



Redes Neurais Convolucionais com *Keras*: Teoria e Prática

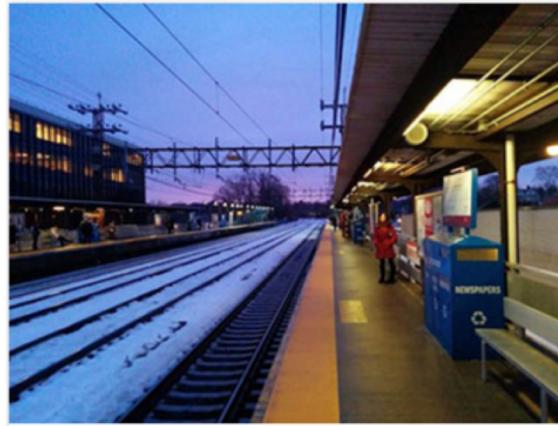
Disciplina: Tópicos Especiais em Visão Computacional

Motivação (Aplicações)

- Descrição de Cenários:



noite ponte cidade ponte suspensa
 rio



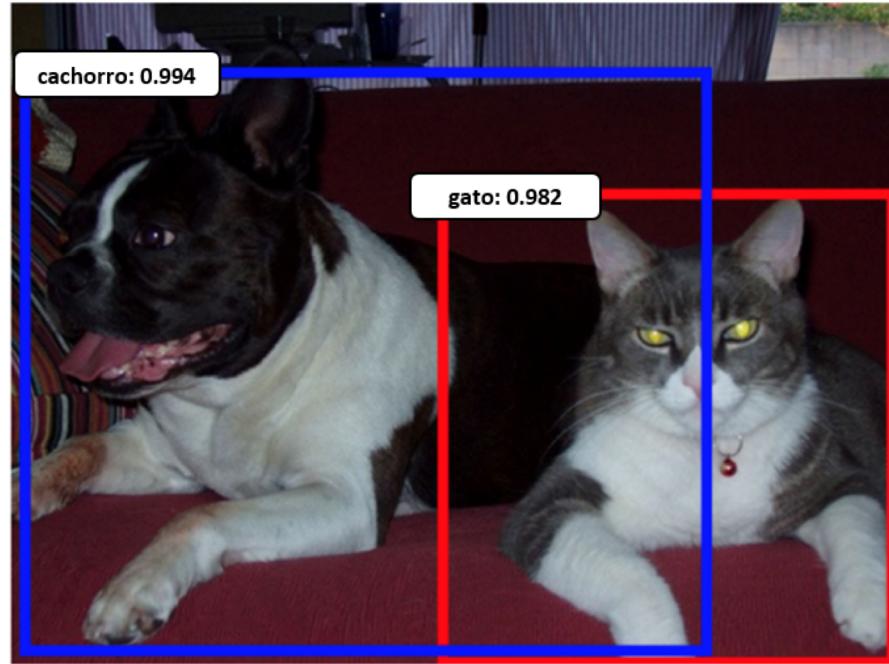
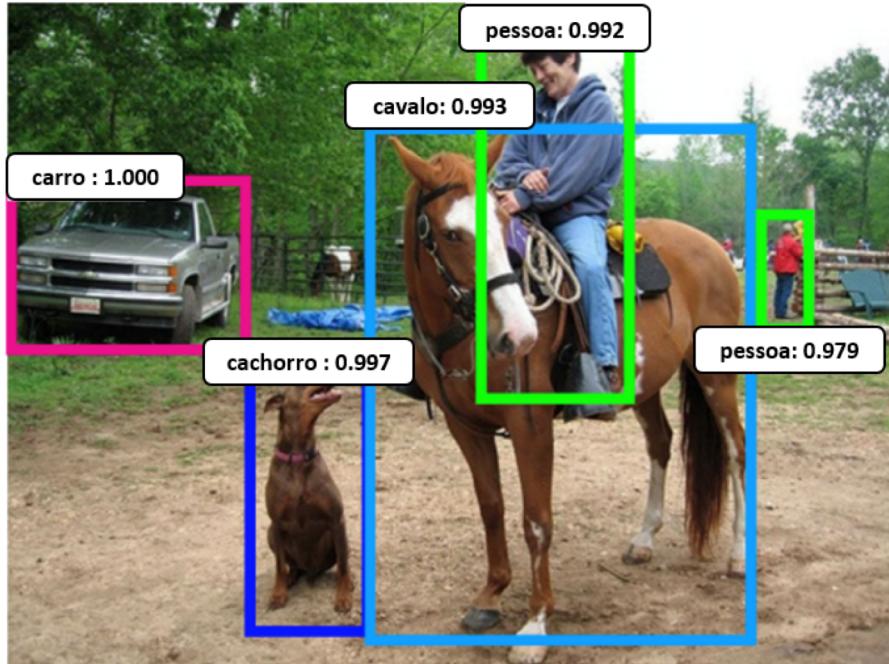
trem metrô ferrovia via férrea
 estação transporte



competição trem tênis estádio bola
 multidão espectadores

Motivação (Aplicações)

- Detecção de objetos:



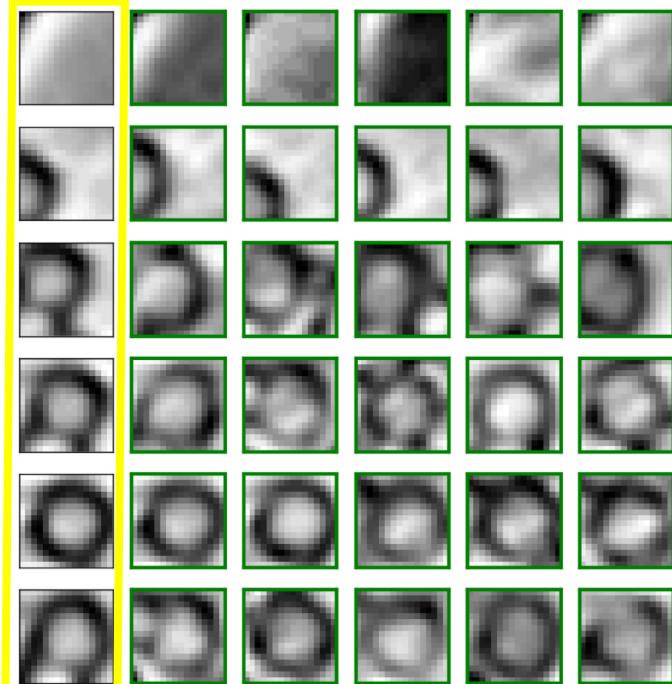
Motivação (Aplicações)

