MAC5921 – Deep Learning

Aula 04 – 22/08/2023

Nina S. T. Hirata

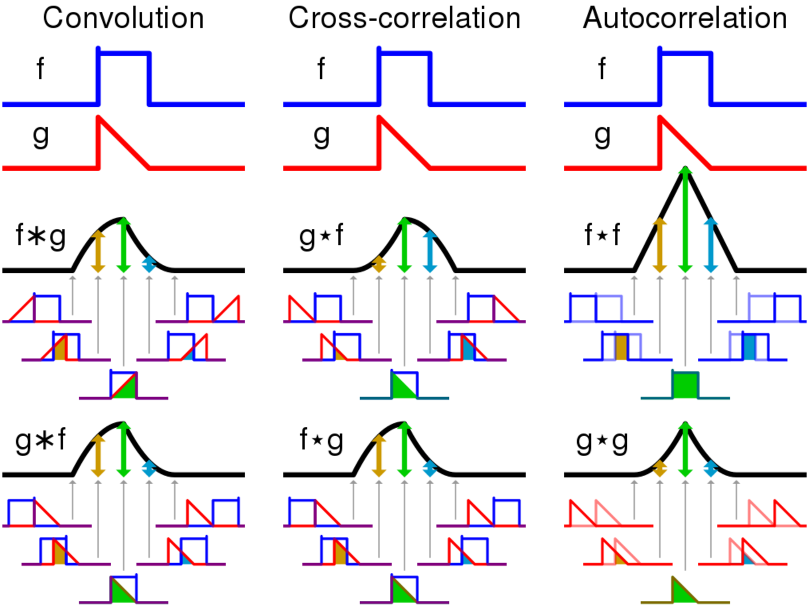
Convolution

Domínio contínuo

$$\mathbf{(f * g)(t) = \int_{-\infty}^{+\infty} f(a)g(t-a)da = \int_{-\infty}^{+\infty} f(t-a)g(a)da}$$

Domínio discreto

$$\mathbf{(f * g)(t) = \sum_{-\infty}^{+\infty} f(a)g(t-a) = \sum_{-\infty}^{+\infty} f(t-a)g(a)}$$


Convolution
Cross-correlation
Autocorrelation
f
g
f∗g
g∗f
g⋆f
f⋆g
f⋆f
g⋆g

Convolução domínio discreto 2D

$$S(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) = \sum_m \sum_n I(i-m,j-n)K(m,n)$$

Cross-correlation domínio discreto 2D

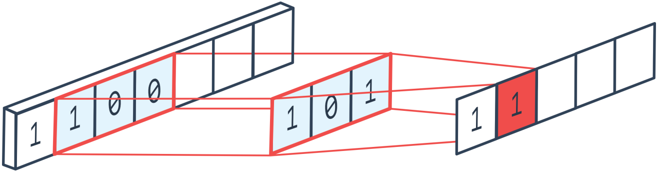
$$S(i,j) = \sum_m \sum_n I(i+m,j+n)K(m,n)$$

Supondo que K tem suporte finito,

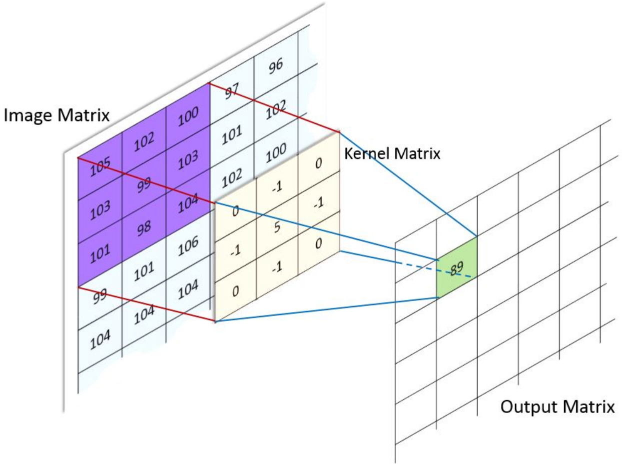
$$S(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

representa a ideia de "sliding window"

