UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
ENGENHARIA DA COMPUTAÇÃO

KELVIN BATISTA DA CUNHA

# Reconhecimento e detecção de logotipos a partir de redes neurais convolucionais profundas

RECIFE
2017

KELVIN BATISTA DA CUNHA

# Reconhecimento e detecção de logotipos a partir de redes neurais convolucionais profundas

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Pernambuco – UFPE, como requisito parcial para obtenção do título de Bacharel em Engenharia da Computação.

Orientadora: Veronica Teichrieb
Coorientador: Francisco Simões

RECIFE
2017

# Resumo

Recentemente um grande esforço é empregado no desenvolvimento de novos sistemas para reconhecimento de logotipos. O rápido desenvolvimento de novas aplicações web e o grande fluxo de informações existente atualmente torna o reconhecimento de logotipo um problema chave para aplicações industriais.

Com o avanço do Big Data, um grande volume de dados é armazenado diariamente. No entanto, nem toda informação disponível é adequadamente rotulada, e a grande variedade e complexidade de informação torna desafiadora a classificação destes dados para humanos.

A maioria das técnicas e estudos atuais não é capaz de manipular essa grande quantidade de informação de forma eficiente, por serem treinadas em bases de dados de baixa escala ou por não conter características suficientemente descritivas para o reconhecimento de logotipos. Por este motivo, é necessário um sistema capaz de analisar e classificar os dados em larga escala de forma rápida e precisa.

O desenvolvimento de técnicas de redes neurais profundas convolucionais (*Deep Convolutional Neural Networks*, DCNN) vem obtendo bastante sucesso para aplicação na resolução de problemas de reconhecimento e detecção em bases de dados de larga escala. Sistemas pré-treinados em bases de larga escala com informações genéricas podem ser adaptados para resolução de problemas específicos de forma eficiente, como por exemplo, o uso destes sistemas para reconhecimento de logotipos.

Neste trabalho aplicamos arquiteturas DCNN para construção de um sistema para reconhecimento de logotipos, aplicando o sistema em uma tarefa de reconhecimento de logotipos em larga escala.

Foi proposta uma arquitetura baseada nos principais trabalhos existentes da literatura, com o objetivo de replicar os resultados alcançados com o uso de DCNNs.

Analisamos o potencial do sistema para aplicação em problemas de marketing e análise de mercado, comparando o desempenho com uma aplicação disponível para uso comercial.

É avaliado também o impacto da escolha dos conjuntos de parâmetros e métodos de treinamento no desempenho final da arquitetura.

Obtemos uma taxa de acurácia de 97% para testes de classificação realizados em imagens do Instagram e uma taxa de 82% de acertos na localização dos logotipos nestas imagens.

# *Abstract*

*Recently, a great effort is employed at the development of new systems for logo recognition. Quick development of new web applications and large scale information makes logo recognition a key problem for industrial applications.*

*With the advancement of Big Data technologies, a large amount of data is stored daily but not all available information is adequately labeled. A wide variety and complexity of information makes it challenging the analysis of these data by humans.*

*Most of the current techniques and studies are not capable of manipulating this large amount of information efficiently, because they are trained in small scale databases or because they are not able to extract specific characteristics for the recognition of logos. For this reason there is a need for the development of new systems to analyze and classify data on a large scale database quickly and accurately.*

*The development of Deep Convolutional Neural Networks (DCNNs) has been quite successful for application in the resolution of recognition and detection problems in large scale databases. Pre trained systems on these databases with generic information can be adapted to solve specific problems efficiently as logo recognition systems.*

*In this work we apply DCNNs architectures to build a system for the recognition of logos, applying the system in a large scale logo recognition task.*

*An analysis based on the main works of the literature was performed, in order to replicate the results obtained with the use of DCNNs.*

*We analyze the potential of the system for application in marketing problems and market analysis, comparing performance with an application available for commercial use.*

*It is also evaluated the impact of the choice of training parameters and training methods in the final performance of the architecture developed.*

*We get an accuracy rate of 97% for classification tests on Instagram images and an 82% hit rate on the localization of the logos in these images.*

# Sumário

# Lista de figuras

# Lista de tabelas

# Lista de siglas

**ANN** – *Artificial Neural Network*
**API** – *Application Programming Interface*
**ART** – *Adaptive Ressonance Theory*
**CNN** – *Convolutional Neural Network*
**CPU** – *Central Processing Unit*
**CRT** – *Complete Rank Transform*
**DCNN** – *Deep Convolutional Neural Network*
**DNN** – *Deep Neural Network*
**DRCN** – *Deep Logo Detection Region-Based Convolutional Networks*
**GDS** – Gradiente Descendente Estocástico
**GPU** – *Graphics Processing Unit*
**ILSVRC** – *ImageNet Large Scale Visual Recognition Competition*
**MLP** – *Multilayer Perceptron*
**MSE** – *Mean Squared Error*
**M-SIFT** – *Multi-Spectral Scale Invariant Features Transform*
**PNN** – *Probabilistic Neural Networks*
**RANSAC** – *Random Sample Consensus*
**RBF** – *Radial Basis Function*
**RCNN –** *Rapid Convolutional Neural Networks*
**ReLU** – *Rectified Linear Unit*
**RMS** – *Root Mean Square*
**SIFT** – *Scale Invariant Feature Transform*
**SOM** – *Self Organizing Map*
**SS –** *Selective Search*
**VPN** – Valor Preditivo Negativo
**VPP** – Valor Preditivo Positivo

# 01. Introdução

Reconhecimento e detecção de logotipos é uma classe de problemas que vem sendo extensivamente estudada no campo de visão computacional, com diversas aplicações e métodos propostos na literatura (D. Doermann et al., 1993; E. Francesconi et al., 1998; G. Zhu; D. Doermann, 2007; R. Boia et al., 2014). Este tipo de técnica tem como objetivo reconhecer as características de uma marca contidas numa imagem de entrada, como estilos de fonte, ilustrações, objetos característicos, entre outros. Entre suas aplicações é possível destacar o auxílio à proteção de propriedade intelectual, o reconhecimento de logotipos em veículos para transportes e o gerenciamento de logotipos em redes sociais.

