Deep Learning – Parte 2

Redes Convolucionais + Dropout

Índice Geral

1 Redes Neurais Convolucionais

- Parte das figuras desta seção foram extraídas de:
  - ✓ ⟨ https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050 ⟩
  - ✓ ⟨ https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ ⟩
- Parte das figuras desta seção foram extraídas de vídeos de Leo Isikdogan:
  - ✓ ⟨ https://www.youtube.com/watch?v=-I0lry5ceDs ⟩
  - ✓ ⟨ https://www.youtube.com/watch?v=fTw3K8D5xDs ⟩
- Parte das figuras desta seção foram extraídas de vídeos de Siraj Raval:
  - ✓ ⟨ https://www.youtube.com/watch?v=FTr3n7uBIuE&t=1339s ⟩
- Para a extração de atributos relevantes de imagens, camadas convolucionais, seguidas de operações de sub-amostragem, como *max pooling*, têm se mostrado o estado-da-arte, ao conduzir a modelos de aprendizado de elevado desempenho.
- Rede neural convolucional (*Convolutional Neural Network*) = ConvNet = CNN

1.1 Inspiração no campo de receptores

- Os operadores convolucionais foram inspirados na forma como os sensores visuais cerebrais de mamíferos operam, usando campos de receptores: Robertson, B. "A celebration of the 50th anniversary of David Hubel and Torsten Wiesel's Receptive fields of single neurones in the cat's striate cortex", The Journal of Physiology, vol. 587, no. 12, pp. 2721-2732, 2009.

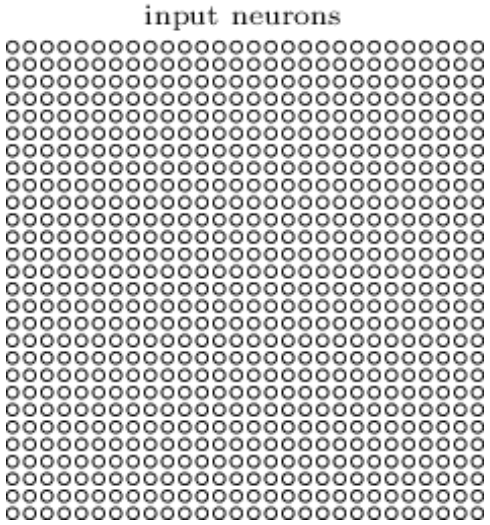


Figura 1 – Campo de receptores em um sensor visual

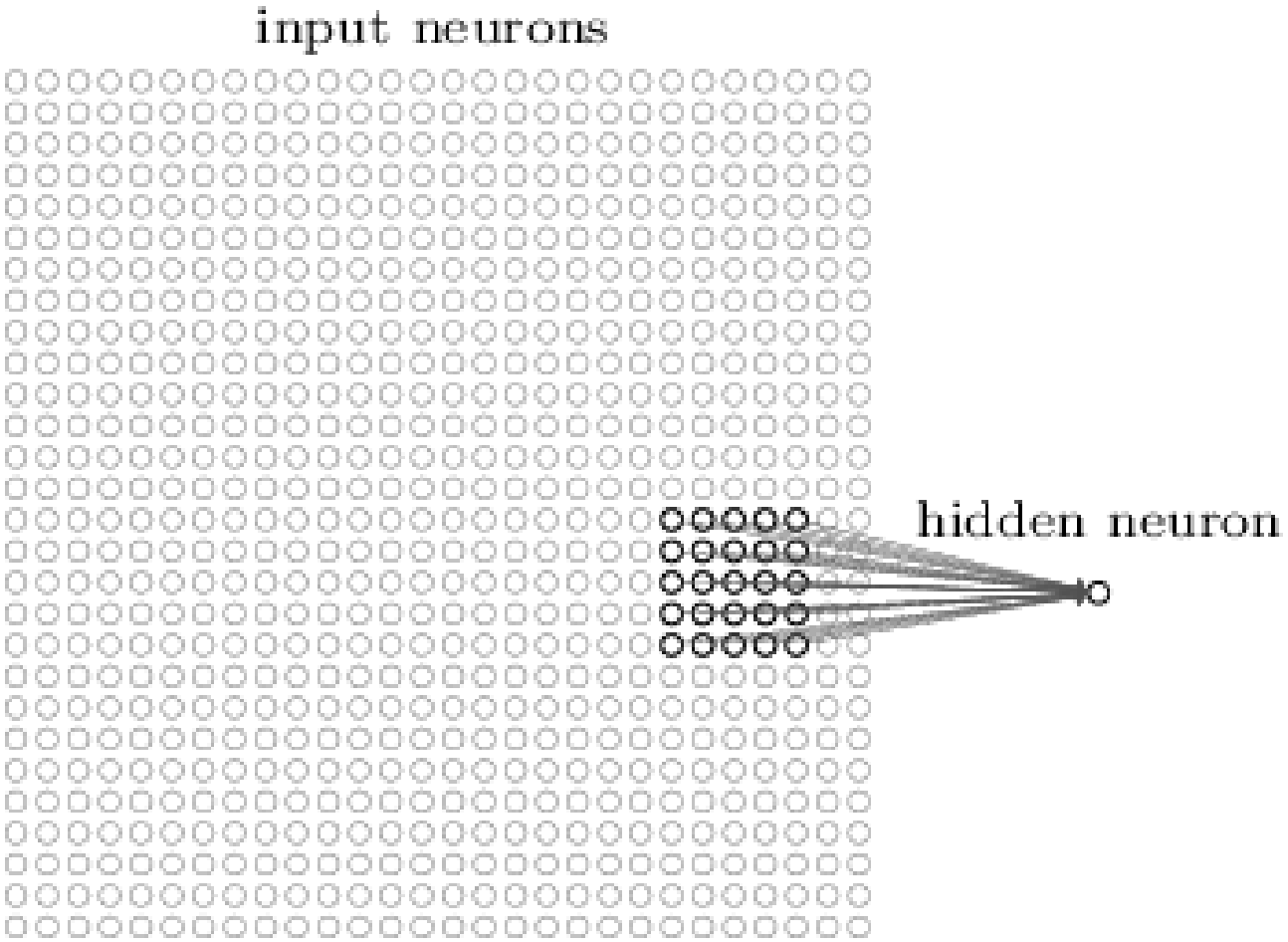


Figura 2 – Campo de receptores como entrada para filtros locais

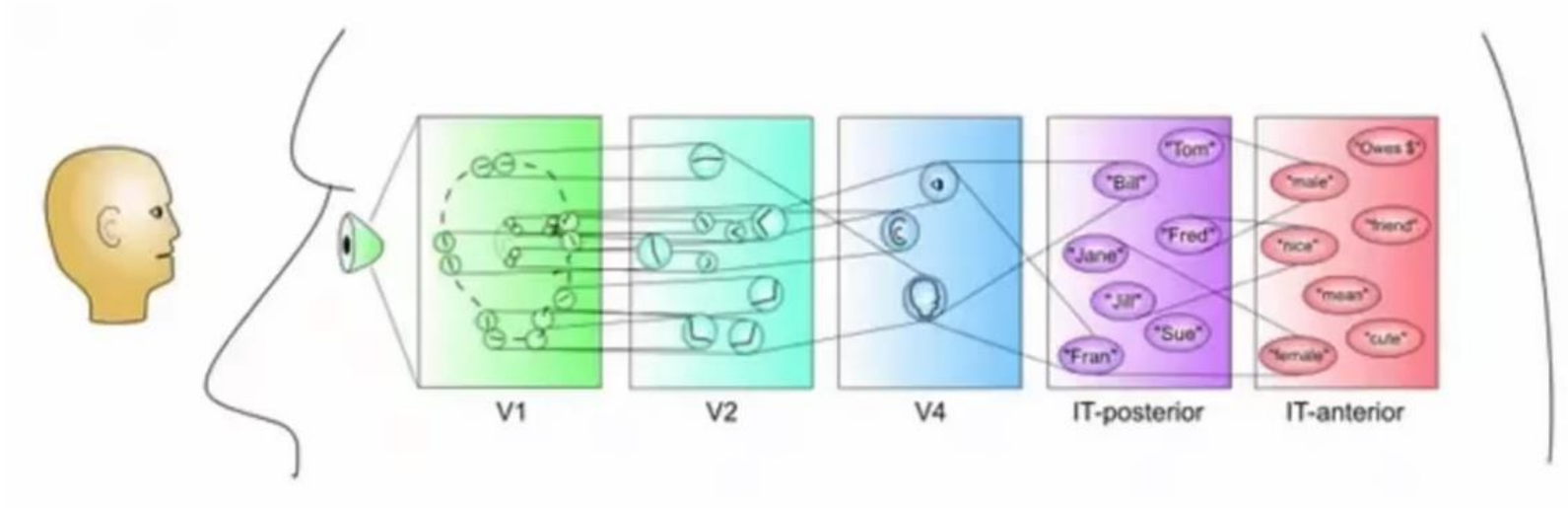


Figura 3 – Uma arquitetura em camadas que implementa uma hierarquia de campos de receptores, caracterizados por grupos de neurônios que reagem a conceitos específicos

- A inspiração levou a modelos de visão computacional caracterizados por:
  - ✓ Conexões locais;
  - ✓ Estrutura em camadas;
  - ✓ Operadores com algum grau de invariância espacial, ou seja, invariância a posição, tamanho, contraste, rotação e orientação.

- Mas tenha em mente que computadores e cérebros não “enxergam as coisas” do mesmo jeito.

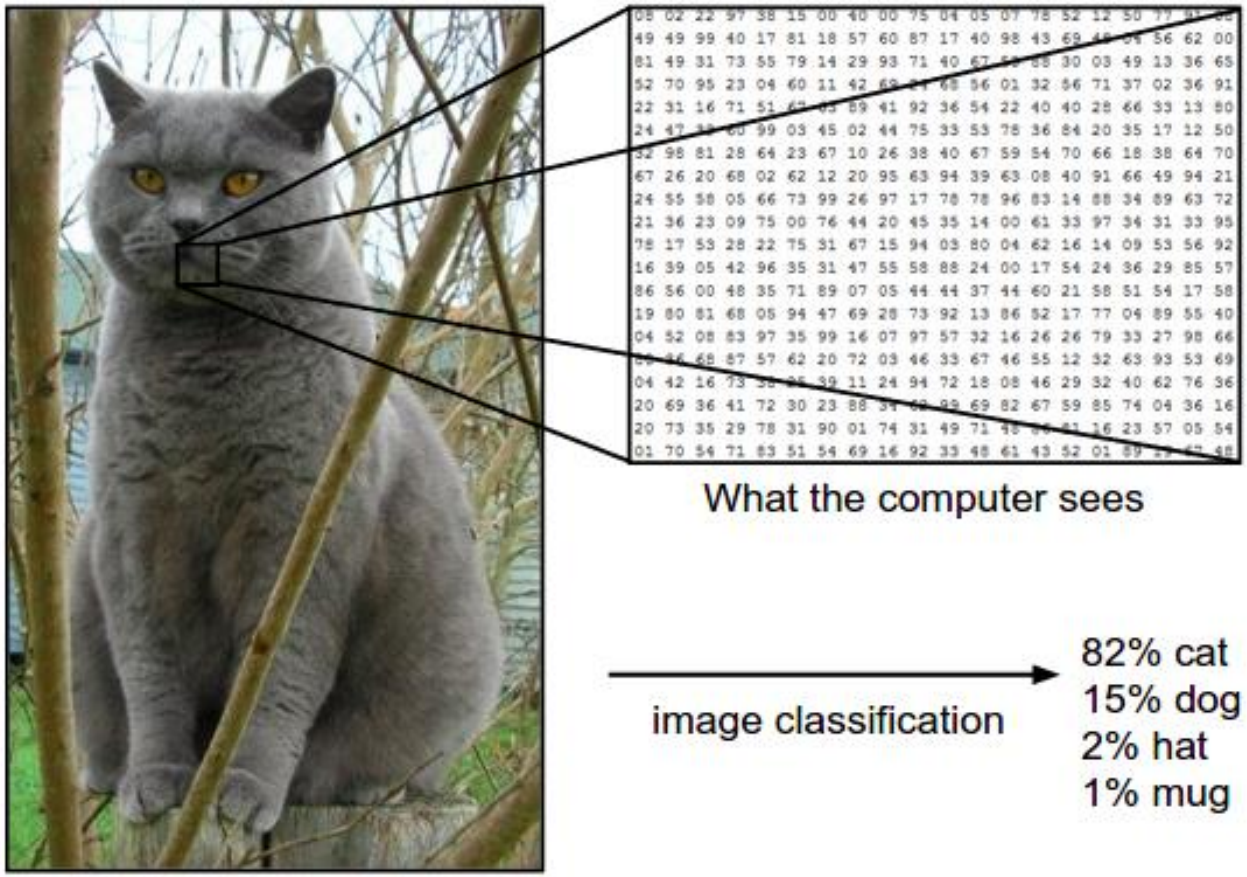
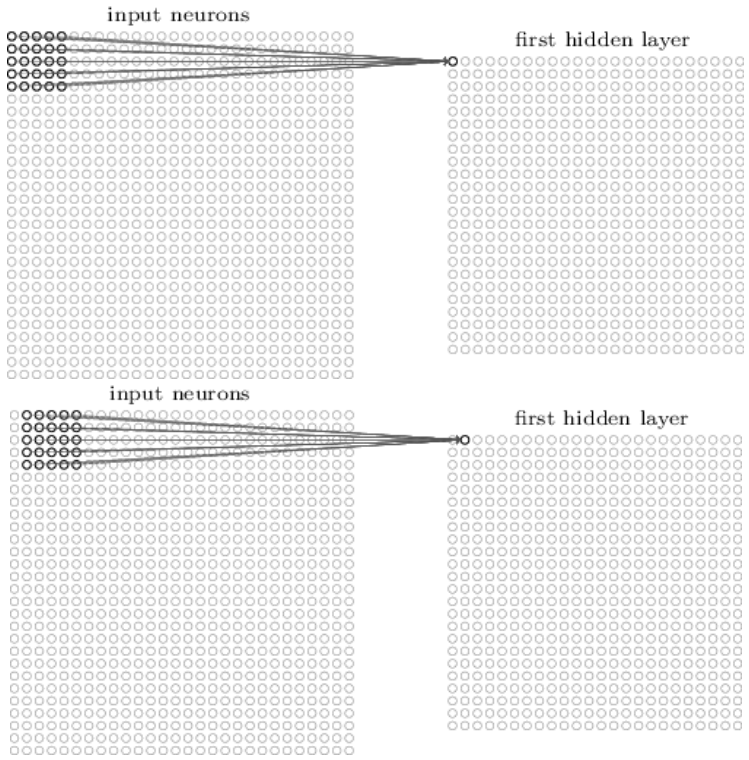


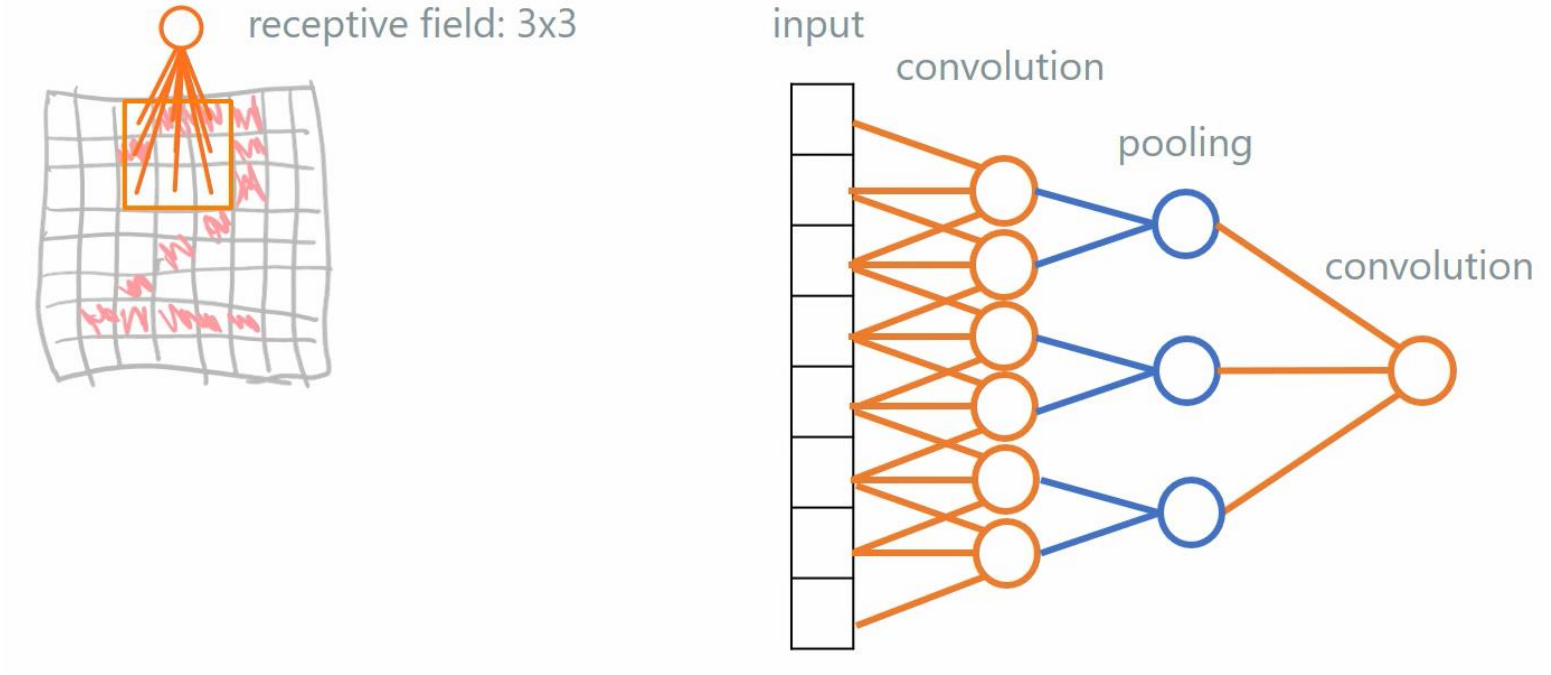
Figura 4 – Entrada e saída de um processo computacional para classificação de imagens

- Cada filtro local obtém uma saída a partir de *matching* com uma região do campo de receptores, passando o grau de *matching* por uma função não-linear.
- Imagens coloridas vão apresentar 3 campos de receptores, para o padrão RGB.
- É possível reposicionar cada filtro, varrendo o campo (os 3 campos, também chamados de canais) de receptores segundo uma regra específica. O resultado desta operação de varredura é uma outra imagem, produzida pela ativação não-linear do filtro, ao varrer todo o campo de receptores.
- O passo de varredura (*stride*) é um hiperparâmetro a ser definido, assim como as dimensões do filtro, o número de filtros e a presença ou não de *padding*.
- Para que o filtro sempre possa estar centrado no pixel corrente, mesmo sendo este pixel de borda ou canto, pode-se recorrer ao conceito de preenchimento (*padding*).
- Não há um procedimento sistemático para a especificação desses hiperparâmetros, mas uma sugestão é partir de configurações que já levaram a redes neurais de alto desempenho na literatura.

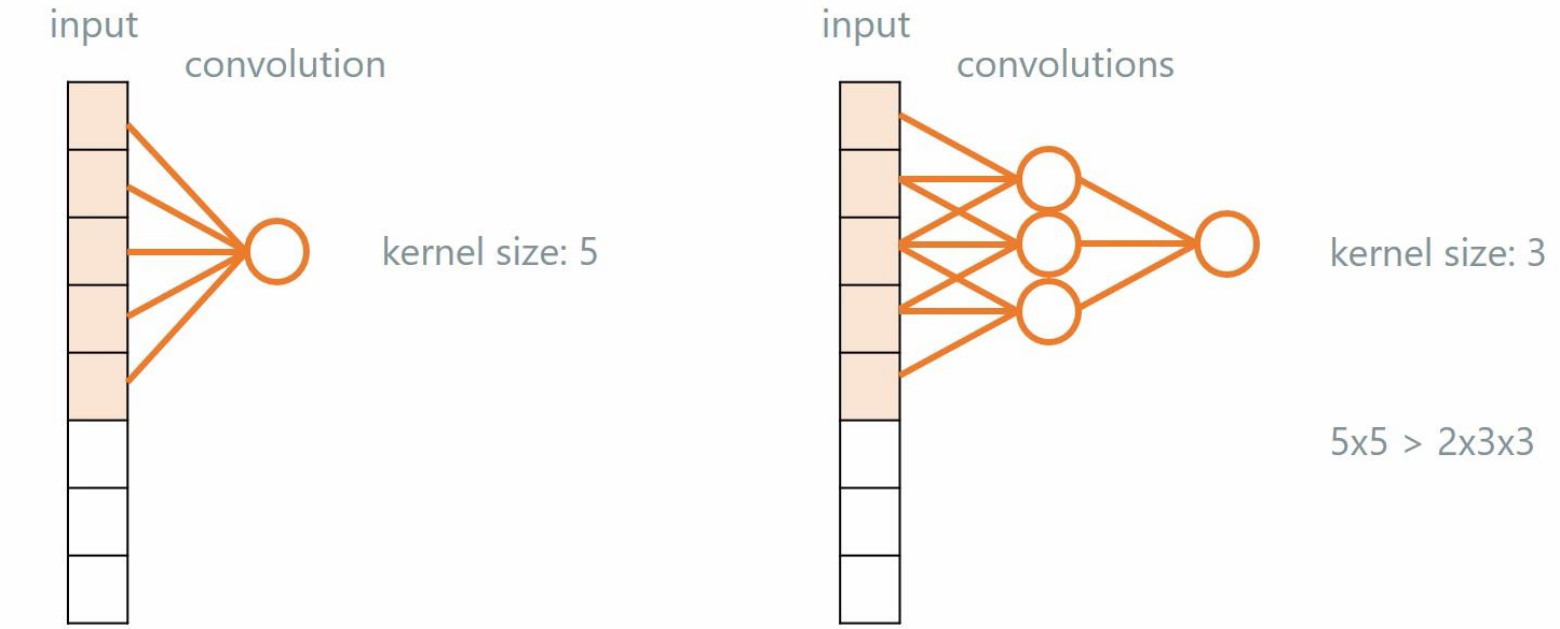


- Em termos fisiológicos, não é um mesmo e único filtro que varre o campo de receptores, mas sim um conjunto de filtros que operam em paralelo.
- Também é importante observar que o próprio campo de receptores está em permanente deslocamento, realizando uma varredura local da imagem que se encontra no campo de visão.