- Recuperação de Imagens Baseado em Conteúdo (CBIR):

Query

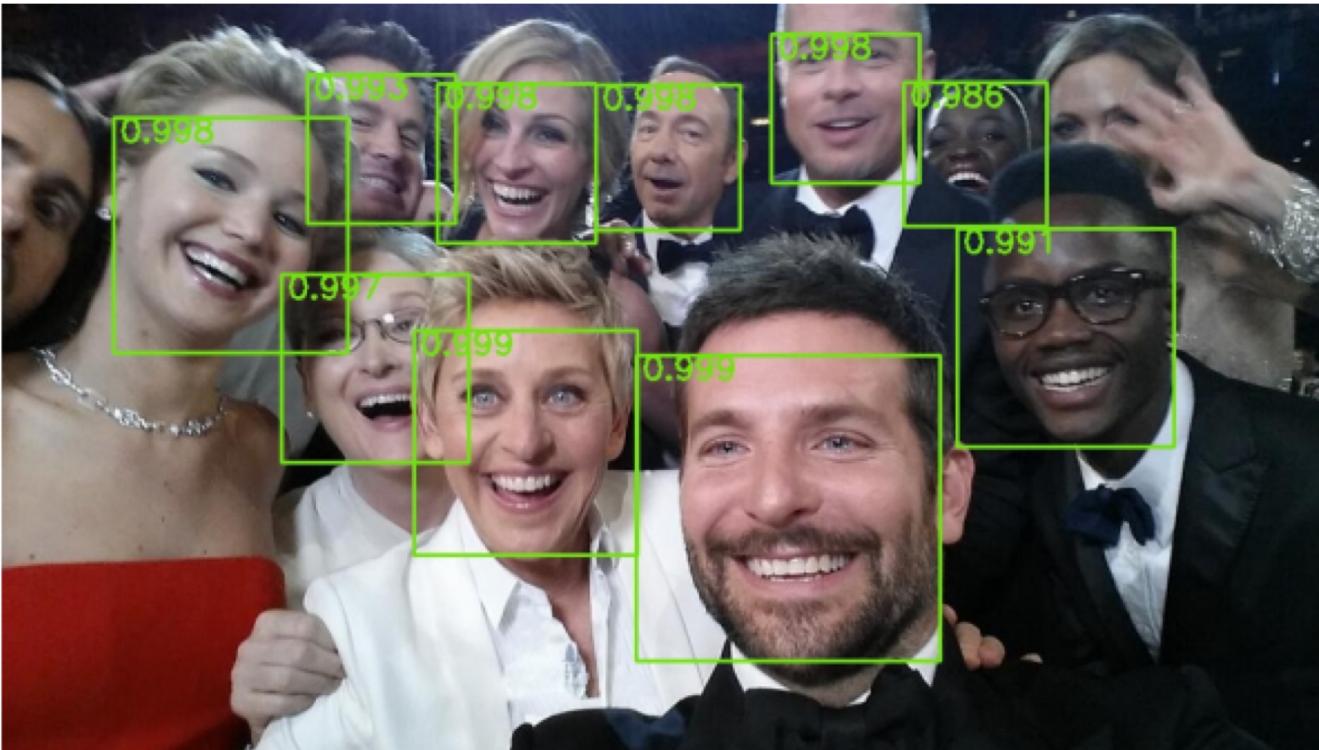


Query



Motivação (Aplicações)

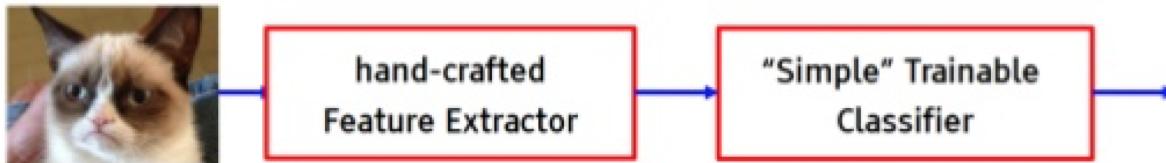
- Detecção e Reconhecimento Facial:



Métodos Tradicional X Deep Learning

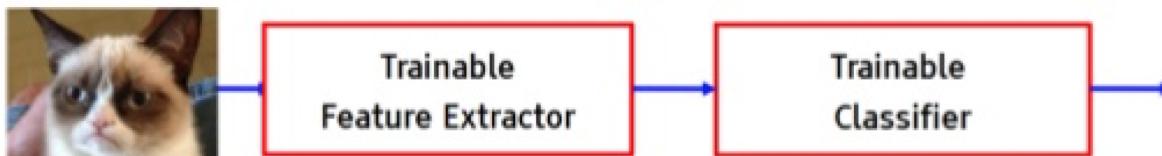
TRADITIONAL APPROACH

The traditional approach uses fixed feature extractors.



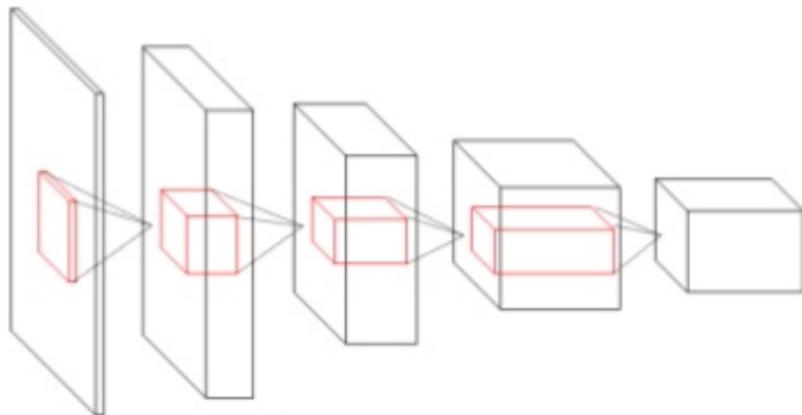
DEEP LEARNING APPROACH

Deep Learning approach uses trainable feature extractors.