Atualmente, a maioria dos estudos e técnicas para reconhecimento de logotipos são baseadas em bases de dados de pequena escala (bases com poucos exemplos rotulados). Tais abordagens usualmente não são abrangentes o suficiente para obtenção de bons resultados para treinamento e classificação (S. Hoi et al., 2015). Neste cenário, uma das soluções possíveis é a construção de uma base de dados de larga escala. Construir uma base de dados suficientemente grande para um problema específico requer muito esforço manual, sendo um processo bastante lento e propenso a diversos erros durante sua construção.

Nos últimos anos, Redes Neurais Convolucionais (*Convolutional Neural Networks*, CNNs) e Redes de Aprendizagem Profunda (*Deep Neural Networks*, DNNs) tiveram grande evolução em pesquisas com o desenvolvimento de novos modelos para resolução de problemas de classificação e detecção de objetos em imagens (DNNs que utilizam estruturas CNNs são normalmente chamadas de *Deep Convolutional Neural Networks* (DCNNs)). Este avanço é dado pelo rápido desenvolvimento de novos recursos computacionais capazes de suportar tais modelos, como novos componentes de *hardware* e a obtenção de grande quantidade de informação genérica em larga escala (utilizando a *web*, por exemplo). Além disto, vários conceitos e algoritmos vêm sendo propostos na literatura, o que também é essencial para melhorar a capacidade e desempenho destas arquiteturas (Y. LeCun et al., 1989).

Redes Neurais Convolucionais são usadas atualmente para resolução de diversos tipos de problemas, com grande foco na aplicação em reconhecimento de imagens, como por exemplo, reconhecimento de caracteres manuscritos (Y. LeCun, 1998), classificação e detecção de várias classes em imagens (A. Krizhevsky et al., 2012; C. Szegedy et al., 2015) e classificação de sinais e pedestres (P. Sermanet; Y. LeCun, 2011).O desempenho das arquiteturas baseadas em DCNNs pode ser limitado quando treinado para classificação em bases de dados de pequena escala. Contudo, o desenvolvimento de novas bases de dados de larga escala vem contribuindo para um avanço significativo na geração de novos modelos de alta capacidade de classificação e detecção em imagens (J. Deng et al., 2009).

Uma das principais vantagens na utilização de DCNNs é a possibilidade de treinamento fim-a-fim, o que elimina a necessidade de um grande esforço na criação de projetos manuais para algoritmos de extração de características. Em contrapartida, a necessidade de um grande número de amostras rotuladas e a grande quantidade de parâmetros necessários para treinamento em arquiteturas profundas pode limitar o uso das DCNNs em diversas aplicações de reconhecimento (S. Hoi et al., 2015).

A criação do *ImageNet database* (J. Deng et al., 2009) é um dos fatores que vem tornando o uso de DCNNs mais acessível na construção de modelos e aplicações. Esta base de dados contém cerca de 15 bilhões de imagens rotuladas em

1000 categorias de classes distintas. Acredita-se que com o esforço para produção desta base de dados de larga escala, o desenvolvimento de sistemas cada vez mais robustos seja acelerado. Anualmente é proposto um desafio de reconhecimento de objetos em geral (*ImageNet Large Scale Visual Recognition Competition*, ILSVRC). O desafio possibilita aos pesquisadores obter uma fonte de resultados para os melhores modelos de reconhecimento de objetos, onde os modelos e arquiteturas propostas podem ser testados e validados igualmente.

Nos últimos anos (2010-2015), arquiteturas DCNNs (A. Krizhevsky et al., 2012; C. Szegedy et al., 2015; K. Simonyan; A. Zisserman, 2014) vêm dominando a categoria de reconhecimento genérico de objetos. Utilizando a base *ImageNet*, estas arquiteturas podem empregar uma abordagem de construção em grande escala, com milhões de parâmetros e um grande número de camadas.

As arquiteturas utilizadas na competição contêm um alto grau de generalização, utilizando diferentes representações em sua computação para classificar um objeto. Estas arquiteturas treinadas podem ser reutilizadas para a resolução de um problema de classificação mais especifico. A reutilização destas arquiteturas poupa um grande esforço de treinamento, diminuindo a quantidade de parâmetros necessários a serem definidos, e requer menos tempo e recursos computacionais. Foi mostrado por Landola e colegas (F. Landola et al., 2015) que adaptar uma arquitetura genérica treinada numa base de dados de grande escala, para resolver um problema mais restrito de domínio específico, mantém o desempenho da arquitetura praticamente inalterado.

Esta abordagem de reuso vem sendo bastante utilizada atualmente, como por exemplo em aplicações para reconhecimento de faces (R. Stewart; M. Andriluka, 2015), reconhecimento de pedestres (W. Ouyang; X. Wang, 2013), reconhecimento de tráfego e sinais de trânsito (P. Sermanet; Y. LeCun, 2011) e detecção e classificação de logotipos (F. Landola et al., 2015; S. Hoi et al., 2015).

## 1.1 Objetivo

Neste contexto, o objetivo deste trabalho é construir e/ou aplicar um modelo baseado em DCNNs para reconhecimento de logotipos. O modelo deve ser projetado para resolver os problemas de classificação e localização de logotipo em imagens, com potencial para aplicação em problemas de *marketing* e análise de mercado.

Para desenvolvimento do modelo utilizamos a ideia de pós-treino em redes neurais profundas como apresentado em (F. Landola et al., 2015; R. Stewart; M. Andriluka, 2015). Durante o treinamento do modelo a base de dados de larga escala ImageNet será utilizada para o pré-treinamento e geração dos filtros de extração de características da rede DCNN, e uma base de dados com menos exemplos contendo dados específicos para o problema de reconhecimento de logotipos para especializar a rede na resolução do problema proposto. A segunda base foi construída durante o desenvolvimento deste trabalho, a necessidade de construção da base de dados específica se deve à falta de bases de dados que contenham uma grande quantidade de exemplos específicos para logotipos. Para validação do nosso modelo analisaremos um caso base do problema de reconhecimento de logotipos, analisando a presença apenas do logotipo da Coca-Cola.