Receptive field

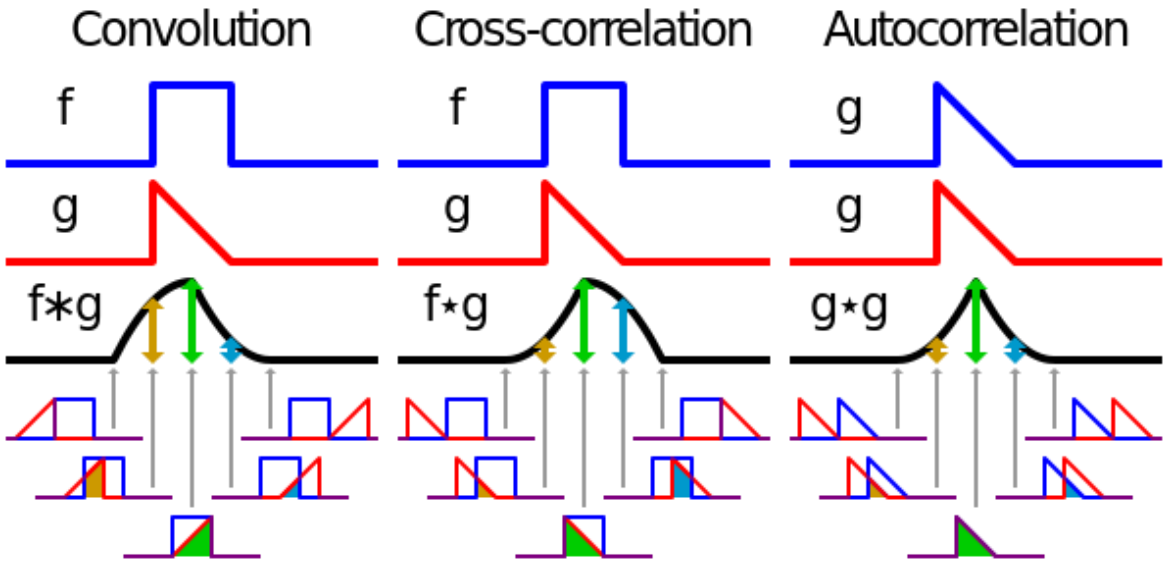


Kernel size


input
convolution
kernel size: 5
input
convolutions
kernel size: 3
5x5 > 2x3x3

1.2 Operação de convolução

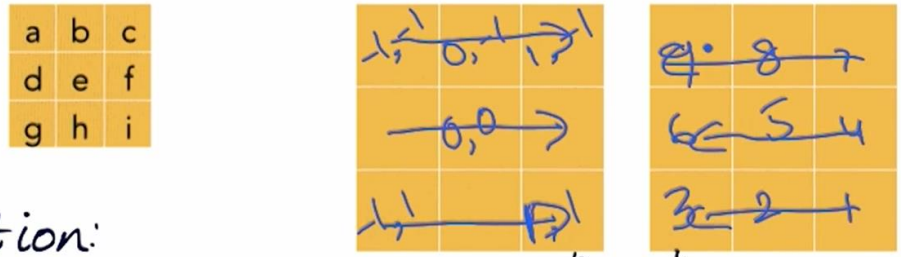
- Em aprendizado de máquina, as operações de convolução e de correlação cruzada são tomadas como sinônimos, embora elas só sejam equivalentes quando os sinais envolvidos são simétricos. O que se faz em camadas convolucionais de redes neurais profundas é correlação cruzada (*cross-correlation*). Em processamento digital de sinais, a correlação cruzada é uma medida de similaridade entre duas séries temporais, como uma função do atraso de uma série relativa à outra.



Convolution vs. Cross-Correlation

Cross-Correlation:

$$G = h \otimes F \qquad G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i+u, j+v]$$



Convolution:

$$G = h * F \qquad G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i-u, j-v]$$

- A convolução é uma operação comutativa ($h * F = F * h$), enquanto que a correlação cruzada não ($h \otimes F \neq F \otimes h$), a menos de sinais simétricos.
- De qualquer modo, vamos empregar o termo convolução daqui em diante, embora a operação realizada seja de fato uma correlação cruzada.

$a * b$

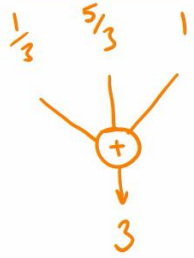
$a = [1, 5, 3, 7, 5, 9, 7, 14]$

$b = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$

$a * b = [3$

$a = [1, 5, 3, 7, 5, 9, 7, 14]$

$b = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$



$a * b = [3, 5$

$a = [1, 5, 3, 7, 5, 9, 7, 14]$

$b = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$

$5 \times \frac{1}{3} + 3 \times \frac{1}{3} + 7 \times \frac{1}{3} = 5$

$a * b = [3, 5, 5, 7, 7, 10]$

$a = [1, 5, 3, 7, 5, 9, 7, 14]$

$b = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$

- O filtro também pode ser chamado de *kernel*. No exemplo considerado, o papel do filtro é realizar médias locais do sinal, produzindo uma versão suavizada na saída.

Convolution

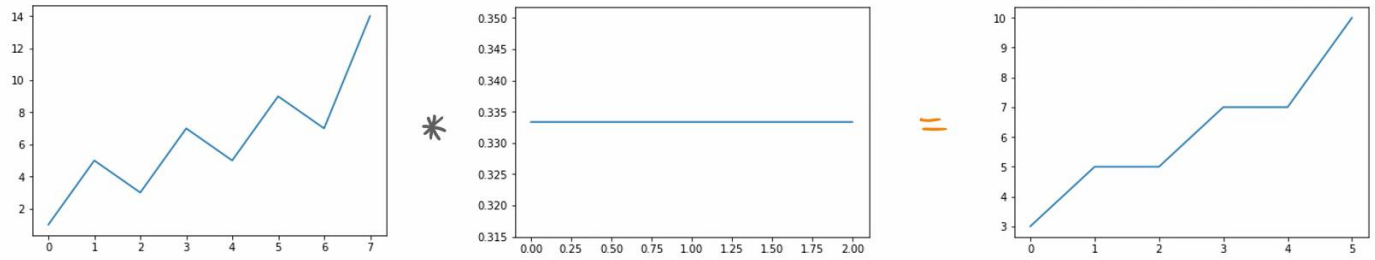
$a * b = [3, 5, 5, 7, 7, 10]$

Signal

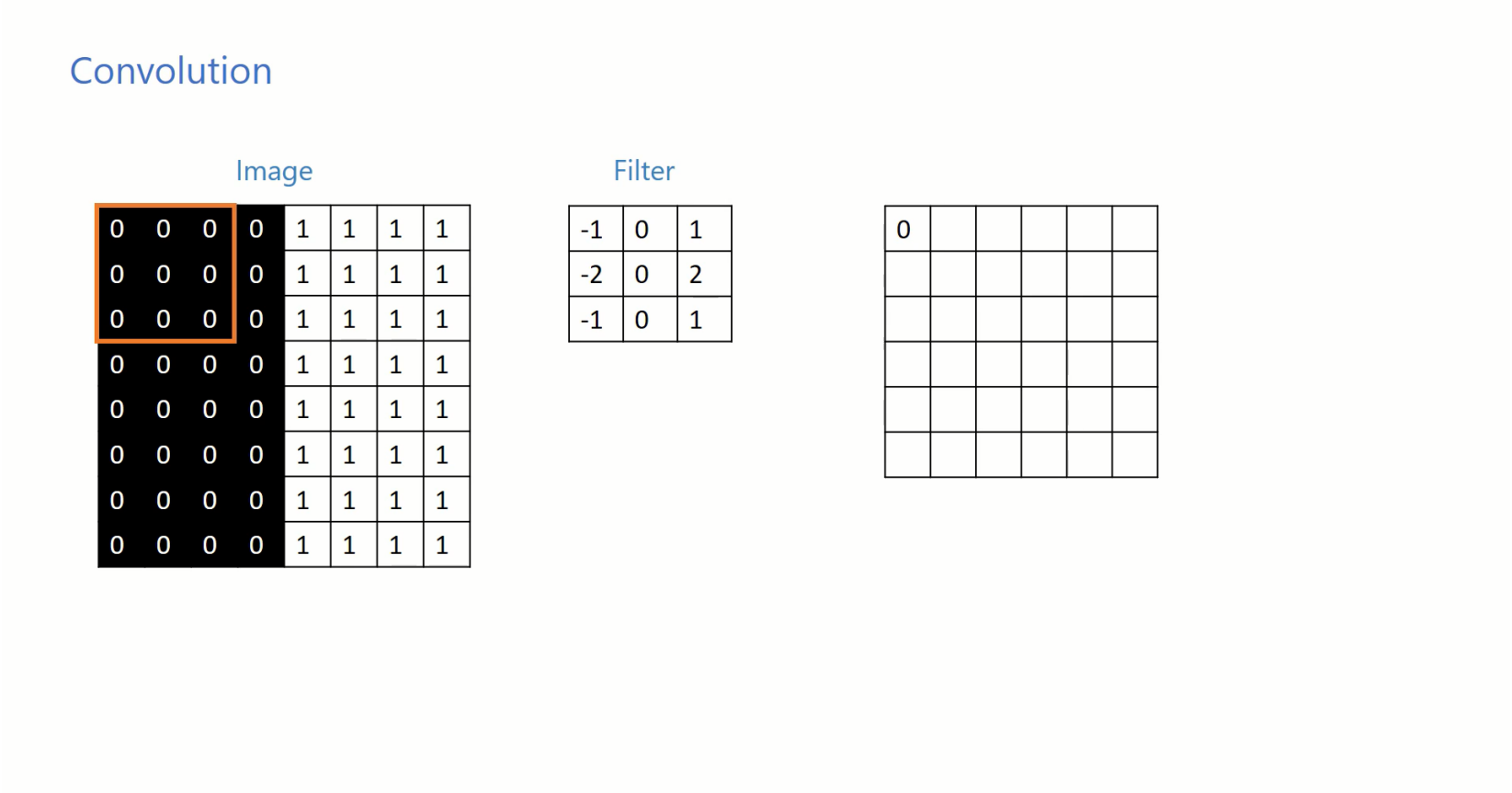
$a = [1, 5, 3, 7, 5, 9, 7, 14]$

Kernel (Filter)

$b = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$



- O que faz o filtro a seguir, numa operação de convolução?



0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

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

0					

Convolution

Image

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

Filter

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

0	0	4	4	0	0
0	0	4	4	0	0
0	0	4			

Convolution

Image

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

Filter

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

0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0

Convolution

Image

Filter

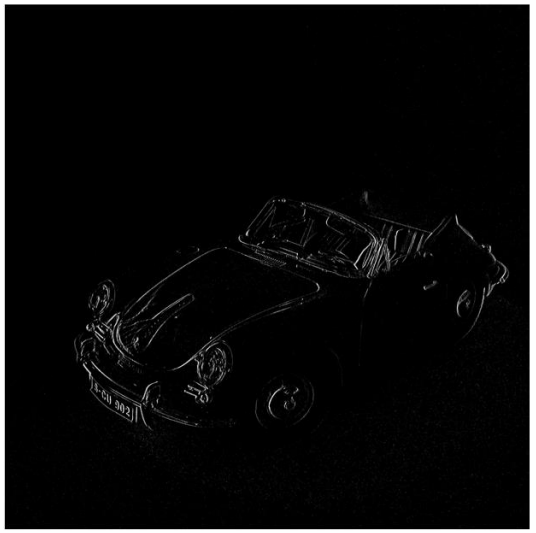
-1	0	1
-2	0	2
-1	0	1

Convolution

Image

Filter

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



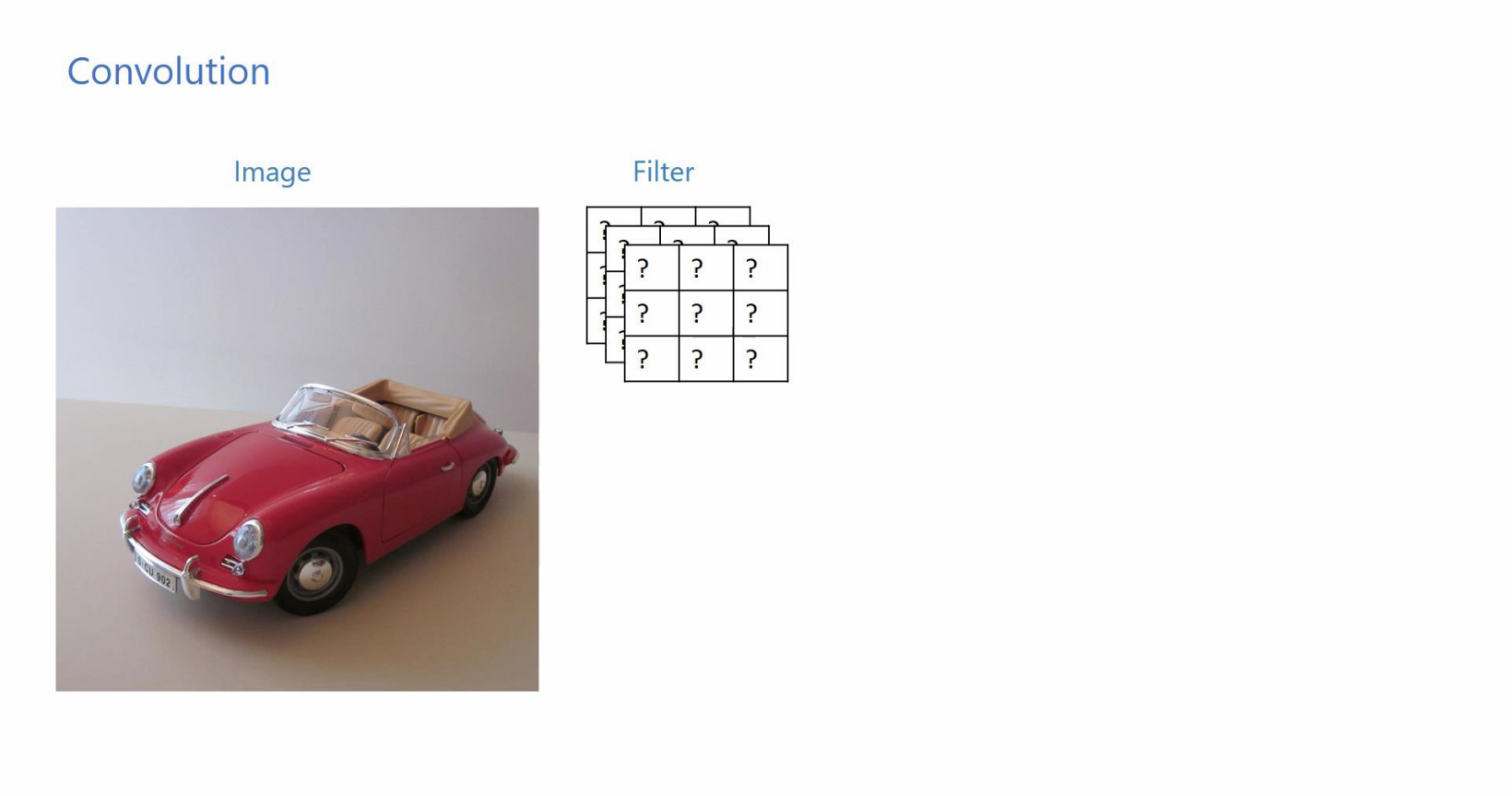
Convolution

Image

Filter

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

- Para imagens coloridas, pode ser definido um filtro para cada canal (R, G e B).
- Repare que cada matriz em qualquer camada é chamada de canal ou mapa de ativação.



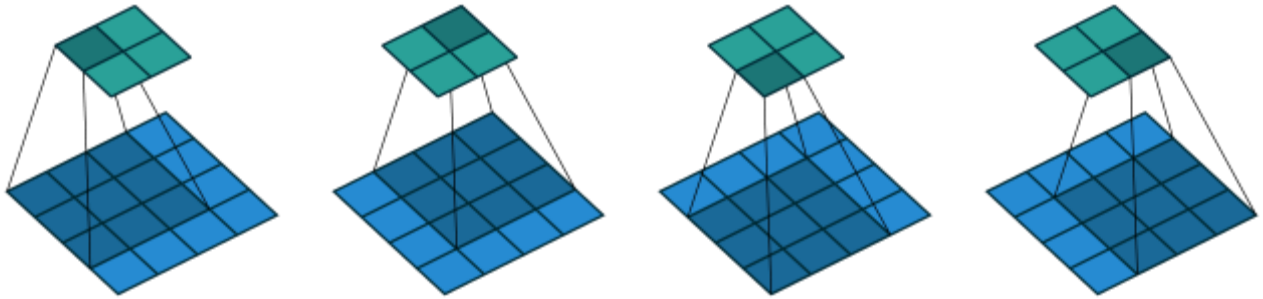


Figura 5 – Campo de receptores (em azul) sem contar com preenchimento (*padding*) e operação do filtro com *stride* = 1, formando um novo campo de receptores (em verde)

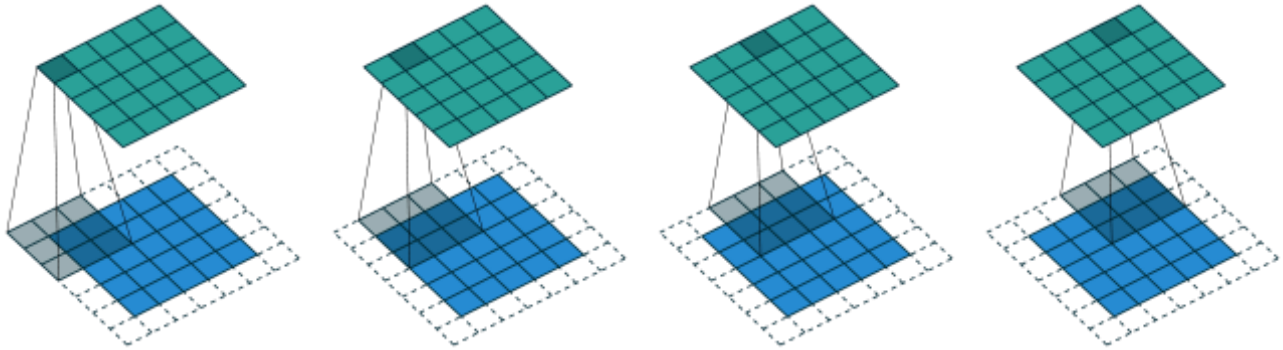


Figura 6 – Mesmo cenário da Figura 5, mas agora contando com preenchimento de 1 (*padding*, em branco)

Fonte das Figuras 5 a 8: ⟨ https://www.groundai.com/project/a-guide-to-convolution-arithmetic-for-deep-learning/ ⟩

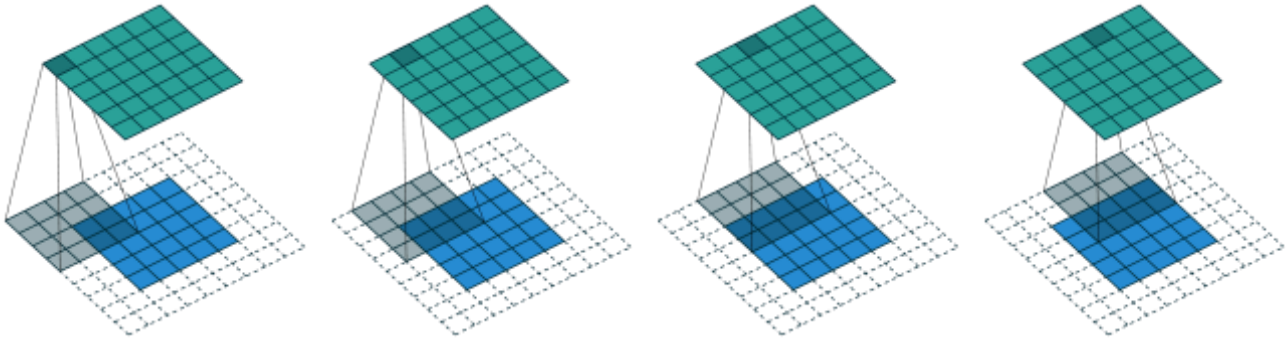


Figura 7 – Mesmo cenário da Figura 5, mas agora contando com preenchimento de 2 (*padding*, em branco)

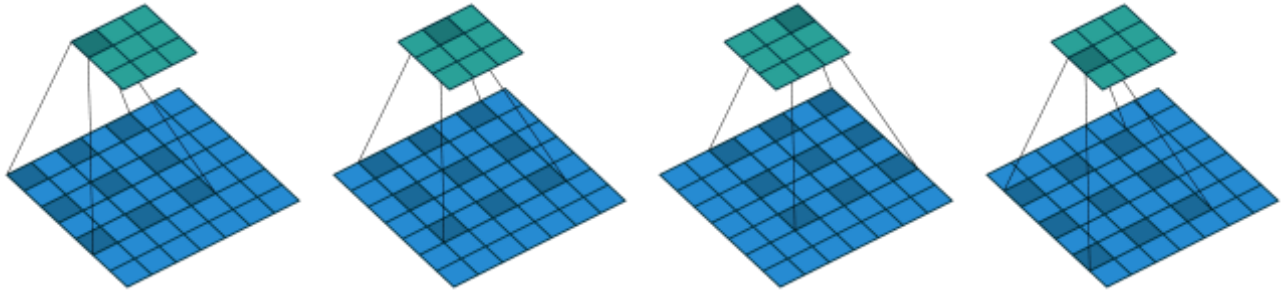
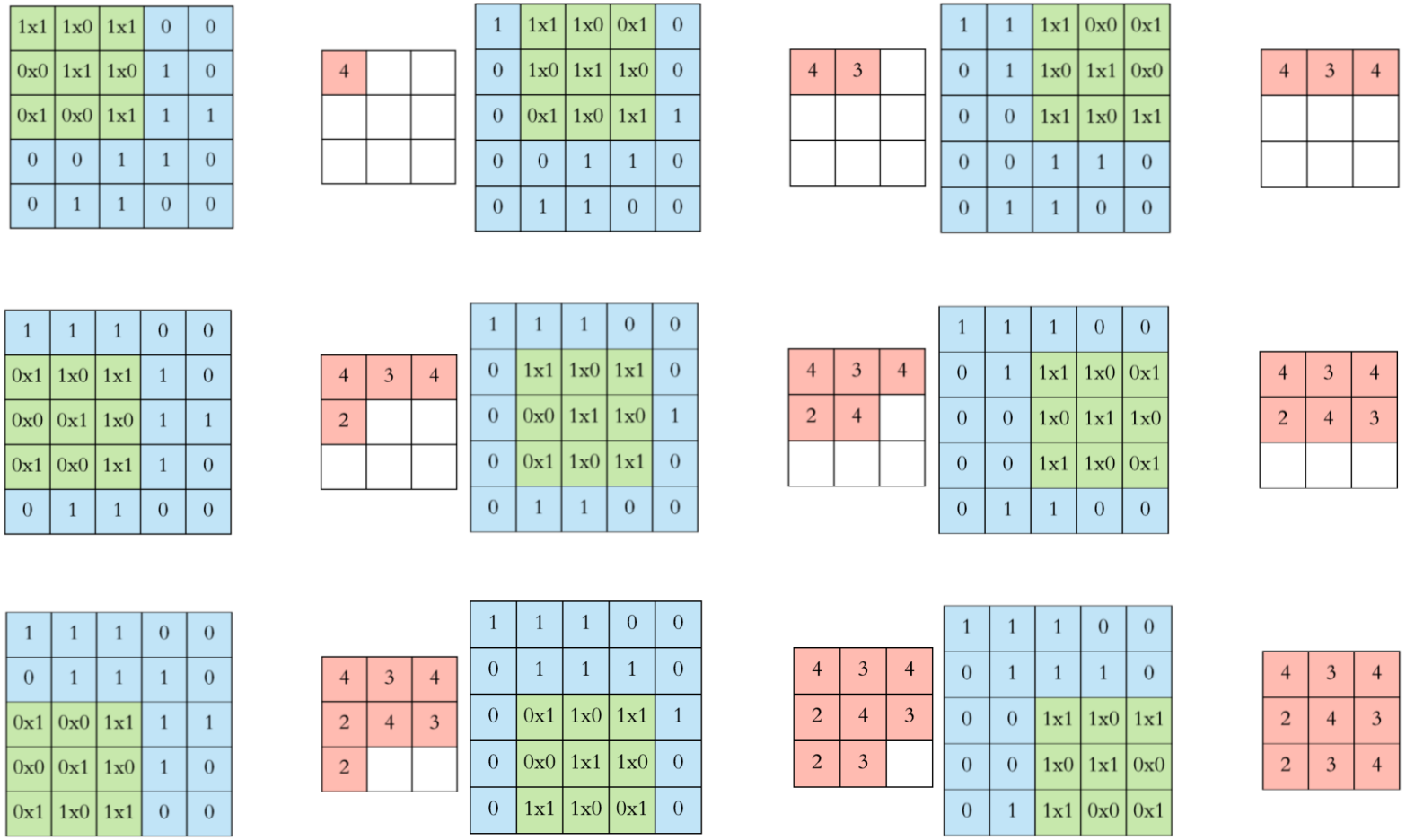
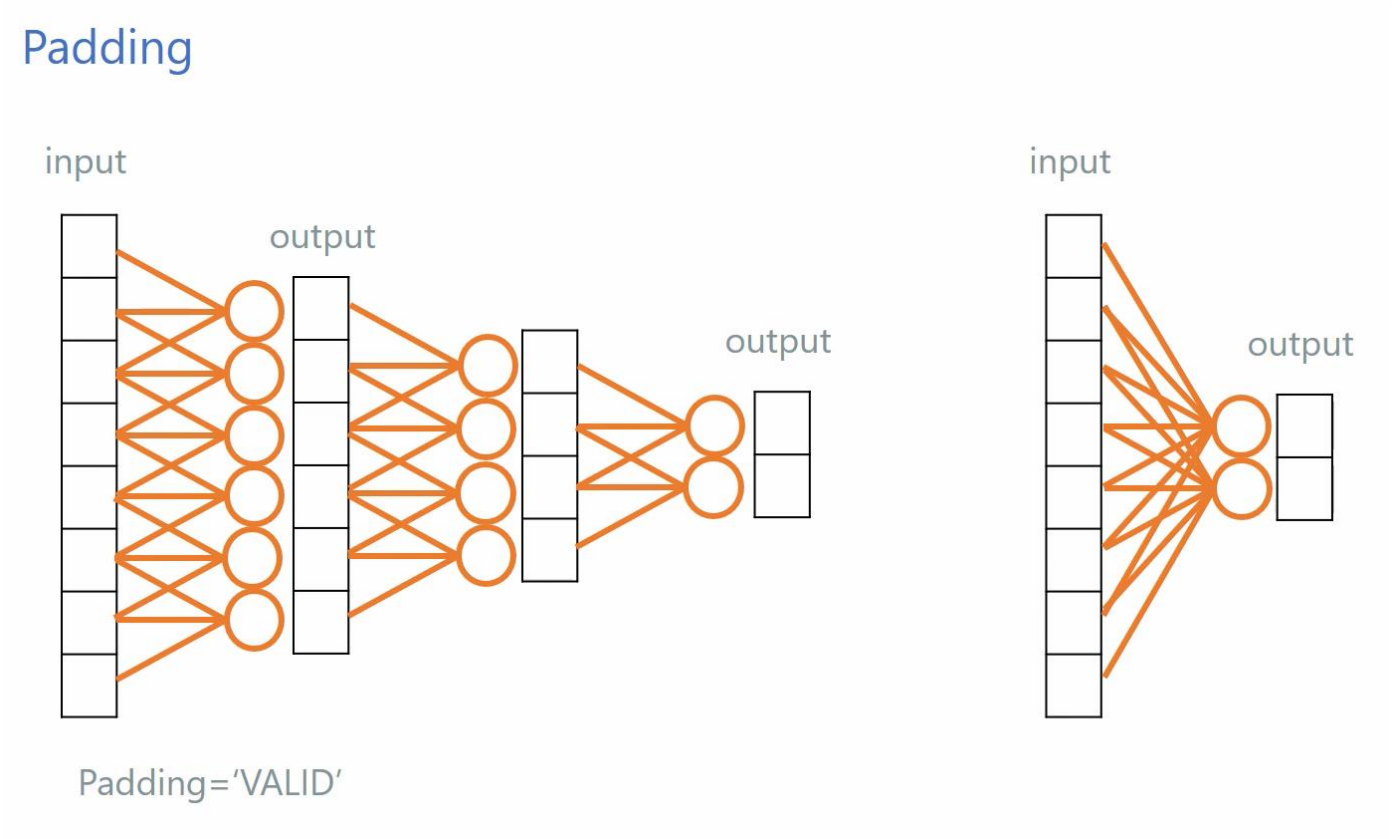


Figura 8 – Campo de receptores (em azul) e filtro numa operação de convolução dilatada (*dilated convolution*)

- A convolução dilatada pode ser usada como uma alternativa às operações de *pooling*.

