

Redes Neurais Convolucionais

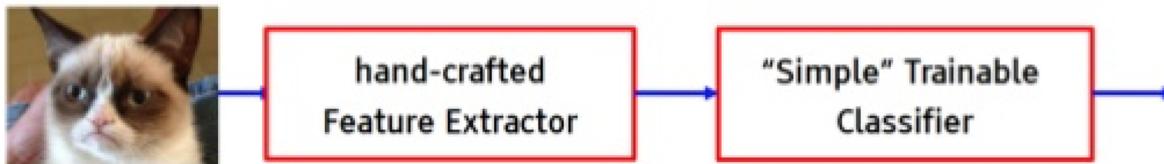
Flávio Araújo

flavio86@ufpi.edu.br

Métodos Tradicional X Deep Learning

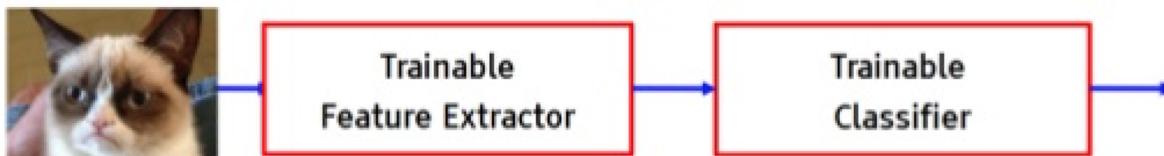
TRADITIONAL APPROACH

The traditional approach uses fixed feature extractors.



DEEP LEARNING APPROACH

Deep Learning approach uses trainable feature extractors.



O que é *Deep Learning*?

- 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.



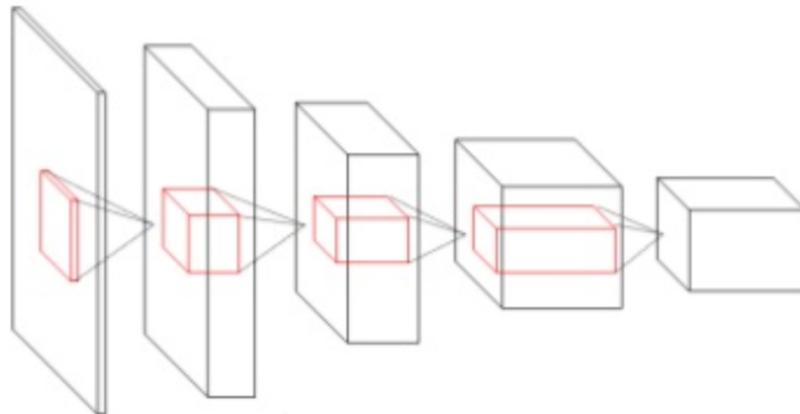
ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*)

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

Erro humano:
5.1%

O que é *Deep Learning*?

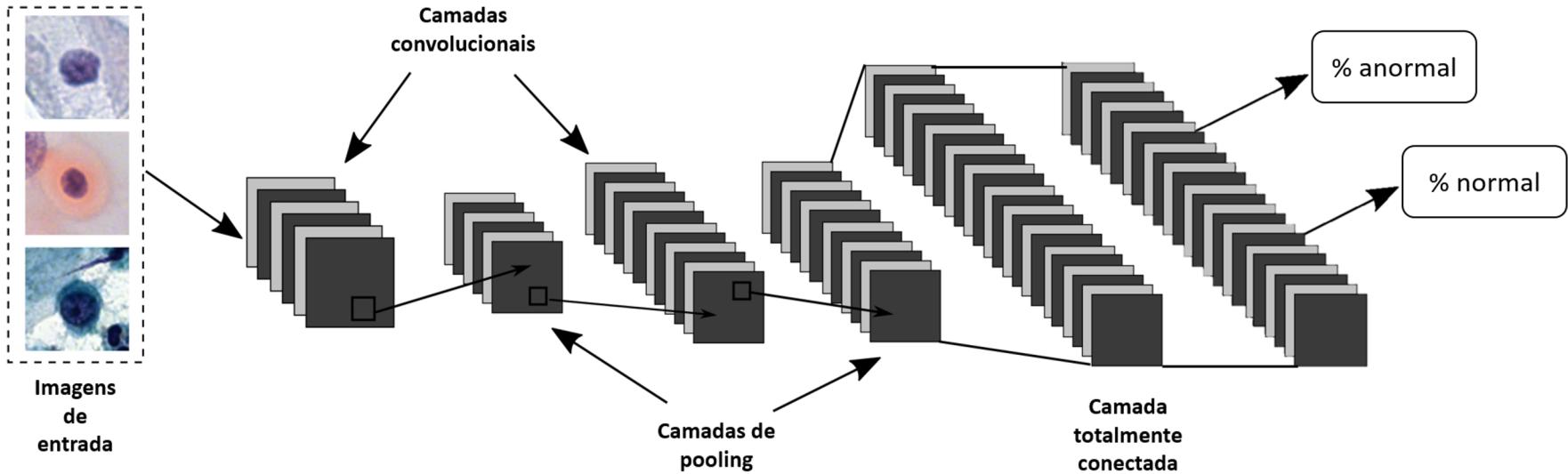
- Múltiplas definições, porém todas possuem em comum:
 - **Múltiplas camadas** de unidades de processamento;
 - As camadas formam uma hierarquia de features *low-level* para *high-level*.



