

Inteligência Artificial para Robótica Móvel

Redes Neurais Convolucionais

Professor: Marcos Maximo

Roteiro

- Motivação;
- Aplicações de CNNs;
- Convolução;
- *Pooling*;
- Estudos de Casos;
- *Inception Network*;
- ResNets;
- *Transfer Learning*.

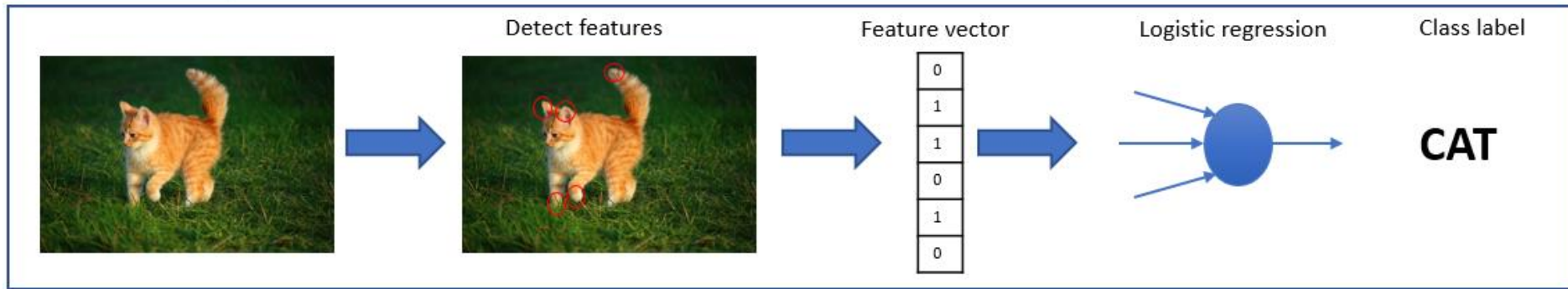
Motivação

Motivação

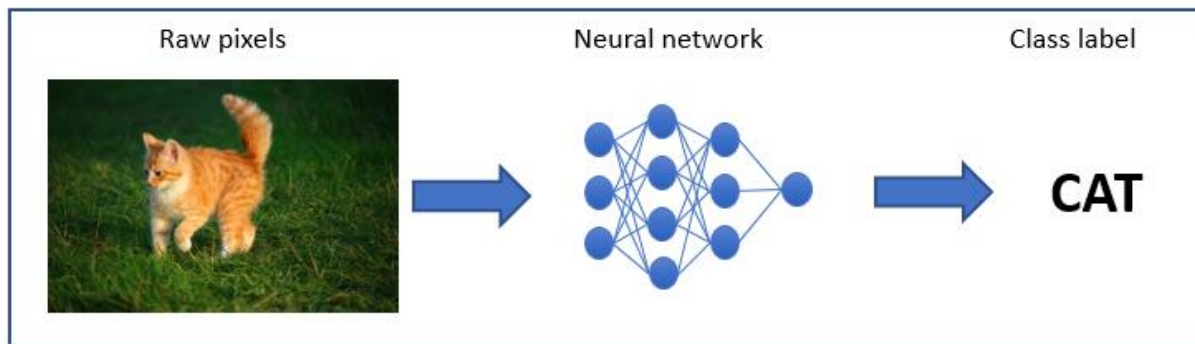
- Inglês: *Convolutional Neural Networks (CNN)*.
- Arquitetura inspirada no córtex visual do cérebro humano.
- Nome vem da operação matemática de “convolução”.
- Desempenho excepcional em tarefas de visão computacional.
- Algumas redes chegam a ter dezenas ou centenas de camadas.
- Algumas redes possuem milhões de parâmetros.

Visão Clássica x CNN

Traditional Computer Vision



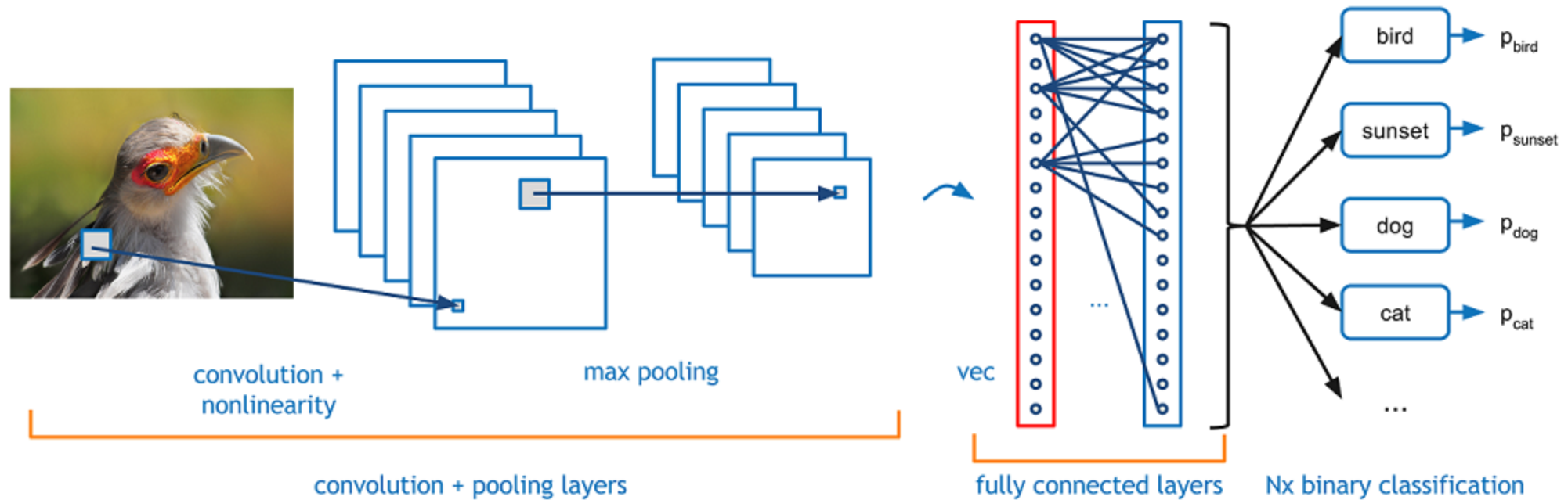
Deep learning



Fonte: <https://blog.esciencecenter.nl/mcfly-time-series-classification-made-easy-e47de8d29838>

Visão Geral de uma CNN

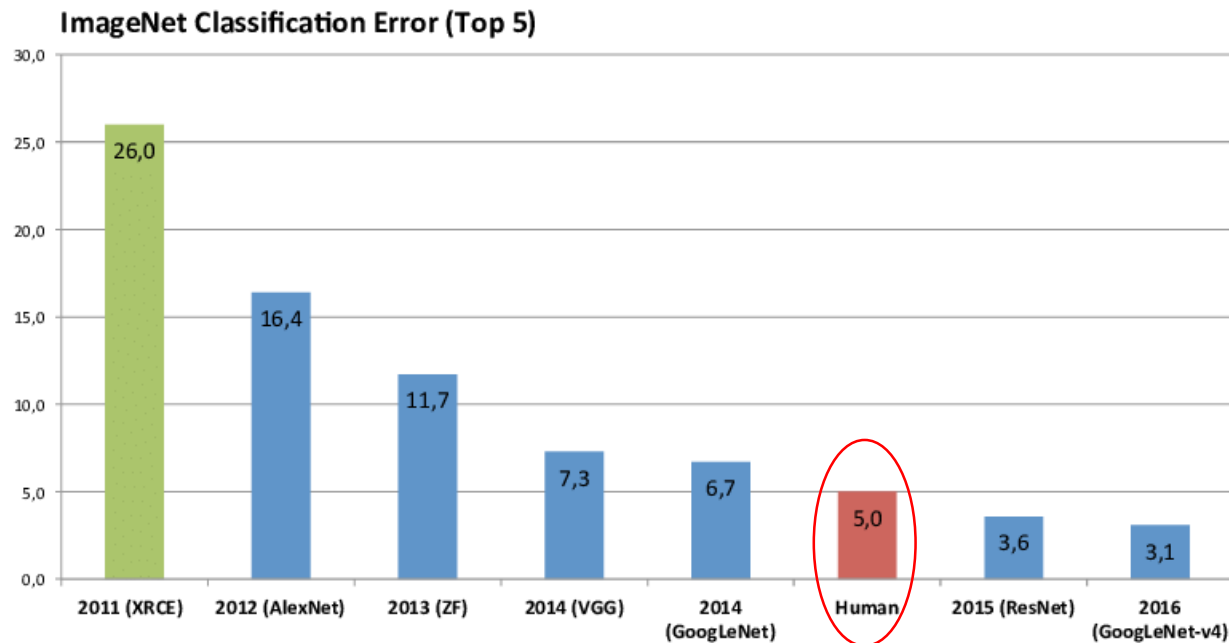
- Redes neurais convolucionais: uso de mascaras de convolução.