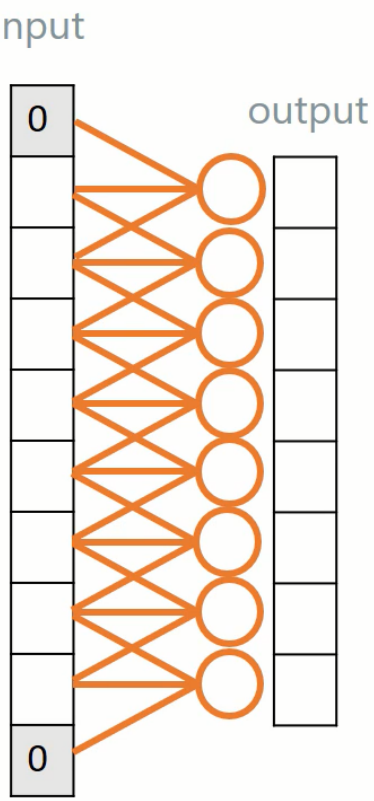
- Segue um exemplo numérico da aplicação de convolução com *stride* = 1 e sem *padding*.

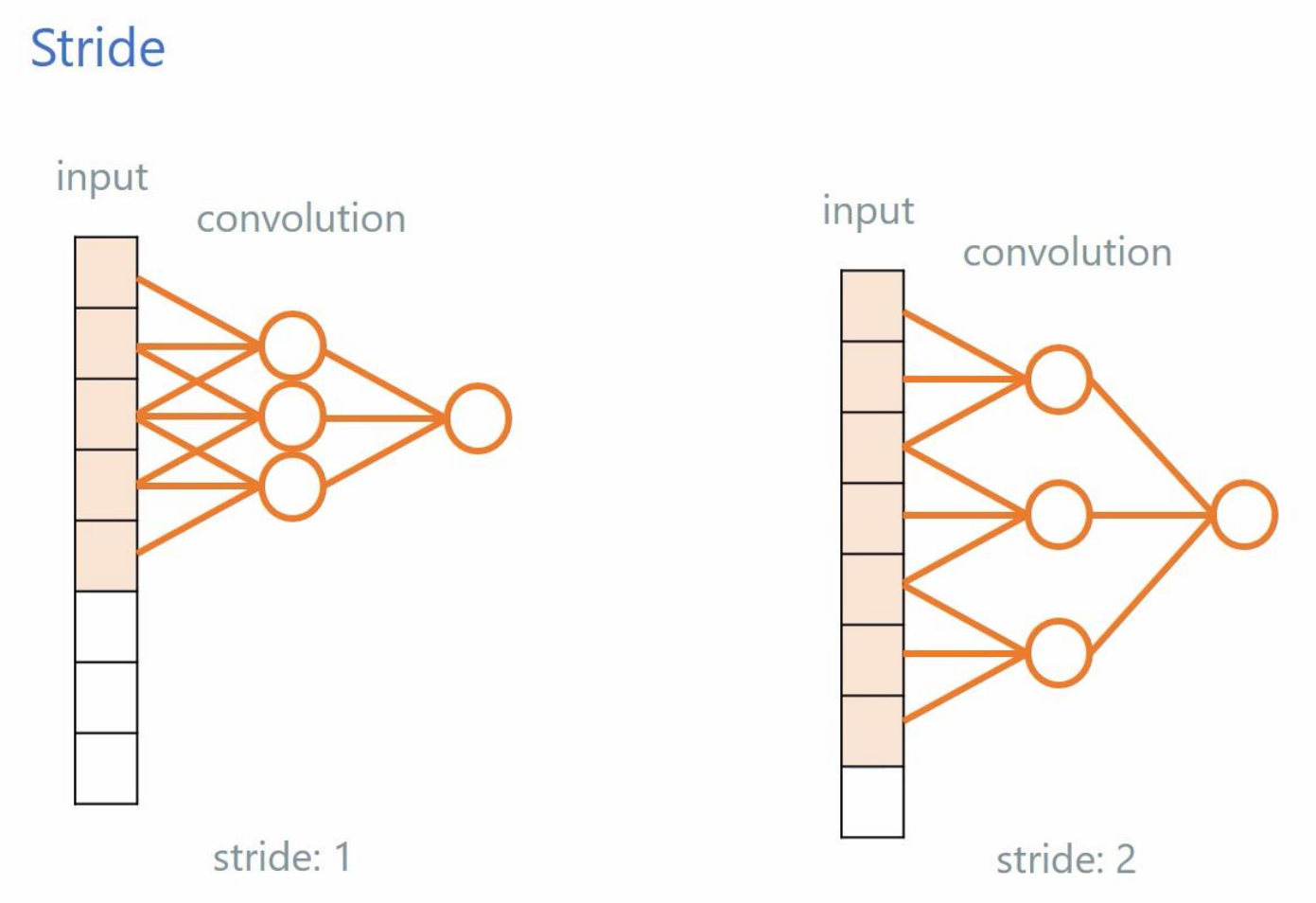



Padding
input
output
output
Padding='VALID'
input
output

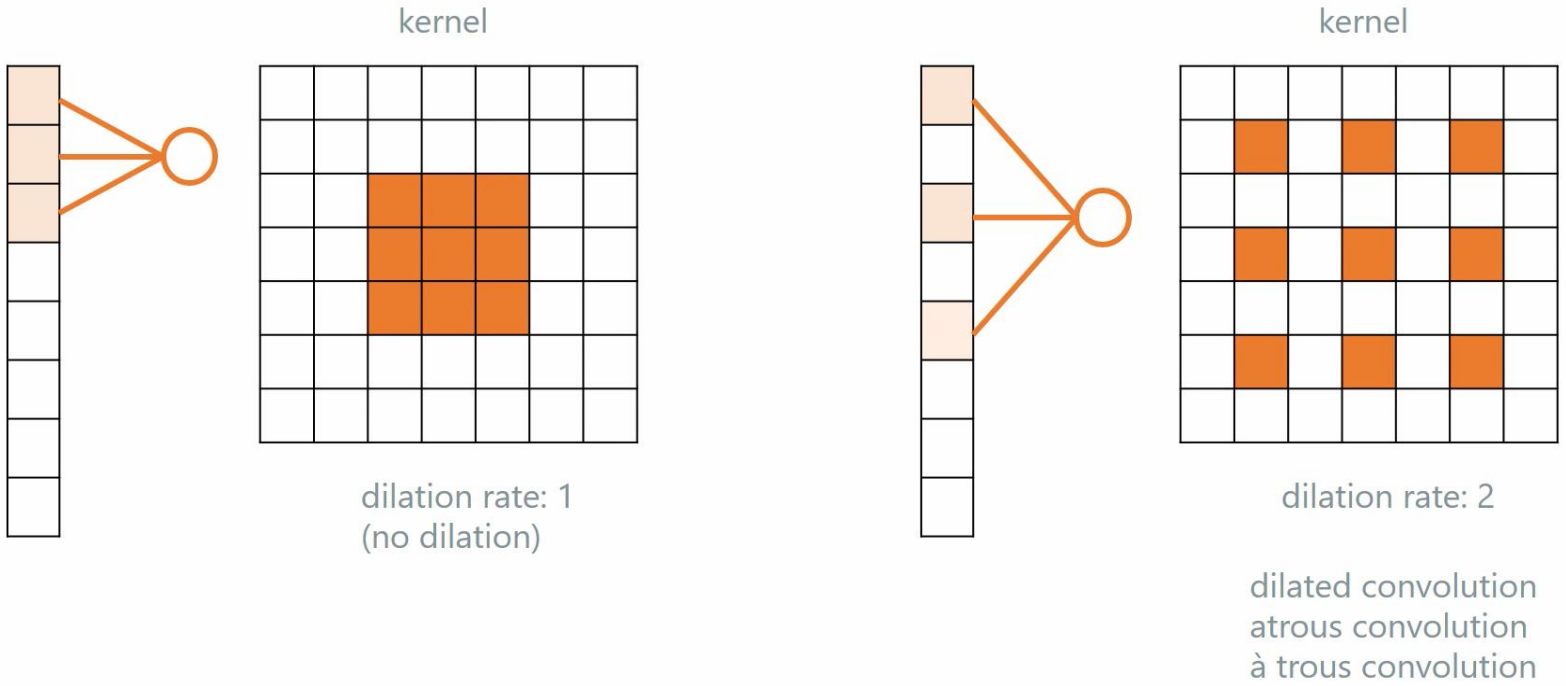
Padding

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	0	0	0	0	0	0	0	0




Stride
input
convolution
stride: 1
input
convolution
stride: 2

Dilation rate



1.3 Filtros extratores de atributos

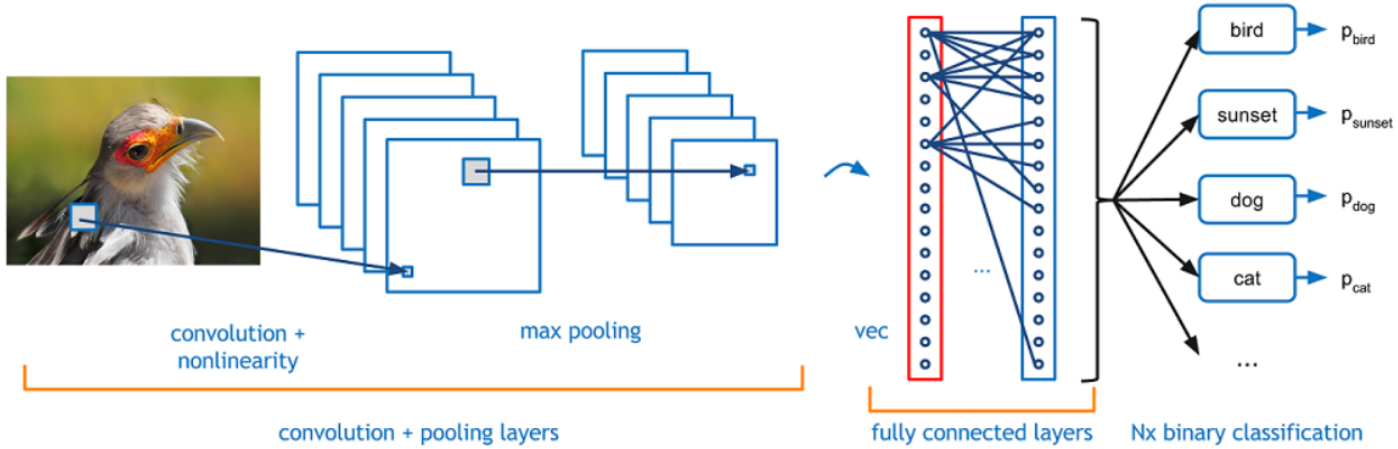


Figura 9 – Rede neural convolucional que recebe a imagem bruta, na forma de pixels.

- Existe um padrão bem característico: múltiplas camadas convolucionais, que operam como uma cascata de filtros não-lineares extratores de atributos relevantes da imagem de entrada, seguidas por camadas *fully connected*, que sintetizam o mapeamento não-linear dos [atributos disponíveis no final da cascata de filtros] para um [vetor de probabilidades], indicando assim a probabilidade da imagem de entrada pertencer a cada uma das classes do problema de classificação.

- Apesar de vigorar um certo padrão na construção das redes neurais convolucionais, as possibilidades de configuração da topologia de toda a rede ainda são enormes. Partir de topologias que já conduziram a bons resultados em aplicações afins e/ou ir inserindo módulos adicionais aos poucos são recomendações válidas.
- Todas as camadas realizam operações não-lineares e o ajuste dos pesos sinápticos deve se dar por técnicas de otimização não-linear, empregando a retropropagação do erro.
- Como já enfatizado em outros momentos do curso, o objetivo do treinamento é explicitamente reduzir a taxa de erro de classificação na saída da rede neural. Entretanto, para conquistar esta redução na taxa de erro de classificação, a rede acaba, indiretamente, concebendo uma cascata de filtros competentes e coerentes entre si, na extração de atributos relevantes das imagens do conjunto de treinamento.

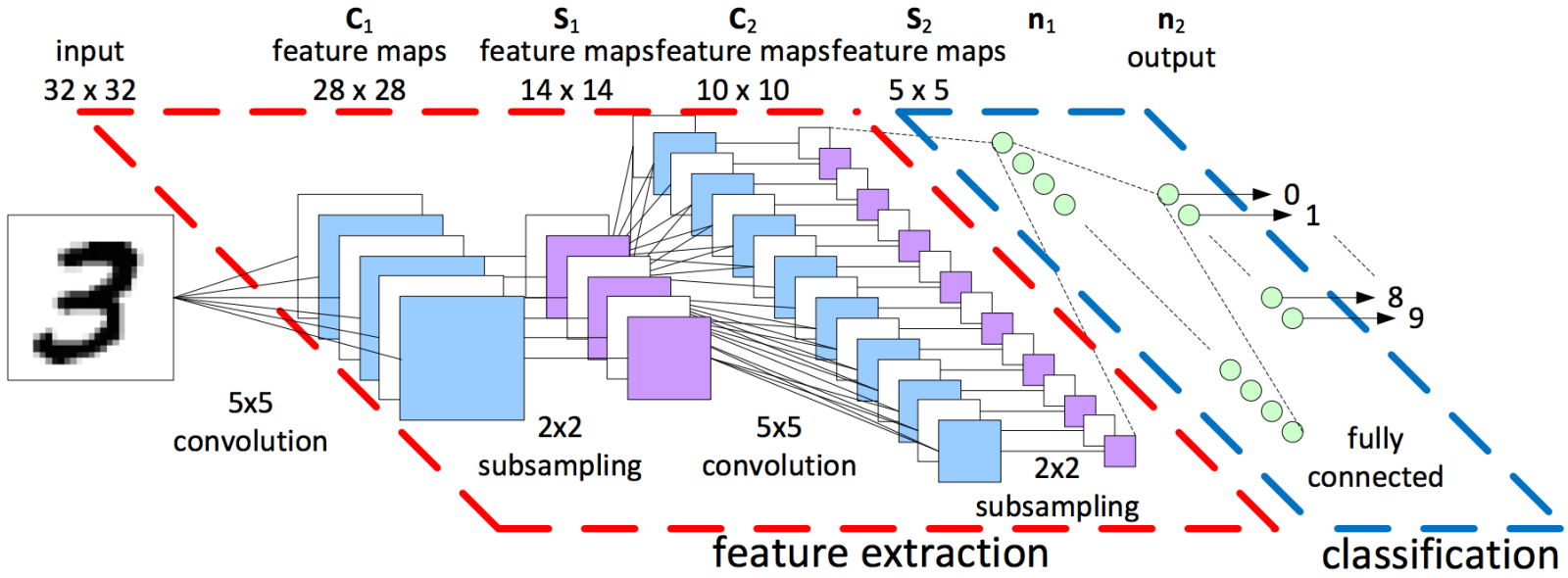
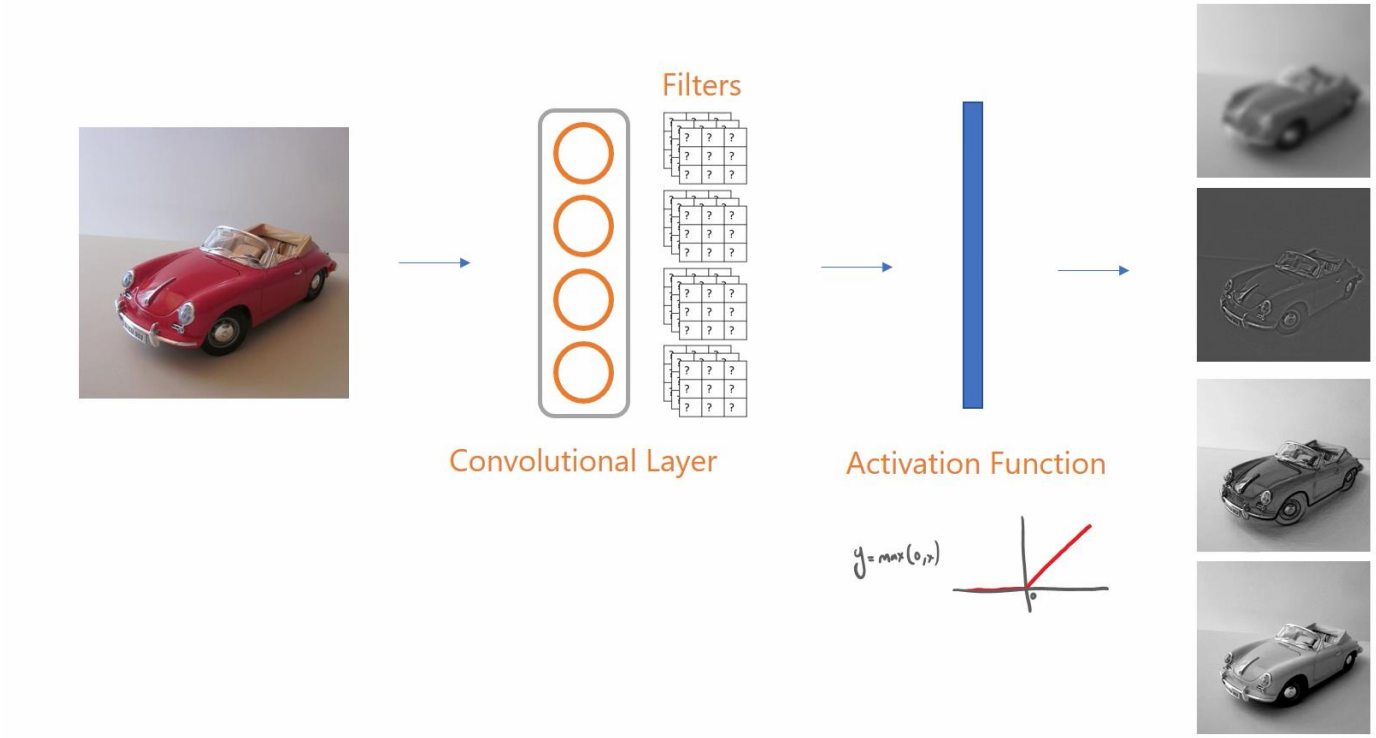
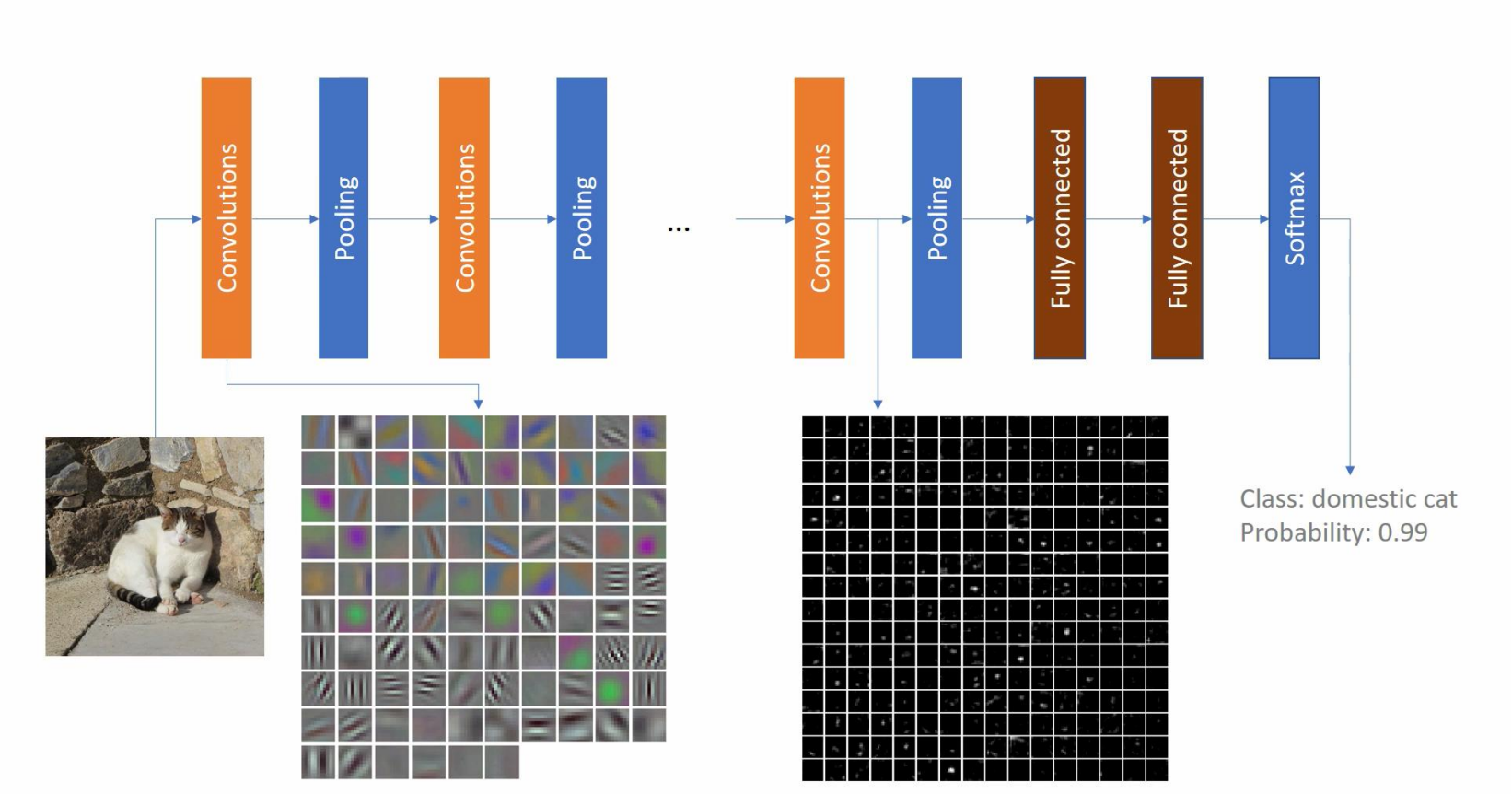