O que é *Deep Learning*?

- Múltiplas definições, porém todas possuem em comum:
 - **Múltiplas camadas** de unidades de processamento;
 - As camadas formam uma hierarquia de features *low-level* para *high-level*.



O que é *Deep Learning*?

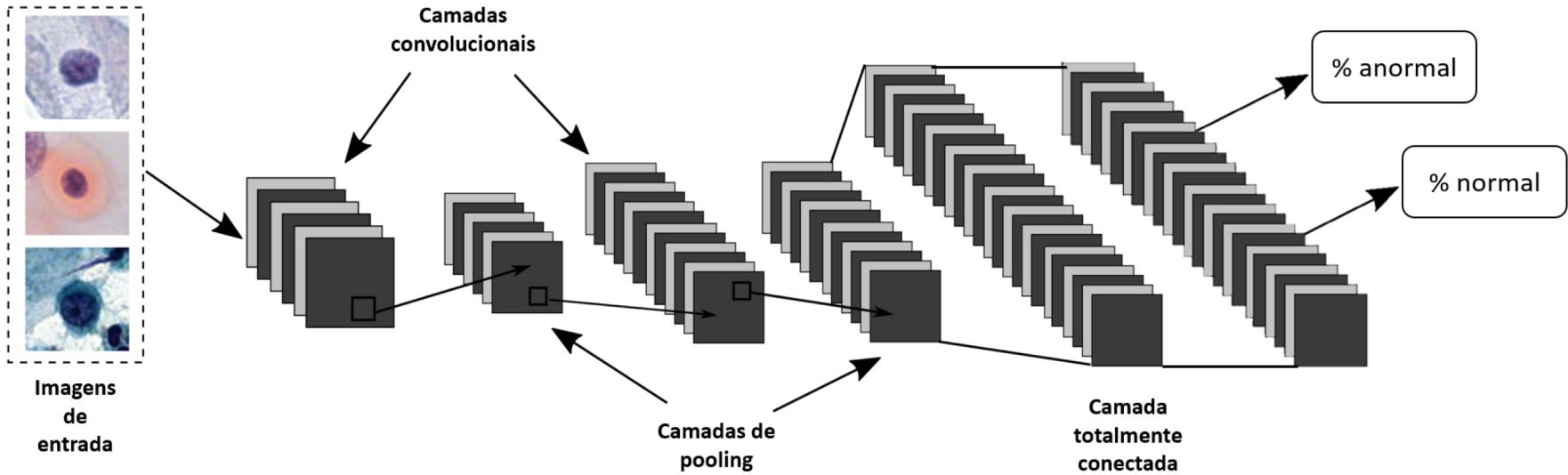
- Em 1998 Yann LeCun e seus colaboradores desenvolveram uma rede para reconhecimento de dígitos manuais:
 - **SVM** classifica corretamente **9.435** de 10.000;
 - **SVM com otimização** de hiperparâmetros classifica corretamente **98.5%**;
 - Atualmente o recorde de acerto é **9.979** de 10.000.



LeNet

Estrutura da *LeNet*

- A *LeNet* é formada por 3 camadas principais e cada uma dessas possui uma função específica na propagação do sinal de entrada:

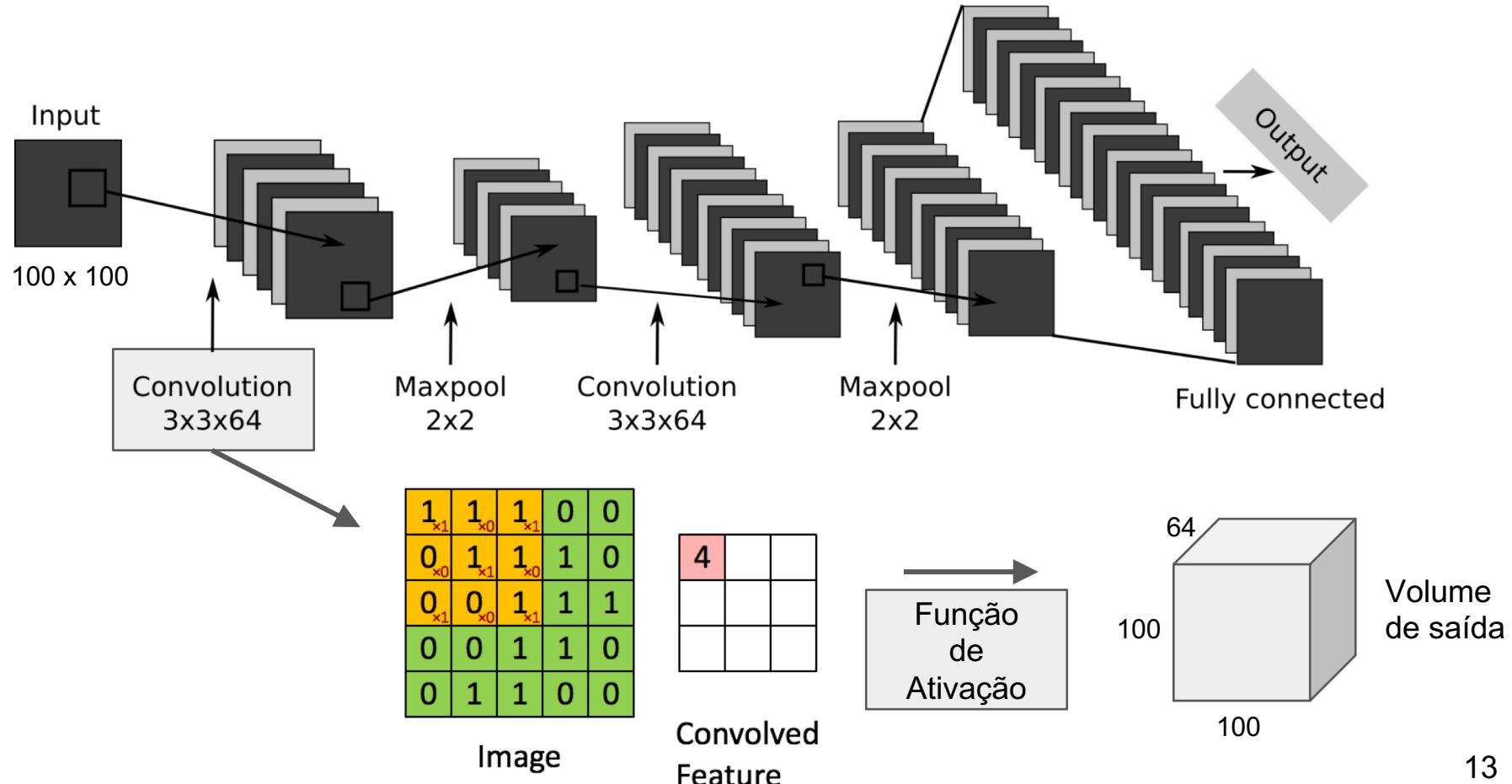


Camada Convolucional

Camada Convolucional

- Consiste num conjunto de filtros com dimensões reduzidas, mas que se estendem por toda a profundidade de um volume de entrada, em outras palavras, se a entrada possui profundidade 3, os filtros também terão profundidade 3;
- Durante o processo de treinamento da rede os valores dos filtros são ajustados para que sejam ativados na presença de características importantes dos volumes de entrada, como orientação de bordas e manchas de cores;
- Nessa camada é realizado a convolução entre os filtros convolucionais e o volume de entrada, em seguida os valores resultantes passam por uma função de ativação.

Camada Convolucional

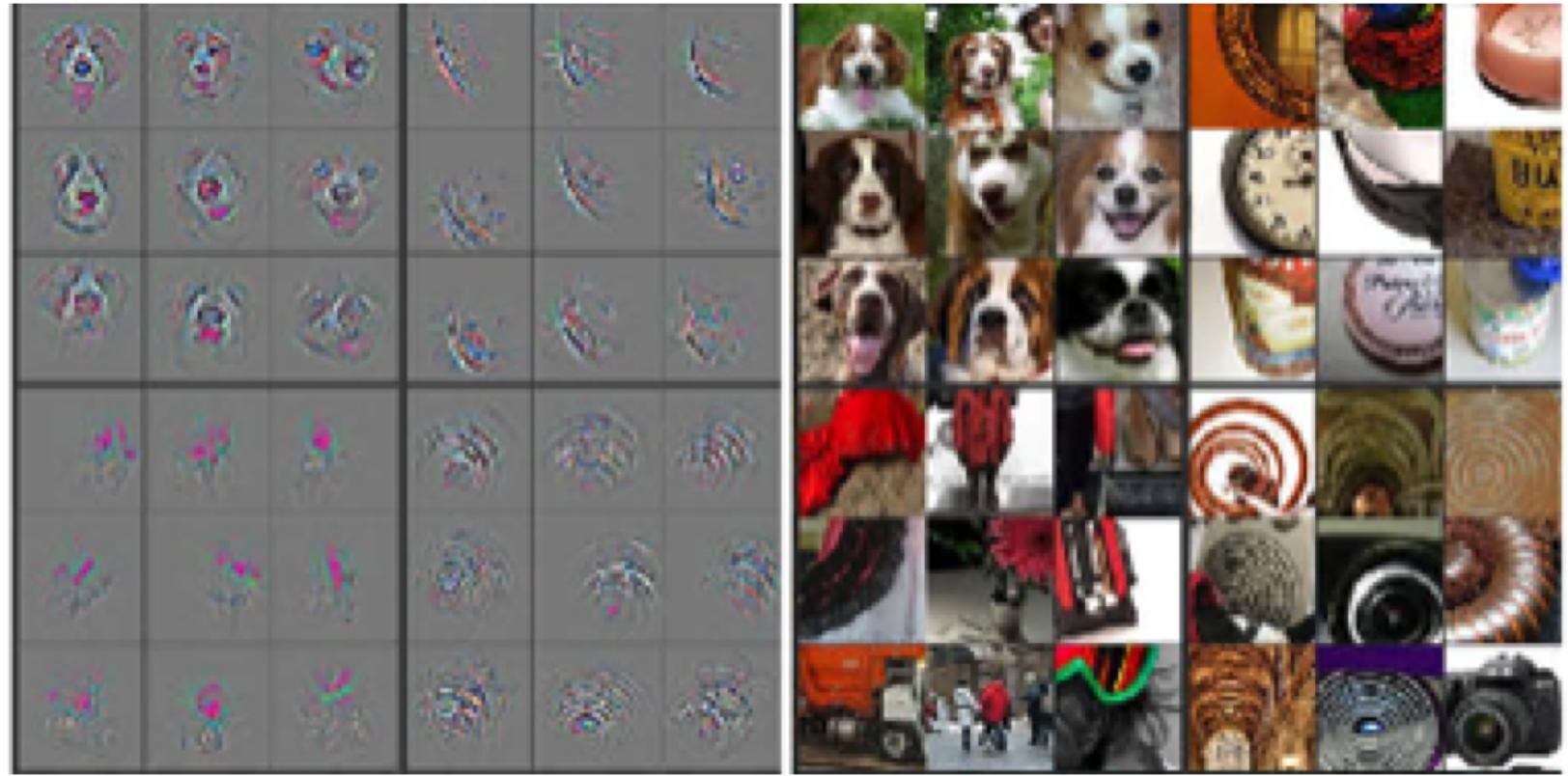


Camada Convolucional

- Existem três parâmetros que controlam as dimensões do volume de saída:
 - **Profundidade (*Depth*)**: quantidade de filtros na camada;
 - **Passo (*Stride*)**: tamanho do salto utilizado na convolução;
 - **Zero-padding**: preenchimento das bordas do volume de entrada com zeros.
- A profundidade do volume de saída sempre é igual a quantidade de filtros convolucionais da camada;