1D Convolution



2D Convolution



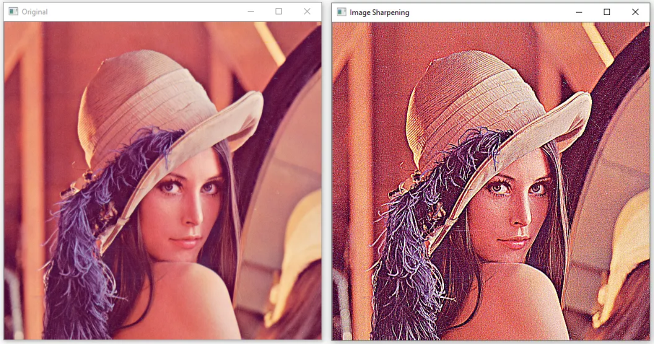
(http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html)

Smoothing

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

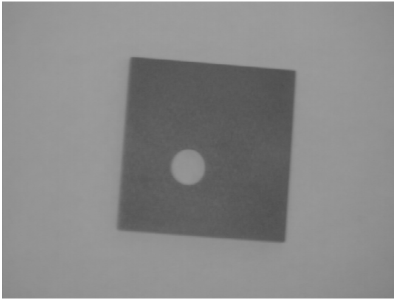
(http://aishack.in/tutorials/image-convolution-examples/)

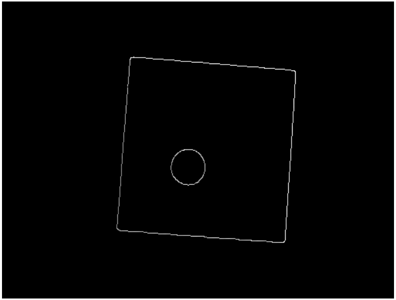
Sharpen



$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Examples of convolution kernels – edge detection

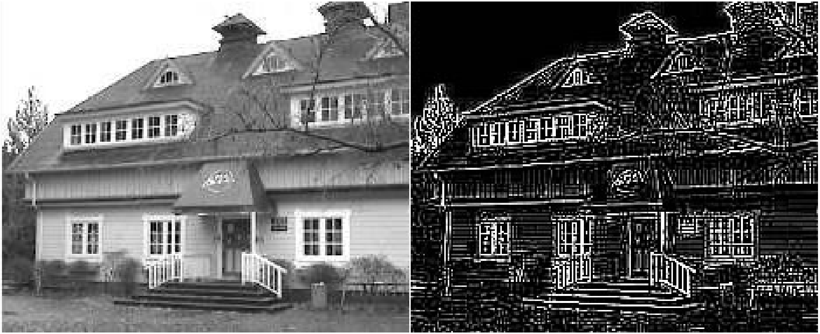




-1	-1	-1
-1	8	-1
-1	-1	-1

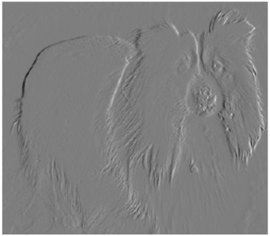
(http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm)

Examples of convolution kernels – edge detection



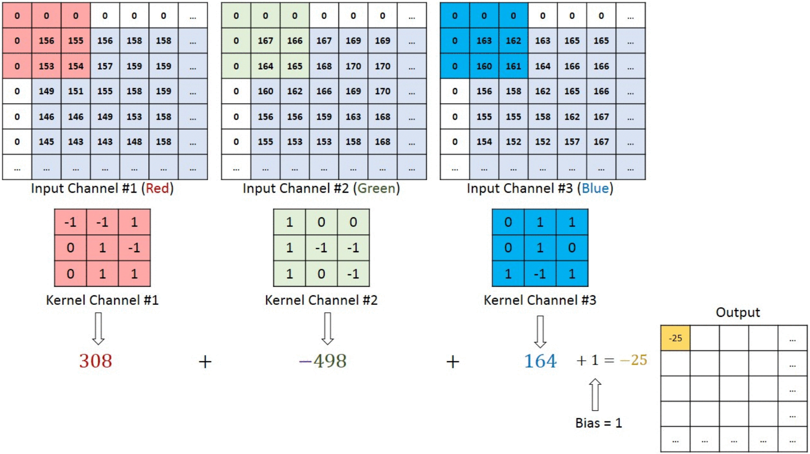
(http://aishack.in/tutorials/image-convolution-examples/)