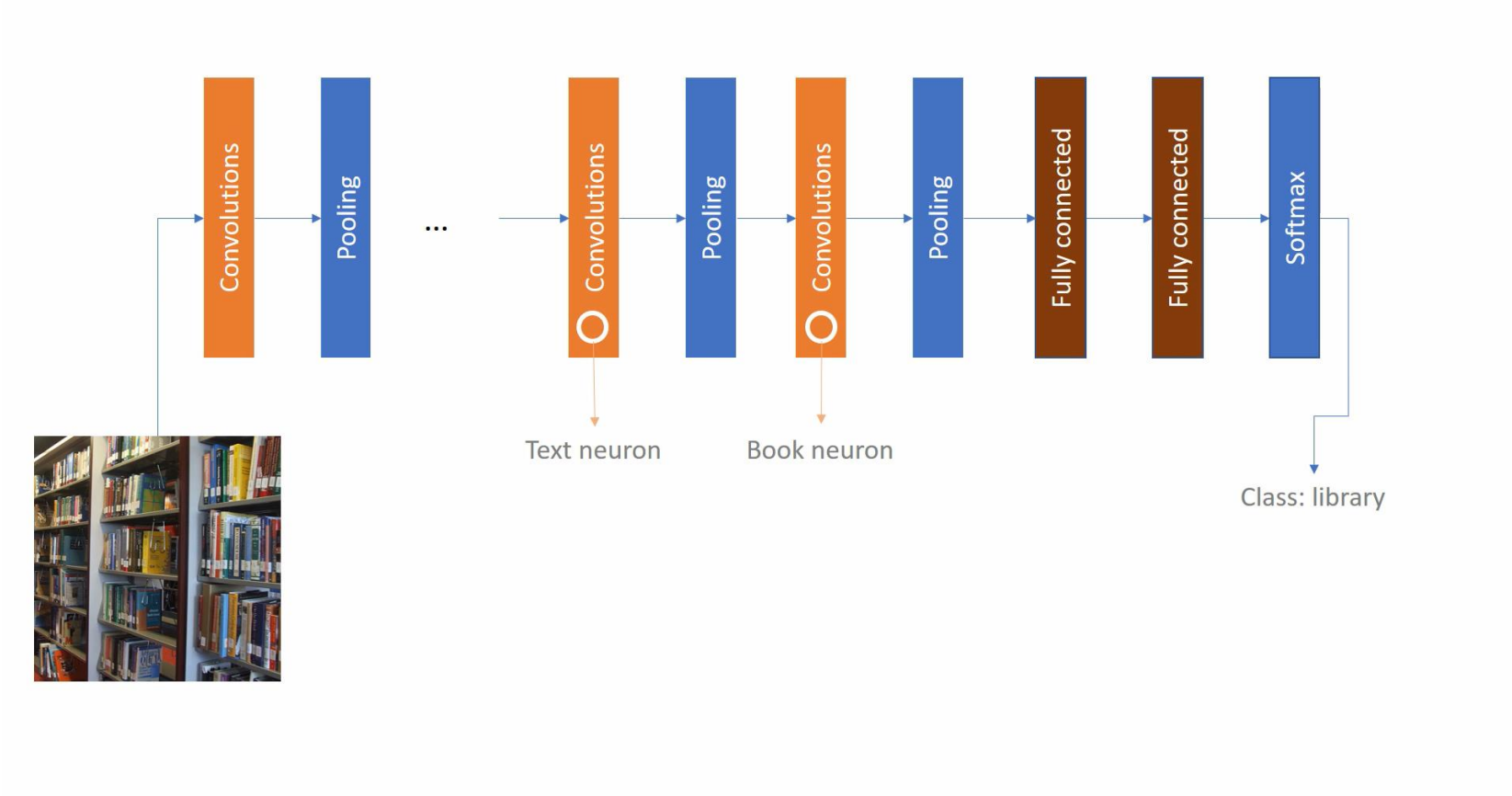
Figura 10 – Exemplo de arquitetura de rede convolucional para classificar imagens de dígitos manuscritos, com resolução 32 × 32, nas dez classes possíveis

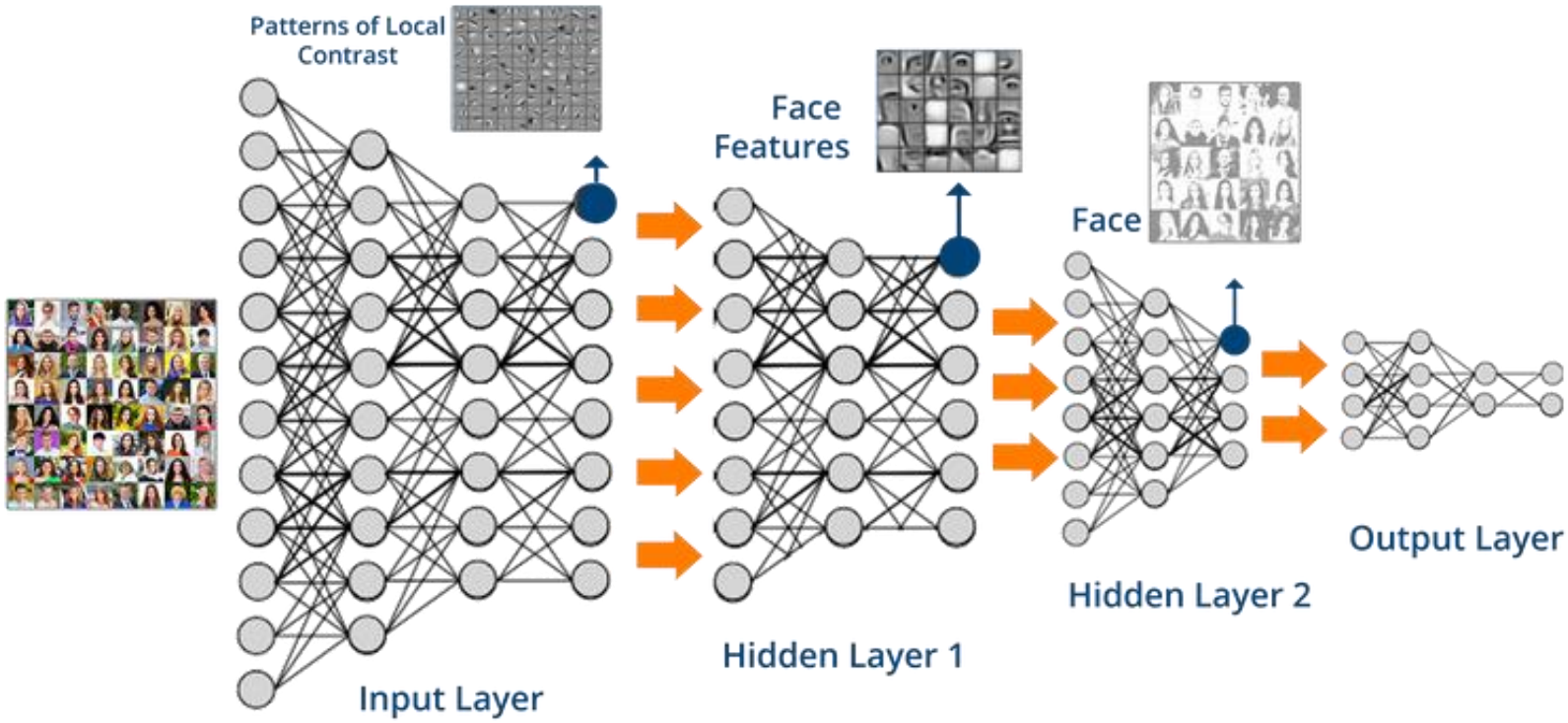
- As “fichas” em paralelo nas camadas convolucionais da figura acima representam o resultado produzido pela operação de convolução que os filtros fazem nas imagens da camada anterior. Uma mesma imagem pode ser convoluída com múltiplos filtros, gerando múltiplas “fichas” na camada seguinte. Essas “fichas” são chamadas de mapas de ativação, um para cada filtro local.

Fonte da Figura 10 acima: Peemen, M.; Mesman, B.; Corporaal, H. “Speed sign detection and recognition by convolutional neural networks, Eindhoven University of Technology, the Netherlands.



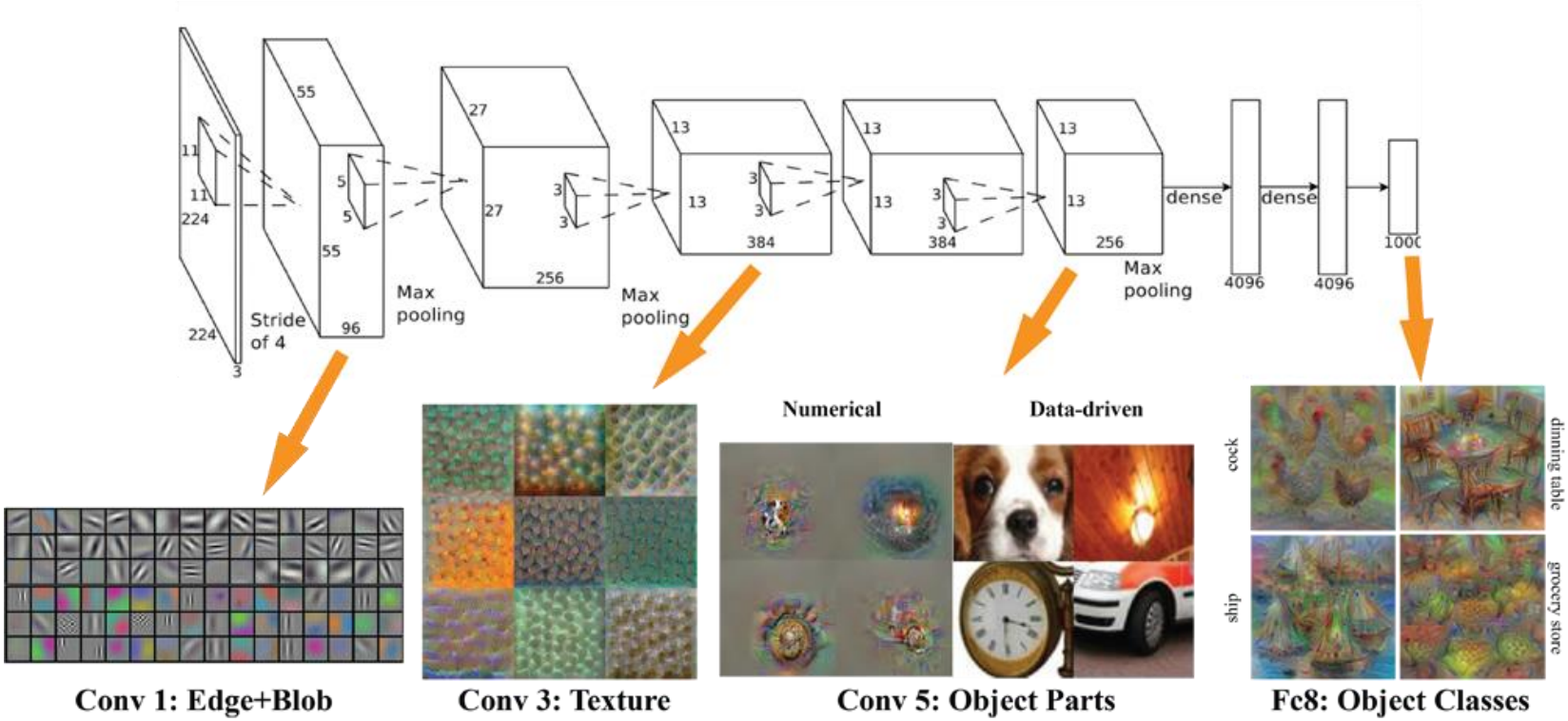

Convolutions
Pooling
Convolutions
Pooling
...
Convolutions
Pooling
Fully connected
Fully connected
Softmax
Class: domestic cat
Probability: 0.99


Convolutions
Pooling
...
Convolutions
Pooling
Convolutions
Pooling
Fully connected
Fully connected
Softmax
Text neuron
Book neuron
Class: library



Fonte: ⟨ https://towardsdatascience.com/creating-a-movie-recommender-using-convolutional-neural-networks-be93e66464a7 ⟩

ZEILER, M.D. & FERGUS, R. "Visualizing and Understanding Convolutional Networks", arXiv:1311.2901v3, 2013.



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

Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture 5 - 62 April 18, 2017

- Fonte: ⟨ http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf ⟩

- Para entradas 3D, é possível aplicar uma convolução 3D, como ilustrado na rede convolucional abaixo.

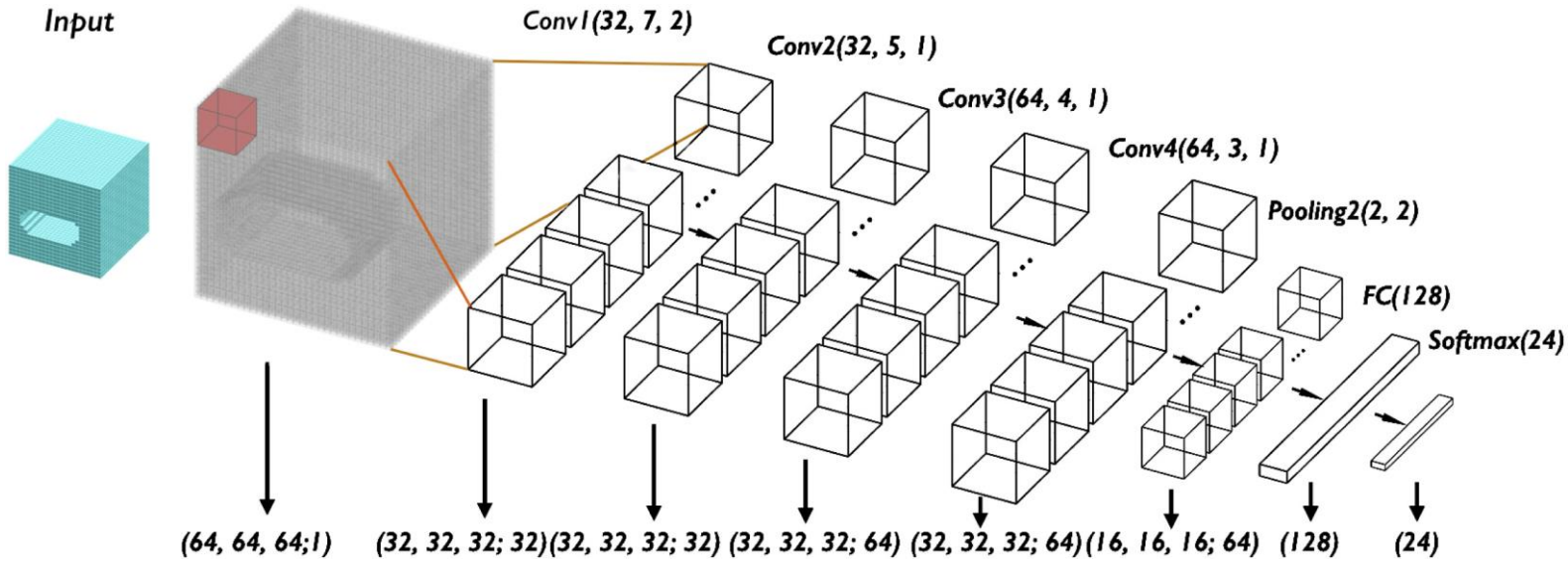
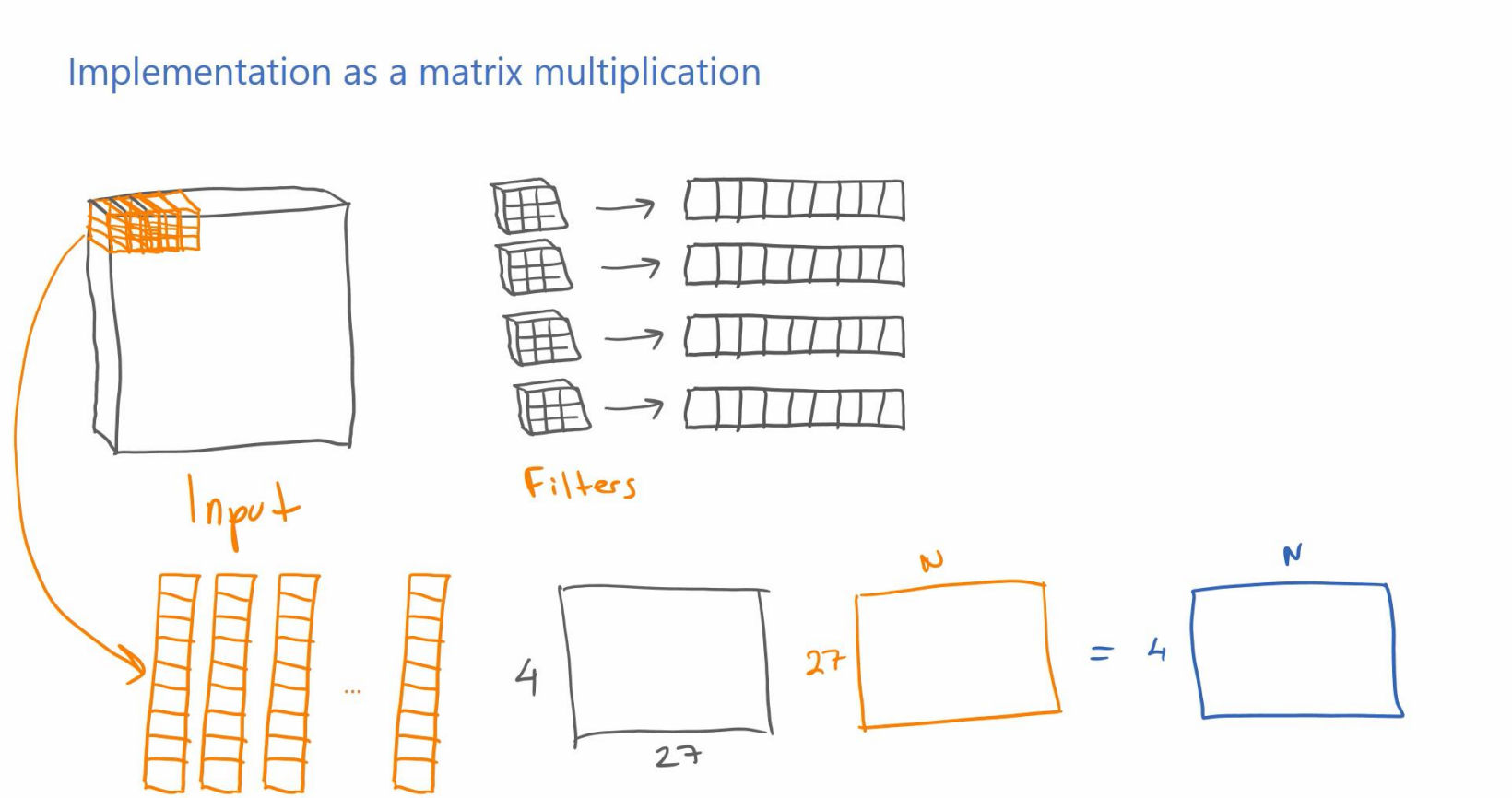


Figura extraída de: Zhang, Z.; Jaiswal, P.; Rai, R. "FeatureNet: Machining feature recognition based on 3D Convolution Neural Network", Computer-Aided Design, vol. 101, pp. 12-22, 2018.

- Voltando ao caso 2D, a convolução pode ser implementada computacionalmente como um produto de matrizes, após uma etapa de pré-processamento ilustrada a seguir.



1.4 Camadas convolucionais × camadas densas

- A primeira motivação para o emprego de camadas convolucionais é a economia de conexões sinápticas, quando comparado com a demanda por conexões sinápticas de uma rede neural de camadas densas (*fully connected*).

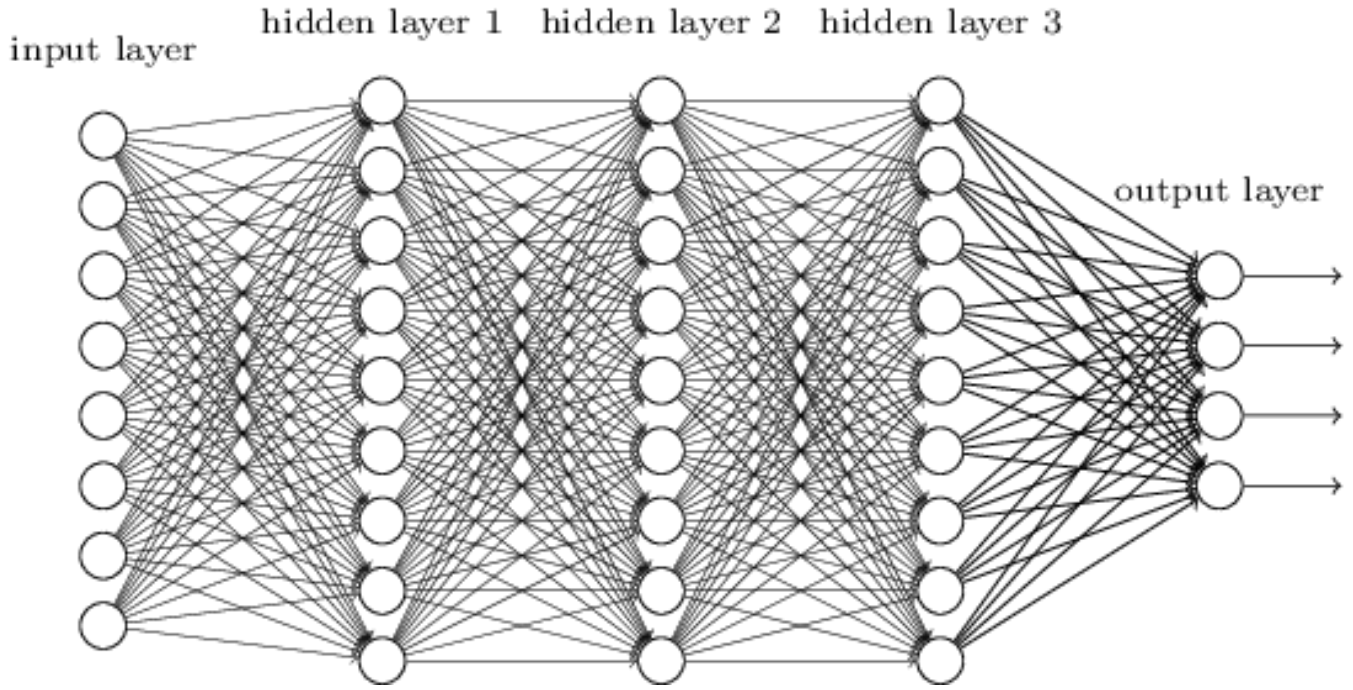


Figura 11 – Exemplo de uma rede neural com todas as camadas *fully connected*. Fonte:

⟨ http://neuralnetworksanddeeplearning.com/chap6.html ⟩

- Múltiplas camadas *fully connected* são também capazes de extrair atributos relevantes de imagens, mas requerem uma quantidade muito elevada de conexões sinápticas e de neurônios. Aqui, cada neurônio é um filtro não-linear que faz *matching* (produto interno) com toda a imagem produzida pela camada anterior.
- Além desse número elevado de conexões sinápticas, como o número de neurônios é finito, existe ao menos um outro grande desafio associado a esses filtros de *matching* global, o qual é ilustrado na figura a seguir.

