



UNIVALI

Universidade do Vale do Itajaí
Escola Politécnica

Redes Neurais Convolucionais

Prof. Felipe Viel

Baseado no material dos Profs. Eric Antonelo, Danilo Silva e Ian Goodfellow

Agenda

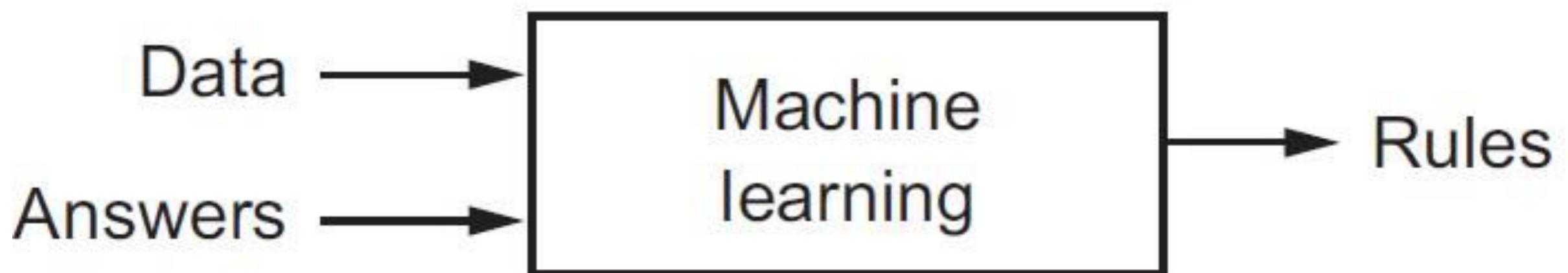
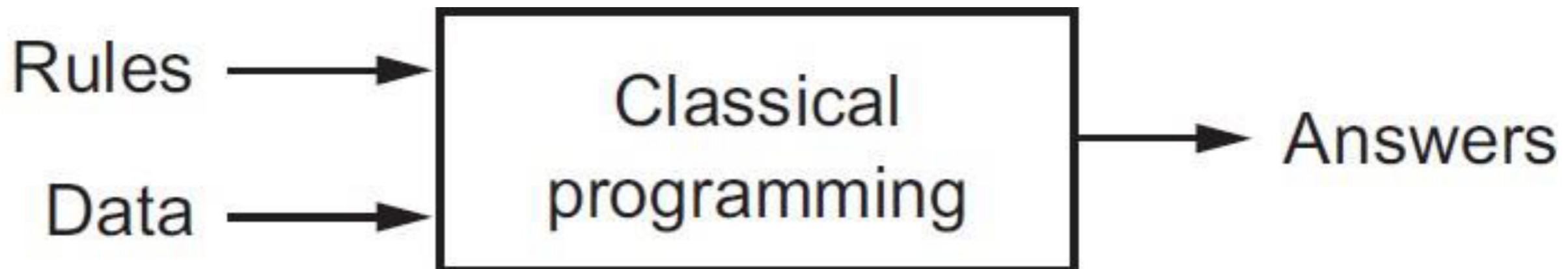
- Introdução à Machine Learning
- Conceito de Rede Neural
- Rede neural convolucional
- Convolução + Exemplo

Introdução

- Aprendizado de máquina (*machine learning*) é um campo de estudo voltado ao projeto e análise de métodos computacionais para realizar tarefas sem necessitar de instruções explícitas
- Aprendizado refere-se à capacidade de um programa de computador de melhorar seu desempenho em uma dada tarefa a partir da experiência
- Experiência refere-se à observação de exemplos (conjunto de variáveis observadas) e/ou de feedback (recompensa/punição) sobre seu desempenho na tarefa
- Exemplo: reconhecimento de faces
 - Difícil descrever ou programar
 - Fácil a partir de exemplos

- Abordagem tradicional de engenharia:
 - Análise do problema
 - Definição de um modelo matemático
 - Soluções são criadas a partir do modelo
- Abordagem do aprendizado de máquina:
 - Coletar dados (exemplos) que relacionam entrada e saída desejada
 - Definir um métrica de desempenho
 - Treinar um algoritmo de aprendizado genérico para executar a tarefa
- Motivação: Em muitas situações, dados + computação podem produzir uma solução em menos tempo e com menor custo do que contratar especialistas para resolver o problema

Introdução



Tarefas adequadas para aprendizado de máquina

- Critérios sugeridos por Brynjolfsson & Mitchell (2017)
 - A tarefa envolve uma função que mapeia entradas bem definidas para saídas bem definidas
 - Grandes conjuntos de dados existem ou podem ser criados contendo pares de entrada-saída
 - A tarefa fornece feedback claro com metas e métricas claramente definíveis
 - A tarefa não envolve longas cadeias de lógica ou raciocínio que dependam de diversos conhecimentos prévios ou bom senso
 - A tarefa não requer explicações detalhadas sobre como a decisão foi tomada
 - A tarefa tem tolerância para erros e não precisa de soluções comprovadamente corretas ou ótimas
 - O fenômeno ou função que está sendo aprendida não deve mudar rapidamente ao longo do tempo
 - Nenhuma destreza especializada, habilidades físicas ou mobilidade são necessárias.

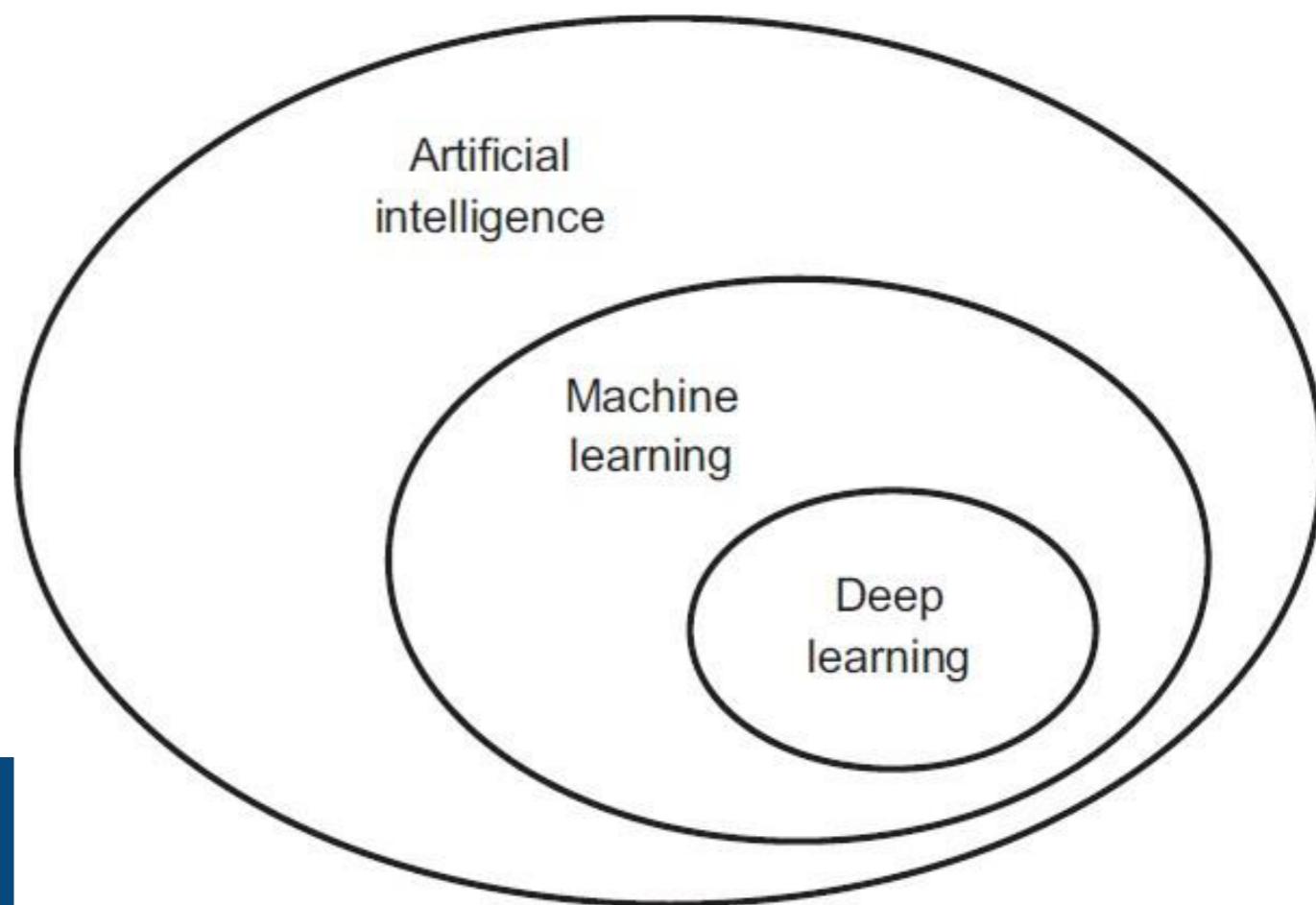
Artificial Intelligence / Machine Learning / Deep Learning

- Aprendizado de máquina:

- Difere da inteligência artificial simbólica clássica (baseada em regras lógicas e buscas estruturadas), por permitir o aprendizado a partir de dados
- Difere da estatística convencional apenas pelo enfoque computacional e em modelos que fazem uso de um grande volume de dados

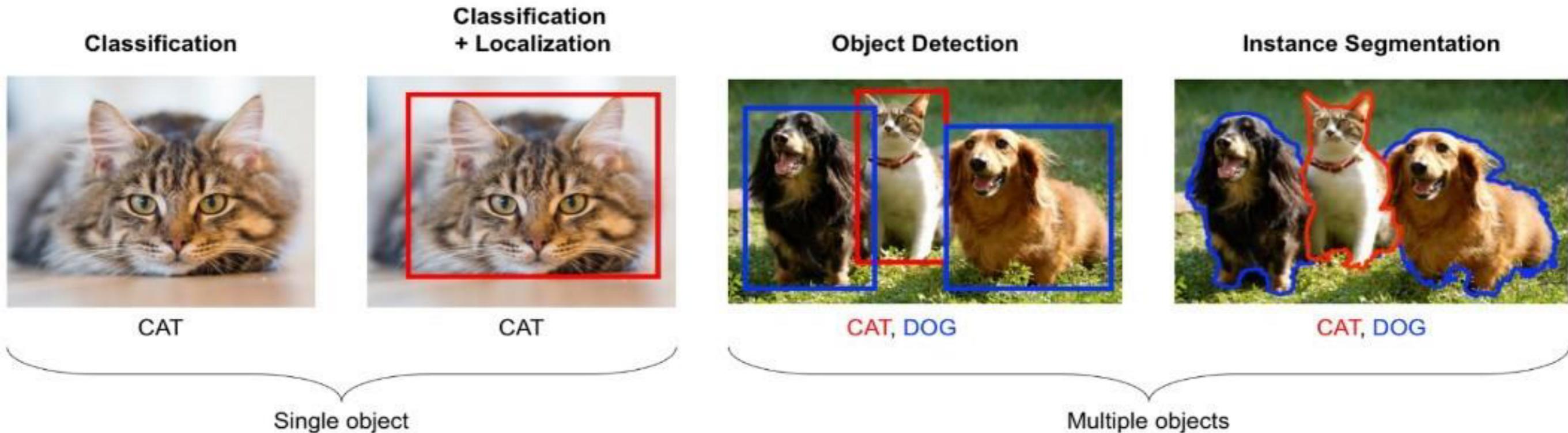
- Aprendizado profundo (deep learning):

- Refere-se a redes neurais com múltiplas camadas
- Responsável pelo boom da inteligência artificial a 2012



Exemplos de aplicações

- Visão computacional (classificação de imagens, detecção de objetos em imagens, segmentação de imagens, etc)
- Reconhecimento e síntese de fala, classificação de sons
- Processamento de linguagem natural (detecção de spam, análise de sentimento, tradução, geração automática, etc)
- Predição/detecção de preço, demanda, risco, falhas, fraude, etc
- Recomendação de produtos
- Sistemas “inteligentes” / autônomos / auto-otimizáveis / etc



Tipos de Aprendizado

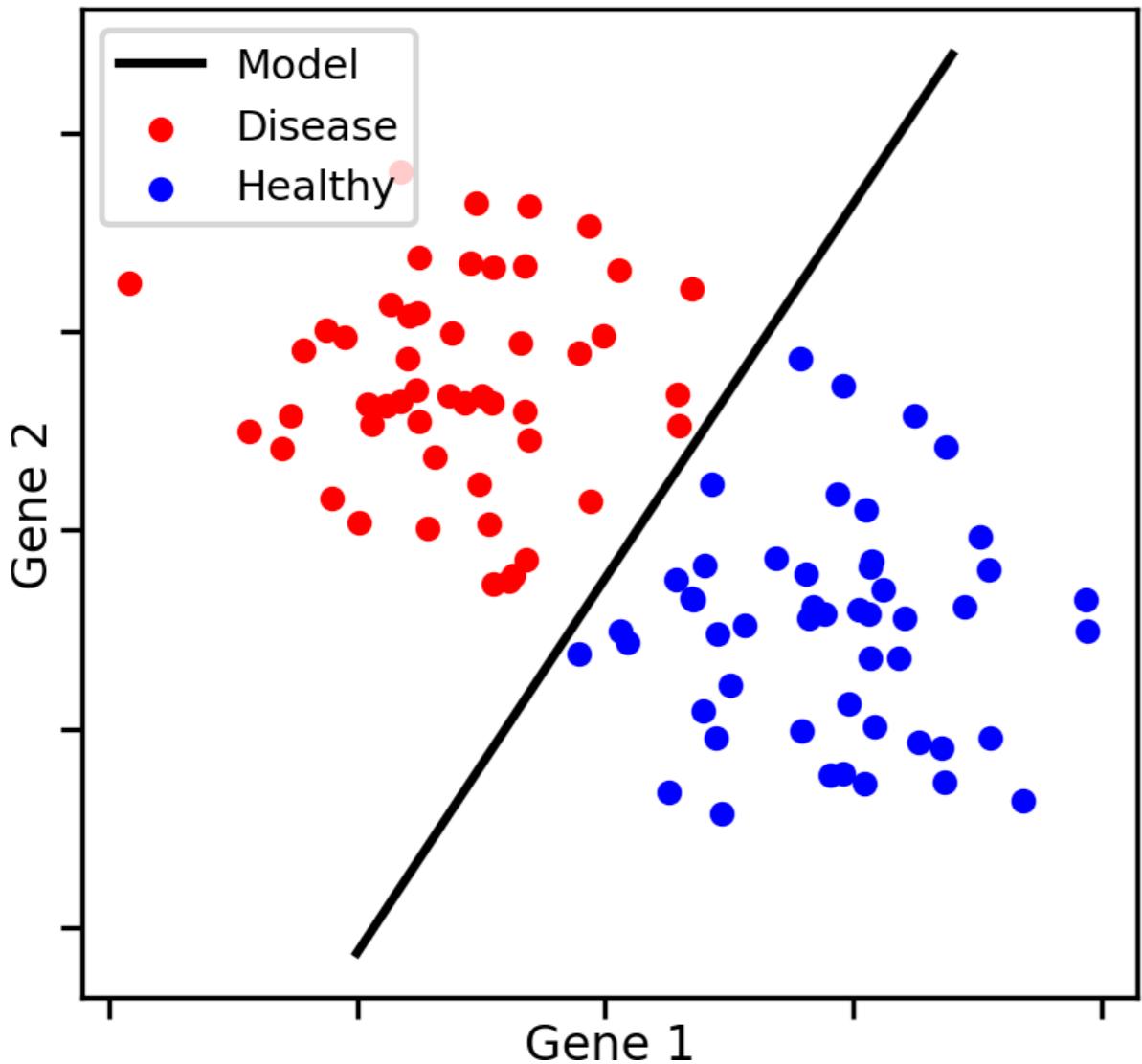
- Aprendizado supervisionado
- Aprendizado não-supervisionado
- Aprendizado semi-supervisionado
- Aprendizado por reforço

Aprendizado Supervisionado

- Dispõe-se de um coniunto de dados rotulados (entrada x , saída y)
$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$$
 - provenientes de uma distribuição $p(x, y)$ desconhecida
- O objetivo é construir uma função $y = f(x)$ (modelo) para prever o rótulo y de uma nova amostra x (não-previamente observada) da mesma distribuição
- Tarefas
 - Classificação: a variável de saída é discreta: $y \in \{1, \dots, K\}$
 - Exemplos: classificação de objetos em imagens, detecção de patologias, reconhecimento de fala, detecção de spam
 - Regressão: a variável de saída é contínua: $y \in \mathbb{R}$
 - Exemplos: predição do preço de um imóvel, predição de demanda por um serviço, avaliação de risco de um empréstimo

Exemplos: Classificação e Regressão

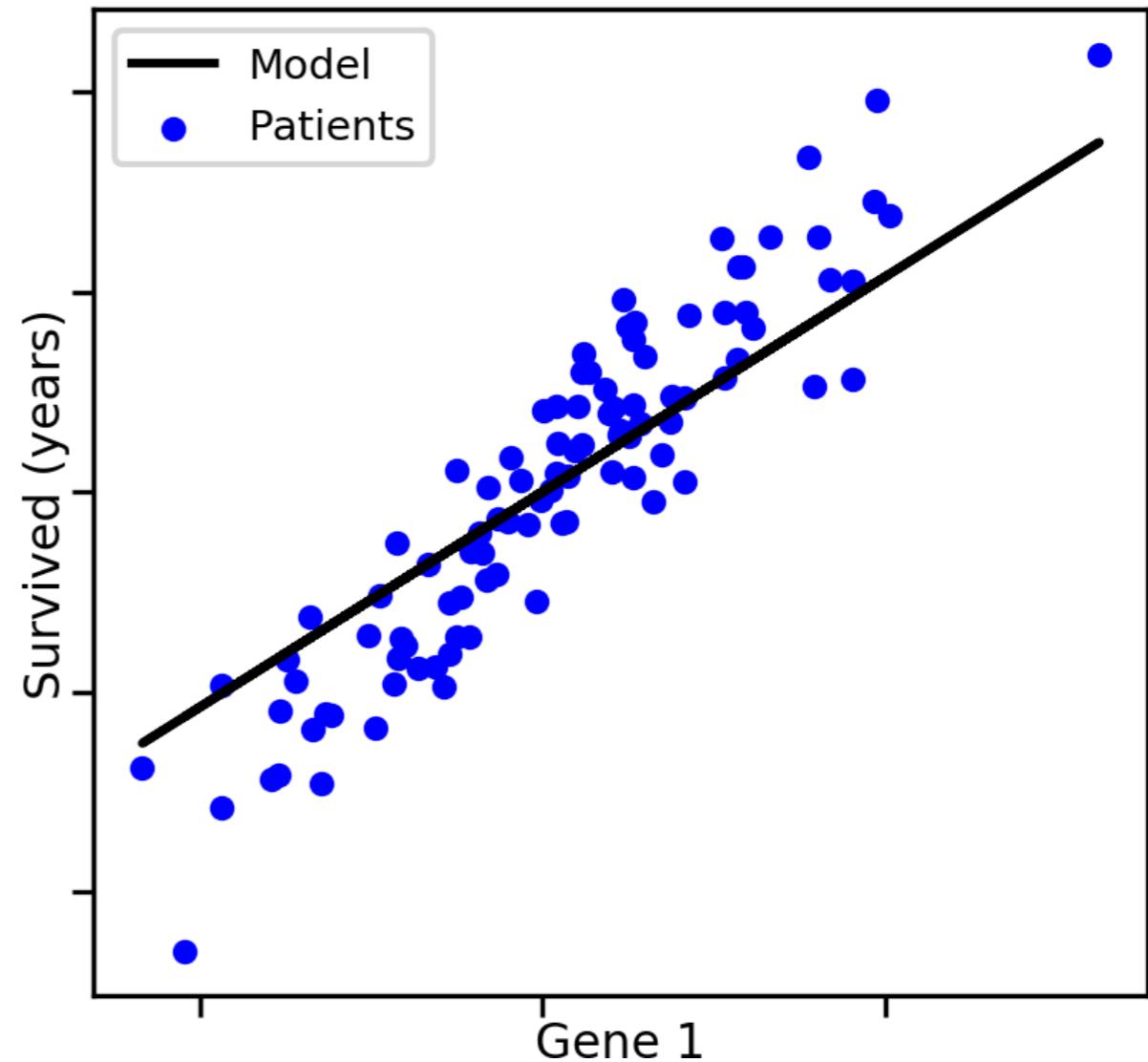
Classification



Object recognition

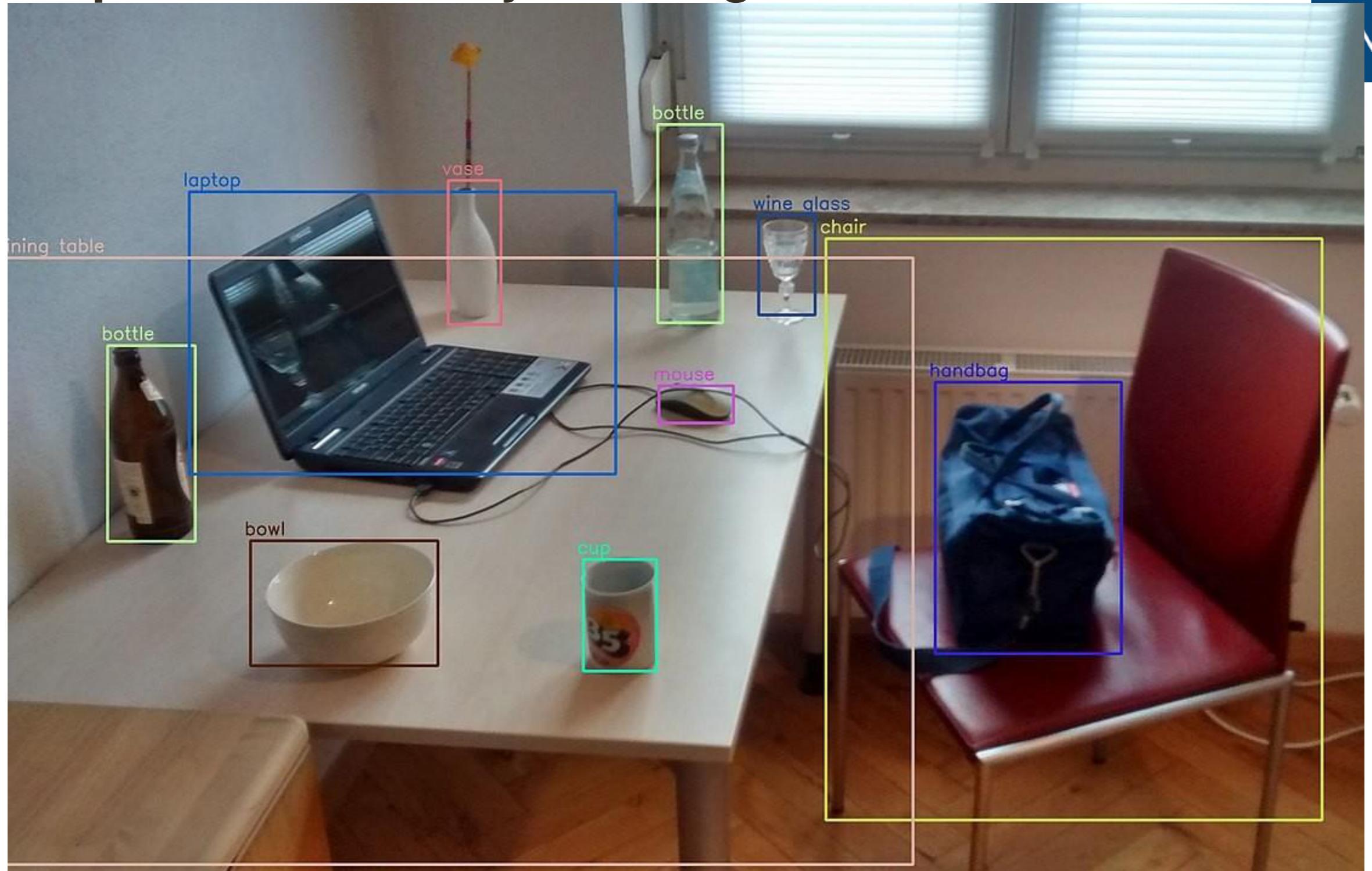
<https://ai.googleblog.com/2014/09/building-deeper-understanding-of-images.html>