Mapa de Ativação

- Regiões ativadas pelos filtros convolucionais:

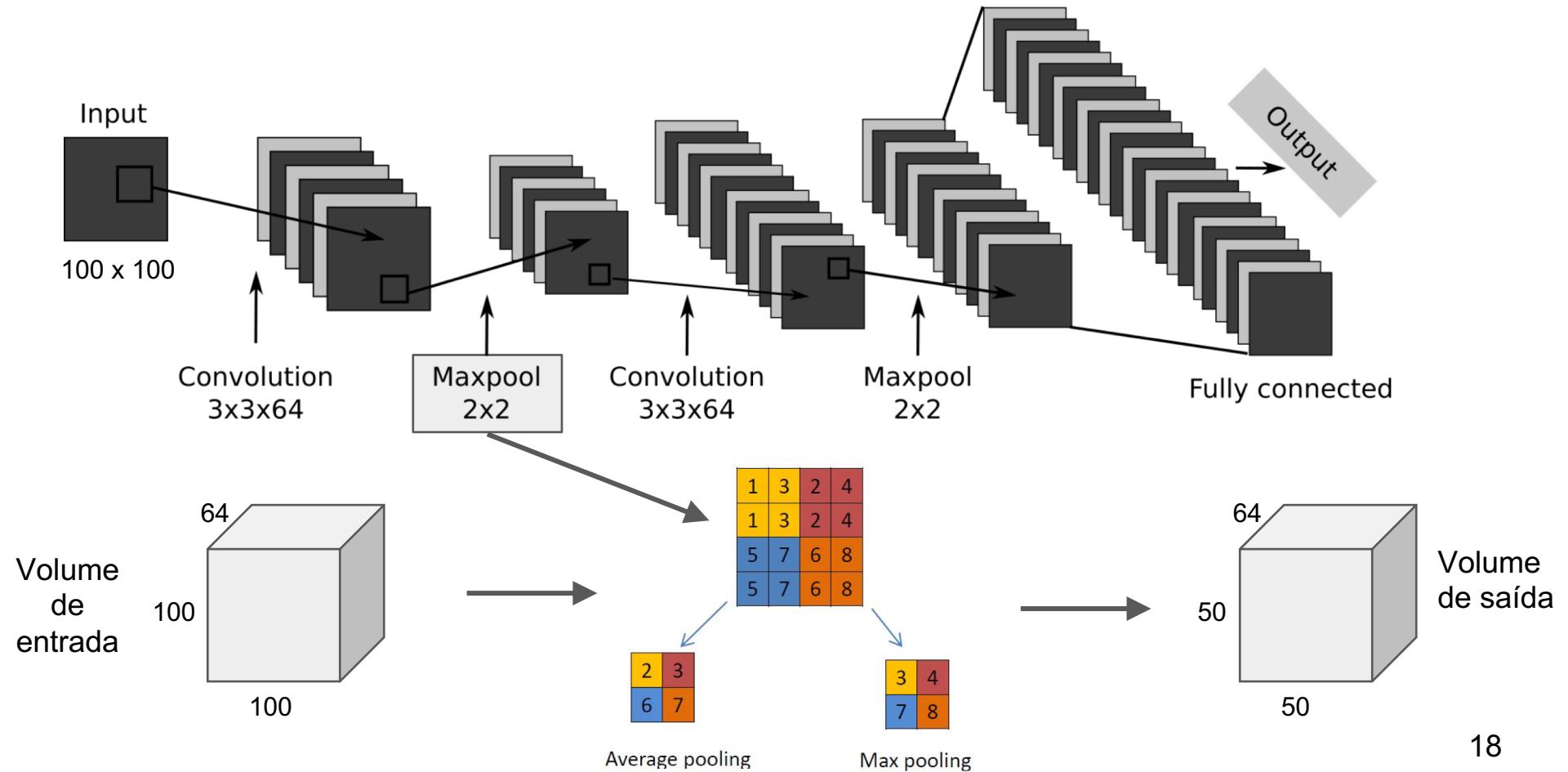


Camada de *Pooling*

Camada de *Pooling*

- Na operação de *pooling*, os valores pertencentes a uma determinada região do mapa de atributos, gerados pelas camadas convolucionais, são substituídos por alguma métrica dessa região;
- Essa operação é útil para eliminar valores desprezíveis, **reduzindo a dimensão** da representação dos dados e **acelerando a computação** necessária para as próximas camadas, além de criar uma **invariância a pequenas mudanças e distorções locais**.

