

•
•
•
•
•

Aprendizagem Profunda (Deep Learning)

Marcondes Ricarte e Aluizio Fausto Ribeiro Araújo
Universidade Federal de Pernambuco
Centro de Informática



• • • • •

Conteúdo

- Motivação
- Por que aprendizagem profunda?
- Redes Neurais Convolucionais
- Ferramentas

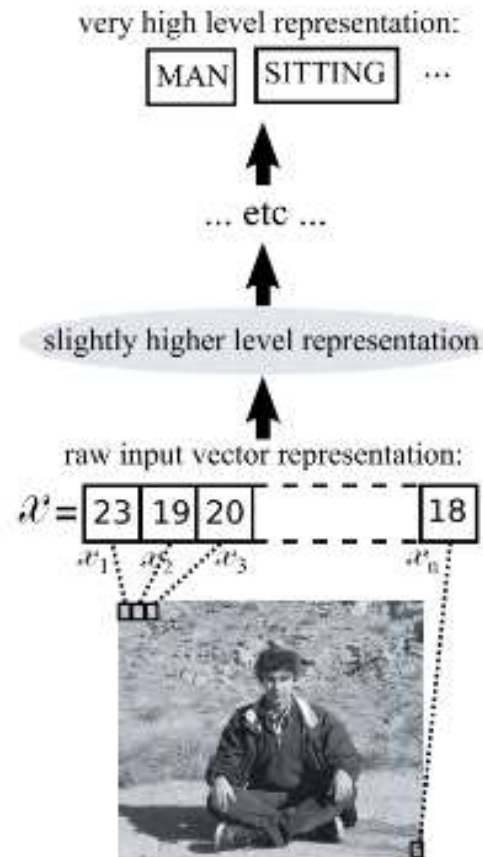
Motivação

- Permitir aos computadores modelar nosso mundo bem o suficiente para exibir o que nós chamamos de inteligência tem sido o foco de pesquisas de mais da metade de um século.
- Para alcançar esse objetivo, é claro que a grande quantidade de informação sobre o nosso mundo deve ser de alguma forma armazenada, explicitamente ou implicitamente, no computador:
 - Para isto se utiliza algoritmos de aprendizagem.

Motivação

- Muito esforço (e progresso!) tem sido feito em entender e melhorar algoritmos de aprendizagem, mas o desafio permanece:
 - Há algoritmos capazes de entender cenas e descrevê-las em linguagem natural?
 - Há algoritmos capazes de inferir conceitos semânticos suficientes a ponto de interagir com humanos?

Motivação

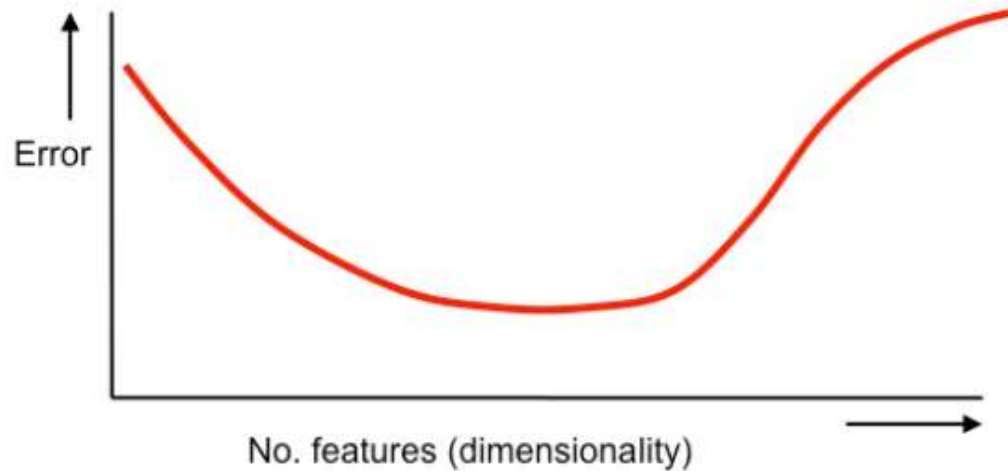


Motivação

- Um modo plausível e comum de extrair informação útil a partir de imagens naturais envolve transformar gradualmente os pixels com valores “brutos” em representações mais abstratas:
 - Detecção de borda, detecção de formas locais, identificação de categorias;
- Assim, uma máquina que expresse “inteligência” requer funções matemáticas que expressem variabilidade:
 - Não-linearidades para processar entradas cruas, sendo capazes de apresentar muitas variações.

Motivação

- Redes Neurais Artificiais *feed-forward*:
 - Maldição da dimensionalidade;



Motivação

- Uma das soluções encontradas para maldição de dimensionalidade é pré-processamento de dados:
 - Redução da dimensionalidade (às vezes por humanos);
 - Desafiante e altamente dependente da tarefa;
- Não há indícios de que o cérebro humano trabalhe com pré-processamento;
- Aprender características automaticamente em múltiplos níveis de abstração permite ao sistema mapear funções complexas sem depender de características intermediárias inteligíveis aos humanos.

Motivação

- O MLP pode não funcionar bem por causa de:
 - Difusão do gradiente;
 - Treinamento muito lento;
 - Camadas mais distantes da saída tendem a fazer um mapeamento aleatório;
 - Frequentemente há mais dados não rotulados do que dados rotulados em problemas reais;
 - Existência de mínimo locais.

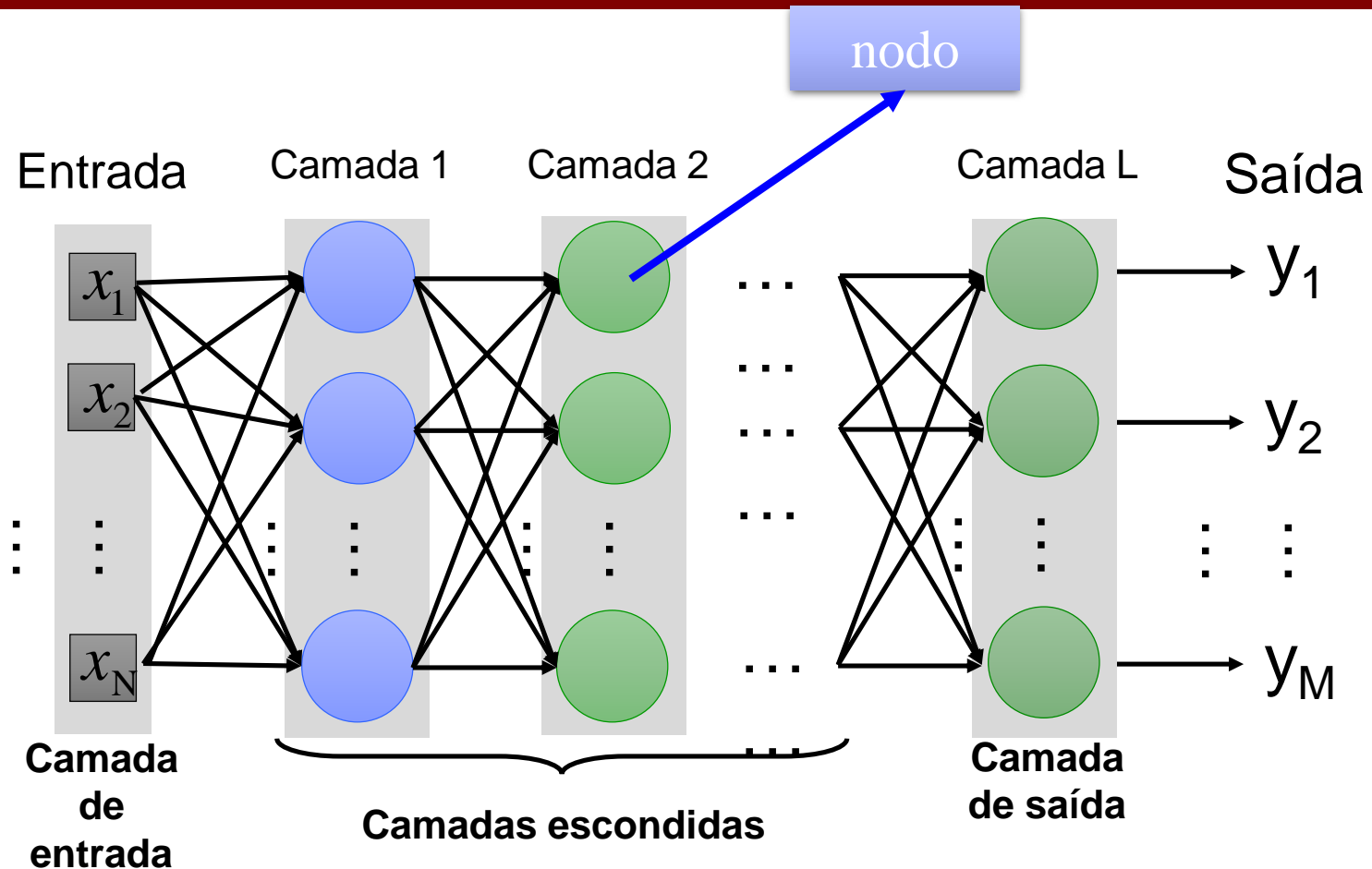
Motivação

- Aprendizagem Profunda aprende em vários níveis de abstração;
 - Representações mais abstratas extraem informações para classificadores ou preditores;
 - Características intermediárias aprendidas podem ser compartilhadas entre diferentes tarefas;
- Por décadas, diversos pesquisadores tentaram, sem sucesso esperado, treinar redes neurais de múltiplas camadas profundas,
 - Inicializações com pesos aleatórios levavam a mínimos locais;
- Hinton *et al* (2006) melhoraram o desempenho de uma rede neural profunda com etapa de pré-treinamento por aprendizagem não-supervisionada, uma camada após outra a partir da primeira.

Por que Aprendizagem Profunda (*Deep Learning*)?

- Teorema de Kolmogorov-Smirnov: Uma única camada escondida pode resolver qualquer função e generalizar,
 - Grande número de nodos pode inviabilizar solução do problema.
- Múltiplas camadas constroem melhor espaço de características:
 - Primeira camada aprende as características de primeira ordem (por exemplo, bordas em imagem);
 - Segunda camada aprende características de maior ordem (por exemplo, combinação de bordas e outras características);
 - Camadas são treinadas por método não-supervisionado e as características alimentam uma camada supervisionada;
 - A rede inteira é então ajustada de modo supervisionado.

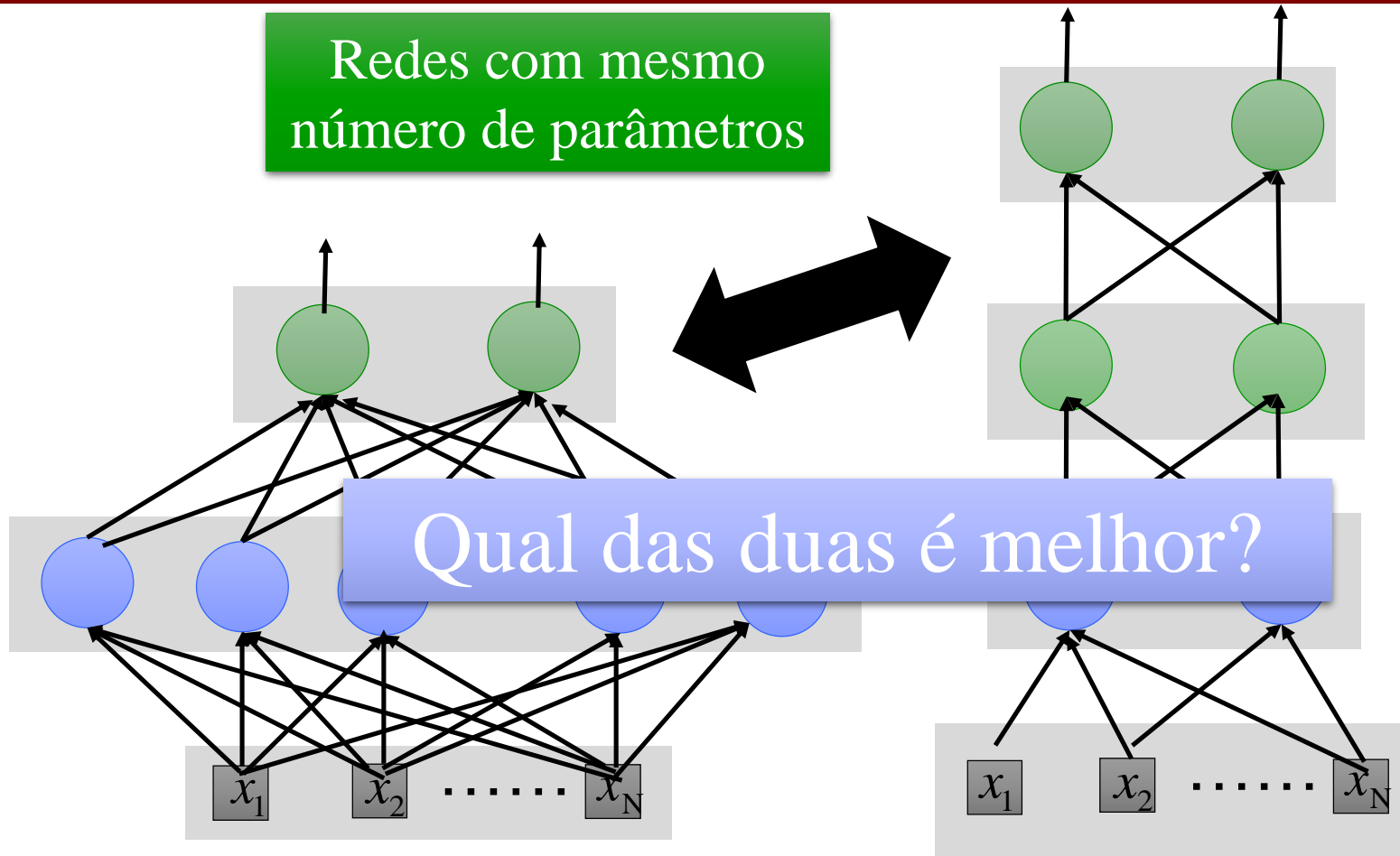
Por que Aprendizagem Profunda (*Deep Learning*)?