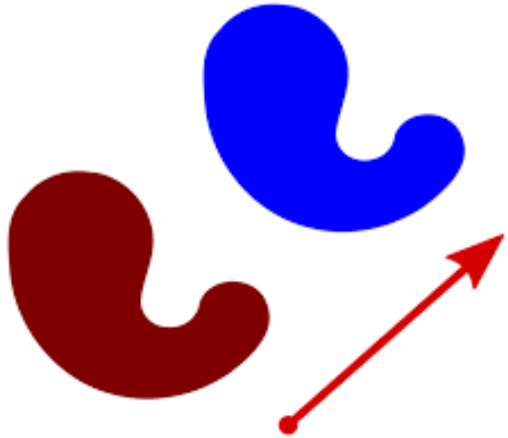
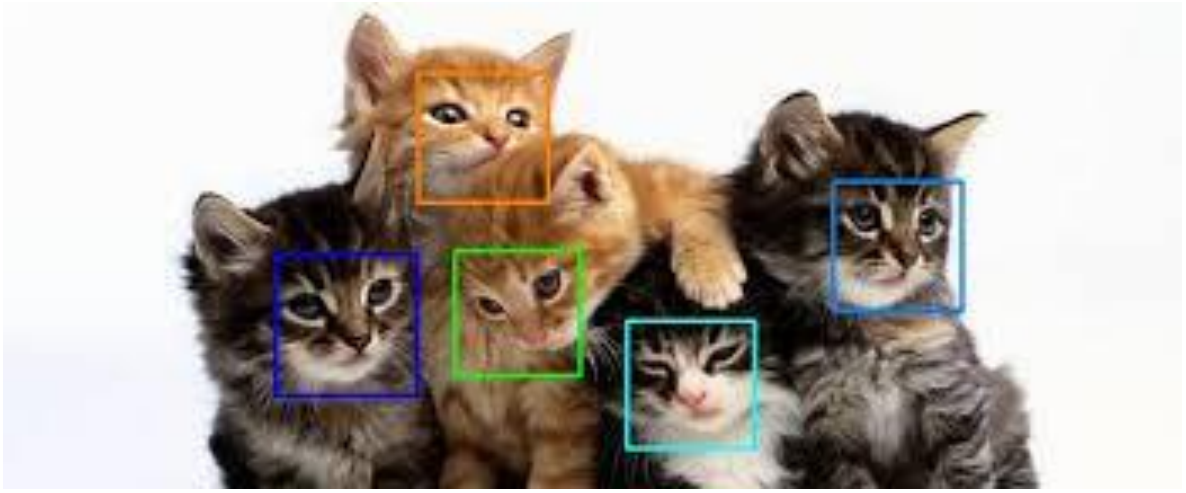


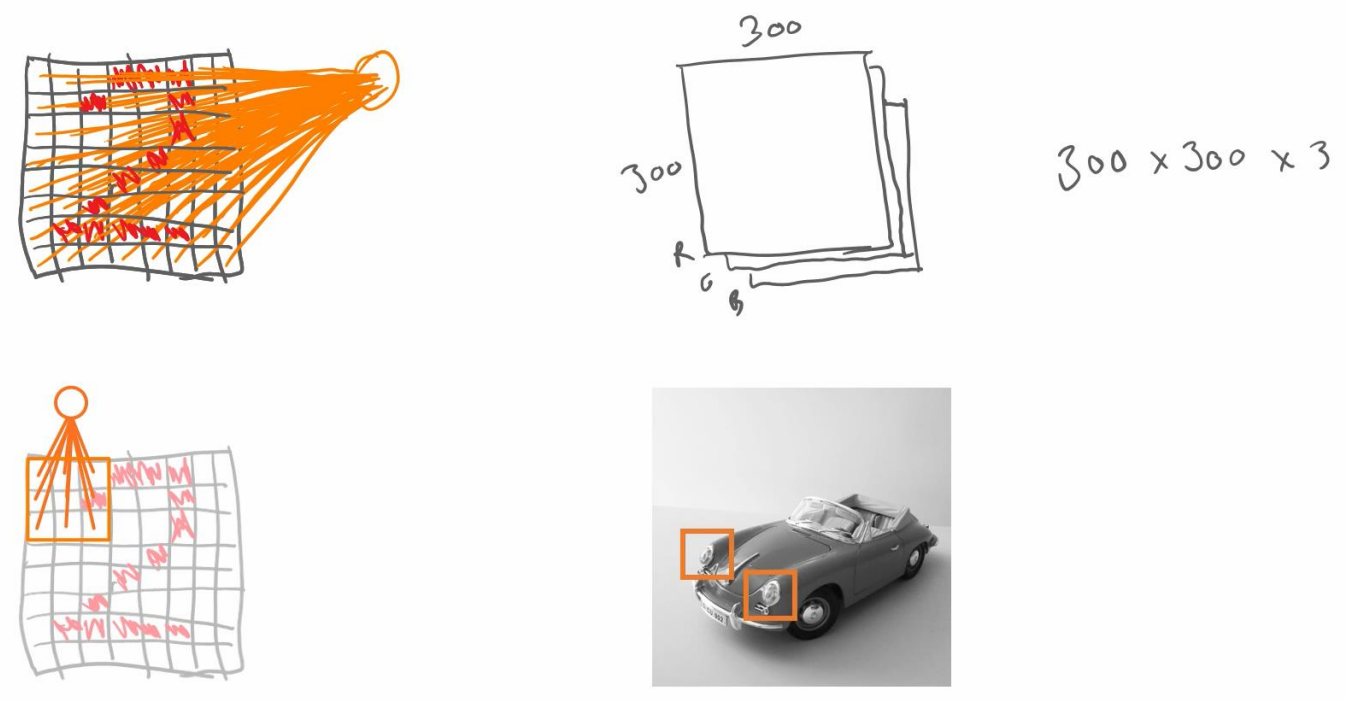
Figura 12 – Translação / Deslocamento de um mesmo objeto numa imagem

- É evidente que existem infinitos deslocamentos possíveis que podem ser aplicados a um mesmo objeto numa imagem. Logo, no caso de filtros globais, para se detectar o referido objeto sempre que ele estiver presente, independente de sua posição relativa na imagem, é necessário um número muito elevado de filtros globais, supondo inclusive a possibilidade de *matching* parcial.
- O uso de convolução e de filtros locais permitem atenuar de forma significativa esses dois problemas detectados, pois requerem um único filtro por objeto a ser detectado e cada filtro local possui um número bem reduzido de conexões sinápticas, quando comparado com filtros globais.
- Neste sentido, quando se comparam diferentes imagens, se os objetos não sofressem rotações e deformações não-lineares, apenas deslocamentos, uma única camada convolucional, com filtros de dimensões apropriadas, seria suficiente.
- Esta questão é abordada na comparação entre os cenários de classificação ilustrados a seguir.



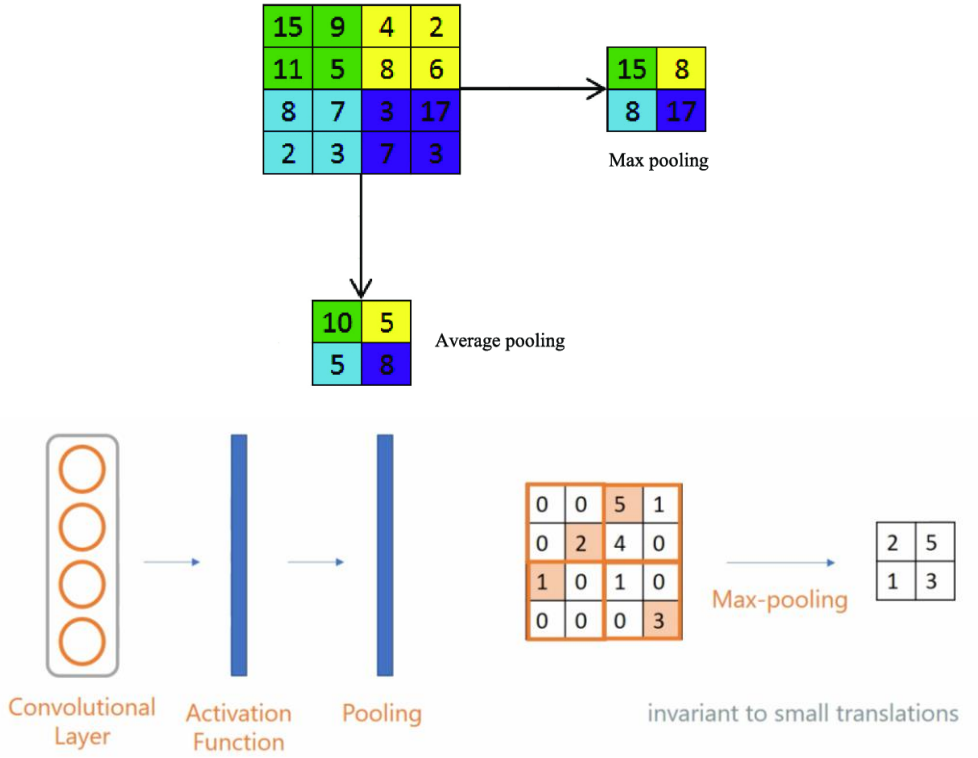
- Você consegue perceber em que diferem as duas tarefas de reconhecimento de padrões (ambas devem detectar gato) ilustradas acima?

- Na presença de transformações afins e deformações, recomenda-se o emprego de múltiplas camadas convolucionais, as quais representam uma cascata de filtros não-lineares locais extratores de atributos, os quais tendem a ser representados de forma hierárquica, ao longo das camadas.


300
300
R
G
B
300 x 300 x 3

1.5 Operadores max pooling e average pooling

- *Pooling* se refere a uma técnica de sub-amostragem após uma operação de convolução sobre uma imagem. Preserva-se apenas a máxima ativação (*max pooling*) ou a ativação média (*average pooling*) do filtro em cada sub-região do mapa de ativação.



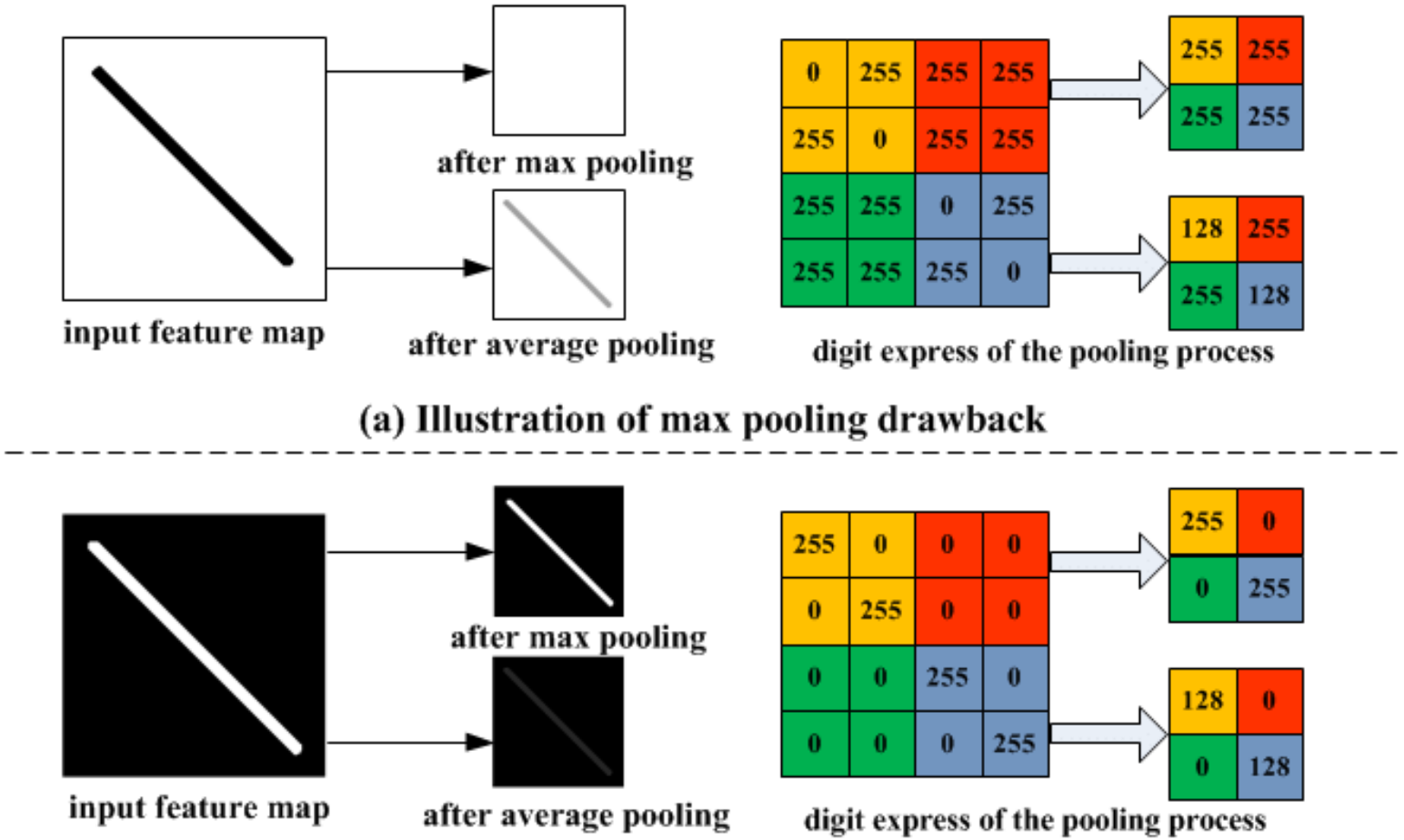
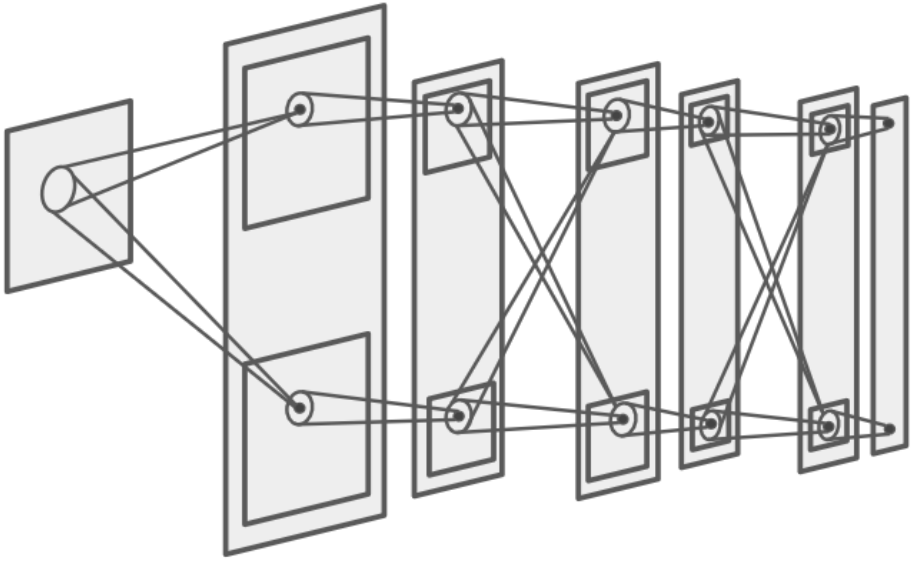


Figura 13 – Perda de informação com *max* e *average pooling*

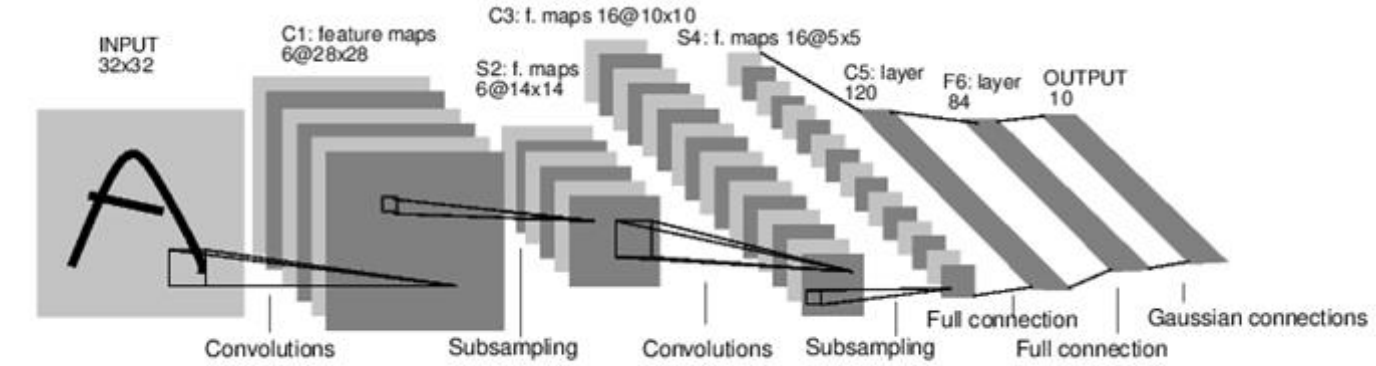
1.6 Redes convolucionais históricas

Neocognitron
[Fukushima 1980]



"sandwich" architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling

- **LeNet**: LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. "Gradient-Based Learning Applied to Document Recognition", Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.



A Full Convolutional Neural Network (LeNet)

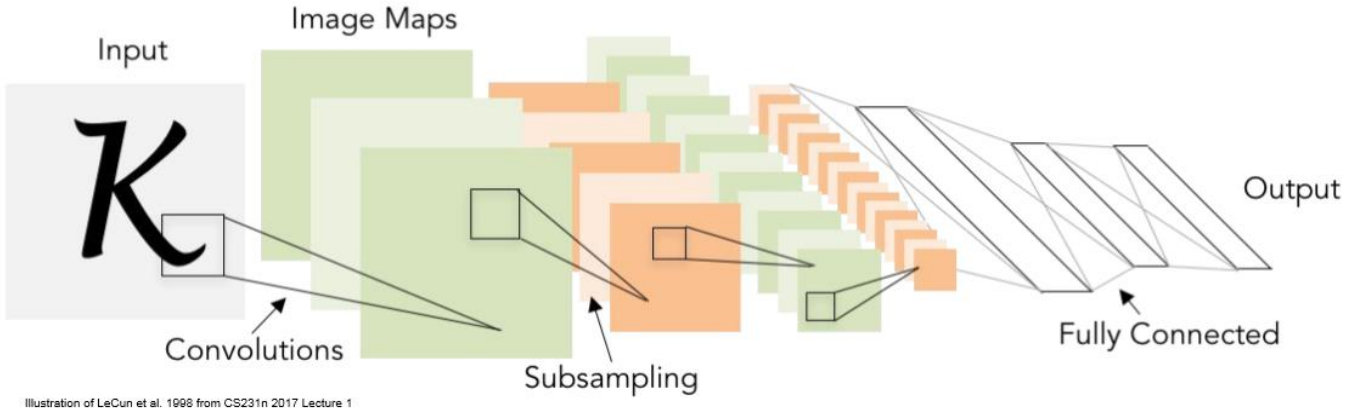


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

- **AlexNet**: KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. "ImageNet Classification with Deep Convolutional Neural Networks", Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'2012), pp. 1097-1105, 2012.
- ILSVRC (ImageNet Large-Scale Visual Recognition Challenge): top-5 classification error rate in ImageNet dataset goes from 26,2% to 15,3%.

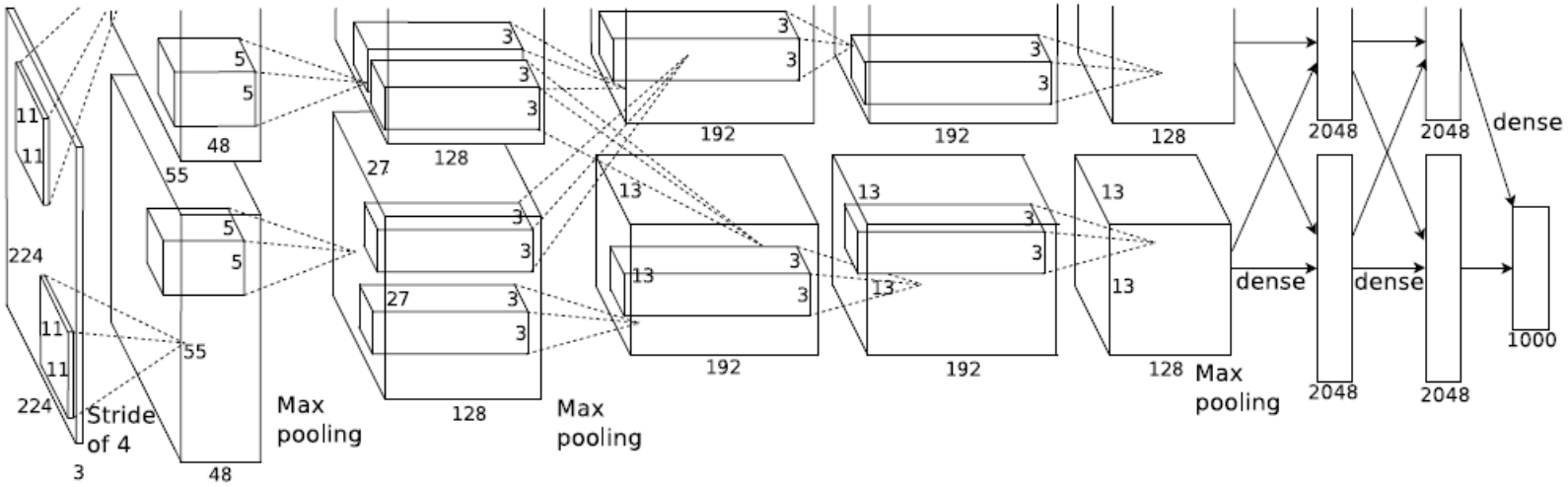


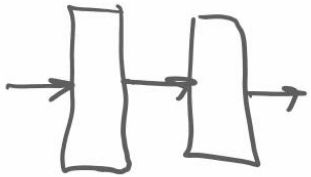
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

- 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax.
- To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective.
- ImageNet consists of over 15 million labeled high-resolution images in over 22,000 categories.
- DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. "ImageNet: A Large-Scale Hierarchical Image Database", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09), 2009.

1.7 Como projetar redes convolucionais?

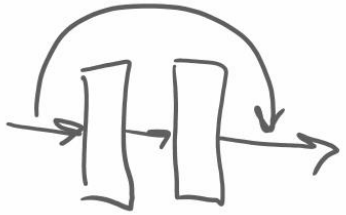
No human intervention in deep learning?

- humans are still in the loop!



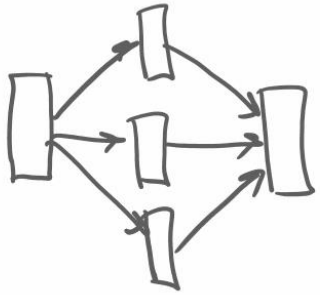
Grid search on all hyperparameters?

- too many hyperparameters
- infinitely many ways to design a network



Designing a deep neural network involves

- human expertise
- trial and error



Building blocks of CNNs

- Convolution
- Pooling

Hyperparameters

- Kernel size, stride, dilation rate
- Number of filters, number of layers
- Pooling size & type
- Padding type
- The way we arrange the layers

How to design a ConvNet?

TL;DR

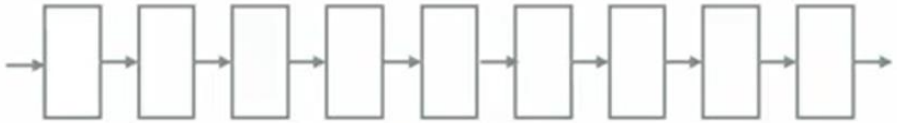
- you don't design
- pick something that works and use it

Working on novel problems?

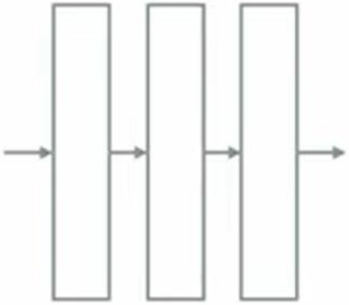
- borrow ideas from successful models
- design your own model

How do choose the number of layers and units?