Regression



Colorize B&W images automatically
<https://tinyclouds.org/colorize/>

Exemplos: Classificação e Regressão

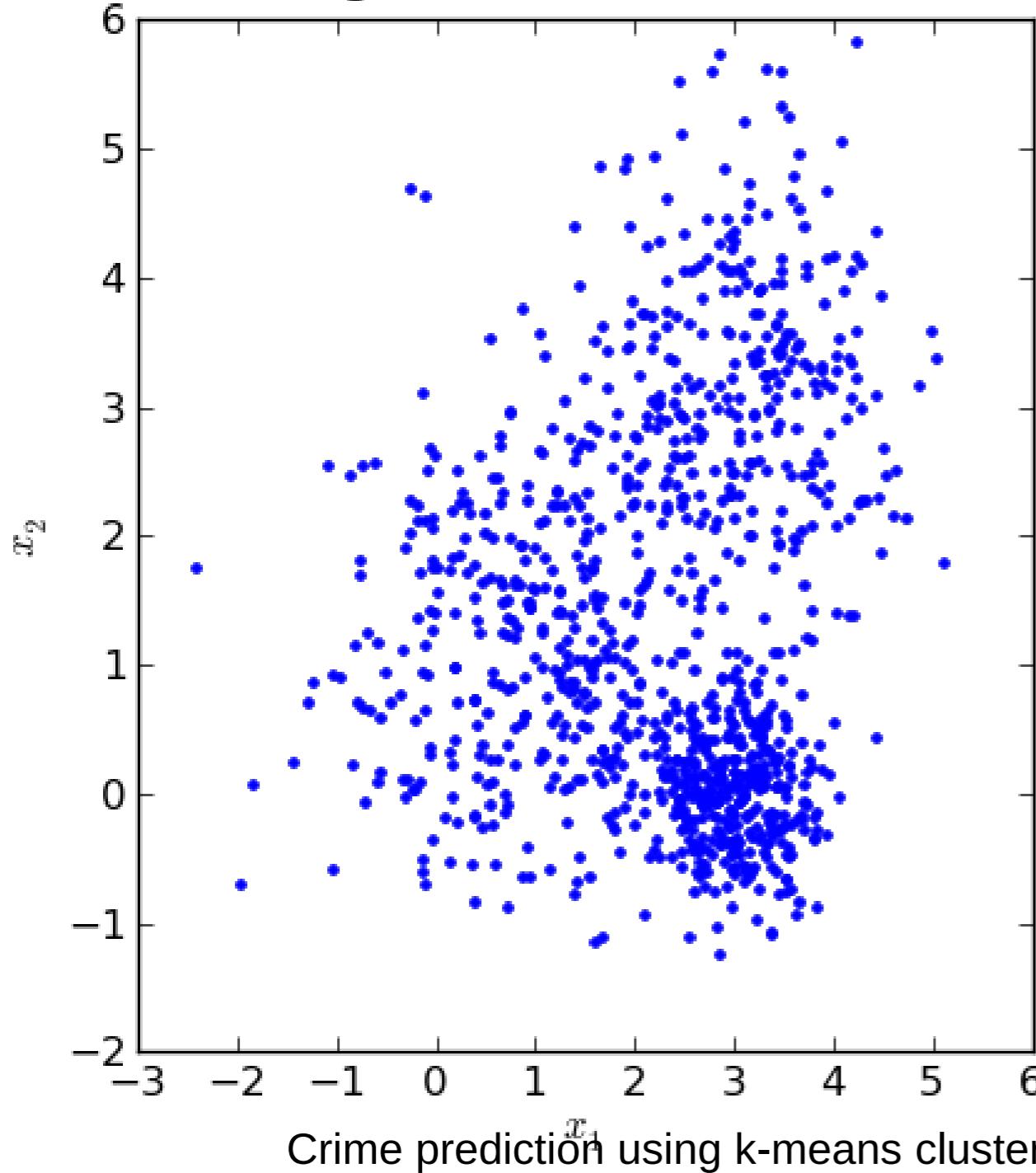


Aprendizado Não-Supervisionado

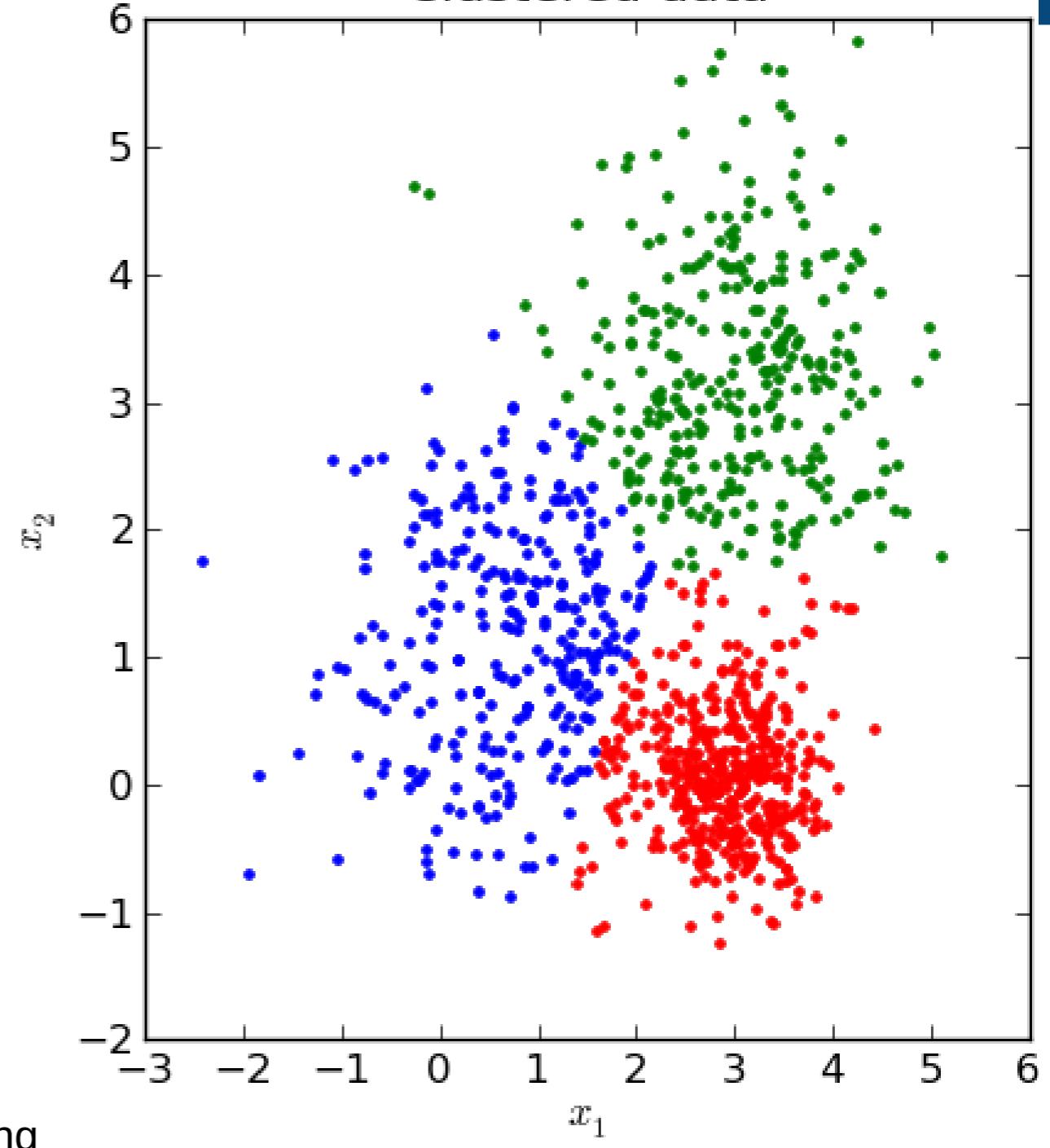
- Conjunto de dados não-rotulados: $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$
- O objetivo é descobrir propriedades da estrutura do conjunto de dados (caracterizada pela densidade $p(\mathbf{x}) = p(x_1, \dots, x_n)$)
- Tarefas
 - Clustering: descobrir grupos de exemplos (dados) similares
 - Exemplos: segmentação de mercado, agrupamento de resultados de busca, identificação de famílias de genes, segmentação de imagens
 - Redução de dimensionalidade: encontrar uma representação mais simples dos dados para agilizar algoritmos ou permitir visualização
 - Detecção de anomalias: identificar casos que fogem ao padrão esperado
 - Exemplos: detecção de fraudes, detecção de falhas em sistemas, monitoramento de saúde
 - Modelos gerativos: gerar novos exemplos (imagens, vídeos, etc), tipicamente com características específicas

Exemplo: Clustering

Original unclustered data

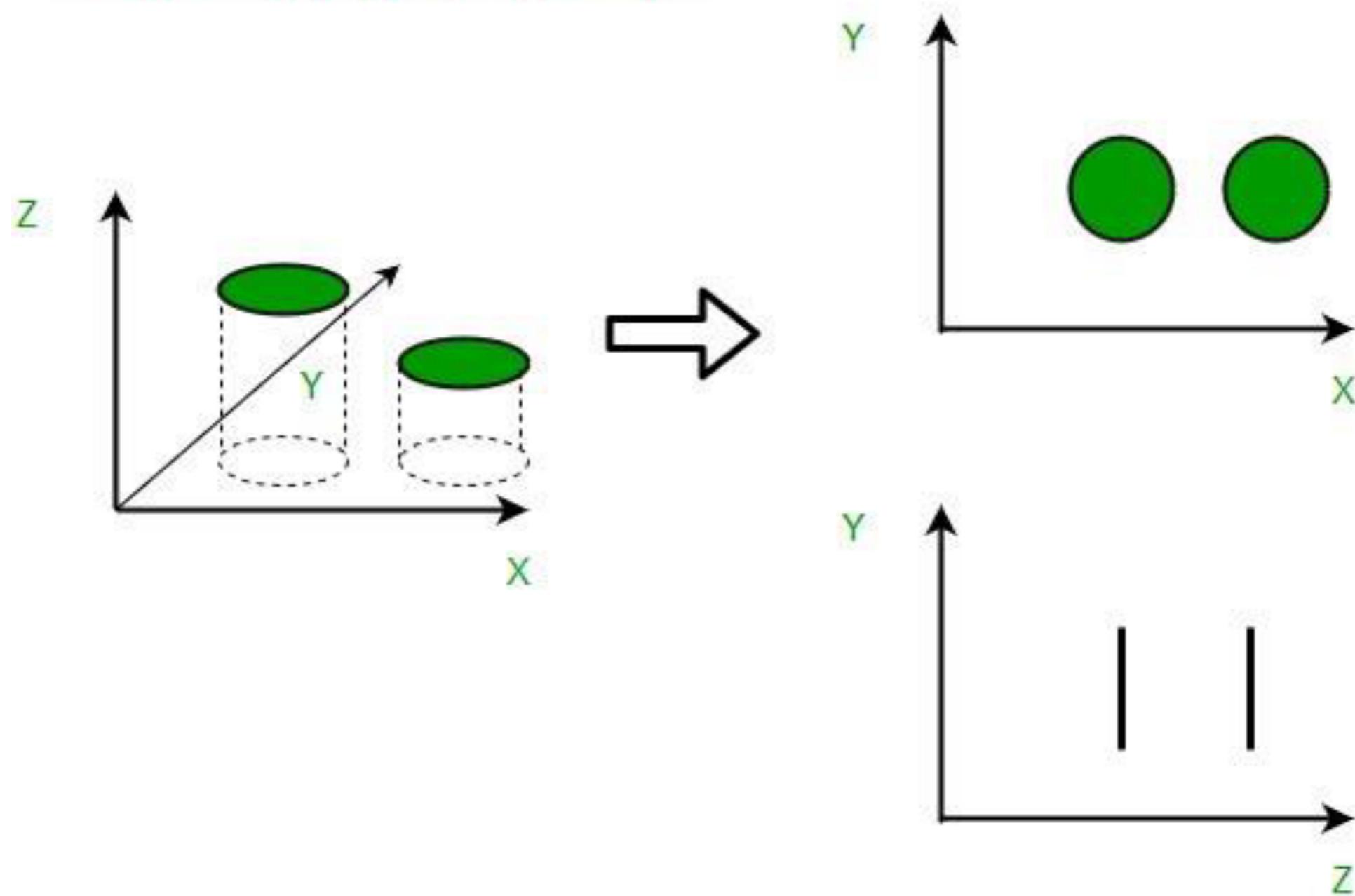


Clustered data

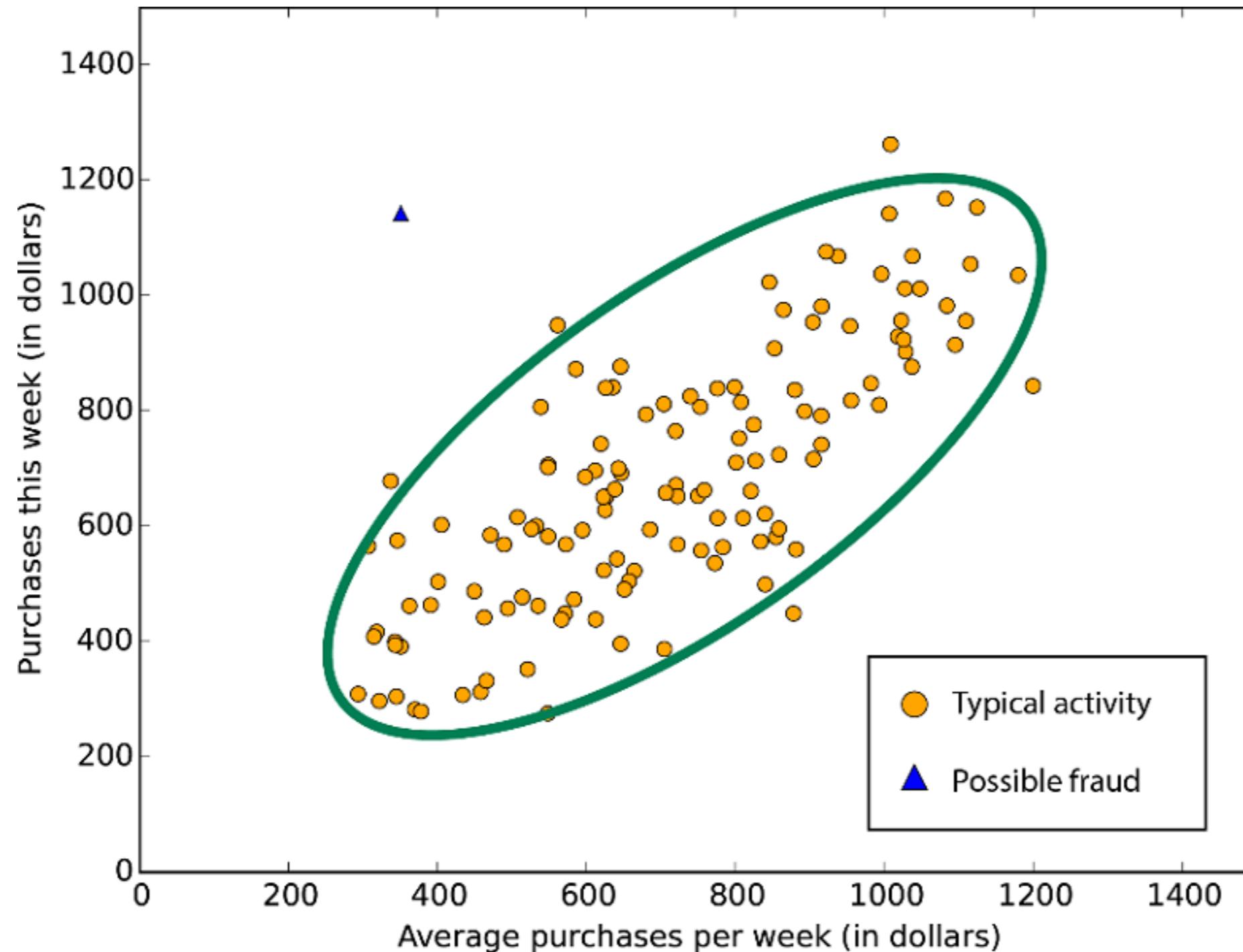


Exemplo: Redução de Dimensionalidade

Dimensionality Reduction



Exemplo: Detecção de Anomalias



Exemplo: Modelos Generativos

<https://www.thispersondoesnotexist.com>



Exemplo: Modelos Generativos

A**B****C****D**

Exemplo: Modelos Generativos

Deep Fakes:

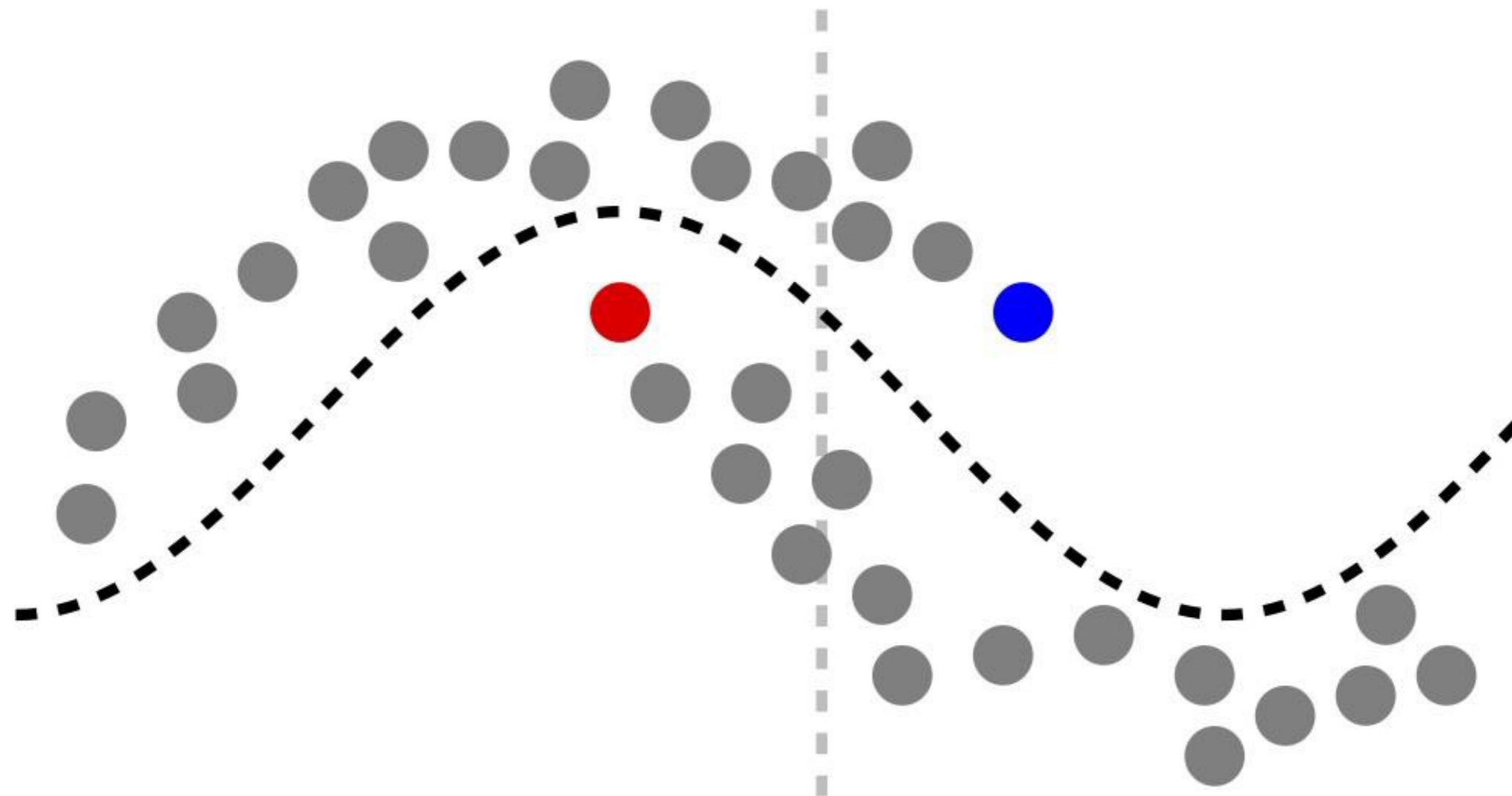
<https://www.youtube.com/watch?v=cQ54GDm1eL0>

<https://www.youtube.com/watch?v=p1b5aiTrGzY>

<https://www.youtube.com/watch?v=0ybLCfVeFL4>

Aprendizado Semi-Supervisionado

- Semelhante ao aprendizado supervisionado, porém dispõe-se também de dados não-rotulados



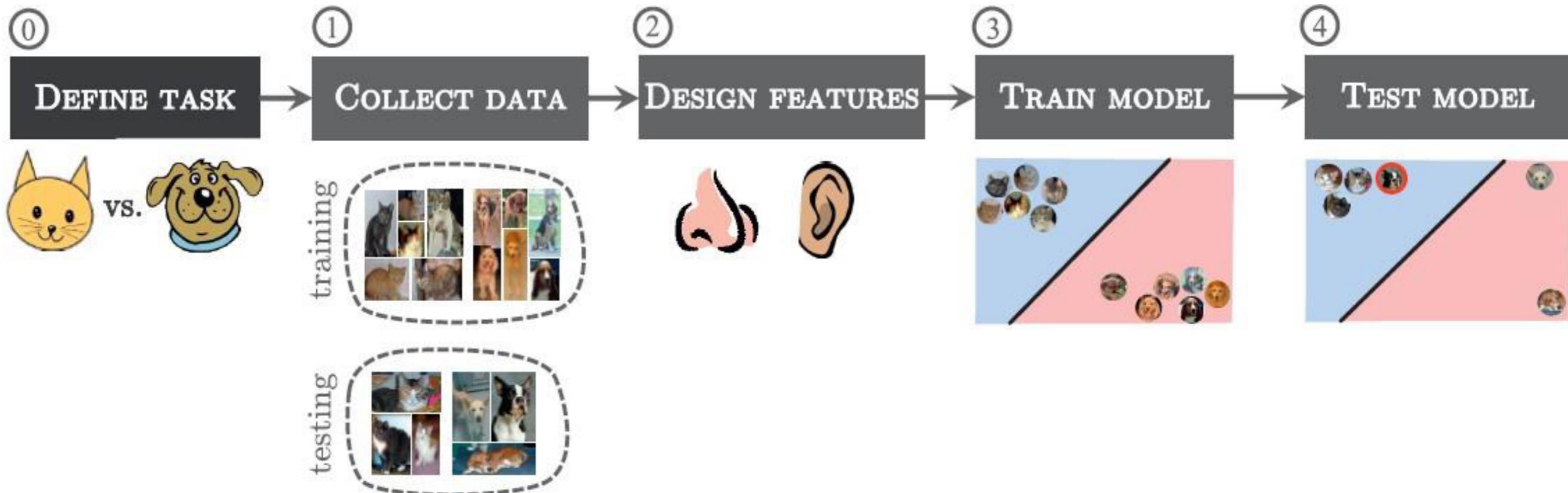
Aprendizado por Reforço

- O algoritmo interage com o ambiente, recebendo um sinal de feedback (recompensa/punição) a cada ação tomada
 - Formulação envolve um conjunto de estados \mathcal{S} , um conjunto de ações \mathcal{A} , uma probabilidade de transição de estados $p(s'|s, a)$ e uma recompensa associada $R_a(s, s')$
- O objetivo é descobrir e executar as melhores ações em cada situação de forma a maximizar a recompensa obtida
- O aprendizado é feito por tentativa e erro e deve balancear descoberta (exploration) e aproveitamento (exploitation)
 - Exemplos: movimentação de robôs, jogos eletrônicos, otimização de redes de comunicação, aplicações financeiras, publicidade
- Frequentemente combinado com técnicas de aprendizado supervisionado para aprender uma função utilidade $Q(s, a)$