Examples of convolution kernels – edge detection



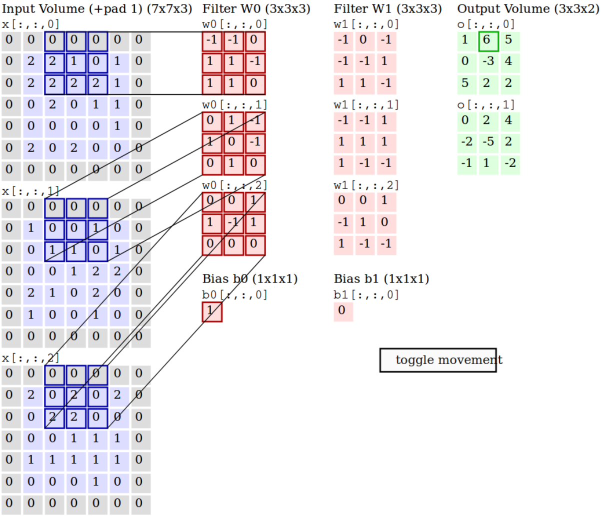
$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Convolution on multiple channels



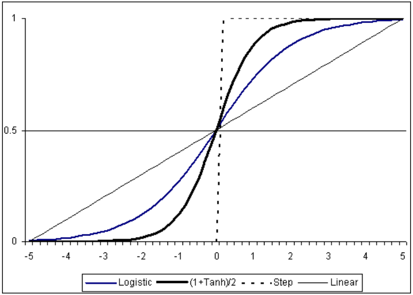
(http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html)

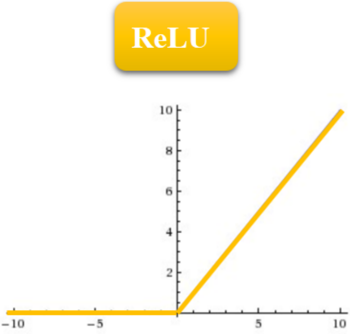
Multiple filters



Concepts: padding, stride, kernel size, number of filters/maps

Activation functions



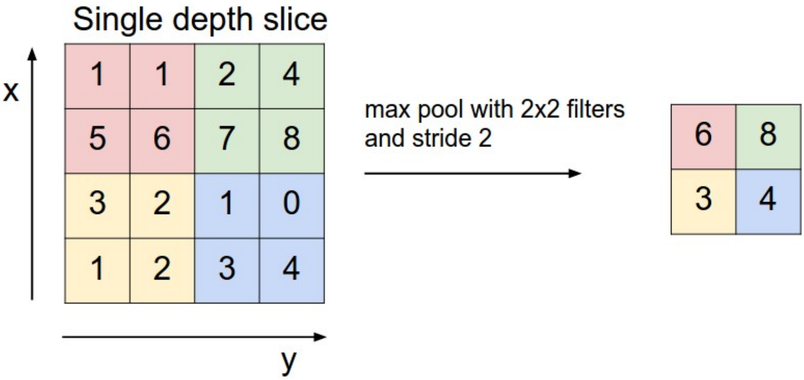


$$F(x) = \max(0, x)$$

Subsampling

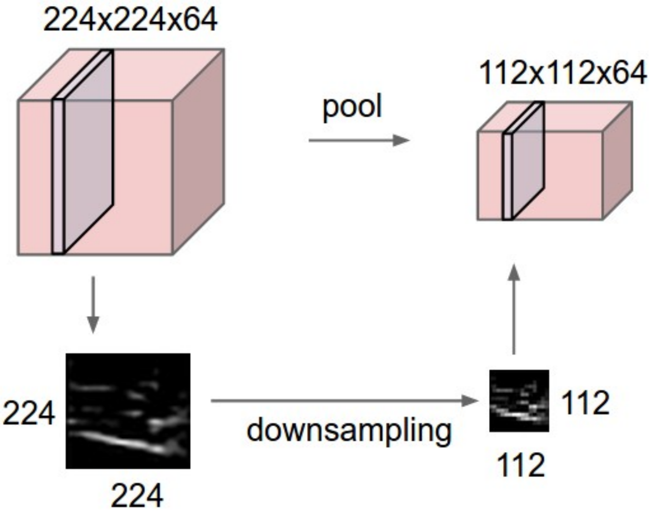
- via strides – skip pixels by steps of size d
- via pooling – aggregate $\mathrm{d} \times \mathrm{d}$ pixels