Camada de *Pooling*

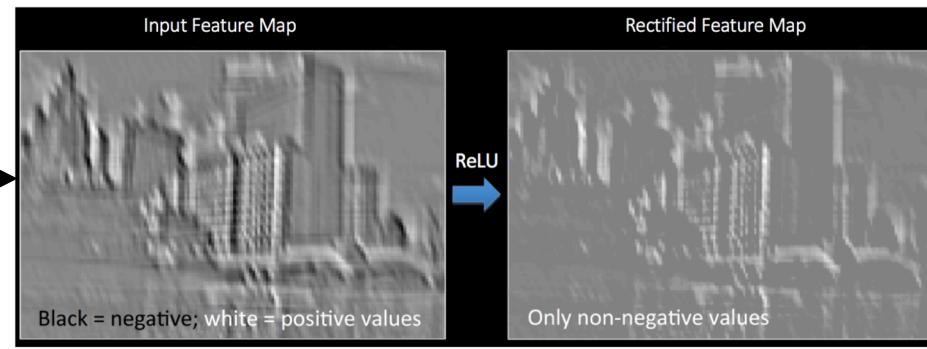


Decomposição da imagem

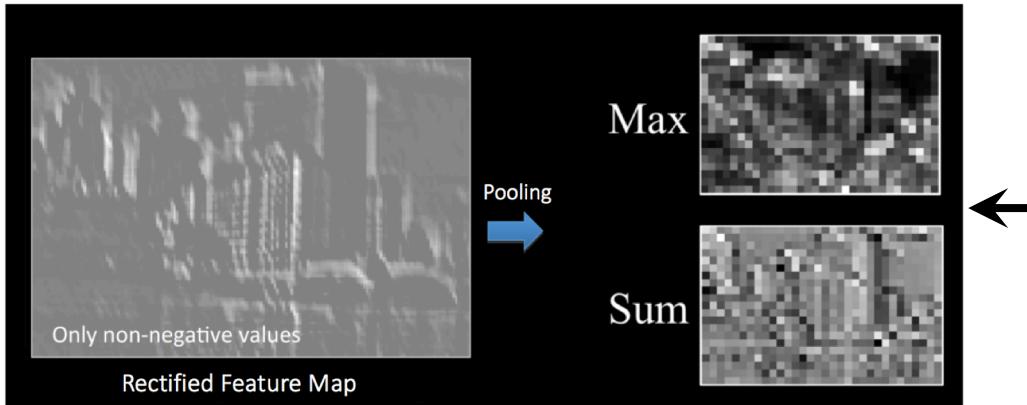
Convolução



Função de ativação



Pooling

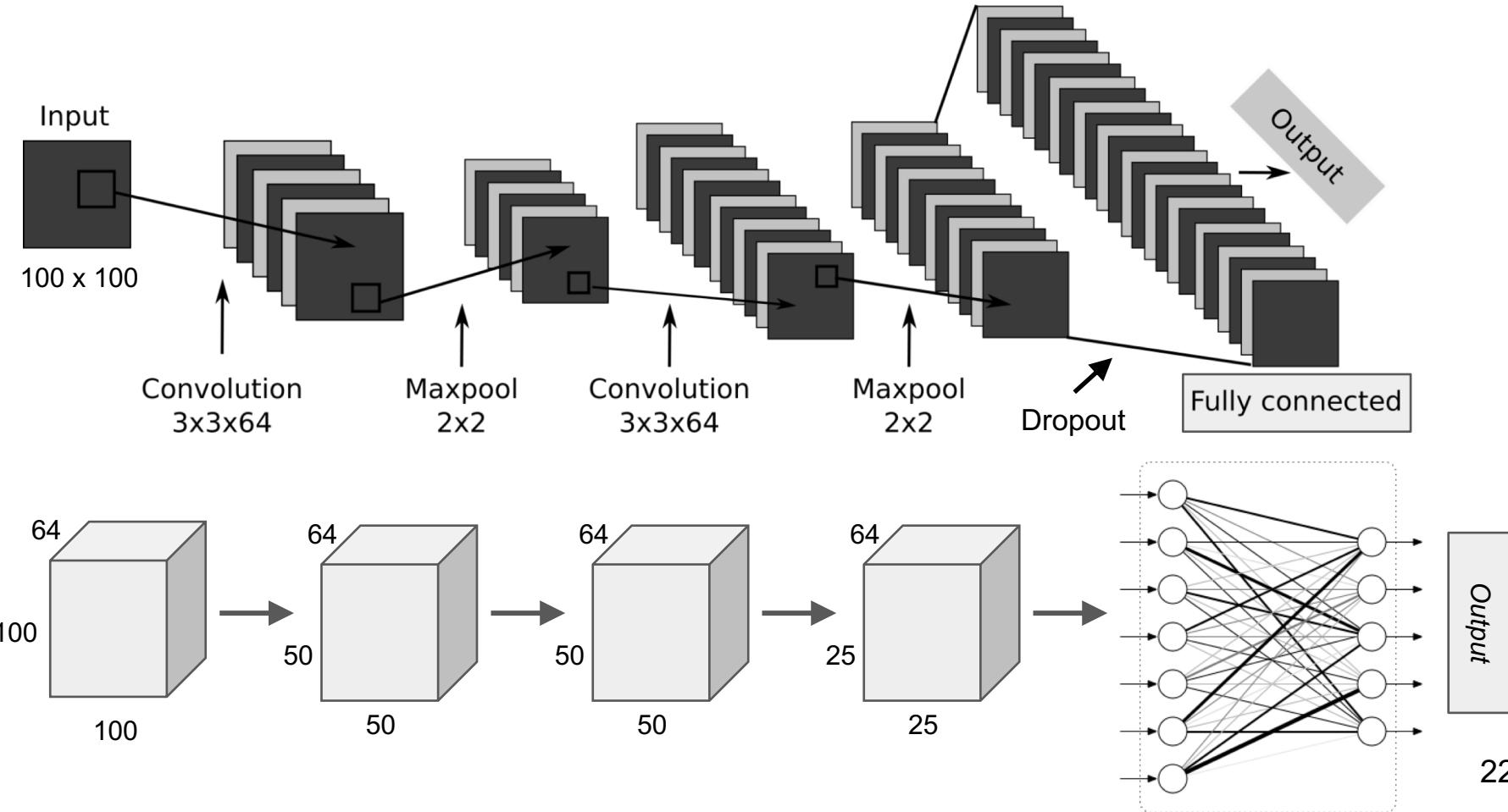


Camada Totalmente Conectada

Camada Totalmente Conectada

- A saída das **camadas convolucionais** e de *pooling* representam os **features extraídos** das imagens de entradas, com isso, o objetivo das camadas **totalmente conectadas** é utilizar essas características para **classificar a imagem** em uma classe pré-determinada;
- Essas camadas são formadas por unidades de processamento conhecidas como **neurônio**, e o termo "totalmente conectado" significa que **todos os neurônios** da uma camada **estão conectados a todos os neurônios** da camada seguinte.
- A **última camada** da rede utiliza **softmax** como função de ativação. Essa função recebe um vetor de valores e produz a **distribuição probabilística** da imagem de entrada pertencer a cada uma das classes na qual a rede foi treinada.