Fonte: <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>

ImageNet

- CNN popularizada com a rede AlexNet (2012).
- Vencedores das últimas edições da competição ImageNet tem sido sempre CNNs, cada vez mais profundas.

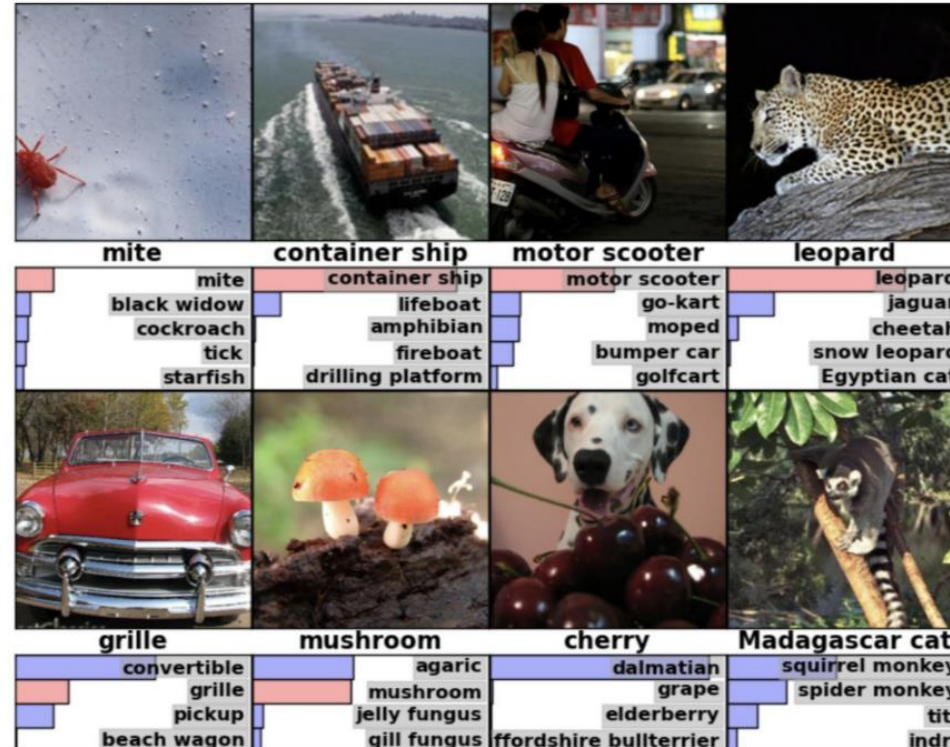


ImageNet

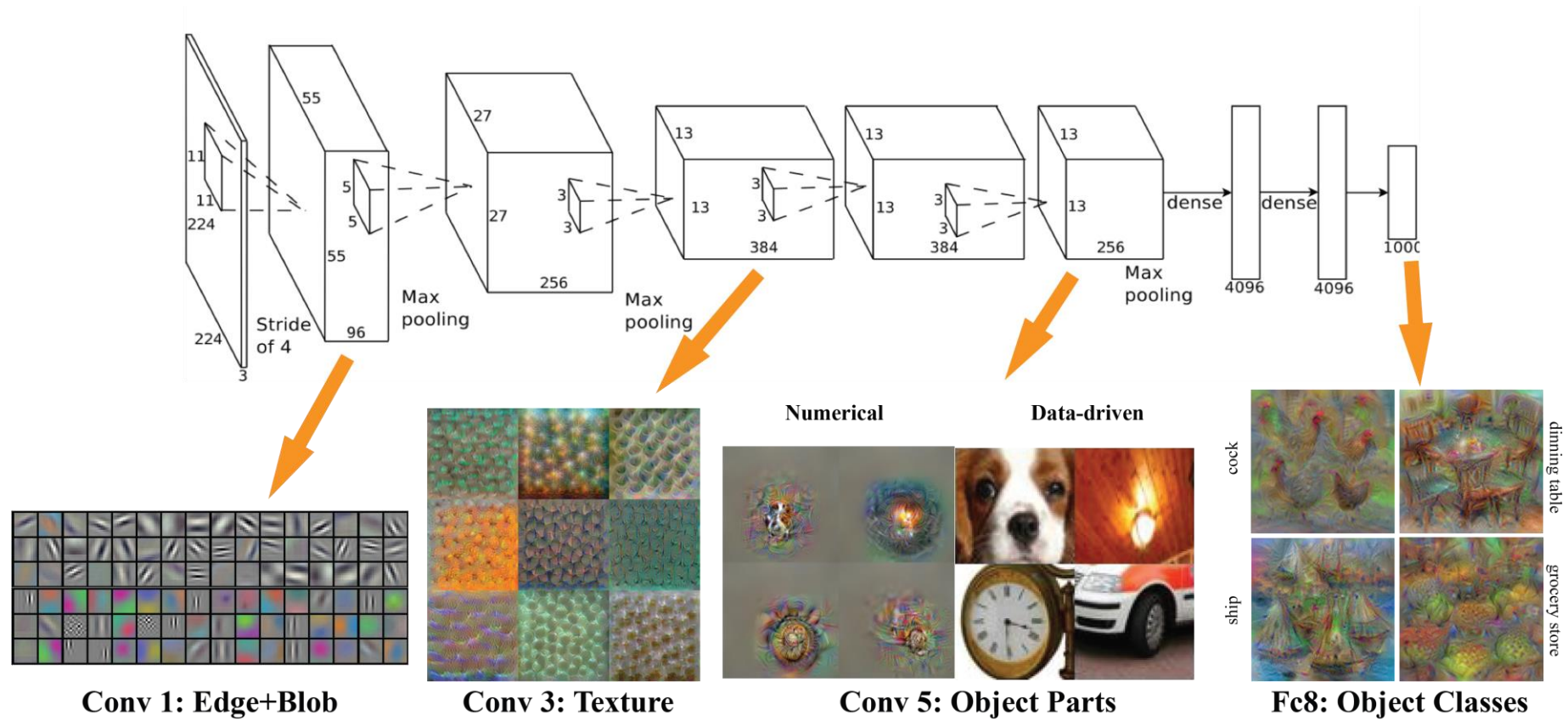
ImageNet Challenge



- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.



