Redes Neurais Convolucionais

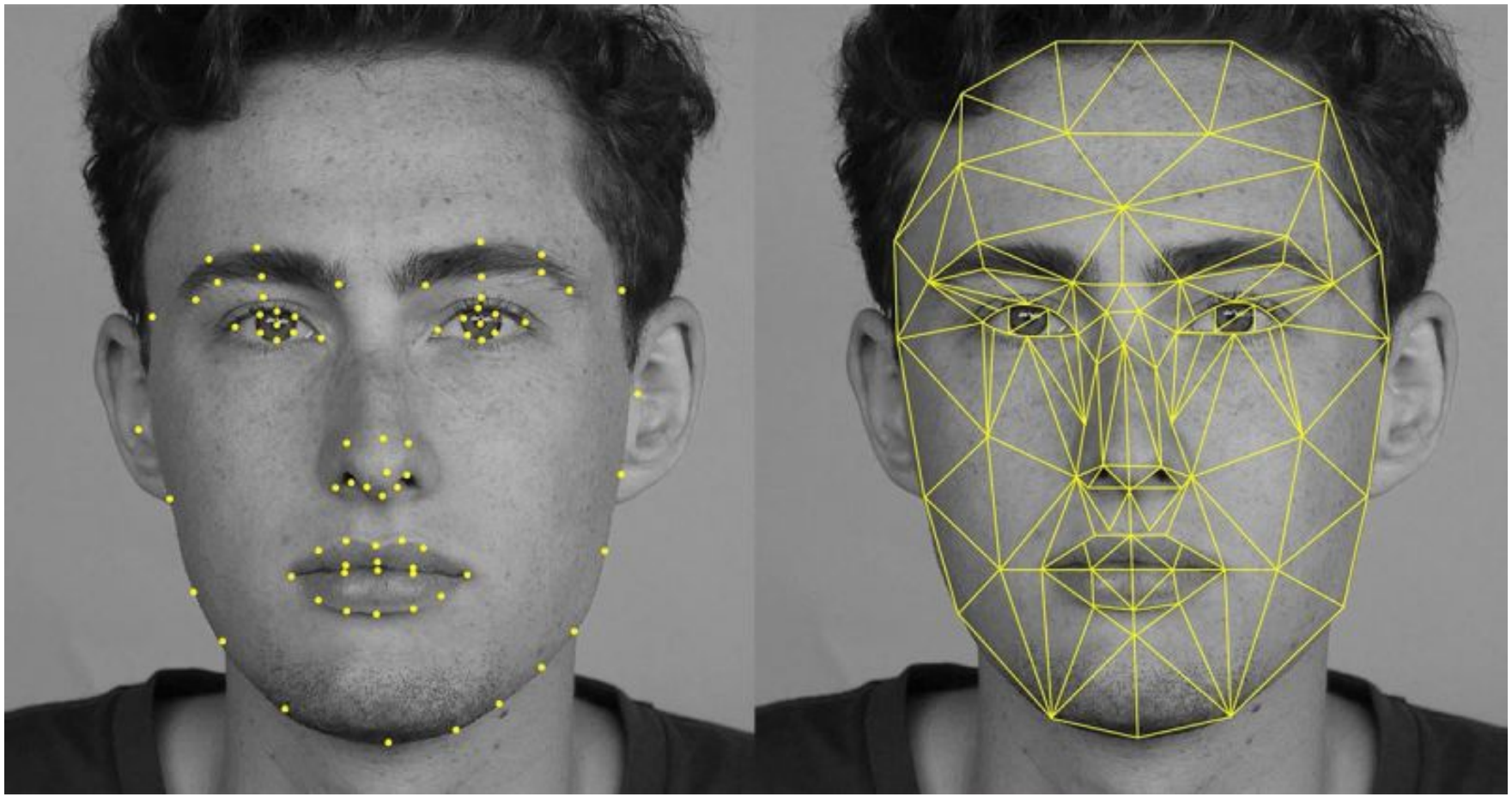
Jorge K. S. Kamassury

EEL7514/EEL7513 - Tópico Avançado em Processamento de Sinais
EEL410250 - Aprendizado de Máquina

EEL / CTC / UFSC

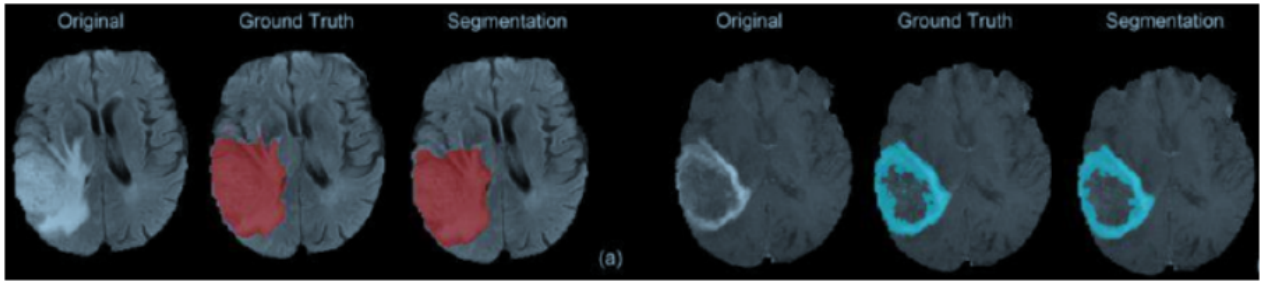
Algumas aplicações

Detecção e reconhecimento facial

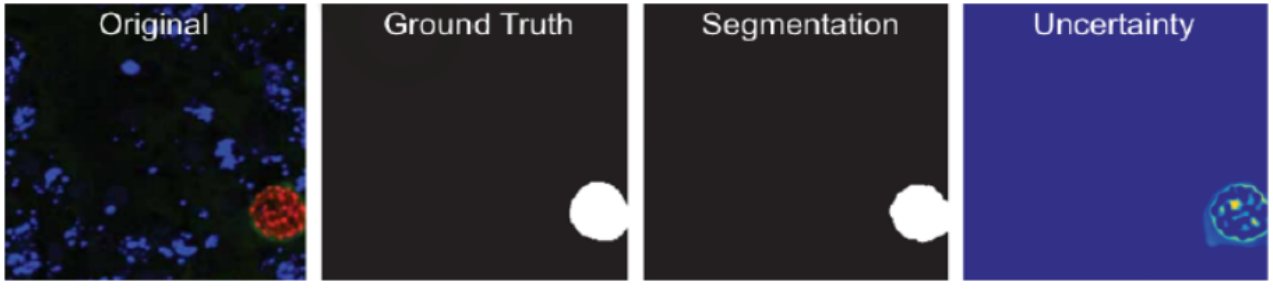


Segmentação semântica: Análise biomédica

Tumor Cerebral

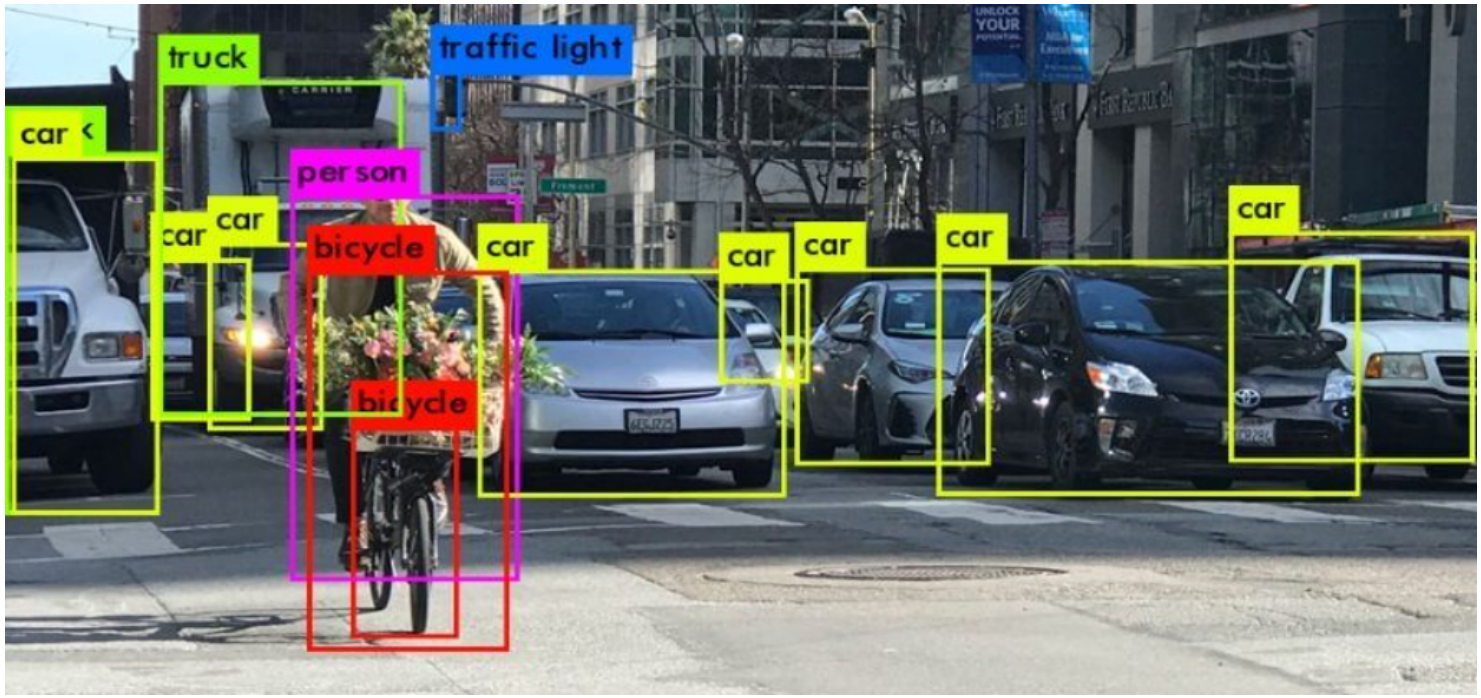


Infecção devido à malária



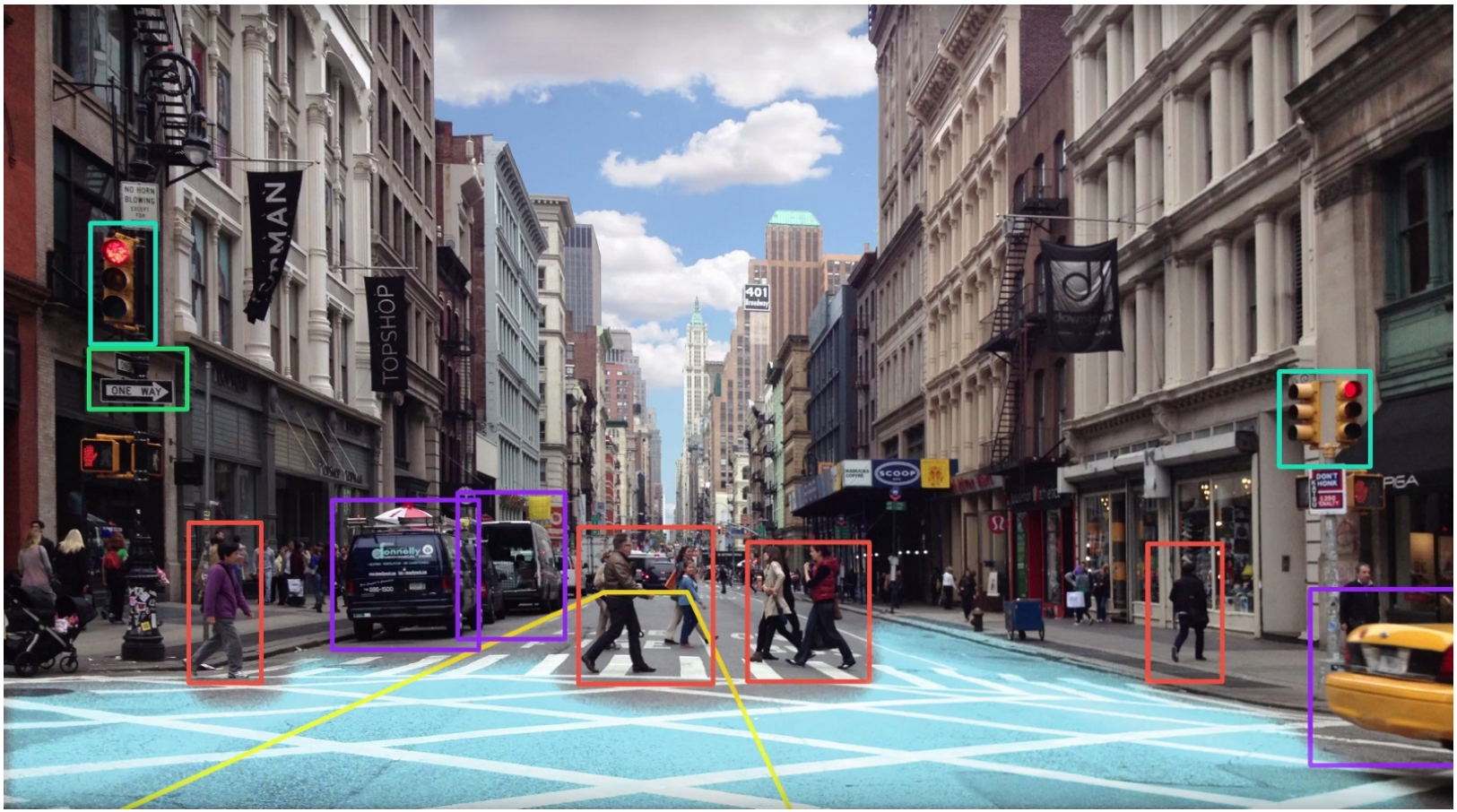
- Detecção/classificação para cada pixel: problema complexo;
- Deep convolutional inverse graphics networks (DCIGNs).

Detecção e localização



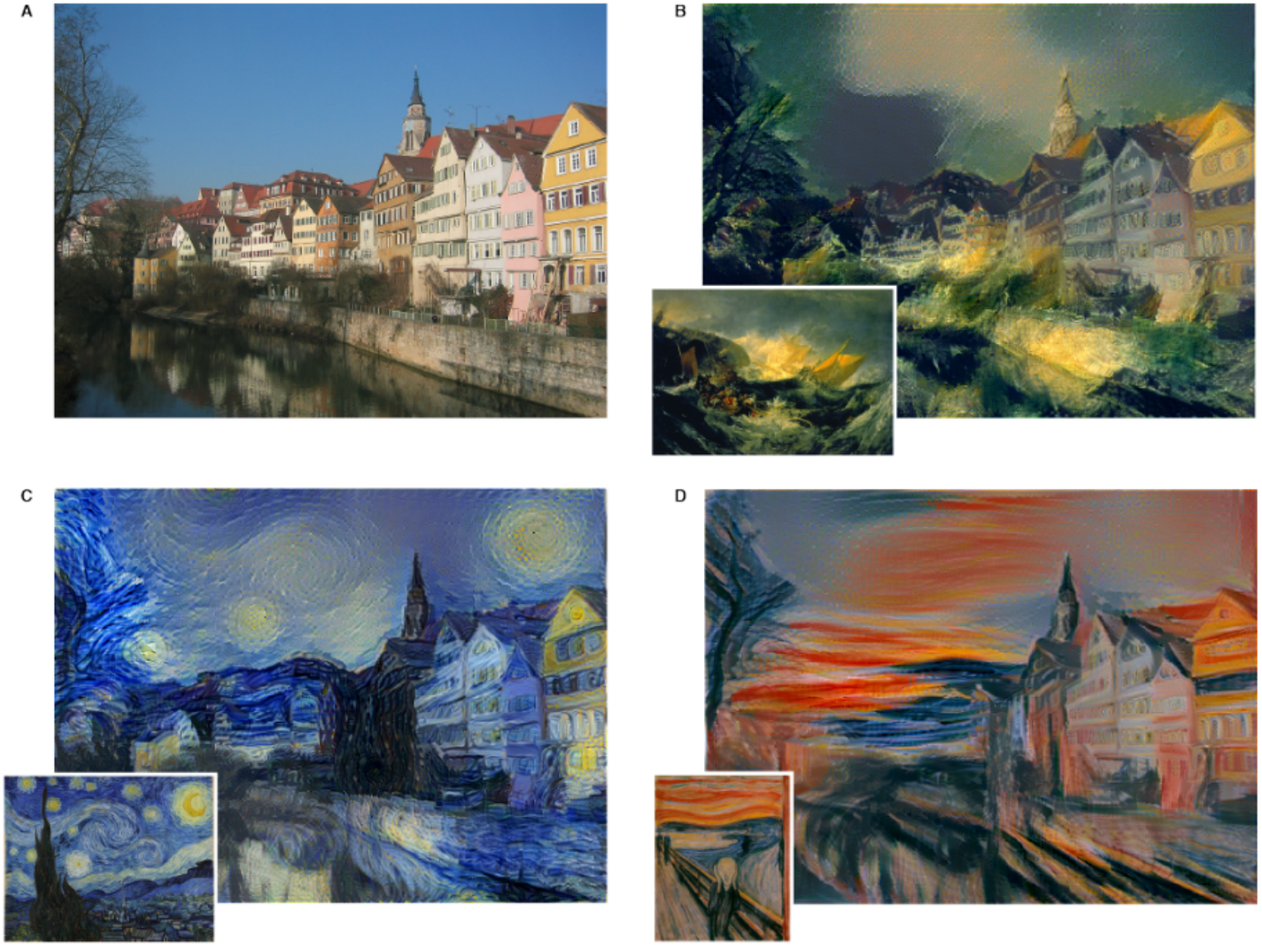
- Region-based Convolutional Neural Networks (R-CNN).