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:

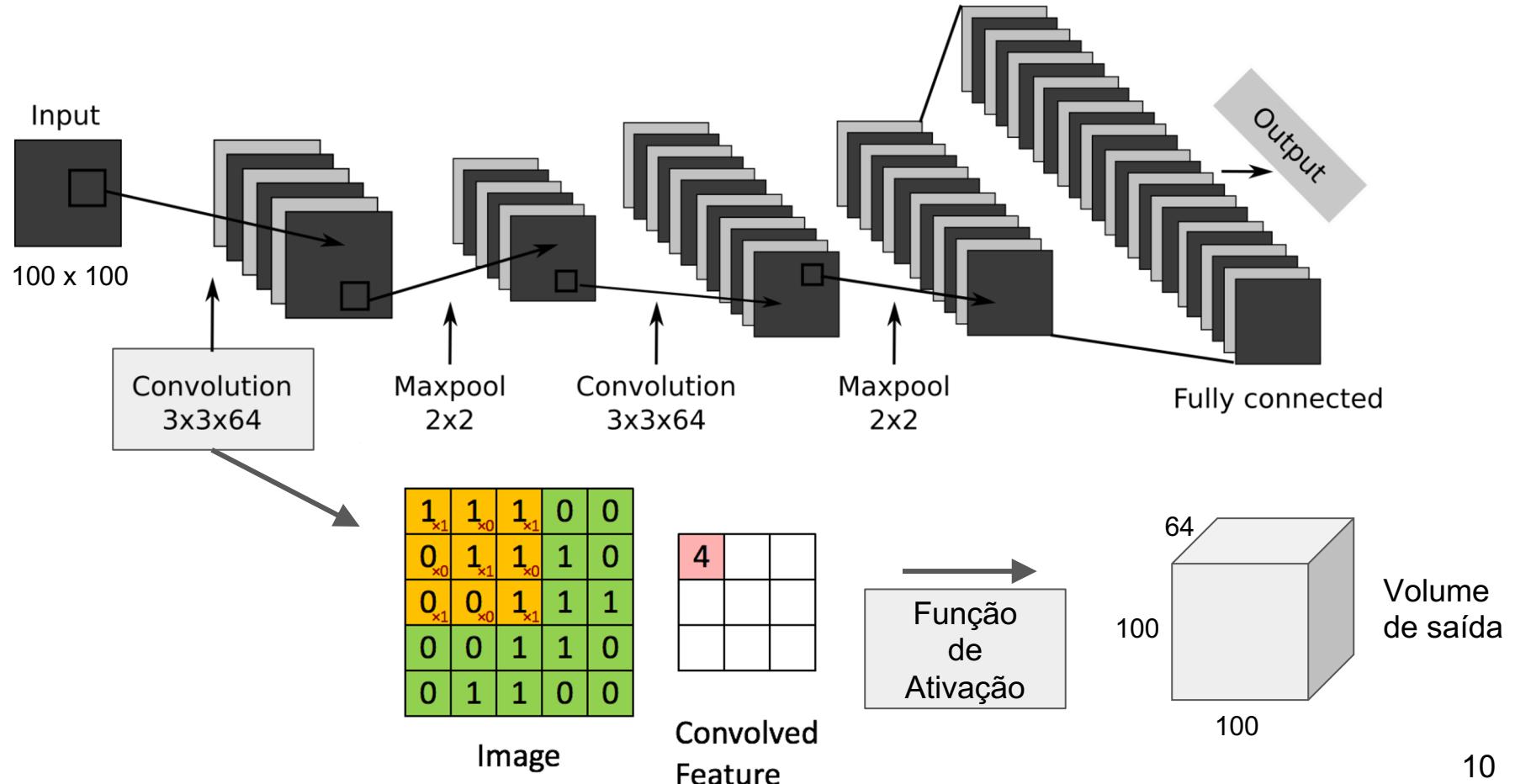


Camada Convolucional

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

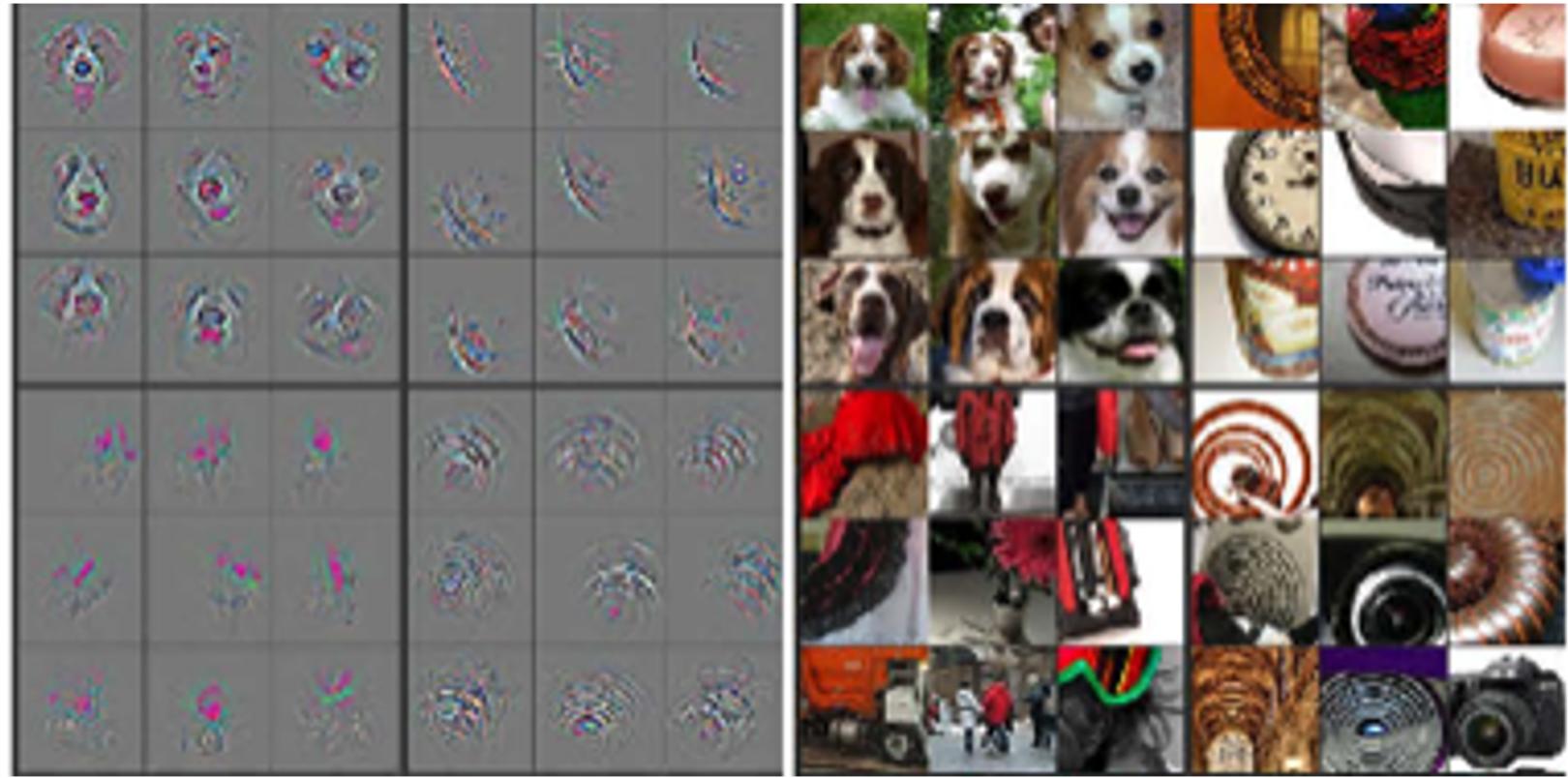


Camada Convolucional

- 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:

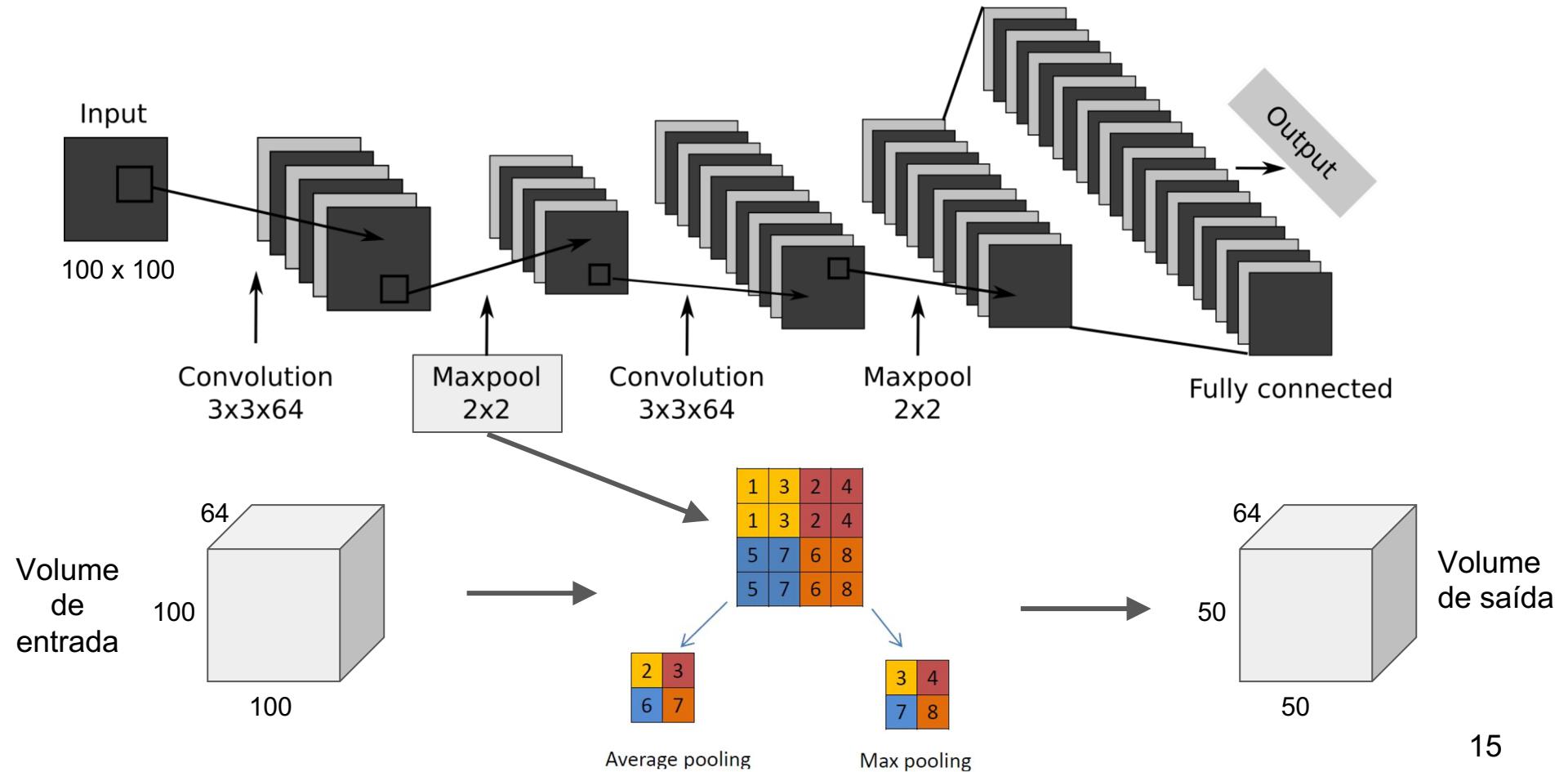


Camada de *Pooling*

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*

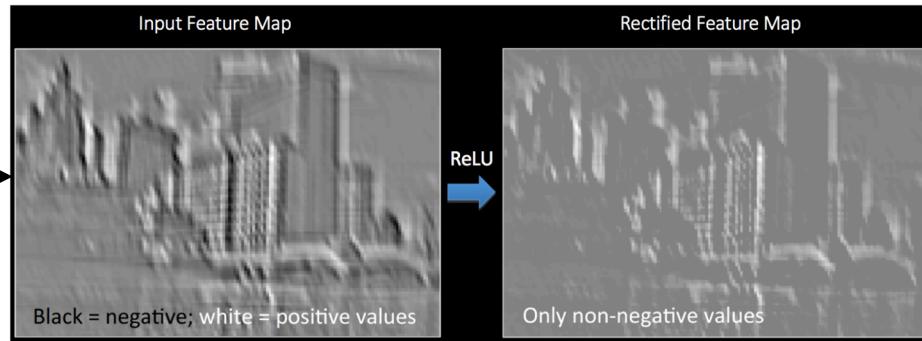


Decomposição da imagem

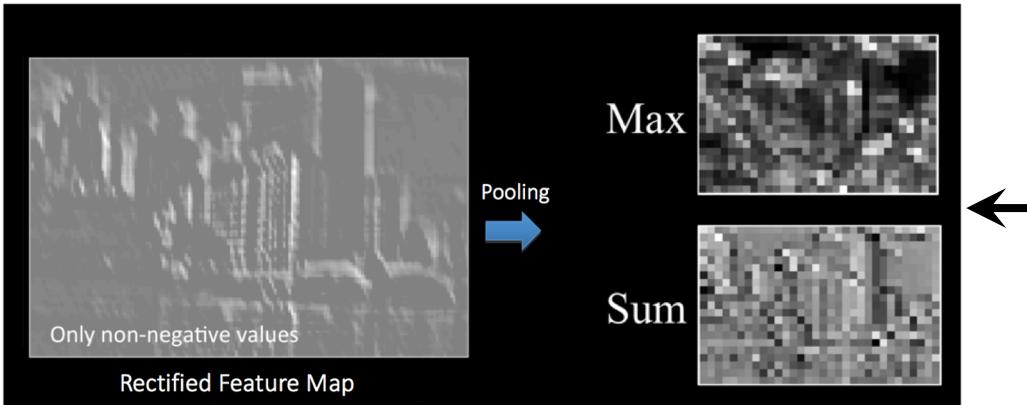