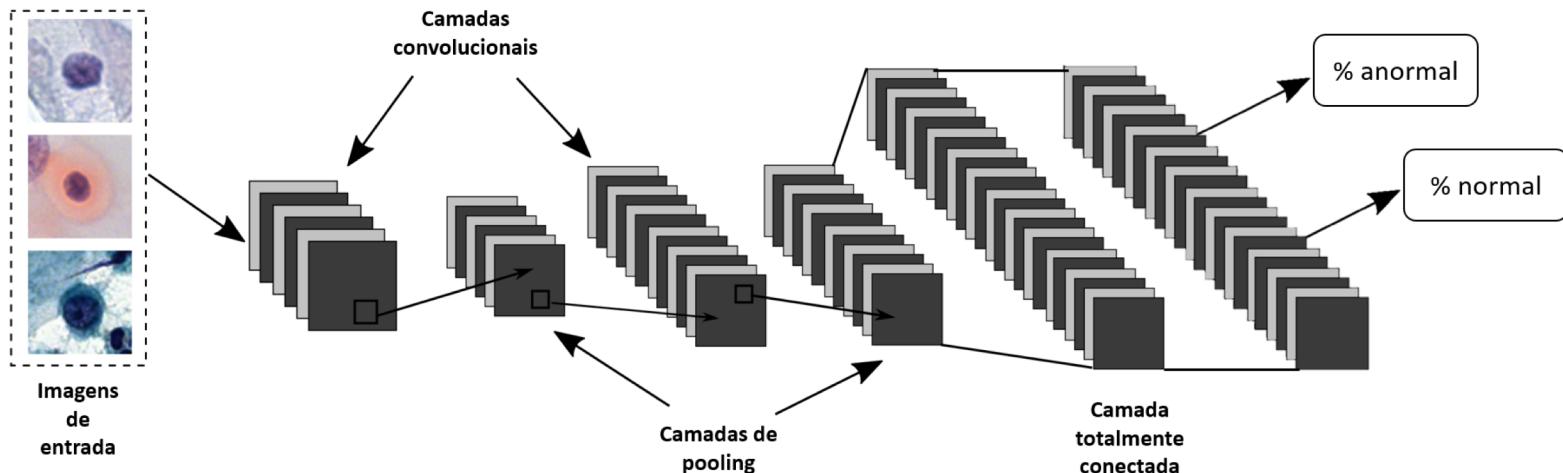
Camada Totalmente Conectada



Treinando a CNN

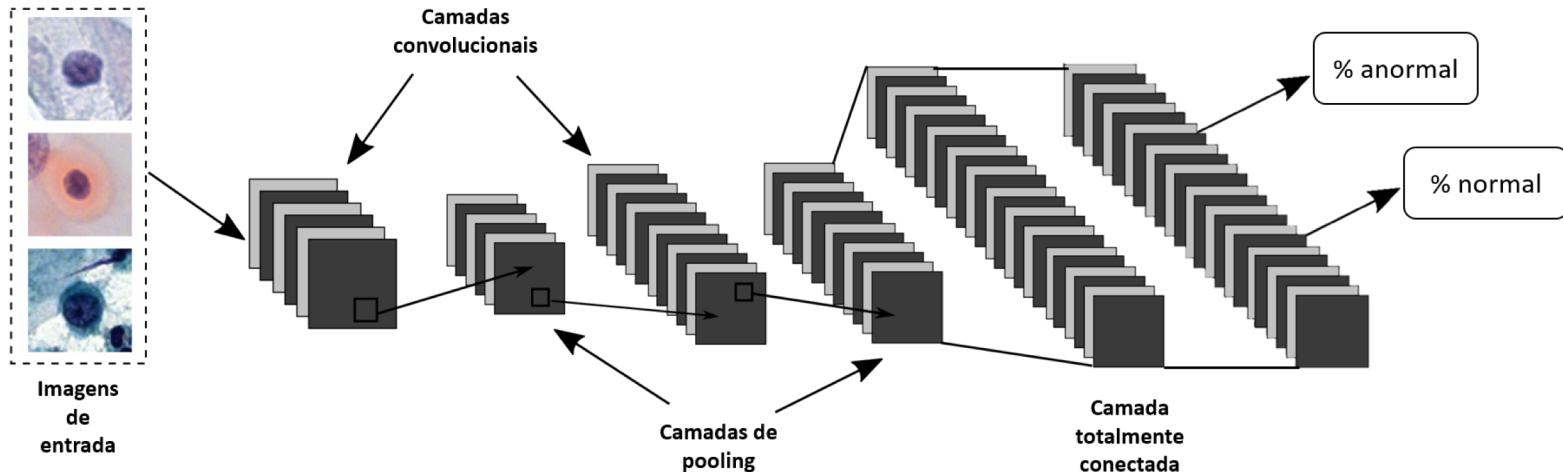
Treinando a CNN

- **Passo 1:** todos os filtros e pesos da rede são inicializados de forma aleatória;
- **Passo 2:** a rede recebe uma imagem de treino como entrada e realiza o processo de propagação, com isso são obtidos os valores de probabilidade da imagem pertencer a cada classe;



Treinando a CNN

- **Passo 3:** é calculado o erro total obtido na camada de saída;
- **Passo 4:** o algoritmo do *backpropagation* é utilizado para calcular os valores do gradiente do erro, em seguida os valores dos filtros e pesos são ajustados;



Treinando a CNN

- **Passo 5:** os passos 2-4 são repetidos para todas as imagens do conjunto de treinamento;
- Devido ao ajuste realizado no passo 4, o erro obtido pela rede é menor a cada vez que uma mesma imagem passa pela rede. Essa redução no erro significa que a rede está aprendendo a classificar corretamente as imagens do treinamento;
- Caso o conjunto de treinamento seja abundante e variado o suficiente, a rede apresentará capacidade de generalização e conseguirá classificar corretamente novas imagens que não estavam presentes no processo de treinamento.