Navegação/Direção autônoma



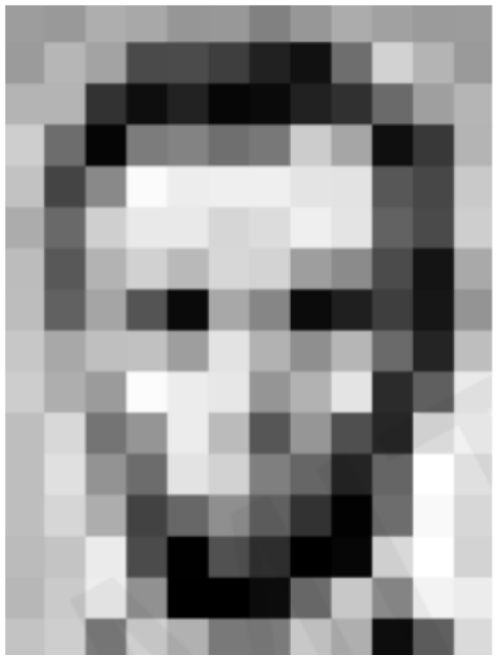
- End-to-End Deep learning: Entrada (imagens) e saída (condução);

Transferência de estilo: arte



- Conteúdo e referência de estilo.

Imagens são números



157	153	174	158	150	152	129	151	172	161	155	156
155	182	163	74	76	62	33	17	110	210	180	154
180	180	50	14	84	6	10	33	48	105	159	181
206	109	6	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	160
189	97	166	84	10	168	134	11	31	62	22	148
189	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	239
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	4	217	255	211
183	202	237	145	0	0	12	108	200	138	245	235
199	206	123	207	177	121	123	200	175	13	96	218

157	153	174	158	150	152	129	151	172	161	155	156
155	182	163	74	76	62	33	17	110	210	180	154
180	180	50	14	84	6	10	33	48	105	159	181
206	109	6	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	160
189	97	166	84	10	168	134	11	31	62	22	148
189	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	239
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	4	217	255	211
183	202	237	145	0	0	12	108	200	138	245	235
199	206	123	207	177	121	123	200	175	13	96	218

- Imagem é apenas uma matriz de números $[0, 255]$

Detecção de *features* de nível alto

- Identificar as principais *features* em cada categoria de imagem

Olhos, nariz, boca, ...

Escadas, porta, janelas ...

Faróis, rodas, placa, ...

Extração manual de recursos

Problemas?

Extração manual de recursos

Conhecimento de domínio → Definir features → Detectar features p/ classificar

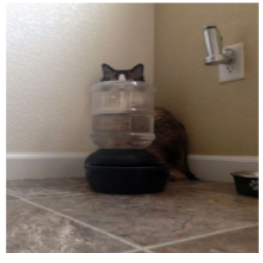
Variação do ponto de vista

Desordem de fundo

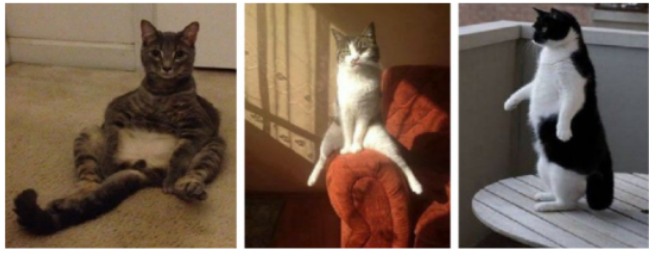
Variação de escala

Condições de iluminação

Oclusão



Deformação

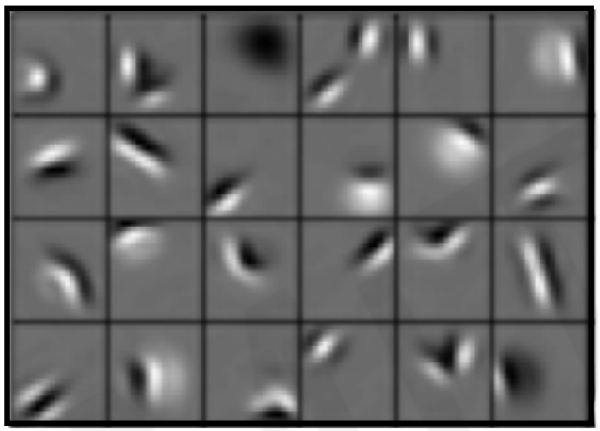


Variação Intraclasse

Representação de *features* de aprendizagem

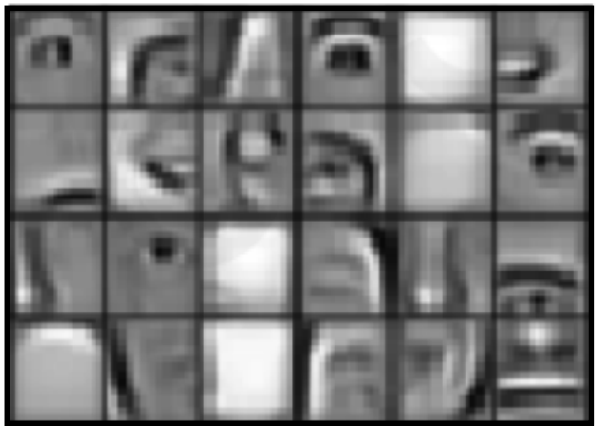
- Questão: Como aprender a hierarquia das *features* diretamente dos dados em vez de engenharia manual?

Features de nível baixo



Bordas, manchas escuras, ...

Features de nível médio

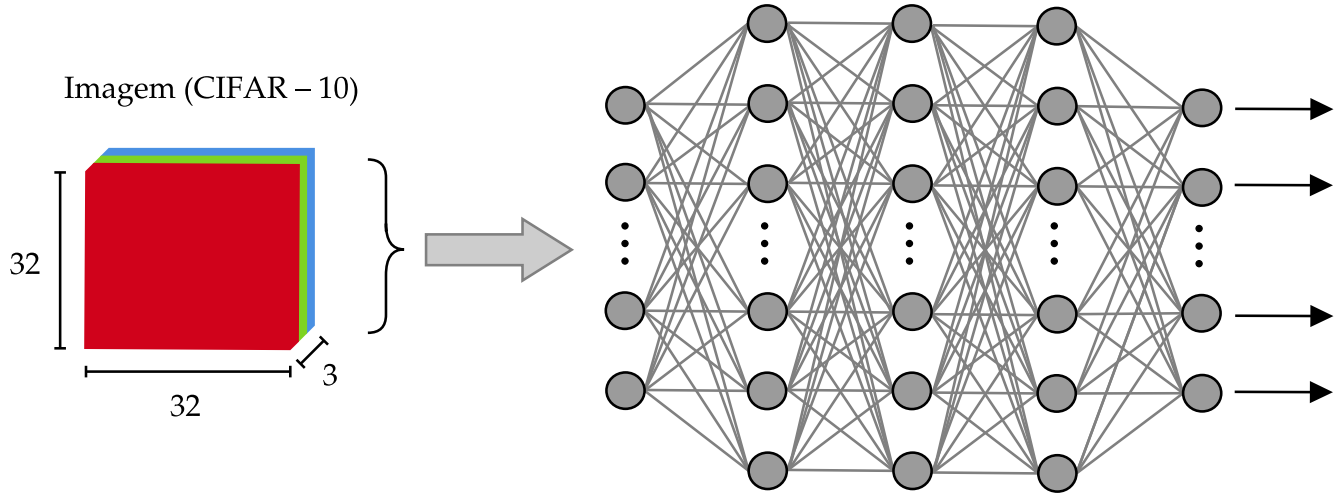


Olhos, orelhas, nariz, ...

Features de nível alto

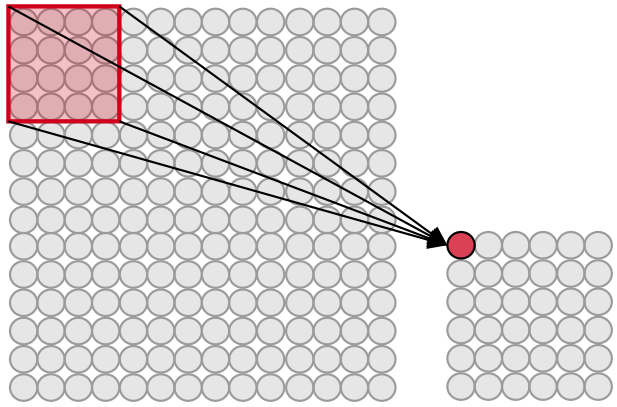
Estrutura facial

Redes Totalmente Conectadas



- Para a primeira camada oculta: um único neurônio terá 3072 pesos;
- Para uma imagem maior (p. ex., $200 \times 200 \times 3$): um único neurônio terá 120000 pesos;
- Problemas:
  - Muitos parâmetros: treinamento lento;
  - Muitos dados para evitar *overfitting*;
  - Sem informação espacial.

Como explorar a informação espacial?

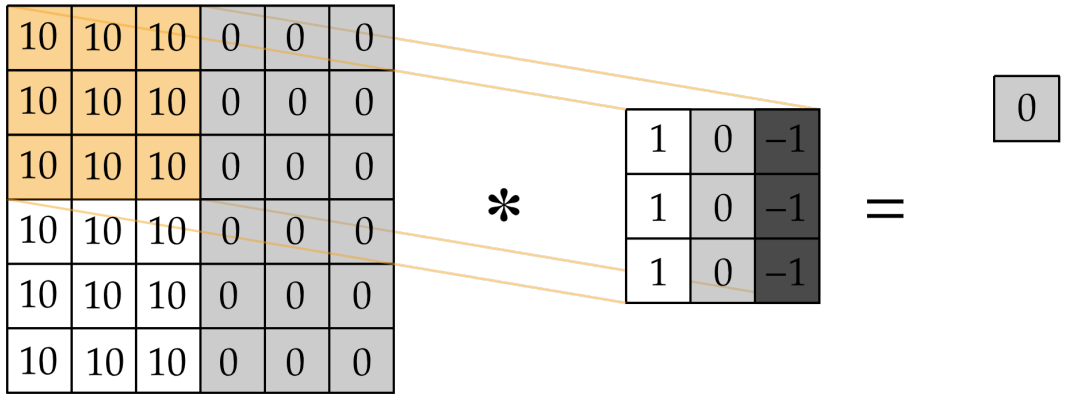


- Conectividade local
  - Um neurônio em uma camada é ligado a um subconjunto de saídas da camada anterior;
  - Múltiplos neurônios podem ser conectados à mesma região do volume de entrada para sua camada;
  - Permite a detecção de *features* locais em pequenas regiões.
- Compartilhamento de pesos
  - Uso da janela de convolução para restringir o compartilhamento das conexões;
  - Permite a detecção das mesmas *features* locais em toda a imagem.

Aplicando filtros para extrair *features*

- Aplica um conjunto de pesos (filtros) para extrair as *features* locais;
- Emprega vários filtros para extrair diferentes *features*;
- Compartilhe espacialmente os parâmetros de cada filtro.

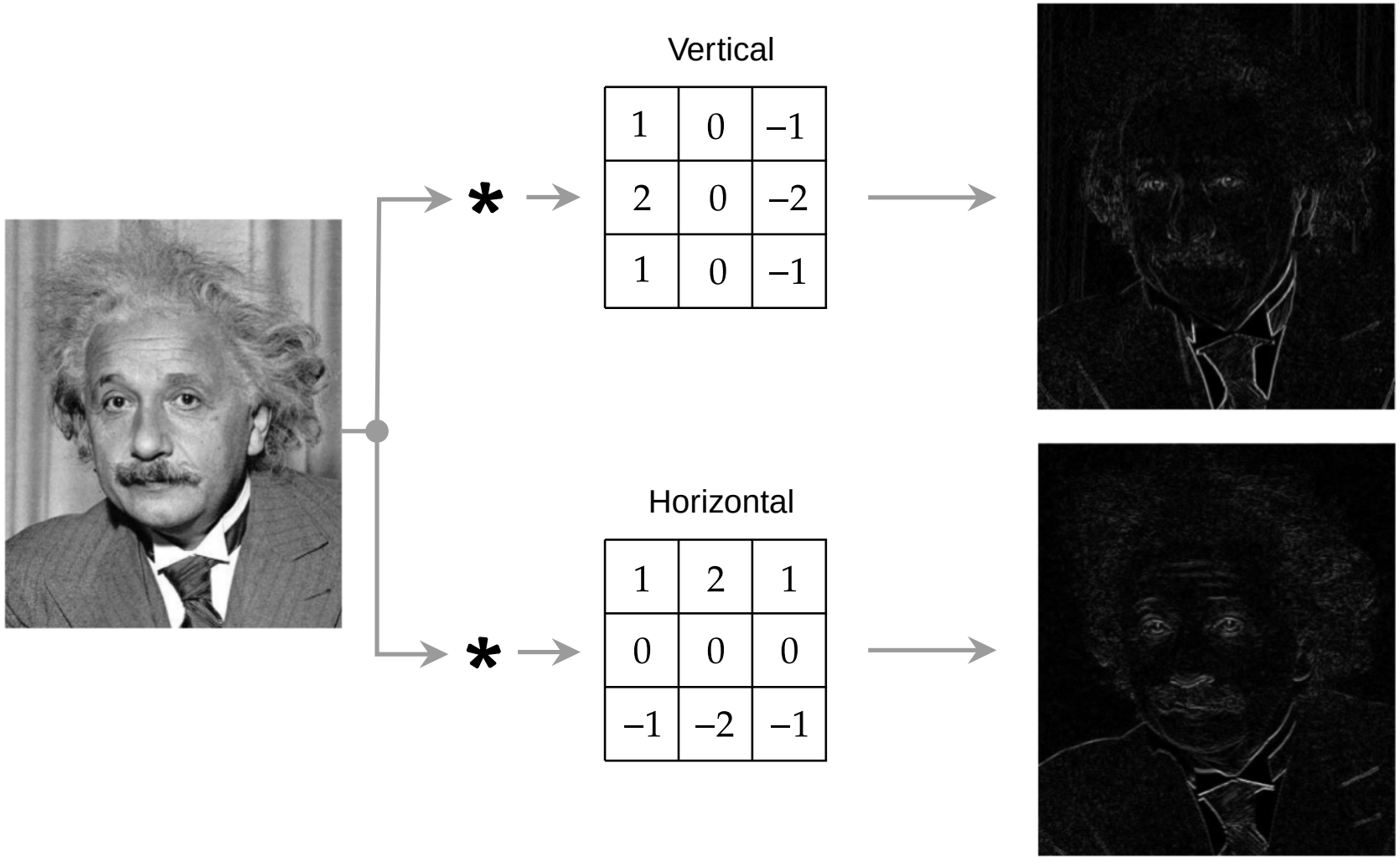
Exemplo: Extração de *features* com convolução



- Filtro/Kernel (3×3): matriz de pesos $\mathsf{K}_{i,j}$;
- Deslocamento unitário por conjunto (janela);
- Aplicação do mesmo filtro aos 3×3 conjuntos da entrada (I);

$$Conv_{x,y} = \sum_{i=1}^{3} \sum_{j=1}^{3} \mathsf{K}_{i,j} \mathsf{I}_{x+i-1,y+j-1}$$