Convolução



Função de ativação



Pooling

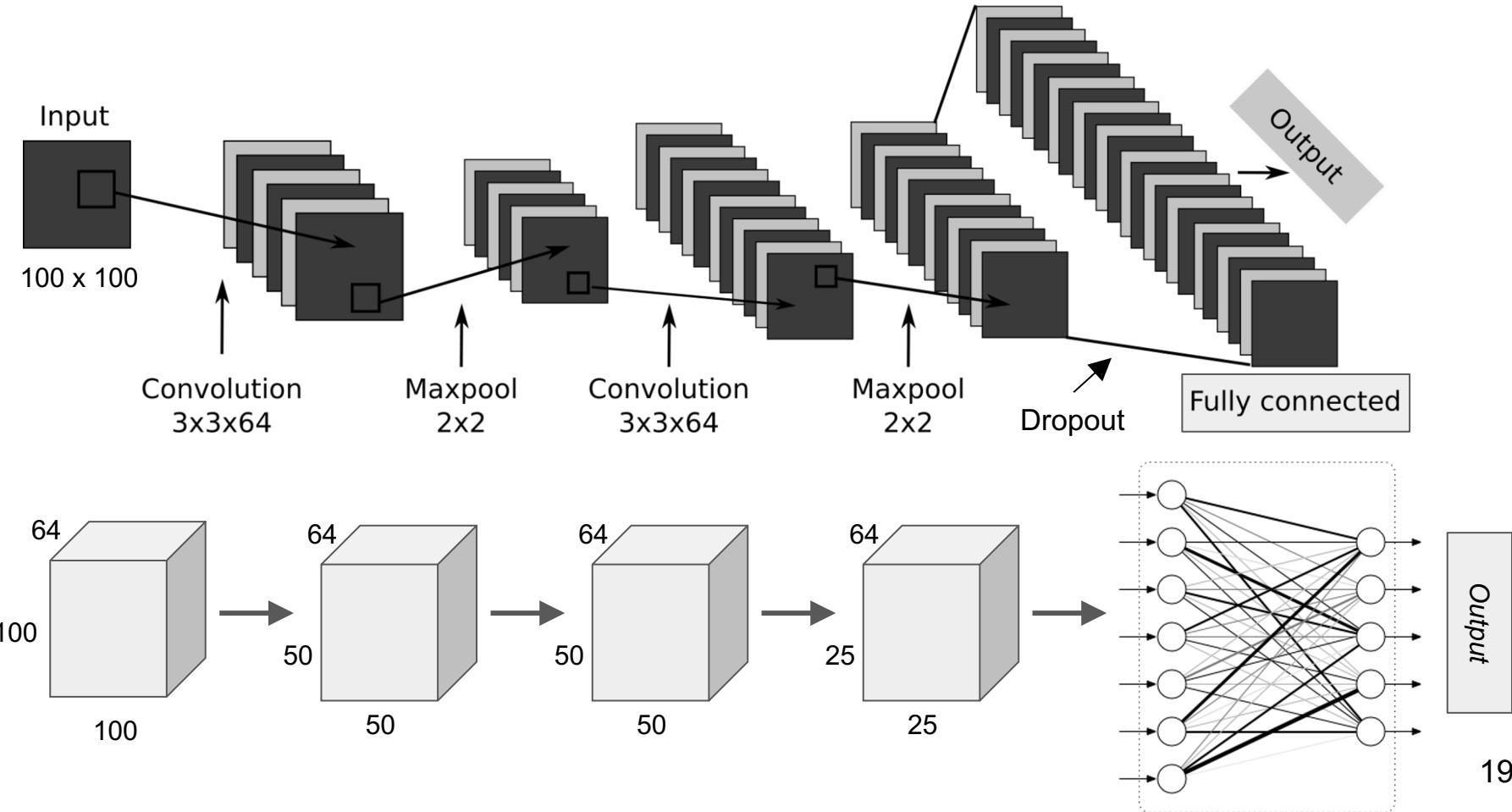


Camada Totalmente Conectada

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.

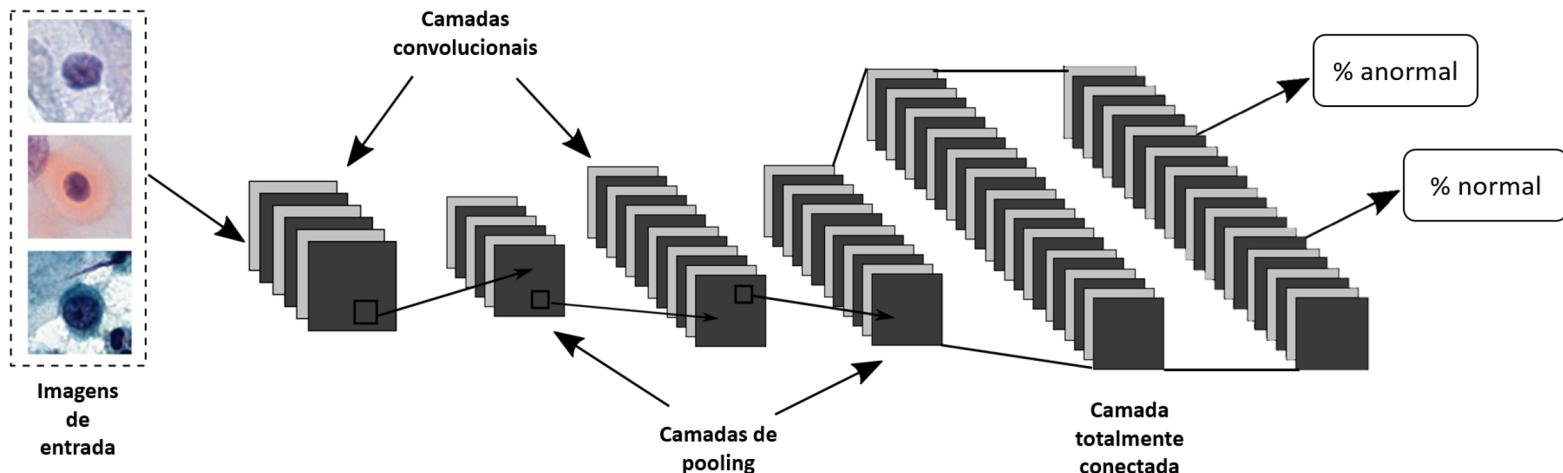