Profundo significa muitas camadas

Por que Aprendizagem Profunda (*Deep Learning*)?

Redes com mesmo
número de parâmetros

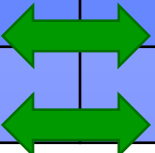


Rasa: gorda e baixa

Profunda: Magra e alta

Por que Aprendizagem Profunda (*Deep Learning*)?

DBN: # de camadas X tamanho	Taxa de erro das palavras (%)	BP # de camadas X tamanho	Taxa de erro das palavras (%)
1 X 2k	24.2	1 X 2k	24.3
2 X 2k	20.4	1 X 2k	22.2
3 X 2k	18.4	1 X 2k	20.0
4 X 2k	17.8	1 X 2k	18.7
5 X 2k	17.2	1 X 3772	18.2
7 X 2k	17.1	1 X 4634	17.4



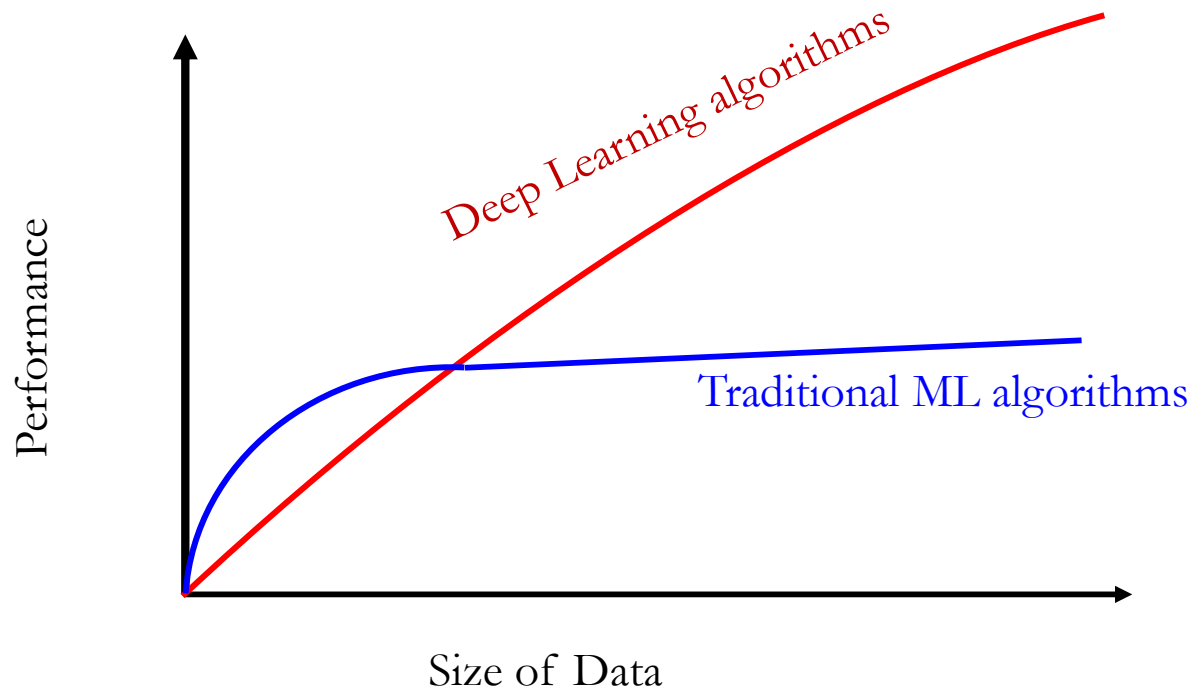
Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Por que Aprendizagem Profunda (*Deep Learning*)?

- Primeiras MLPs com muitas camadas, constatações do treinamento:
 - Gradiente é muito reduzido com retropropagação dos erros para primeiras camadas, causando lentidão excessiva no treinamento;
 - As camadas superiores costumam aprender bem, reduzindo o erro, portanto, o erro retropropagado para camadas anteriores cai rapidamente, logo as primeiras camadas não conseguem se adaptar para melhorar os resultados, elas fazem o papel de um mapa aleatório de características;
 - Necessidade de um modo efetivo de treinar as camadas iniciais.

Por que Aprendizagem Profunda (*Deep Learning*)?

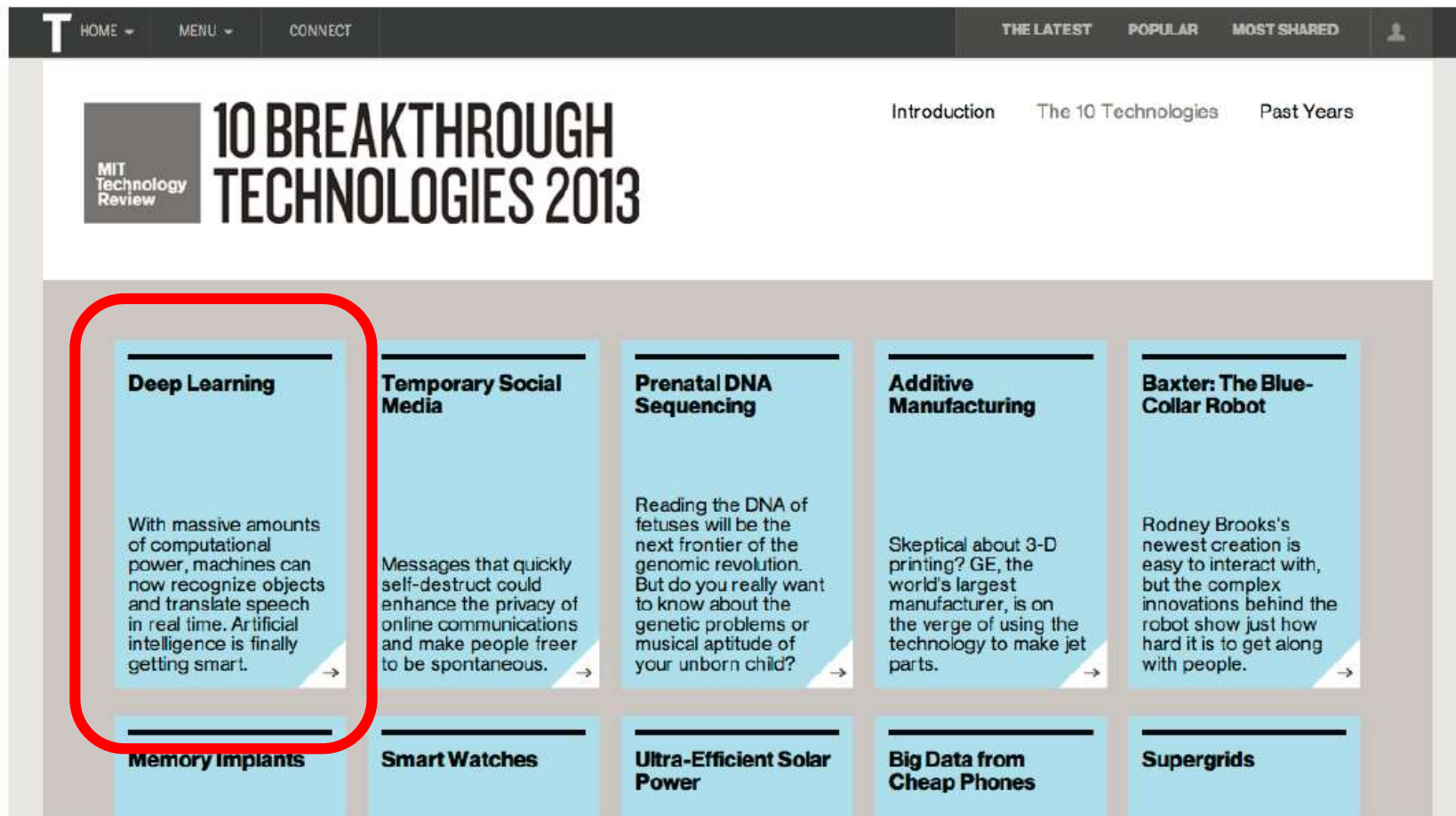
Desempenho vs Tamanho dos dados



Por que Aprendizagem Profunda (*Deep Learning*)?

- Plausibilidade biológica – córtex visual
- Problemas que podem ser representados com um número polinomial de nodos com k camadas podem requerer número exponencial de nodos com $k-1$ camadas:
 - Funções muito variáveis podem ser eficientemente representadas com arquiteturas profundas.

Por que Aprendizagem Profunda (*Deep Learning*)?



MIT Technology Review, April 23rd, 2013

Por que Aprendizagem Profunda (*Deep Learning*)?

- Alguns modelos importantes:
 - Supervisionado:
 - *Convolutional NN* (LeCun);
 - *Recurrent Neural Nets* (Schmidhuber);
 - Não-supervisionado:
 - *Deep Belief Nets / Stacked Restrited Boltzmann Machines RBMs* (Hinton);
 - *Stacked denoising autoencoders* (Bengio) ;
 - *Sparse AutoEncoders* (LeCun, A. Ng);

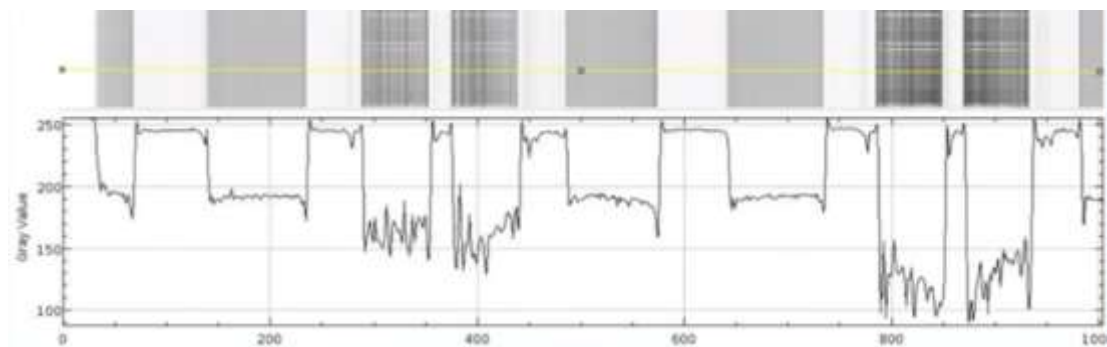
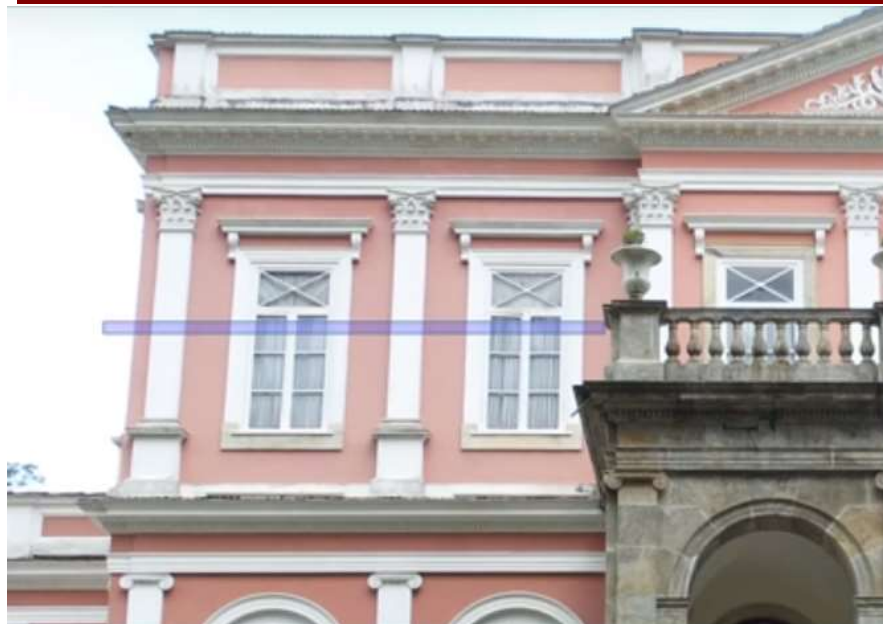
Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

- Operações básicas empregadas no modelo:
 - Convolução;
 - Inserção de não-linearidade;
 - Sub-amostragem.

Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Convolução



Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Convolução

- **Convolução** é um operador linear que toma duas funções no domínio do tempo e produz uma terceira função que calcula a área subentendida pela superposição delas em função do deslocamento existente entre elas.

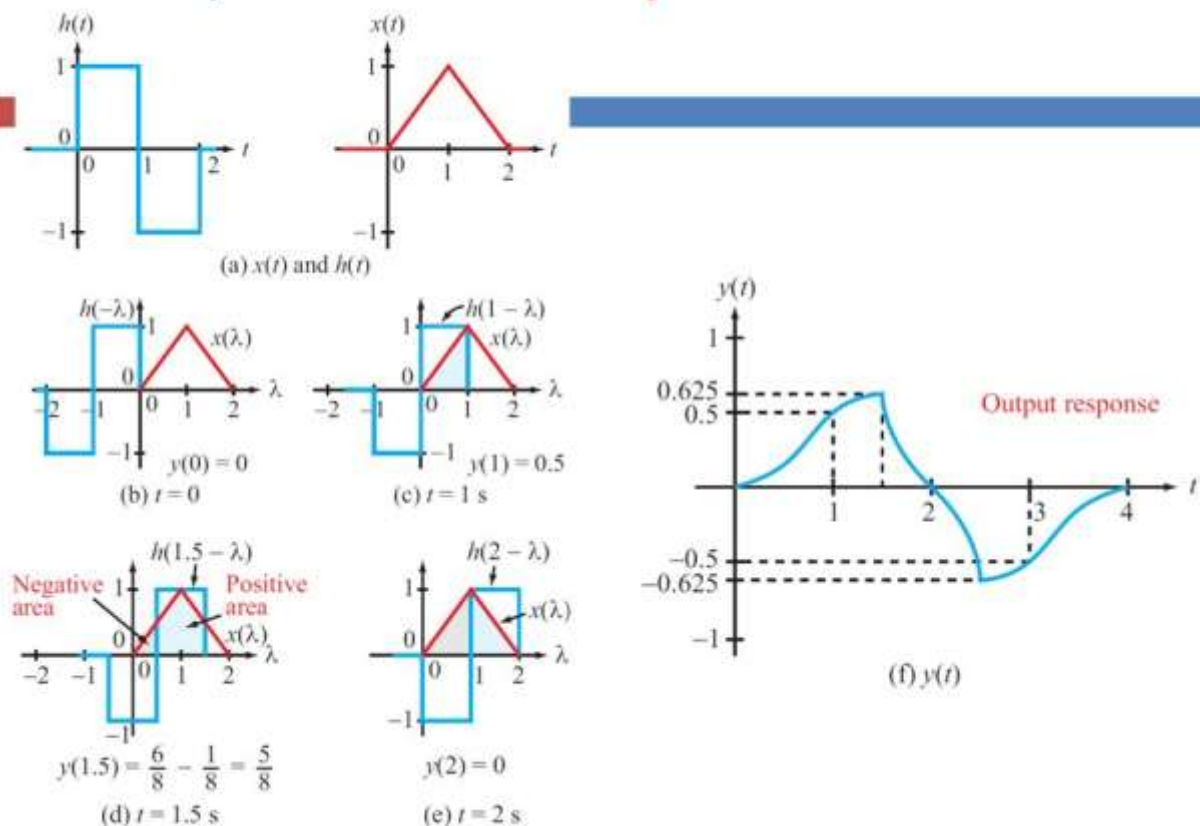
- Contínua:
$$(f * g)(x) = h(x) = \int_{-\infty}^{\infty} f(u) \cdot g(x - u) du$$

- Discreta:
$$(f * g)(k) = h(k) = \sum_{j=0}^k f(j) \cdot g(k - j)$$

Redes Neurais Convolucionais

(Convolutional Neural Networks) - Convolução

Example 10-16: Graphical Convolution



Convolução é formada pelas operações:

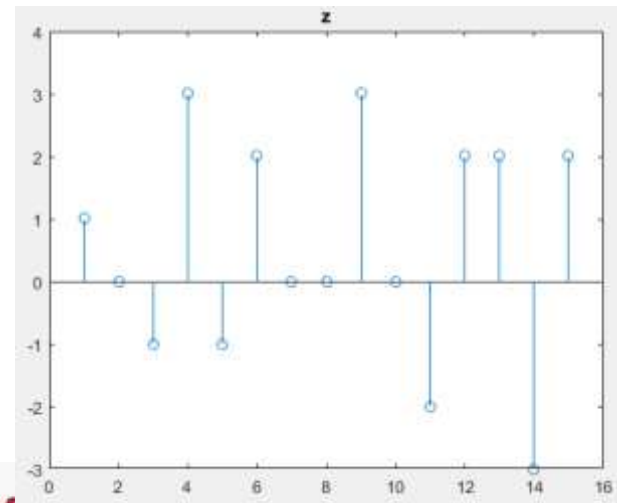
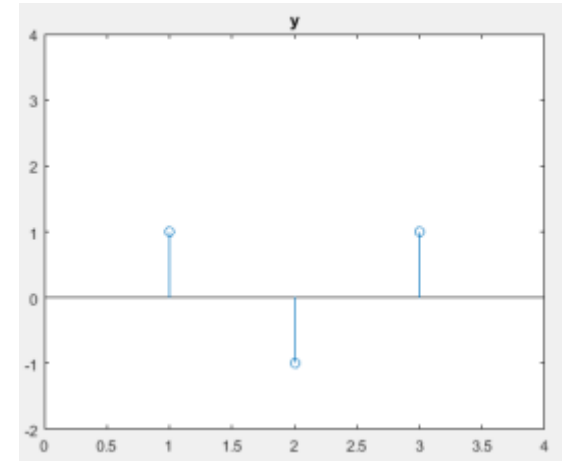
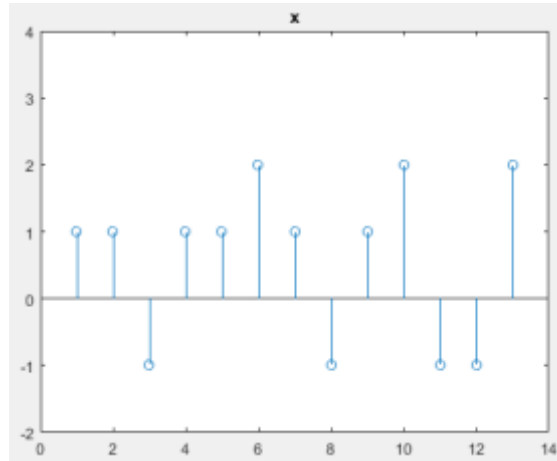
- Muda-se a variável independente;
- Escolhe-se uma das funções que é invertida com respeito ao eixo vertical e defasada no tempo;
- Move-se a ‘função móvel’ com respeito à ‘função fixa’, integrando-se os intervalos de coexistência das funções.

Redes Neurais Convolucionais

(Convolutional Neural Networks) - Convolução

- Convolução discreta:

```
>> x = [1 1 -1 1 1 2 1 -1 1 2 -1 -1 2];  
>> y = [1 -1 1];  
>> z = conv(x,y)
```

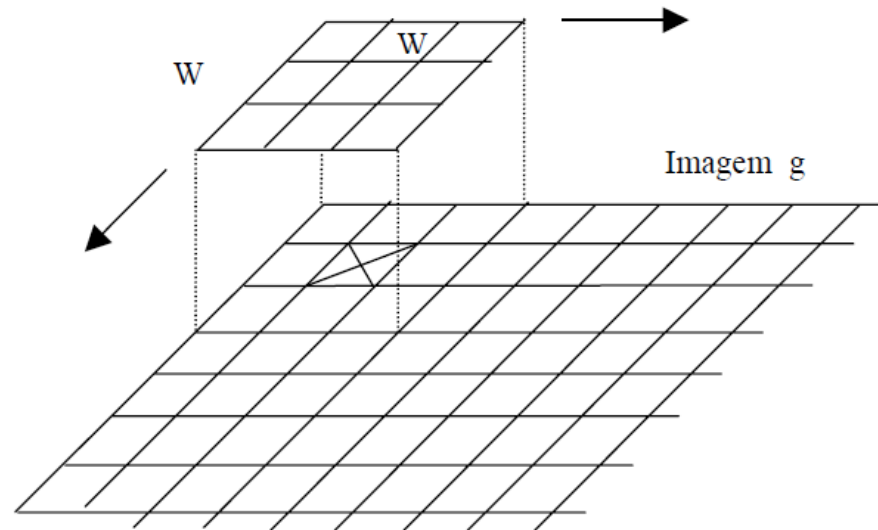


Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Convolução

- Convolução 2D (para imagens):

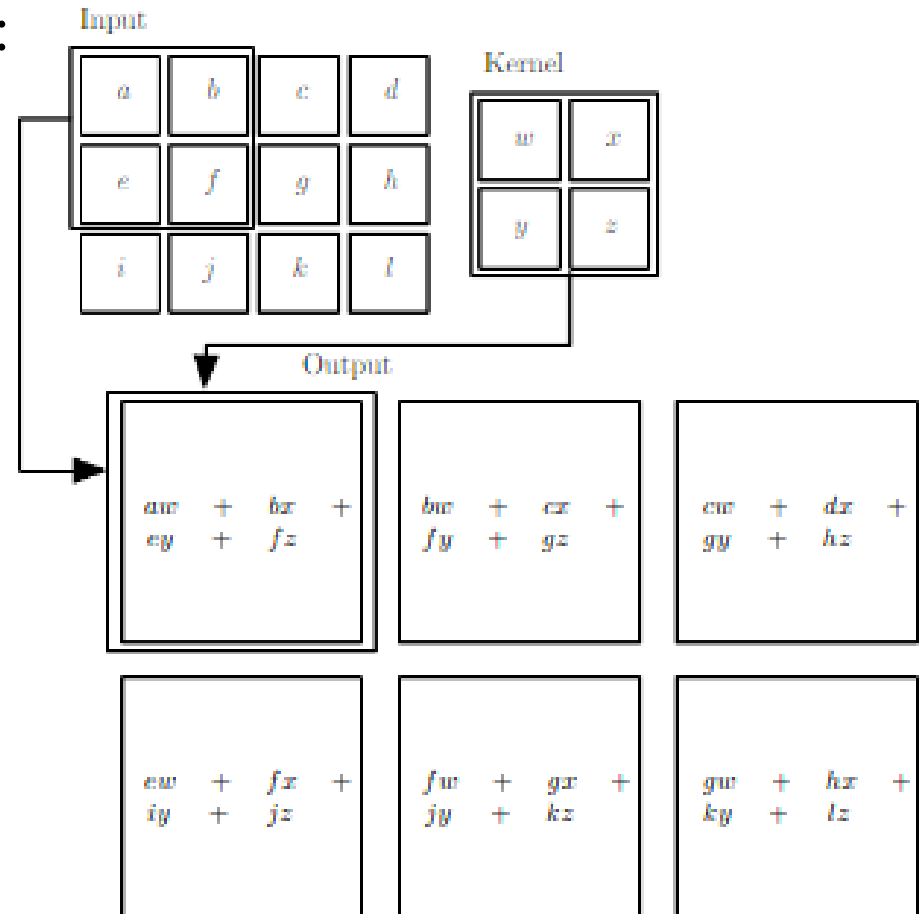
$$g_f(l, p) = \sum_{m=-\text{int}(w/2)}^{\text{int}(w/2)} \sum_{n=-\text{int}(w/2)}^{\text{int}(w/2)} g(l+m, p+n) h(m, n)$$



Redes Neurais Convolucionais

(Convolutional Neural Networks) - Convolução

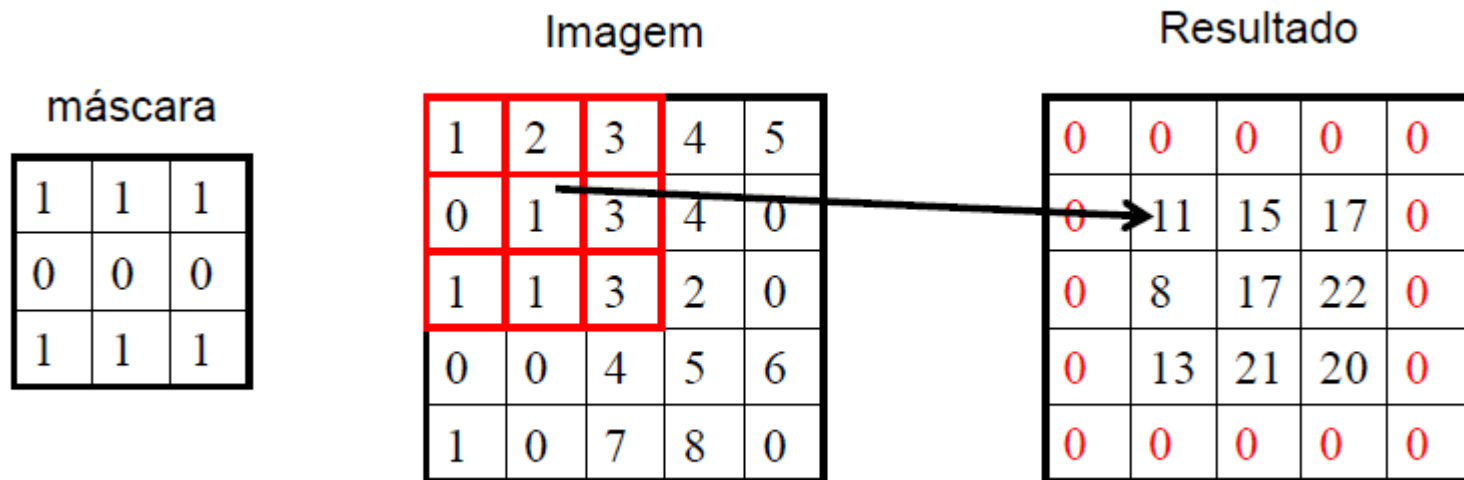
- Convolução 2D (para imagens):



Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Convolução

- Convolução 2D (para imagens):



Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Convolução

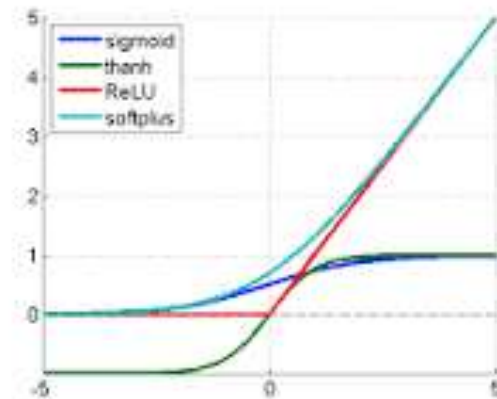
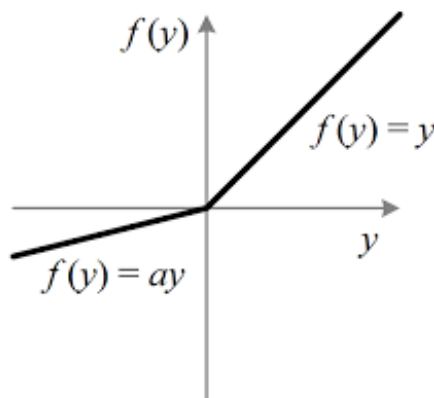
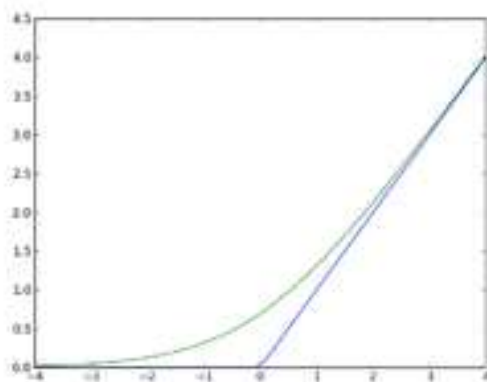


Input

Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Inserção de não linearidade

- Definições das funções de ativação:
 - ReLU: Rectified Linear Unit: $f(x) = \text{Max}(0, x)$, derivada constante ou nula;
 - Leaky ReLU $f(x) = x$ if $x > 0$ else ax para $0 \leq a \leq 1$, assim a derivada fica constante negativa mas não vai para zero;
- Ativação esparsa para produzir uma saída, i.e., muitas unidades escondidas não produzem saída;

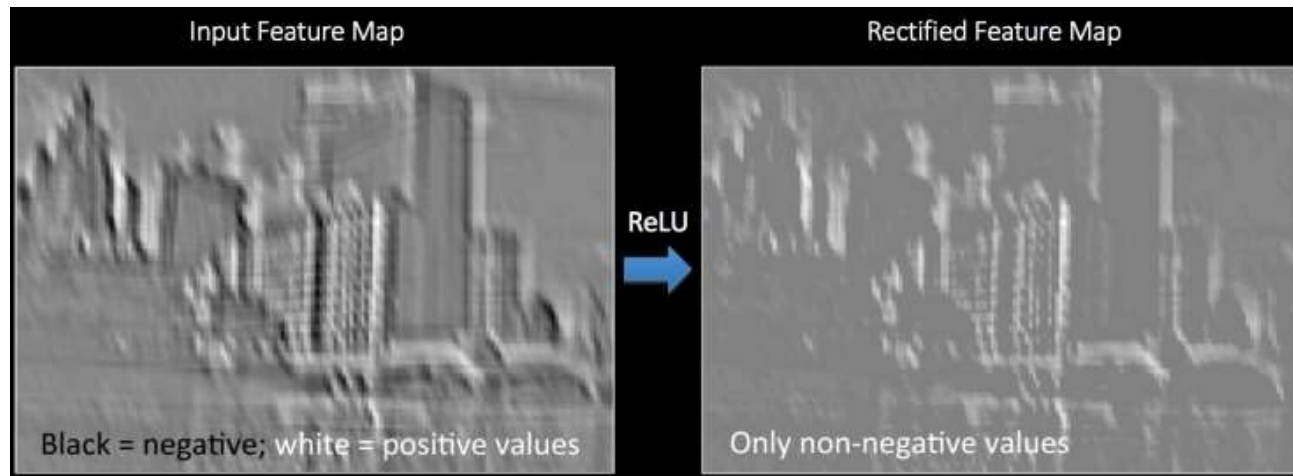
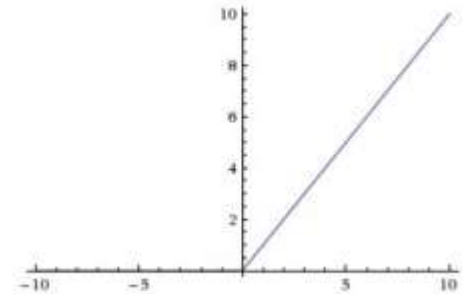


Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Inserção de não linearidade

- Exemplo com ReLU:

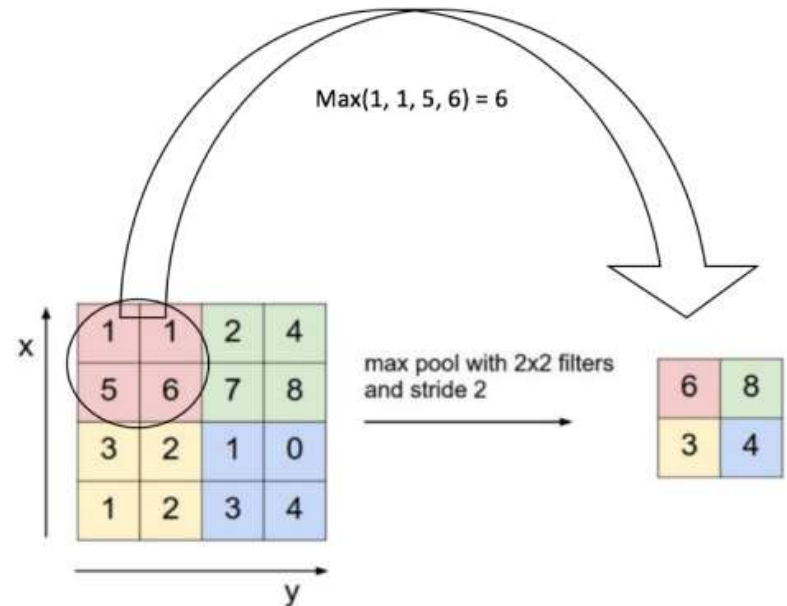
$$\text{Output} = \text{Max}(\text{zero}, \text{Input})$$



Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Subamostragem

- *Pooling* (junção, agrupamento):
 - Esse passo comprime e suaviza os dados;
 - Normalmente toma a média ou o máximo entre trechos disjuntos;
 - Dá robustez a pequenas variações espaciais dos dados.



Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Evidências

- Hubel e Wiesel em 1968, estudaram o sistema visual de felinos e constataram o papel importante das chamadas *Receptive Cells* que agiam sobre como filtros locais sobre o espaço de entrada e tinham dois comportamentos:
 - *Simple Cells*: Respondem a padrões de bordas na imagem;
 - *Complex Cells*: Possuem campos receptivos grandes e são invariantes à posição do padrão.

Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Evidências

A bit of history:

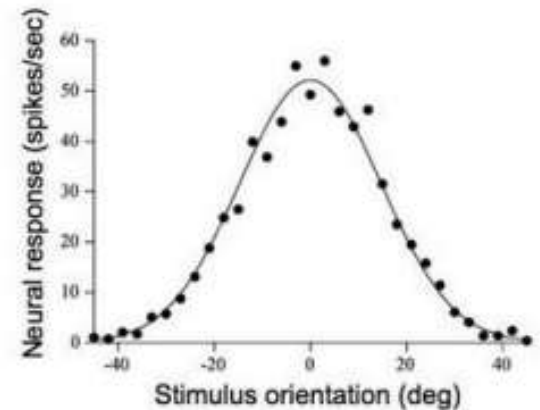
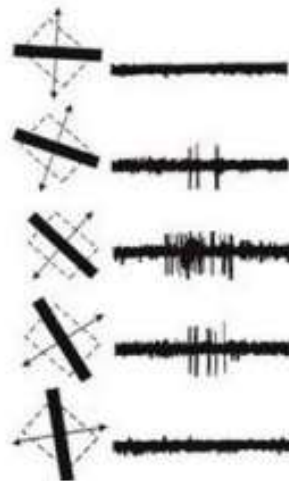
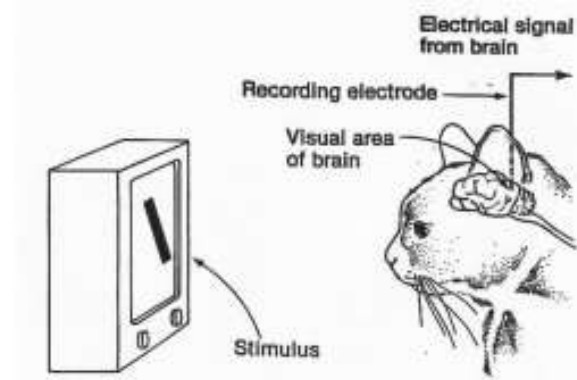
**Hubel & Wiesel,
1959**

RECEPTIVE FIELDS OF SINGLE
NEURONES IN
THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR
INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX

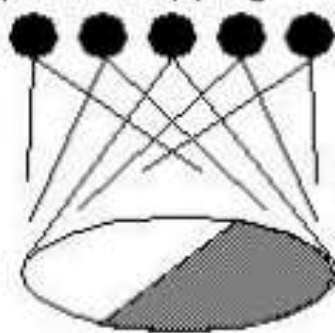
1968...



Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - Evidências

Hubel & Weisel
topographical mapping

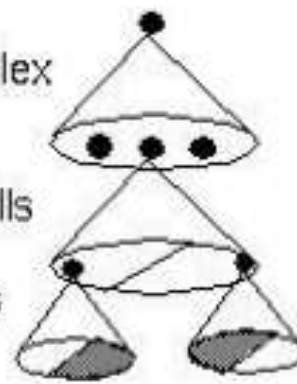


featural hierarchy

hyper-complex cells

complex cells

simple cells



Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - História

- Fukushima (1980) – *Neo-Cognitron*;
- MLP (Rumelhart et al., 1986): marco importante;
- Novos modelos relevantes com SVM, final dos 1990s;
- *Convolutional Neural Nets* (LeCun, 1998): Aplicado a imagens, fala;
 - Apresenta similaridades com *Neo-Cognitron*;
- MLP treinada com BP com muitas camadas:
 - As primeiras tentativas tiveram baixo índice de sucesso pois
 - Eram muito lentas;
 - Havia gradiente que ia para zero.

Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - História

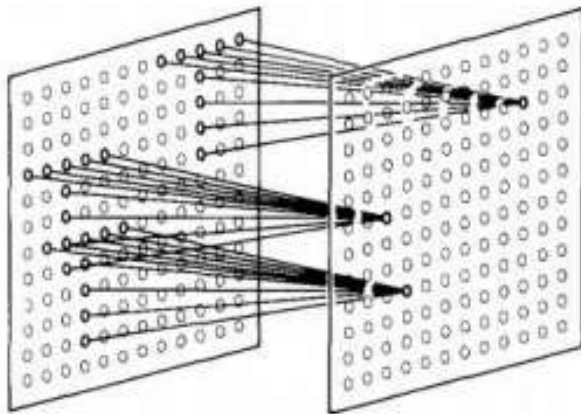
- Trabalhos mais recentes mostraram avanços com treinamento lento de MLPs com BP profundas usando máquinas com GPUs;
- *Deep Belief nets* (Hinton) and *Stacked auto-encoders* (Bengio), ambas em 2006: Pré-treinamento não-supervisionado precede treinamento supervisionado;
- Sucessos iniciais em 2012 com aprendizagem supervisionada que conseguiu superar gradientes que desaparecem e se torna mais aplicável.