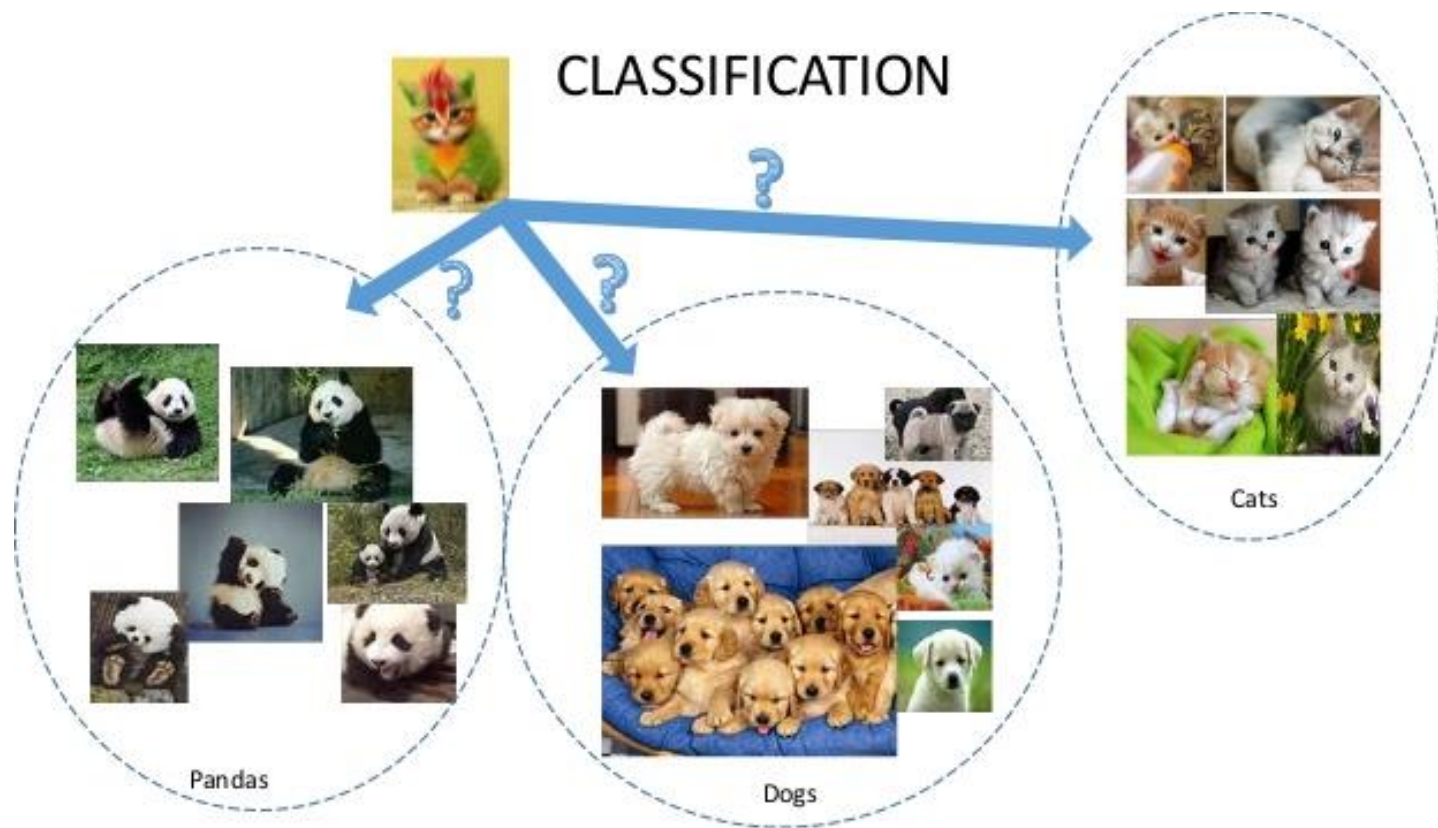
O que uma CNN aprende?



Fonte: http://vision03.csail.mit.edu/cnn_art/index.html

Aplicações de CNNs

Classificação



Fonte: <https://guide.freecodecamp.org/machine-learning/deep-learning/>

Detecção (e Localização) de Objetos

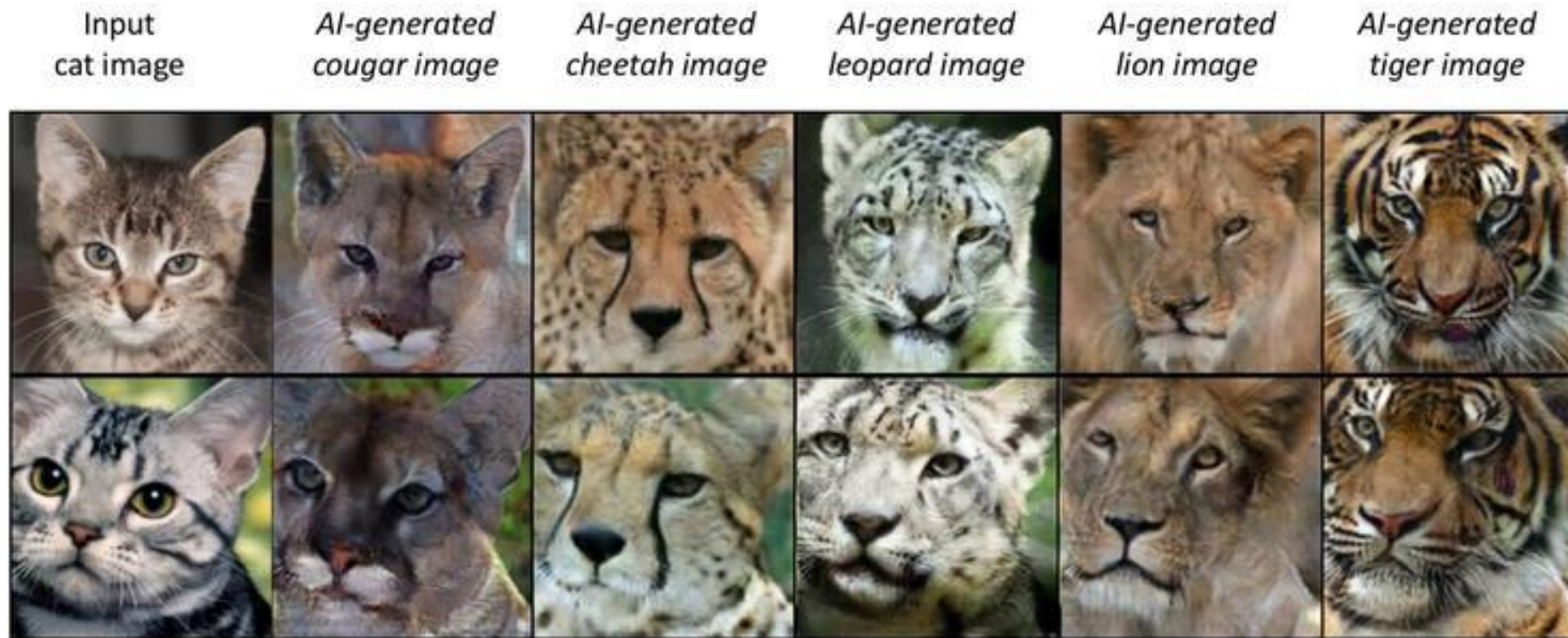


Neural Style Transfer



Fonte: <http://kvfrans.com/neural-style-explained/>

Generative Adversarial Networks



Fonte: <https://www.zdnet.com/article/nvidia-looks-to-reduce-ai-training-material-through-imagination/>

Generative Adversarial Networks

INPUT



OUTPUT



Fonte: https://motherboard.vice.com/en_us/article/8qeyyb/build-your-own-terrifying-cat-blob-with-machine-learning

Convolução

Operação de Convolução

- Camadas convolucionais representam o principal tipo de camada em uma CNN.
- Aplica-se uma máscara (filtro/*kernel*) de convolução à imagem.
- Já vimos essa ideia na aula de Visão Clássica.
- Vamos motivar com um exemplo de detecção de borda.

Operação de Convolução

Imagem de entrada I

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



Máscara de convolução $C (f \times f)$

1	0	-1
1	0	-1
1	0	-1



=

Image de saída O

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



$$\bullet O_{p,q} = \sum_{i=-f/2}^{f/2} \sum_{j=-f/2}^{f/2} I_{p+i,q+j} C_{i,j}$$

Operação de Convolução

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Operação de Convolução

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Operação de Convolução

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Operação de Convolução

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Operação de Convolução

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Operação de Convolução

0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10



*

1	0	-1
1	0	-1
1	0	-1



=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



Outras Máscaras

1	1	1
0	0	0
-1	-1	-1

Bordas Horizontais

1	0	-1
2	0	-2
1	0	-1

Sobel

3	0	-3
10	0	-10
3	0	-3

Schaar

Operação de Convolução

0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10



*

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

=

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

- Ideia: que tal aprender o filtro?

Camada de Convolução

- Filtros aprendidos nas primeiras camadas de uma CNN.



Nota Técnica sobre Convolução

- Na verdade, na MAT, a operação de convolução reflete o *kernel* horizontalmente e verticalmente antes da operação.
- A rigor, o nome correto da operação mostrada é correlação cruzada (*cross-correlation*).
- Apesar disso, na Literatura de DL, as pessoas chamam convolução mesmo.
- No caso em que se aprende os pesos, refletir o *kernel* seria apenas adicionar mais operações.

Camada de Convolução

- Aplica um filtro aprendido na saída da camada anterior.
- Número de pesos é muito menor que em camada totalmente conectada.
- Operação muito boa para GPU: mesma conta para cada *pixel*.
- Observação: tamanho do filtro f em geral é ímpar.

Padding

- Na convolução, perde-se informação das bordas da imagem.
- Solução: completar com zeros (*padding*). Exemplo com $p = 1$.

0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

*

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

=

	?	?	?	?	
	?	?	?	?	
	?	?	?	?	
	?	?	?	?	

Stride

- Pode-se dar saltos maiores no deslocamento da janela da convolução.
- Vamos mostrar exemplo com *stride* $s = 2$ e *padding* $p = 0$ (sem *padding*).