Learning to play Break Out

<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

Mas focar mais - Pipeline do Aprendizado Supervisionado



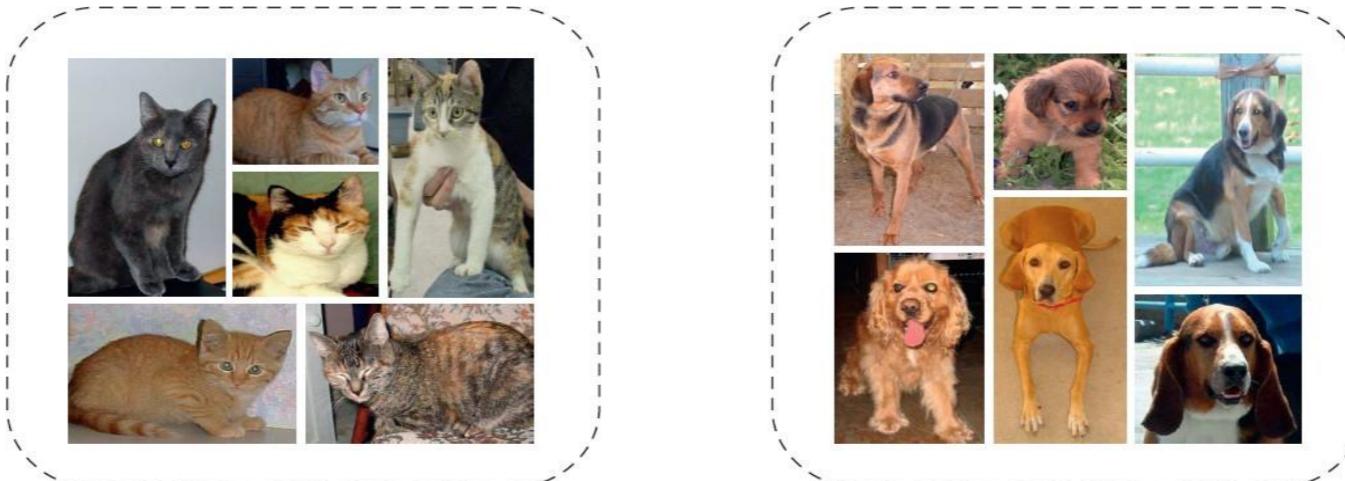
1. Definição da tarefa
2. Coleta de dados
3. Desenvolvimento de atributos
4. Treinamento do modelo
5. Teste do modelo

Definição da tarefa

- Especificação do espaço de possibilidades Y da variável de saída y
 - Classificação: $\mathcal{Y} = \{1, \dots, K\}$
 - Regressão: $\mathcal{Y} = \mathbb{R}$
- Especificação de uma métrica de avaliação
 - Classificação: ex: \uparrow acurácia (taxa de acerto) = 1 - taxa de erro
 - Regressão: ex: \downarrow erro quadrático médio
- Obs: Em geral, a avaliação do desempenho de um preditor não é um problema trivial, podendo envolver múltiplos critérios

Coleta de dados

- Coletar e dividir o conjunto de dados (dataset) em
 - Conjunto de treinamento (training set)

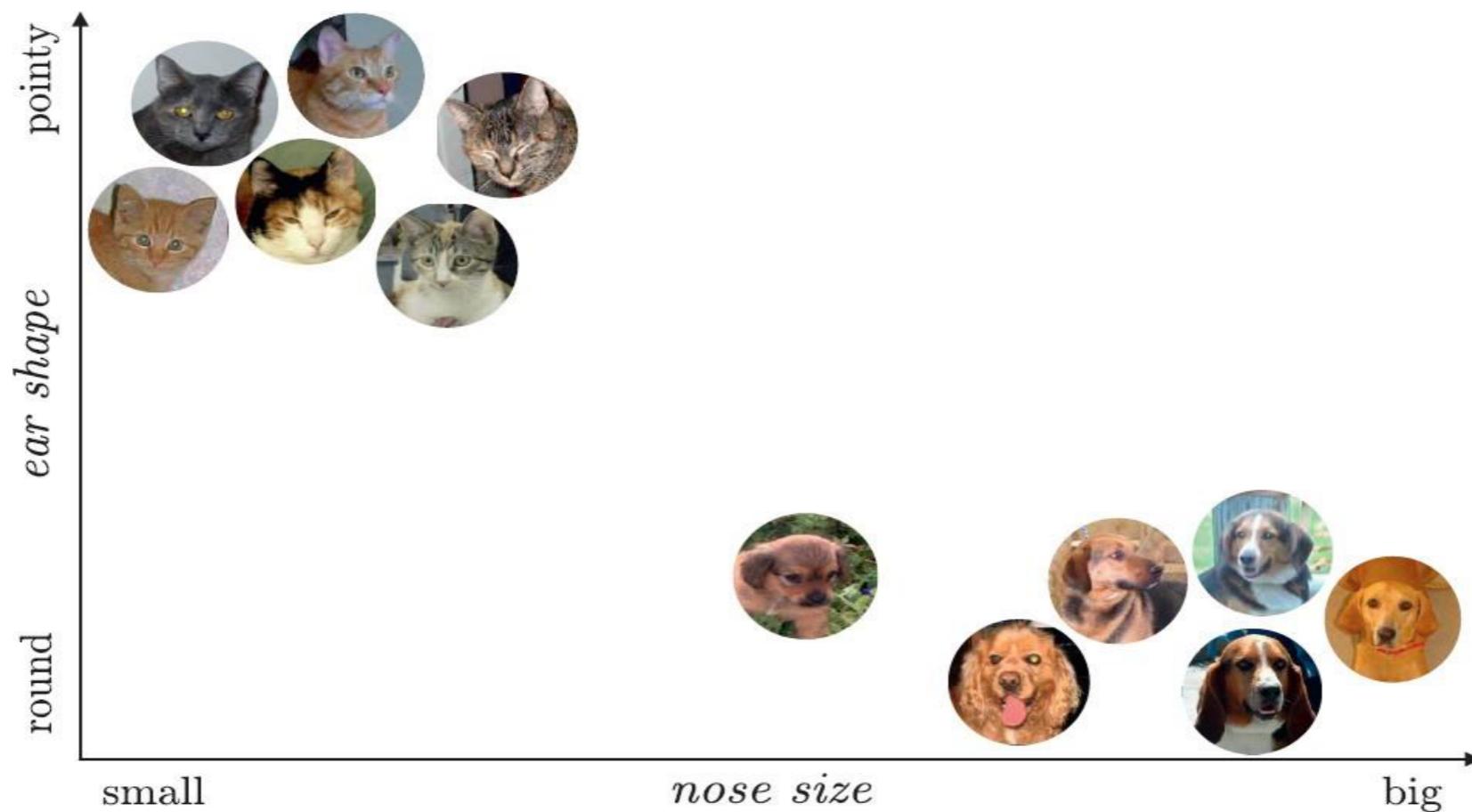


- Conjunto de teste (*test set*)



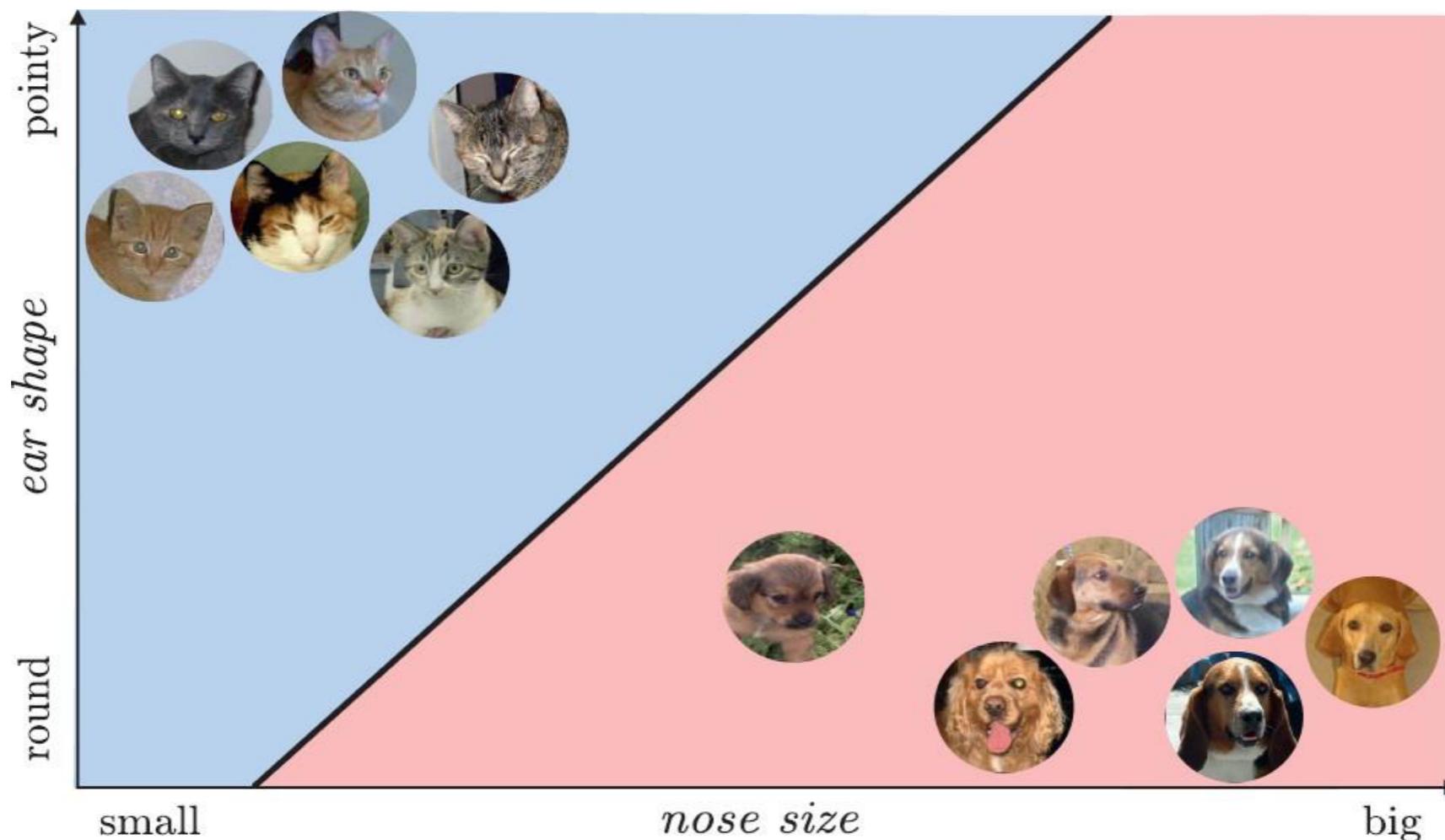
Desenvolvimento de atributos (*features*)

- Extrair atributos (ou características — features) reduz a dimensão do problema, facilitando o aprendizado
 - Vetor de atributos (*feature vector*): $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$
 - Requer conhecimento específico da área de aplicação



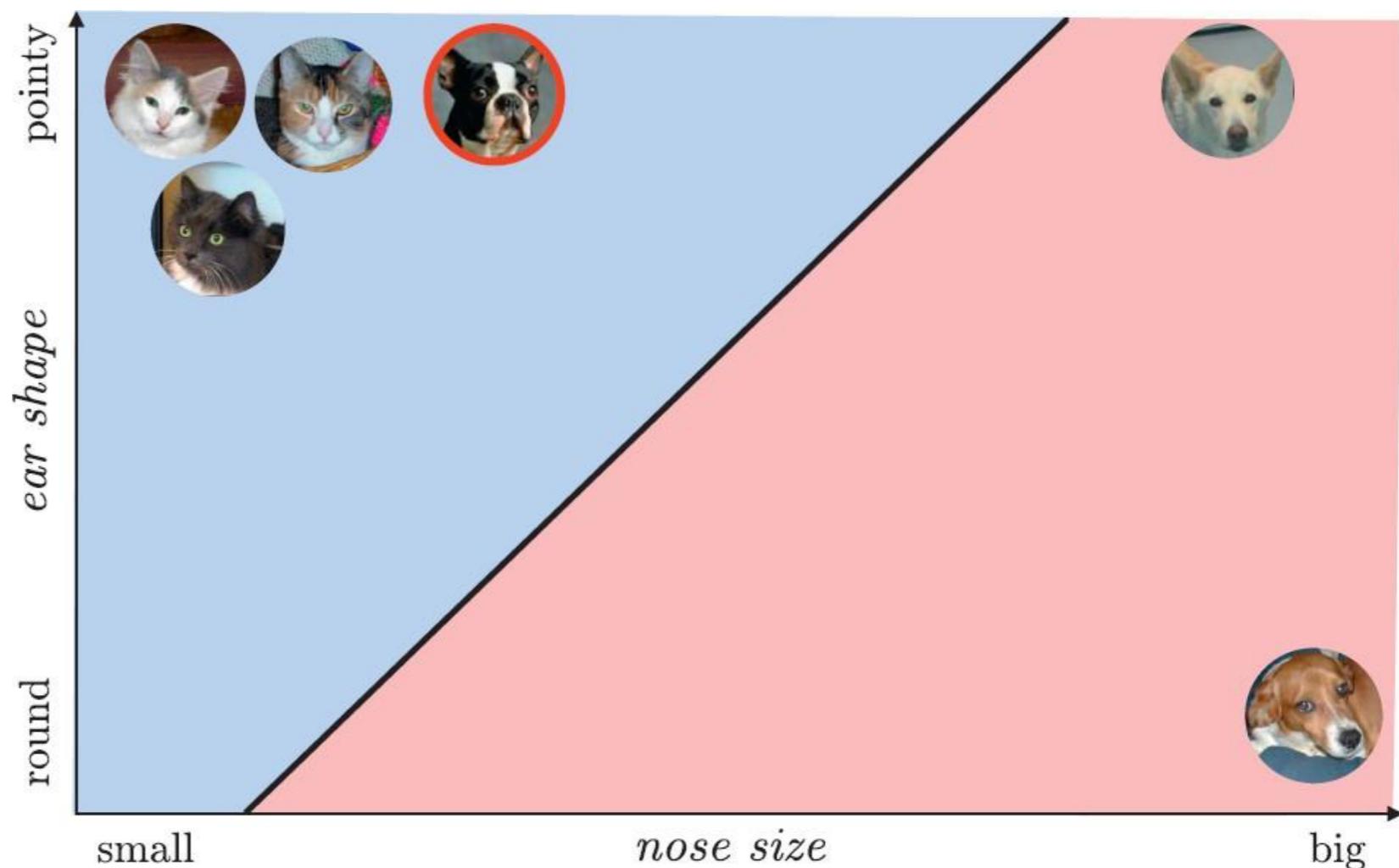
Treinamento do modelo

- Definição de uma família de modelos: classe/espaço de hipóteses \mathcal{H}
 - Ex: modelo linear
- Treinar = escolher um modelo $f \in \mathcal{H}$ por meio de otimização numérica, de forma a minimizar o erro no conjunto de treinamento



Teste do modelo

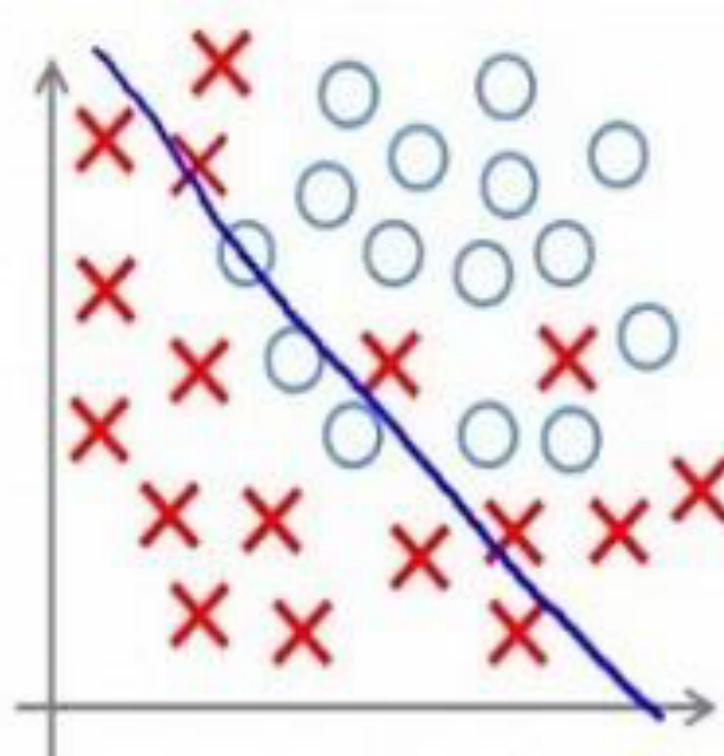
- A partir do modelo treinado, realiza-se uma predição $\hat{y} = f(\mathbf{x})$ para cada amostra x do conjunto de teste
- Usando a métrica pré-definida, avalia-se o desempenho das predições realizadas
 - Ex: acurácia de $5/6 = 83.3\%$



Teste do modelo

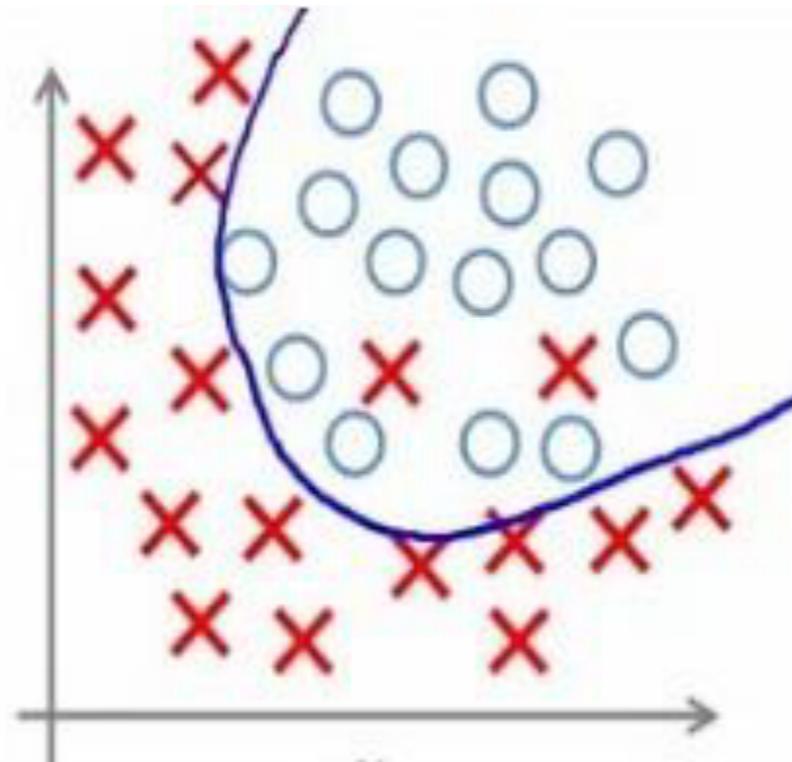
- Se o conjunto de teste é suficientemente grande, representativo e estatisticamente independente do modelo treinado, o desempenho no conjunto de teste produz uma boa estimativa do desempenho real do modelo
- Caso o desempenho seja insatisfatório, pode-se iterar o desenvolvimento do modelo, isto é: desenvolver melhores atributos, escolher outro (tipo de) modelo, e treiná-lo novamente
- No entanto, deve-se evitar realizar muitas iterações de desenvolvimento usando o mesmo conjunto de teste, caso contrário viola-se a hipótese de independência e o desempenho não será mais representativo
 - Nesse caso, deve-se coletar um novo conjunto de teste
- Em geral, utilizar um conjunto de teste permite detectar *overfitting*

Underfitting e Overfitting

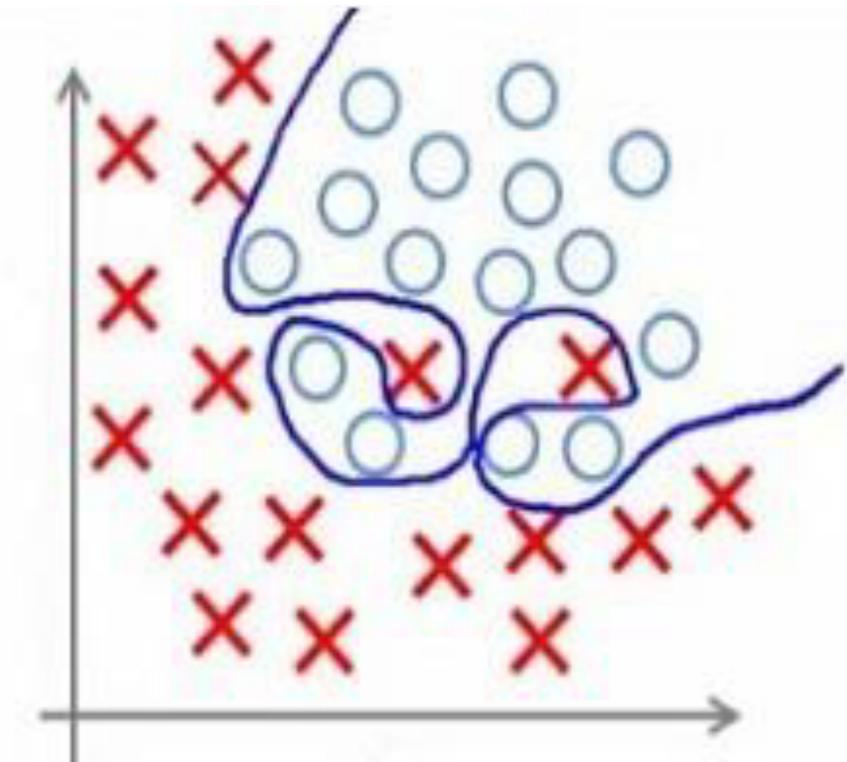


Under-fitting

(too simple to
explain the
variance)