Exemplo: Filtro de Sobel



Exemplo: Feature Maps

Original

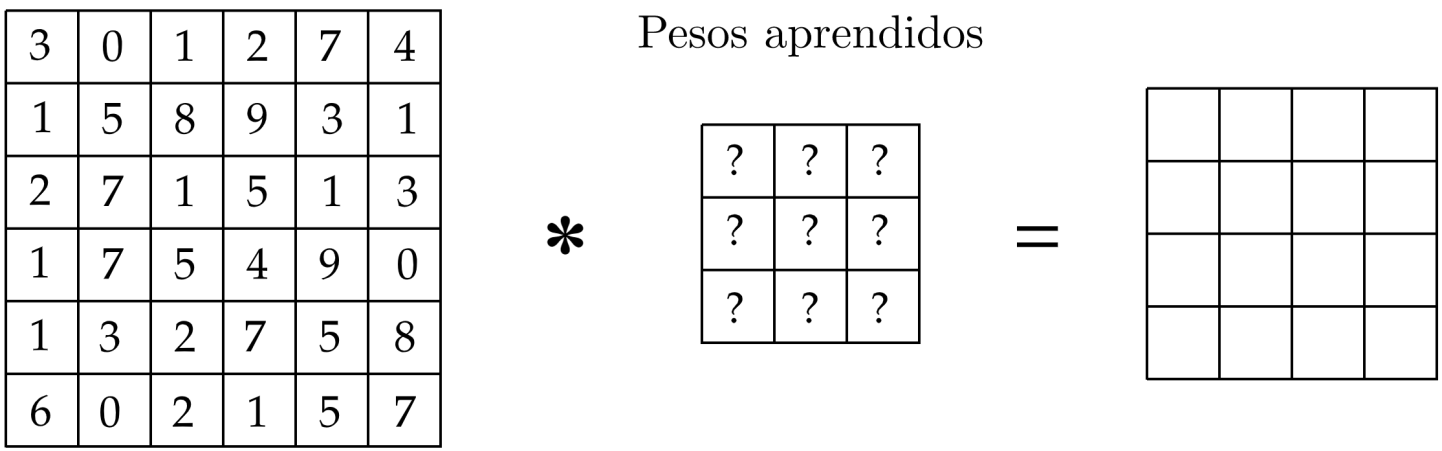
Sharpen

Edge Detect

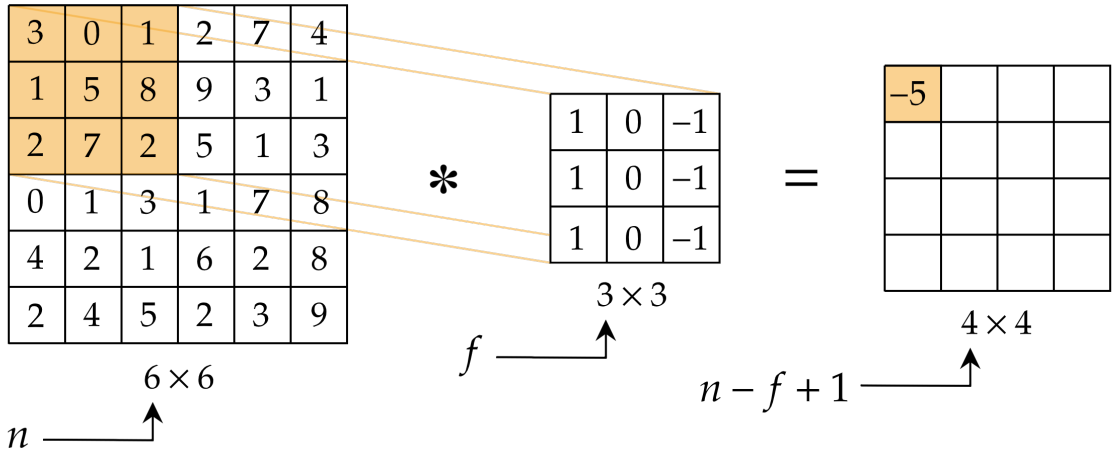
"Strong" Edge Detect

Pesos

- A seleção manual dos filtros é difícil: uso de heurísticas;
- Ideia: tratar os números dos filtros como parâmetros a serem aprendidos.



Padding

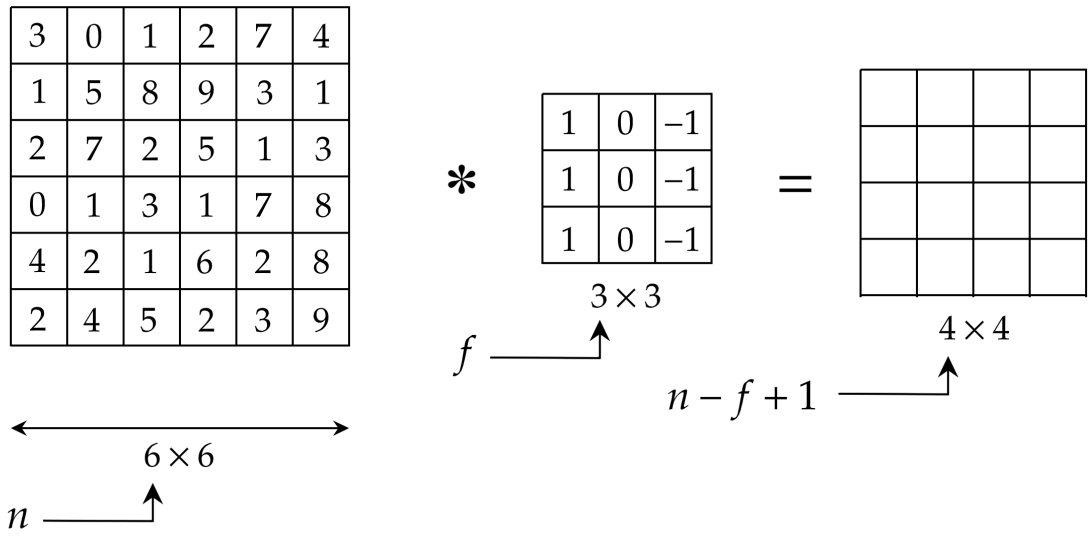


- Uma matriz de entrada $(n \times n)$ convolvida com um filtro $f \times f$ resulta em:

$$\text{matriz:} \quad (n - f + 1) \times (n - f + 1)$$

- Observe: os pixels da borda são menos usados do que outros pixels da imagem;
- Consequências:
  - Redução da matriz de saída;
  - As informações da borda são jogadas fora.

Padding

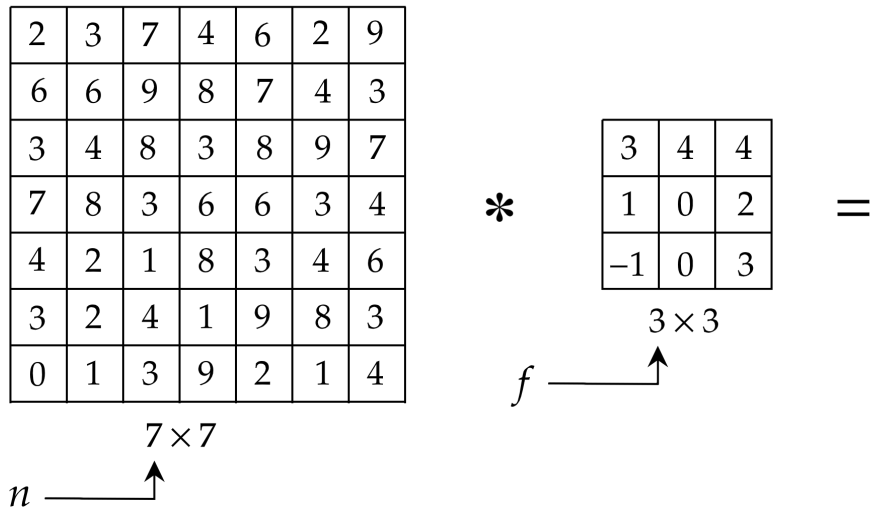


- Solução: preencher a imagem de entrada (antes da convolução), adicionado algumas linhas e colunas a ela;
- Em quase todos os casos, os valores de preenchimento são zeros;
- Para um preenchimento p em uma matriz $n \times n$ convolvida com filtro $f \times f$, a ordem da matriz de saída será:

$$(n + 2p - f + 1) \times (n + 2p - f + 1)$$

- O que ocorre se $p = (f - 1)/2$?

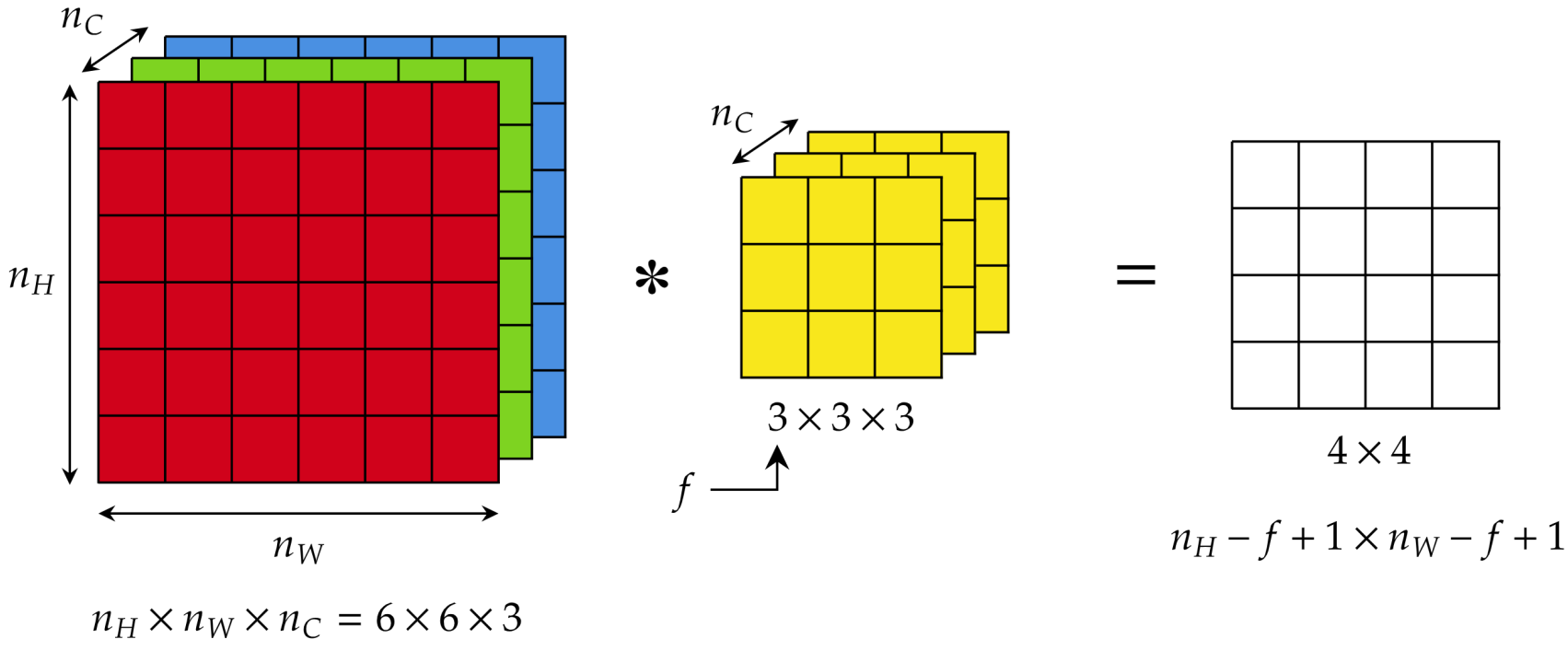
Stride



- *Stride* (s) consiste no número de pixels que saltamos durante a convolução;
- Para $s > 1$, há uma redução da ordem da matriz de saída;
- A ordem da matriz de saída será dada por:

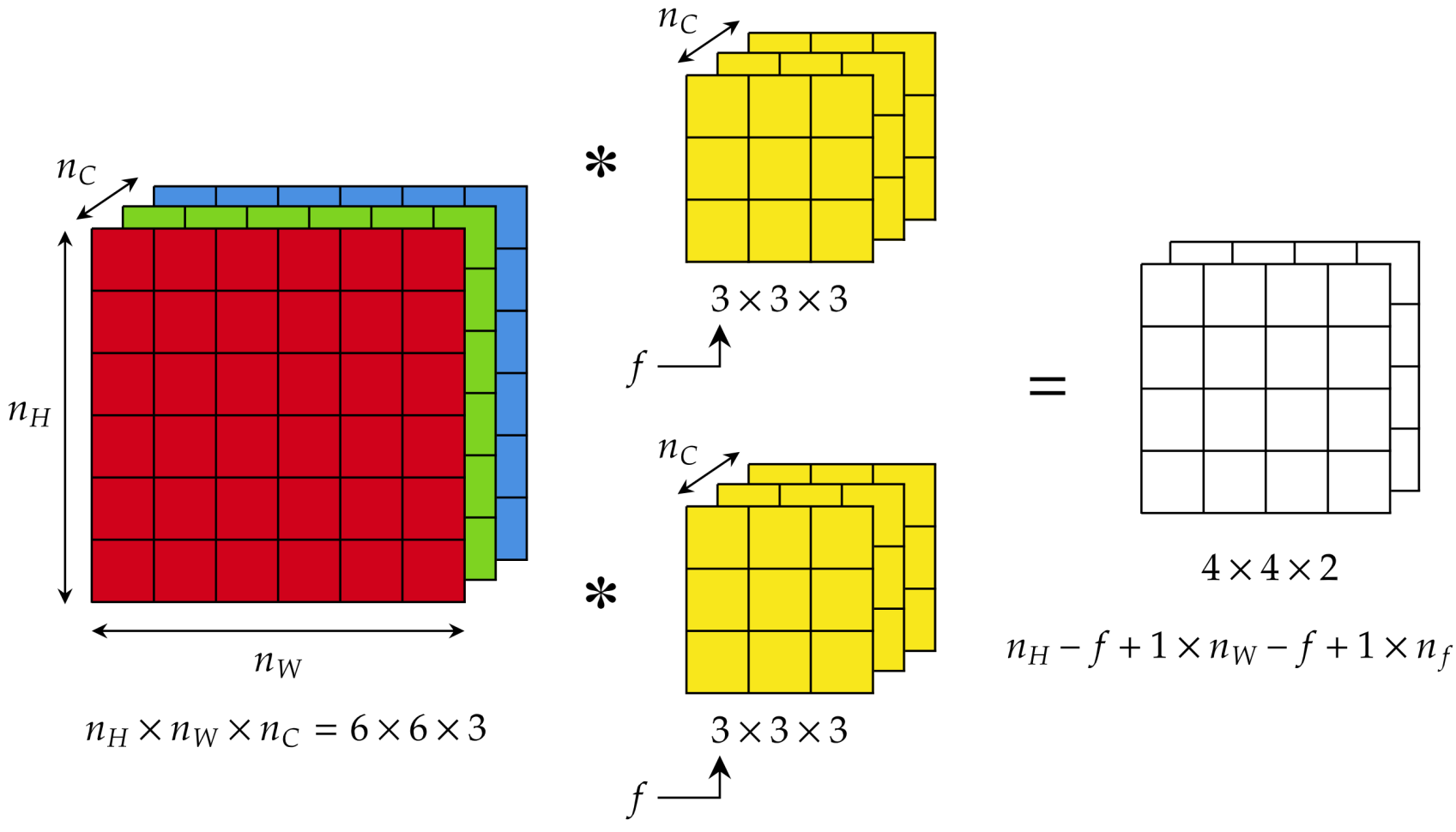
$$\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right)$$

Convolução sobre volumes



- Imagem 3D (RGB): n_W (largura), n_H (altura) e n_C (canais);
- Observe: o número de canais do filtro deve ser igual ao número de canais da imagem;
- A saída é apenas 2D.

Convolução sobre volumes: múltiplos filtros



- Múltiplos filtros para detectar várias *features*.

Camada convolucional: matematicamente

Para a $l-$ésima camada:

- Input: $a^{[l-1]}$ com dimensão (n_H^{l-1}, n_W^{l-1}, n_C^{l-1}), sendo $a^{[0]}$ a entrada;
- Padding ($p^{[l]}$);
- stride ($s^{[l]}$);
- Número de filtros: n_C^l onde cada filtro K tem as dimensões: (f^l, f^l, n_c^{l-1});
- Bias: b_n^l;
- Função de ativação: g^l;
- Output: $a^{[l]}$ com dimensões (n_H^l, n_W^l, n_C^l).