Stride

10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	0
0	30	0
0	30	0

Stride

10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	0
0	30	0
0	30	0

Stride

10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	0
0	30	0
0	30	0

Stride

10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0
10	10	10	0	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

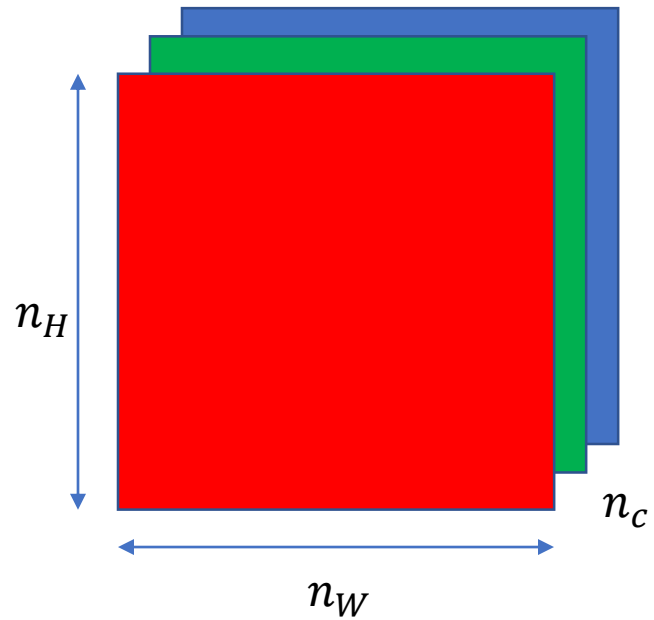
0	30	0
0	30	0
0	30	0

Tamanho da Saída

- Imagem de entrada: $n_H \times n_W$.
- *Stride* $s \times s$.
- *Padding* p .
- Filtro $f \times f$.
- Imagem de saída: $\left\lfloor \frac{n_H + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W + 2p - f}{s} + 1 \right\rfloor$.

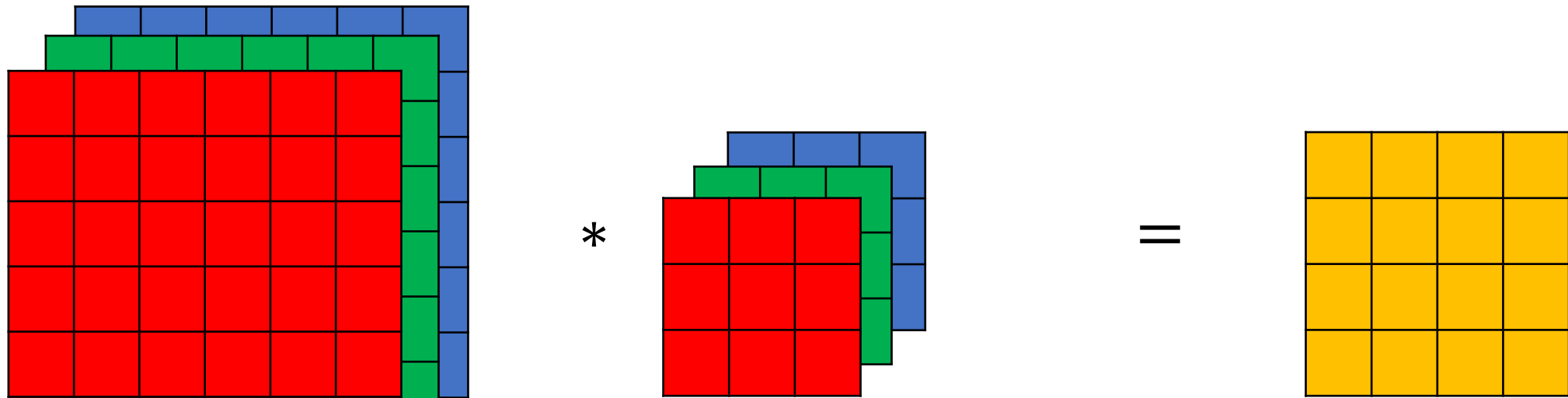
Convolução em Volume

- Imagem como um tensor:



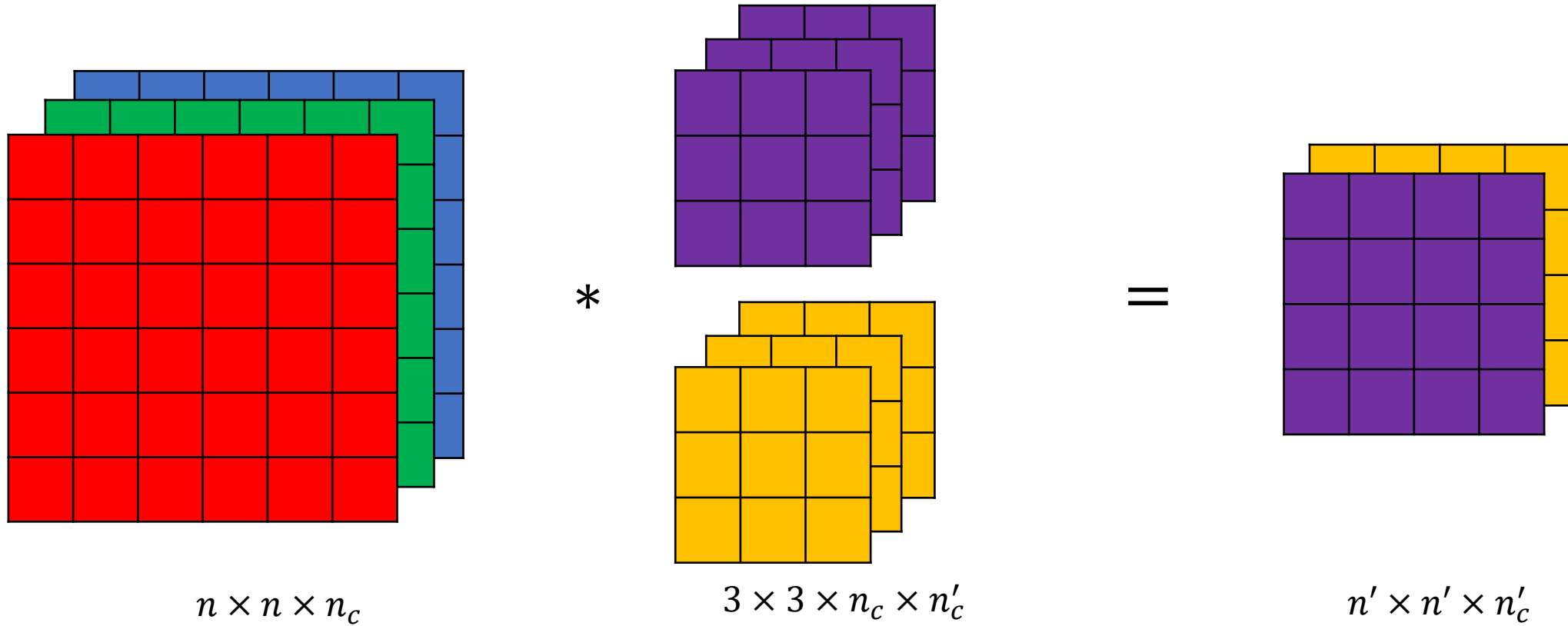
- n_H : altura.
- n_W : largura.
- n_C : número de canais de cor.
- Imagem é um *array* 3D de dimensão $n_H \times n_W \times n_C$.