Appropriate-fitting

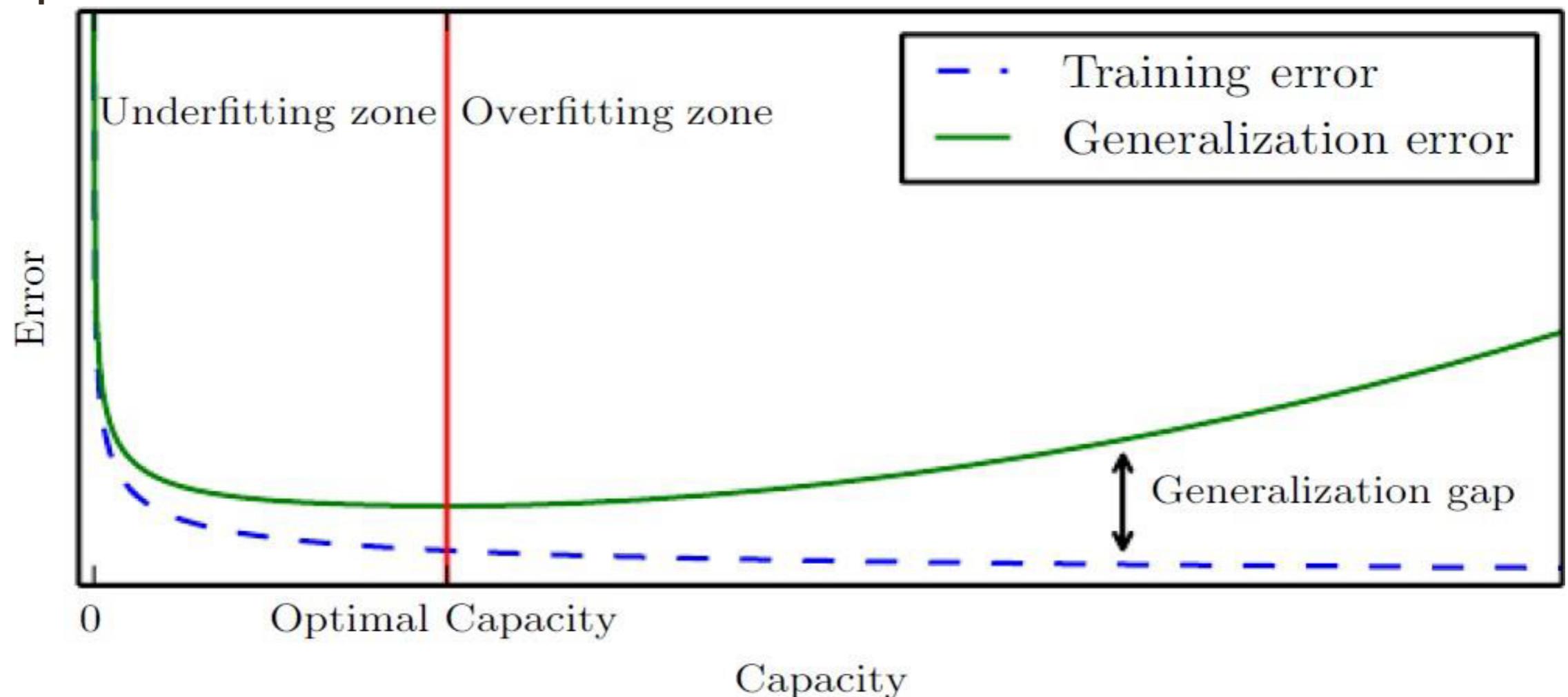


Over-fitting

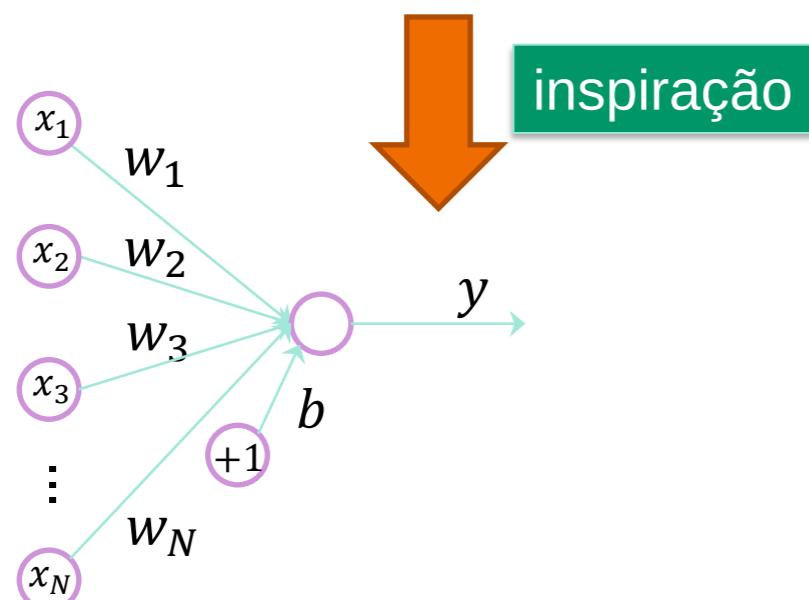
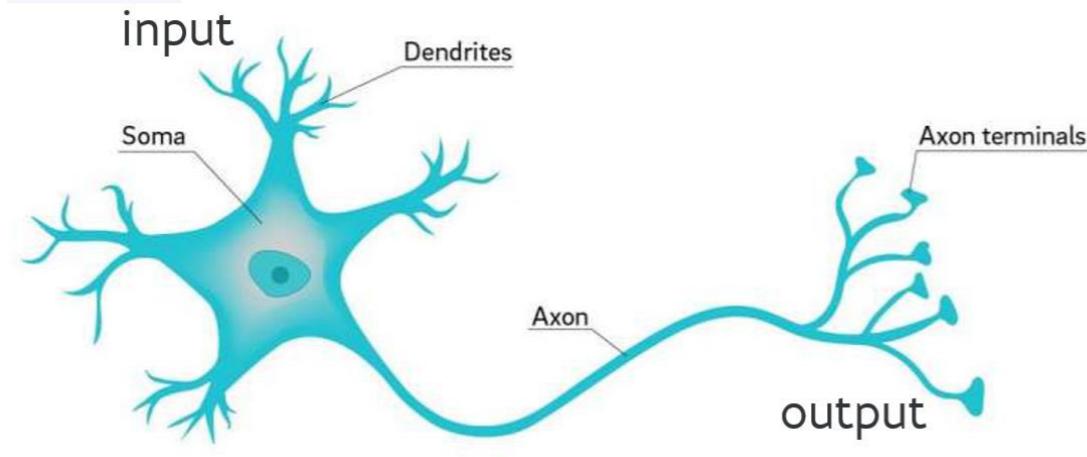
(forcefitting -- too
good to be true)

Underfitting e Overfitting: Tradeoff

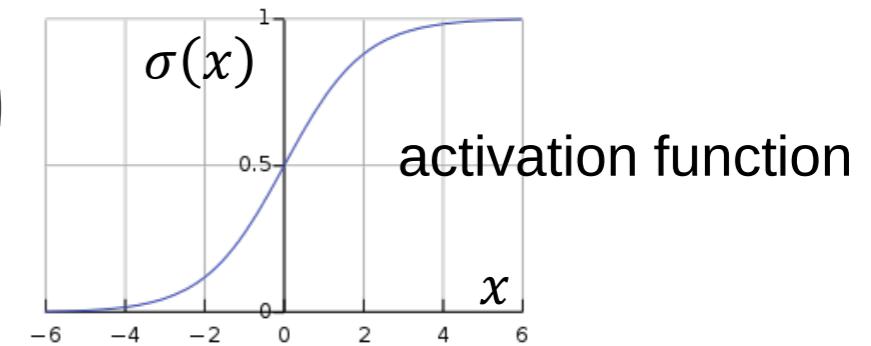
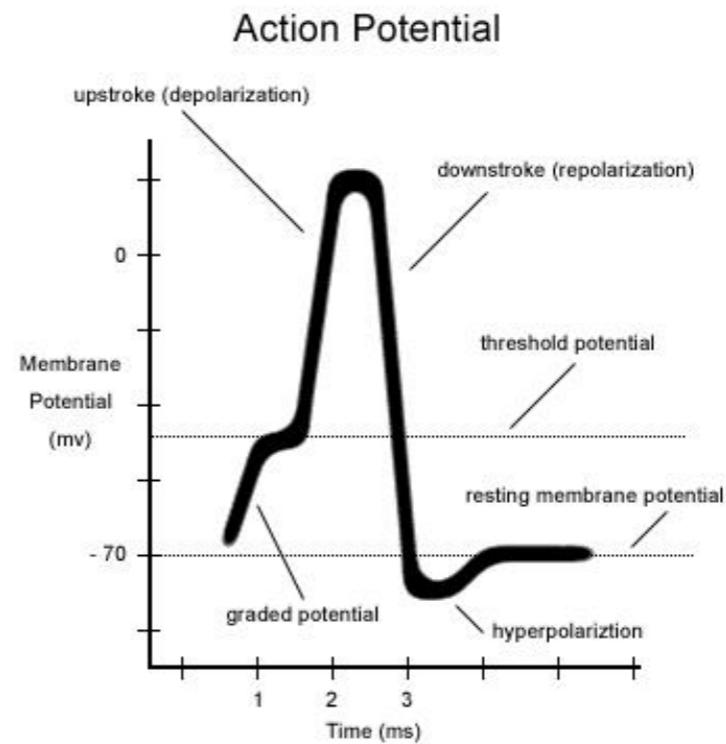
- Capacidade do modelo refere-se à capacidade de representar com precisão o conjunto de treinamento
- Está associada à complexidade do modelo, por exemplo, número de parâmetros treináveis



De neurônios a ANNs

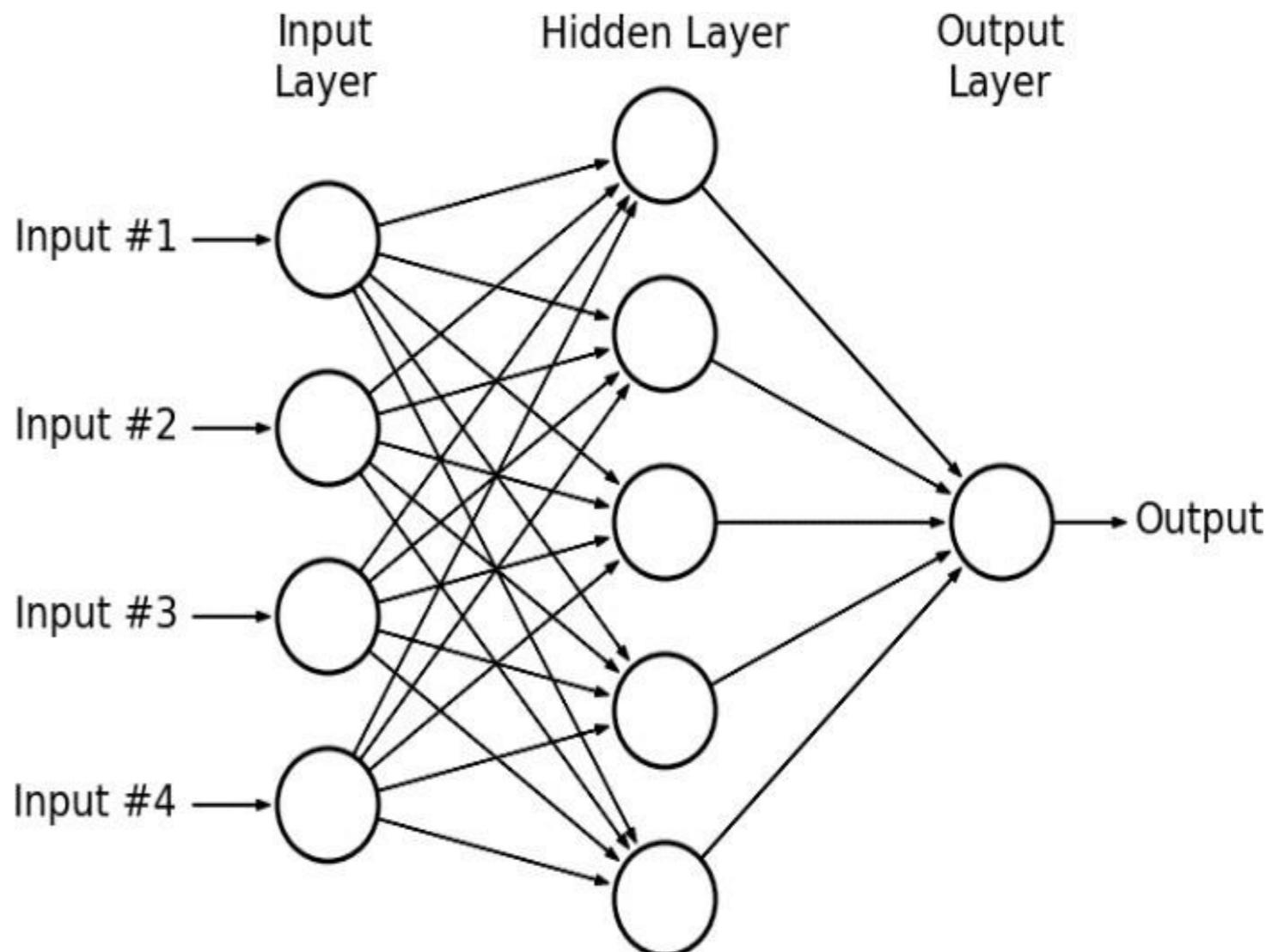


$$y = \sigma \left(\sum_{i=1}^N w_i x_i + b \right)$$



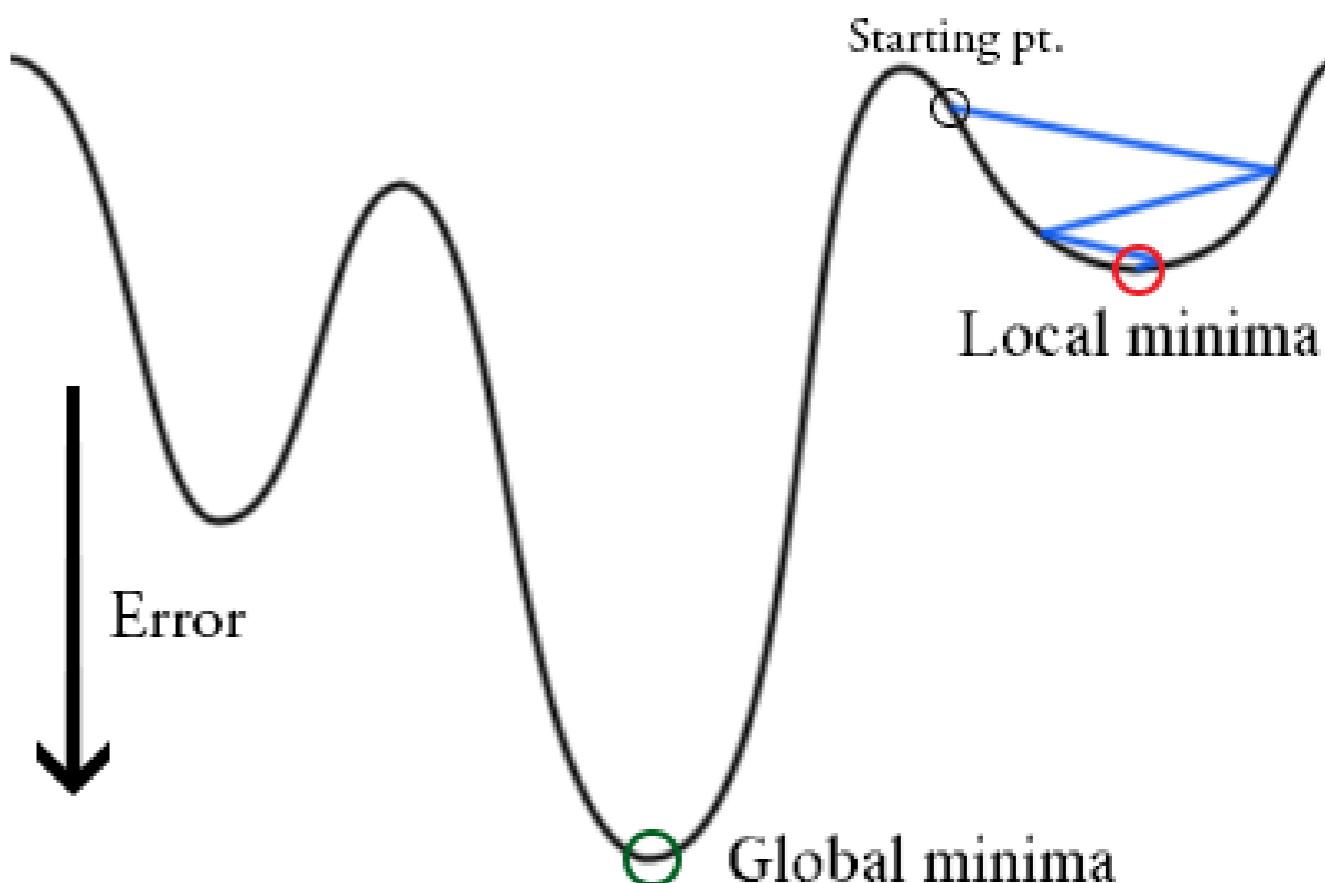
Conceitos de Rede Neural

- Redes neurais multi-camadas perceptron
 - Aproximador universal de funções contínuas
 - Totalmente conectadas
 - Treinamento: gradiente descendente com retropropagação dos erros para o cálculo das derivadas



Treinamento: backpropagation

- Inicializar pesos "aleatoriamente"
- Para todas as épocas de treinamento
 - para todas as entradas-saída no conjunto de treinamento
 - usando entrada, saída de computação (para a frente)
 - comparar saída computada com saída de treinamento
 - adaptar pesos (para trás) para melhorar a produção
 - se a precisão é boa o suficiente, pare



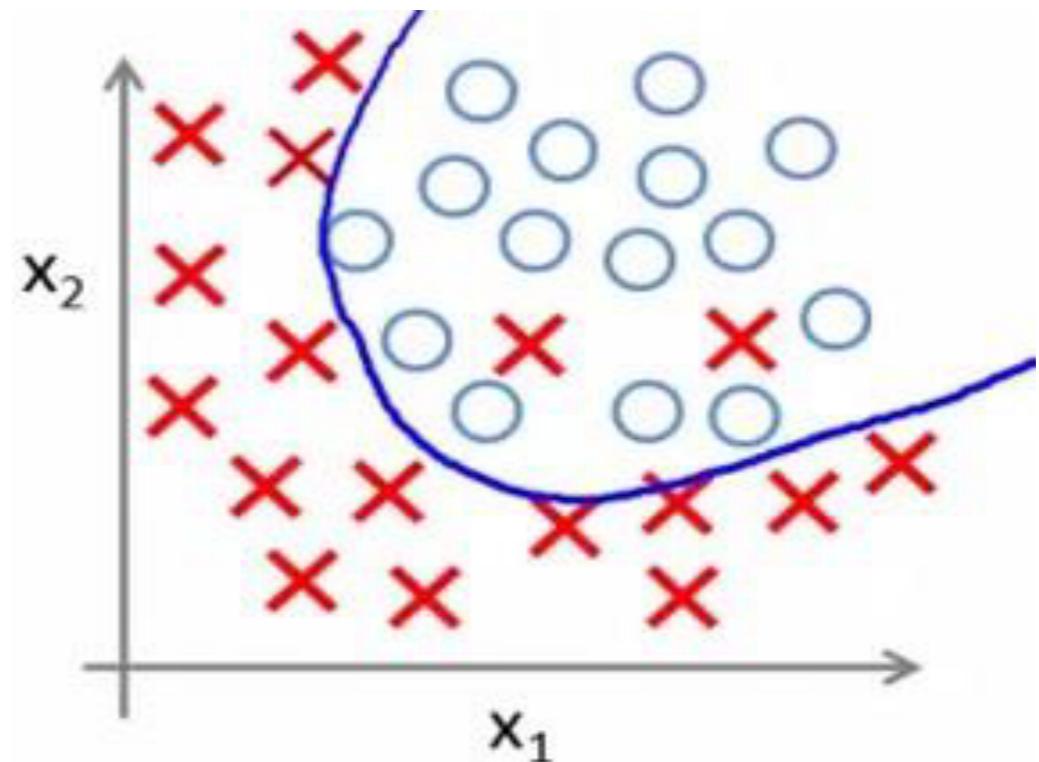
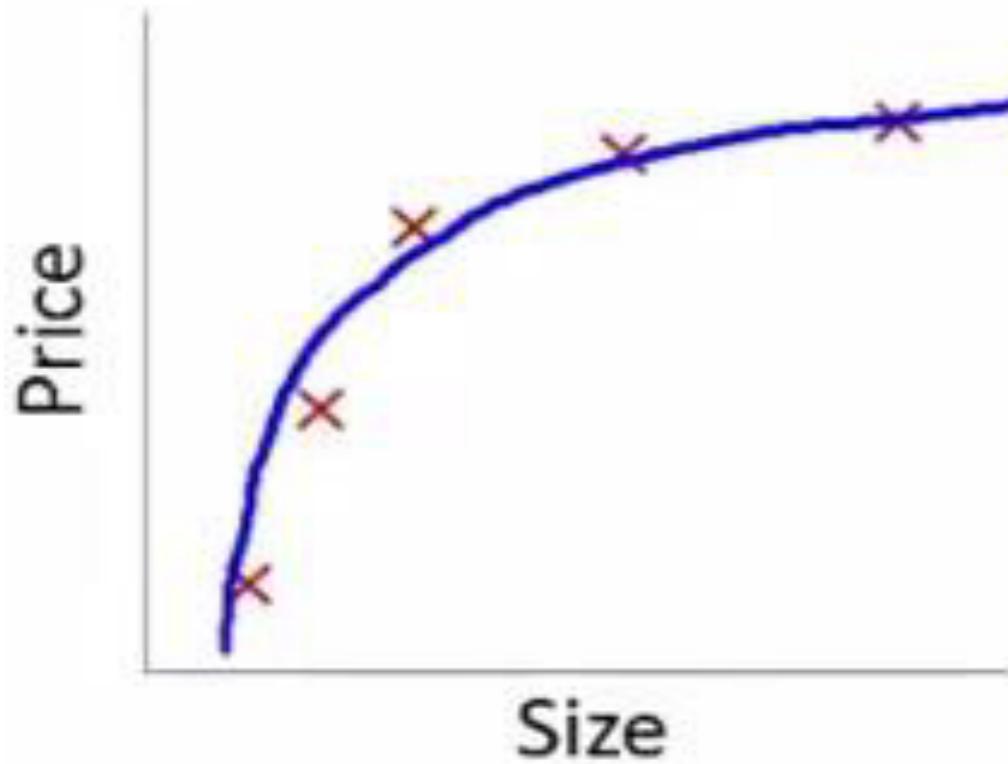
Conceitos de Rede Neural