Max pooling

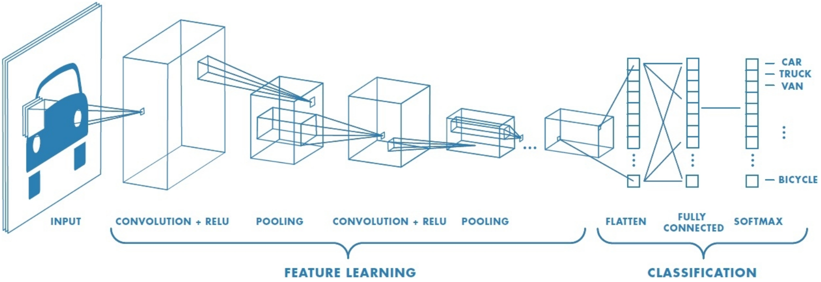


Single depth slice
x
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
y
max pool with 2x2 filters
and stride 2
6 8
3 4

Pooling



Typical architecture of Convnets



Source: https://iitmcvg.github.io/summer_school/DLSession3/

- Convolutonal layers: feature extraction

- Fully connected layers: classification

Por que usar convolução?

weight sharing: Redução no número de pesos a serem ajustados

propriedade de “equivariance to translation”

Main CNN architectures used in ILSVRC

CNN models

- **AlexNet**, 2012: winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), 60M network parameters

  (Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012, pages 1097–1105)

- **VGG-11, 16 e 19**, 2014: 8, 13 e 16 convolutional layers, VGG-19 138M network parameters

  (Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. CoRR, abs/1409.1556, 2014)

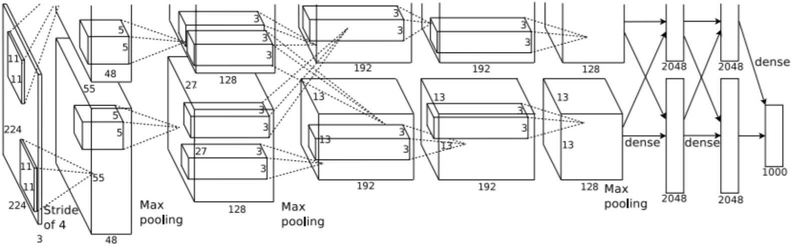
- **GoogleLeNet** (Inception), 2014: winner of ILSVRC 2014, inception layers, 7M network parameters

  (C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CVPR 2015, pages 1–9)

- **Residual Network** (ResNet), 2015: winner of ILSVRC 2015, 25.5M network parameters, residual block, vanishing gradient

  (K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR 2016, pages 770–778)

AlexNet (winner ILSVRC 2012)

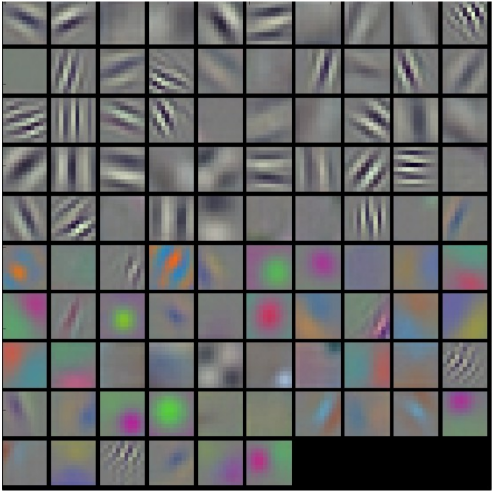


- **AlexNet**, 2012: winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), 60M network parameters

  (Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012, pages 1097–1105)