Convolução em Volume



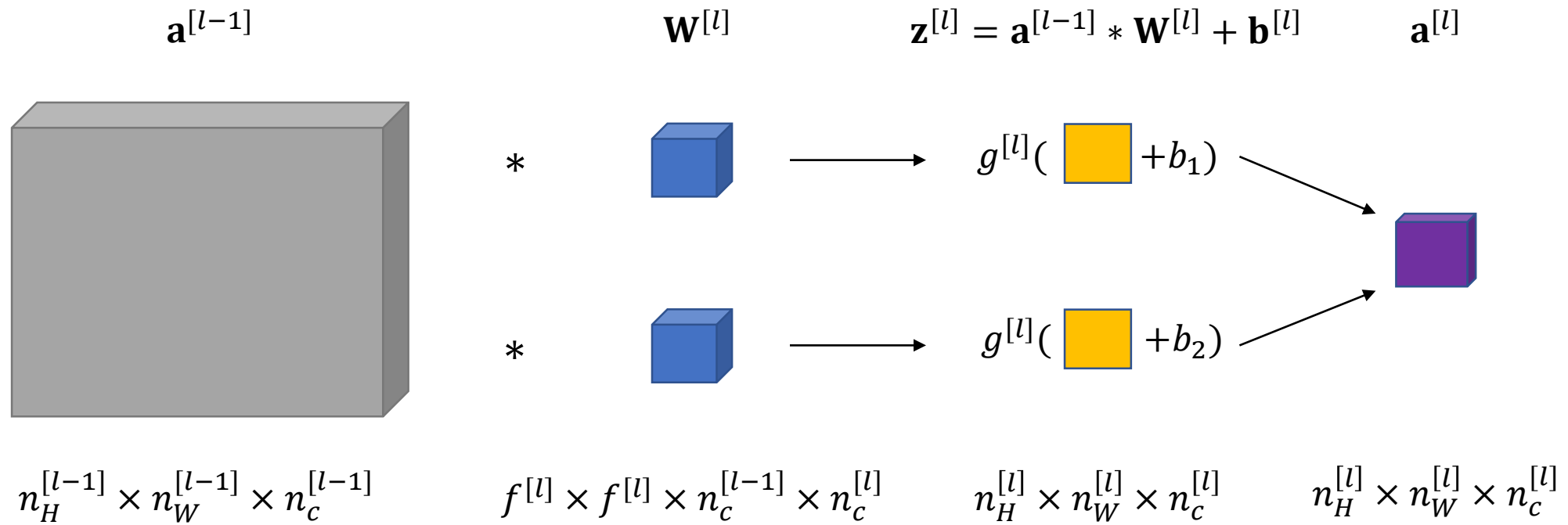
$$O_{p,q} = \sum_{i=-f/2}^{f/2} \sum_{j=-f/2}^{f/2} \sum_{k=1}^{n_c} I_{p+i,q+j,k} C_{i,j,k}$$

Convolução em Volume



Camada de Convolução

- Incluindo *bias* e função de ativação.



Camada de Convolução

$f^{[l]}$: tamanho do filtro.

$p^{[l]}$: *padding*.

$s^{[l]}$: *stride*.

$n_c^{[l]}$: número de filtros.

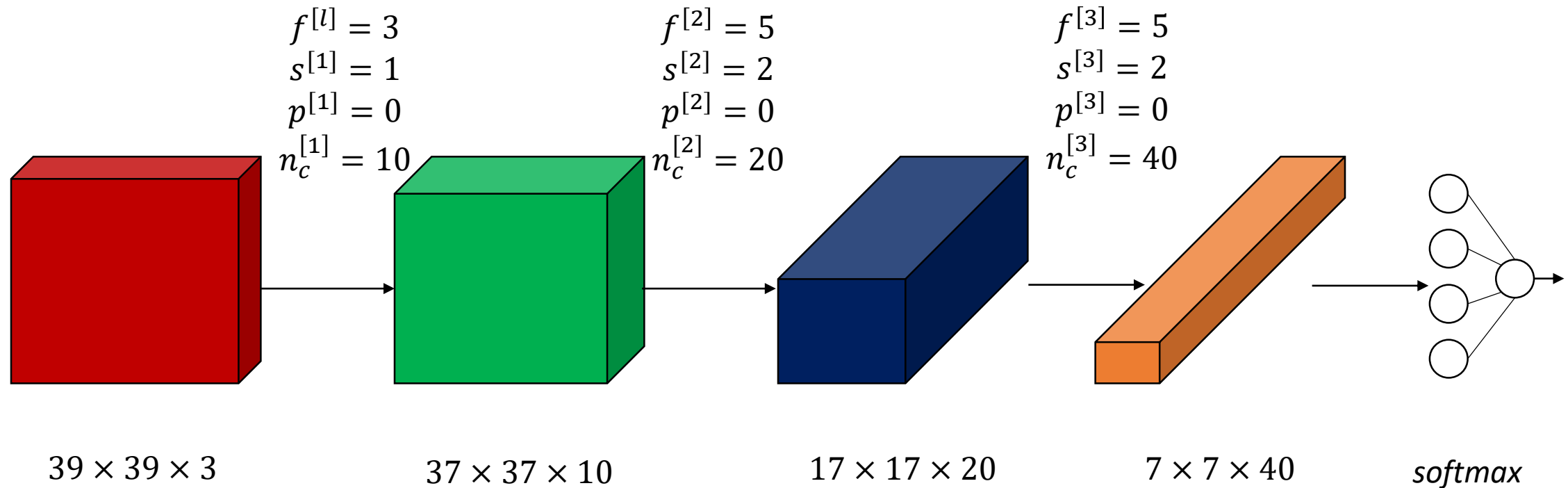
hiperparâmetros

Número de pesos: $f^{[l]} f^{[l]} n_c^{[l-1]} n_c^{[l]}$

Número de biases: $n_c^{[l]}$

Número de parâmetros: $f^{[l]} f^{[l]} n_c^{[l-1]} n_c^{[l]} + n_c^{[l]}$

Exemplo CNN Simples



Pooling

Tipos de Camadas de uma CNN

- Convolucional (CONV).
- *Pooling* (POOL).
- *Fully Connected* (FC): camada comum de rede *feedforward*.

Pooling

- Também aplica uma operação numa janela da imagem.
- Operação não envolve convolução.
- Janela de tamanho $f \times f$ é deslocada com certo *stride* s , assim como numa camada convolucional.
- Tipicamente, faz *downsample* na imagem.
- Tipicamente, não envolve parâmetros a serem ajustados.

Max Pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2



9	2
6	3

Max Pooling

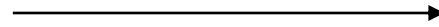
1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9



9	9	5
9	9	5
8	6	9

Max Pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9



9	9	5
9	9	5
8	6	9

Max Pooling

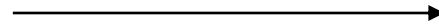
1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9



9	9	5
9	9	5
8	6	9

Max Pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9



9	9	5
9	9	5
8	6	9


Average Pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2



3,75	1,25
4	2

Pooling

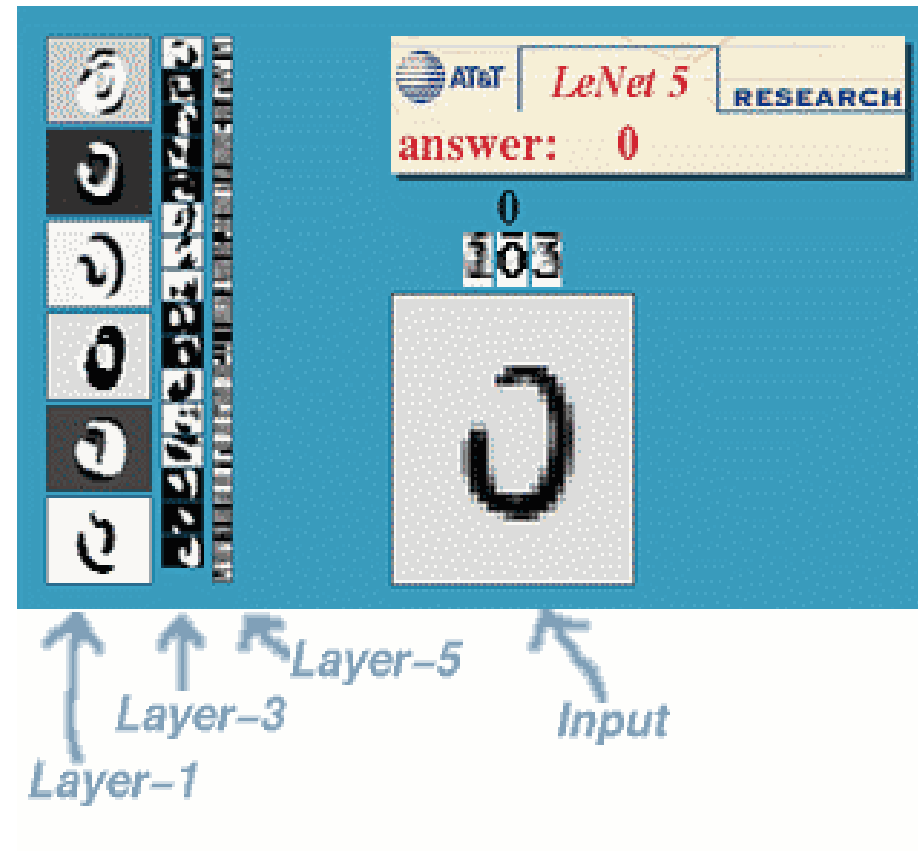
- $f^{[l]}$: tamanho do filtro.
 - $s^{[l]}$: *stride*.
 - Operação (*max*, *average* etc.).
- 
- hiperparâmetros