Redes Neurais Convolucionais

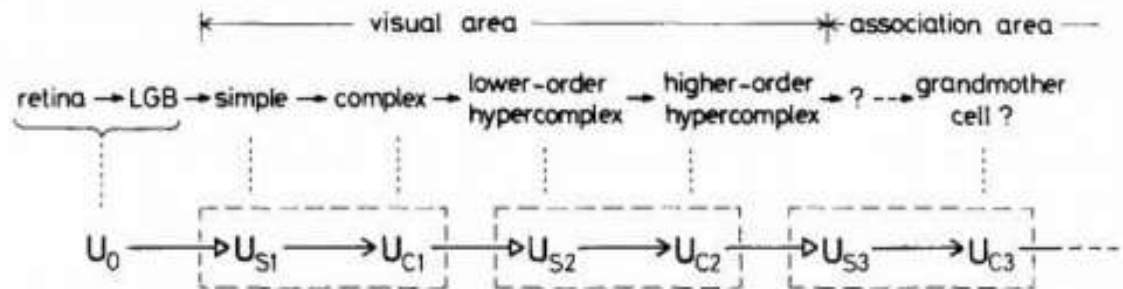
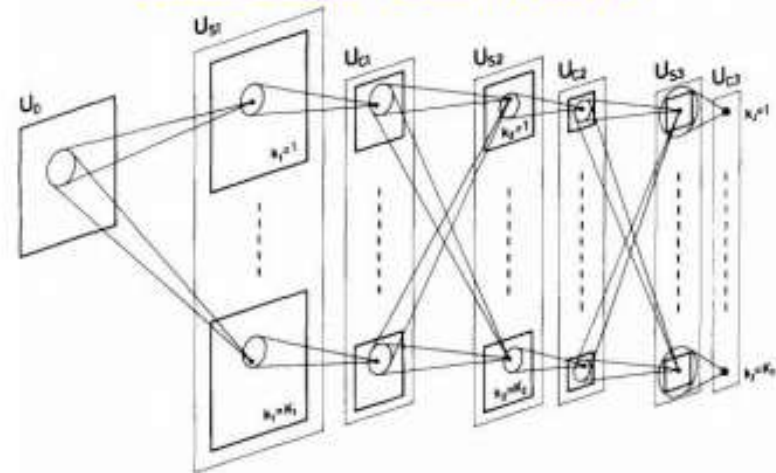
(Convolutional Neural Networks) - História

A bit of history:

Neurocognitron
[Fukushima 1980]



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



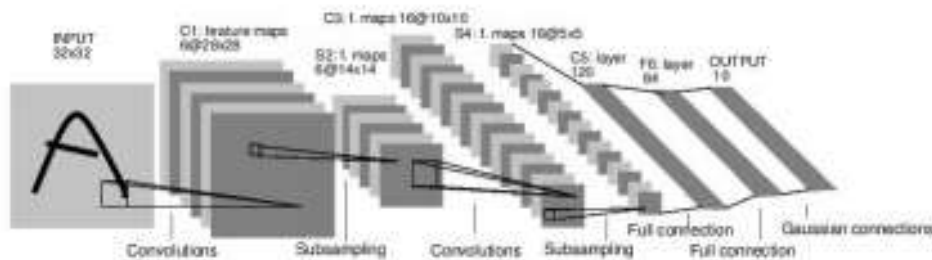
Redes Neurais Convolucionais

(*Convolutional Neural Networks*) - História

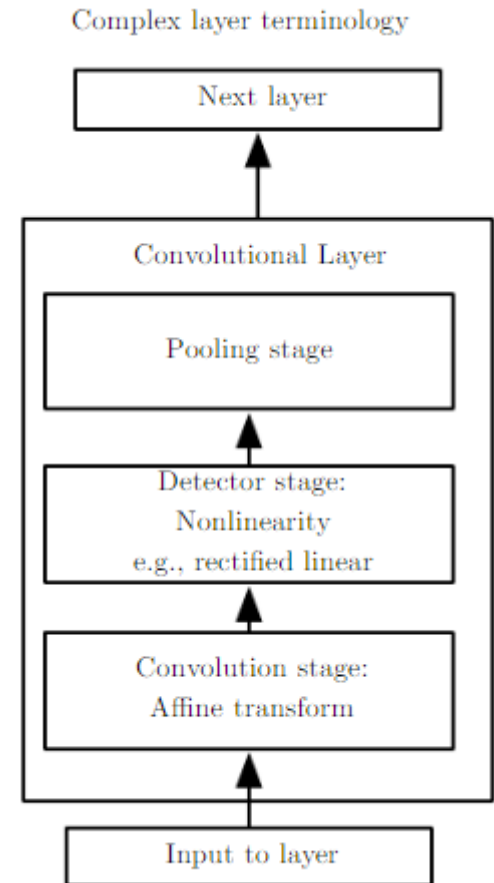
A bit of history:

**Gradient-based learning
applied to document
recognition**

[LeCun, Bottou, Bengio, Haffner
1998]

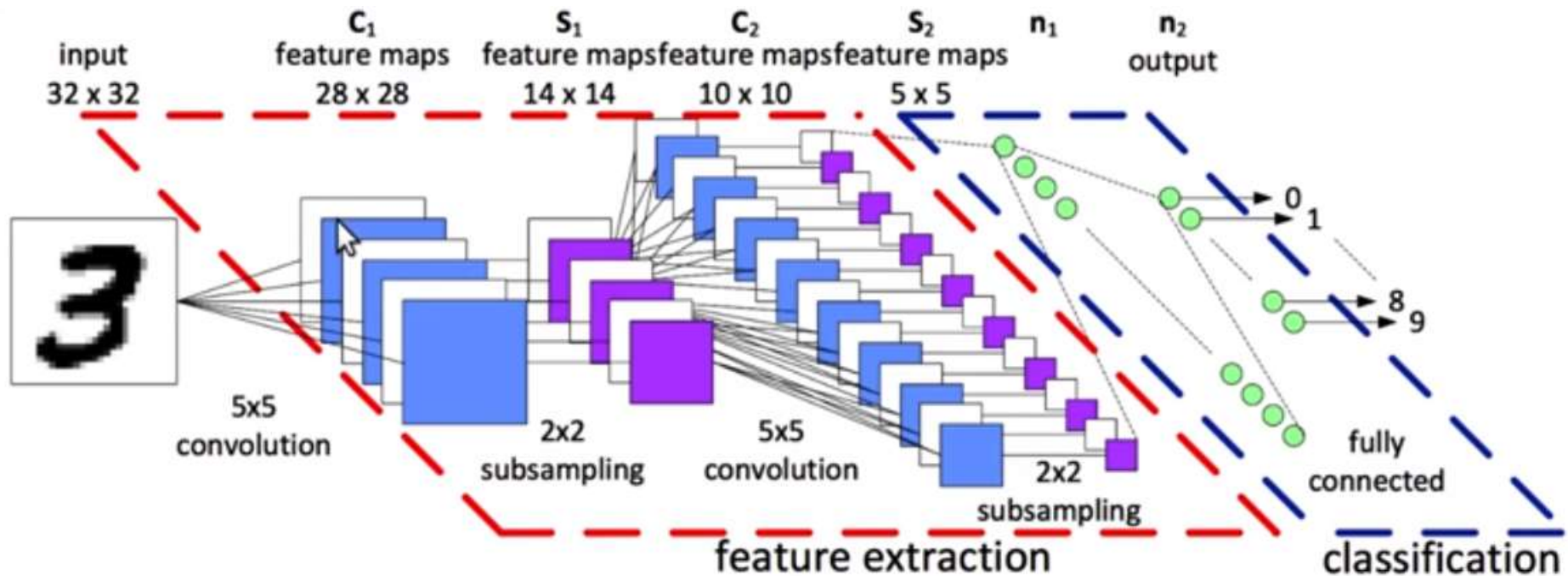


LeNet-5



Redes Neurais Convolucionais

(Convolutional Neural Networks)



Redes Neurais Convolucionais

(Convolutional Neural Networks)

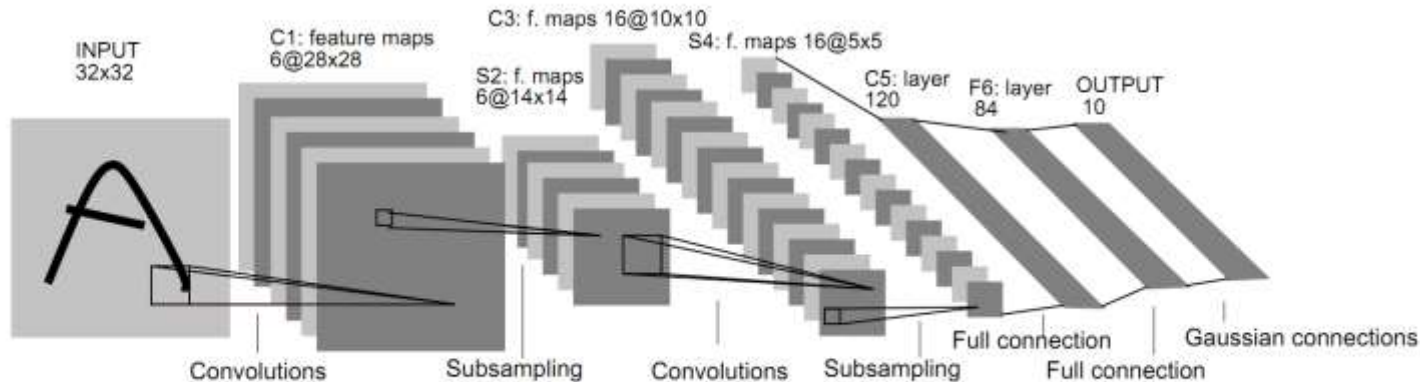
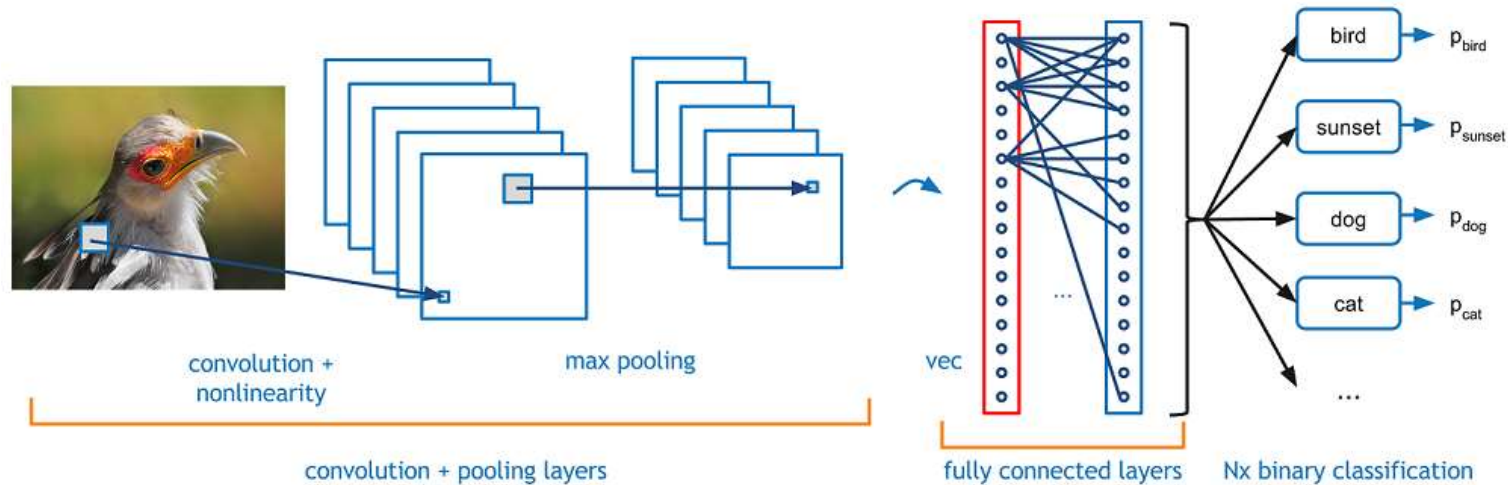


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

- Redes Neurais Convolucionais são uma extensão de MLPs tradicionais a partir de 3 ideias:
 - Campos receptivos locais (*local receive fields*);
 - Pesos compartilhados (*shared weights*);
 - Sub-amostragem espaço-temporal (*Spatial / temporal sub-sampling*).