Camada convolucional: matematicamente

$\forall n \in \left[1, 2, \cdots, n_C^l\right]$:

$$\begin{aligned}\text{conv}\left(a^{[l-1]}, K^{(n)}\right)_{x,y} &= \sum_{i=1}^{n_H^{[l-1]}} \sum_{j=1}^{n_W^{[l-1]}} \sum_{k=1}^{n_C^{[l-1]}} K_{i,j,k}^{(n)} a_{x+i-1,y+j-1,k}^{[l-1]} + b_n^{[l]} \\ &= Z^{[l]}\end{aligned}$$

com

$$\text{dim}\left(\text{conv}\left(a^{[l-1]}, K^{(n)}\right)\right) = \left(n_H^{[l]}, n_W^{[l]}\right)$$

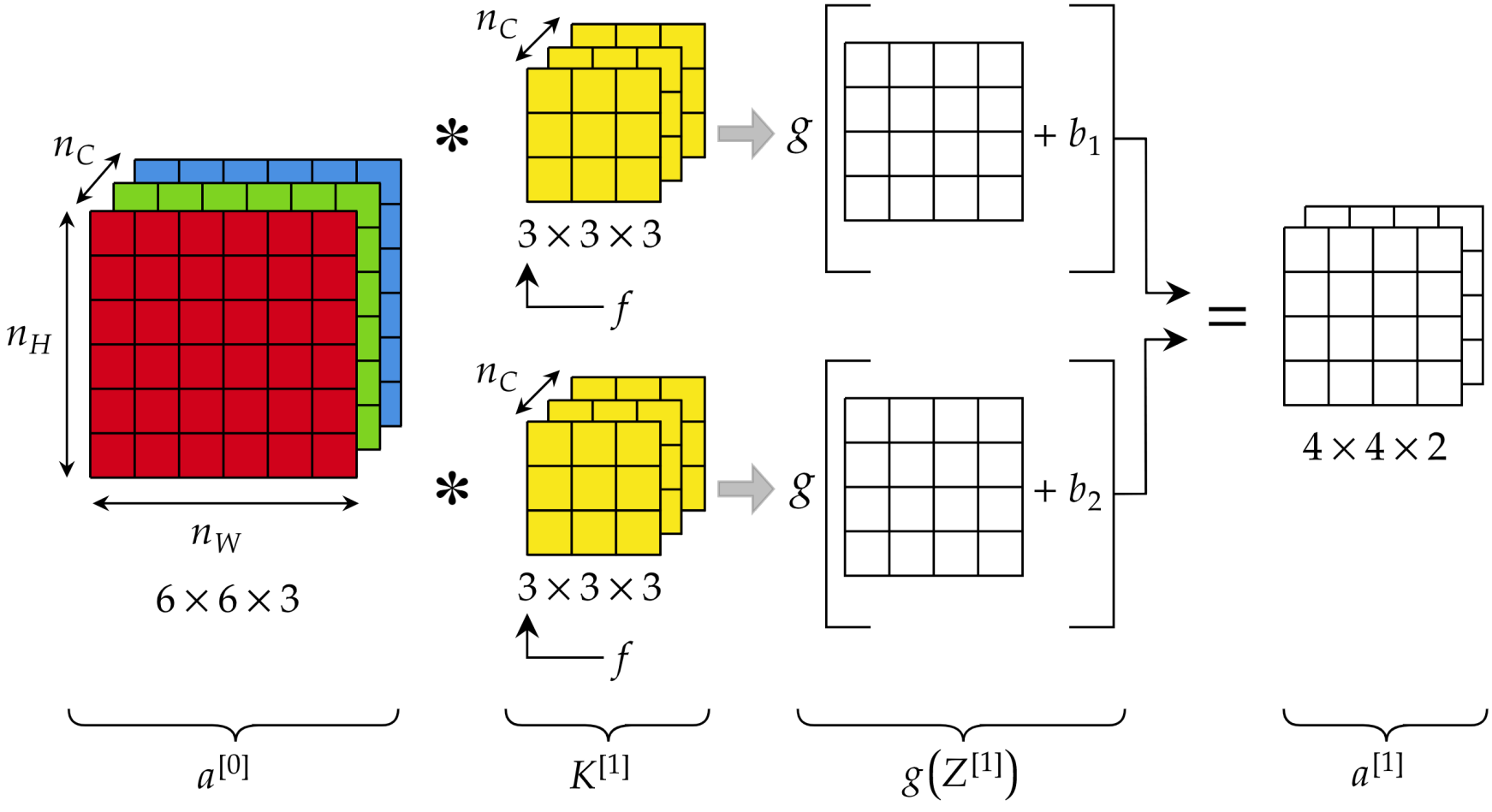
Então:

$$a^{[l]} = \left\{g^{[l]}\left[\text{conv}\left(a^{[l-1]}, K^{(1)}\right)\right], \cdots, g^{[l]}\left[\text{conv}\left(a^{[l-1]}, K^{(n_C^{[l]})}\right)\right]\right\}$$

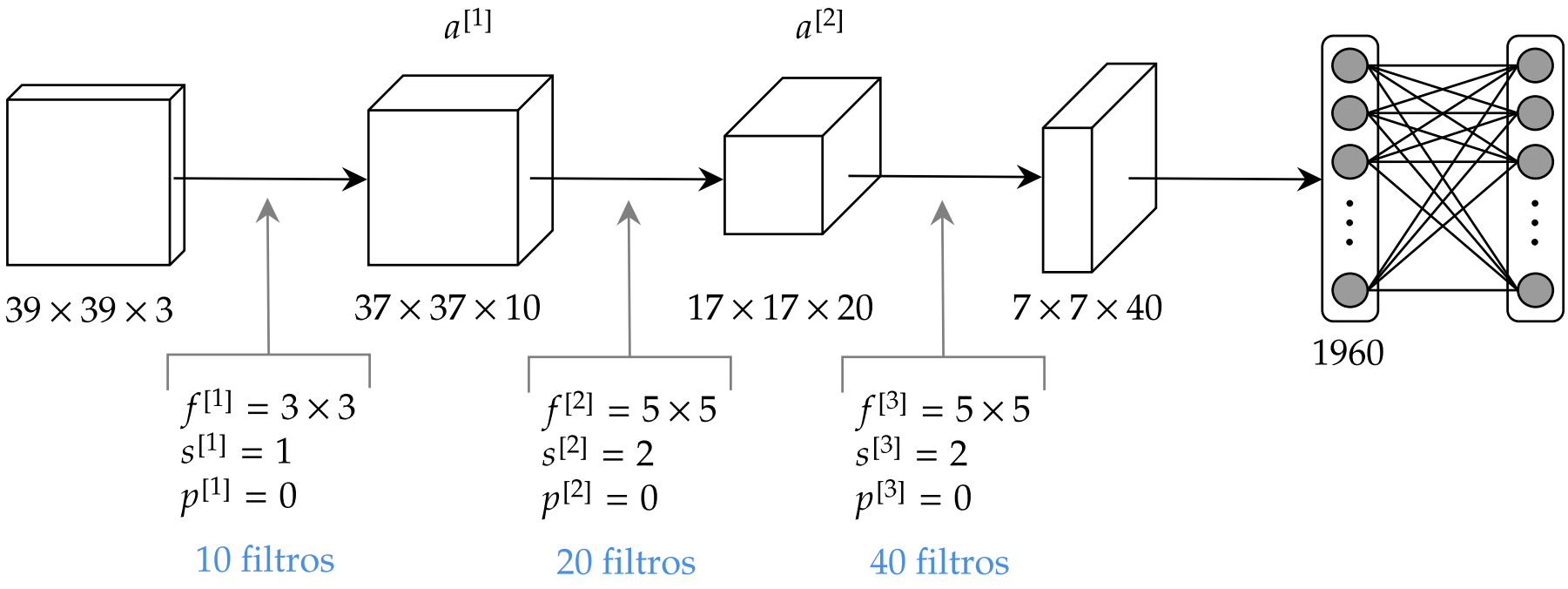
com

$$\text{dim}\left(a^{[l]}\right) = \left(n_H^{[l]}, n_W^{[l]}, n_C^{[l]}\right)$$

Camada convolucional



Exemplo: Camadas convolucionais + FC



Camadas Pooling

- Reduzem do tamanho das entradas acelerando os cálculos;
- Tornam mais robustos algumas *features* detectadas;
- A operação é realizada através de cada canal, afetando apenas as dimensões (n_H, n_W);
- Não há parâmetros para aprender.

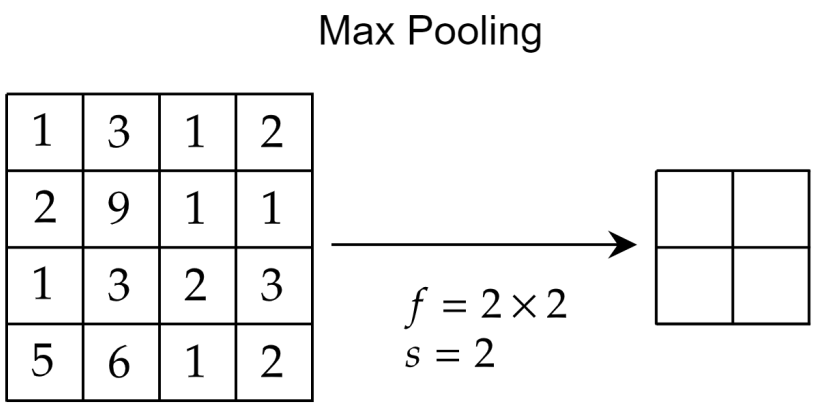
$$\text{dim}\,(\text{Pooling}) = \left(\left[\frac{n_H + 2p - f}{s} + 1 \right], \left[\frac{n_W + 2p - f}{s} + 1 \right], n_C \right), \text{ p/ } s > 0$$

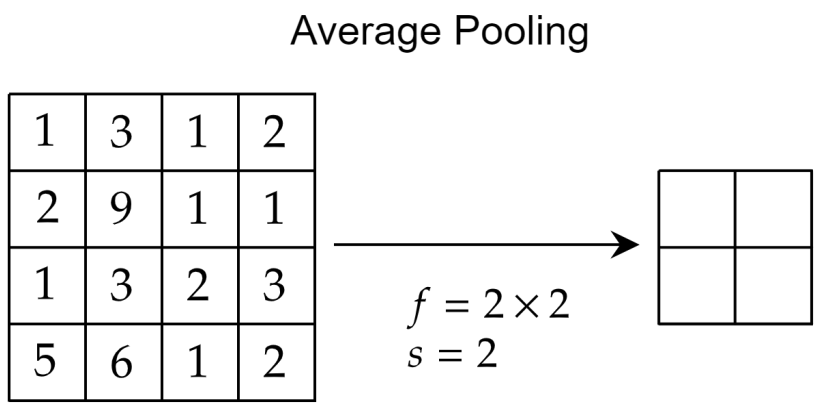
$$\text{dim}\,(\text{Pooling}) = (n_H + 2p - f, n_W + 2p - f, n_C), \text{ p/ } s = 0$$

Camadas Pooling

Tipos mais empregados:

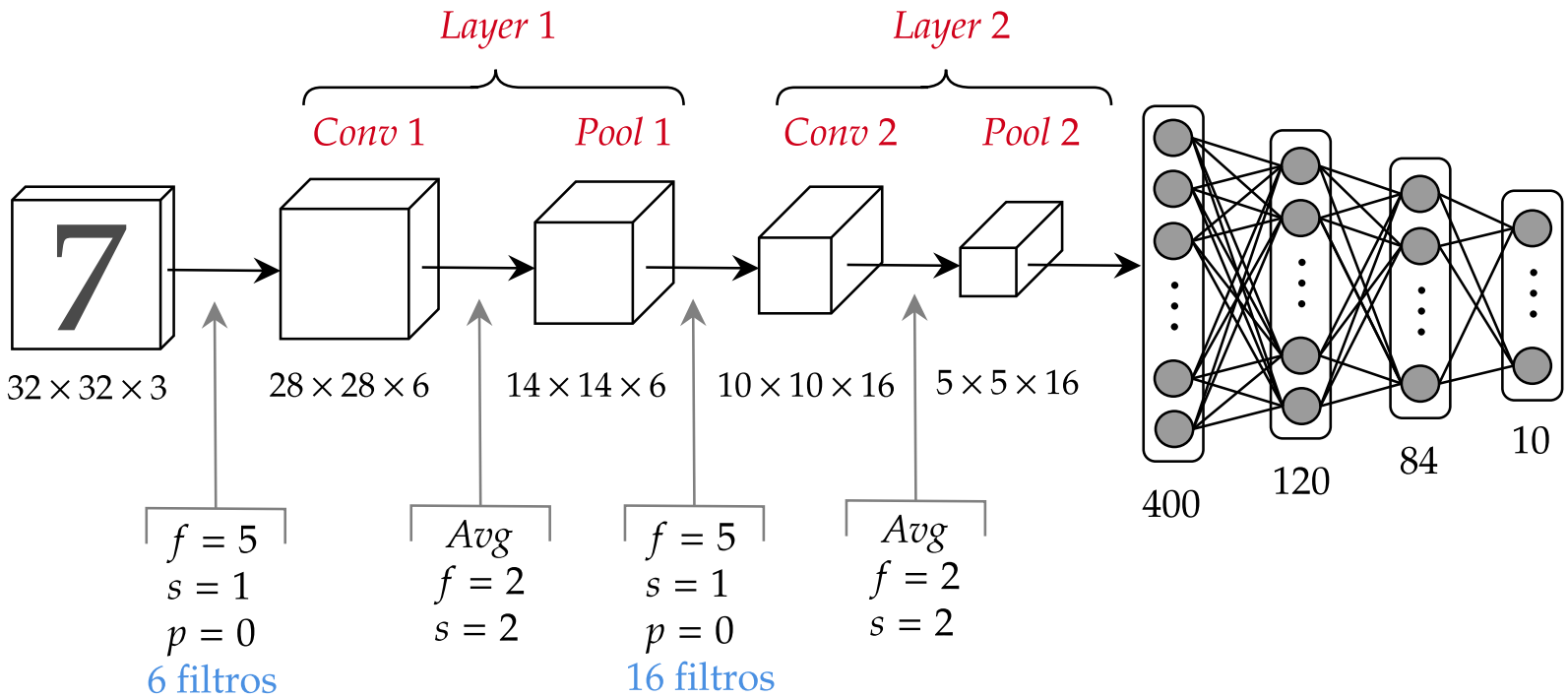
- Average pooling: média dos valores da região da imagem definida pela janela do filtro;
- Max pooling: retorna o maior valor daqueles presentes na região da imagem definida pela janela do filtro.