Estudos de Casos

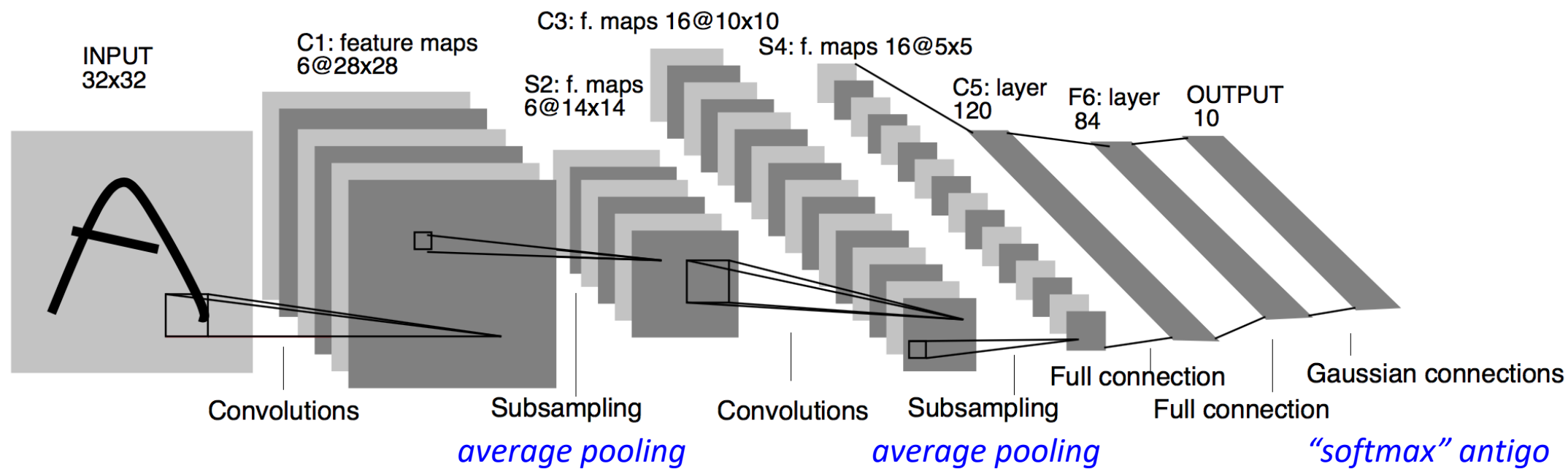
LeNet-5

- CNN famosa.
- Detecção de caracteres.
- Desenvolvida por Yan LeCun, Yoshua Bengio et al.

LeNet-5



LeNet-5



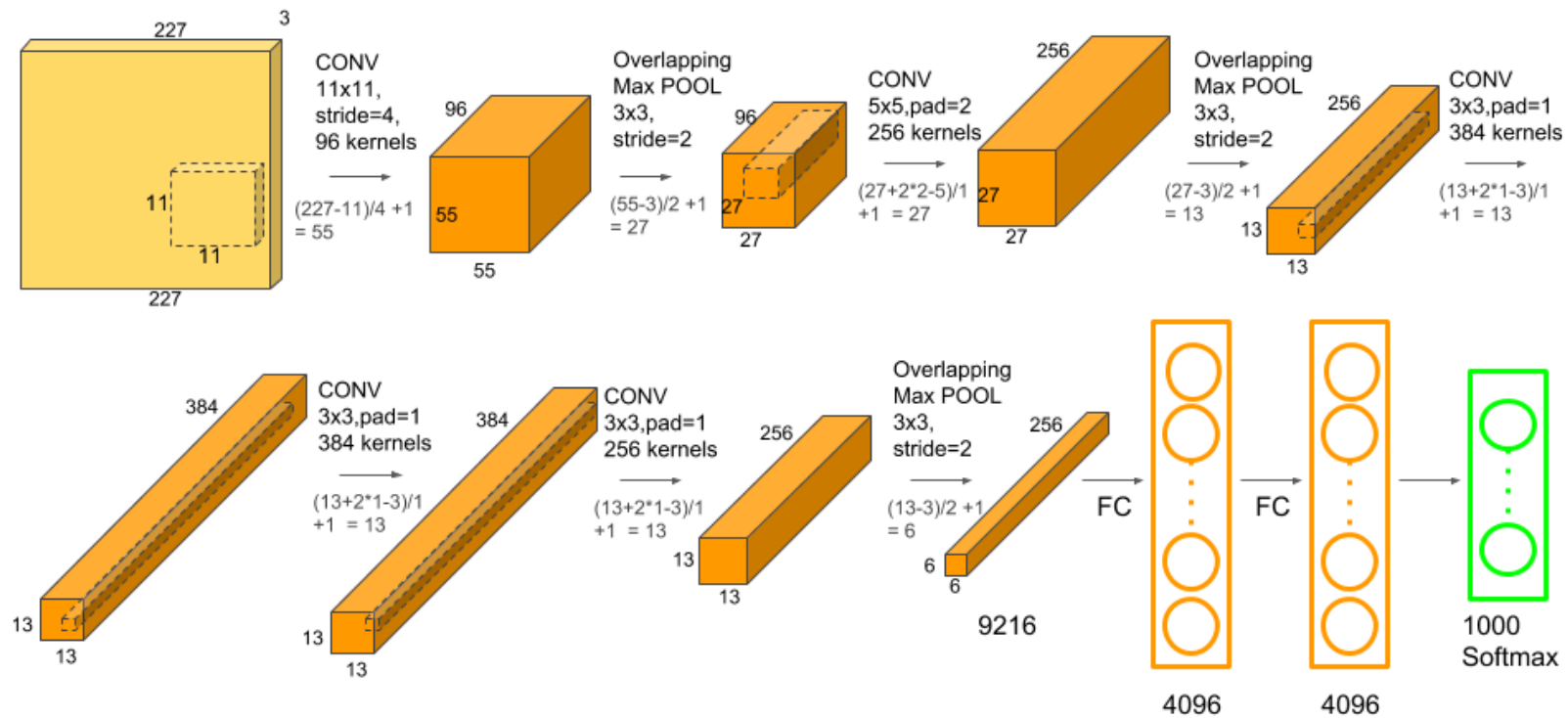
Fonte: <https://engmrk.com/lenet-5-a-classic-cnn-architecture/>