- Redes neurais multi-camadas perceptron

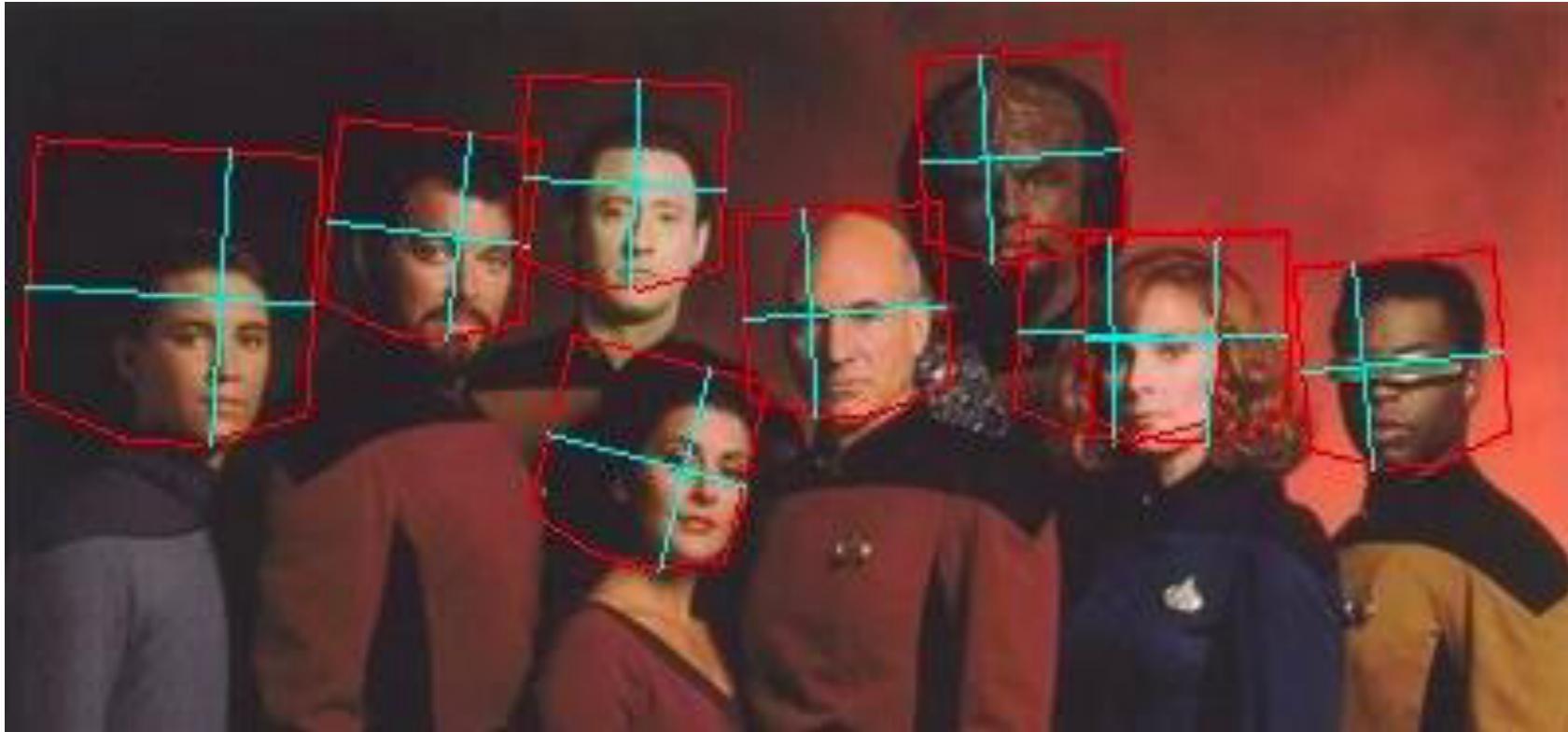
- Problemas de Regressão

e

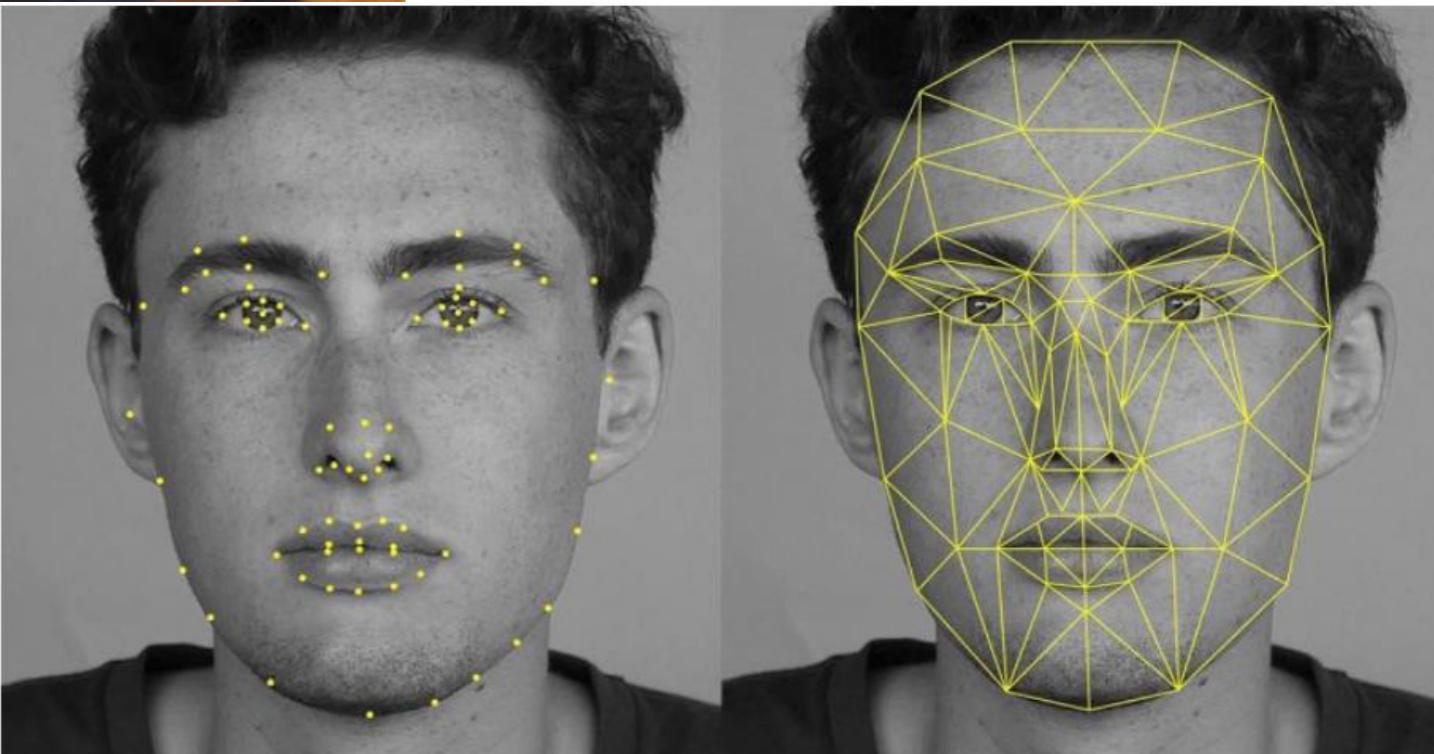
- Classificação



Conceitos de Rede Neural

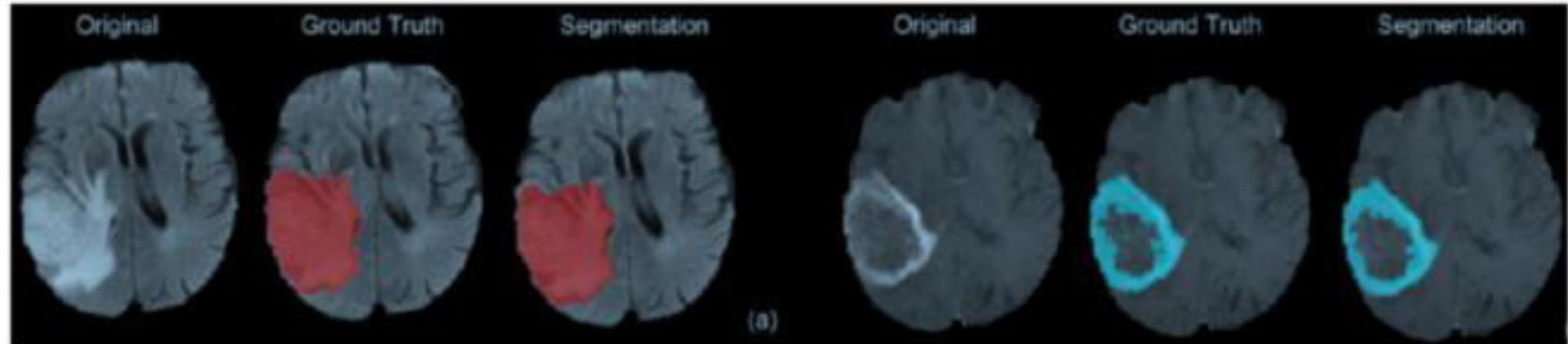


Detecção e
reconhecim
ento facial

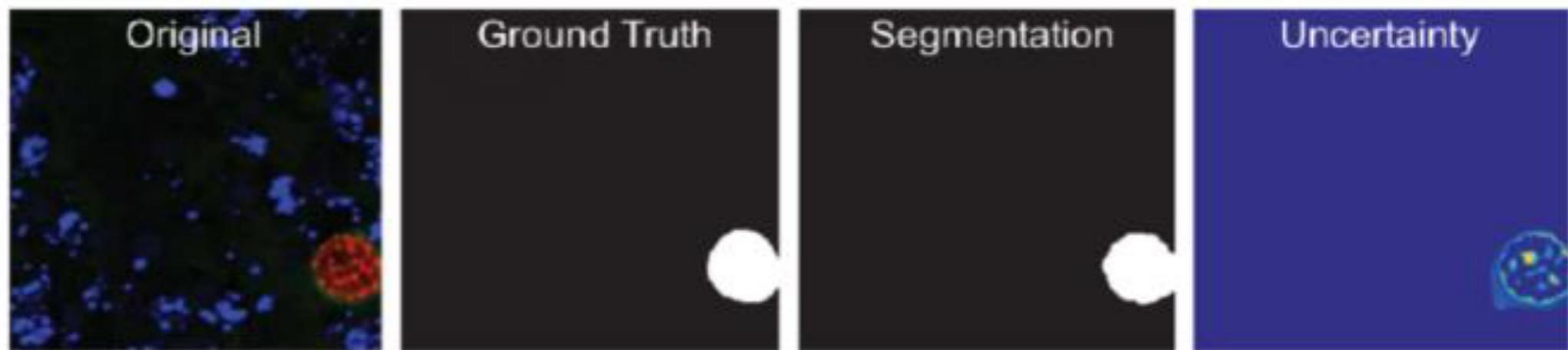


Conceitos de Rede Neural

Tumor
cerebral

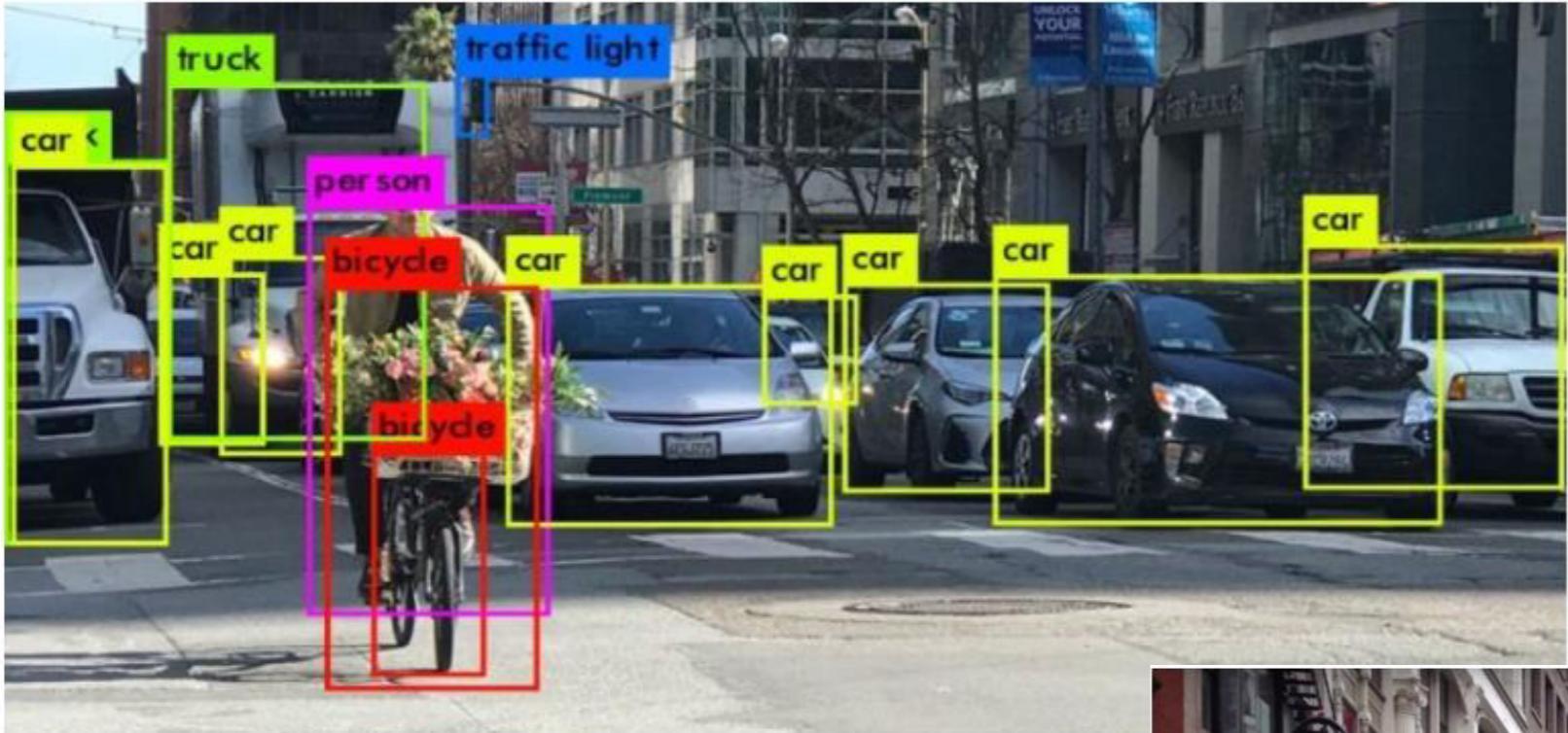


Infecção
devido à
malária



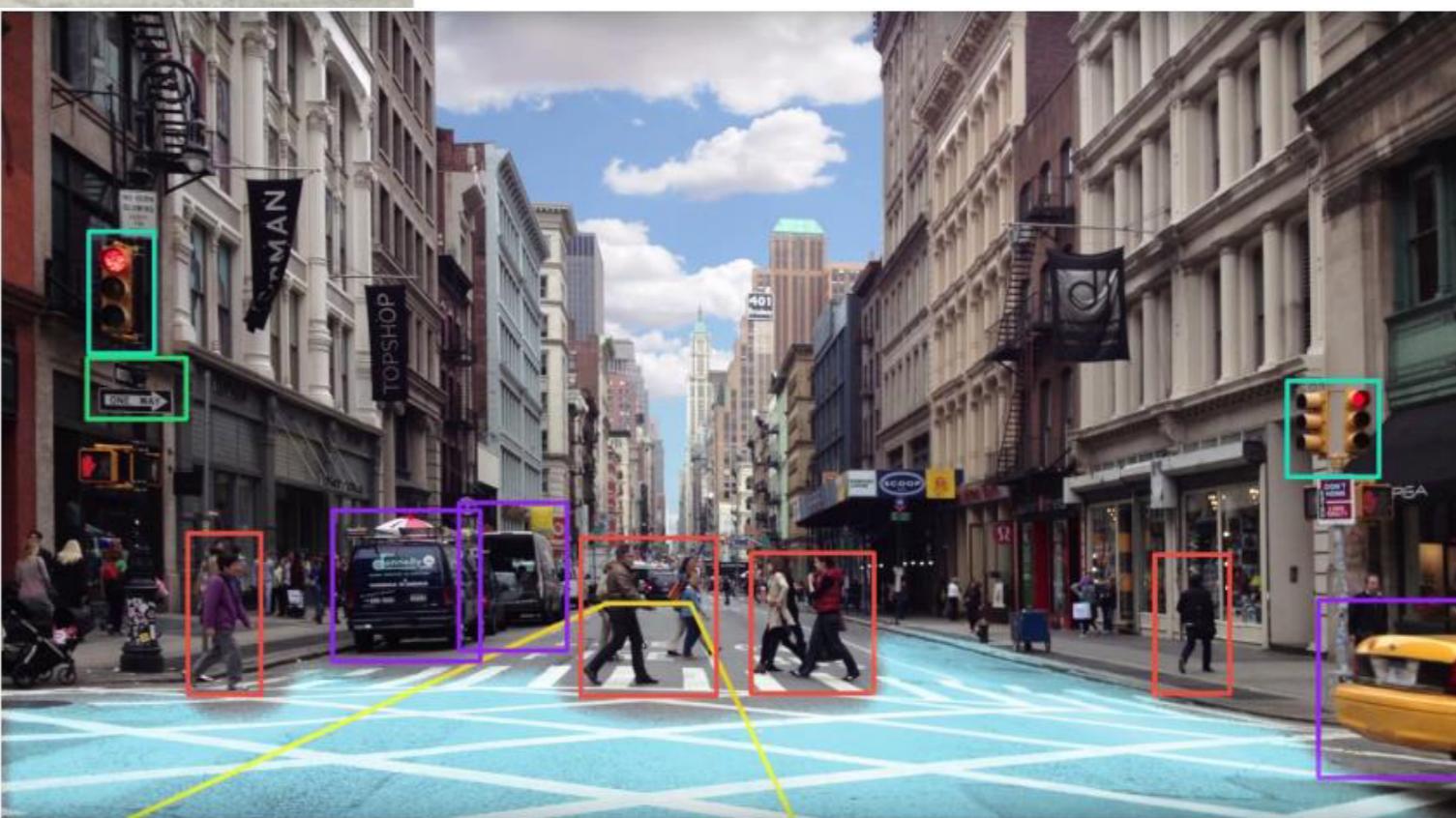
Segmentação semântica: Análise
biomédica

Conceitos de Rede Neural



Region-based
Convolutional Neural
Networks (R-CNN)

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



Conceitos de Rede Neural

Identificar as principais *features* em cada categoria de imagem



Olhos, nariz, boca, ...



Escadas, porta, janelas ...



Faróis, rodas, placa, ...

Conhecimento de domínio

Variação do ponto de vista



Variação de escala



Condições de iluminação



Deformação



Definir features

Detectar features p/ classificar

Desordem de fundo



Oclusão

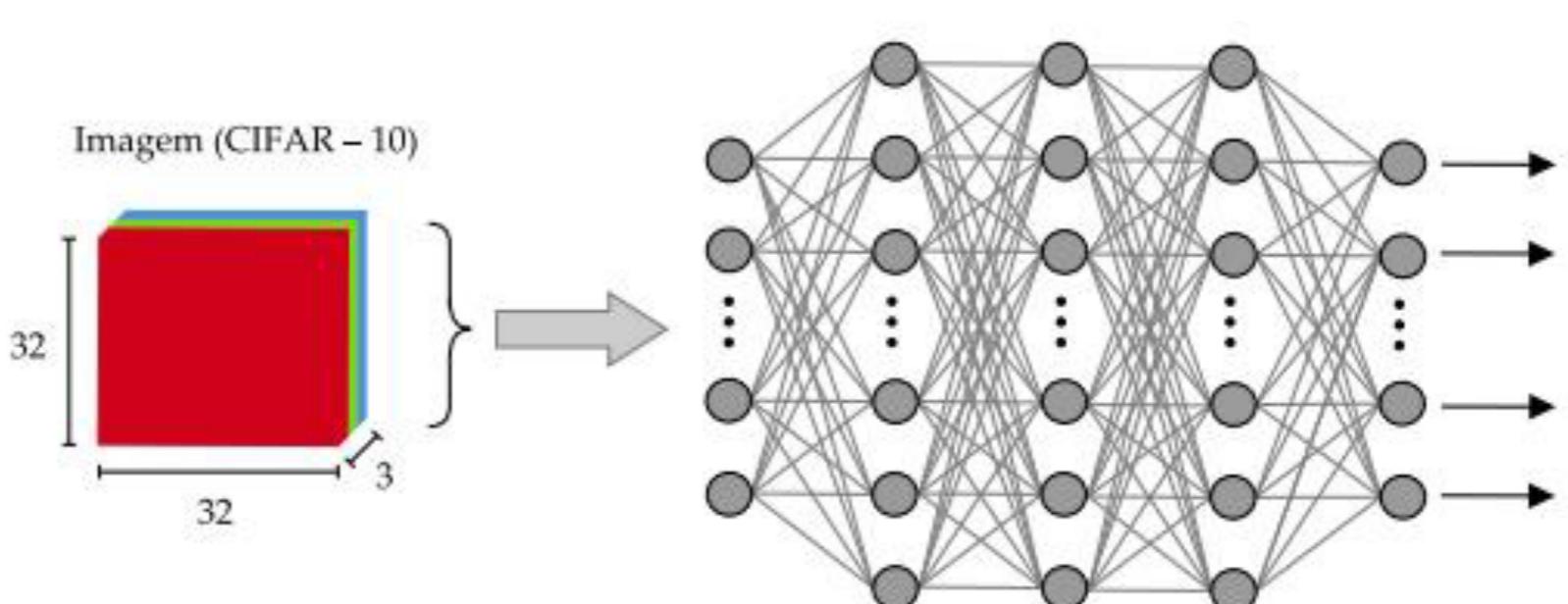


Variação Intraclass



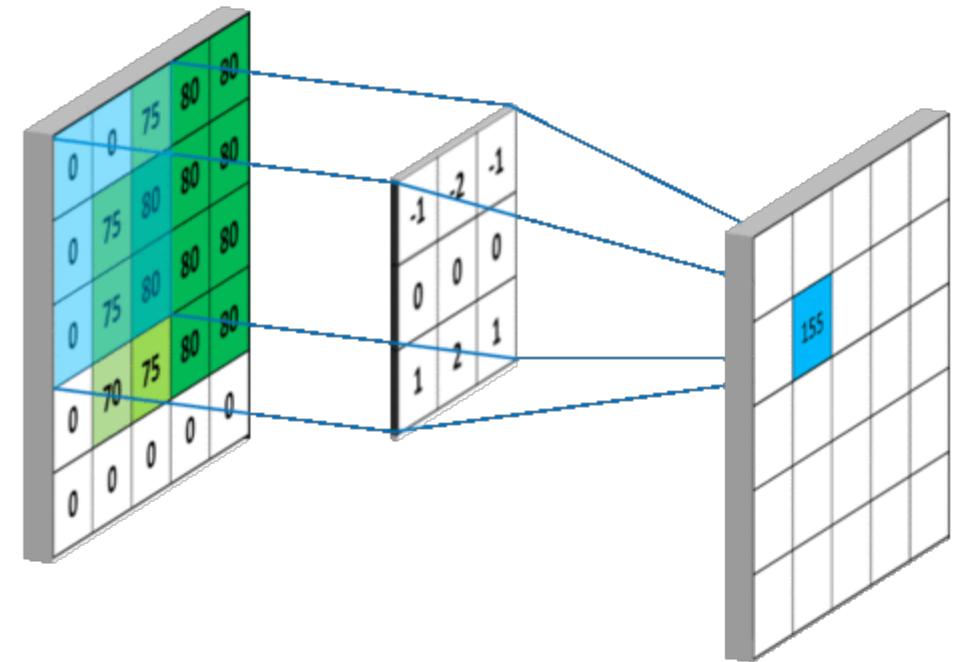
Conceitos de Rede Neural

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



Conceitos de Rede Neural

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



Conceitos de Rede Neural

- Aplicando filtros para extrair *features*

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

- Exemplo

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

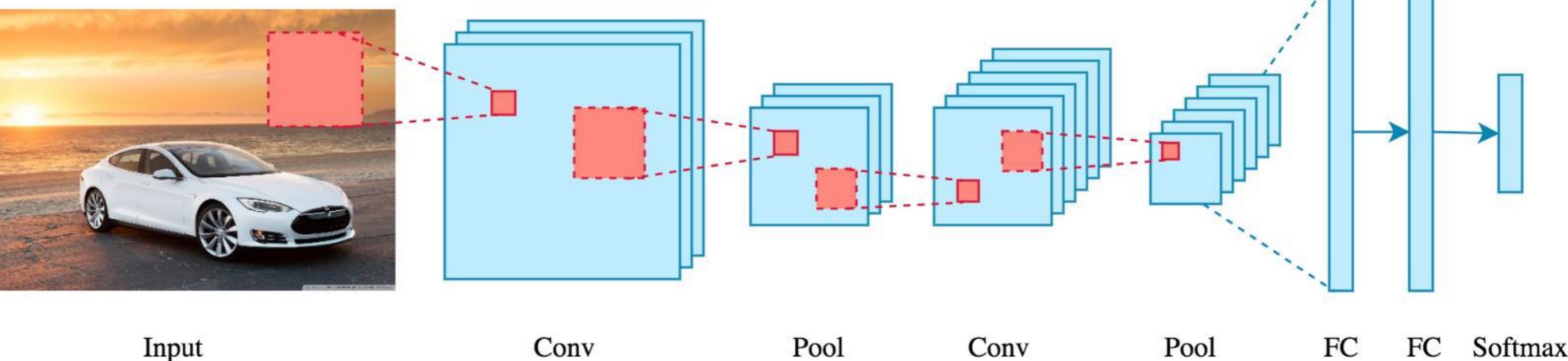
$$\begin{array}{|c|c|c|c|c|} \hline 7 & 2 & 3 & 3 & 8 \\ \hline 4 & 5 & 3 & 8 & 4 \\ \hline 3 & 3 & 2 & 8 & 4 \\ \hline 2 & 8 & 7 & 2 & 7 \\ \hline 5 & 4 & 4 & 5 & 4 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 6 & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

$7x1+4x1+3x1+$
 $2x0+5x0+3x0+$
 $3x-1+3x-1+2x-1$
 $= 6$

$$Conv_{x,y} = \sum_{i=1}^3 \sum_{j=1}^3 K_{i,j} I_{x+i-1, y+j-1}$$

Rede Neural Convolucional

- Origem: neocognitron por Kunihiko Fukushima em 1980
- Tipo especial de Redes neurais multi-camadas
- Arquitetura especializada em reconhecer padrões visuais diretamente de pixels de imagens com mínimo pré-processamento