Durante a validação do trabalho serão avaliados a taxa de reconhecimento do sistema proposto, comparando os resultados obtidos com os resultados gerados por uma aplicação comercial disponível para testes. Além disto, também será avaliado o

desempenho do modelo para diferentes parâmetros de treino, avaliando a influência dos resultados que o modelo alcança a partir da escolha dos algoritmos de aprendizagem, taxas de aprendizagem e função de ativação. Também serão avaliados o tempo necessário para o modelo classificar um exemplo e o intervalo de classificação que o modelo é capaz de obter.

Para o desenvolvimento do trabalho foram utilizados os seguintes passos:

1 – Estudo dos métodos e arquiteturas disponíveis de DCNN para reconhecimento de imagens e reconhecimento de logotipos.

2 – Definição da arquitetura que será desenvolvida durante o trabalho.

3 – Escolha da ferramenta de desenvolvimento da arquitetura escolhida.

4 – Construção do modelo.

5 – Construção de uma aplicação para reconhecimento através de uma API externa para comparação dos resultados obtidos pelo modelo desenvolvido.

6 – Construção, seleção e divisão da base de dados.

7 – Escolha dos métodos de treinamento, critérios de correção e parâmetros de treino.

8 – Treinamento do modelo desenvolvido.

9 – Validação dos resultados, comparando o modelo de DCNN construído com a aplicação de reconhecimento de gerada.

## 1.2 Organização do documento

Este trabalho está organizado da seguinte forma: o capítulo 2 introduz os fundamentos básicos utilizados para a realização deste trabalho. São descritos os temas de visão computacional, aprendizagem de máquina, redes neurais artificiais, redes neurais profundas, e o problema de reconhecimento de logotipos. O capítulo 3 mostra um estudo sobre trabalhos relacionados existentes que propõem soluções similares ao presente trabalho. No capítulo 4 é descrita a técnica utilizada e o desenvolvimento do sistema proposto. No capítulo 5 são analisados os resultados obtidos para validação do sistema. O capítulo 6 conclui o trabalho.

# 02. Fundamentos

Este capítulo tem como objetivo realizar uma descrição dos principais conceitos utilizados para a construção deste trabalho. É feita uma revisão sobre a área de visão computacional, com foco no problema de reconhecimento de objetos. São apresentados os princípios básicos de aprendizagem de máquina e redes neurais, e as principais pesquisas que levaram ao desenvolvimento das arquiteturas de Redes Neurais Convolucionais Profundas, ou simplesmente DCNNs. Também é introduzido o problema de reconhecimento de logotipos, que compreende a sua descrição, aplicações e desafios.

## 2.1 Visão computacional

Visão computacional é uma área da computação que busca entender e classificar características de alto nível de elementos presentes em imagens e vídeos digitais, ou seja, é uma área capaz de desenvolver ferramentas para automatizar as tarefas de manipulação de imagens e vídeos, classificação e reconhecimento de objetos, detecção de cenas e ambientes, entre outros. Estas atividades são realizadas de forma eficiente inspirada muitas vezes no sistema visual humano (D. Ballard; C. Brown, 1982).

Tarefas de visão computacional incluem métodos para aquisição, processamento, análise e entendimento de imagens. De forma geral, é realizada a descrição das propriedades dos objetos e do ambiente (formas, iluminações e distribuição de cores) presentes em uma ou mais imagens, a fim de gerar informações numéricas e/ou simbólicas como resposta (L. Shapiro; G. Stockman, 2001; R. Klettle, 2014). Enquanto estas tarefas podem ser realizadas facilmente por pessoas e animais, as técnicas computacionais têm certas dificuldades sendo mais propensas a erros e limitações (R. Szesliski, 2011).

A visão computacional é atualmente usada em uma grande variedade de aplicações na indústria, como por exemplo, reconhecimento automático de caracteres em texto (Figura 2.1a), reconhecimento e detecção de íris em imagens (Figura 2.1b), reconhecimento facial (Figura 2.1c) e reconstrução de modelos 3D (Figura 2.1d).

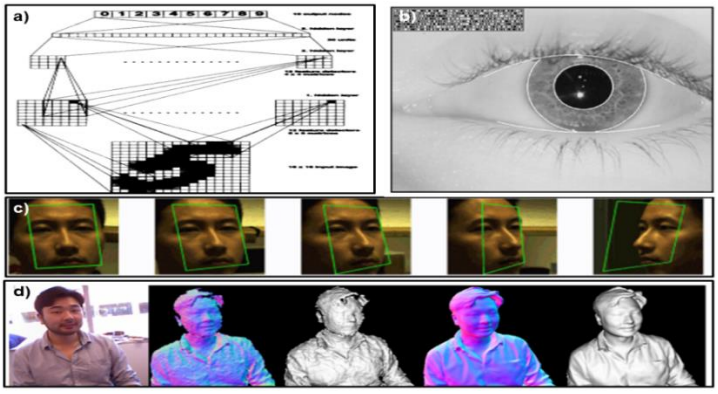


**Figura 2.1 –** Algumas aplicações de visão computacional: a) Reconhecimento automático de caracteres. b) Detecção e reconhecimento de íris. c) Reconhecimento facial. d) Reconstrução de modelos 3D. Fontes: O. Trie et al., 1996; J. Daugman, 2004; A. Wagner et al., 2012; S. Izadi et al., 2011.

Cada uma das áreas de aplicação descritas acima emprega uma variedade de tarefas abordadas pela visão computacional; algumas destas tarefas são descritas abaixo.