- start small
- gradually increase model size
- smallest model: linear regression
- deeper:



- wider:



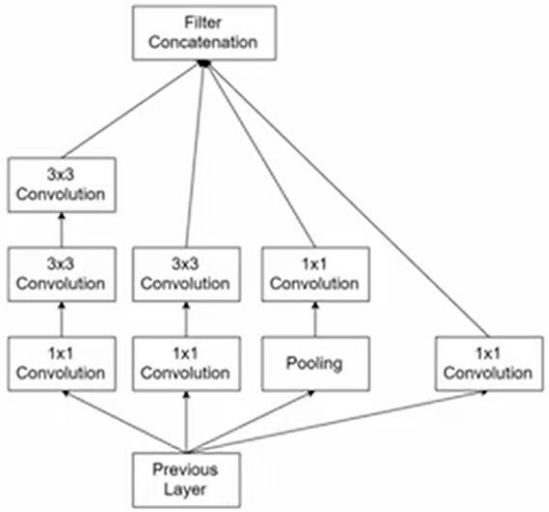
How to choose kernel size?

- 3x3 and 1x1 kernels usually work the best

Pointwise (1x1) filters

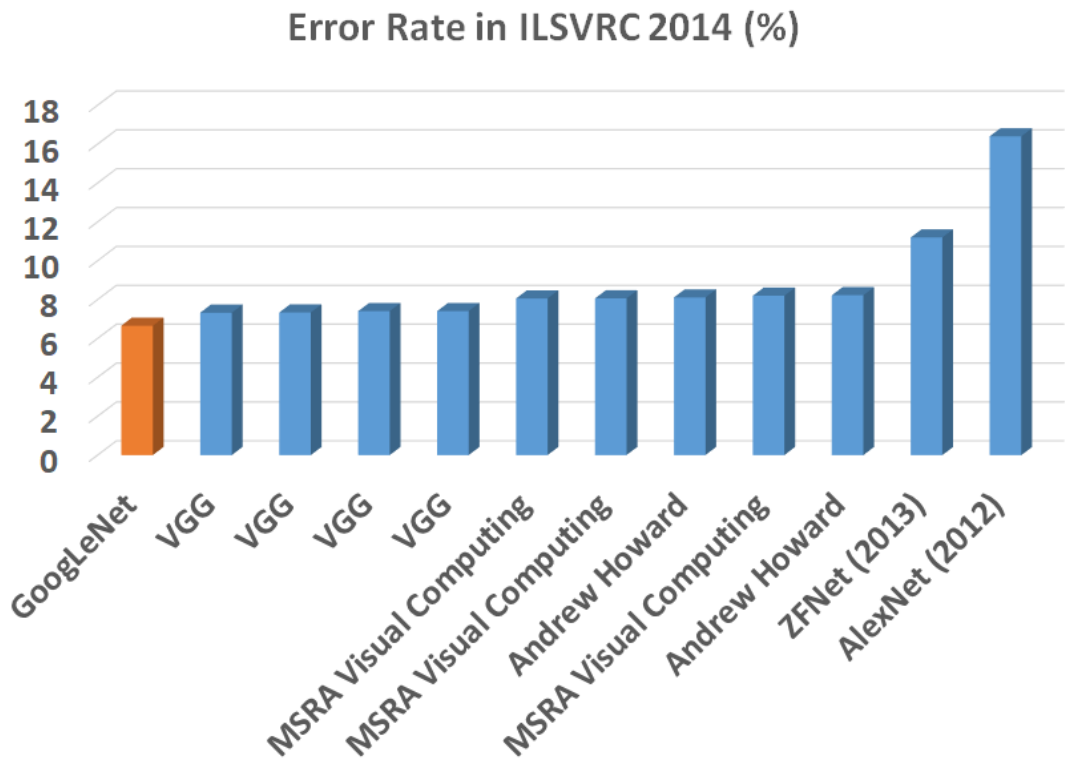
- channel-wise dense layers
- learn cross-channel features

Inception module



- Inception module: GoogLeNet.

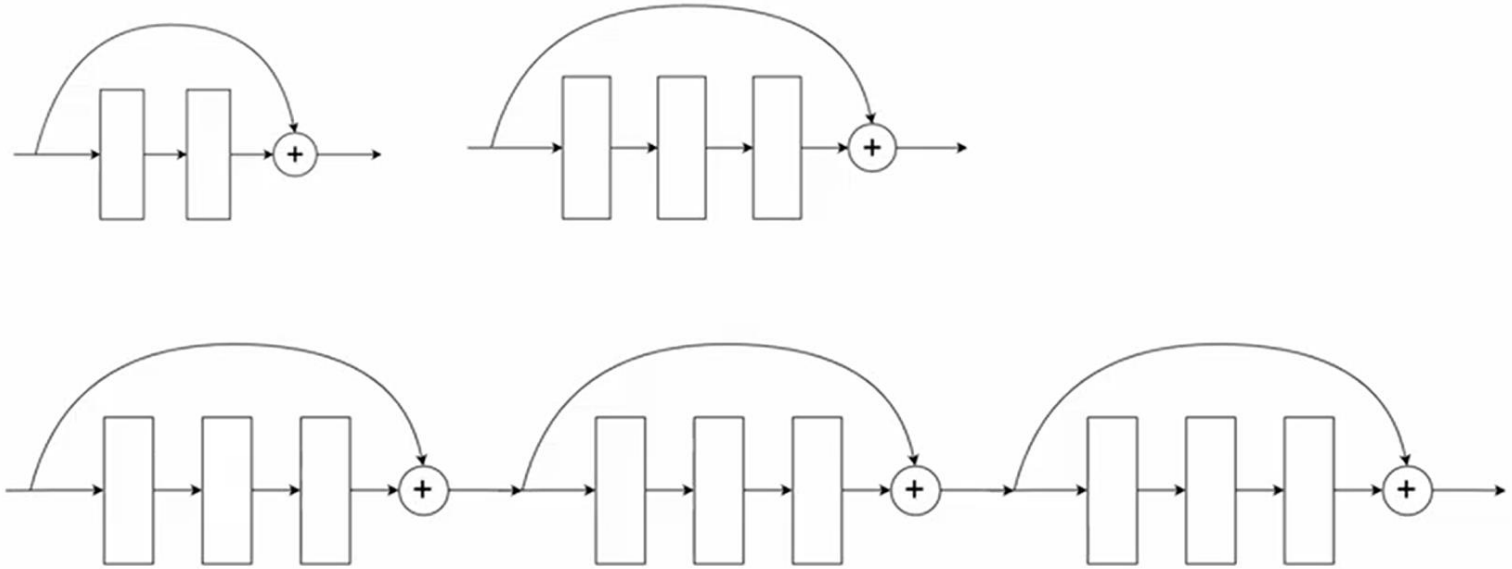
GoogLeNet: Winner of ILSVRC 2014



ImageNet, is a dataset of over 15 millions labeled high-resolution images with around 22,000 categories. ILSVRC uses a subset of ImageNet of around 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images and 100,000 testing images. Fonte: ⟨ https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7 ⟩

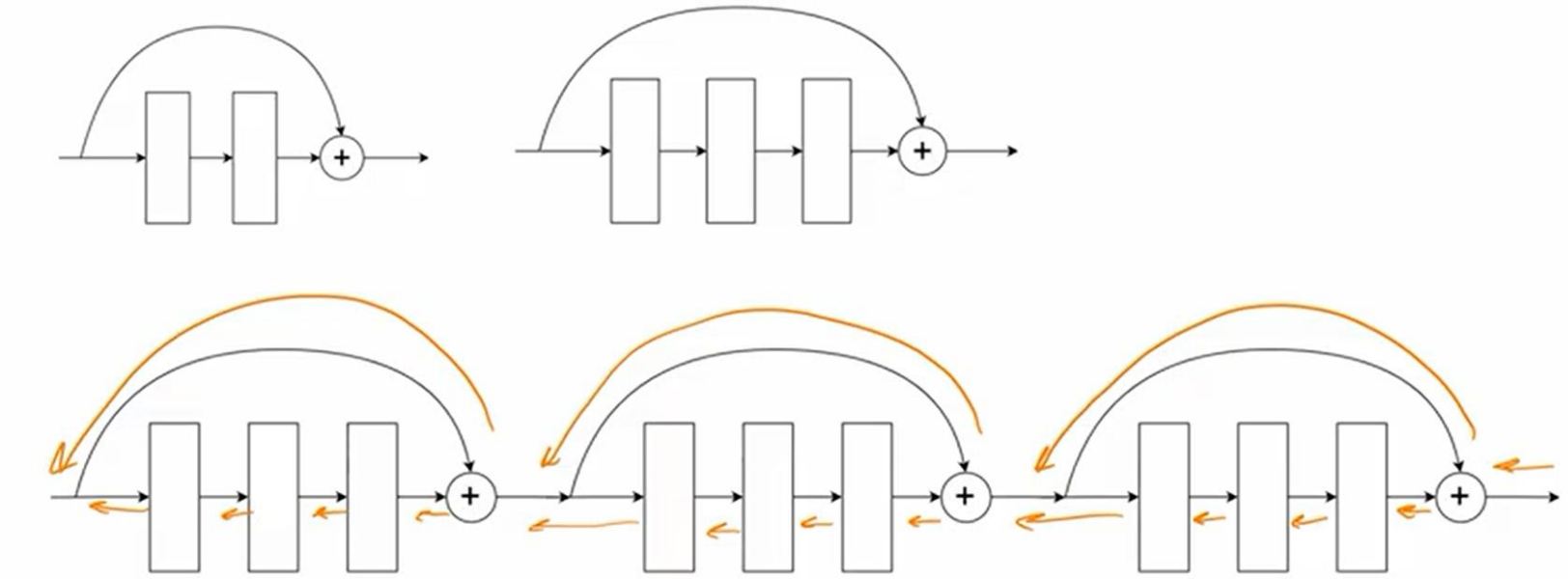
Skip connections

- building blocks of the ResNet architecture



Skip connections

- building blocks of the ResNet architecture



method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

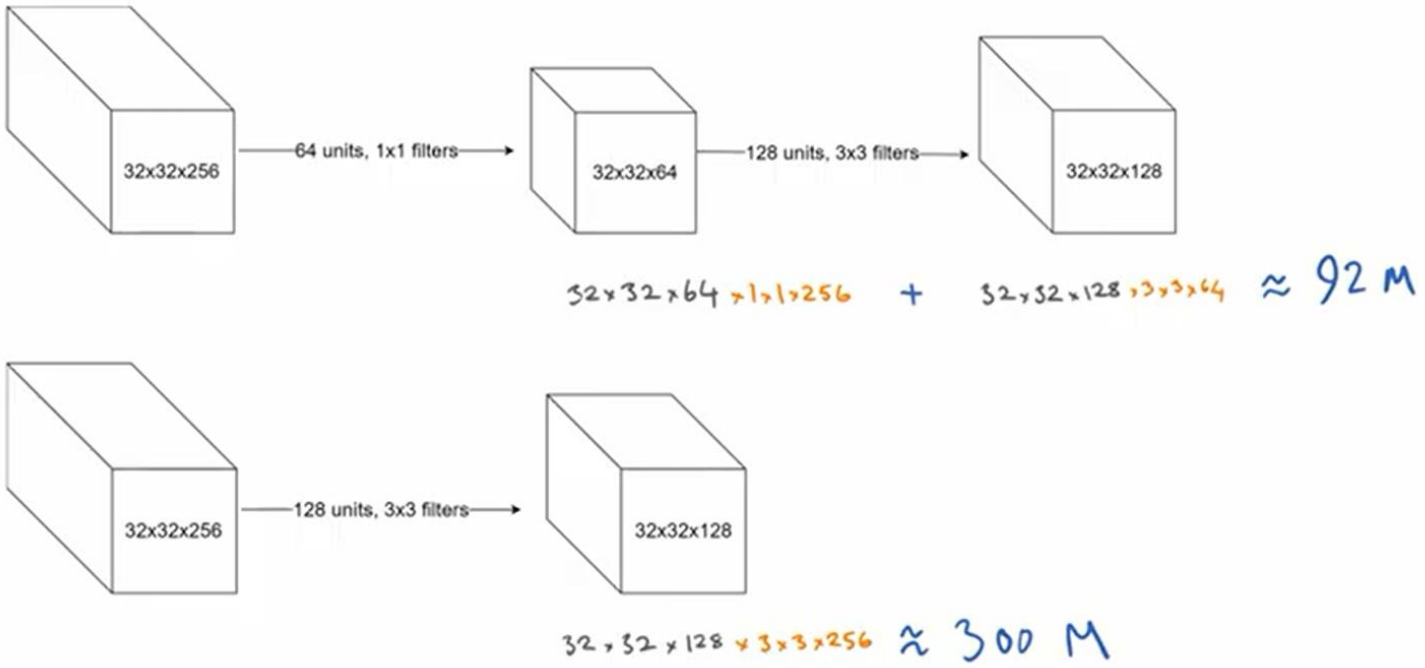
method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

HE, K., ZHANG, X., REN, S. & SUN, J. "Deep Residual Learning for Image Recognition", arXiv:1512.03385v1, 2015.

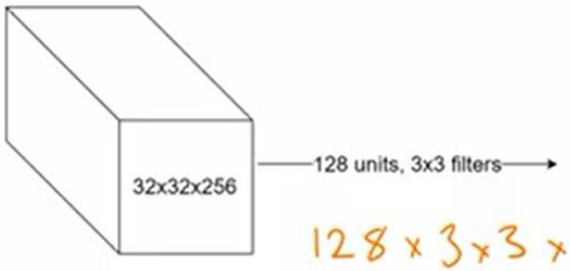
Pointwise (1x1) filters

- Dimensionality reduction

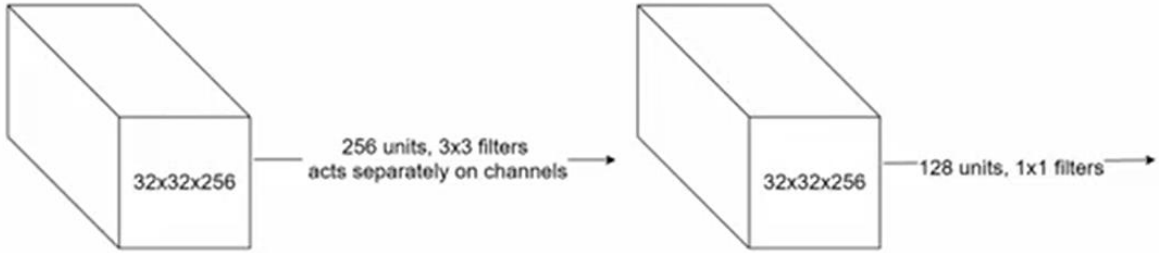


Pointwise (1x1) filters

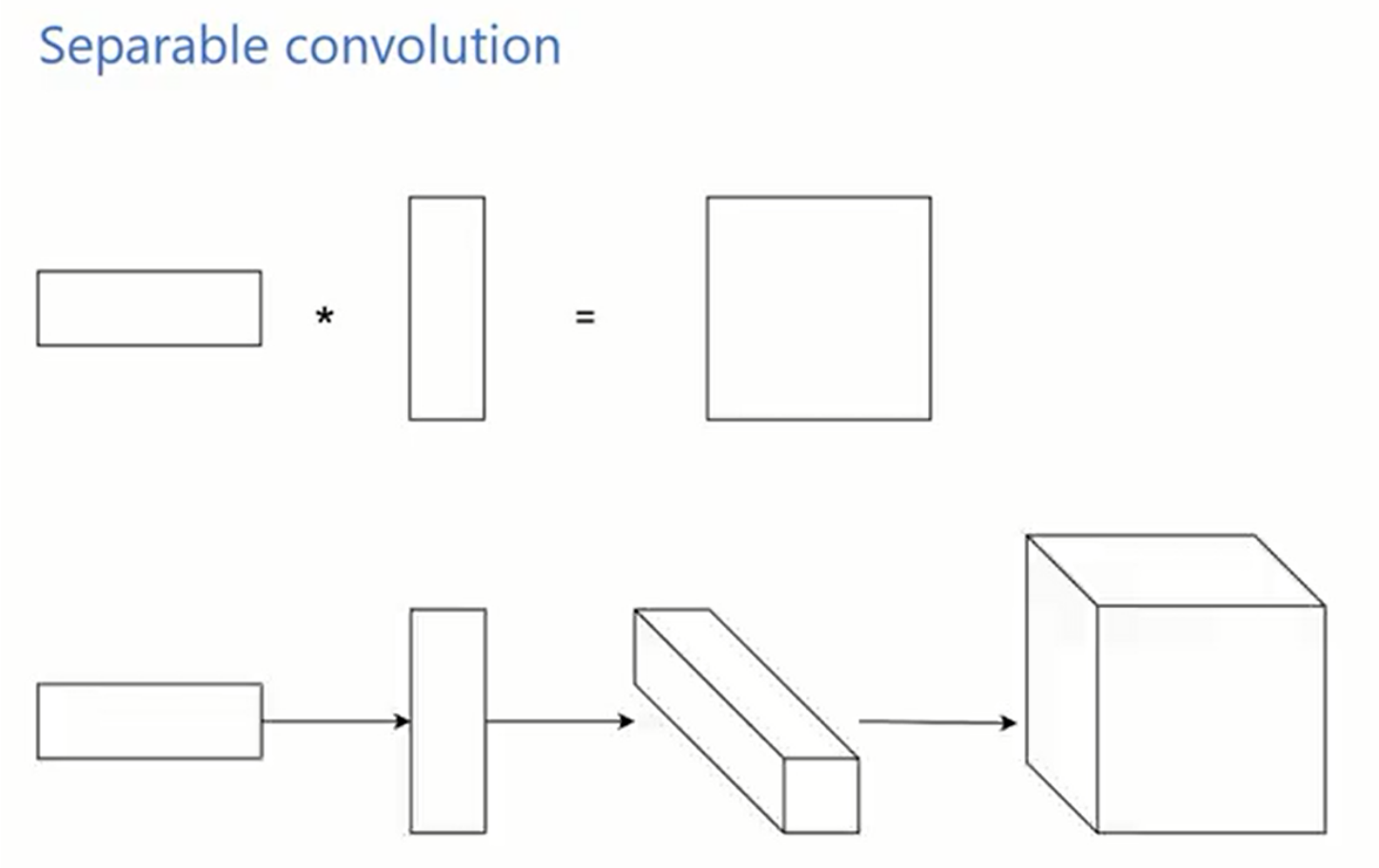
- Depthwise separable convolution



$128 \times 3 \times 3 \times 256 = 294,912$



$256 \times 3 \times 3 \times 1 \quad + \quad 128 \times 1 \times 1 \times 256 \quad = \quad 35,072$


Separable convolution
*
=

How to choose the batch size?

- Image recognition tasks
  - batch size: 32
- Noisy gradient?
  - use larger batches
- Stuck in local minima? Out of memory?
  - use smaller batches

2 Dropout

- Dada a enorme flexibilidade associada aos mapeamentos de entrada-saída que podem ser sintetizados em redes neurais com arquiteturas profundas, o problema de *overfitting* é muito acentuado e precisa ser combatido com técnicas eficazes para a promoção de regularização.
- Sabe-se que a combinação de modelos, como realizado em comitê de máquinas, tende a melhorar o desempenho de máquinas de aprendizado, particularmente quando os modelos divergem no erro (PERRONE & COOPER, 1992). Duas formas alternativas de promover diferença de comportamento é propor modelos com arquiteturas distintas ou treinar com dados distintos.
- De fato, se houvesse recursos computacionais ilimitados, a melhor forma de regularizar um modelo de tamanho fixo é tirar a média de todas as possíveis atribuições de parâmetros, ponderando cada atribuição por sua probabilidade a posteriori, extraída dos dados de treinamento (BALDI & SADOWSKI, 2013).

- Particularmente no contexto de *deep learning*, a síntese independente de múltiplos modelos de aprendizado, fixos ou variáveis, a serem adequadamente combinados, é proibitiva. Além disso, se o modelo não for fixo, encontrar um conjunto adequado de hiper-parâmetros para cada proposta de arquitetura é uma tarefa custosa e desafiadora. Se o modelo for fixo e a diversidade for promovida pelo treinamento com subconjuntos de dados distintos, pode não haver dado suficiente para se evitar o *overfitting* de cada modelo de aprendizado. E mesmo que se obtenha múltiplas redes profundas, colocar todas em operação simultaneamente, para se coletar uma saída de consenso, pode ser proibitivo em termos computacionais.
- Uma técnica capaz de superar todas essas limitações para se combinar vários modelos visando uma boa regularização se dá pela aplicação de *dropout* (SRIVASTAVA et al., 2014), que nada mais é do que a eliminação aleatória de neurônios da rede neural, juntamente com suas conexões sinápticas, durante o

treinamento da rede neural, visando a redução acentuada de um fenômeno denominado de co-adaptação.

- Uma outra interpretação para *dropout* é o treinamento de uma quantidade exponencial de redes neurais mais enxutas que compartilham pesos e que podem ser combinadas de forma bastante eficiente numa rede neural completa, com pesos de menor magnitude.
- A capacidade de regularização de *dropout* é superior ao que se consegue com outras técnicas de regularização já propostas na literatura, como: (1) parar o treinamento mais cedo, ao minimizar o erro junto a um conjunto de validação; (2) penalizar o aumento do módulo do vetor de pesos via norma L_1 (LASSO) ou norma L_2 (*ridge regression*); (3) promover o compartilhamento de pesos.
- A figura a seguir ilustra a aplicação de *dropout* numa rede neural do tipo perceptron de múltiplas camadas. Uma taxa de remoção da ordem de 50% para as unidades escondidas é sugerida como uma escolha adequada, enquanto para as

unidades de entrada ela deve mais baixa, preservando a grande maioria das entradas.

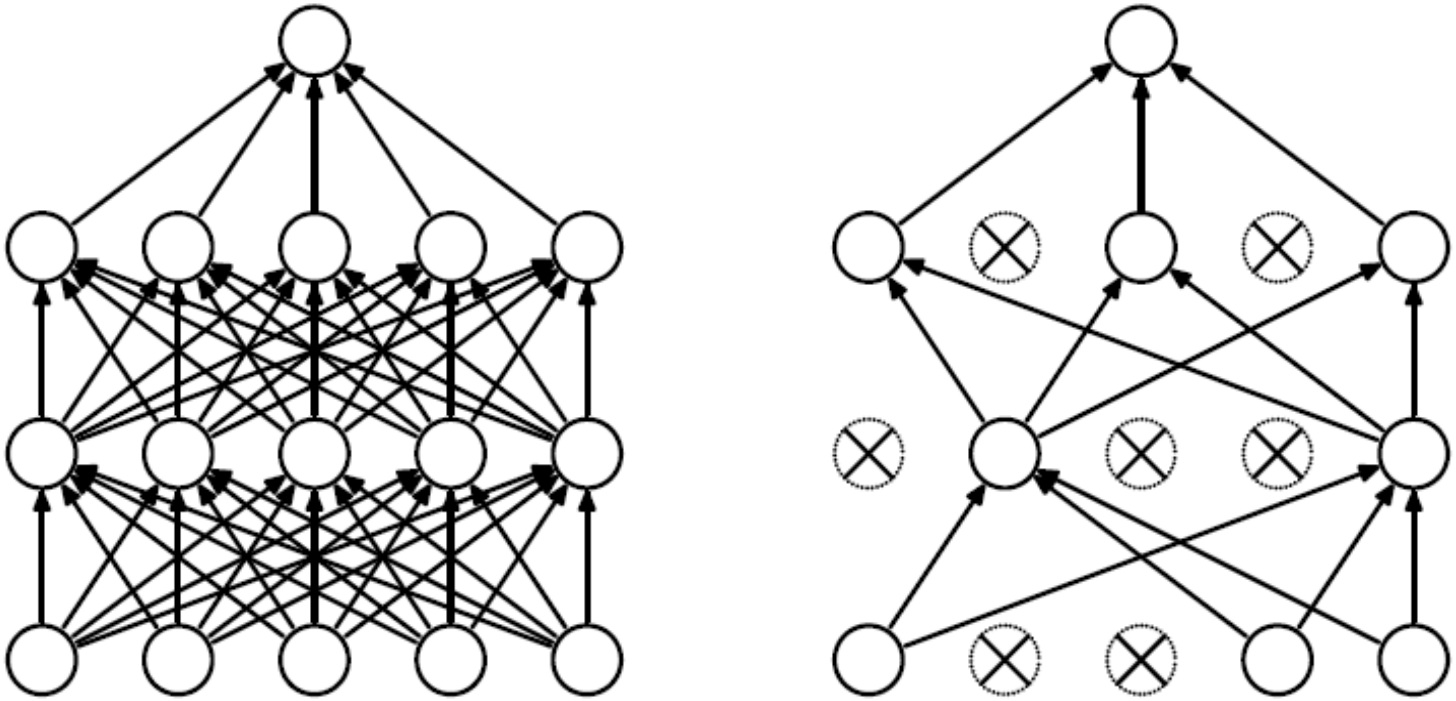


Figura 14 – Remoção temporária de neurônios de uma rede neural multicamadas do tipo *fully connected*. Fonte: SRIVASTAVA et al. (2014).