Camada Totalmente Conectada



Treinando a CNN

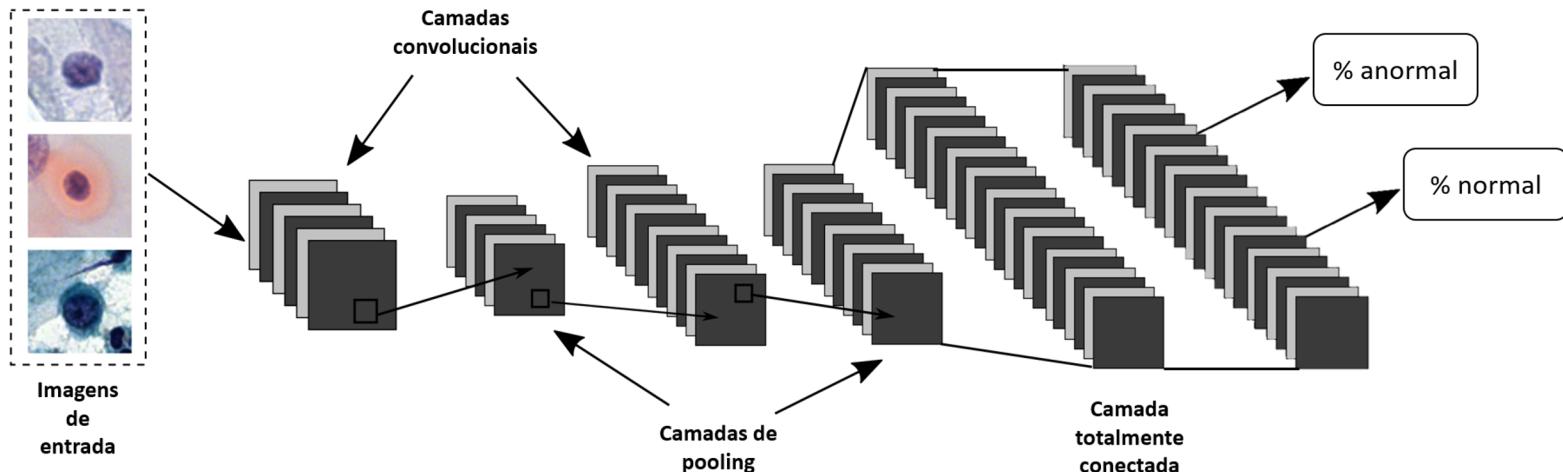
Treinando a CNN

- **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 na proporção que eles contribuíram no erro total;