**Reconhecimento:** Um dos principais problemas da visão computacional é determinar se existe algum dado, objeto ou atividade presente em uma imagem. Técnicas de reconhecimento são capazes de classificar conjuntos de classes em cenas através de características previamente conhecidas do objeto desejado (S. Belongie et al., 2002; D. Lowe, 1999).

**Análise de movimento:** Com base em uma sequência de imagens é possível estimar o movimento e a velocidade de um objeto com posição conhecida. Este tipo de método pode ser utilizado para rastreamento de objetos em cena, ou determinação do movimento da câmera que produz as imagens em relação ao ambiente observado (J. Aggarwal; Q. Cai, 1997; F. Simões, 2016; W. Wolf, 1996).

**Reconstrução de cenários:** Este tipo de método visa computar um modelo 3D, com base em várias imagens 2D em diferentes posições, de algum cenário ou objeto específico. O modelo pode também ser construído estimando os pontos 3D no ambiente ou até mesmo relacionando pontos obtidos de sensores de profundidade para reconstrução do ambiente (F. Xu; K. Mueller, 2007; A. Davison et al., 2007).

Nas próximas subseções é descrito com mais detalhes o funcionamento das técnicas de reconhecimento de objetos em imagens digitais, foco deste trabalho.

### 2.1.1 Reconhecimento de objetos

Reconhecimento de objetos é uma subárea da visão computacional que está em grande desenvolvimento atualmente. Métodos de reconhecimento de objetos utilizam, em sua vasta maioria, conceitos e técnicas de processamento de imagens, estatística e aprendizagem de máquina (R. Szesliski, 2011).

Reconhecer um objeto está geralmente associado a aprender uma quantidade significativa de características estruturais e relacionais dos padrões (objetos desejados) de forma a descrever e identificar unicamente um objeto ou um conjunto de objetos similares dentro de uma imagem em várias situações e/ou ambientes diferentes (R. Gonzales; R. Woods, 2008) (Figura 2.2).

**Figura 2.2** – Sistema de reconhecimento de objetos baseado em características SIFT. Fonte: D. Lowe, 1999.

Sistemas de reconhecimento de objetos em imagens são geralmente projetados para resolução de três tipos diferentes de problemas: classificação, detecção e localização.

O principal foco no reconhecimento está em aprender as características dos padrões desejados. Estas características são representadas na forma de descritores. Cada padrão a ser reconhecido contém um conjunto de descritores que o definem no espaço observado.

Um padrão pode ser visto como um vetor $\vec{x} = (x_1, x_2, \ldots, x_n)$, onde $n$ é o número total de descritores associado ao padrão pertencente a uma classe $c_i \in C = (c_1, c_2, \ldots, c_M)$, com $M$ sendo o número total de classes conhecidas.

Um dos maiores problemas para aplicações de reconhecimento está em definir qual método deve ser utilizado para a obtenção dos descritores (extração de características) e em como representar, interpretar e armazenar estes dados de forma eficiente no sistema (D. Lowe, 1999).

Idealmente os descritores de cada objeto devem ser invariantes a rotação, escala e translação, ou seja, devem identificar o objeto desejado de forma única independente da sua posição e/ou tamanho dentro da imagem.

Após a obtenção dos descritores é realizada a correspondência entre os descritores obtidos e os descritores já conhecidos armazenados no sistema. As correspondências podem ser realizadas por métodos que consideram a distância mínima de dissimilaridade entre as características observadas ou casamento destas características por correlação (R. Gonzales; R. Woods, 2008).

O padrão observado será atribuído à classe de maior correspondência do conjunto $C$ de acordo com a métrica utilizada, retornando a resposta desejada do sistema. Um grande esforço é aplicado em técnicas para representação de características. Atualmente existem várias formas e algoritmos na literatura para extração de descritores de objetos presentes na imagem. Além disso, há uma grande variedade de técnicas para armazenamento e entendimento destas características.

### 2.1.2 Sistemas de reconhecimento de objetos

Sistemas para reconhecimento de objetos podem ser definidos através de uma sequência de passos necessários para a classificação e/ou localização dos padrões, como visto na Figura 2.3.

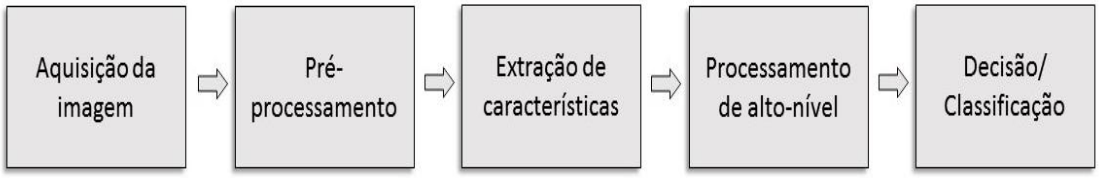


**Figura 2.3** - Passos necessários para construção de um sistema de reconhecimento de objetos. Fonte: o autor.

A fase de **aquisição da imagem** é relacionada com a produção da imagem digital por meio de um sensor, por exemplo, uma câmera. Dependendo do sensor utilizado, a imagem obtida pode ter um conjunto de pontos 2D ou 3D. Os valores dos *pixels* da imagem produzida normalmente são relacionados com a intensidade de luz refletida nos objetos, mas também podem ser quantificados de outra forma relacionada com diferentes medidas físicas, dependente do sensor utilizado.

Na fase de **pré-processamento** são aplicadas transformações na imagem com a finalidade de padronizar e facilitar a forma como a imagem é reconhecida. Nesta fase são realizadas correções do sistema de coordenadas do mundo para o sistema de coordenadas do sensor (Figura 2.4a), projeções do sistema de coordenadas do sensor para os *pixels* da imagem (Figura 2.4b), redução dos ruídos introduzidos durante a aquisição da imagem, melhoria do contraste, etc. (F. Simões, 2016).