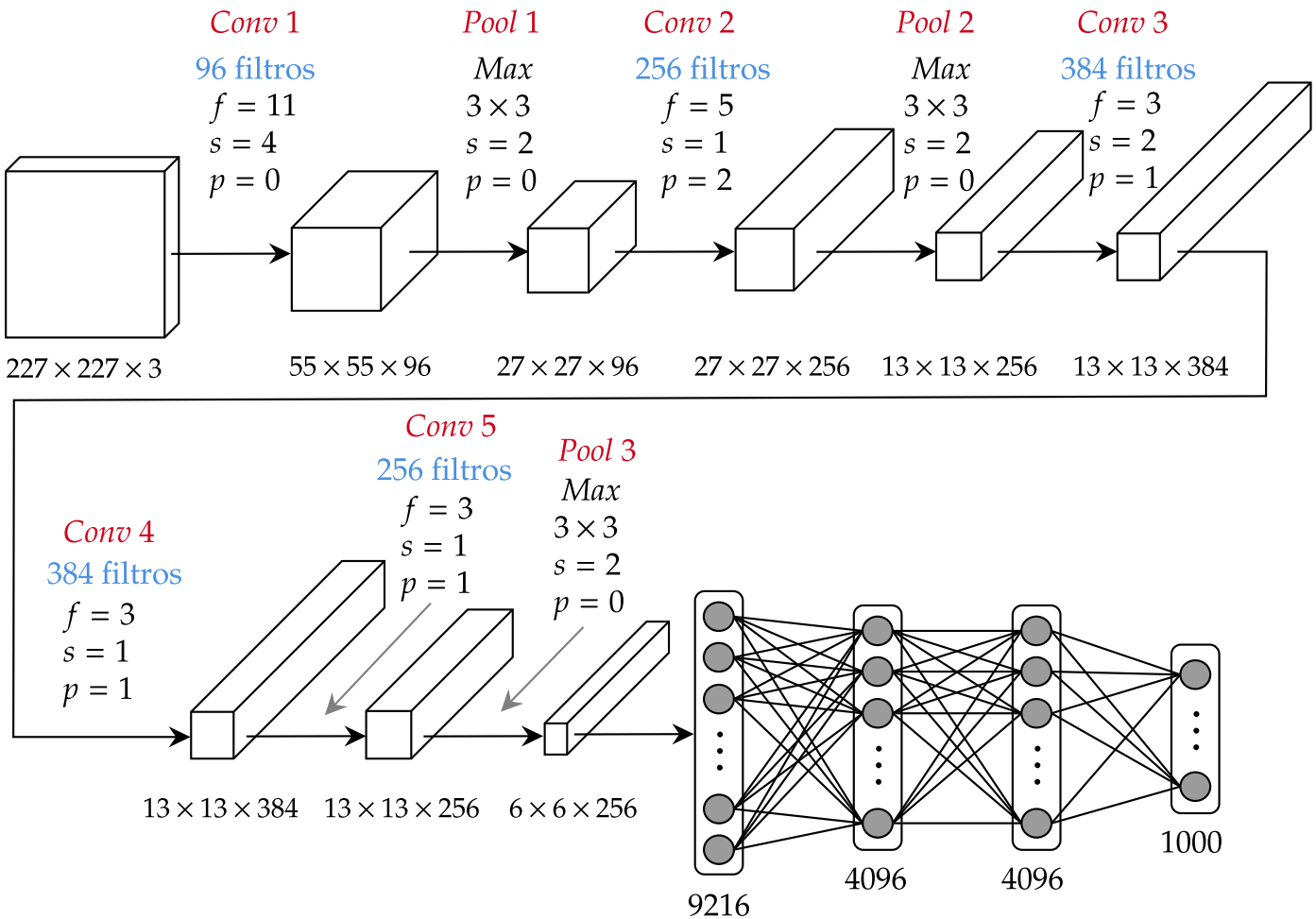
Exemplos de CNNs

LeNet-5



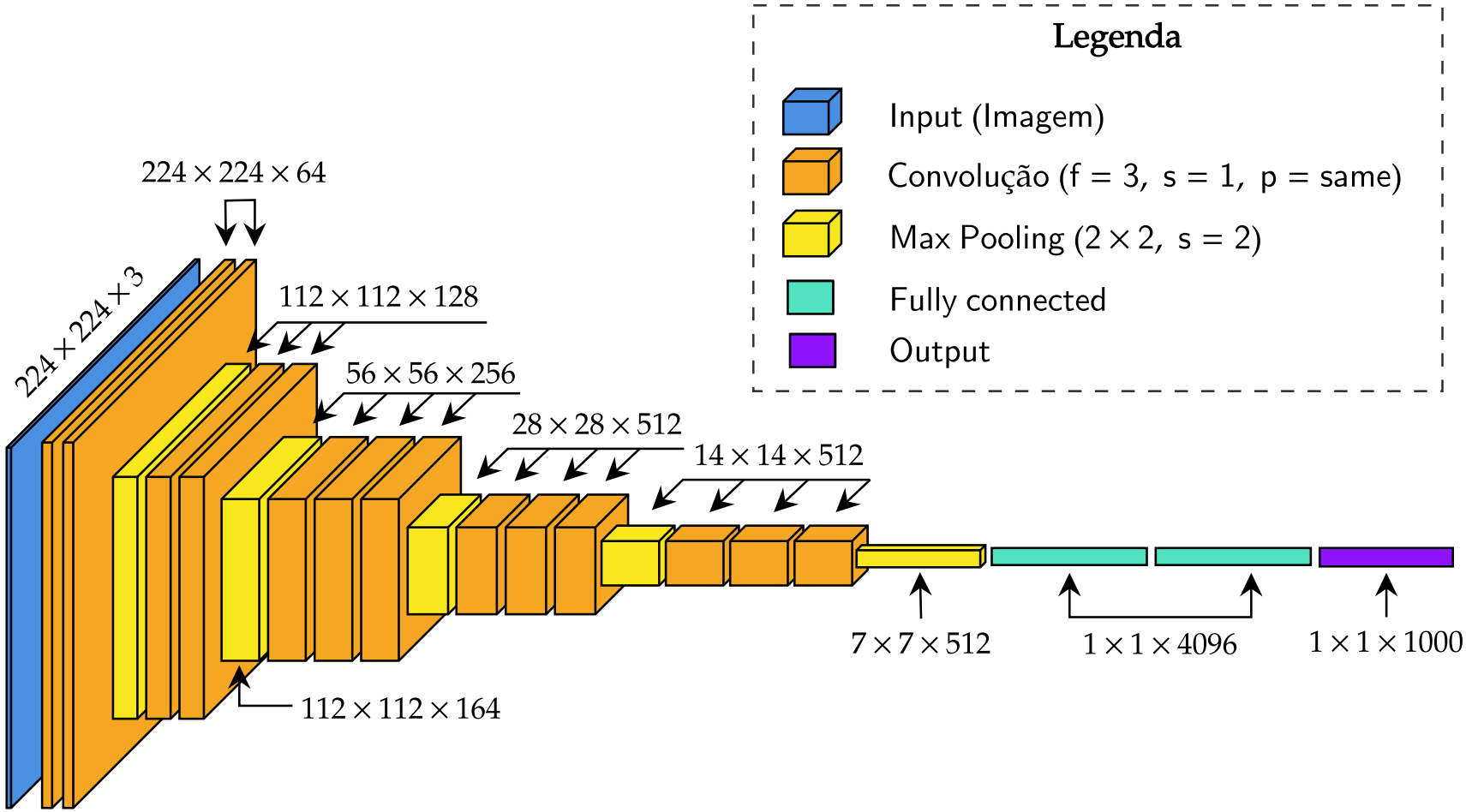
- Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, **Gradient-based learning applied to document recognition**, in **Proceedings of the IEEE**, vol. 86, no. 11, pp. 2278−2324, Nov. 1998

AlexNet



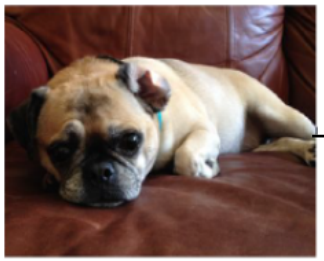
- A. Krizhevsky, I. Sutskever, and G. Hinton. **Imagenet classification with deep convolutional neural networks**. In *NIPS*, 2012.

VGG-16



- K. Simonyan and A. Zisserman. **Very deep convolutional networks for large-scale image recognition**. In *ICLR*, 2015.

VGG-16: visualização das *features*

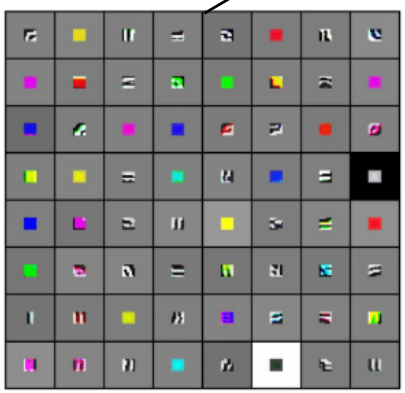


Features de nível baixo

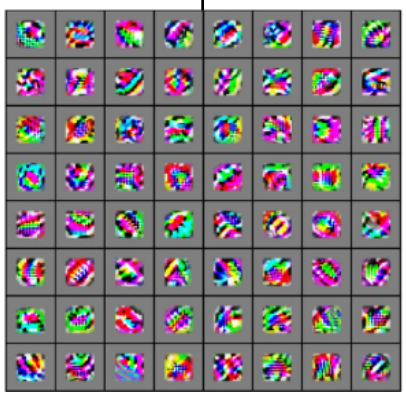
Features de nível médio

Features de nível alto

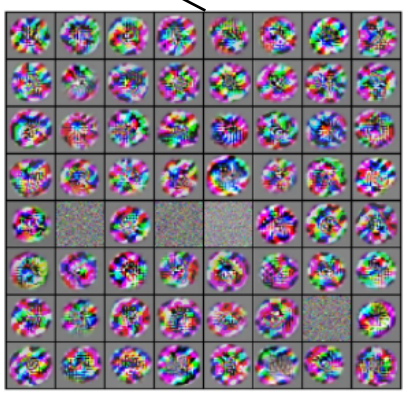
Classificador linearmente separável



VGG-16 Conv1_1

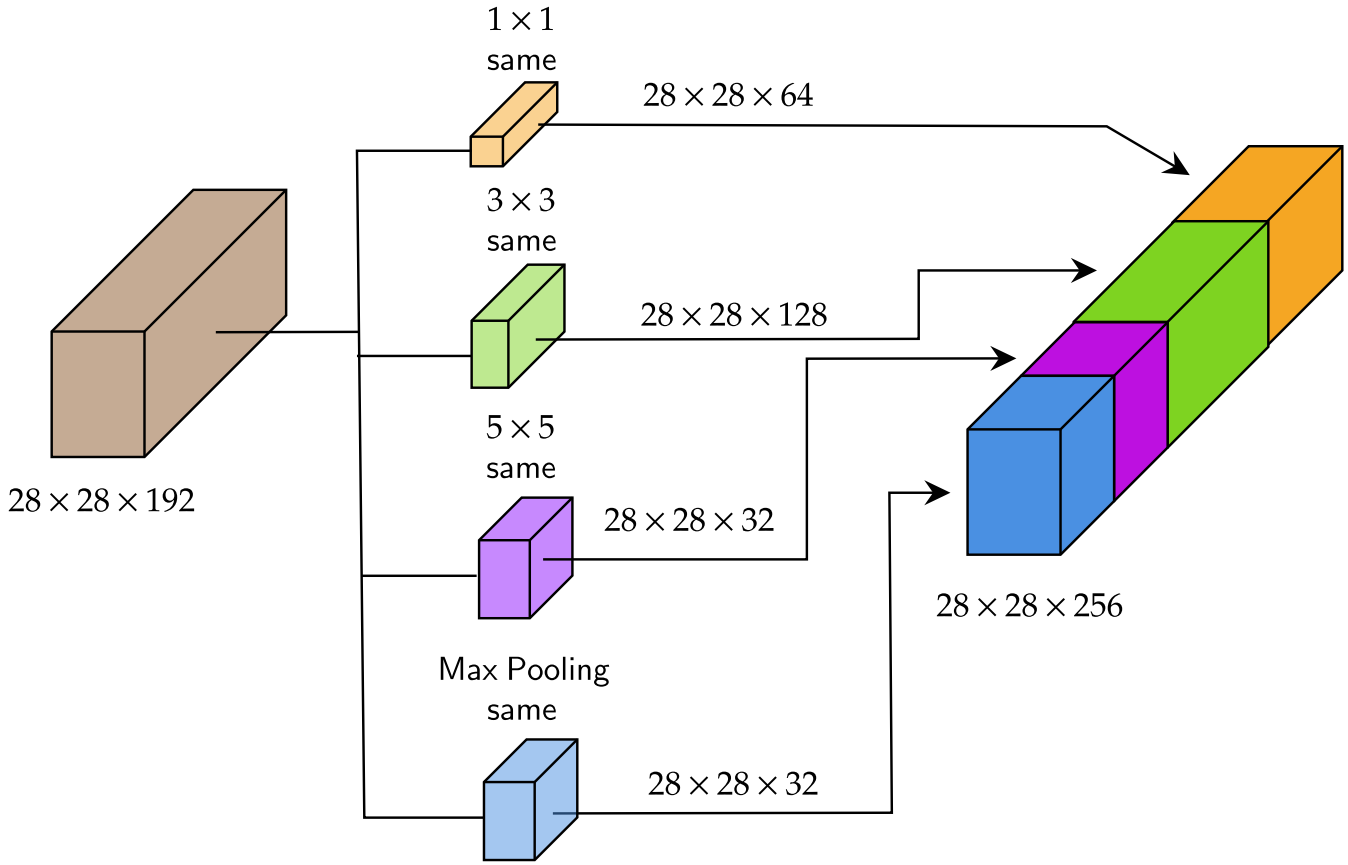


VGG-16 Conv3_2



VGG-16 Conv5_3

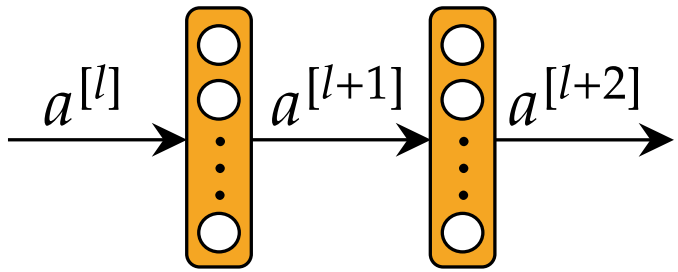
Inception Module



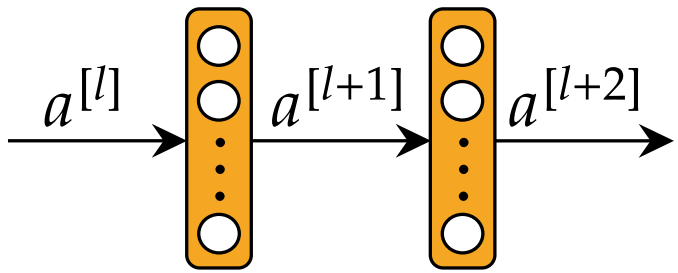
- Base da rede GoogLeNet;
- C. Szegedy, et al. **Going deeper with convolutions**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1−9, 2015

ResNet

ResNet



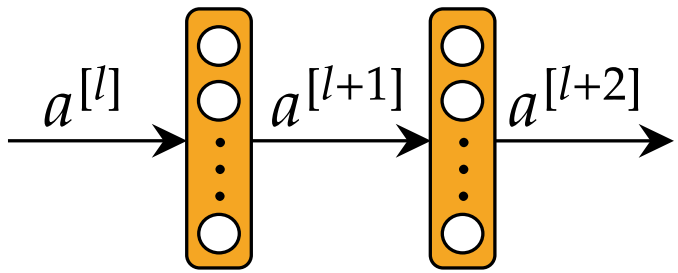
ResNet

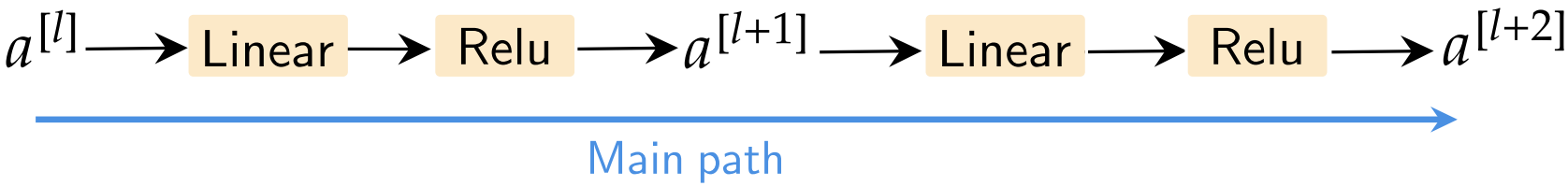


$a^{[l]}$ → Linear → Relu → $a^{[l+1]}$ → Linear → Relu → $a^{[l+2]}$

$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g\left(z^{[l+1]}\right) \qquad z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g\left(z^{[l+2]}\right)$$

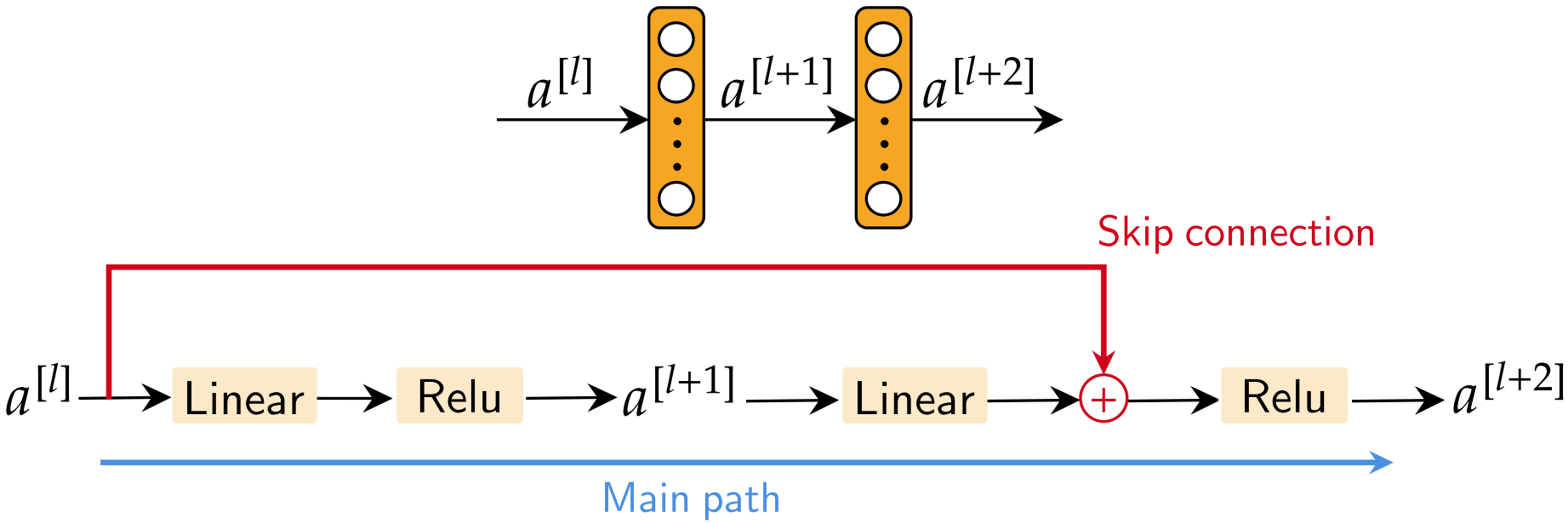
ResNet





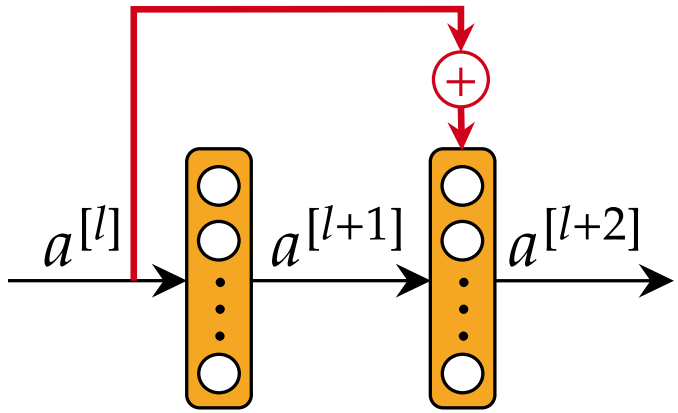
$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g\left(z^{[l+1]}\right) \qquad z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g\left(z^{[l+2]}\right)$$