- Após o treinamento, os pesos produzidos passam a operar com os valores apresentados na figura a seguir.

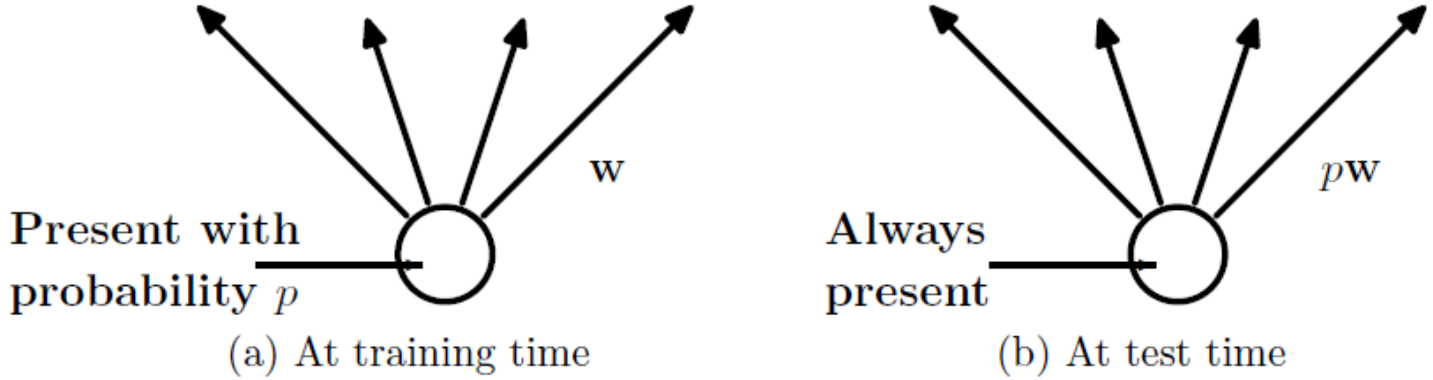


Figura 15 – Pesos durante o treinamento (a) e para a rede em operação, após o treinamento (b).
Fonte: SRIVASTAVA et al. (2014).

- Com esta política de uso, a saída da rede neural após o treinamento vai apresentar o valor médio produzido por um número exponencialmente elevado de redes neurais com remoção de unidades, que poderiam ser produzidas durante o treinamento. De fato, uma rede neural com *n* neurônios pode dar origem a 2^n diferentes arquiteturas que compartilham os mesmos pesos sinápticos (quando não removidos).

- A cada época ou *minibatch* de treinamento, uma rede neural enxuta é amostrada aleatoriamente da rede neural completa. Desse modo, treinar uma rede neural com n unidades aplicando *dropout* pode ser interpretado como o treinamento de 2^n redes neurais mais enxutas e que compartilham pesos sinápticos, com cada uma das possíveis redes neurais mais enxutas sendo treinada muito raramente, se tanto.
- Em fase de operação, após terminado o treinamento, em lugar de se obter a média do comportamento de entrada-saída de 2^n redes neurais mais enxutas, o que seria inviável computacionalmente, aplica-se um método bem simples de aproximação da média, que opera bem em aplicações práticas. Usa-se uma única rede neural sem sofrer *dropout*, com cada peso obtido após o treinamento multiplicado pela probabilidade de inclusão de sua correspondente unidade.
- Em estudos recentes (FRAZIER-LOGUE & HANSON, 2018), *dropout* foi interpretado como um caso especial do algoritmo de treinamento *stochastic delta rule*, proposto em 1990.

3 Comitê de máquinas – Ensemble e Mistura de Especialistas

- Método de aprendizado (supervisionado ou não-supervisionado) cujo objetivo é aumentar a *capacidade de generalização* de modelos de aprendizado.

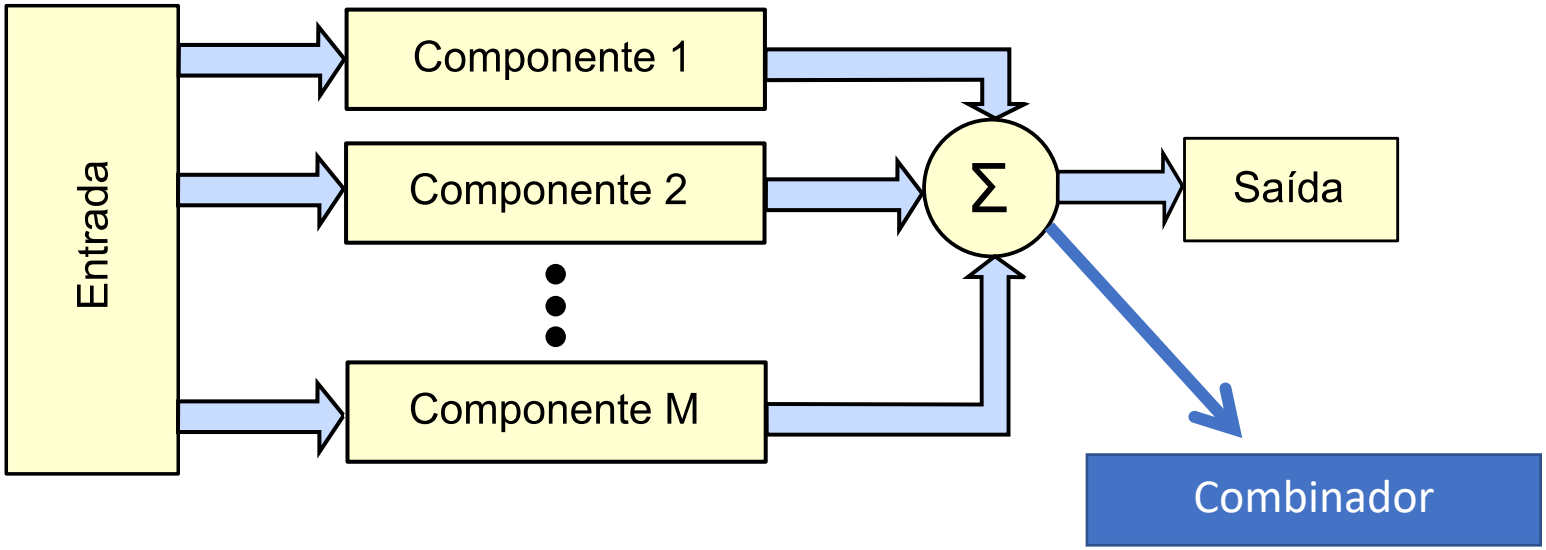


Figura 16 – Estrutura de um comitê de máquinas estático (*ensemble*)

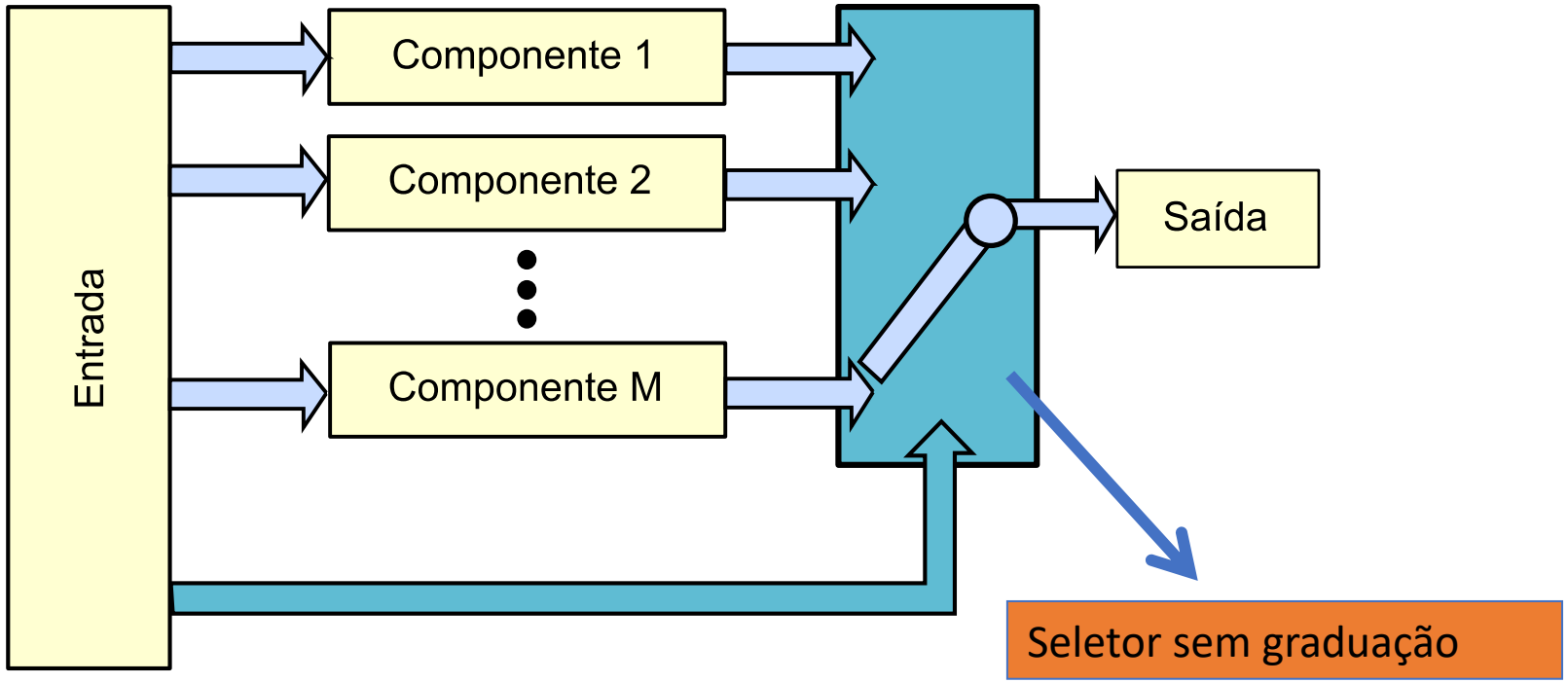


Figura 17 – Estrutura de um comitê de máquinas dinâmico, com seleção binária (*mixture of experts*)

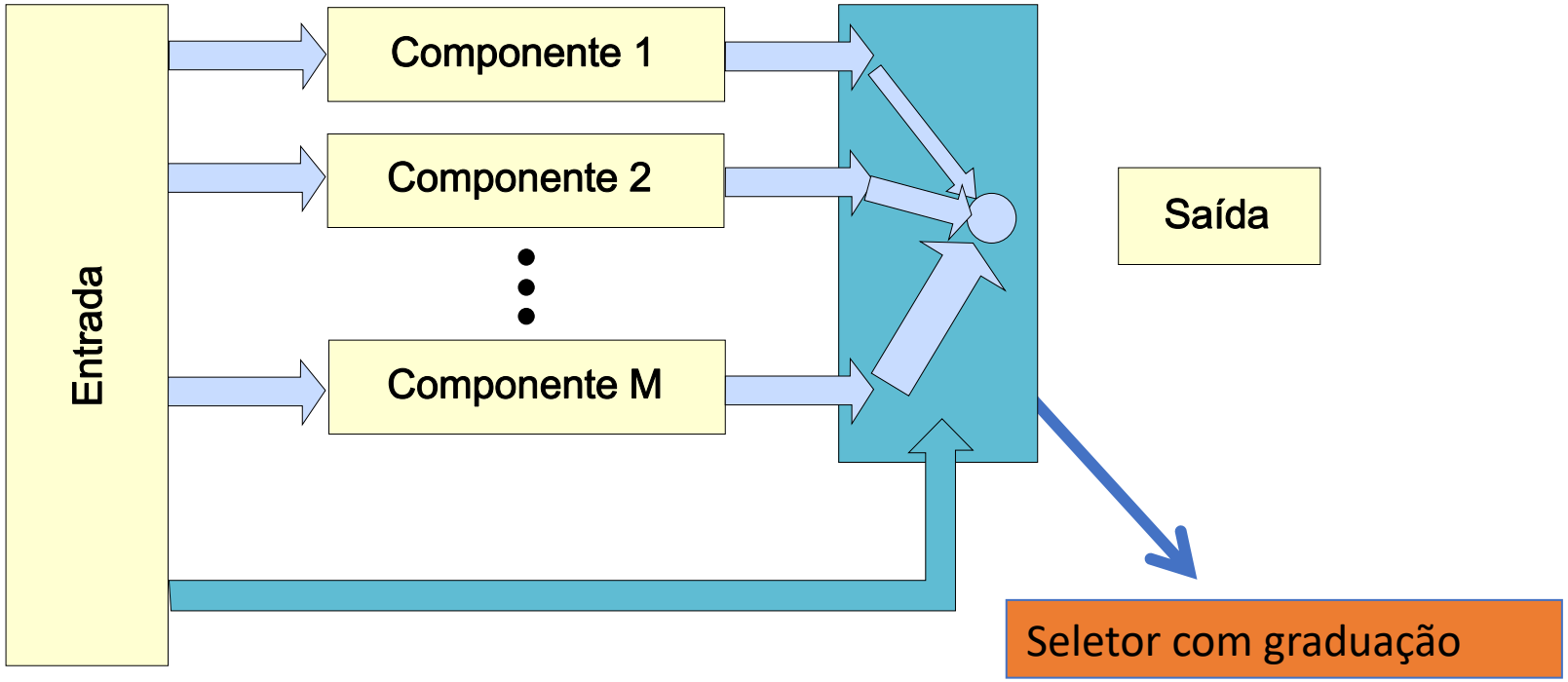
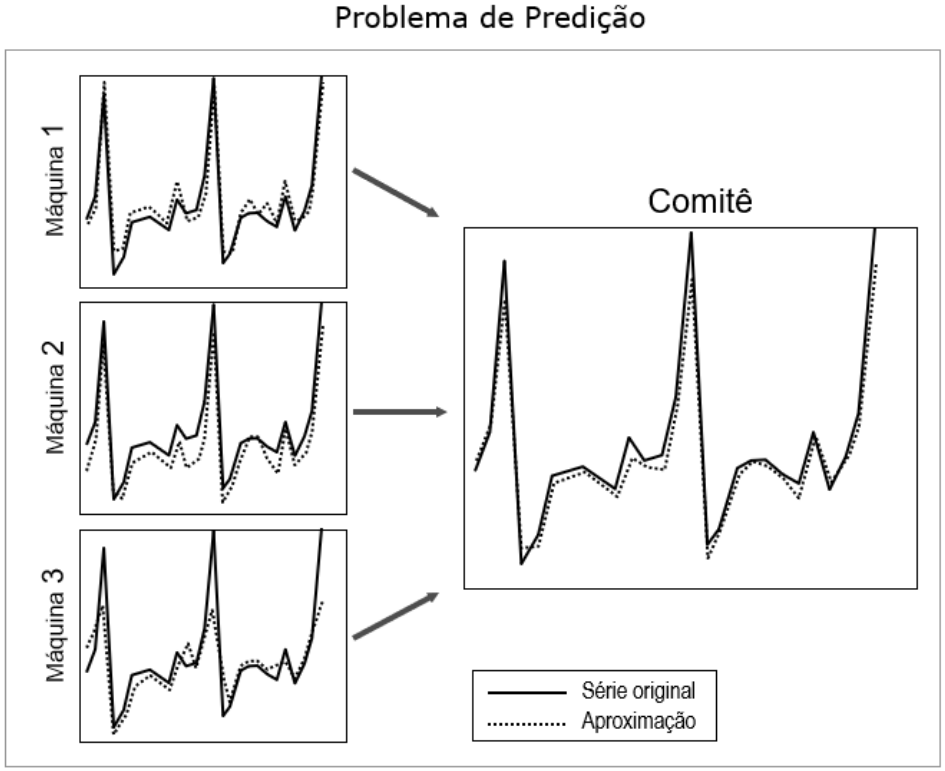


Figura 18 – Estrutura de um comitê de máquinas dinâmico, com seleção graduada (*mixture of experts*)



Figuras extraídas de
Puma-Villanueva (2006)

Figura 19 – Exemplo de um comitê de máquinas em predição (ensemble)

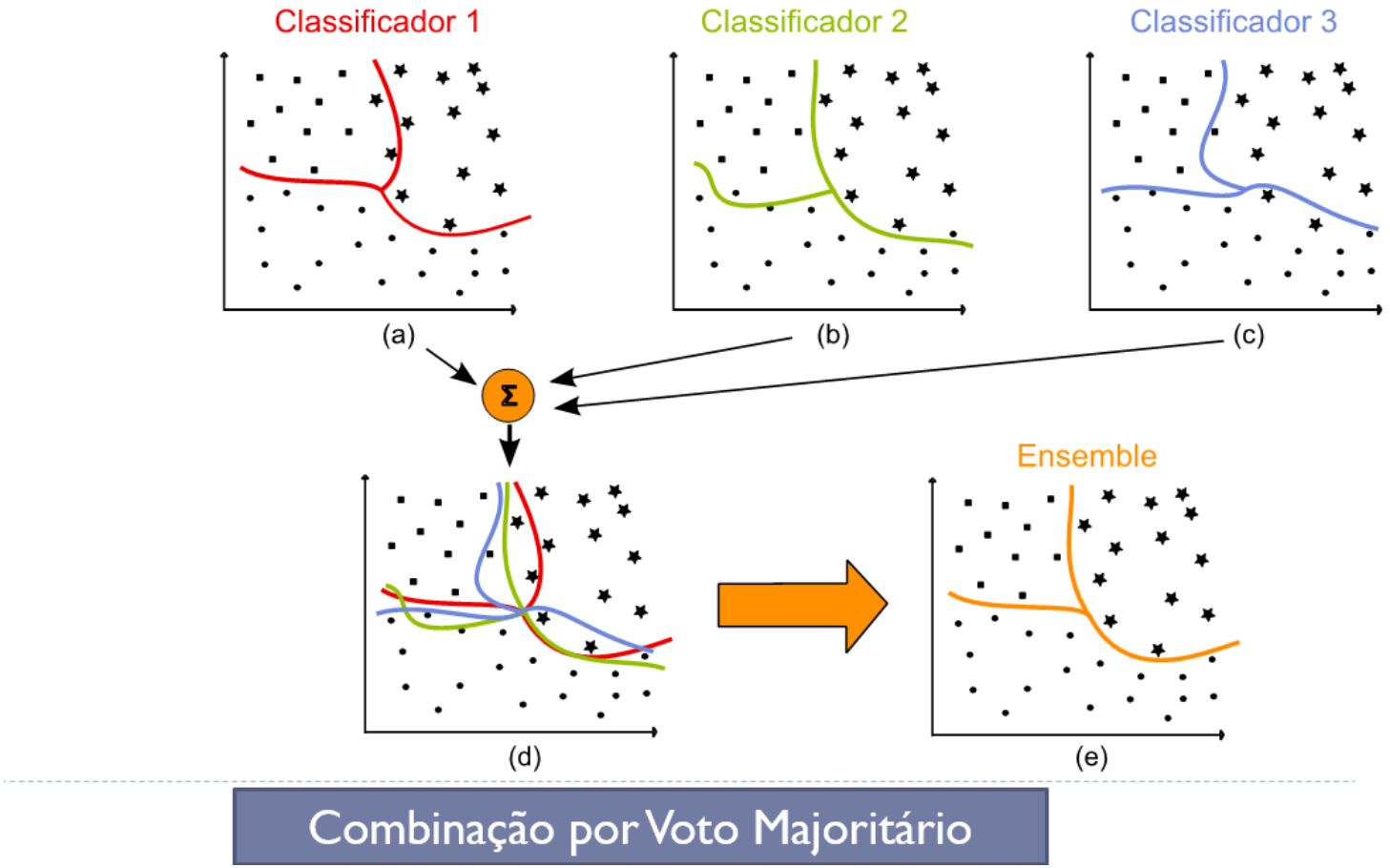


Figura 20 – Exemplo de um comitê de máquinas em classificação (ensemble). Fonte: PUMA-VILLANUEVA (2006).

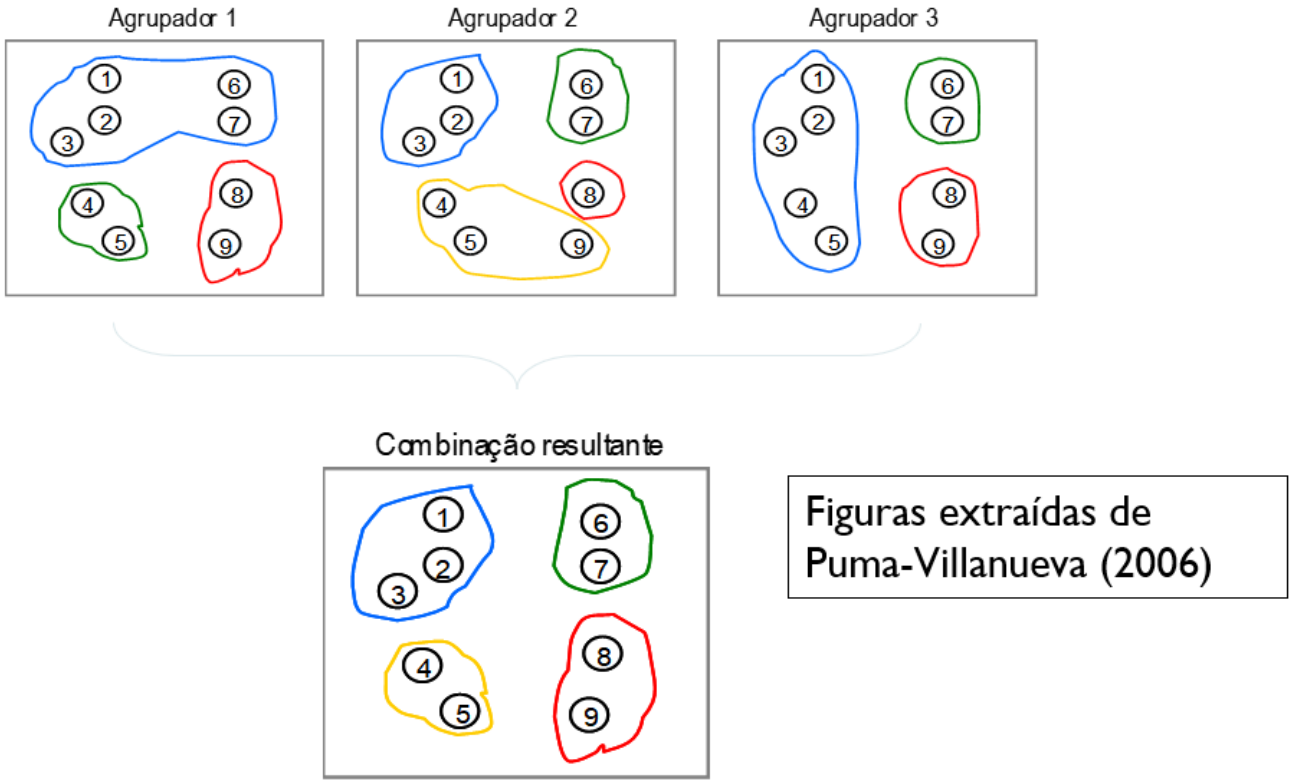


Figura 21 – Exemplo de um comitê de máquinas em agrupamento de dados (ensemble)

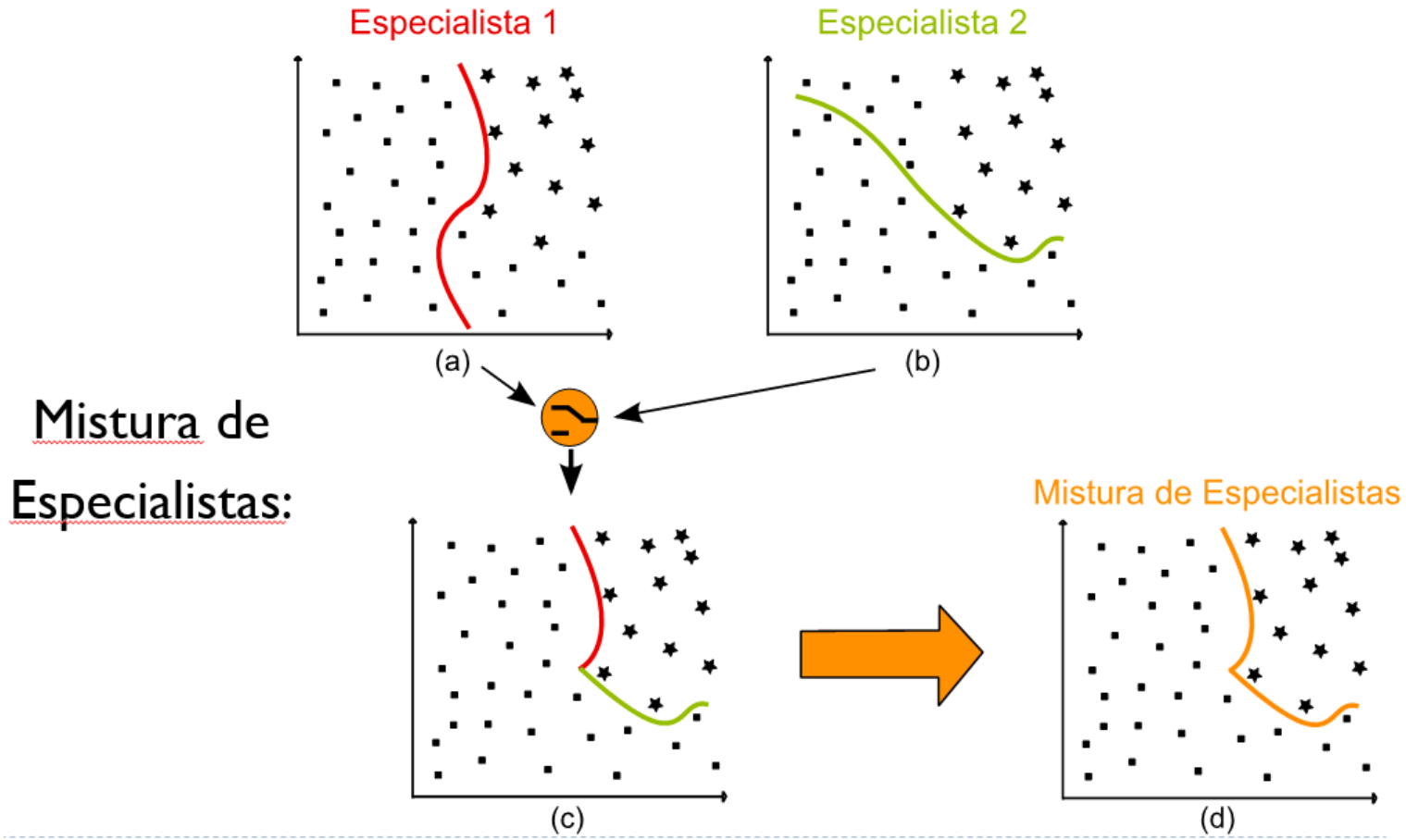


Figura 22 – Exemplo de um comitê de máquinas em classificação (mistura de especialistas).