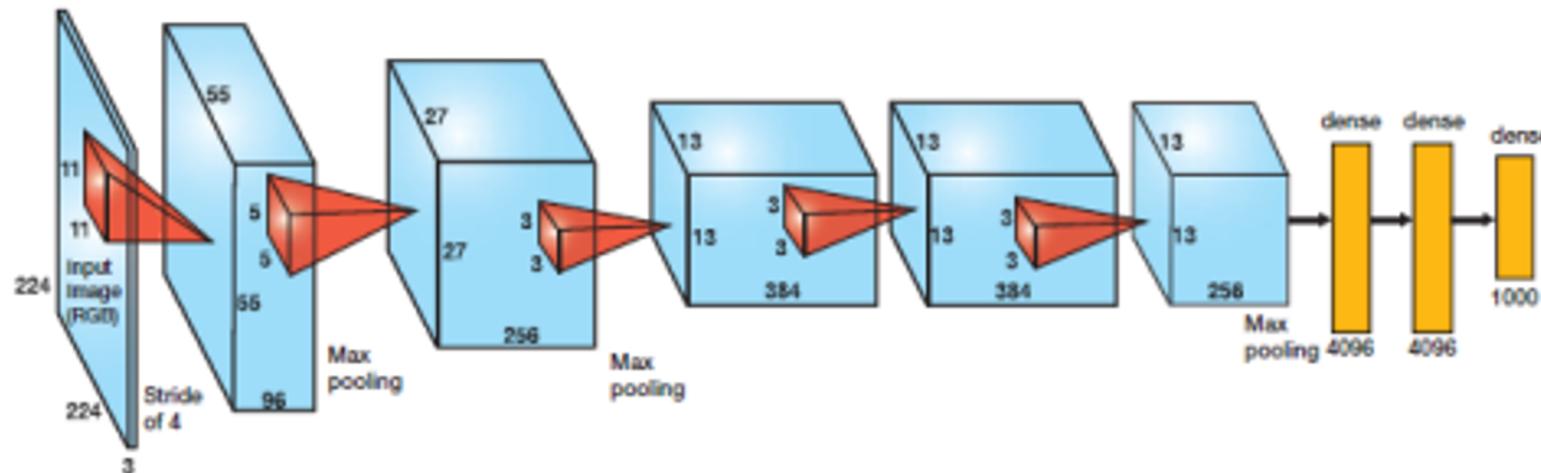
Treinando a CNN

- **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

AlexNet (2012)

- Em 2012, Krizhevsky e colaboradores projetaram a *AlexNet*, que consistia numa versão mais profunda da *LeNet*;
- Essa rede foi utilizada para classificar imagens em 1000 possíveis categorias e foi treinada por cerca de uma semana utilizando duas GPUs.



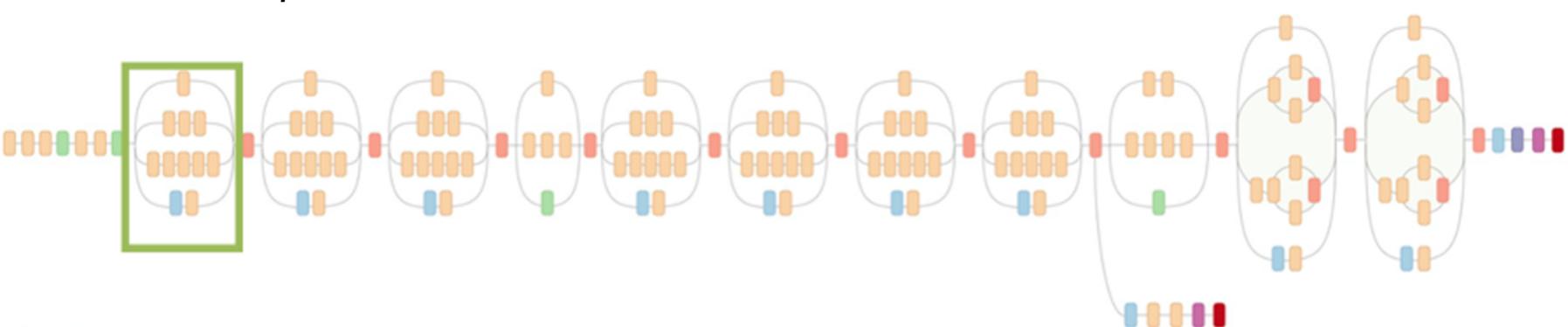
VGG (2014)

- A grande contribuição da VGG foi a ideia de que múltiplas convoluções 3x3 em sequência podiam substituir efeitos de filtros de máscaras maiores (5x5 e 7x7), e que resultavam em maior custo computacional;
- Foram testadas 6 arquiteturas e a que apresentou melhor desempenho foi a marcada pelo retângulo amarelo.

Entrada (imagem RGB - 224x224)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
softmax					

GoogLeNet (2014)

- Diversas empresas, como a *Google*, estavam interessadas em diminuir a complexidade e melhorar a eficiência das arquiteturas existentes;
- Com isso eles criaram módulos que eram executados em paralelo, conhecido como *Inception*.