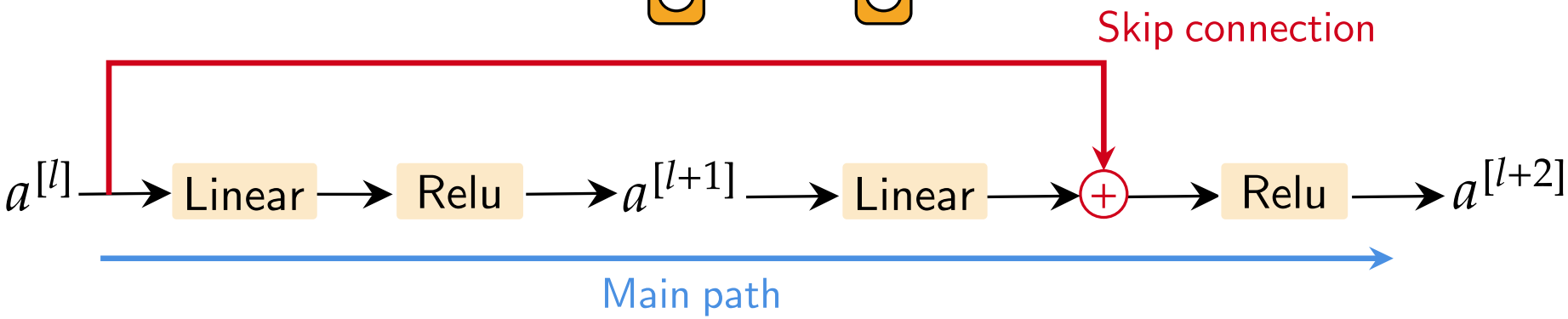
ResNet



$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g\left(z^{[l+1]}\right) \qquad z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g\left(z^{[l+2]}\right)$$

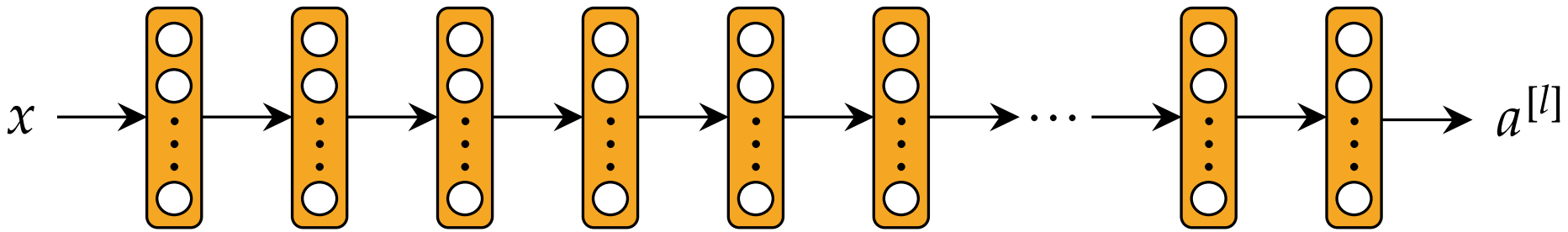
ResNet



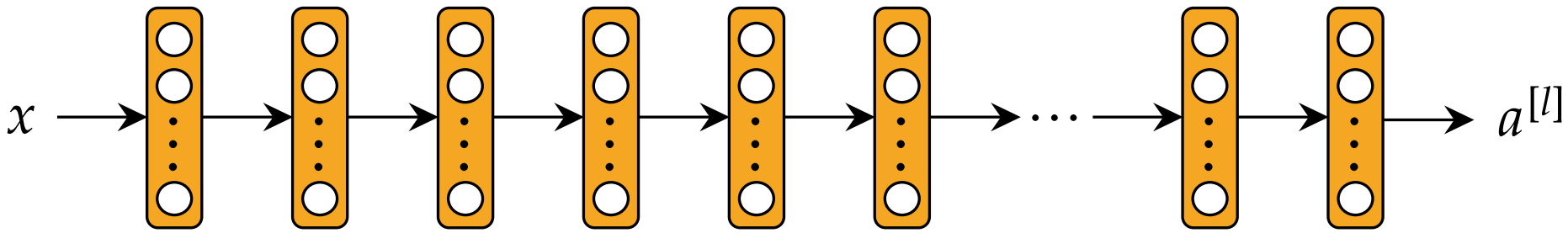


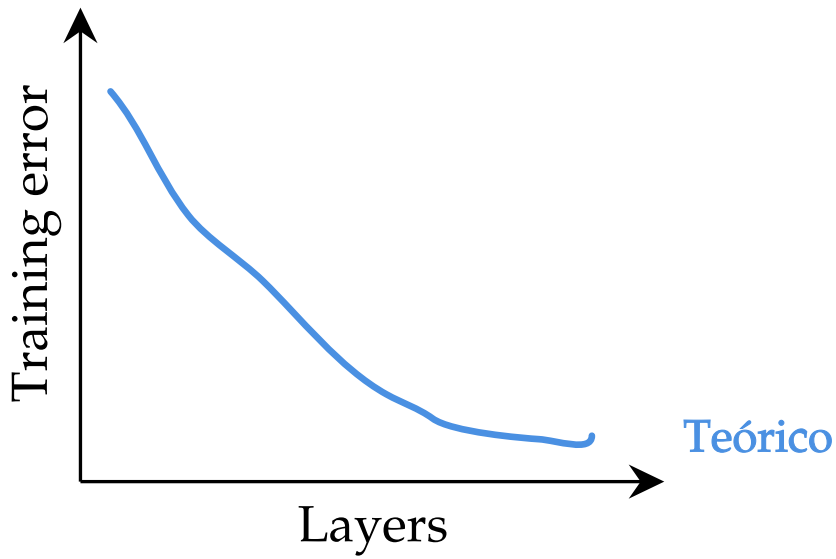
$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g\left(z^{[l+1]}\right) \qquad z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g\left(z^{[l+2]}\right)$$

ResNet

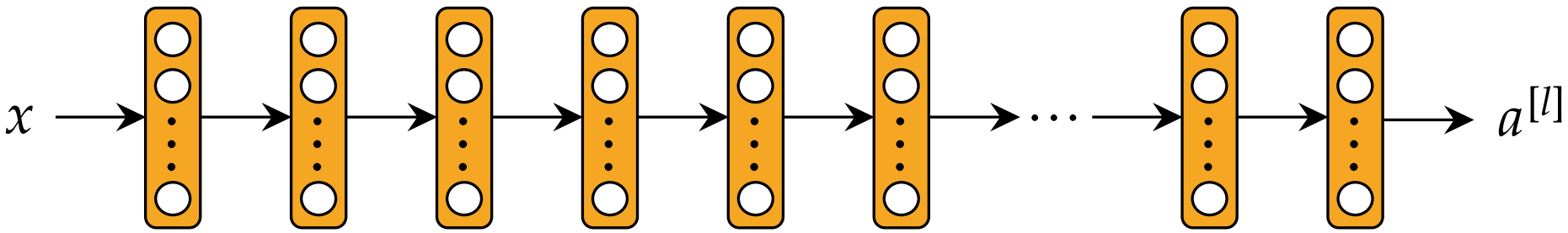


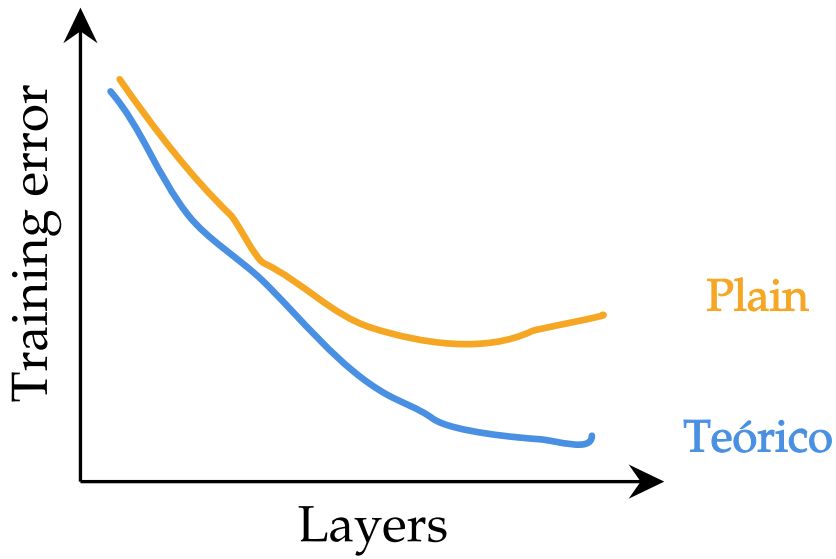
ResNet



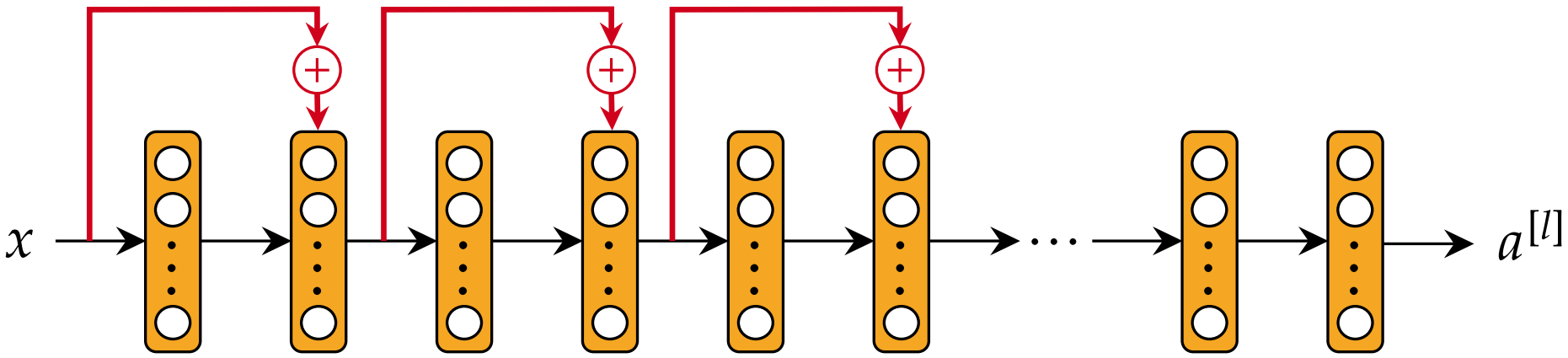


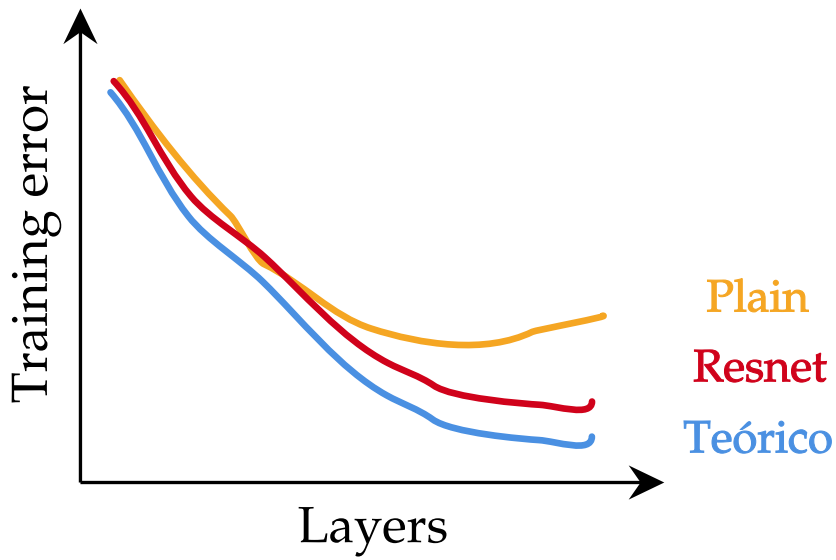
ResNet



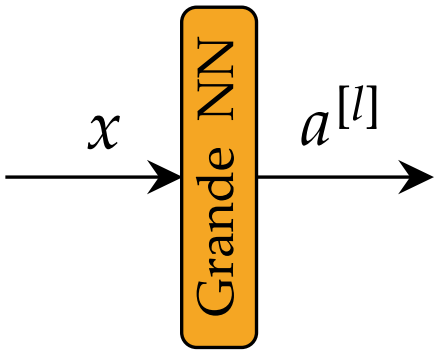


ResNet

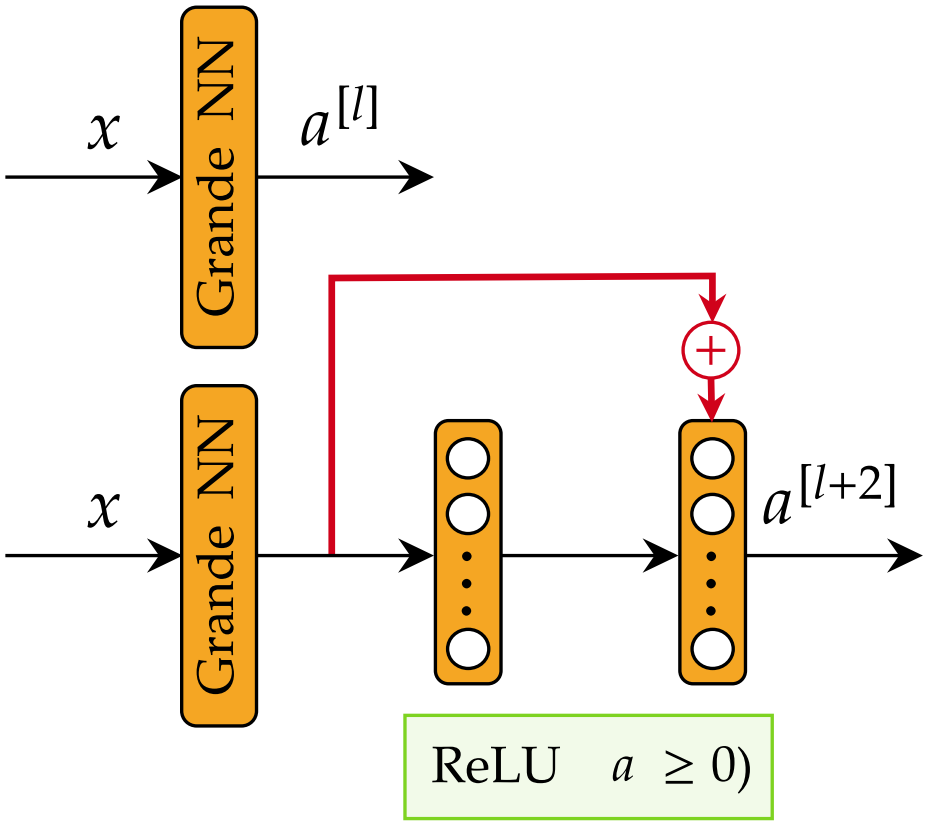




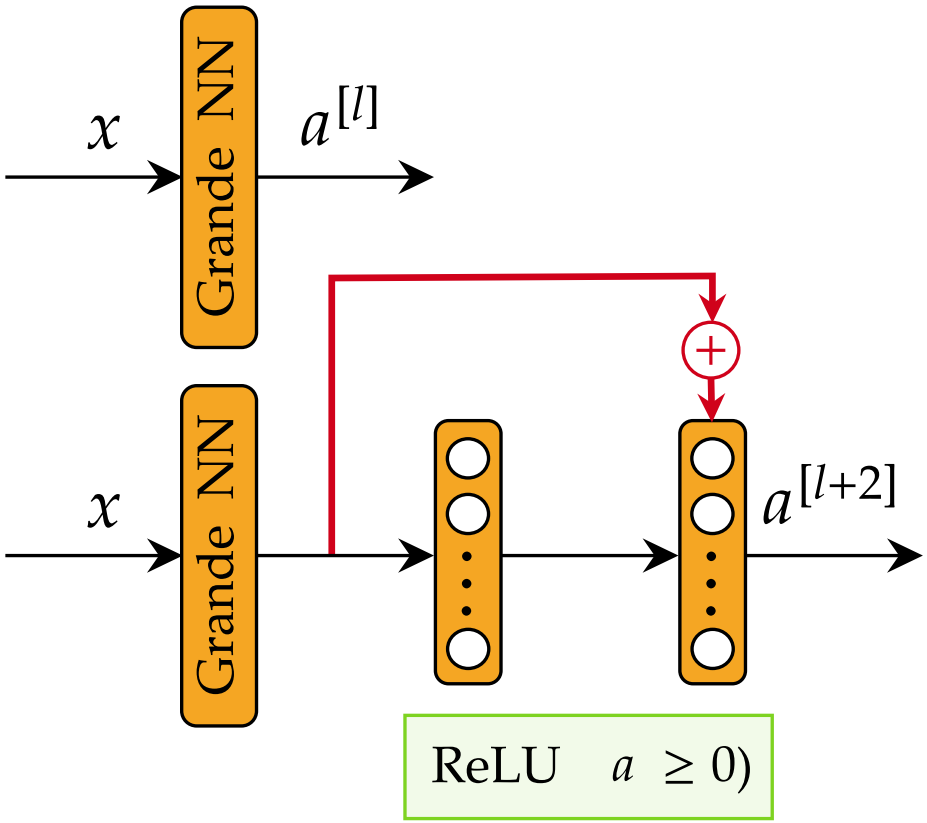
ResNet - Como ela funciona?