Input

Conv

Pool

Conv

Pool

FC

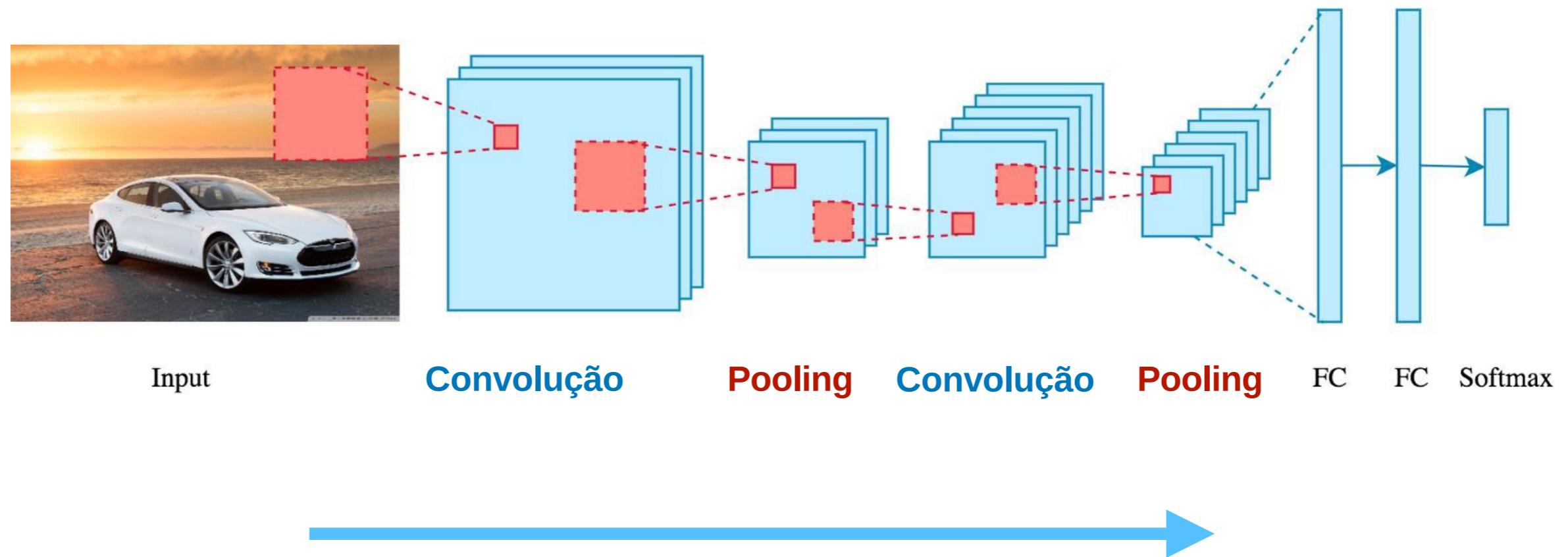
FC

Softmax

Rede Neural Convolucional

Deep learning

Redes neurais convolucionais



Rede Neural Convolucional

Abstrações das camadas

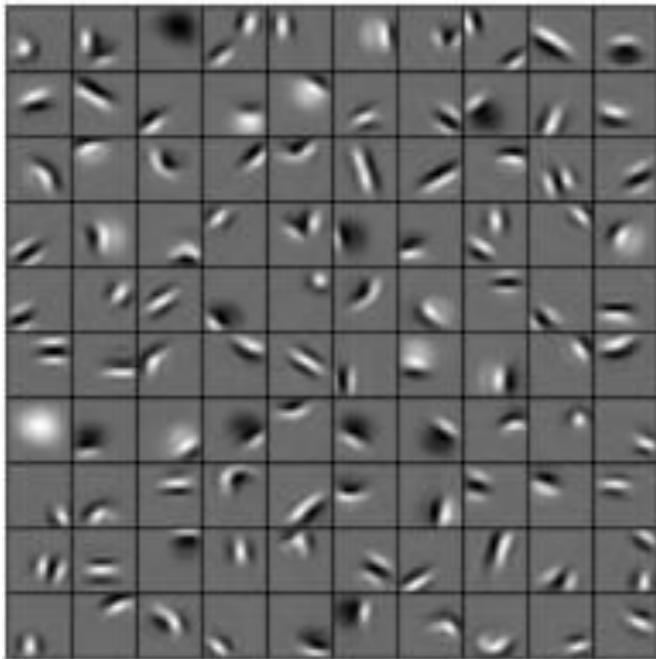


Features de nível baixo

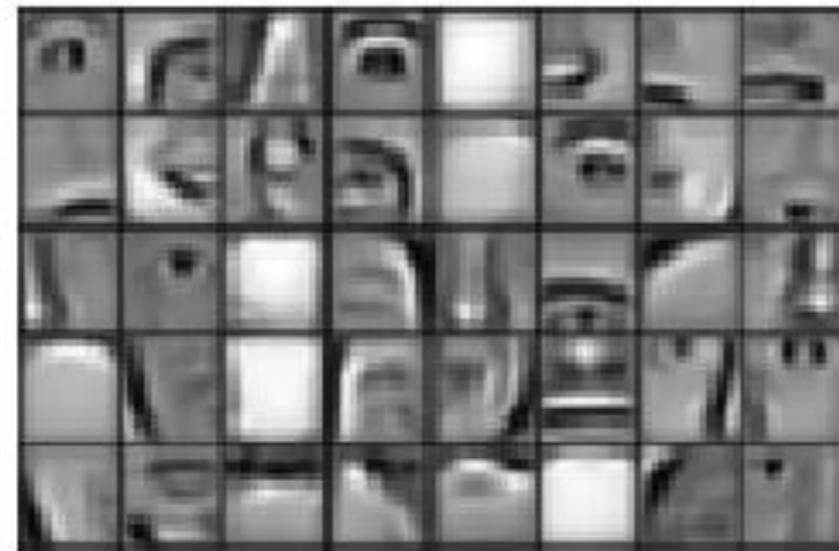
Features de nível médio

Features de nível alto

Arestas



Formas simples:
olho, boca



Faces



Entrada

Ex.: filtro Gabor
(Pesquisar)



Saída

Rede Neural Convolucional

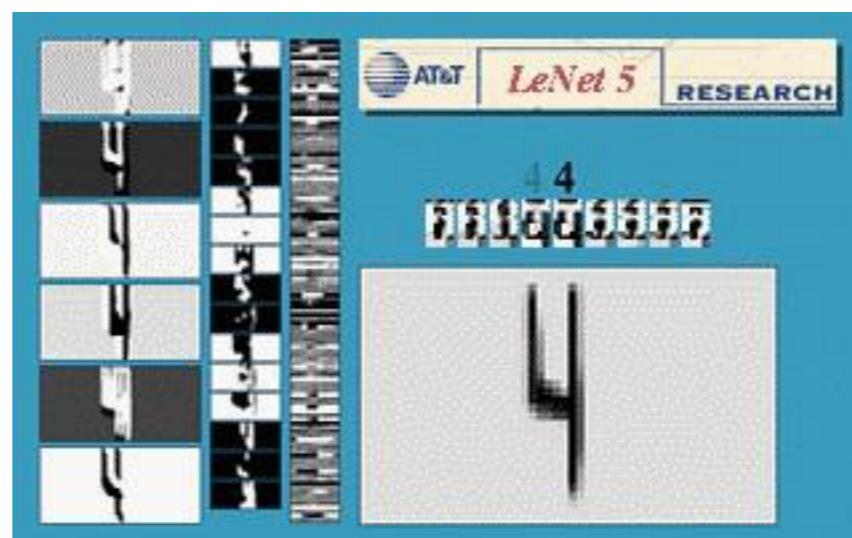
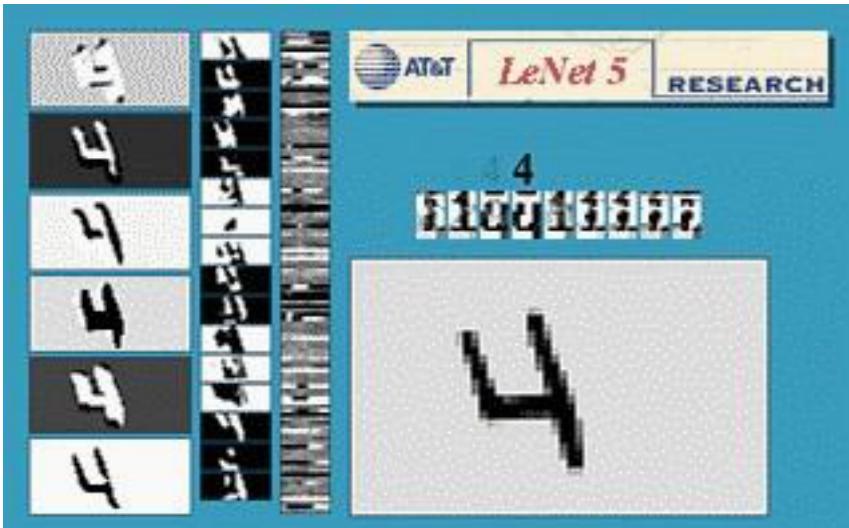


- O uso destas redes tem se difundido extensivamente tanto na academia quanto em empresas e startups, com alto potencial de inovação em diversos segmentos
- Reconhecimento de faces
- Reconhecimento e segmentação de pedestres, faixa, e sinais de trânsito para veículos autônomos
- Monitoramento de grandes áreas via imagens de satélite ou obtida por drones
 - Agricultura e segurança

Rede Neural Convolucional

- Por quê relevante?

- Podem reconhecer padrões com extrema variabilidade (como caracteres manuscritos), sendo robustas a distorções e transformações geométricas



Rede Neural Convolucional

- Para treinamento, usam uma versão do algoritmo de retropropagação dos erros assim como fazem outras redes neurais de treinamento supervisionado
- Inspirada pela organização dos neurônios do cortex visual de animais
- Conectividade local por campos receptivos

Rede Neural Convolucional

- Exemplo: rastreamento da localização uma nave espacial com um sensor laser

$x(t)$: Posição da nave espacial no tempo t

A leitura do sensor apresenta **ruídos**



Média ponderada dos últimas leituras

$$s(t) = \int x(a)w(t-a)da = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

Operação de
Convolução

↓
Mapa de
características

↓
Entrada

↓
Kernel

Rede Neural Convolucional

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

Entrada Kernel

Correlação cruzada

Imagen I

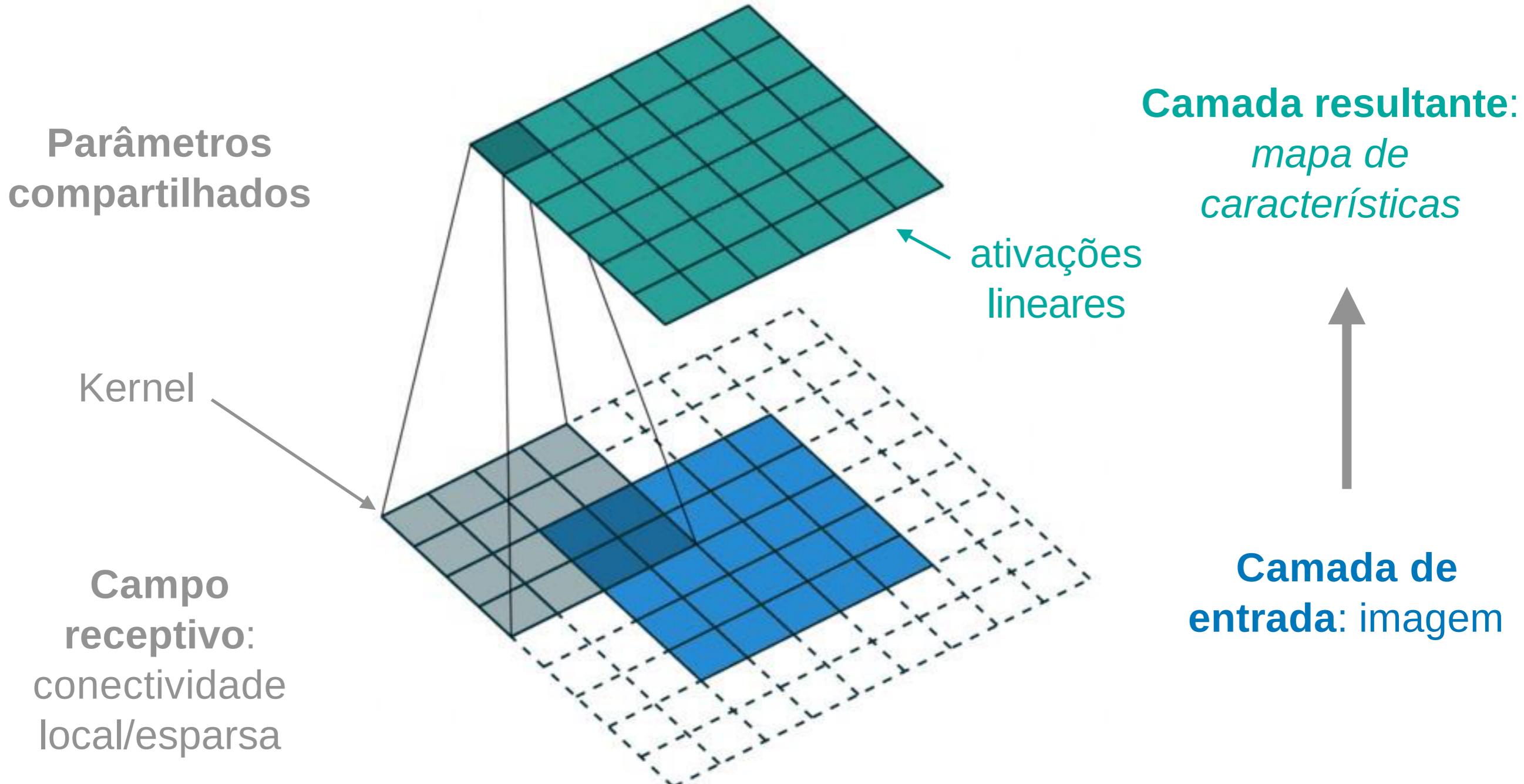
| | | | |
|---|---|---|---|
| a | b | c | d |
| e | f | g | h |
| i | j | k | l |

Kernel K

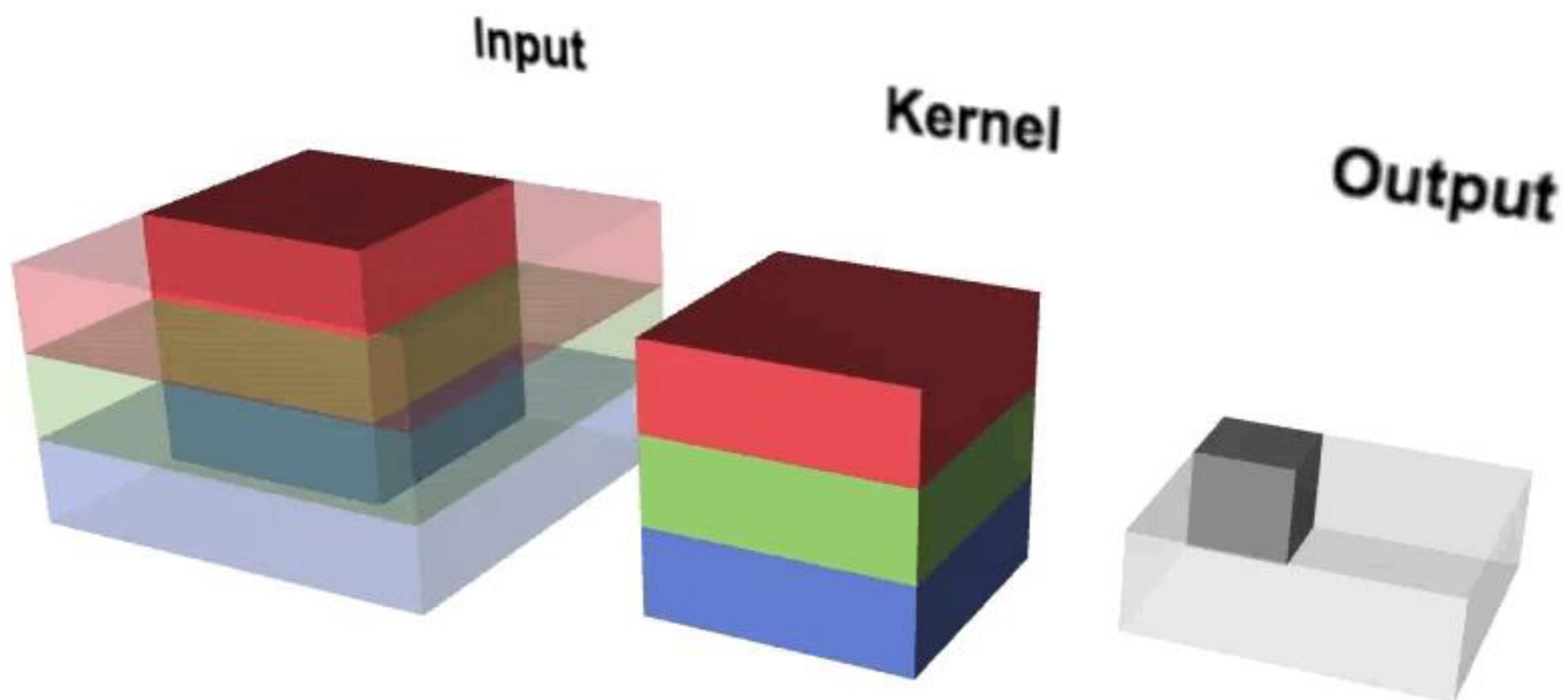
| | |
|---|---|
| w | x |
| y | z |

Operação de Convolução
em 2 dimensões

Rede Neural Convolucional



Rede Neural Convolucional



Rede Neural Convolucional

- E os pesos?

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

- Padding

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

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

- Observe: os pixels da borda são menos usados do que outros pixels da imagem

- Consequências:

- Redução da matriz de saída
- As informações da borda são jogadas fora

| | | | | | |
|---|---|---|---|---|---|
| 3 | 0 | 1 | 2 | 7 | 4 |
| 1 | 5 | 8 | 9 | 3 | 1 |
| 2 | 7 | 1 | 5 | 1 | 3 |
| 1 | 7 | 5 | 4 | 9 | 0 |
| 1 | 3 | 2 | 7 | 5 | 8 |
| 6 | 0 | 2 | 1 | 5 | 7 |

Pesos aprendidos

*

| | | |
|---|---|---|
| ? | ? | ? |
| ? | ? | ? |
| ? | ? | ? |

=

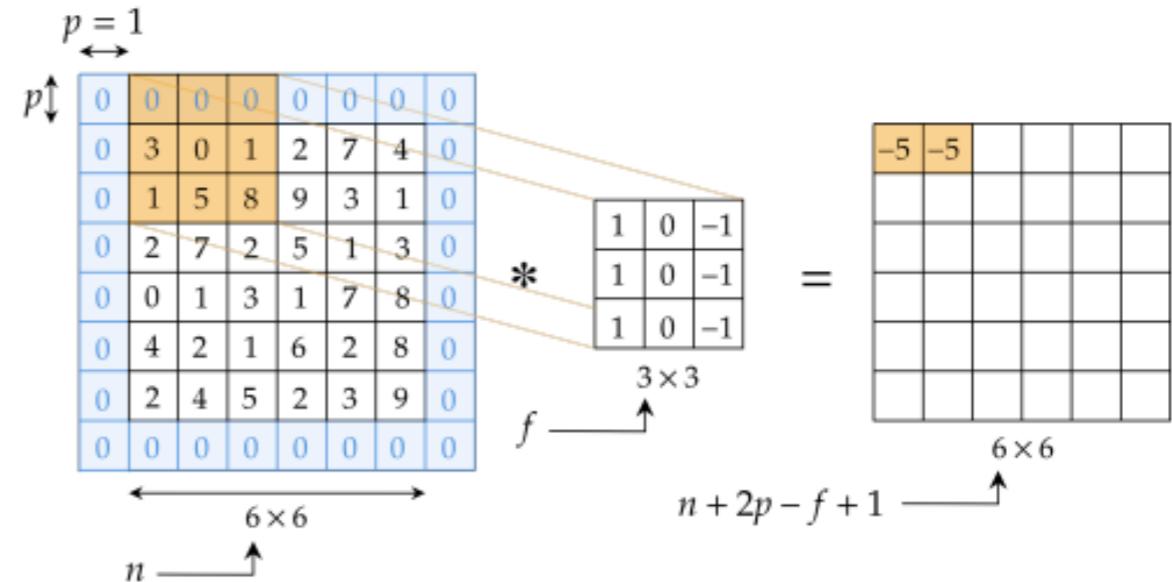
| | | |
|--|--|--|
| | | |
| | | |
| | | |
| | | |

Rede Neural Convolucional

• Padding

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

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



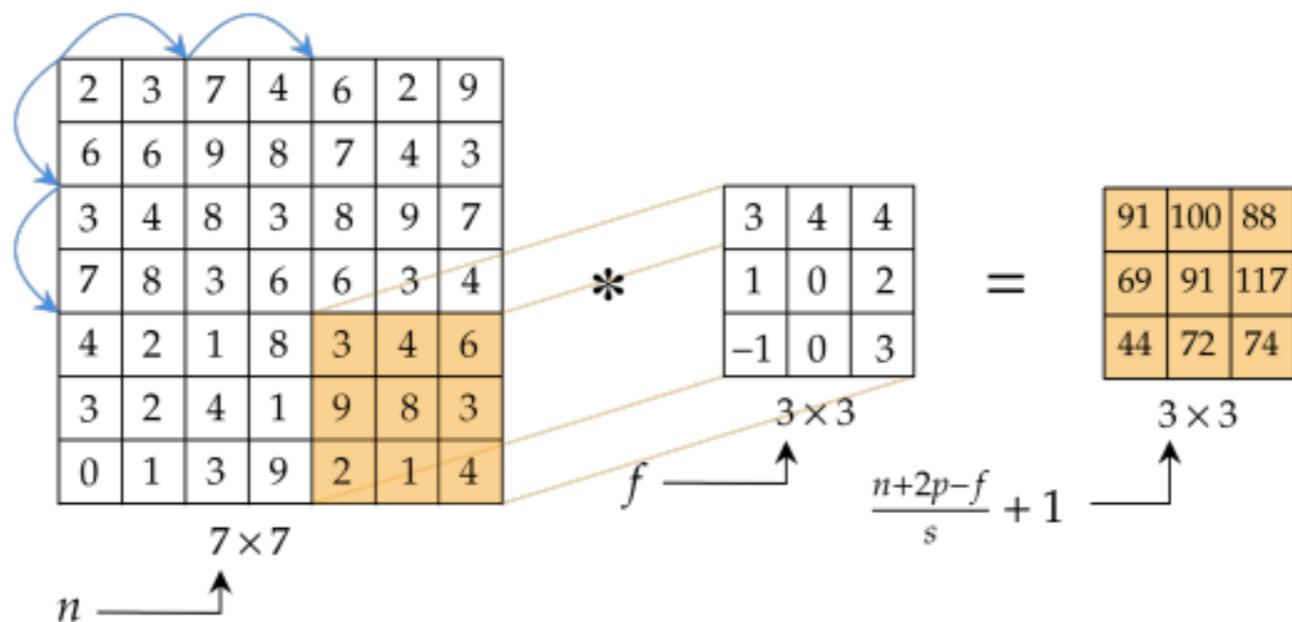
Rede Neural Convolucional

- Stride

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

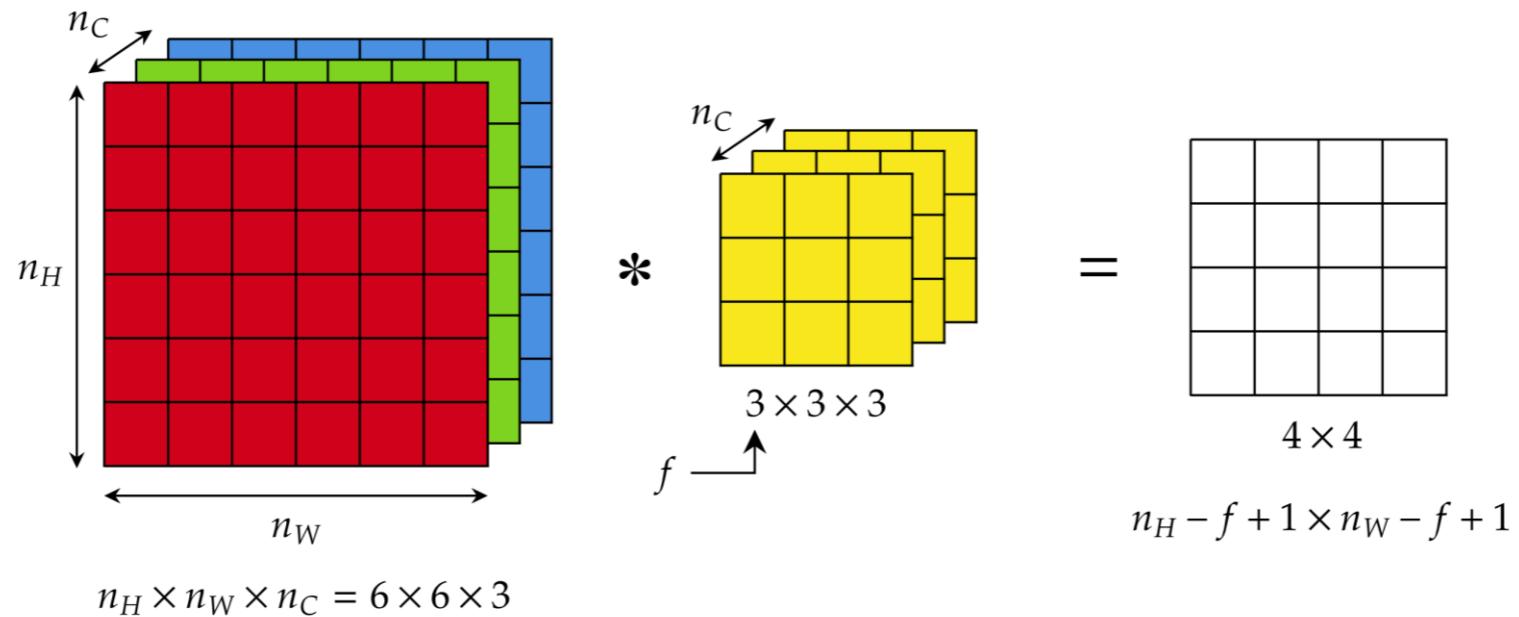
$$\left(\frac{n + 2p - f}{s} + 1 \right) \times \left(\frac{n + 2p - f}{s} + 1 \right)$$