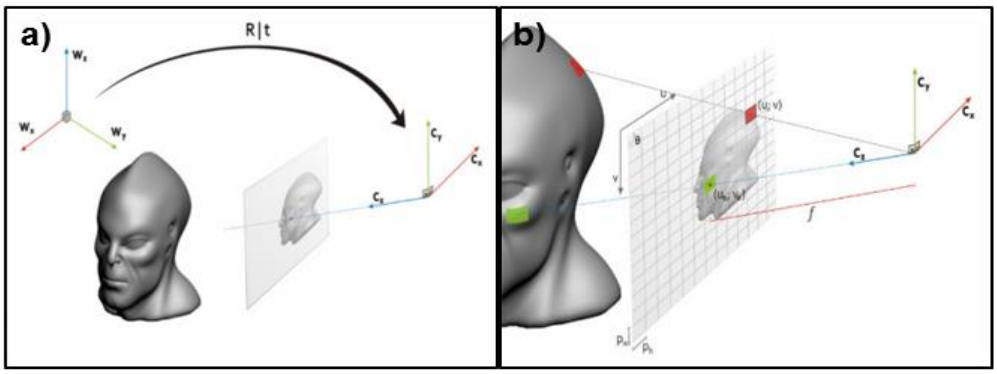


**Figura 2.4** - a) Conversão de coordenadas do mundo para coordenadas da câmera (sensor), realizada na etapa de pré-processamento. b) Projeção da imagem capturada em coordenadas de *pixels* após transformação para coordenadas da câmera. Fonte: Adaptado de F. Simões, 2016.

A obtenção e o armazenamento dos descritores para cada padrão é realizada na fase de **extração de características**. Os descritores podem ser definidos como linhas, arestas, texturas ou formas em geral. Os descritores são obtidos com a construção de filtros que capturam as características desejadas da imagem. Esta etapa está diretamente relacionada com o desempenho final do sistema de reconhecimento. A qualidade das características obtidas e armazenadas é um fator determinante para melhorar a taxa de precisão na classificação dos padrões e aperfeiçoar o tempo gasto para classificar cada exemplo. Por este motivo, um grande esforço é empregado para escolha do sistema de extração de características que melhor se adapte para a resolução do problema desejado.

Os descritores observados são passados para a fase de **processamento de alto-nível,** onde será realizada a verificação dos dados recebidos e a estimação dos parâmetros para realização da verificação de correspondências entre as características conhecidas e as características encontradas na imagem.

Na fase de **decisão/classificação**, se obtém a resposta do sistema. Esta resposta pode ser dada como a posição na imagem de um objeto ou um conjunto de objetos desejados (localização) ou uma resposta informando se a imagem contém um objeto de interesse (detecção). Geralmente algoritmos de aprendizagem de máquina são usados nesta etapa para compreender os dados observados e retornar uma decisão baseado no que se conhece previamente do problema.

## 2.2 Aprendizagem de máquina

Aprendizagem de máquina é um subcampo da ciência da computação onde o foco é a geração de algoritmos capazes de aprender e fazer predições sobre dados a fim de tomar decisões automaticamente. Inicialmente foi definido por Arthur Samuel em 1959 como “o campo de estudo que dá ao computador a habilidade de aprender sem ser explicitamente programado” (A. Samuel, 1959).

Uma definição mais formal foi fornecida posteriormente por Tom M. Mitchell e descreve aprendizagem de máquina da seguinte forma: “Um programa de computador é dito aprender com a experiência E com relação a alguma classe de tarefas T e medida de desempenho P, se o seu desempenho em tarefas em T, medido por P, melhora com a experiência E” (T. Mitchell, 1997), ou seja, em aprendizagem de máquina a preocupação é em utilizar o raciocínio indutivo extraindo regras e padrões de grandes conjuntos de dados para aprender e realizar predições. A aprendizagem de máquina é usada para o desenvolvimento de modelos e algoritmos complexos capazes de realizar análises preditivas e decisões confiáveis através das relações dos dados fornecidos.

Atualmente há um grande esforço acadêmico e comercial no desenvolvimento de otimizações de técnicas e geração de novos modelos de aprendizagem.

Em 2006, a empresa de filmes *online* Netflix realizou a primeira competição "*Netflix Prize*" (www.netflixprize.com) para encontrar o melhor algoritmo que prevê as preferências do usuário e melhorar a precisão em seu algoritmo de recomendação de filmes existente em pelo menos 10%. Uma equipe conjunta composta por pesquisadores da *AT&T Labs* e *Big Chaos and Pragmatic Theory* construíram um modelo que venceu o Grande Prêmio em 2009 por US$ 1 milhão.

Grandes empresas como Google, Microsoft, Apple e Amazon estão empenhando um grande esforço no desenvolvimento e aprimoramento de assistentes virtuais. Peter Sondergaard, Vice-Presidente sênior de Pesquisa da Gartner afirmou no *Gartner Symposium 2015*: “Em 2020, as pessoas não irão usar aplicativos em seus aparelhos. Na realidade, os *apps* estarão esquecidos. As pessoas vão contar com os assistentes virtuais pra tudo. A era pós-*app* está vindo” (J. Kuntz, 2015).

Outra grande área com grande foco atualmente é o desenvolvimento de carros autônomos. Carros autônomos são carros capazes de se auto conduzirem sem a necessidade de intervenção humana pois, através de sensores estes carros são capazes de capturar diversas informações de situações que possam ocorrer no ambiente. As técnicas de visão computacional e aprendizagem de máquina são utilizadas para a interpretação dos dados obtidos por estes sensores, provendo a capacidade de percepção e reação autônoma do veículo (E. Dickmanns; A. Zapp, 1998; L. Meier et al., 2012; M. Turk et al., 1988).

Os exemplos citados são apenas alguns casos de um grande número de aplicações e ferramentas disponíveis atualmente graças aos avanços obtidos nas pesquisas de técnicas de aprendizagem de máquina.