Outras Arquiteturas

ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*)

- O ILSVRC 2012 ficou marcado como o primeiro ano no qual uma CNN atingiu o primeiro lugar desse desafio.

Ano	Descrição	Erro
2010	SIFT + LBP + Fisher Vector + PCA + SVM	28.2
2011	Otimização do método de 2010	25.8
2012	AlexNet	16.4
2013	Zf Net	11.7
2014	GoogLeNet	6.7
2015	ResNet	3.6
2016	DenseNet	3.0
2017	SENets	2.3

Transferência de Aprendizado

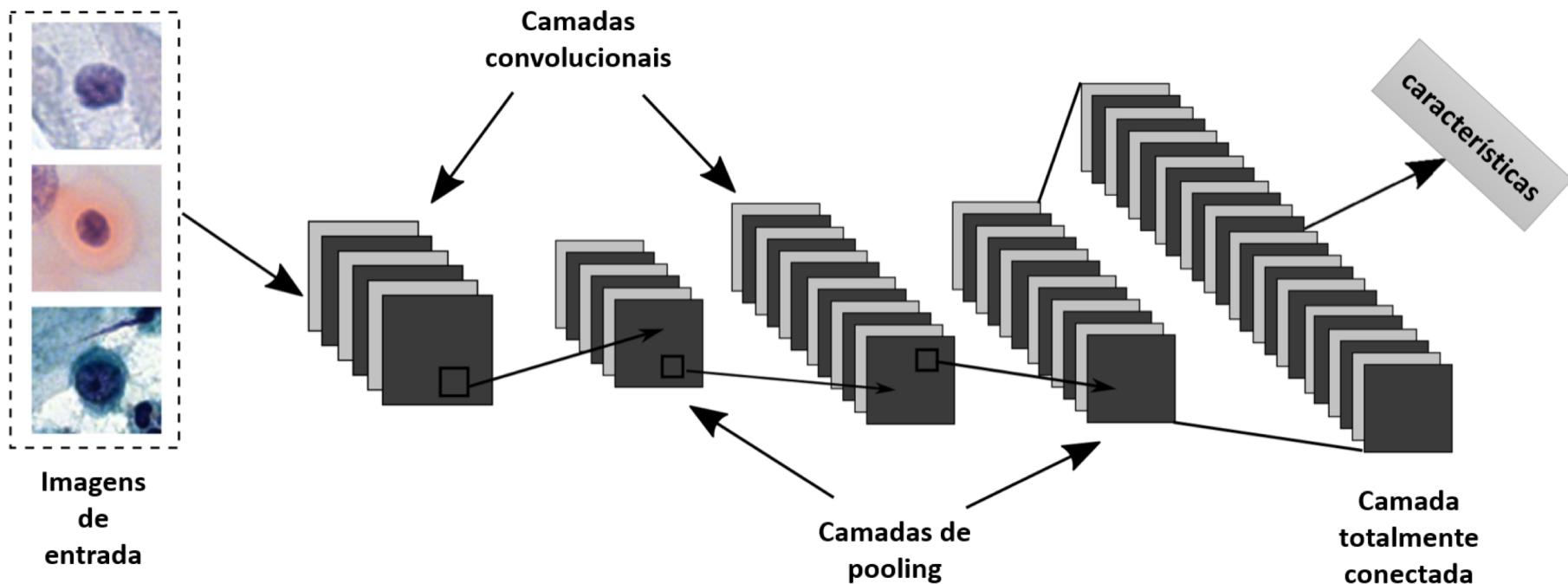
Transferência de Aprendizado

- Na prática, não é comum treinar uma CNN com inicializações aleatórias de pesos, pois para isso seria necessário uma grande quantidade de imagens e algumas semanas de treinamento utilizando múltiplas GPUs;
- Com isso, uma prática comum consiste em utilizar os pesos de uma rede já treinada para uma base muito grande, como a *ImageNet* que possui mais de 1 milhão de imagens e 1000 classes;
- Em seguida, esses pesos podem ser utilizados para inicializar e retreinar uma rede, ou mesmo para a extração de características de imagens.

CNN como Extrator de Características

CNN como Extrator de Características

- Uma forma de utilizar a CNN como extrator de características, é removendo a última camada da rede e utilizar a saída final da nova rede como características que descrevem a imagem de entrada.



CNN como Extrator de Características

- Também podem ser usados como features os valores obtidos após a sequência de camadas convolucionais + *pooling*;
- Os features extraídos das imagens da nova base podem ser utilizadas juntamente com um classificador que requeira menos dados para o treinamento que uma CNN;

Fine-tuning uma CNN

Fine-tuning uma CNN

- A estratégia de *fine-tuning* consiste em dar continuidade ao treinamento de uma rede;
- Os pesos de todas as camadas de uma rede pré-treinada, com exceção da última camada, são utilizados para a inicialização de uma nova CNN;
- É possível fazer o *fine-tuning* de todas as camadas de uma CNN, ou de parte da rede.

CNN com *TensorFlow*

CNN com Keras

- O Keras é uma biblioteca de código aberto para computação numérica e aprendizado de máquina;
- Ela é uma interface mais intuitiva para a utilização de algoritmos de aprendizagem profunda;
- Quase sempre utiliza a biblioteca *TensorFlow* como background;

O *TensorFlow* foi disponibilizada pelo *Google Brain Team* em novembro de 2015 e atualmente já é uma das bibliotecas mais utilizadas para *deep learning*;
- Também possui suporte para CNTK e Theano;
- Possui interface para execução em CPU e GPU;
- O processo de instalação e integração com o *Python* é bem simples e todos esses passos são explicados no site oficial do Keras: <https://keras.io>.

Resumo

CNNs podem ser utilizadas basicamente de 3 formas:

- Treinadas com os dados do problema em questão;
- Treinadas com dados de outro problema (transferência de aprendizado – *transfer learning*);
- Parcialmente treinadas, utilizando como ponto de partida uma rede previamente treinada com dados de outro problema (ajuste fino – *fine-tuning*).