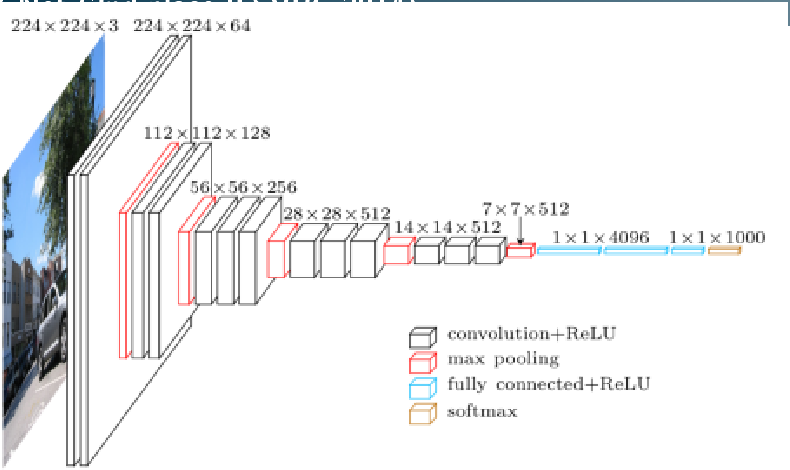
AlexNet (winner ILSVRC 2012)

- succeeded on training a large CNN
- 60 million parameters
- used 2 GPUs
- proposed ReLU
- used dropout
- used data augmentation (rotation, scale, crops, etc)
- indication that number of layers is important



Images that maximize the response of the filters (AlexNet)

VGGNet (2nd place ILSVRC 2014)



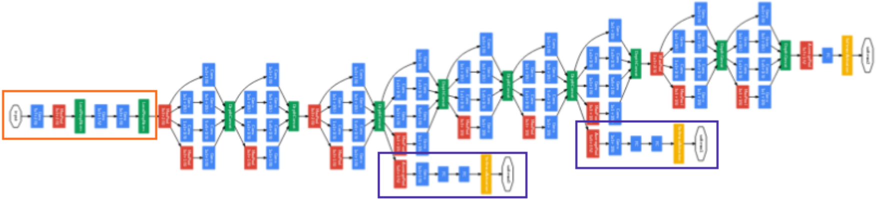
- **VGG-11, 16 e 19**, 2014: 8, 13 e 16 convolutional layers, VGG-19 138M network parameters

  (Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. CoRR, abs/1409.1556, 2014)

VGGNet (2nd place ILSVRC 2014)

- multiple layers (ex., 16 e 19)
- only 3x3 convolutions
- stride 1 for the convolutions (AlexNet used stride ¿ 1)
- first layers with few filters and gradually increasing as we advance throuh the layers
- 144 millions of parameters

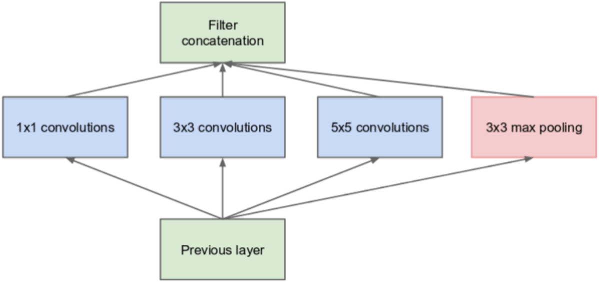
Inception - GoogleLeNet (winner ILSVRC 2014)



- **GoogleLeNet** (Inception), 2014: winner of ILSVRC 2014, inception layers, 7M network parameters

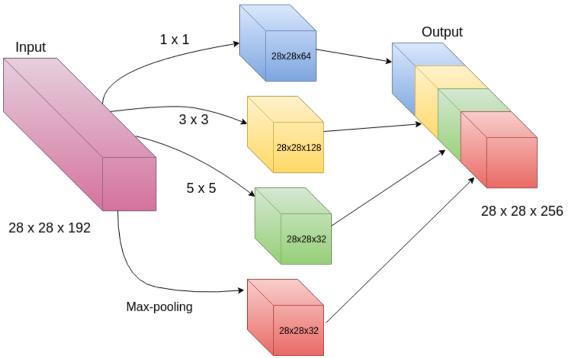
  (C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CVPR 2015, pages 1–9)