Artigo de LeCun paper (1998) sobre reconhecimento de texto:

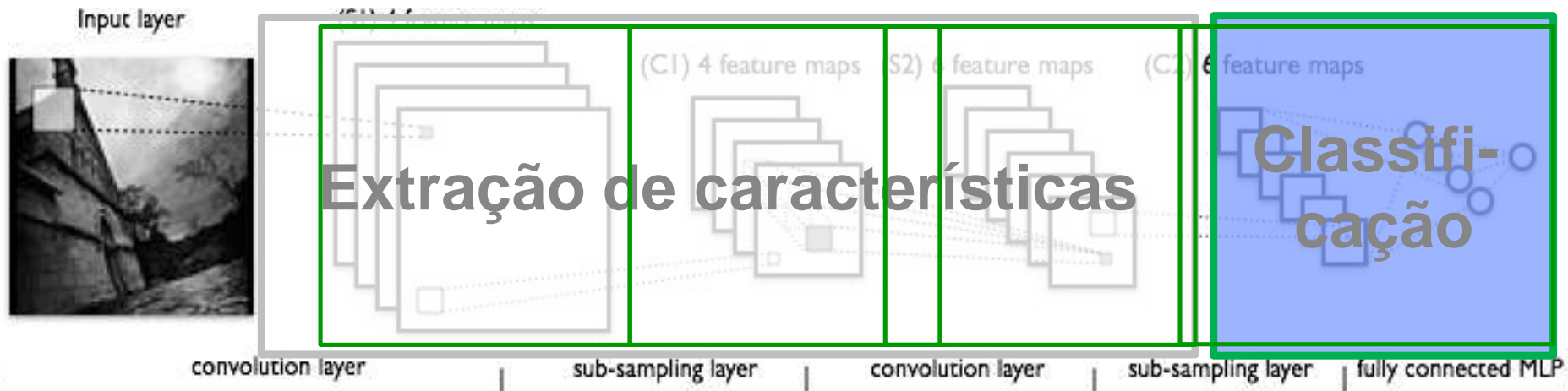
<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

Redes Neurais Convolucionais

(Convolutional Neural Networks)

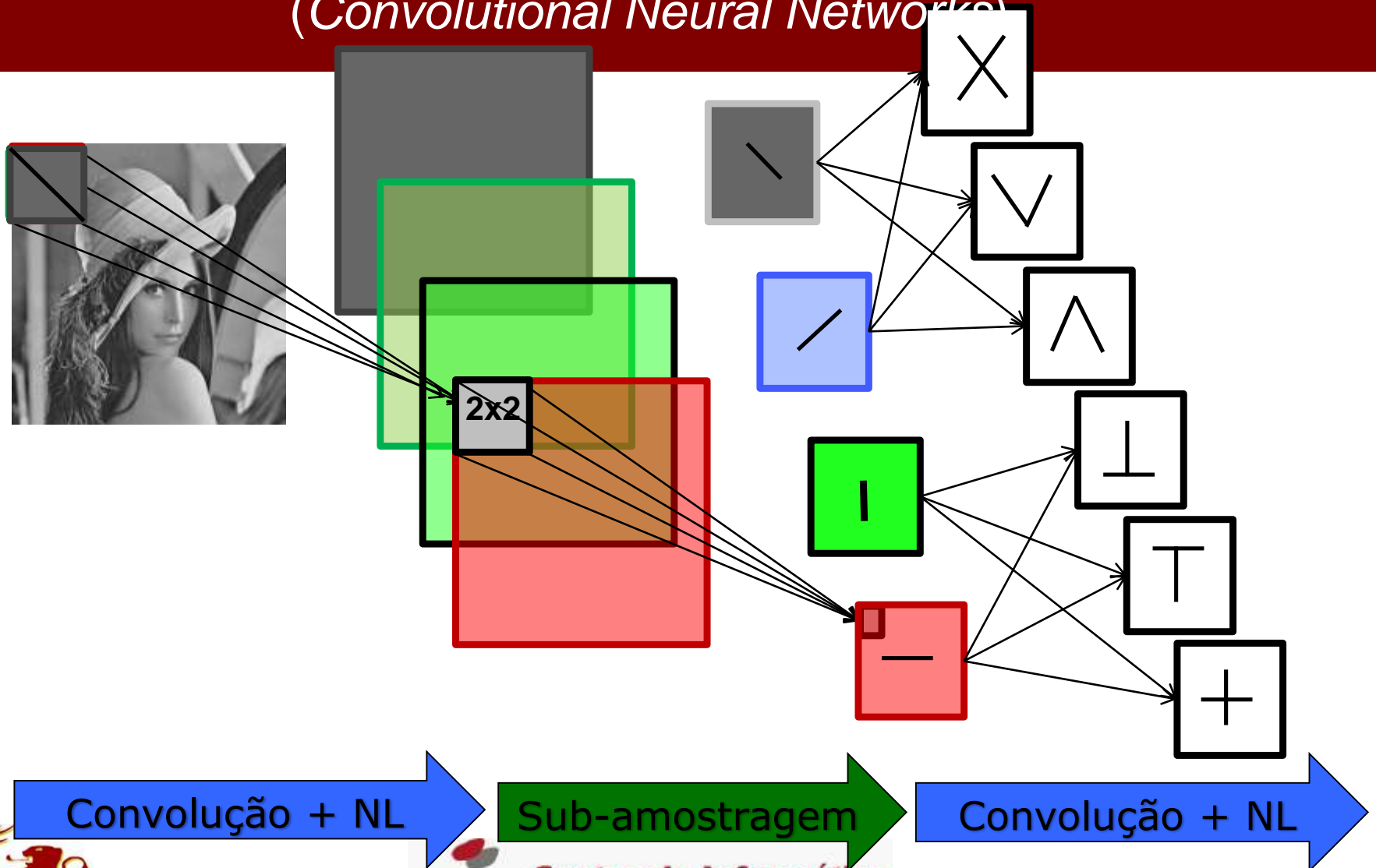
Arquitetura da CNN:

- Camada convolucional + Não-linear (ReLU);
- Camada de subamostragem;
- Camada convolucional + Não-linear (ReLU);
- Linearização da camada + camadas totalmente conectadas de treinamento supervisionado.



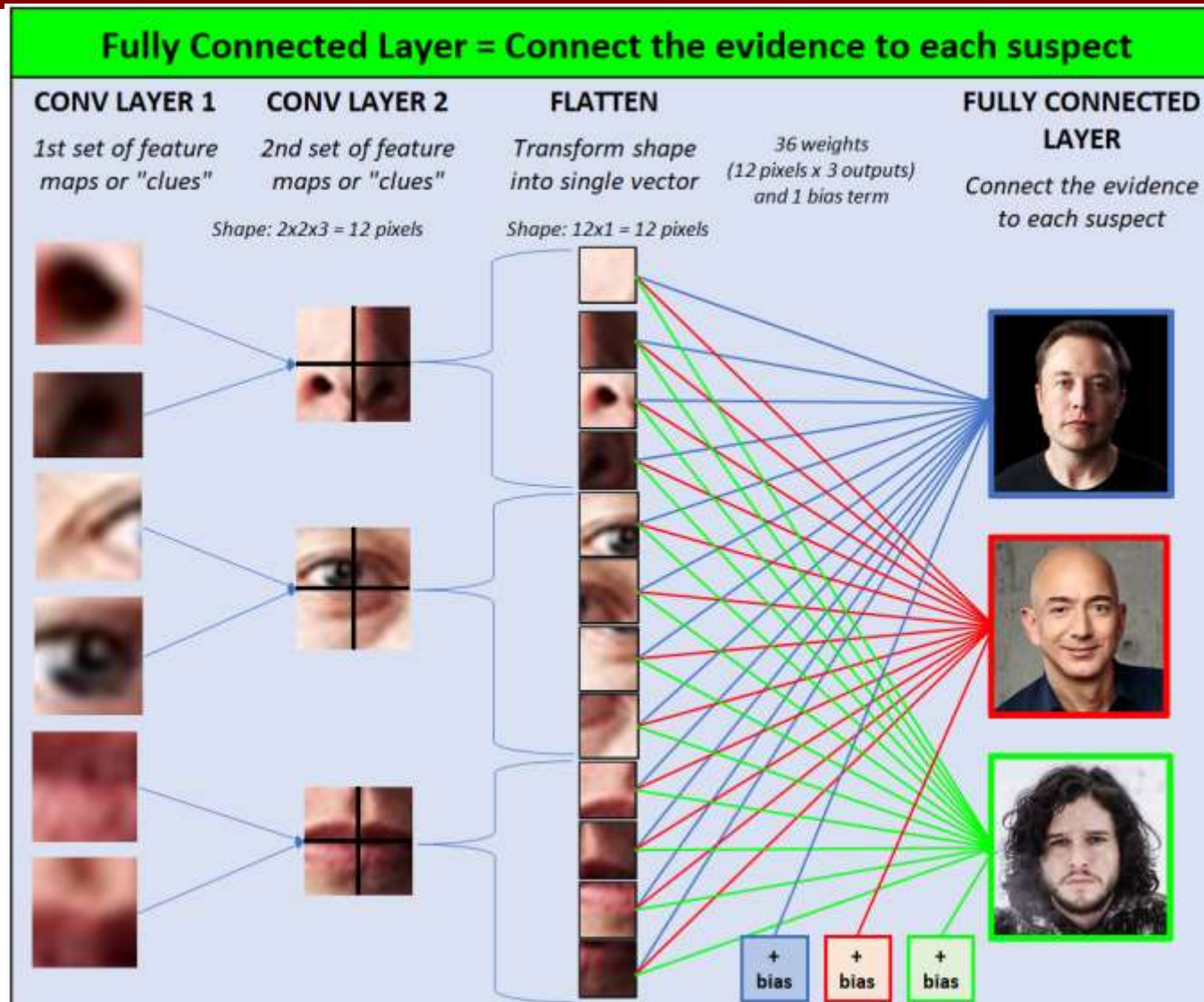
Redes Neurais Convolucionais

(Convolutional Neural Networks)



Redes Neurais Convolucionais

(Convolutional Neural Networks)



Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

- Parâmetros:
 - Margens: (Ignorar/Replicar/Zerar);
 - Tamanho do *kernel*;
 - Tamanho do passo (*stride*);
 - Quantidade de núcleos;
 - Configuração dos núcleos (aprendidos).

Redes Neurais Convolucionais

(Convolutional Neural Networks)

- Conectividade esparsa: CNN explora correlações espaciais focando em conectividade entre unidades de processamento próximas. Os campos receptivos são contíguos.
 - Os nodos são insensíveis às variações fora do seu campo receptivo.
- Analogia (quando a entrada são imagens):
 - Pixels -> Neurônios;
 - Kernel -> Sinapses;
 - Convolução -> Operação básica de um neurônio.

Redes Neurais Convolucionais

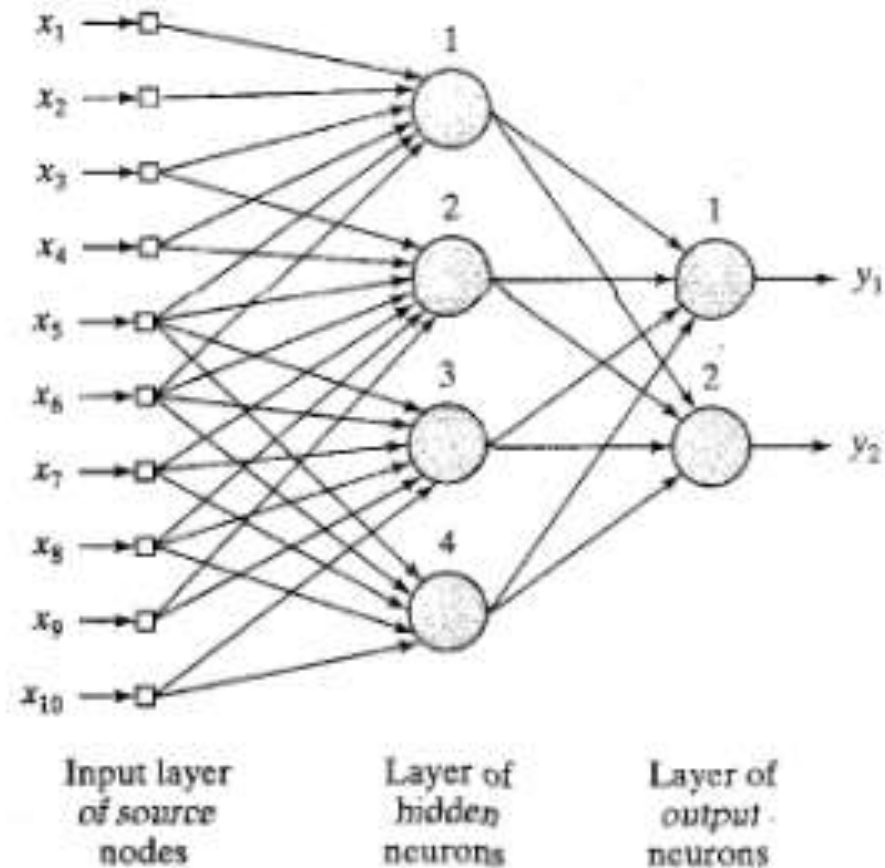
(Convolutional Neural Networks)

- Procedimento *ad-hoc* para considerar conhecimento prévio no projeto (*design*) de uma RNA:
 - Restringir a arquitetura da rede através do uso de conexões locais conhecidas como campos receptivos;
 - Limitar a escolha dos pesos sinápticos através do uso do compartilhamento de pesos.
- Essas duas técnicas, especialmente a segunda, têm um efeito colateral benéfico:
 - Número de parâmetros livres na rede é reduzido significativamente.

Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

- Seja a rede *feedforward* parcialmente conectada cuja arquitetura é restringida por construção:
 - Por exemplo, os 6 primeiros nodos de entrada constituem o campo receptivo para o neurônio escondido 1.