### 2.2.1 Métodos de aprendizagem de máquina

Técnicas de aprendizagem de máquina são geralmente classificadas em três categorias, relativas ao tipo de aprendizado do algoritmo. A classificação do algoritmo depende da informação disponível previamente para o sistema aprender e a resposta fornecida durante o treinamento para auxiliar neste processo de aprendizagem.

Os tipos de algoritmos de aprendizado podem ser classificados como:

(1) Aprendizagem supervisionada: Na aprendizagem supervisionada são passadas várias entradas de exemplo e suas respectivas respostas desejadas. O objetivo do algoritmo será encontrar um mapeamento entre todas as entradas observadas e suas respostas, de forma a inferir uma resposta adequada para um novo valor ainda não conhecido.

(2) Aprendizagem não-supervisionada: Na aprendizagem não-supervisionada não há nenhuma informação prévia sobre o conjunto de dados, apenas as entradas são dadas. O algoritmo deve ser capaz de procurar um padrão entre os dados de forma a extrair certas características ou agrupar um conjunto de dados similares.
(3) Aprendizagem por reforço: Algoritmos de aprendizagem por reforço são usados para ambientes dinâmicos, onde o algoritmo deve ser capaz de adaptar suas respostas às mudanças externas sem nenhum auxílio.

Estas técnicas são utilizadas para resolução de problemas de classificação de padrões, regressão e agrupamento de classes.

Na classificação, ao receber uma entrada, o algoritmo deve observar as características presentes nesta entrada e definir a qual classe ou conjunto de classes a entrada passada pertence, como descrito na seção 2.1. Geralmente um algoritmo classificador utiliza aprendizado supervisionado para entender antes da classificação quais são as características dos padrões que definem cada classe, com base nos exemplos e em suas respostas já conhecidas.

Algoritmos de regressão retornam respostas no domínio contínuo. O objetivo deste tipo de algoritmo é observar características presentes em exemplos passados já conhecidos para inferir uma resposta futura dentro do domínio considerado.

Os algoritmos de agrupamento são em sua maioria não-supervisionados. Neste caso o algoritmo recebe um conjunto de dados e deve, a partir deste, extrair automaticamente informações de forma a nomear e separar o conjunto passado em classes não conhecidas.

## 2.2.2 Representação e função objetivo

Os métodos de aprendizagem de máquina podem ser representados pelo seguinte diagrama (Figura 2.5).

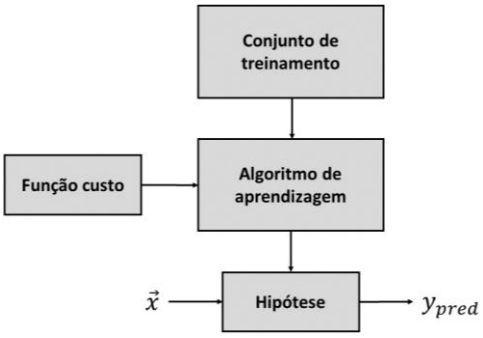


**Figura 2.5** - Representação de um sistema de aprendizagem de máquina. Fonte: o autor.

Um sistema de aprendizagem de máquina deve produzir uma hipótese $h(\vec{x}) = y_{pred}$, onde $\vec{x}$ representa o conjunto de características de um padrão a ser classificado e $y_{pred}$ a resposta que deve ser retornada pelo sistema. Para construção desta hipótese, o sistema deve ser treinado baseado em um conjunto de dados; este conjunto pode ser da forma $T = ((\vec{x}_0, y_0), \dots, (\vec{x}_n, y_n))$ em sistemas com aprendizagem supervisionada (são conhecidos os exemplos e suas respectivas respostas, exemplos rotulados) ou $T = (\vec{x}_0, \dots, \vec{x}_n)$ em sistemas não supervisionados (a resposta de cada

exemplo não é conhecida, exemplos não rotulados). Por fim, um sistema de aprendizagem de máquina deve utilizar um algoritmo para aprendizagem dos dados, que define como os dados do conjunto de treinamento são interpretados e utilizados para construção da hipótese.

Os modelos de aprendizagem de máquina supervisionados geralmente estão associados a resolver problemas que contenham uma função de custo conhecida. A função de custo é utilizada durante a execução do algoritmo de aprendizado para a construção da hipótese do sistema. A tarefa do modelo de aprendizagem é predizer um valor $y_{pred}$ dado um exemplo $\vec{x}$ onde a resposta predita seja igual ou próxima da resposta $y_{real}$, ou seja, a tarefa do modelo é minimizar o erro entre sua resposta e a resposta real do problema. Para isto é necessária a definição da função de custo, onde o custo é representado pelo erro entre a resposta predita e a resposta real do problema.

Se o sistema de aprendizagem supervisionado é utilizado para resolução do problema de classificação, então as respostas possíveis do sistema são definidas dentro de um conjunto fechado de valores, onde cada valor corresponde a uma classe conhecida. Se o sistema resolve o problema de regressão, então ele deve prever a resposta dentro de um intervalo de possibilidades real, tendo como base o conhecimento da resposta para cada exemplo de treinamento.

### 2.2.3 Desenvolvimento de soluções

Em aprendizagem de máquina vários tipos de problemas são abordados, alguns deles já descritos anteriormente. Para resolução destes problemas, uma análise prévia deve ser realizada para decisão dos melhores modelos para resolução do problema e dos processos de aprendizagem utilizados no desenvolvimento da solução.

A escolha do modelo é o primeiro problema que deve ser considerado. Cada modelo funciona de uma forma diferente para cada configuração utilizada. Alguns podem resolver certos tipos de problema de forma mais eficiente enquanto outros podem priorizar a precisão do resultado. Um estudo profundo do modelo é necessário, levando em consideração seu desempenho e grau de adaptação ao problema desejado. Alguns aspectos importantes a serem considerados incluem o consumo de memória utilizado pelo modelo, o tempo necessário para treinamento, a capacidade de generalização para o problema e o tempo de resposta.