Inception – naïve version



(a) Inception module, naïve version

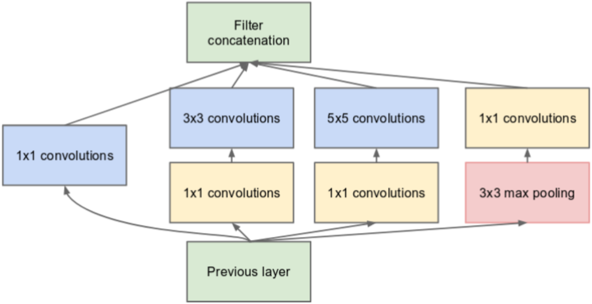
Inception – naïve version example



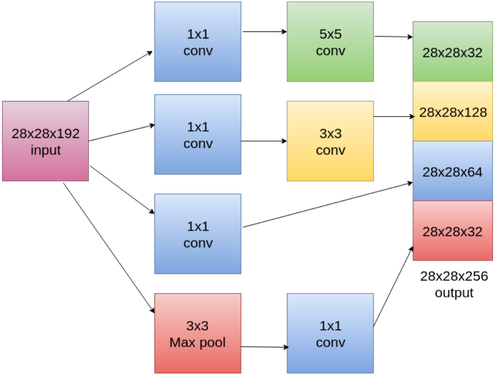
Camada de entrada = 28x28x192

Convolução 5x5: com 32 filtros, camada de saída = 28x28x32. Para cada ponto da camada de saída, o número de multiplicações necessárias é 5x5x192. Como são 28x28x32 pontos na camada de saída, o total de multiplicações necessárias é 5x5x192 x 28x28x32 = 120 milhões.

Inception – effective version

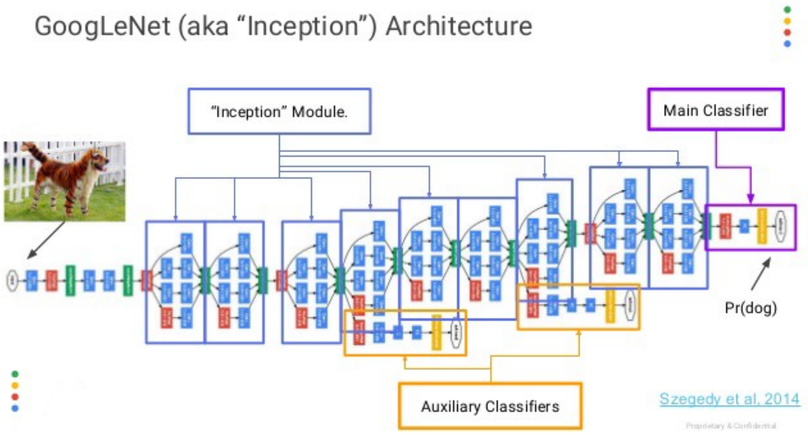


(b) Inception module with dimension reductions



Entrada 28x28x192 $\Longrightarrow$ 16 filtros 1x1 $\Longrightarrow$ camada intermediária 28x28x16 $\Longrightarrow$ 32 convoluções 5x5 $\Longrightarrow$ saída 28x28x32. Total de multiplicações: cada ponto na camada intermediária requer 1x1x192 multiplicações, e portanto 1x1x192 x 28x28x16 = 2.4 milhões para cálculo da camada intermediária; cada ponto na camada de saída requer 5x5x16 multiplicações e portanto 5x5x16 x 28x28X32 = 10 milhões para o cálculo da camada de saída. Total = 12.4 milhões.

Inception - GoogleLeNet (winner ILSVRC 2014)

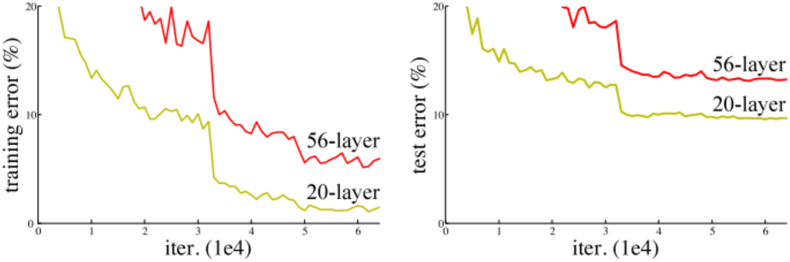


Inception - GoogleLeNet (winner ILSVRC 2014)