ResNet - Como ela funciona?

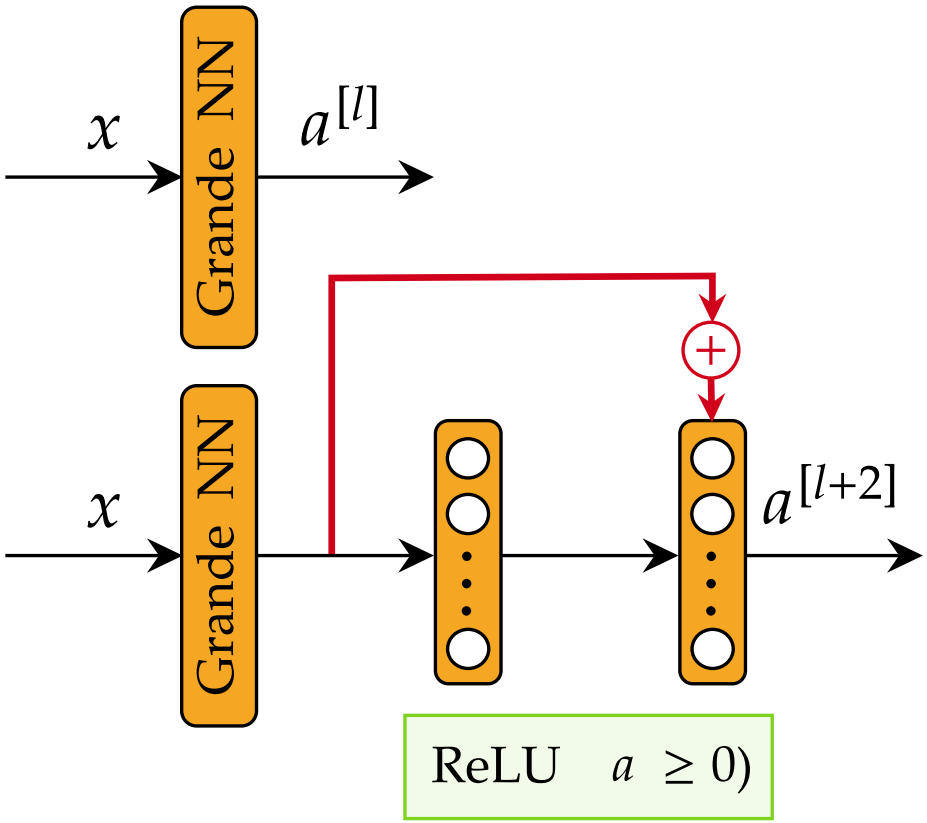


ResNet - Como ela funciona?



$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

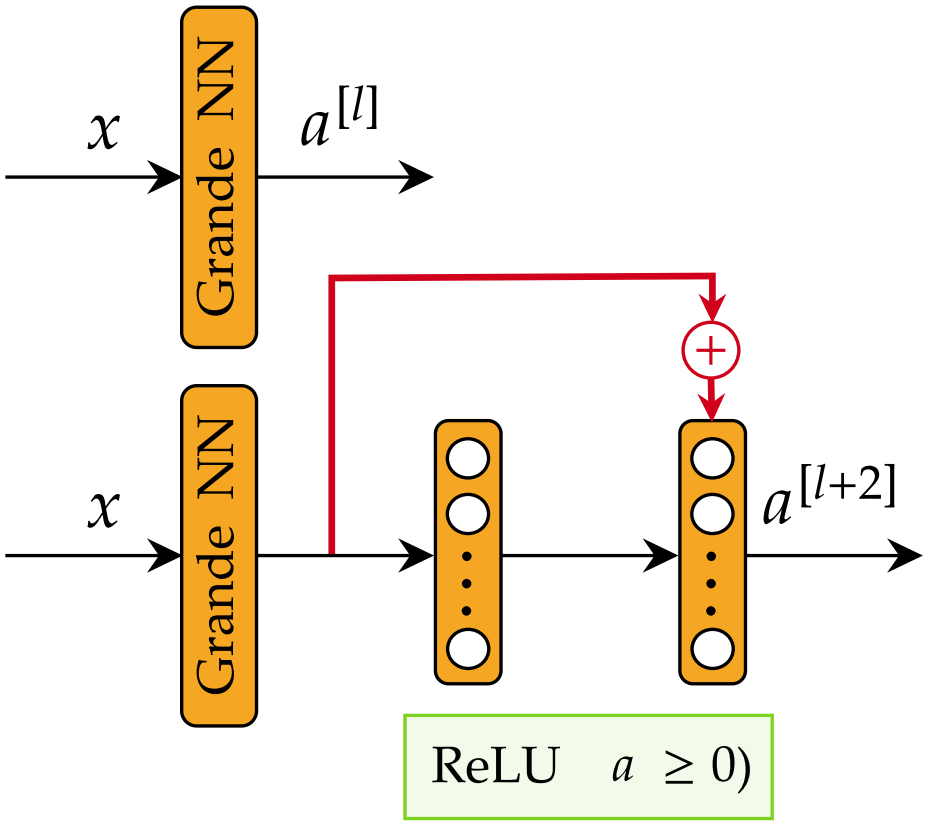
ResNet - Como ela funciona?



$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

$$a^{[l+2]} = g\left(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]}\right)$$

ResNet - Como ela funciona?

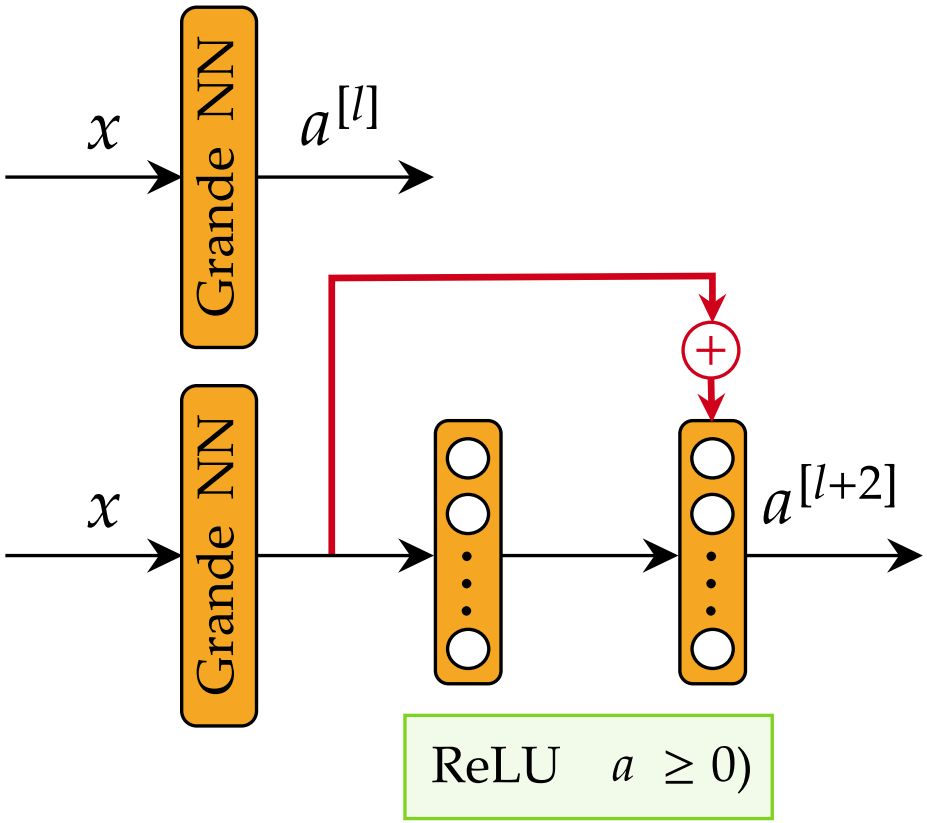


$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

$$a^{[l+2]} = g\left(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]}\right)$$

$= 0$ $= 0$

ResNet - Como ela funciona?



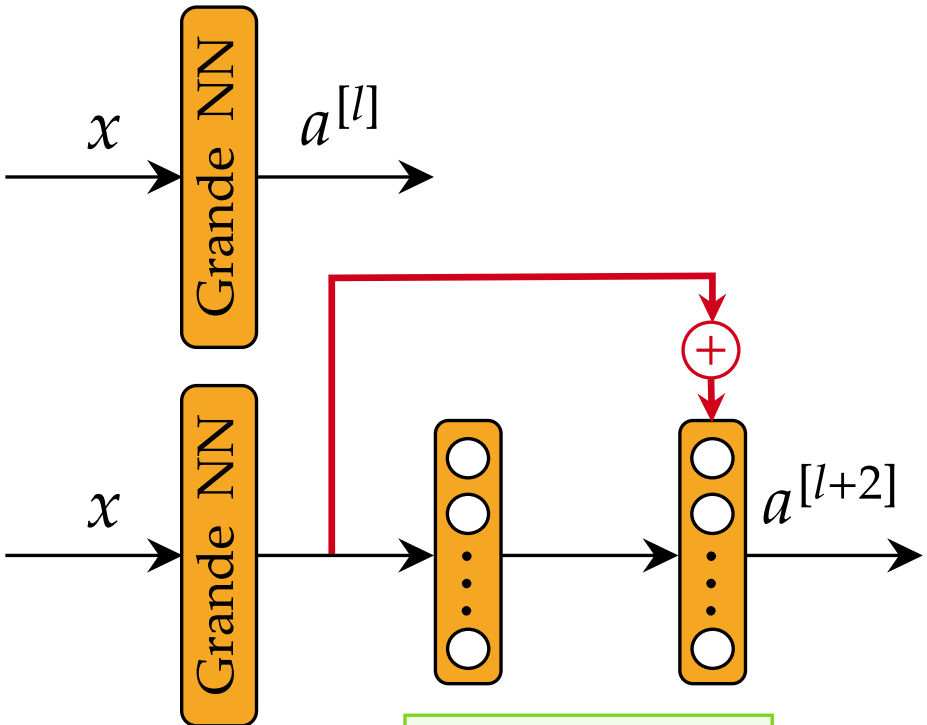
$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

$$a^{[l+2]} = g\left(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]}\right)$$

$= 0$ $= 0$

$$a^{[l+2]} = g\left(a^{[l]}\right)$$

ResNet - Como ela funciona?



$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

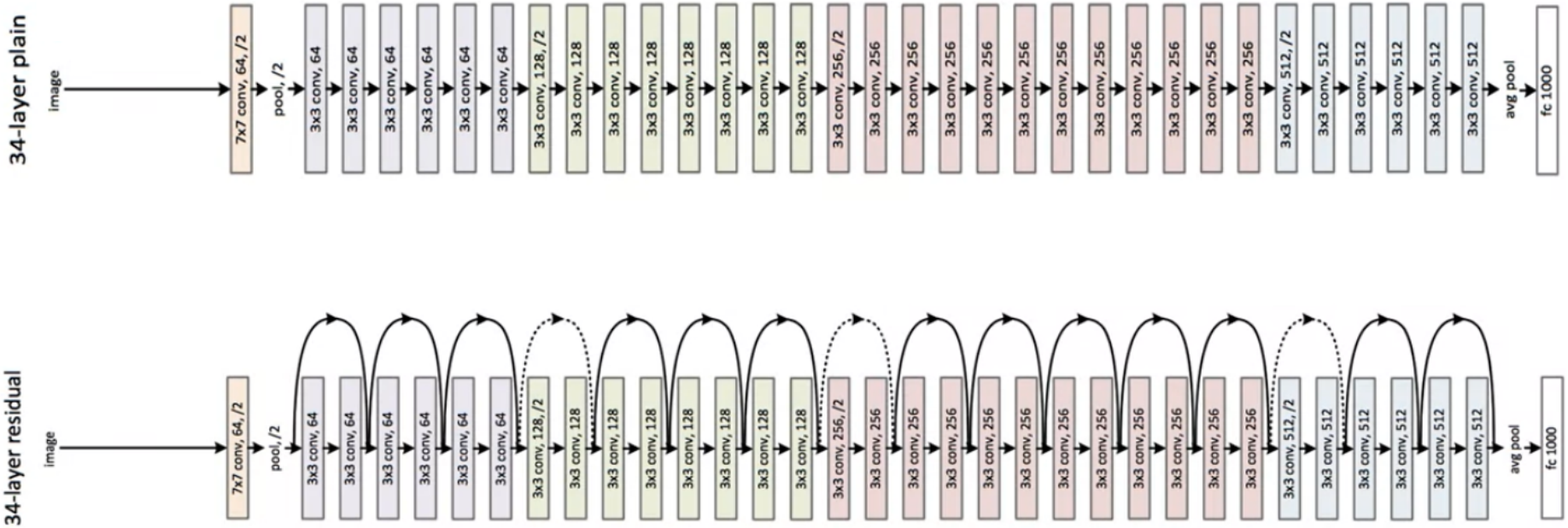
$$a^{[l+2]} = g\left(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]}\right)$$

$= 0$ $= 0$

$$a^{[l+2]} = g\left(a^{[l]}\right)$$

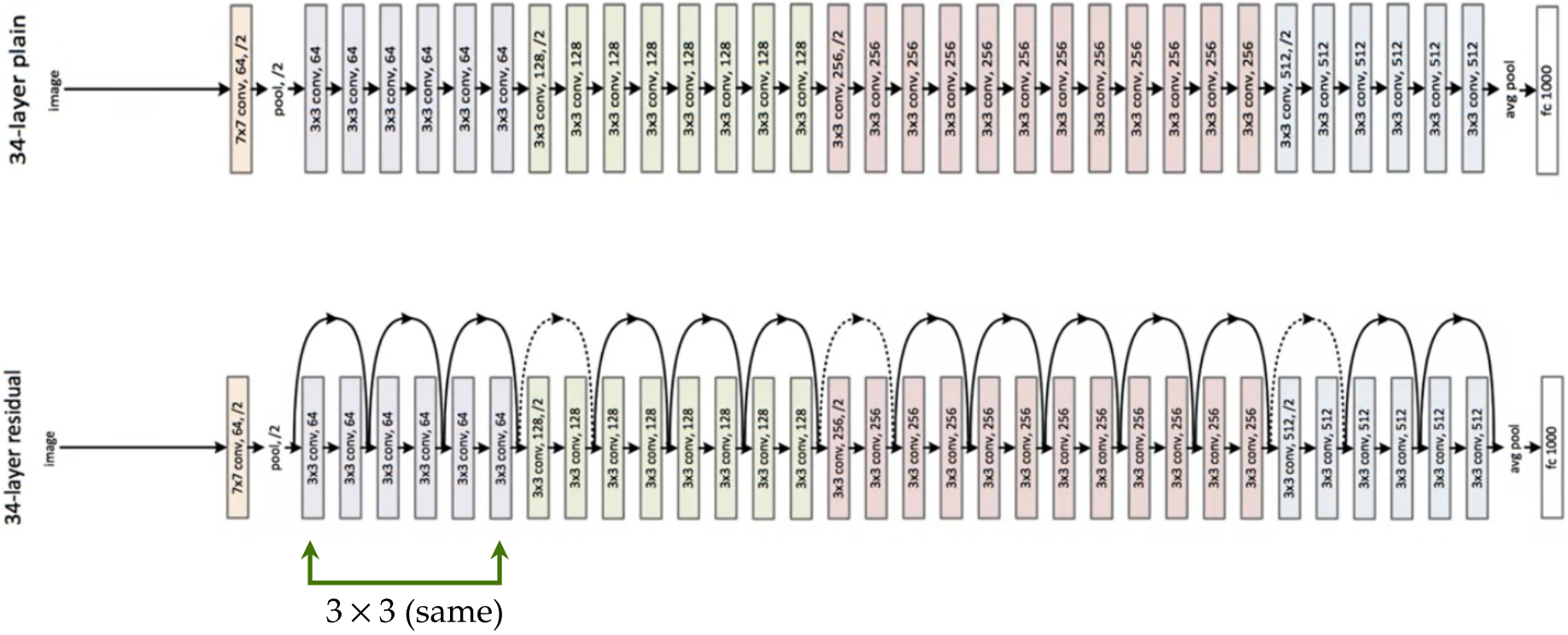
$$a^{[l+2]} = g\left(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + W_s a^{[l]}\right)$$