Fonte: PUMA-VILLANUEVA (2006).

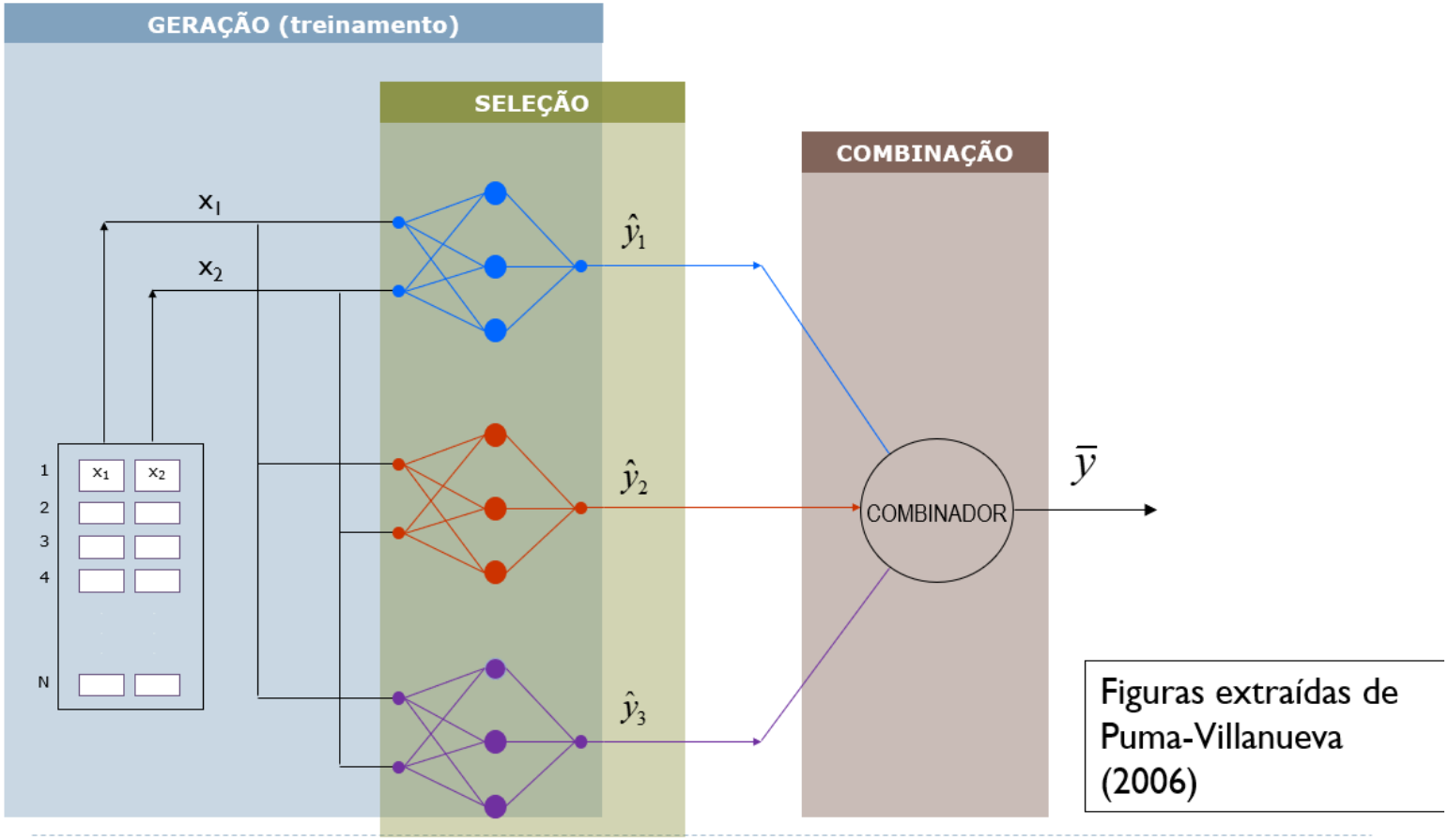


Figura 23 – Etapas na construção de um ensemble

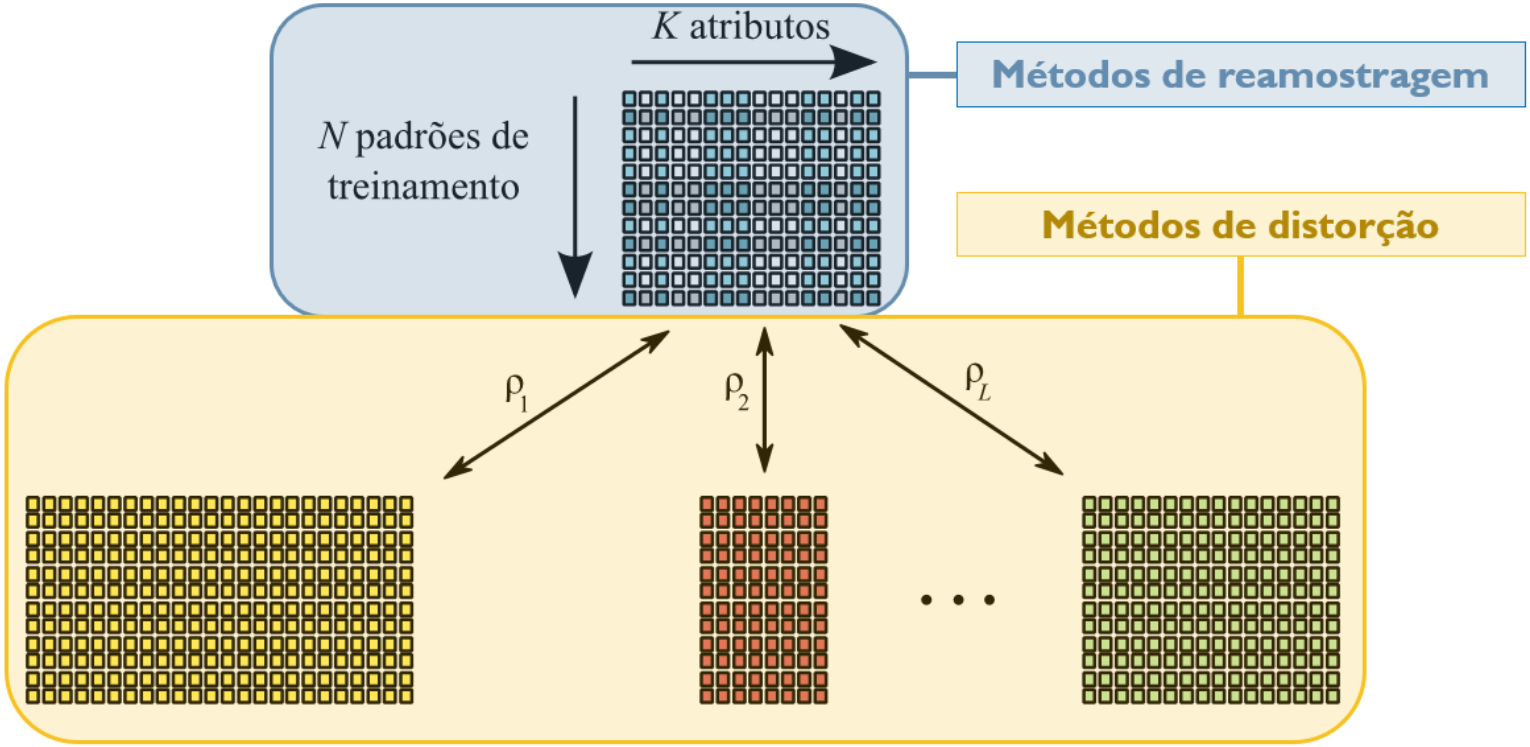


Figura 24 – Manipulação de dados para a geração de diversidade nos modelos de aprendizado.

Fonte: PUMA-VILLANUEVA (2006).

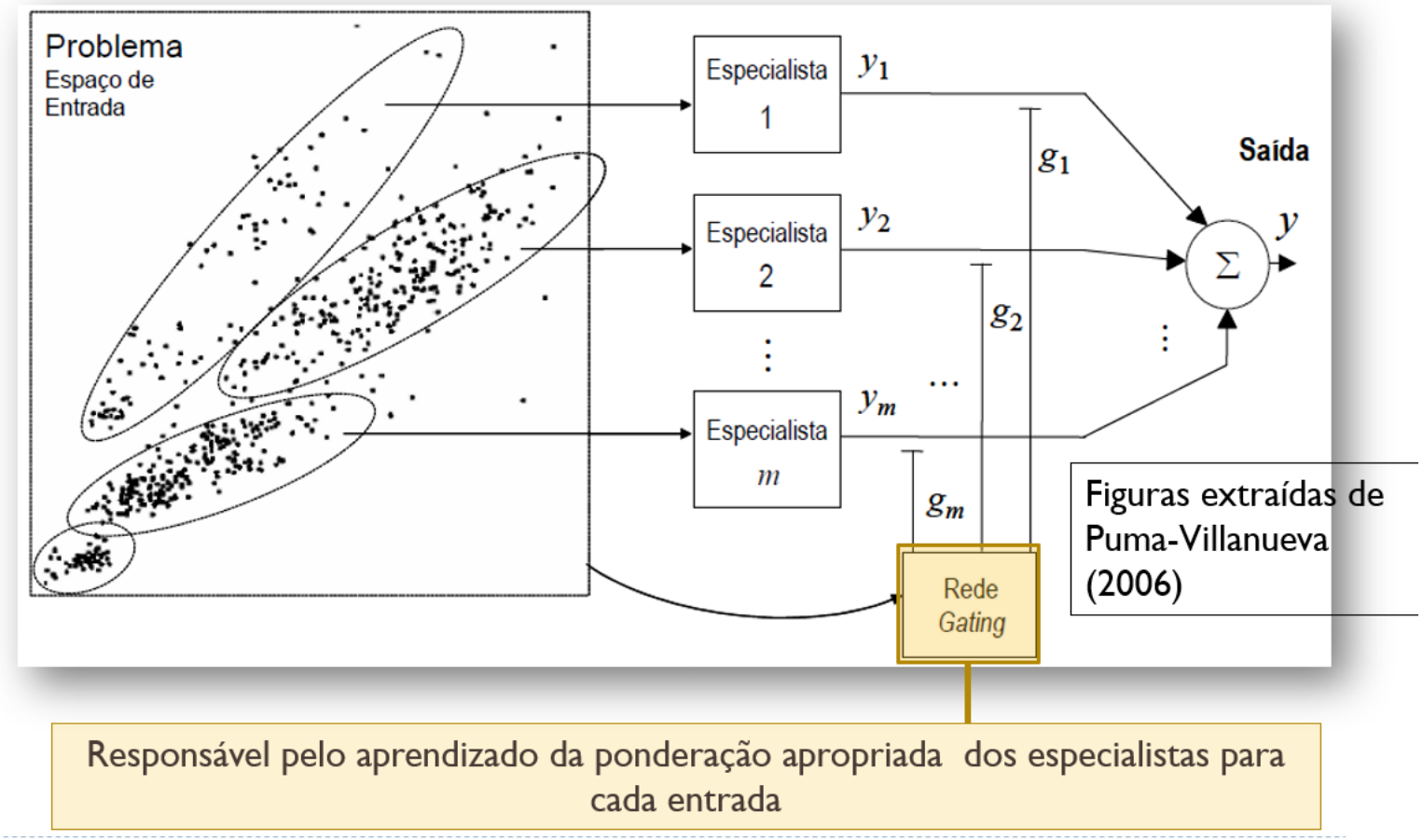


Figura 25 – Princípio de operação da rede *gating* em mistura de especialistas

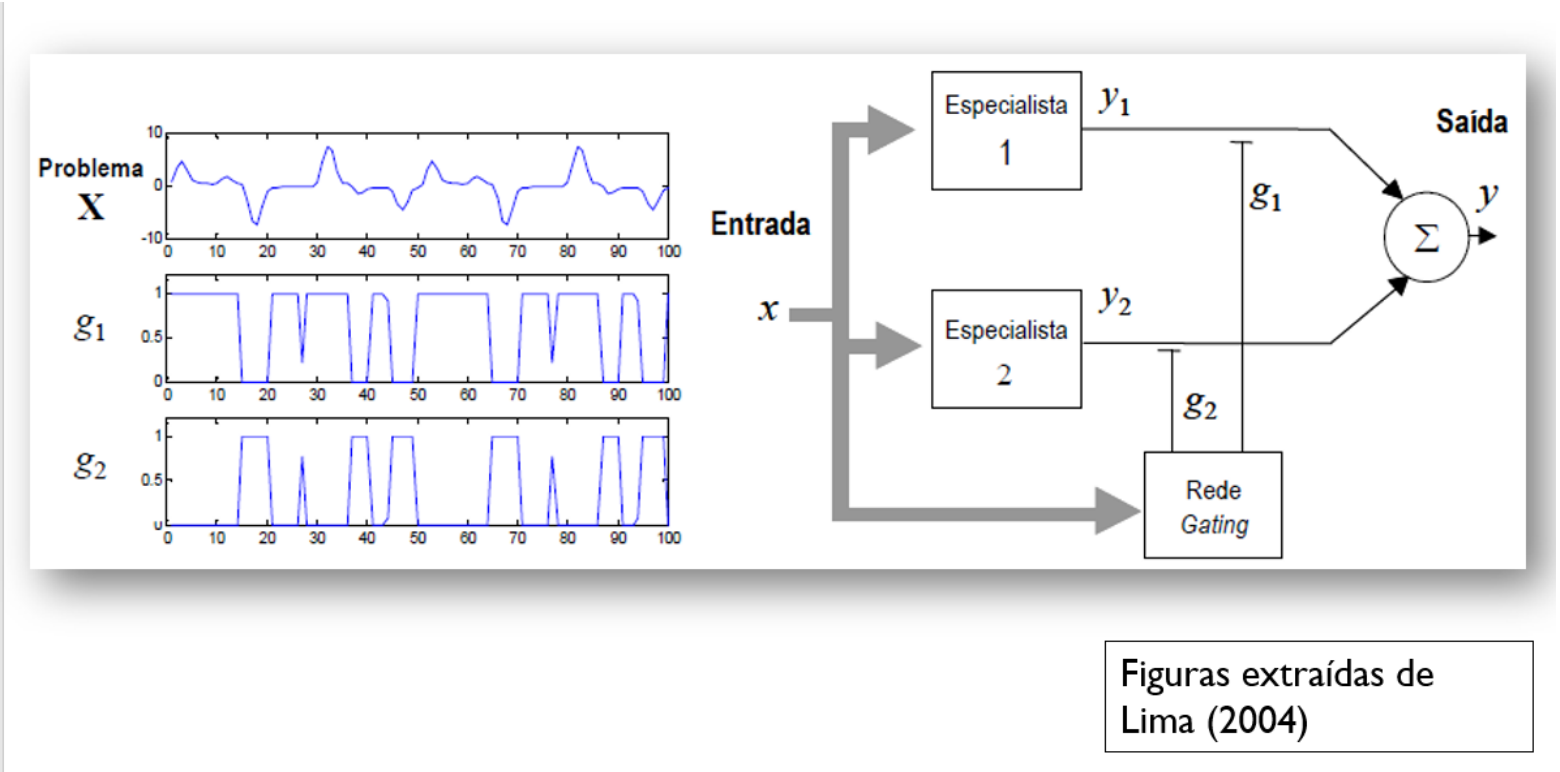


Figura 26 – Dois especialistas em um problema de predição de série temporal, com ênfase no papel da rede *gating*

Figuras extraídas de Lima (2004)

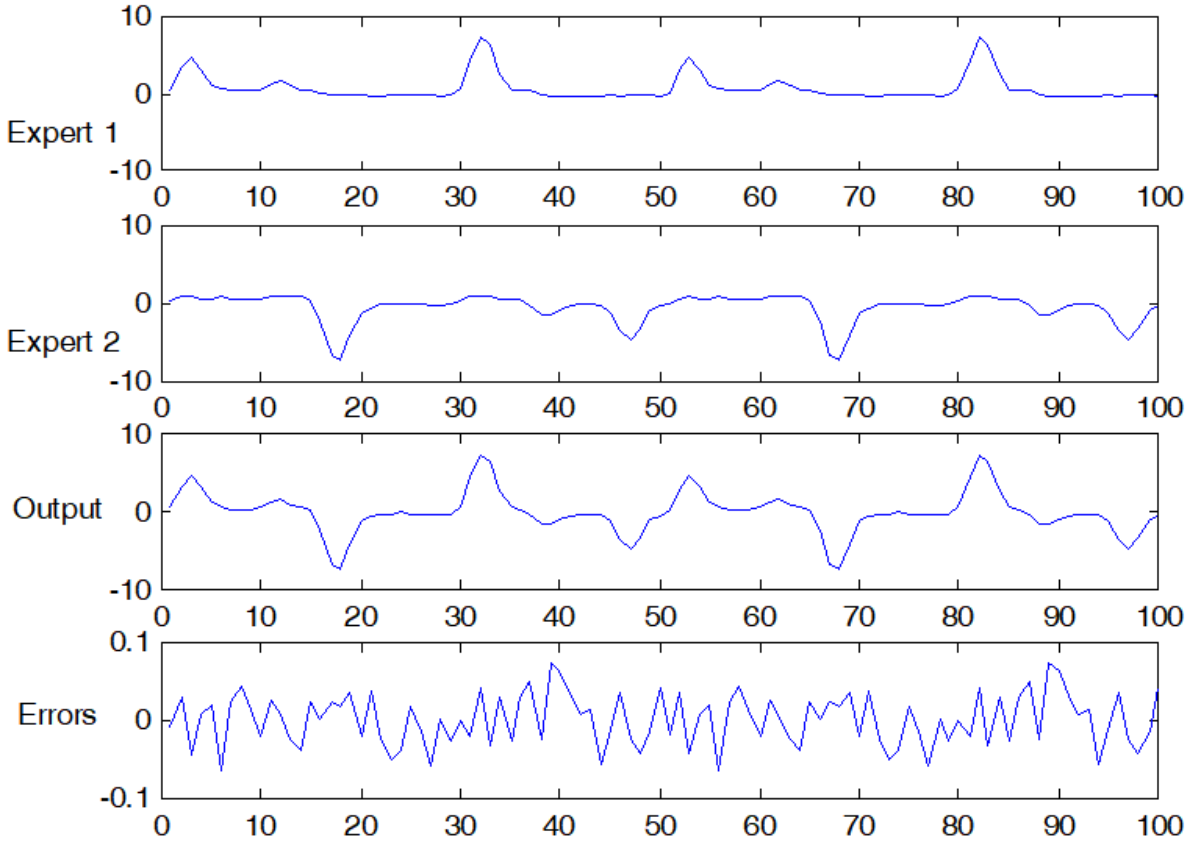


Figura 27 – Dois especialistas em um problema de predição de série temporal, com ênfase no papel de cada especialista

4 Referências bibliográficas

BALDI, P. & SADOWSKI, P. "Understanding Dropout", Advances in Neural Information Processing Systems (NIPS), vol. 26, 2013.

FRAZIER-LOGUE, N. & HANSON, S.J. "Dropout is a special case of the stochastic delta rule: faster and more accurate deep learning", arXiv:1808.03578v1, 2018.

GOODFELLOW, I.; BENGIO, Y. & COURVILLE, A. "Deep Learning", The MIT Press, 2016. (http://www.deeplearningbook.org/)

LIMA, C.A.M. "Comitê de Máquinas: Uma Abordagem Unificada Empregando Máquinas de Vetores-Suporte", Tese de Doutorado, Fac. de Engenharia Elétrica e de Computação, Unicamp, 2004.

MASOUDNIA, S.; EBRAHIMPOUR, R. "Mixture of experts: a literature survey", Artificial Intelligence Review, vol. 42, no. 2, pp. 275-293, 2014.

PERRONE, M.P.; COOPER, L.N. "When Networks Disagree: Ensemble Methods for Hybrid Neural Networks", in R.J. Mammone (ed.) Neural Networks for Speech and Image Processing, Chapman-Hall, 1992.

PUMA-VILLANUEVA, W.J. "Comitê de Máquinas em Predição de Séries Temporais", Dissertação de Mestrado, Fac. de Engenharia Elétrica e de Computação, Unicamp, 2006.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.

YUKSEL, S.E.; WILSON, J.N.; GADER, P.D. "Twenty Years of Mixture of Experts", IEEE Trans. on Neural Networks and Learning Systems, vol. 23, no. 8, pp. 1177-1193, 2012.

ZHOU, Z.-H. "Ensemble Methods: Foundations and Algorithms", Chapman & Hall/CRC Press, 2012.

Outros links relevantes:

https://timdettmers.com/2015/03/26/convolution-deep-learning/

https://arxiv.org/pdf/1803.02129.pdf