- concept of “network in network”
- inception module (“choices” made during training)
- use of 1x1 convolution
- auxiliary output to reinforce activation of intermediary layers
- more aggressive data augmentation tha AlexNet
- 4 million parameters (?)
- variants (e.g., change 5x5 with two 3x3, or change 3x3 with a 1x3 and a 3x1)

ResNet (winner ILSVRC 2015)

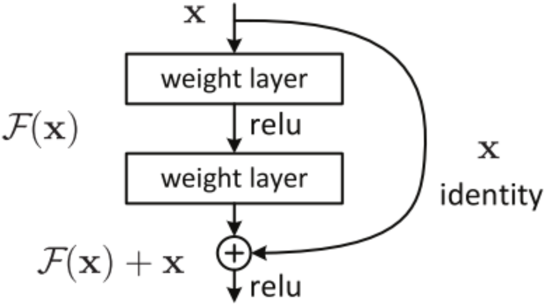
Apesar de ser aceito que profundidade da rede é importante, observou-se que camadas adicionais degradavam a acurácia no conjunto de treinamento



- **Residual Network** (ResNet), 2015: winner of ILSVRC 2015, 25.5M network parameters, residual block, vanishing gradient

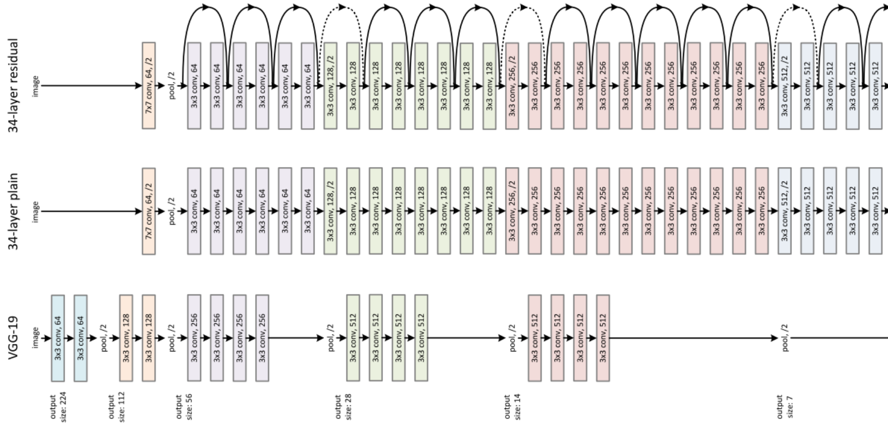
  (K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR 2016, pages 770–778)

ResNet (winner ILSVRC 2015)



Entrada x. Pode ser conveniente aprender um mapeamento ($H(x)$) que seja a identidade. Mas representar identidade por meio de transformações não lineares não é simples. Assim, em vez de H, aprende-se o resíduo $\mathcal{F}(x)$, de mode que $H(x) = \mathcal{F}(x) + x$ e assim a identidade corresponde a resíduo zero

ResNet (winner ILSVRC 2015)



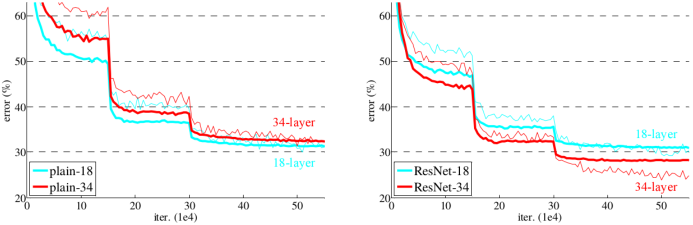


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

ResNet (winner ILSVRC 2015)

- identificam uma degradação associada ao aumento de camadas, que aparentemente não é *overfitting* nem *vanishing gradient*
- propõe módulo para aprender o resíduo da transformação desejada
- aplica *batch normalization* após cada convolução e antes da ativação
- não usa *dropout*
- conseguem treinar redes com mais de 100 camadas
- testado em outros datasets (CIFAR, PASCAL, MS-COCO)
- ResNet 56 layers: 0.85M parameters (?)
- ResNet 110 layers: 1.7M parameters (?)