ResNet



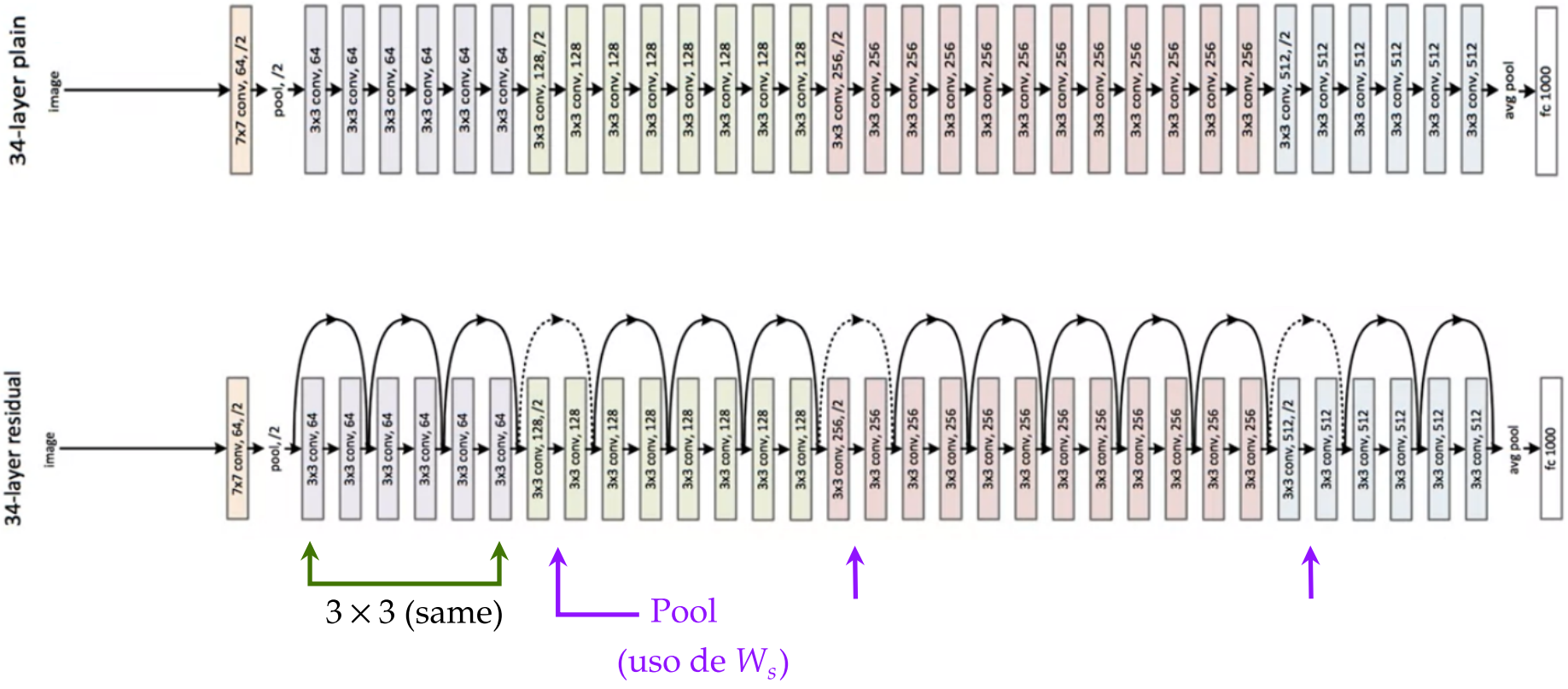
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Deep residual learning for image recognition**.

ResNet



- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Deep residual learning for image recognition**.

ResNet



- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Deep residual learning for image recognition**.

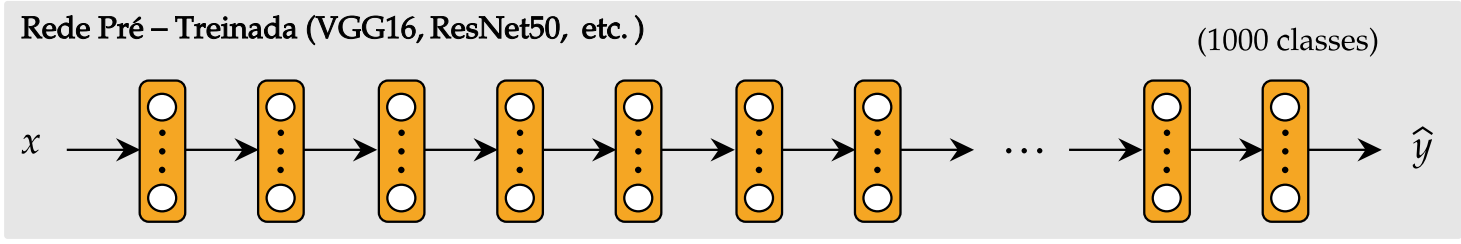
Transfer Learning

Transfer Learning

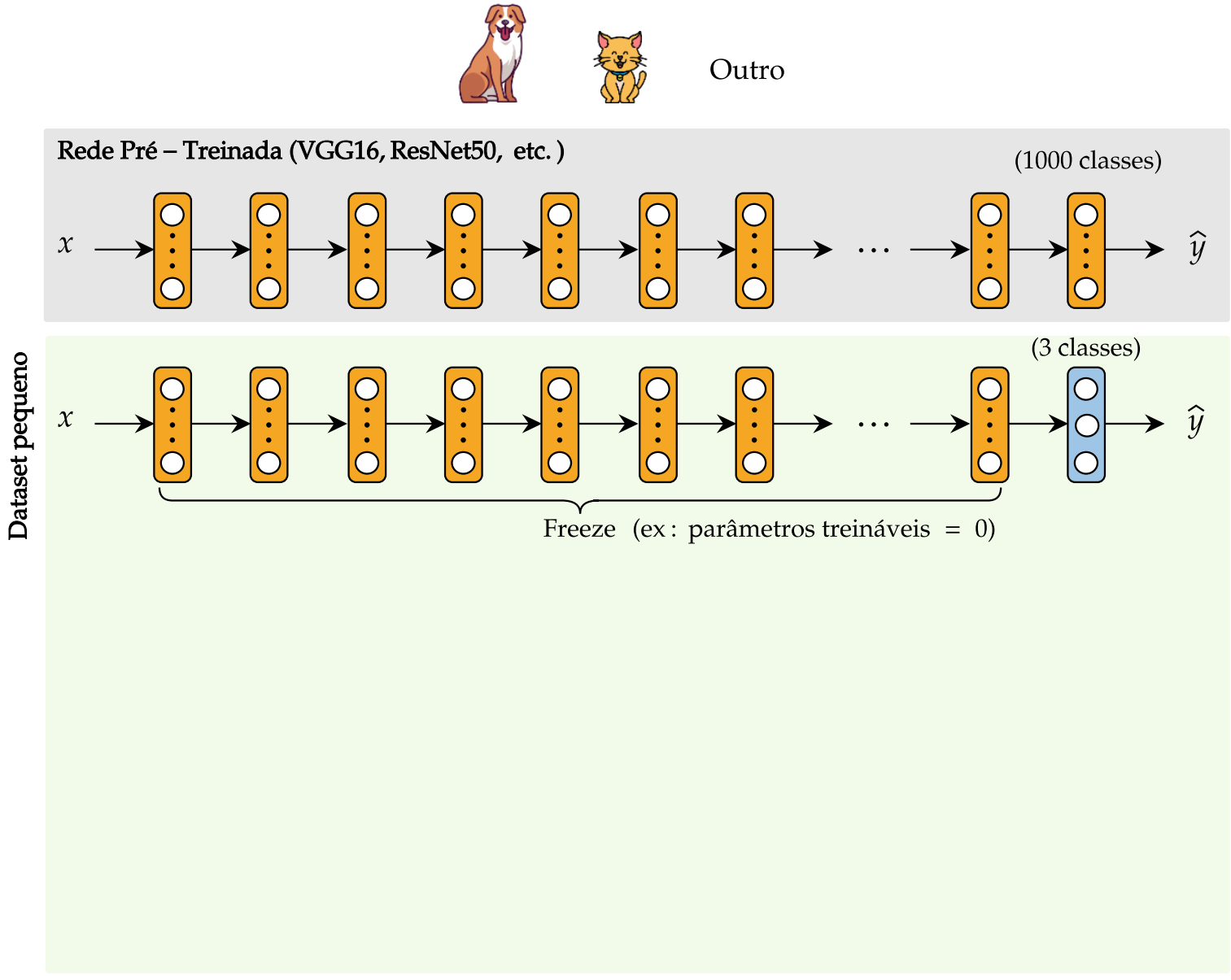
Outro

Transfer Learning

Outro

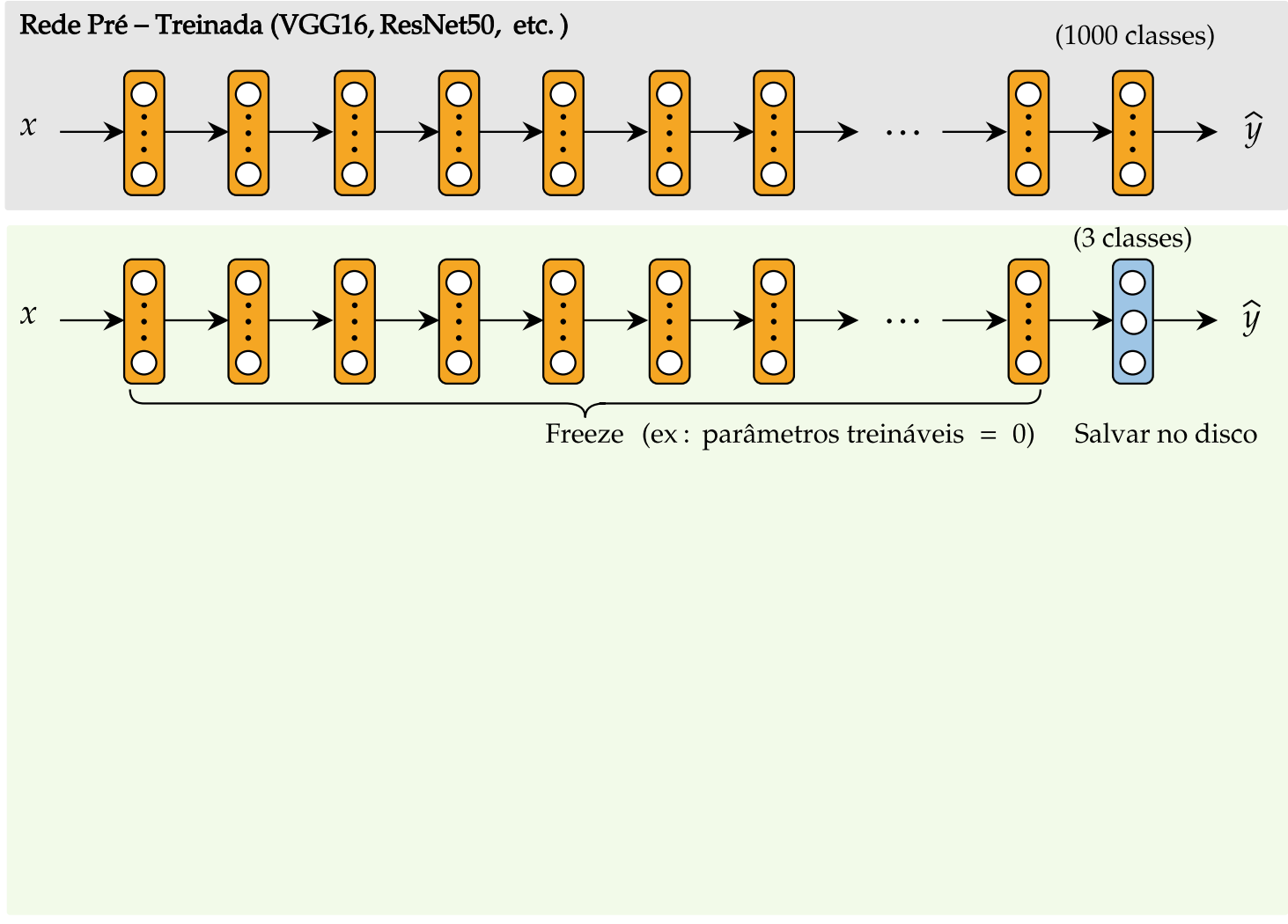


Transfer Learning

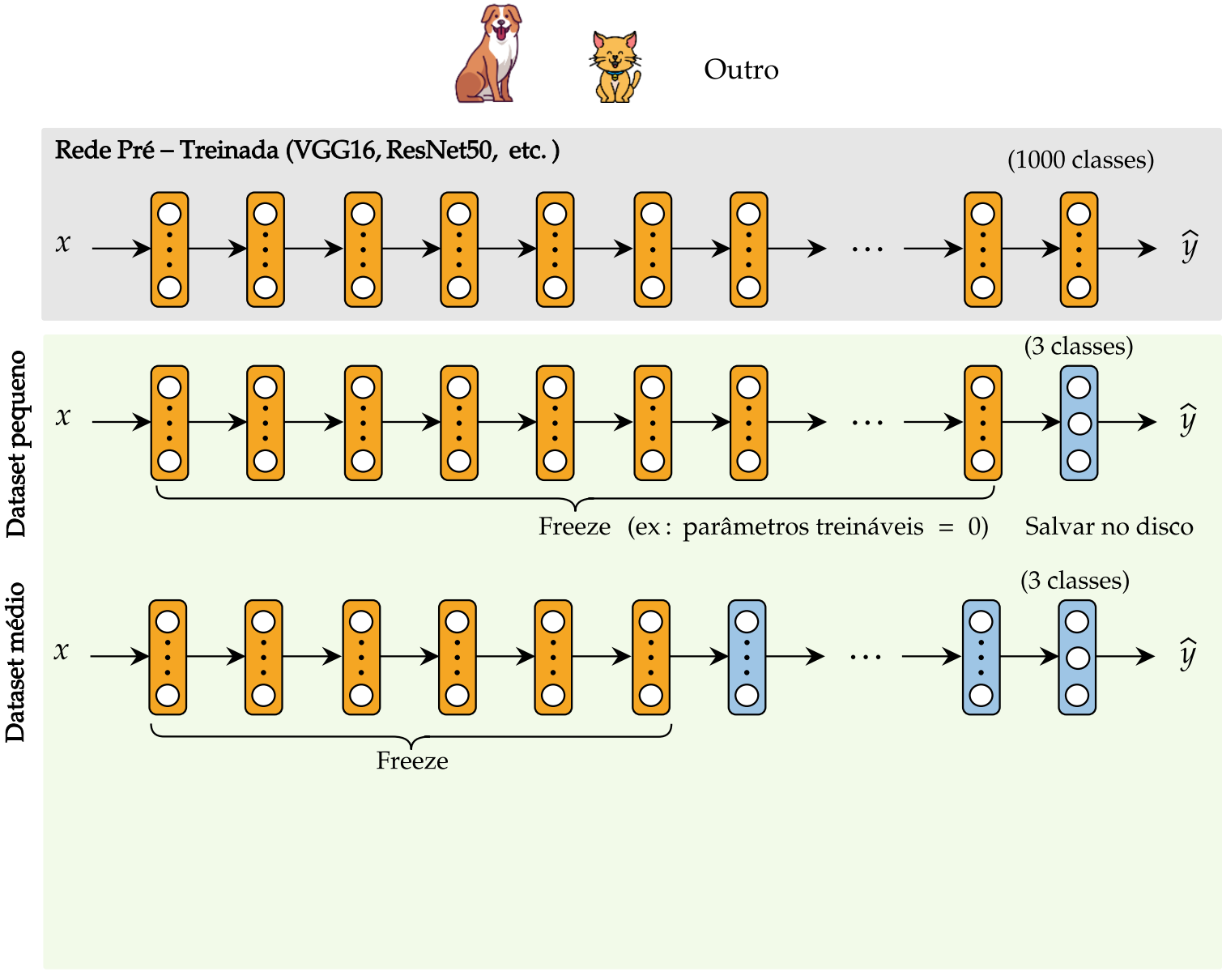


Transfer Learning

Outro



Transfer Learning



Transfer Learning