Definido o modelo a ser utilizado, o próximo passo é a verificação dos dados disponíveis. Algumas vezes pode ser necessária a manipulação dos dados da base disponível. Problemas como dados ausentes, inconsistência de valores, invariâncias e desbalanceamento são fatores que devem ser resolvidos antes do início do treinamento.

Após o ajuste dos dados pré-processados devem ser definidas quais variáveis são importantes para caracterização do problema, ou seja, quais características dos dados devem ser observadas com a finalidade de obter o melhor desempenho para o modelo selecionado.

Outra característica a ser considerada durante o planejamento de desenvolvimento é a escolha dos parâmetros e métodos de ajuste utilizados para treinamento do modelo.

Os parâmetros do modelo definem o quanto ele pode aprender sobre o problema e a escolha de como treinar esses parâmetros tem um impacto direto no resultado e no desempenho final do modelo.

Os principais parâmetros de treinamento para o modelo de aprendizagem de máquina em geral são: algoritmos de aprendizagem, taxa de aprendizagem e critério para correção de *overffiting*.

A taxa de aprendizagem define o passo para variação dos parâmetros do modelo em cada iteração do treinamento e quanto menor a taxa de aprendizagem mais tempo o modelo vai necessitar para o treinamento, porém mais especializado ele vai se tornar para obter os resultados desejados.

Já o critério para correção de *overfitting* tenta minimizar o *overfitting*, problema que ocorre quando a rede é especializada demais no conjunto de treinamento, obtendo assim uma taxa de erro de classificação baixa dentro do conjunto de treinamento e uma taxa de erro alta no conjunto de testes. As escolhas do critério de parada do treinamento e de correção de *overffiting* são importantes parâmetros no planejamento do modelo.

## 2.3 Redes Neurais Artificiais

Redes Neurais Artificiais (*Artificial Neural Networks*, ANNs) é uma categoria de métodos de aprendizagem de máquina, conhecida como sistemas conexionistas (G. Hinton, 1989). Técnicas de redes neurais são baseadas no comportamento cerebral humano, no qual os problemas são resolvidos e as informações são enviadas utilizando grupos de neurônios biológicos conectados entre si. Estes neurônios propagam impulsos elétricos como forma de comunicação e tais impulsos são recebidos por outros neurônios com a finalidade de realização de alguma ação. Cada neurônio possui um estado de ativação ou desativação e o conjunto destes estados na rede de neurônios configura uma resposta do sistema nervoso.

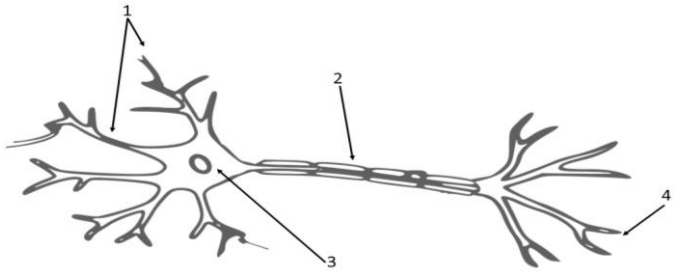


**Figura 2.6** - Estrutura de um neurônio biológico. 1. Dendritos. 2. Axônio. 3. Corpo celular. 4. Sinapses.

Cada unidade neural (Figura 2.6), pertencente ao grupo neural, possui um conjunto de ligações de entrada. Nessa estrutura as informações são recebidas na forma de impulsos elétricos pelo neurônio, passando pelos seus dendritos. Após receber o impulso, o neurônio deve produzir uma resposta de acordo com a sua configuração atual no corpo celular e propagar a resposta para suas conexões de saída. Tais respostas são computadas e passam pelo axônio do neurônio, que é a estrutura que vai ativar ou desativar o estado do neurônio propagando ou não a resposta dele adiante. A resposta do axônio é propagada se ele permanecer ativo a outros neurônios pela conexão terminal sinapse. Estas unidades recebem continuamente sinais elétricos de outros neurônios conectados a ele e, dependendo da configuração de cada neurônio, a rede neural consistirá de um conjunto de neurônios ativos e desativados o que indicará qual resposta o sistema nervoso quer obter.

Os neurônios são a estrutura básica do cérebro e do sistema nervoso. Cada neurônio é especializado em enviar algum tipo de informação. De forma simplificada, os impulsos recebidos por cada neurônio podem ou não gerar uma resposta a outros vizinhos conectados no grupo neural e cada neurônio é capaz de aprender a interagir com o conjunto de células a que são ligados ou a criar novas ligações com outras células se necessário. Os neurônios devem saber responder a cada estímulo recebido e qual resposta deve propagar baseado na experiência de cada indivíduo.Com base nestas características do sistema nervoso, Warren McCulloch e Walter Pitts propuseram em 1943 o primeiro modelo computacional para representar a função de um neurônio biológico (W. McCulloch; W. Pitts, 1943). O modelo ficou conhecido como neurônio de McCulloch&Pitts (Figura 2.7), e nele o neurônio é representado através de funções matemáticas. Posteriormente uma implementação foi realizada por Frank Rosenblatt em 1957, nomeada de *Perceptron*.

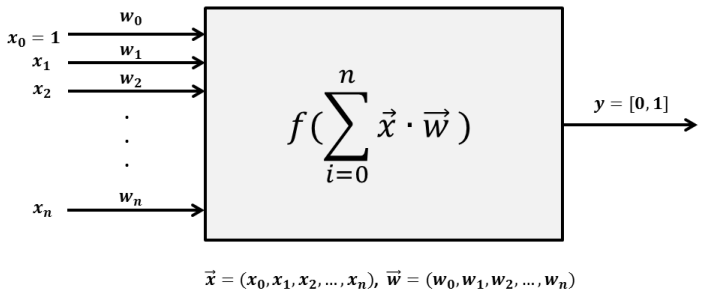


**Figura 2.7** – Representação do modelo de neurônio de McCulloch&Pitts. O modelo recebe um vetor de entrada representando as características do padrão e deve computar o produto vetorial entre o vetor de pesos e a entrada, produzindo um limiar que é usado como entrada para a função de ativação, retornando uma resposta ativa ou não como resultado. Fonte: o autor.

