Decaimento de peso (regularização L2)	• Pouco ou nenhum aplicado a filtros de profundidade, porque eles já têm poucos parâmetros.
Função de ativação	• Todas as camadas convolucionais foram seguidas por ReLU (Rectified Linear Unit), que introduz a não linearidade e acelera o treinamento.

Hiperparâmetro α (*Multiplicador de largura*)

- **Definição: α** ∈ (0,1] reduz uniformemente os canais de entrada e saída em todas as camadas convolucionais.
- **Efeito:** ao reduzir os canais, o custo e os parâmetros diminuem aproximadamente com α^2, o que torna o MobileNet adaptável a diferentes níveis de eficiência.

$Costo\ Computacional$:

$$D_K^2 * (\alpha M) * D_F^2 + (\alpha M) * (\alpha N) * D_F^2$$

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Hiperparâmetro ρ (*Multiplicador de resolução*)

- **Definição:** reduz a resolução espacial da imagem de entrada e de todos os mapas de recursos internos da rede.
- **Efeito:** como o custo da convolução depende do tamanho espacial Df, ao reduzir a resolução, o número de operações diminui em aproximadamente ρ^2.

$Costo\ Computacional$:

$$D_K^2 * M * (\rho D_F^2) + M * N * (\rho D_F^2)$$

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

Vantagens

- ✔ Redução de cálculo ≈8-9× com conv. separável em profundidade.
- ✔ Modelo muito mais leve (~4,2 milhões de parâmetros).
- ✔ Bom desempenho no ImageNet com baixo custo.
- ✔ Escalável com α (multiplicador de largura).
- ✔ Escalável com ρ (multiplicador de resolução).
- ✔ Compatível com bibliotecas otimizadas (GEMM).
- ✔ Ideal para sistemas móveis, de IoT e incorporados.

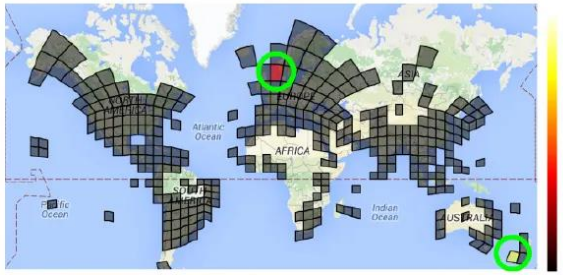
Desvantagens

- ✗ Menor precisão do que arquiteturas maiores.
- ✗ Dependência de convoluções 1×1: embora eficientes, elas representam 95% do cálculo e 75% dos parâmetros → tornam-se o gargalo.
- ✗ Extremidade pesada totalmente conectada: concentra grande parte dos parâmetros restantes.
- ✗ Rápida degradação da precisão com a redução excessiva de α ou ρ.
- ✗ Menor generalização em tarefas complexas (detecção/segmentação) em comparação com modelos mais profundos.
- ✗ Destinado ao hardware de 2017: as versões posteriores (V2, V3) superam-no claramente em termos de precisão e eficiência.

Exemplo(s) de aplicativo

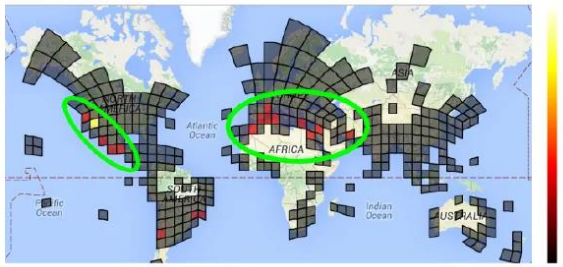
Projeto / Tarefa	Descrição do projeto	Principais resultados
PlaNet (Geolocalização)	Substituiu o Inception V3 pelo MobileNet v1 para classificar a localização de fotos.	Parâmetros reduzidos: 52M → 13M e precisão semelhante.
Atributos faciais	O MobileNet v1 foi usado como aluno de um modelo grande (≈75 milhões de parâmetros).	MobileNet v1: 4 milhões de parâmetros, alcançando uma precisão média semelhante à do modelo grande.