LeNet-5

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

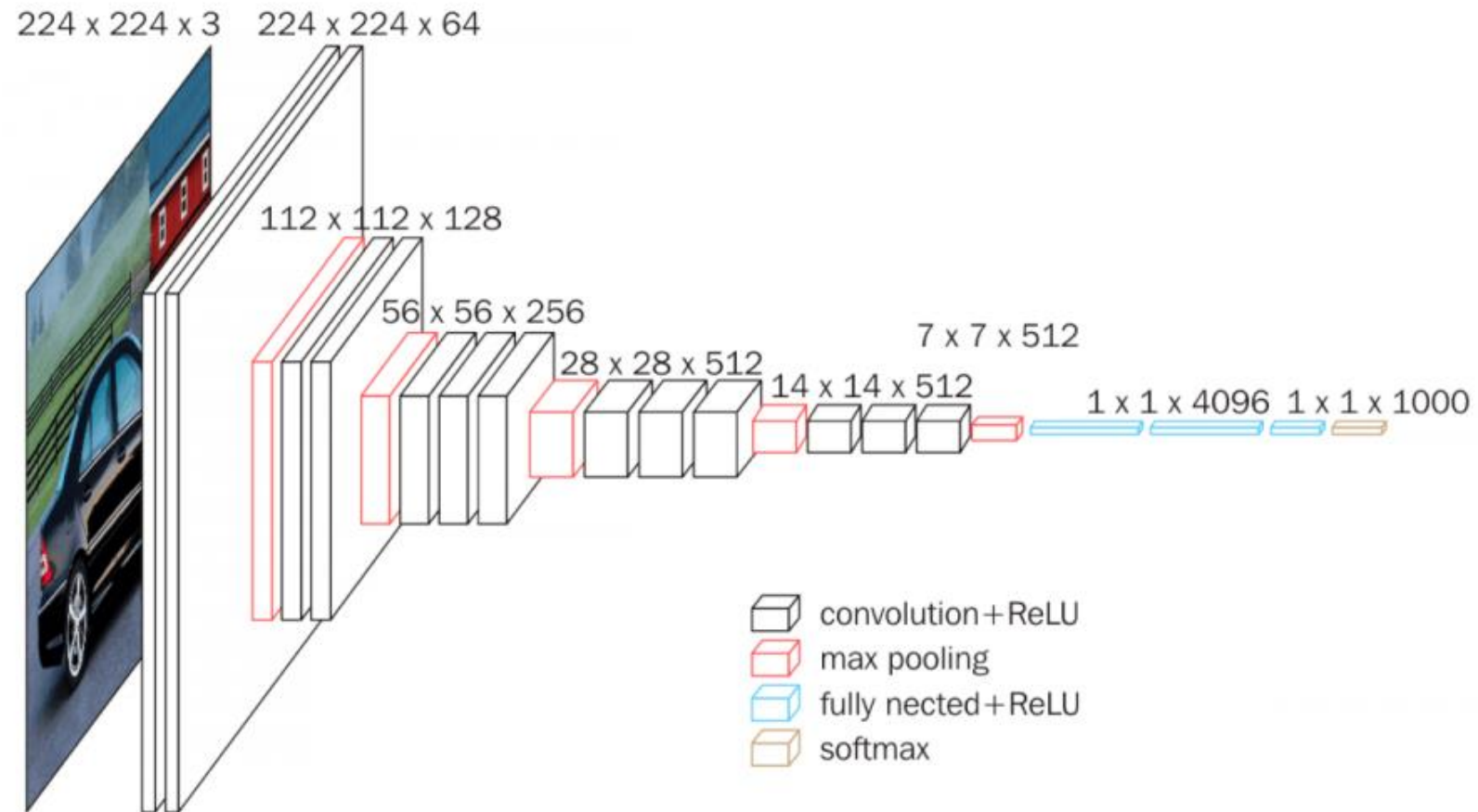
AlexNet

Função de ativação: ReLU



Fonte: <https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/>

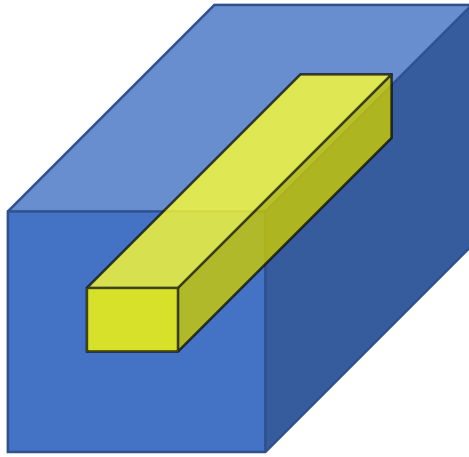
VGG



Fonte: <https://neurohive.io/en/popular-networks/vgg16/>

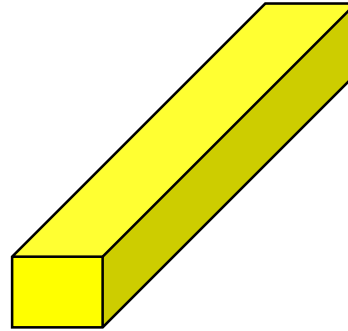
Inception Network

Convolução 1x1



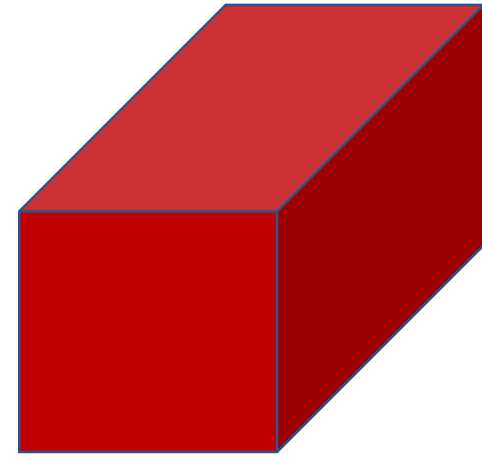
$$n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$$

*



$$1 \times 1 \times n_C^{[l]}$$

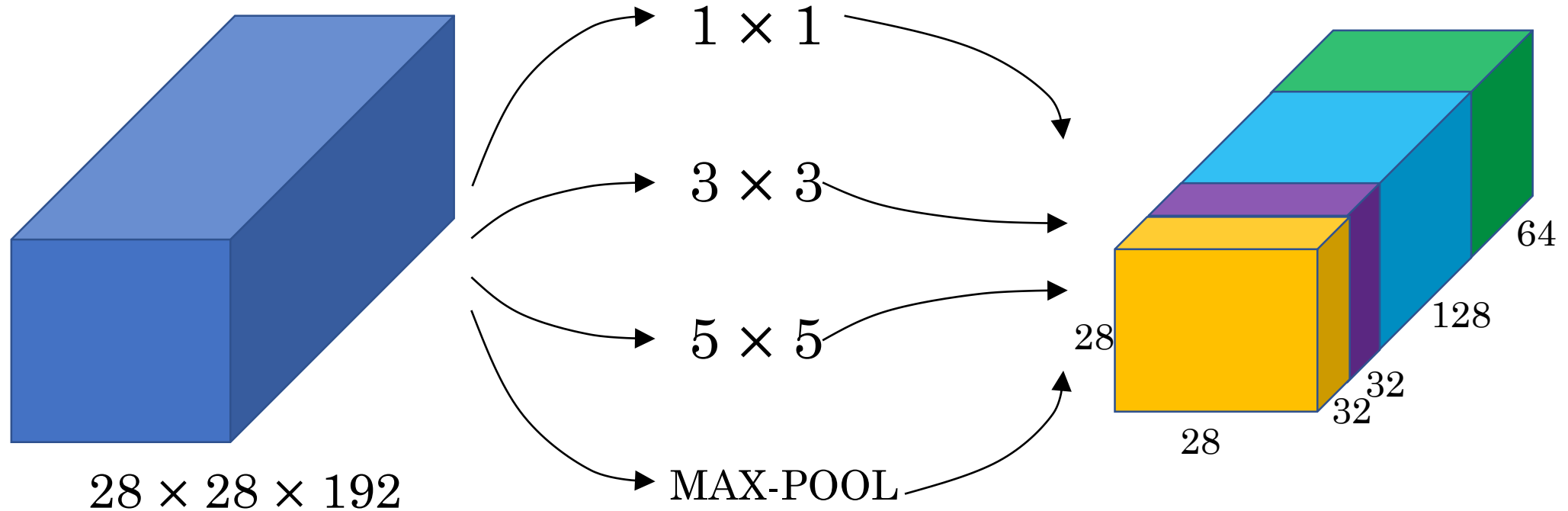
=



$$n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l]}$$

Inception Network

- Escolhemos tipo e tamanho do filtro na “mão”. Por que não deixar a rede decidir?