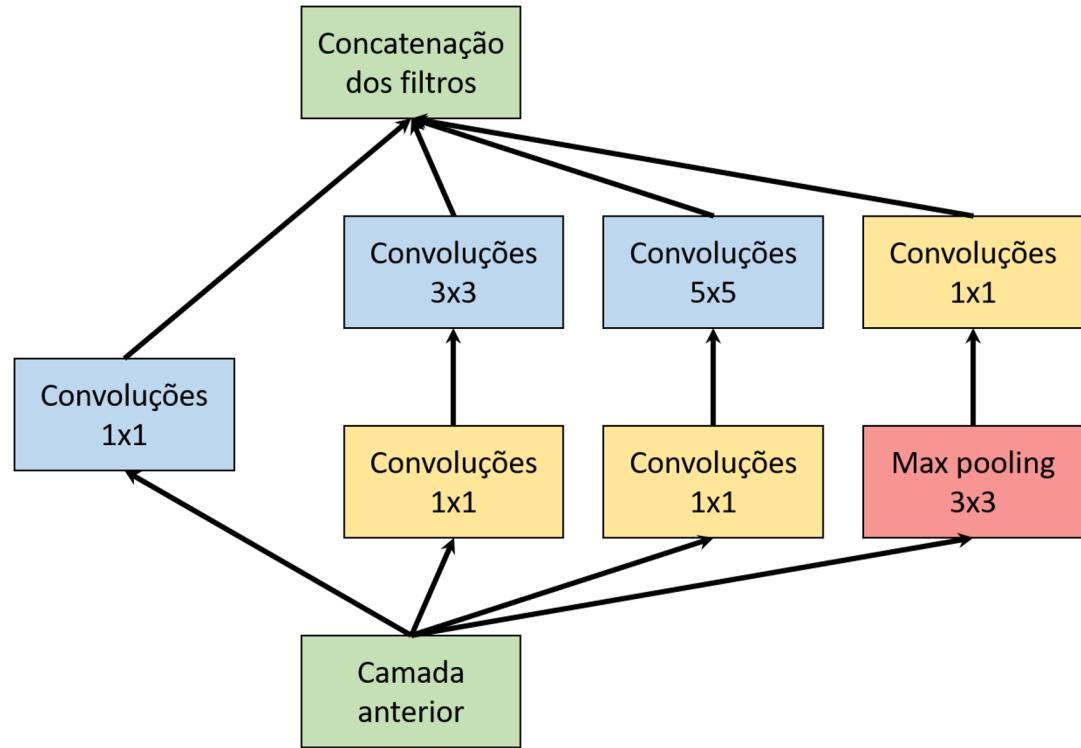


Orange square	Convolução
Blue square	AvgPool
Green square	MaxPool
Red square	Concatenação
Purple square	Dropout
Pink square	Totalmente conectada
Black square	Softmax

A caixa verde indica um bloco executado em paralelo

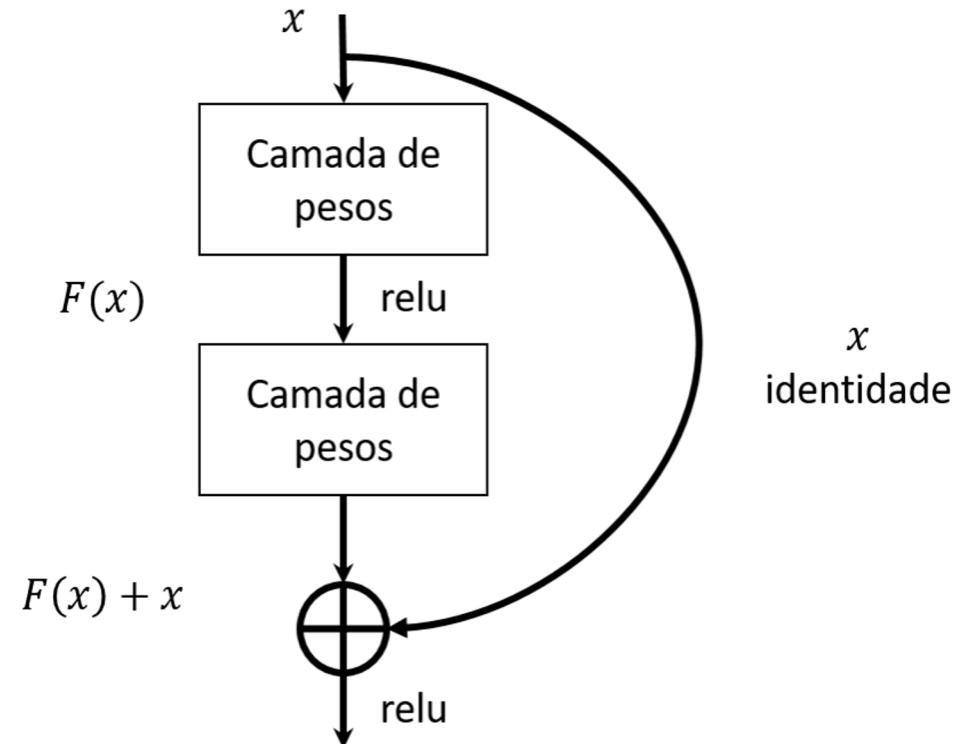
GoogLeNet (2014)

- A grande vantagem do módulo *Inception* era o uso da convolução com filtros 1×1 para reduzir o número de características no bloco paralelo antes de realizar as convoluções com os filtros maiores;
- Esse modelo mostrou que as camadas das CNNs não precisavam ser executadas sequencialmente e que uma estrutura criativa pode melhorar o desempenho da rede e diminuir o custo computacional.



ResNet (2015)

- A ResNet era composta por 152 camadas e formada por blocos residuais;
- Nesses blocos uma entrada x passa por uma série de operações de convolução-relu-convolução, em seguida, o resultado da operação $F(x)$ é adicionado à entrada original x ;
- Na ResNet, o espaço de saída é somente uma alteração do espaço de entrada, pois a função $F(x)$ funciona somente como um termo de regularização. Os criadores da ResNet acreditam que isso facilita a otimização da rede.