Redes Neurais Convolucionais

(Convolutional Neural Networks)

- Para satisfazer a restrição do compartilhamento de pesos, deve-se usar o mesmo conjunto de pesos sinápticos (a janela de convolução) para cada um dos neurônios da camada escondida.
- Portanto, para seis conexões locais por nodo escondido e quatro nodos escondidos (figura anterior), pode-se expressar o campo local induzido do neurônio escondido j como (soma de convolução):

$$v_j = \sum_{i=1}^6 w_i x_{i+j-1}, \quad j = 1, 2, 3, 4$$

- Onde $\{w_i\}_{i=1}^6$, constitui o mesmo conjunto de pesos compartilhados por todos os quatro nodos escondidos e x_k é o sinal do nó fonte $k=i+j$.

Redes Neurais Convolucionais

(Convolutional Neural Networks)

- Uma rede CNN é uma MLP projetada para reconhecer formas bidimensionais com um alto grau de invariância para translação, mudança de escala, e outras formas de distorção.
- Esta tarefa difícil é aprendida de maneira supervisionada por uma rede cuja estrutura inclui as seguintes operações:
 - Extração de características;
 - Mapeamento de características;
 - Sub-amostragem.

Redes Neurais Convolucionais

(Convolutional Neural Networks)

- Extração de características: Cada nodo recebe entradas de estímulos em um campo receptivo local, saídas da camada anterior, processando apenas características locais. A posição relativa de cada característica extraída em relação às outras é preservada.
- Mapeamento de características: Cada camada computacional da rede é composta de múltiplos mapas de características. Cada um forma um conjunto no qual os nodos individuais compartilham o mesmo conjunto de pesos sinápticos (janela para convolução).

Redes Neurais Convolucionais

(Convolutional Neural Networks)

- Os filtros são aprendidos pelo algoritmo;
- A inicialização dos filtros é aleatória;
- O pesos dos filtros são os mesmo em qualquer uma das regiões do mapa;
- A convolução é linear;
- Facilita o paralelismo do processo.

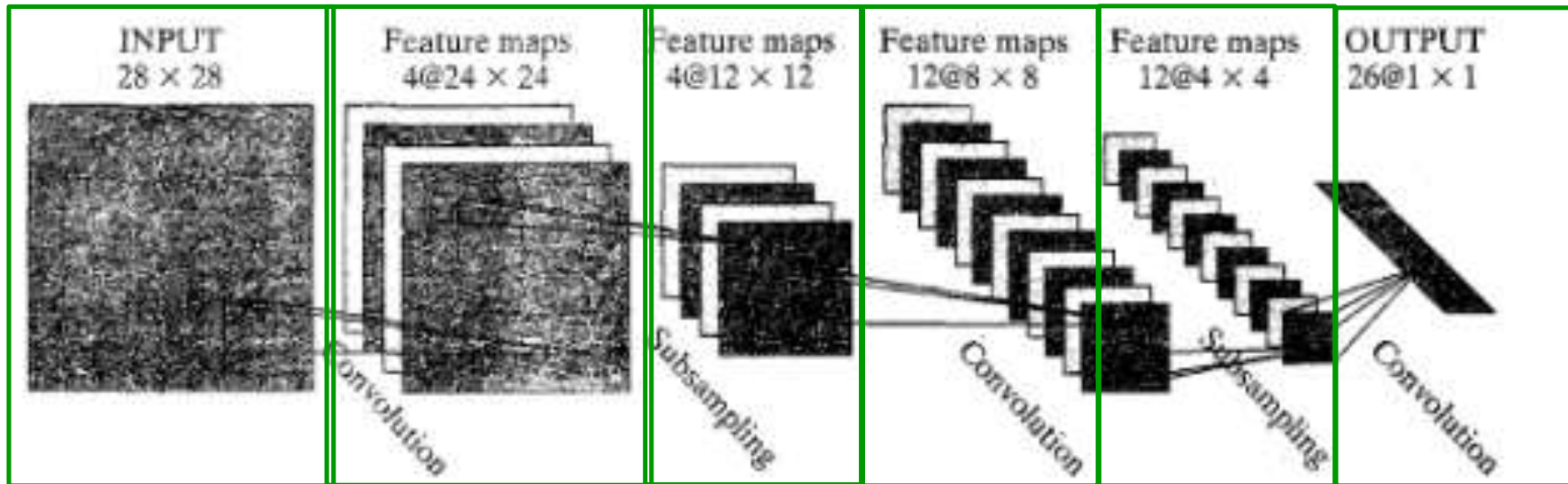
Redes Neurais Convolucionais

(Convolutional Neural Networks)

- Sub-amostragem: Cada camada da CNN é seguida por uma camada computacional que realiza cálculo da média local (ou determinação do valor mais alto) e sub-amostragem, onde a resolução do mapa de características é reduzida. Isto busca reduzir a sensibilidade da saída do mapa a deslocamentos e outras formas de distorção.
- Todos os pesos em todas as camadas de uma CNN são aprendidos através do treinamento.
- No entanto, a rede aprende a extrair suas próprias características automaticamente.

Redes Neurais Convolucionais

(Convolutional Neural Networks)

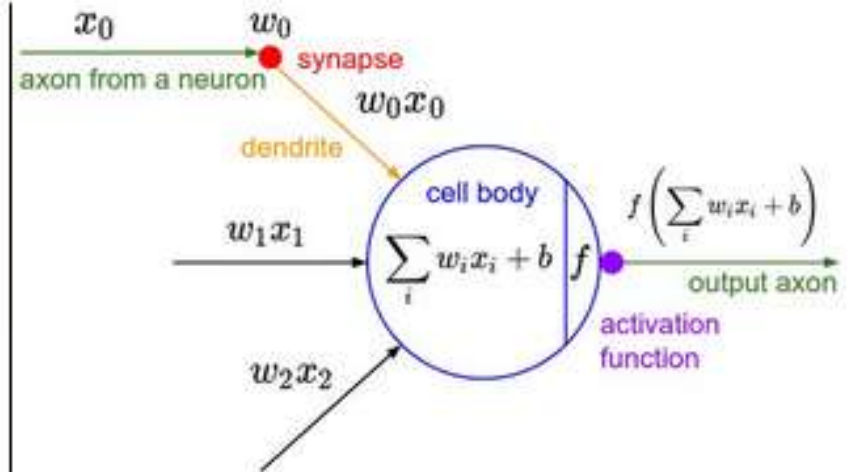
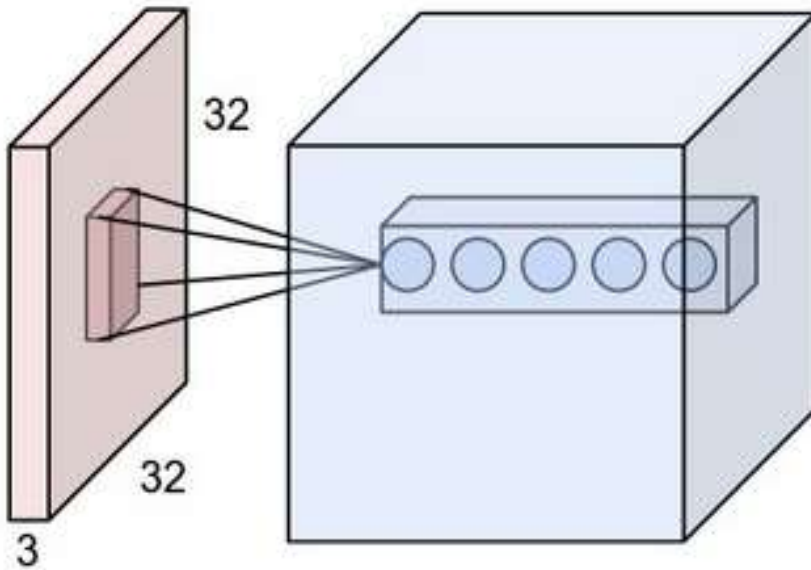


- Camada de saída (convolução) que mapeia as características da imagem para a saída final. A camada de saída é composta por 26 neurônios, cada um representando uma classe de saída.

Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

- |Camada convolutional:
 - Exemplo ilustrativo com uma imagem com 3 filtros (RGB):
 - Coletar a imagem e trabalhar com o hipervolume;



Redes Neurais Convolucionais

(Convolutional Neural Networks)

- |Camada convolutional:

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.

Input Volume (+pad 1) (7x7x3)

$x[:, :, 0]$

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0

0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

$x[:, :, 1]$

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0

0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

$x[:, :, 2]$

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0

0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

$w0[:, :, 0]$

-1	0	1
0	0	1
1	-1	1

$w0[:, :, 1]$

-1	0	1
1	-1	1
0	1	0

$w0[:, :, 2]$

-1	1	1
1	1	0
0	-1	0

Bias $b0$ (1x1x1)

$b0[:, :, 0]$

1

Filter W1 (3x3x3)

$w1[:, :, 0]$

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

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

Bias $b1$ (1x1x1)

$b1[:, :, 0]$

0

Output Volume (3x3x2)

$o[:, :, 0]$

2	3	3
3	7	3
8	10	-3

$o[:, :, 1]$

-8	-8	-3
-3	1	0
-3	-8	-5

$$O_{11} = x[:, :, 0] \cdot w0 + b0$$

toggle movement

$$W_1 = 5, H_1 = 5, D_1 = 3,$$

$$K = 2, F = 3, S = 2, P = 1.$$

Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

- |Camada de *pooling*:

Accepts a volume of size $W_1 \times H_1 \times D_1$

Requires two hyperparameters:

- their spatial extent F ,
- the stride S ,

Produces a volume of size $W_2 \times H_2 \times D_2$ where:

- $W_2 = (W_1 - F)/S + 1$
- $H_2 = (H_1 - F)/S + 1$
- $D_2 = D_1$

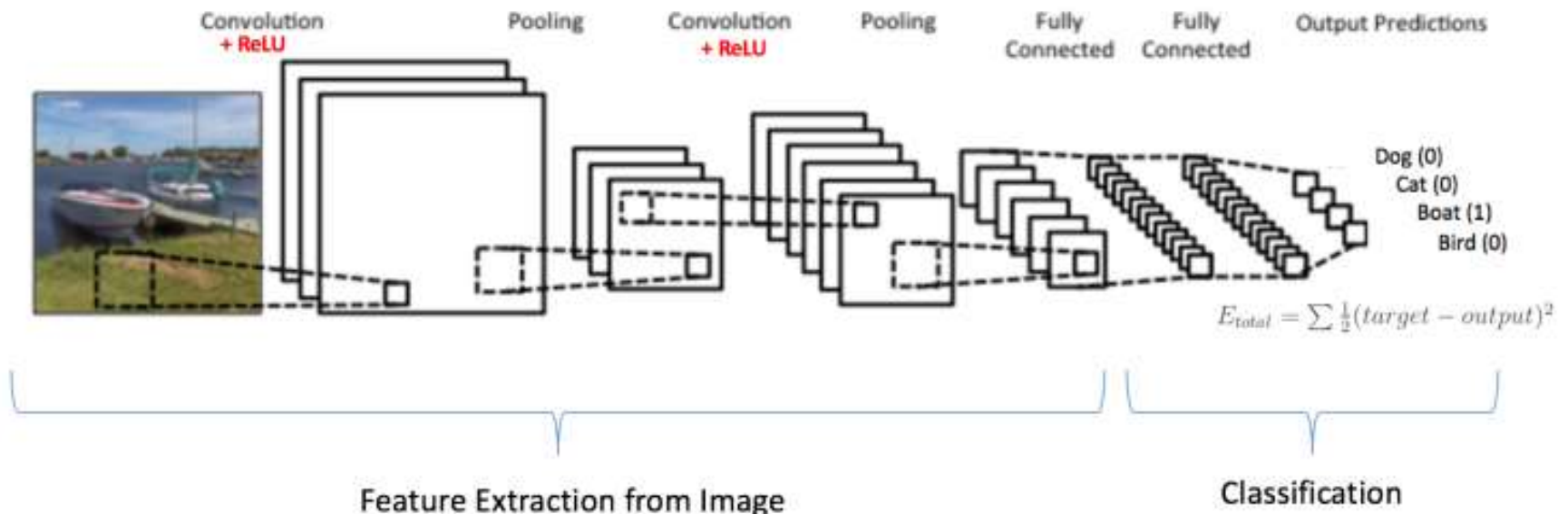
Introduces zero parameters since it computes a fixed function of the input

Note that it is not common to use zero-padding for Pooling layers

Redes Neurais Convolucionais

(Convolutional Neural Networks)

- Exemplo: Camada convolucional de retropropagação
 - Input Image = Boat
 - Target Vector = [0, 0, 1, 0]



Redes Neurais Convolucionais

(Convolutional Neural Networks)

Passo 1: Inicializamos todos os filtros e parâmetros / pesos com valores aleatórios;

Passo 2: A rede recebe uma imagem de treinamento como entrada, passa pela etapa de propagação direta (convolução, ReLU e operações de agrupamento junto com a propagação direta na camada totalmente conectada) e localiza as probabilidades de saída para cada classe;

Vamos dizer que as probabilidades de saída para a imagem do barco acima são [0.2, 0.4, 0.1, 0.3]

Como os pesos são atribuídos aleatoriamente para o primeiro exemplo de treinamento, as probabilidades de saída também são aleatórias de início.