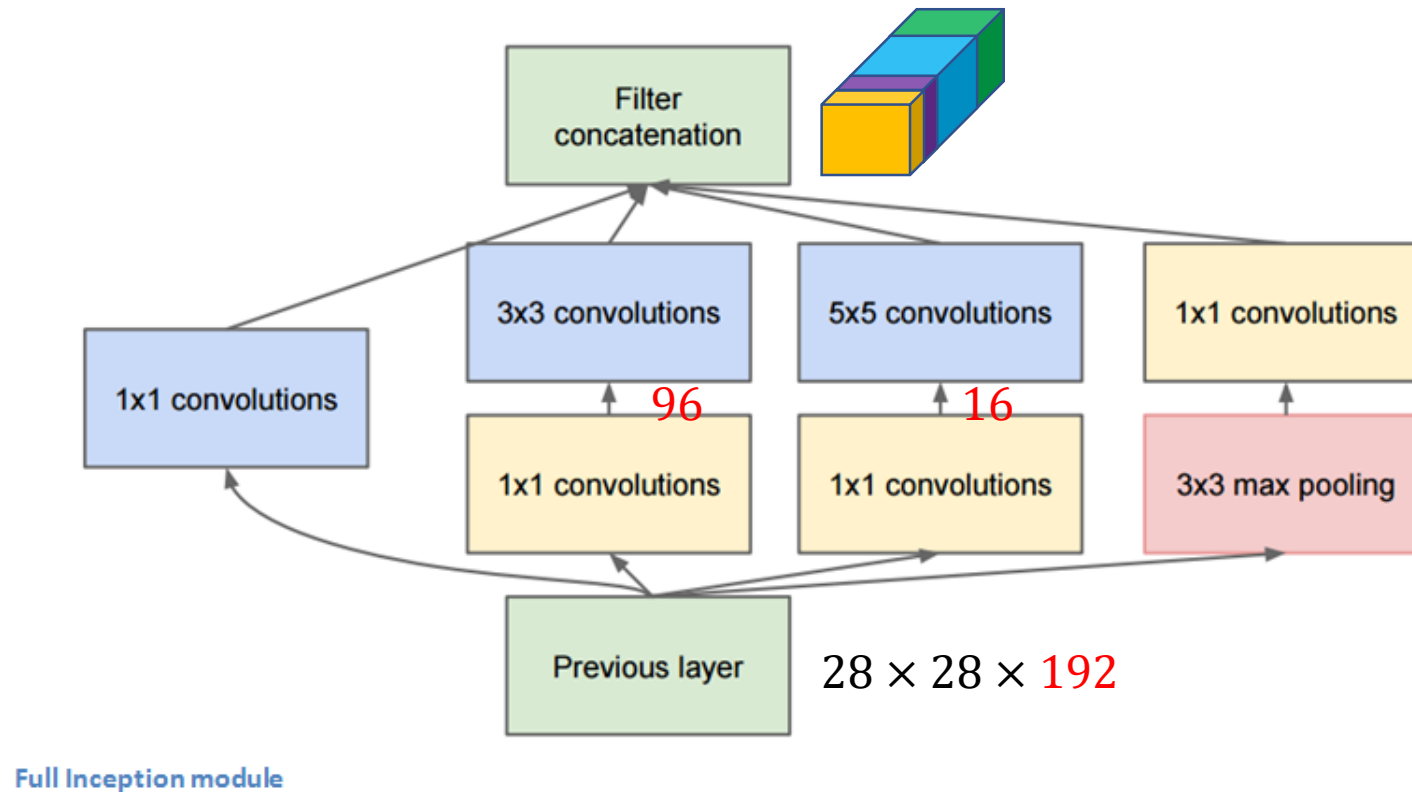


Inception Network

- Isso tem o problema do aumento do custo computacional.
- Em especial, filtros 5x5 introduzem muitos parâmetros.
- Solução: fazer CONV 1x1 para reduzir número de filtros antes de aplicar filtro mais “largo”.

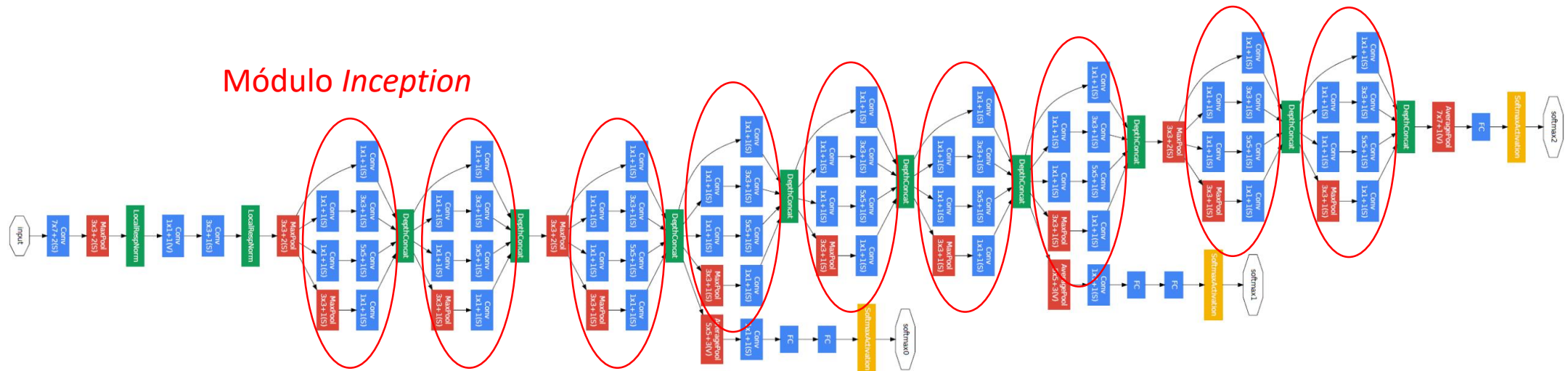
Inception Network

- Módulo *Inception*:



GoogLeNet (Inception v1)

- Erro de 6,7% no ImageNet (vencedora de 2015).
- *Inception*: “rede dentro da rede” (referência ao filme).

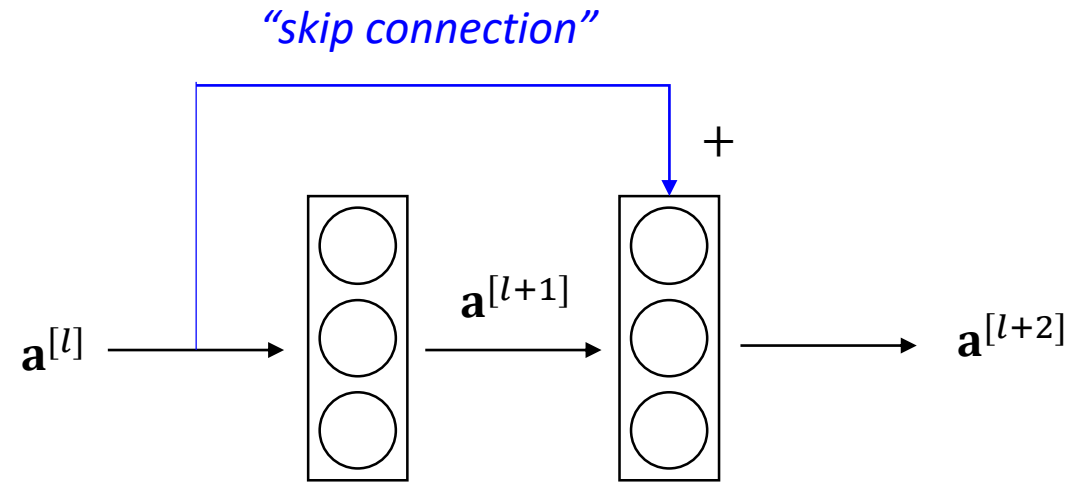


ResNets

Residual Networks

- À medida que as redes vão ficando muito profundas, fica mais difícil treiná-las por conta do problema de *vanishing/exploding gradients*.
- ResNets ajudam a mitigar esse problema.

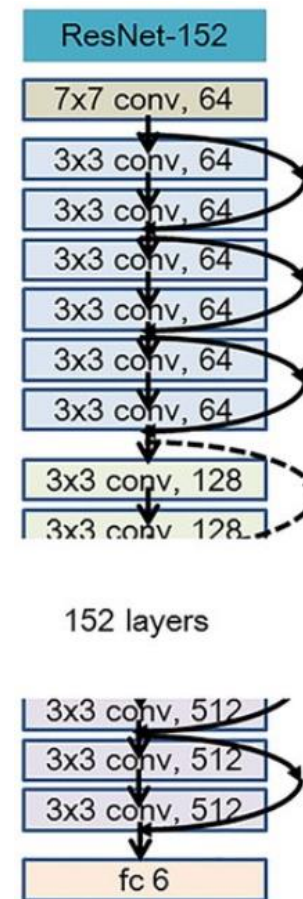
Residual Block



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

ResNet-152

- 152 camadas.
- Uso de ResNet.
- Erro de 3,57% no ImageNet (vencedora de 2015).
- Desempenho super-humano.



Transfer Learning

Transfer Learning

- Treinar CNNs profundas como as apresentadas exigem *hardware* computacional que apenas grandes empresas (e.g. Google) e centros de pesquisa tem acesso.
- Como treiná-las para nossos problemas?
- Já vimos que as camadas iniciais tendem a aprender filtros básicos de visão computacional...
- Ideia: reaproveitar camadas iniciais e treinar apenas camadas finais.
- Camadas iniciais “congeladas” durante o treinamento.

Para Saber Mais

- Especialização de *Deep Learning* do Andrew Ng no Cousera (curso de *Convolutional Neural Networks*).
- Capítulos 9 do livro: GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. The MIT Press, 2016.

Laboratório 9

Laboratório 9

- Implementar a rede LeNet-5 usando Keras e Tensorflow.
- Treinar a rede LeNet-5 no problema de identificação de caracteres.