Transferência de Aprendizado

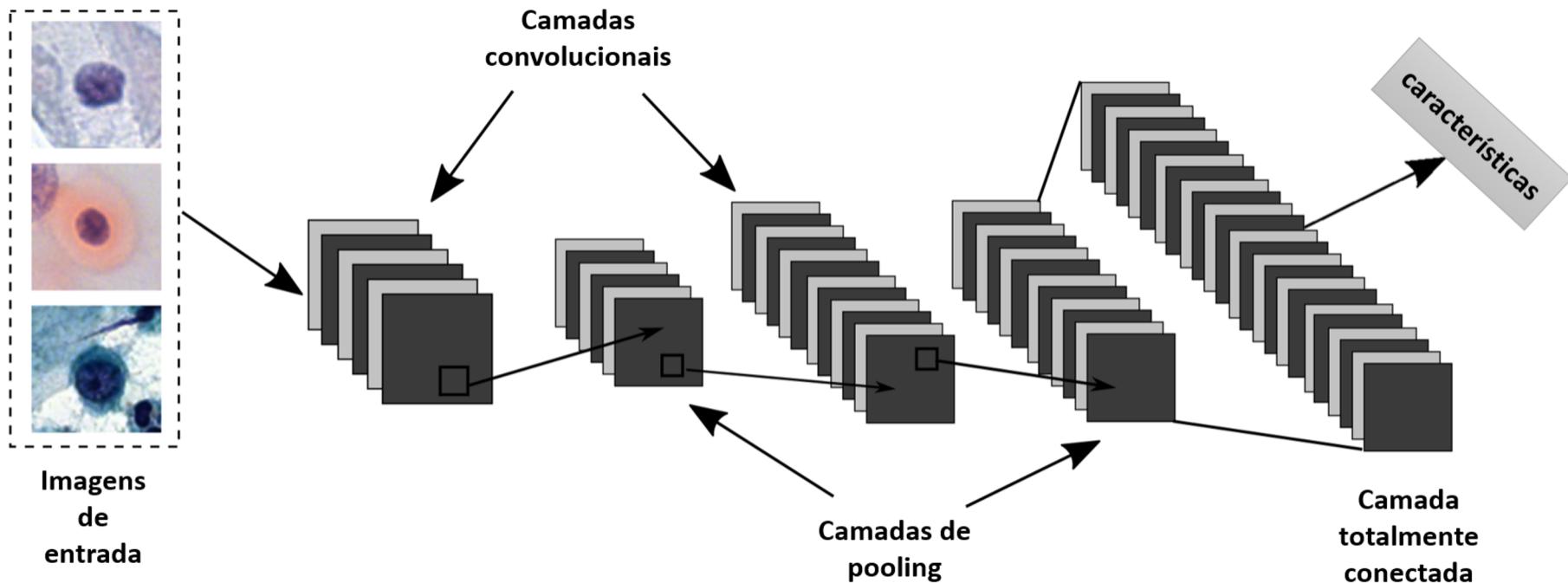
Transferência de Aprendizado

- 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, em outras palavras, 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 somente das últimas camadas. Isso é motivado pelo fato que as primeiras camadas da rede contém extratores mais genéricos que podem ser utilizados para diferentes tarefas, como detectores de bordas e de cores, porém, as camadas mais profundas possuem detalhes específicos da base com a qual a rede foi originalmente treinada.

Como Escolher a Melhor Técnica de Transferência de Aprendizado

Como Escolher a Melhor Técnica de Transferência de Aprendizado

- Dois fatores influenciam na escolha da melhor técnica de transferência de aprendizado utilizar:
 - Tamanho da base;
 - Similaridade com a base original.
- Nesse sentido, quatro cenários devem ser considerados:
 - Nova base de imagens é pequena e similar a base original;
 - Nova base de imagens é grande e similar a base original;
 - Nova base de imagens é pequena e muito diferente da base original;
 - Nova base de imagens é grande e muito diferente da base original.

Como Escolher a Melhor Técnica de Transferência de Aprendizado

- Nova base de imagens é **pequena e similar** a base original:
 - Sempre que a nova base de imagens é pequena não é uma boa ideia aplicar o fine-tuning à CNN, pois isso pode causar problemas de *overfitting*;
 - Nesse caso, como a nova base é similar à base original, a melhor opção é remover a **última camada** da CNN pré-treinada e utilizá-la para **extrair as características** da nova base e treinar um outro classificador linear.
- Nova base de imagens é **pequena e muito diferente** da base original:
 - Como as bases são muito diferentes, a melhor opção é utilizar apenas as primeiras camadas da rede para a extração das características, pois as últimas camadas possuem informações específicas da base original.

Como Escolher a Melhor Técnica de Transferência de Aprendizado

- Nova base de imagens é **grande e similar** a base original:
 - Como as bases de imagens são similares, é bem provável que a utilização da CNN pré-treinada para a extração de características produza bons resultados. No entanto, como a nova base de imagens é grande, é improvável que ocorram problemas de *overfitting*, portanto, provavelmente o ***fine-tuning* de toda a CNN** produzirá melhores resultados.
- Nova base de imagens é **grande e muito diferente** da base original:
 - Neste caso a melhor opção é o ***fine-tuning* de toda a CNN**. Embora as bases de imagens sejam bem diferentes, é bem provável que o *fine-tuning* da CNN reduza bastante o tempo de treinamento em comparação com a inicialização aleatória dos pesos.