$$s = 2$$



Rede Neural Convolucional

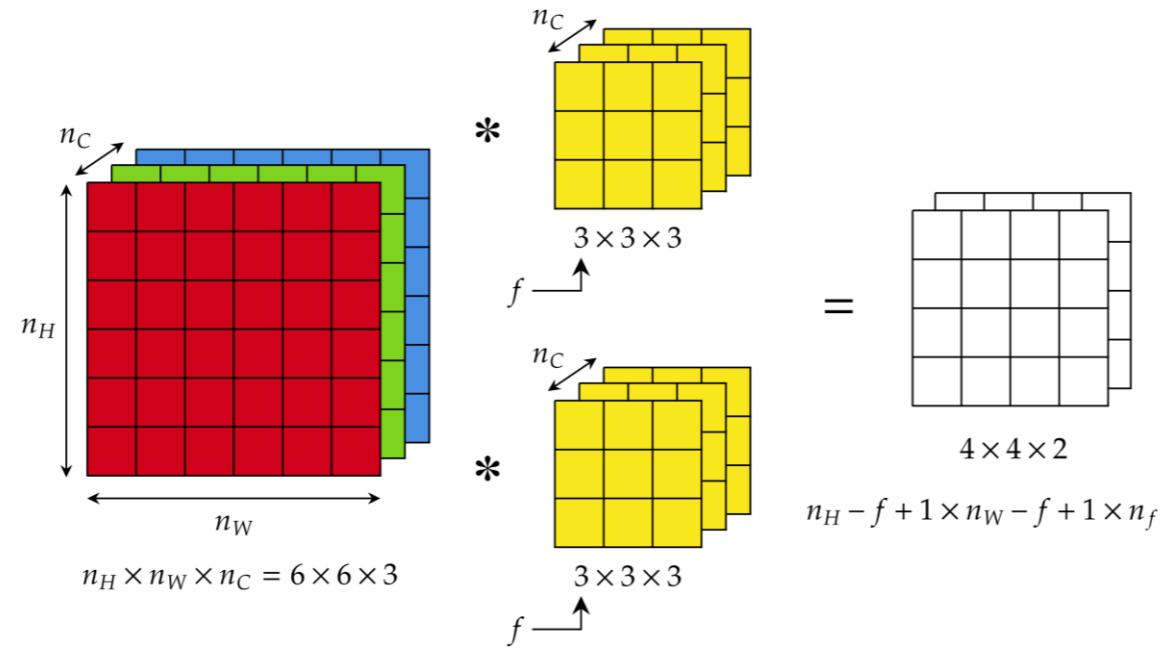
- Convolução sobre volumes



- Imagem 3D (RGB): n_w (largura), n_h (altura) e n_c (canais)
- Observe: o número de canais do filtro deve ser igual ao número de canais da imagem
- A saída é apenas 2D

Rede Neural Convolucional

- Convolução sobre volumes: múltiplos filtros



- Múltiplos filtros para detectar várias *features*

Rede Neural Convolucional

- Camada convolucional:
matematicamente

- Para a l -ésima camada:

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

$$\forall n \in [1, 2, \dots, n_C^l] :$$

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

com

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

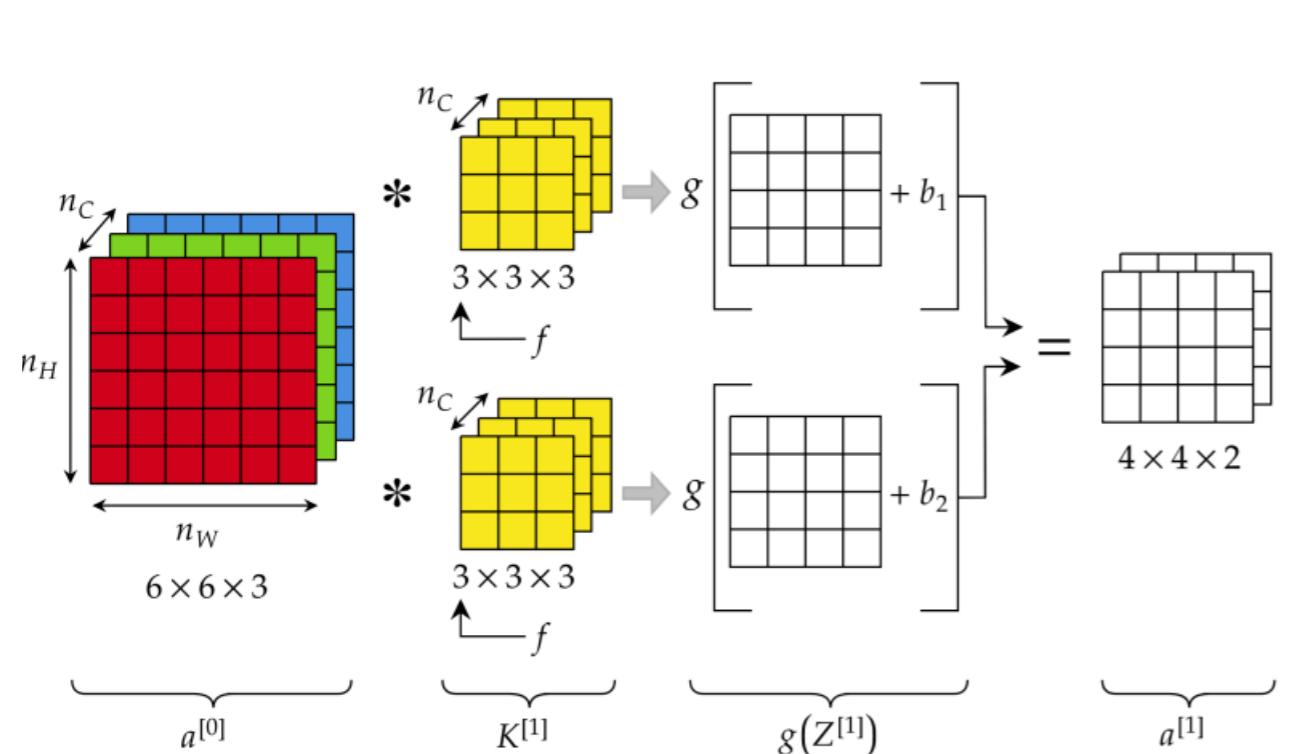
Então:

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

com

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

Rede Neural Convolucional



$\forall n \in [1, 2, \dots, n_C^l] :$

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

$$= Z^{[l]}$$

com

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

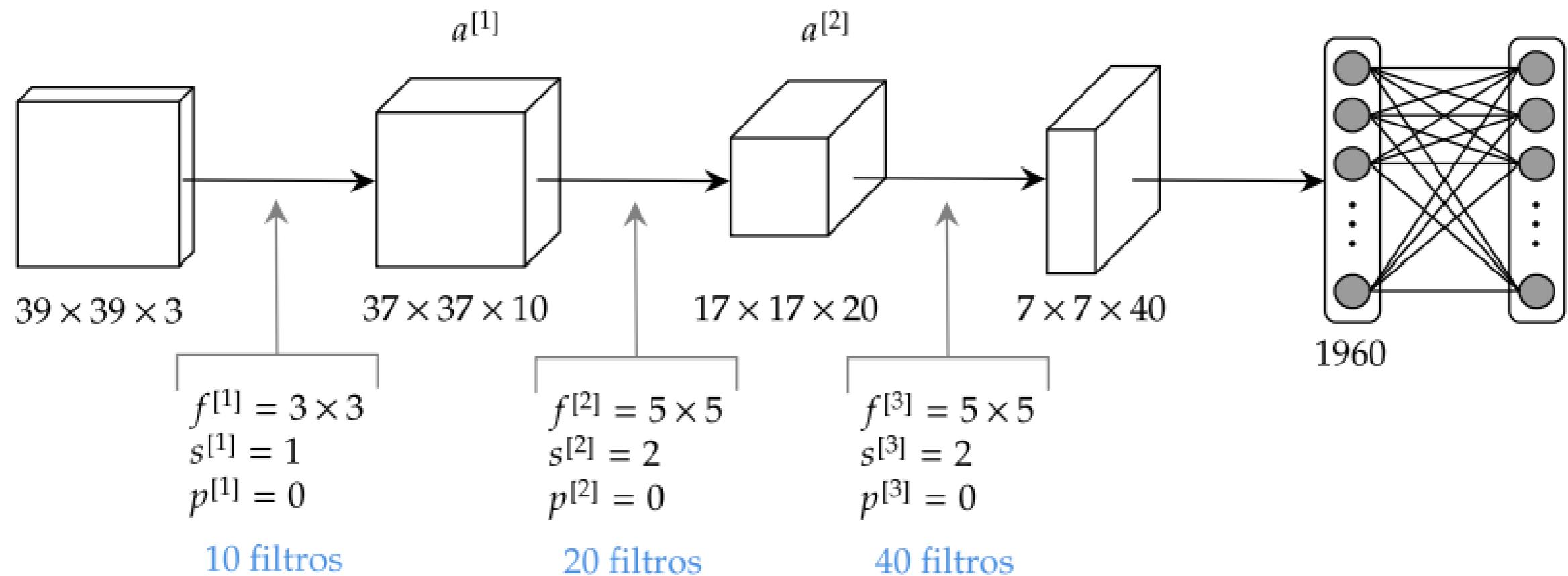
Então:

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

com

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

Rede Neural Convolucional – completa



Rede Neural Convolucional

- Camadas Pooling

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

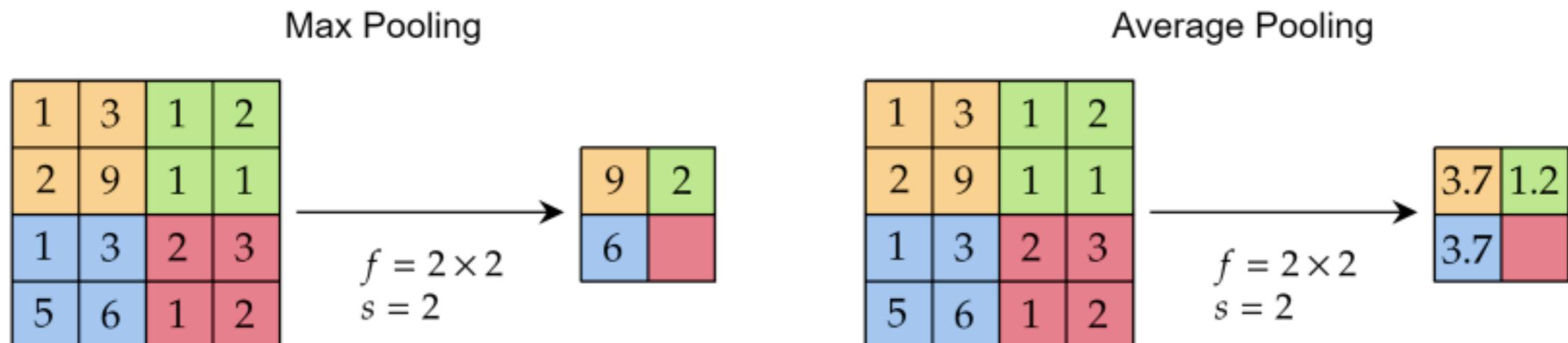
$$\text{dim}(\text{Pooling}) = \left(\left[\frac{n_H + 2p - f}{s} + 1 \right], \left[\frac{n_W + 2p - f}{s} + 1 \right], n_C \right), \text{ p/ } s > 0$$

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

Rede Neural Convolucional

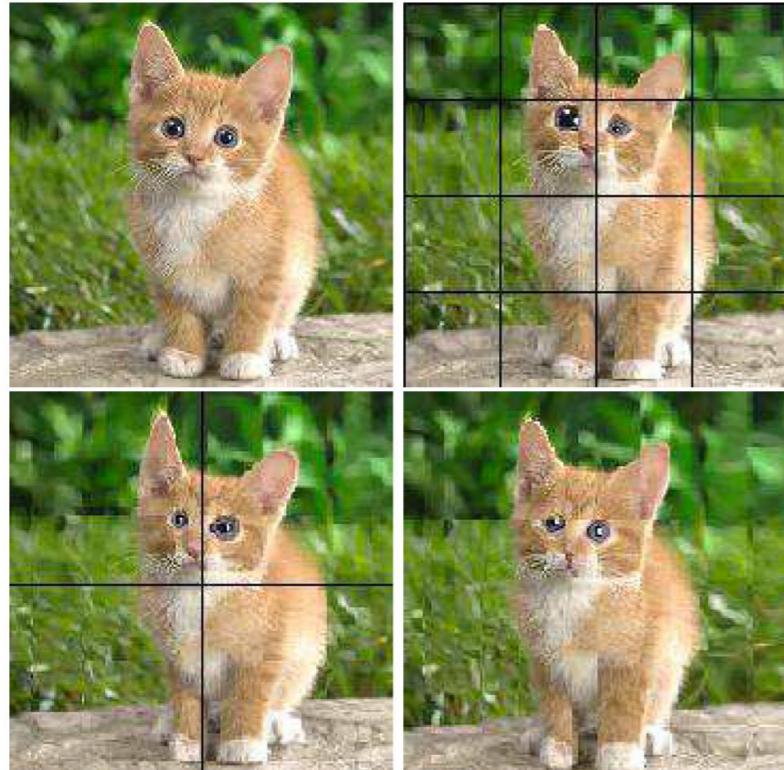
- Camadas Pooling – Tipos mais empregados:

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

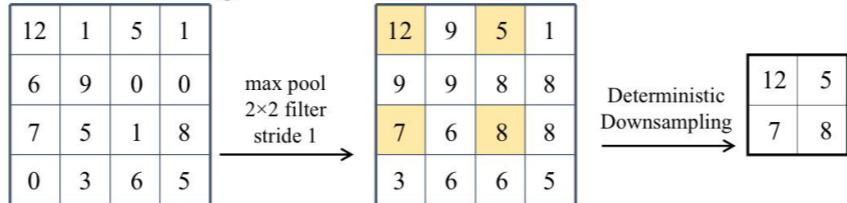


Rede Neural Convolucional

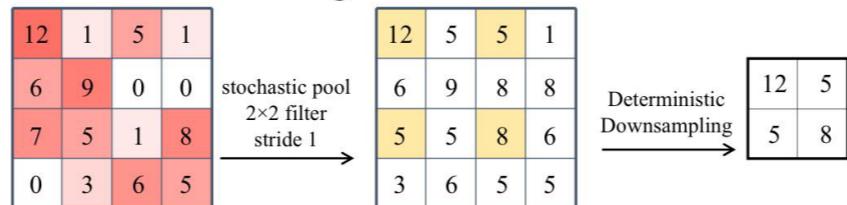
- Camadas Pooling



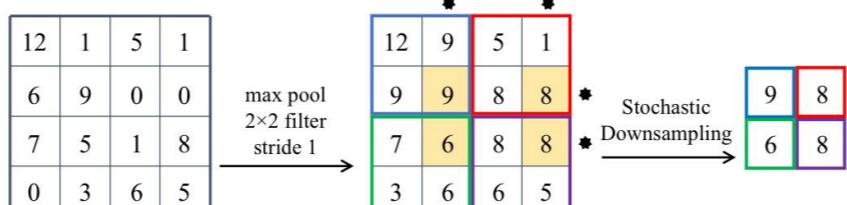
Max Pooling



Stochastic Pooling



S3Pool



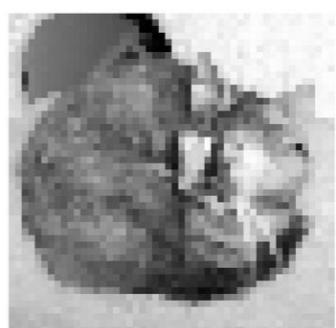
(446, 450)



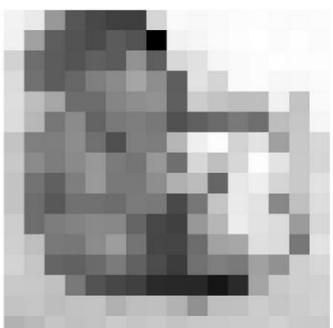
(148, 150)



(49, 50)



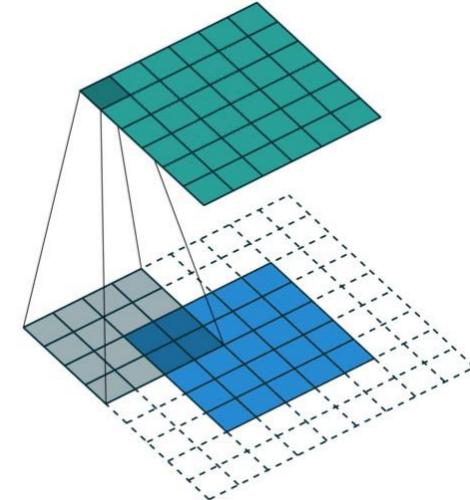
(16, 16)



Rede Neural Convolucional

- Características

- Conectividade esparsa/local: Kernel é menor (centenas de pixels) que a entrada (milhões de pixels), detecta *features* simples como arestas
- Compartilhamento de parâmetros: Economiza memória, e é estatisticamente eficiente (generalização)
- Representações equivariantes a translação
 - Mesmo kernel aplicado a toda entrada detecta as mesmas *features* ao longo da entrada (ex: arestas)
 - Uma translação na entrada implica uma translação na *feature* detectada na camada de saída
- Habilita o processamento de dados que não podem ser lidados com redes com tamanho fixos de matrizes: dados com dimensão de entrada variável



VS. rede MLP

Rede Neural Convolucional

- Operador de convolução usando tensores (matrizes multi-dimensionais)

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

Índice de canal/mapa de features

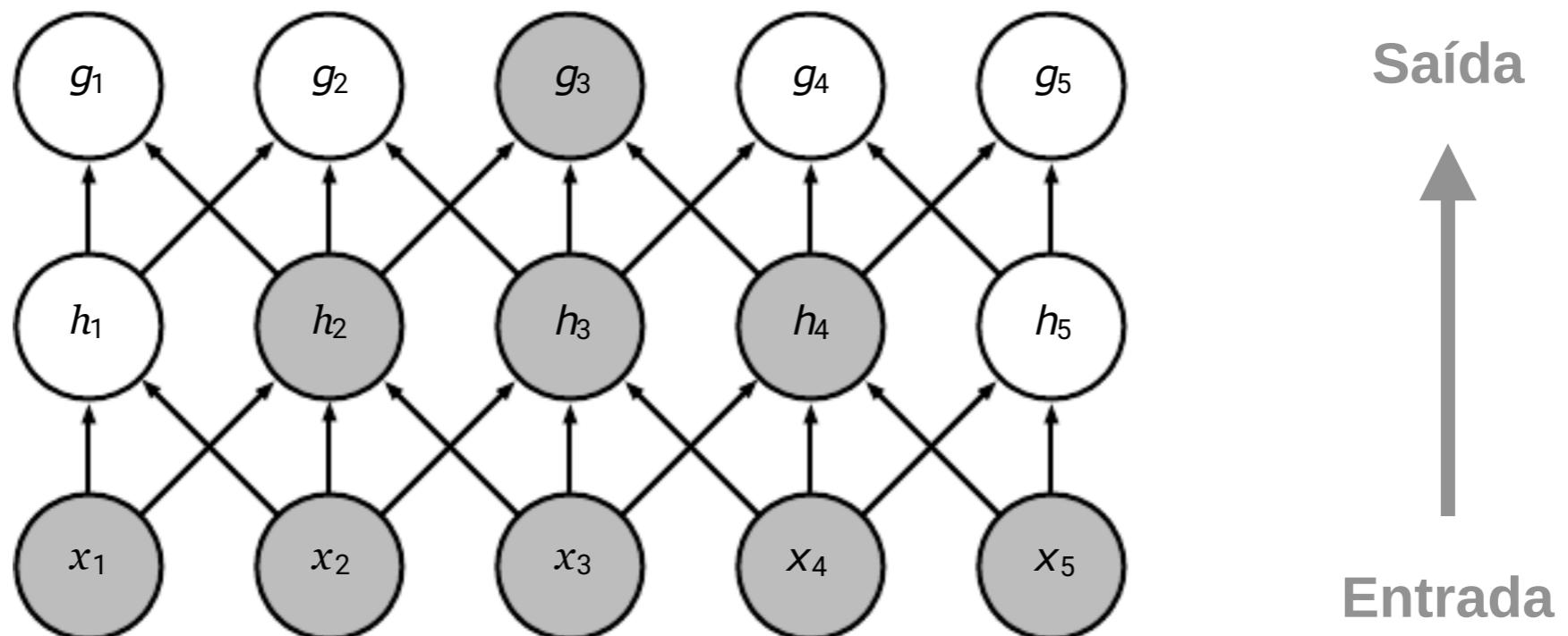
Saída **Entrada observada** **Kernel**

i, j, k: coordenadas espaciais na saída de canal de índice *i*
l, m, n: coordenadas espaciais do Kernel conectando canal de saída de índice *i* ao canal de entrada de índice *l*

- **j, k**: coordenadas espaciais na saída de canal de índice *i*
- **m, n**: coordenadas espaciais do Kernel conectando canal de saída de índice *i* ao canal de entrada de índice *l*

Rede Neural Convolucional

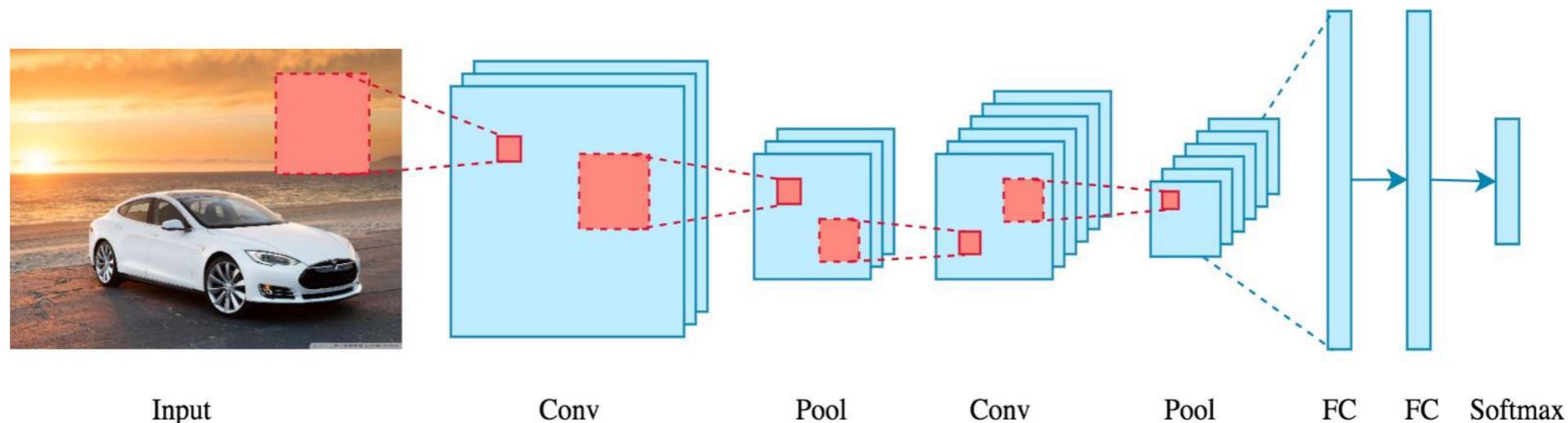
- Campos receptivos de camadas profundas?