Training of CNN

It is done using the backpropagation algorithm
(gradient descent)

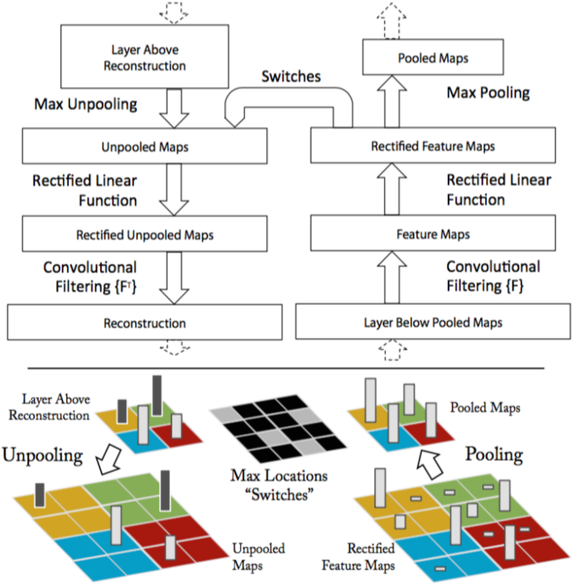
Kernels são ajustados nesse processo – detalhes ficam para depois

Convolutional layers act as feature extractors. How ?

Zeiler et al., 2013, proposed a visualizaton technique

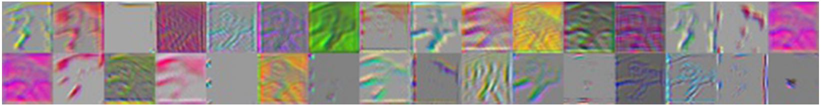
At a given convolutional layer, take the node with largest activation

Reconstruct the input image by backpropagating from that node



(Zeiler et al., 2013)

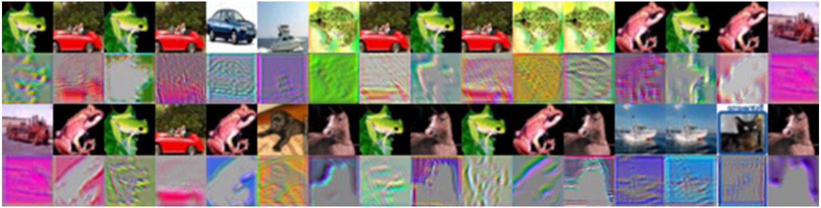
Images reconstructed from neurons 1 to 32 of layer 1, for one input image



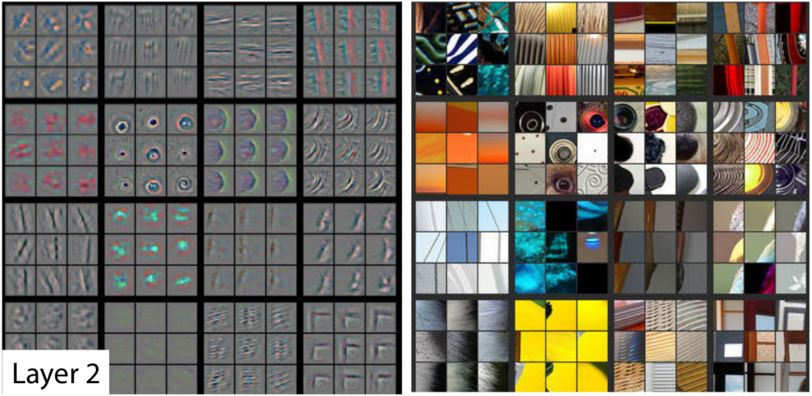
(http://kvfrans.com/visualizing-features-from-a-convolutional-neural-network/)

Images reconstructed from neuron 7 of layer 1, for multiple input images

For each of the 32 neurons, reconstruction of the images that most activated the neuron



(Zeiler et al.) 16 neurons in layer 2; for each neuron, reconstruction of the 9 images that correspond to strongest activation of the node



(Zeiler et al.) 12 neurons in layer 3; for each neuron, reconstruction of the 9 images that correspond to strongest activation of the node