Passo 3: Calcular o erro total na camada de saída (soma das 4 classes)

$$\text{Erro total} = \sum \frac{1}{2} (\text{probabilidade desejada} - \text{probabilidade de saída})^2$$

Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

Passo 4: Use retropropagação para calcular os gradientes do erro em relação a todos os pesos na rede e use o gradiente descendente para atualizar todos os valores / pesos de filtro e valores de parâmetros para minimizar o erro de saída. Os pesos são ajustados proporcionalmente à sua contribuição no erro total.

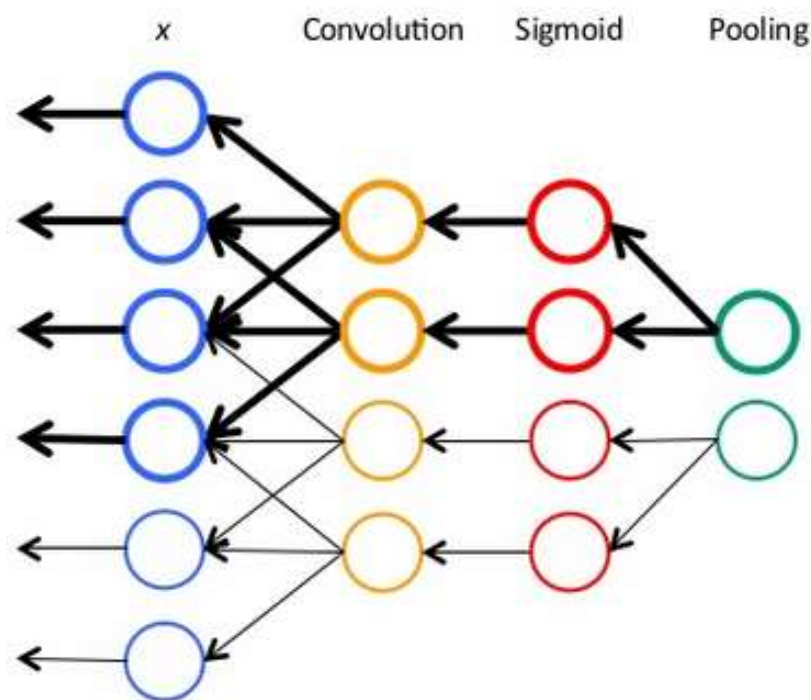
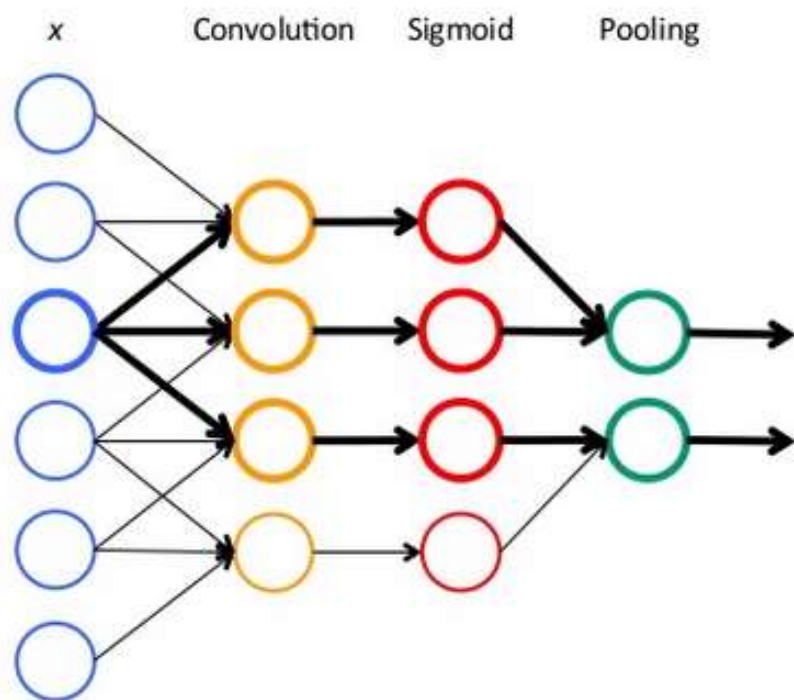
Isso significa que a rede aprendeu a classificar corretamente essa imagem específica ajustando seus pesos / filtros de forma que o erro de saída seja reduzido.

Outros parâmetros (número de filtros, tamanhos de filtros, arquitetura da rede) foram ajustados antes da Etapa 1 e não são alterados durante o treinamento - somente os valores dos pesos de conexão são atualizados.

Passo 5: Repita os passos 2-4 com todas as imagens no conjunto de treinamento.

Redes Neurais Convolucionais

(*Convolutional Neural Networks*)



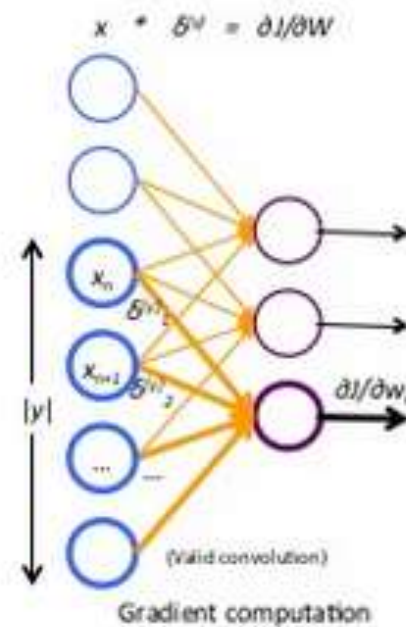
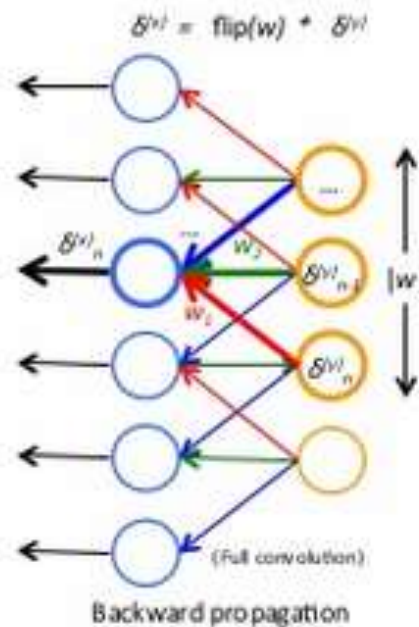
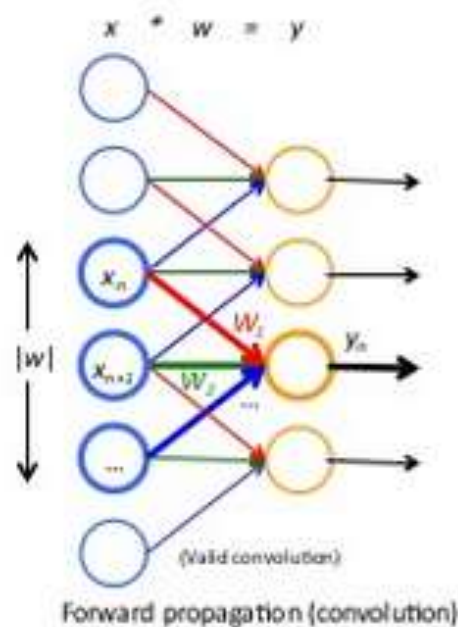
Redes Neurais Convolucionais

(Convolutional Neural Networks)

$$\delta_n^{(i)} = \frac{\partial J}{\partial x_n} = \frac{\partial J}{\partial y} \frac{\partial y}{\partial x_n} = \sum_{j=1}^{|w|} \frac{\partial J}{\partial y_{n-j+1}} \frac{\partial y_{n-j+1}}{\partial x_n} = \sum_{j=1}^{|w|} \delta_{n-j+1}^{(i)} w_j = \left(\delta^{(i)} * \text{flip}(w) \right)[n], \delta^{(i)} = \left[\delta_n^{(i)} \right] = \delta^{(i)} * \text{flip}(w)$$

↑ Reverse order linear combination

$$\frac{\partial J}{\partial w_j} = \frac{\partial J}{\partial y} \frac{\partial y}{\partial w_j} = \sum_{n=1}^{|x|+|w|-1} \frac{\partial J}{\partial y_n} \frac{\partial y_n}{\partial w_j} = \sum_{n=1}^{|x|+|w|-1} \delta_n^{(i)} x_{n+j} = \left(\delta^{(i)} * x \right)[j], \frac{\partial J}{\partial w} = \left[\frac{\partial J}{\partial w_j} \right] = \delta^{(i)} * x = x * \delta^{(i)}$$



Redes Neurais Convolucionais

(*Convolutional Neural Networks*)

- Os ajustes dos parâmetros livres são feitos usando uma forma estocásticas (sequencial) do aprendizado *back-propagation*.
- O uso do compartilhamento de pesos torna possível implementar a CNN de forma paralela: outra vantagem sobre a MLP totalmente conectada.

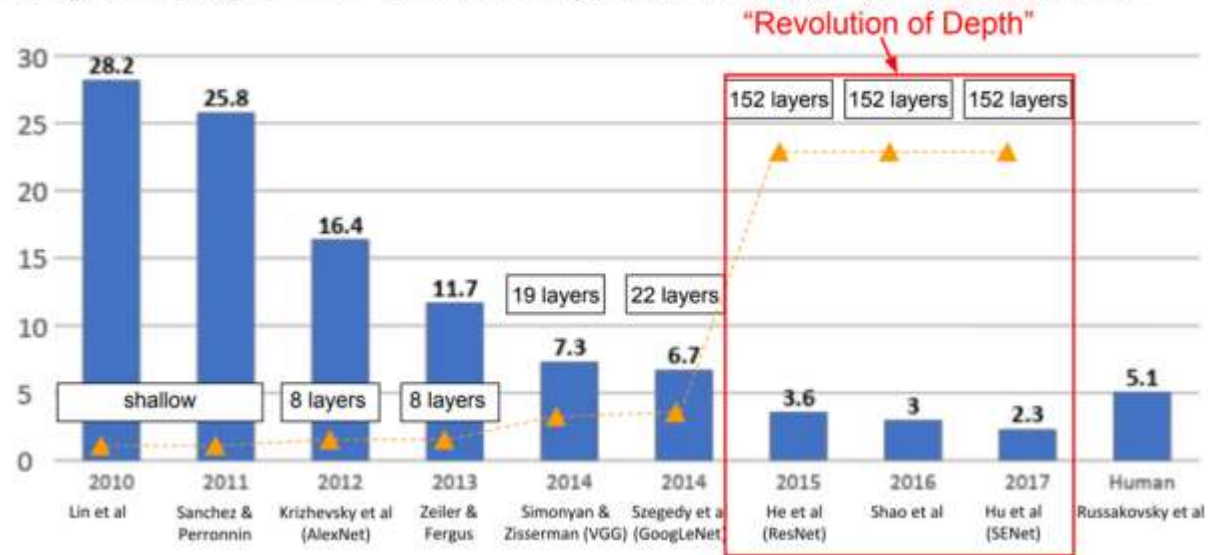
Redes Neurais Convolucionais

(Convolutional Neural Networks)

- CNNs de sucesso:

- LeNet, 1998
- AlexNet, 2012
- VGGNet, 2014
- ResNet, 2015

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



- ILSVRC (<https://www.image-net.org/challenges/LSVRC/>):
 - Competição que avalia algoritmos para detecção de objetos e classificação de imagens em alta escala. a competição procura verificar avanços na detecção de vários objetos.

Demo

- <http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>
- <http://www.cs.toronto.edu/~kriz/cifar.html>

Ferramentas

Software	Creator	Platform	Written in	Interface
Apache MXNet	Apache Software Foundation	Linux, macOS, Windows,AWS, Android,iOS, JavaScript	Small C++core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl
Apache SINGA	Apache Incubator	Linux, macOS, Windows	C++	Python, C++, Java
Caffe	Berkeley Vision and Learning Center	Linux, macOS, Windows	C++	Python, MATLAB, C++
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python(Keras), Kotlin
Intel Data Analytics Acceleration Library	Intel	Linux, macOS, Windows on Intel CPU	C++, Python, Java	C++, Python, Java[10]
Keras	François Chollet	Linux, macOS, Windows	Python	Python, R
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan	Linux, macOS	Python, C, CUDA	Python
TensorFlow	Google Brainteam	Linux, macOS, Windows,Android	C++, Python, CUDA	Python (Keras), C/C++, Java, Go, R, Julia
Theano	Université de Montréal	Cross-platform	Python	Python (Keras)
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	Linux, macOS, Windows,Android, iOS	C, Lua	Lua, LuaJIT,C, utility library for C++/OpenCL

Referências

- Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H., (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19: 153.
- Deep Learning Tutorial. LISA Lab, University of Montreal.
- Du, K.-L. & Swamy M. N. S. (2019). *Neural Networks and Statistical Learning*. Springer, 2nd edition.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61: 85-117.
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A, (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 11: 3371-3408.
- <https://cs.nju.edu.cn/wujx/paper/CNN.pdf>
- <https://arxiv.org/pdf/1812.05069.pdf>