Rede Neural Convolucional - Arquitetura

- Uma camada de uma rede convolucional tipicamente consiste de três estágios:

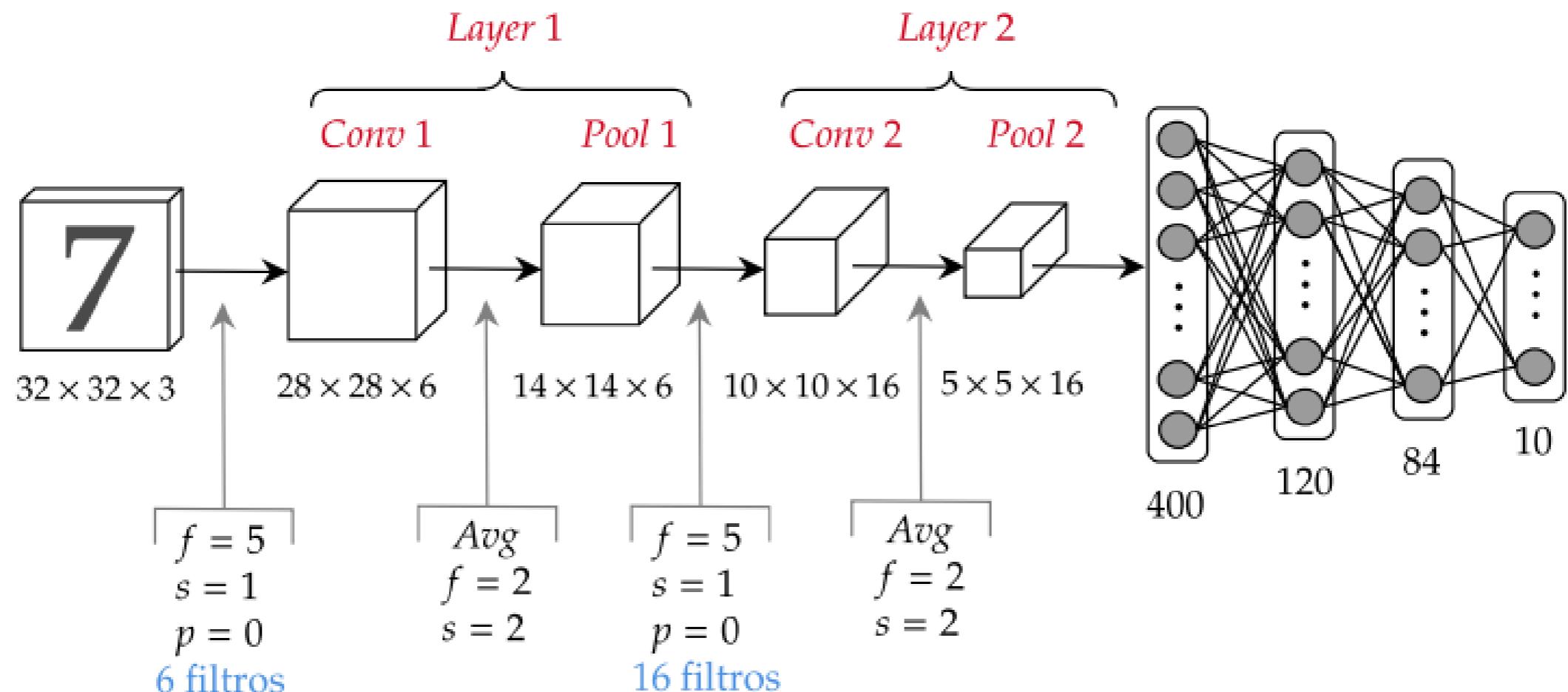
- Múltiplas convoluções são executadas em paralelo para produzir um conjunto de ativações lineares
- Uma função não-linear é aplicada a cada ativação linear (ex.: função de ativação linear retificada)
- Função de pooling modifica a saída produzindo uma estatística das saídas espacialmente próximas (ex.: max-pooling)



Exemplos de CNN

- LeNet-5

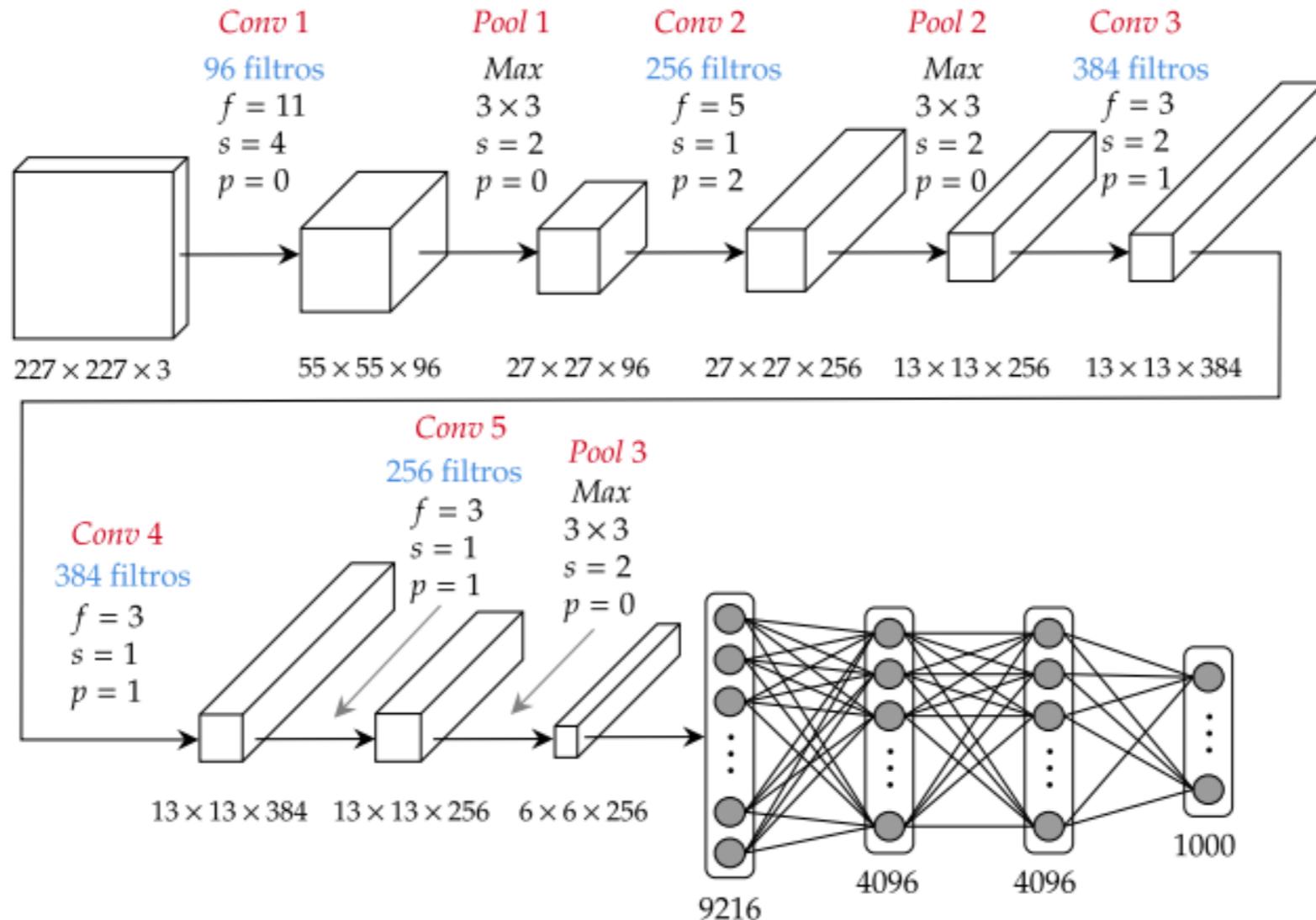
- Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998



Exemplos de CNN

- AlexNet

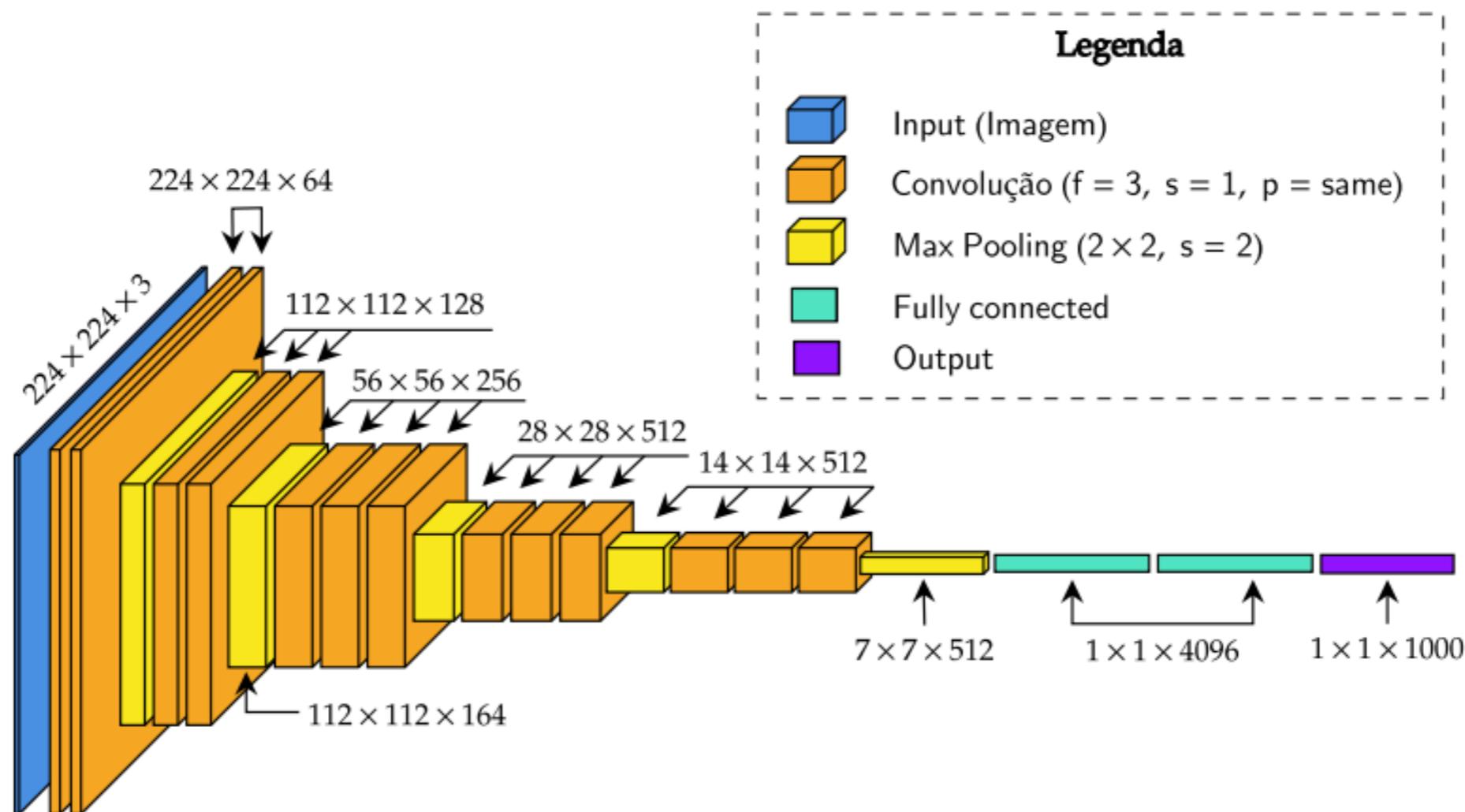
- A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.



Exemplos de CNN

- VGG-16

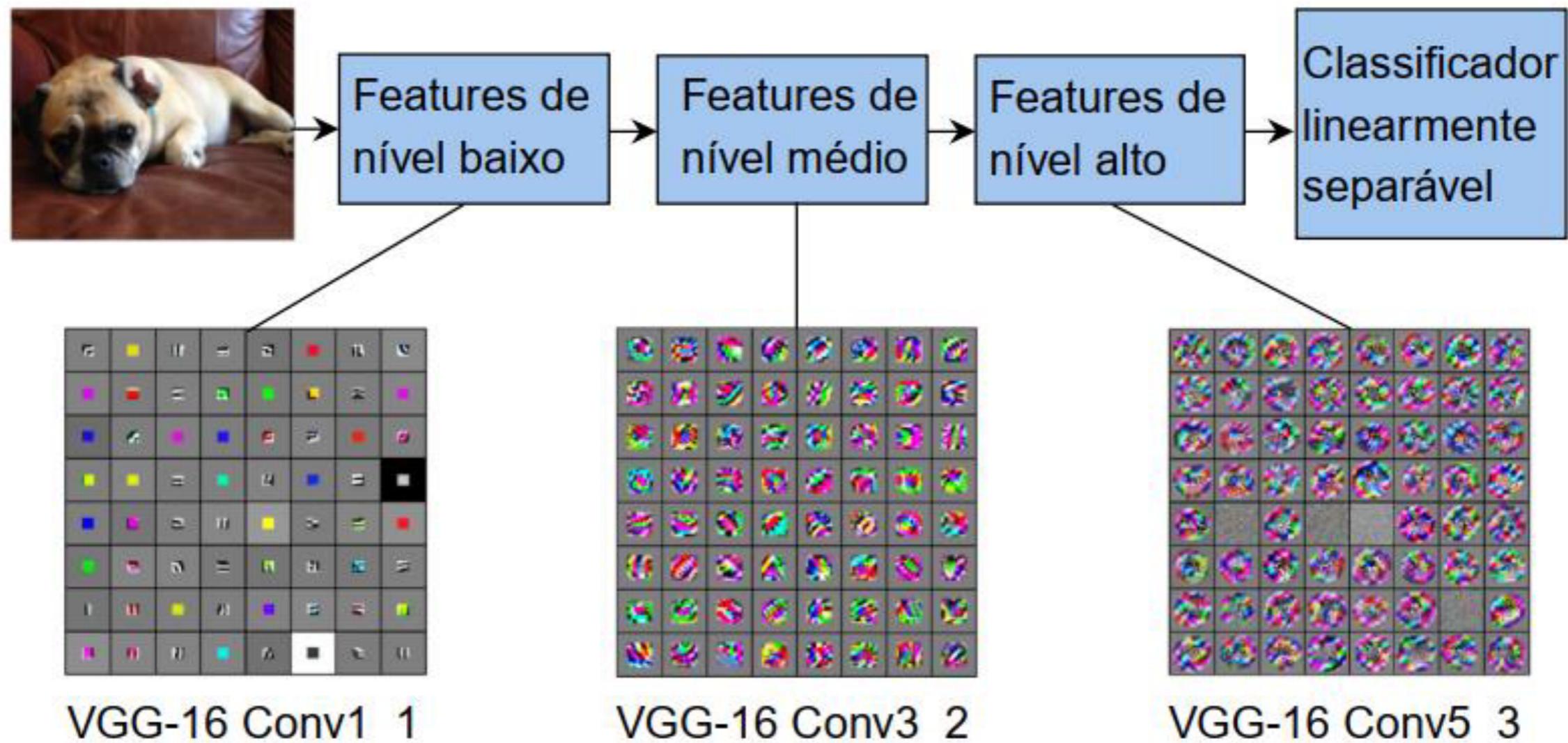
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.



Exemplos de CNN

- VGG-16 - Visualização das features

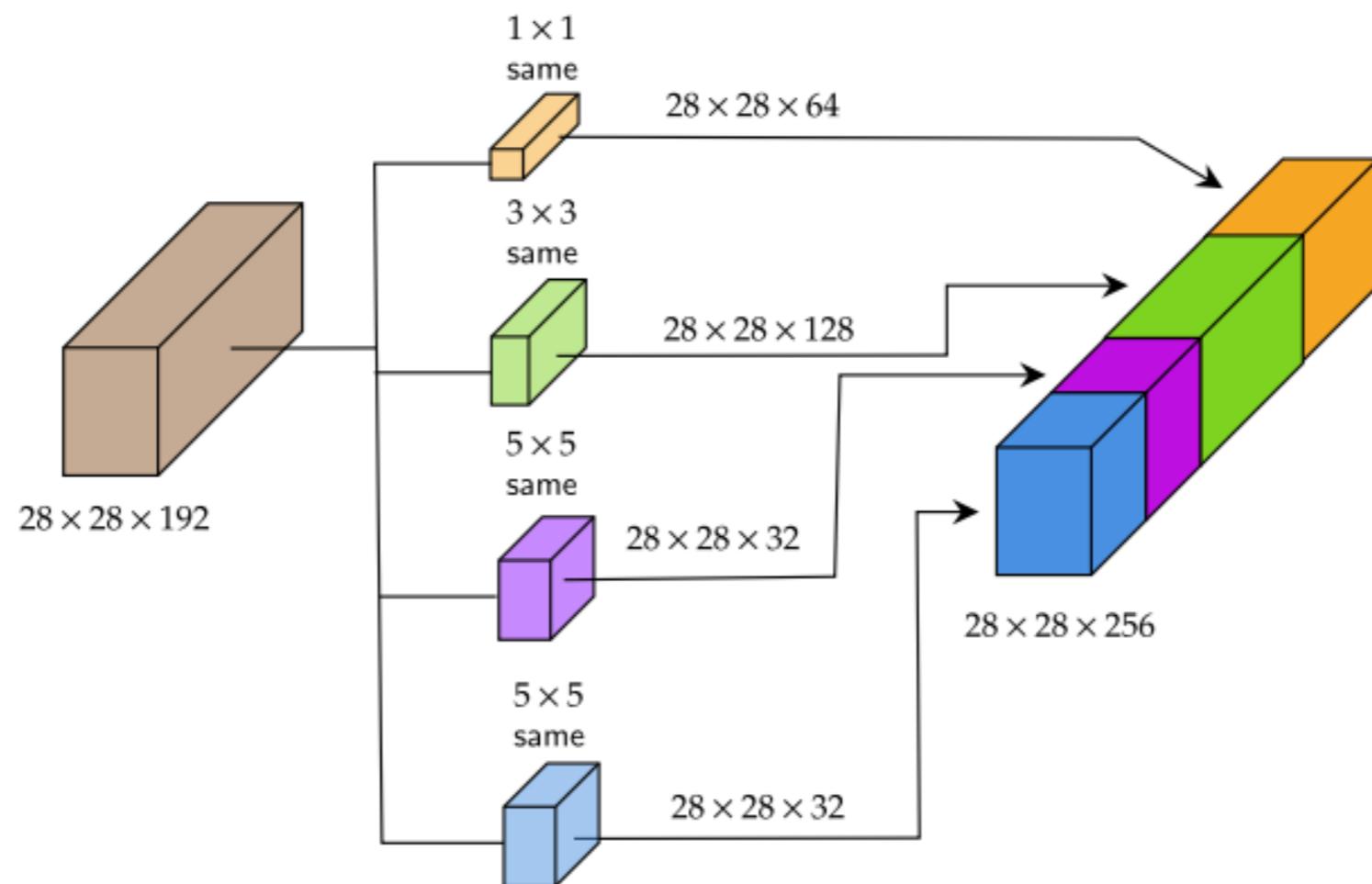
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.



Exemplos de CNN

- Inception Module - Base da rede GoogLeNet

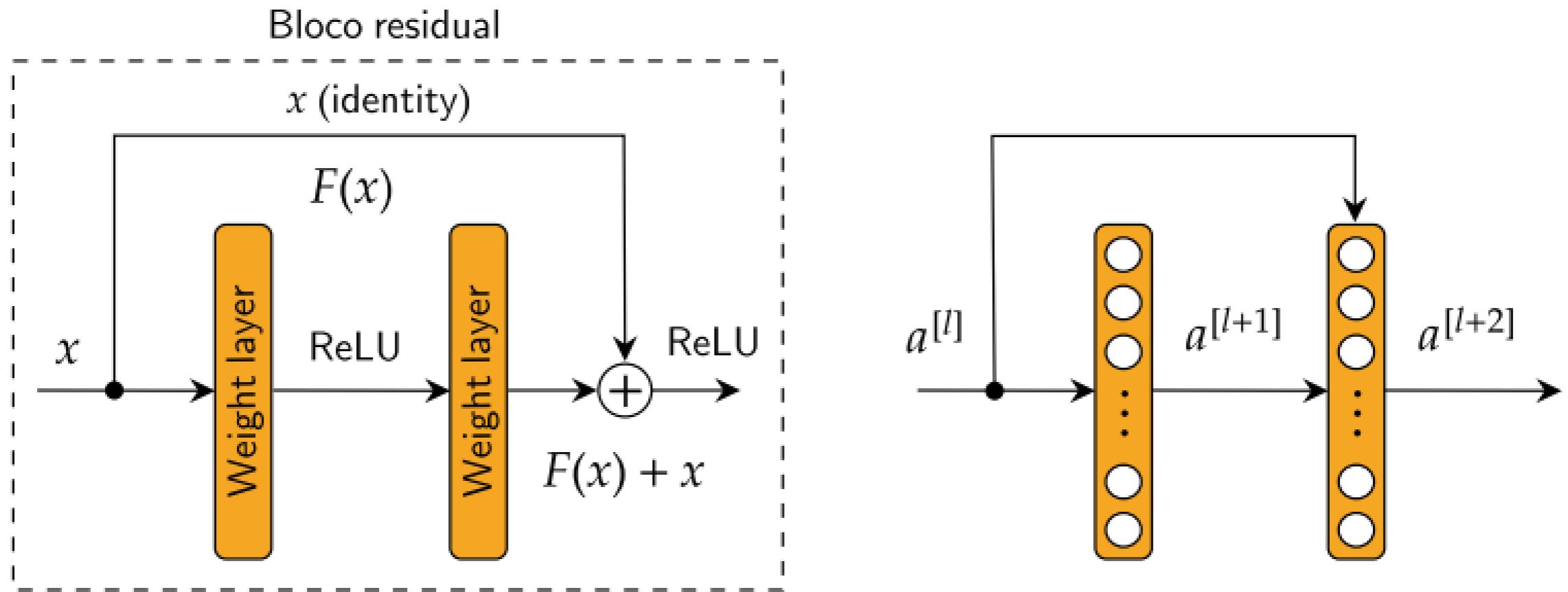
- C. Szegedy, et al. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9, 2015



Exemplos de CNN

- ResNet

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.



Referências principais



- Goodfellow, Ian, et al. **Deep learning**. Vol. 1. No. 2. Cambridge: MIT press, 2016.
- ZHANG, Aston et al. **Dive into deep learning**. arXiv preprint arXiv:2106.11342, 2021.
- PRINCE, Simon JD. **UNDERSTANDING DEEP LEARNING**. MIT PRESS, 2023.