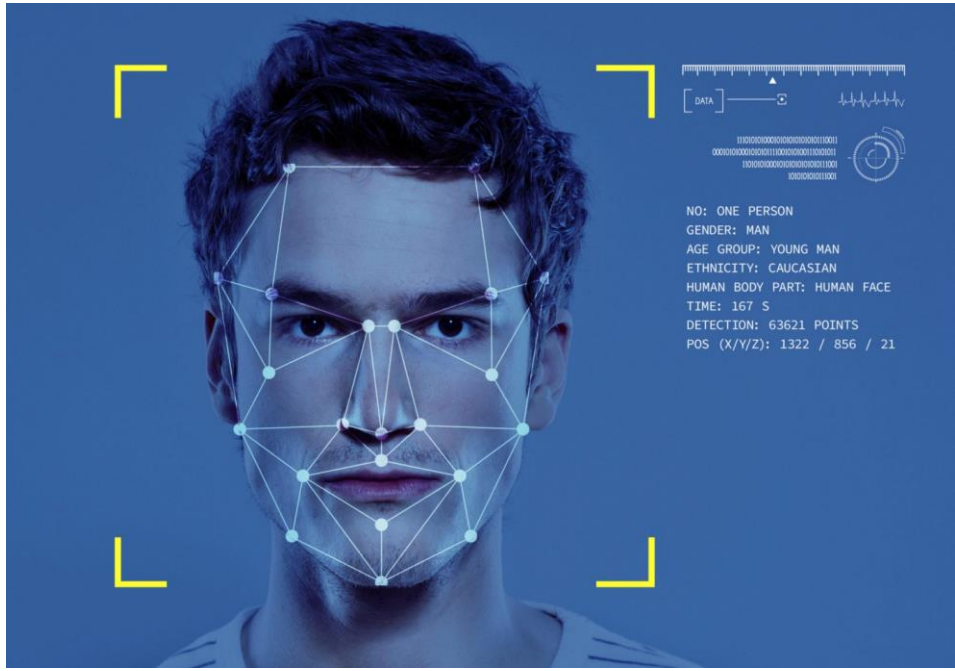
Photo CC-BY-NC by edwin.11



(b)

Photo CC-BY-NC by jonathanfh





Exemplo(s) de aplicativo(s)

Objetivo: treinar o MobileNetV1 do zero em *Cats vs Dogs.*

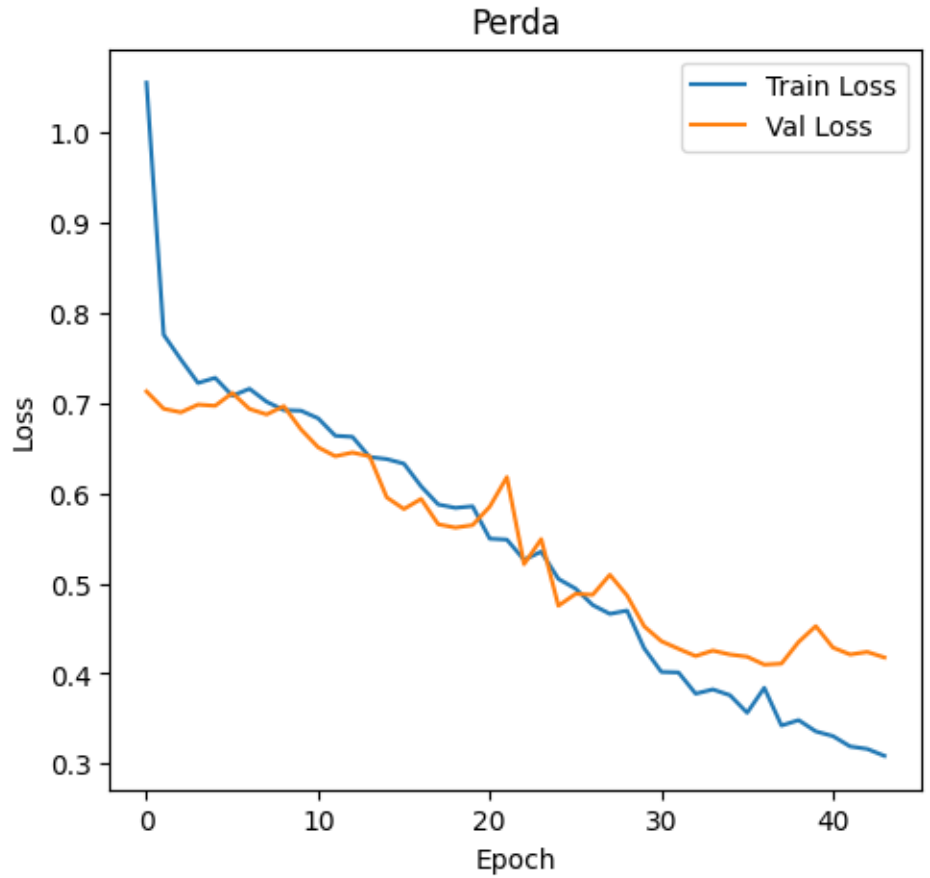
Metodologia:

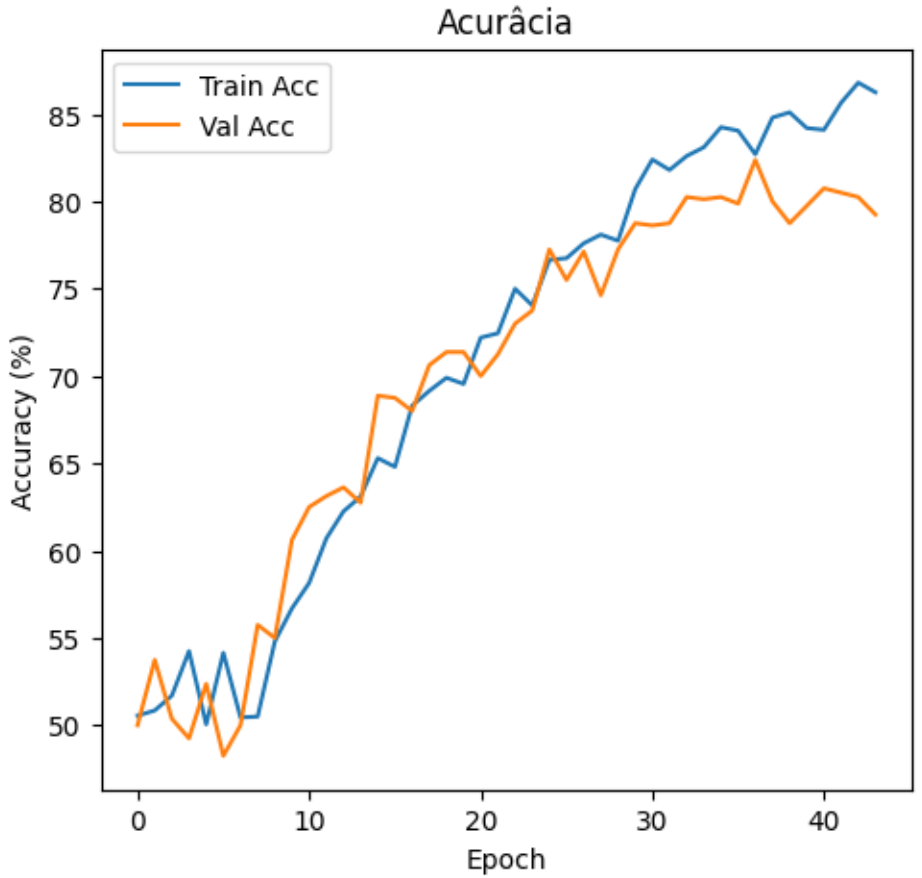
- Conjunto de dados: treinar **2000**, avaliar **800**, testar **200**
- Aumento de dados: flips, rotações, recorte, jitter, cutout
- Modelo: MobileNetV1 (Convs separáveis em profundidade, ReLU6, Dropout=0,5, BatchNorm)
- Otimizador: RMSProp + Decaimento de peso
- Estratégia: parada antecipada + agendador de LR

Principais resultados:

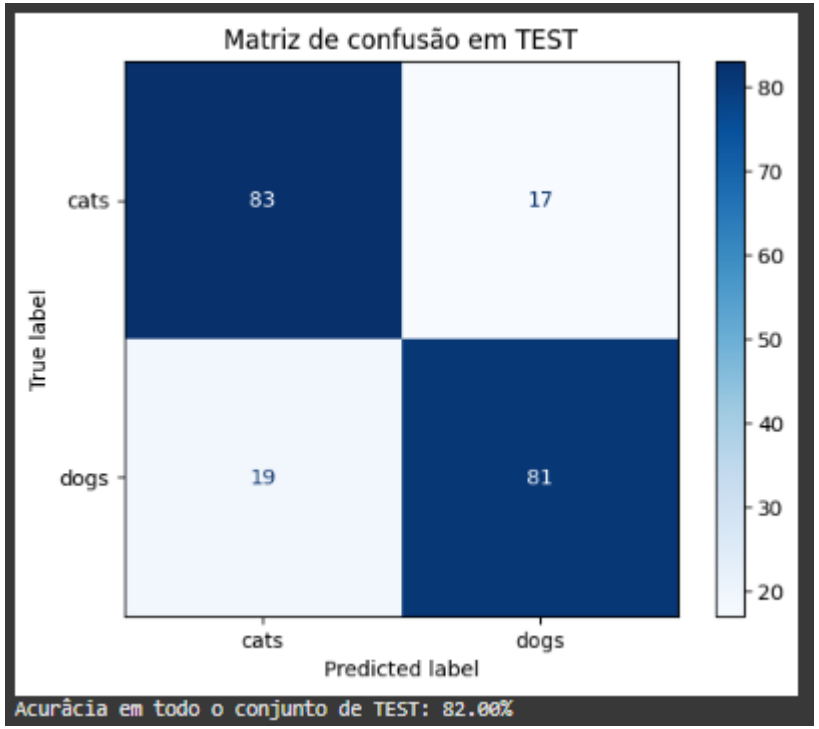
- Precisão da validação ≈ **80%.**
- Precisão do teste ≈ **78-82%.**
- Parâmetros: **~4,2M (~16 MB em disco)**

Exemplo(s) de aplicação





Exemplo(s) de aplicação



Comparação com outros algoritmos

- O MobileNetV1 foi projetado para oferecer um equilíbrio entre eficiência computacional e precisão, o que o diferencia de arquiteturas maiores. Embora sacrifique um pouco da precisão, sua principal contribuição é a redução drástica de parâmetros e operações, o que o torna viável em dispositivos móveis e sistemas incorporados.

Arquitetura	Precisão Top-1	Parâmetros (M)	Operações (Mult-Adds)	Ano
VGG16	71.5%	138 M	15.3 B	2014
ResNet-50	76.0%	25.6 M	3.8 B	2015
Inception V3	78.0%	23.2 M	5.7 B	2015
MobileNetV1 (α=1, 224×224)	70.6%	4.2 M	569 M	2017
MobileNetV1 (α=0,5, 160×160)	60-63%	1.3 M	150 M	2017

Dúvidas?

Referências

- Howard, A. et al. (2017). MobileNets: redes neurais convolucionais eficientes para aplicativos de visão móvel. arXiv:1704.04861.
- Simonyan, K. & Zisserman, A. (2014). Redes convolucionais muito profundas para reconhecimento de imagens em grande escala (VGG). arXiv:1409.1556.
- He, K. et al. (2015). Aprendizagem residual profunda para reconhecimento de imagens (ResNet). arXiv:1512.03385.
- Tan, M. & Le, Q. (2019). EfficientNet: repensando o dimensionamento de modelos para redes neurais convolucionais. arXiv:1905.11946.
- Sandler, M. et al. (2018). MobileNetV2: resíduos invertidos e gargalos lineares. arXiv:1801.04381.
- Howard, A. et al. (2019). Searching for MobileNetV3. arXiv:1905.02244.
- Qin, D. et al. (2024). MobileNetV4: Modelos universais para o ecossistema móvel. arXiv:2404.10518.

Obrigado!

Links

- Github: https://github.com/aadlrei/TP___558-Topicos-Avancados-em-Aprendizado-de-Maquina.git
- Questionário: https://forms.gle/5NGvwrwecRrspMH36