O neurônio de McCulloch&Pitts contém um conjunto de ligações de entrada ponderadas, onde recebe as informações representadas por um vetor numérico $\vec{v} = \vec{x} \cdot \vec{w}$ em que $\vec{x}$ é o vetor de entrada (impulsos recebidos) e $\vec{w}$ é o vetor de pesos da rede caracterizando o parâmetro que representa a experiência da rede. O vetor $\vec{v}$ corresponde à função dos dendritos nos neurônios biológicos. A informação é computada através da soma ponderada destas entradas ($net = \sum_{i=0}^{n} \vec{x} \cdot \vec{w}$); o valor $net$ obtido representa uma característica do problema. O resultado é então passado à função de ativação do neurônio, onde será gerada uma resposta ao valor, ativando ou não a saída do neurônio ($f(net) = [0,1]$) (Figura 2.8). A função de ativação corresponde ao axônio do neurônio biológico e o resultado final é passado ao vetor de saída do modelo ($\vec{y} = [0,1]$) que representa as sinapses.

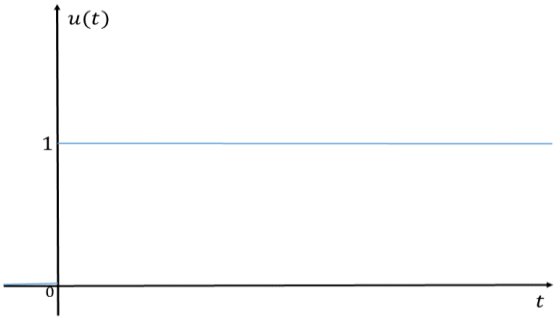


**Figura 2.8** – Função degrau. A função degrau pode ser utilizada para representação de funções de ativações tendo como resposta o valor 1 (estado ativo) para valores maiores que zero ou valor 0 (estado inativo) para valores menores que zero. A definição da classe pertencente a fronteira do hiperplano gerado deve ser definida no projeto da estratégia de aprendizado do sistema. Fonte: o autor.

Uma unidade *Perceptron* é capaz de separar duas classes de dados distintas gerando um hiperplano de separação entre elas através do resultado da regra de propagação, que consiste do produto vetorial entre as entradas e os seus respectivos pesos. Se o problema é classificar dados em duas classes distintas $C_1$ e $C_2$, e sabendo que $net = \sum_{i=0}^{n} \vec{x} \cdot \vec{w}$, é possível treinar os pesos da rede para resolver o problema de classificação definindo o vetor de entrada $\vec{x}$ que descreve um padrão. Se $net > 0$ o padrão pertence à classe $C_1$ e se $net < 0$ o padrão pertence à classe $C_2$ (Figura 2.9).

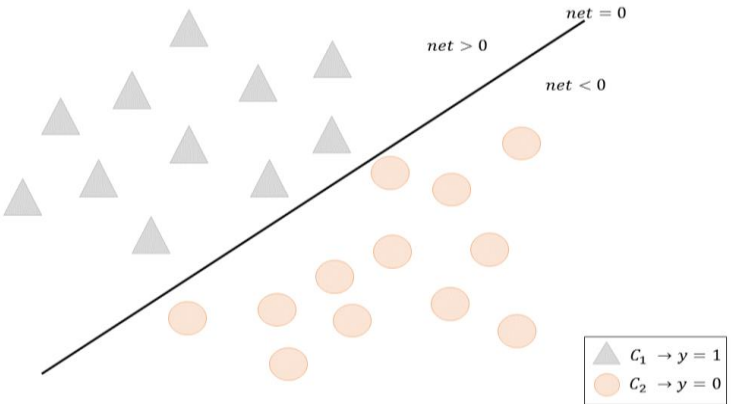


**Figura 2.9** - Hiperplano de separação (simplificado para duas classes). O neurônio deve treinar seus parâmetros (pesos) de forma a gerar o hiperplano em que todos os exemplos de cada classe estejam contidos de um lado especifico da curva gerada. Neste exemplo, se $\boldsymbol{net > 0}$ o exemplo está contido acima da curva tendo como resposta de saída $\boldsymbol{y = 1}$, sendo classificado para a classe $\boldsymbol{C_1}$ e, se $\boldsymbol{net < 0}$ o sistema gera como resposta $\boldsymbol{y = 0}$, sendo classificado para a classe $\boldsymbol{C_2}$. A posição da fronteira do hiperplano corresponde ao valor $\boldsymbol{net = 0}$, se algum exemplo pertence a fronteira do hiperplano a correspondência de classe para este exemplo deve ser definido no projeto do modelo Fonte: o autor.

É necessário realizar um processo de treinamento para a rede ser capaz de aprender a classificar corretamente os padrões em suas respectivas classes, sendo o algoritmo de aprendizagem do *Perceptron* iterativo e supervisionado. Separando o conjunto de treinamento com suas respectivas respostas, a cada passo do treino os pesos são ajustados de forma a adequar a curva de separação aos exemplos de treinamento. As seguintes equações atualizam os valores do peso durante o treinamento:

$$\vec{w}(t+1) = \vec{w}(t), se\ \vec{x} \in C_1\ e\ net > 0 \quad (2.1)$$
$$\vec{w}(t+1) = \vec{w}(t), se\ \vec{x} \in C_2\ e\ net < 0 \quad (2.2)$$
$$\vec{w}(t+1) = \vec{w}(t) + \eta\vec{x}, se\ \vec{x} \in C_1\ e\ net < 0 \quad (2.3)$$
$$\vec{w}(t+1) = \vec{w}(t) - \eta\vec{x}, se\ \vec{x} \in C_2\ e\ net > 0 \quad (2.4)$$

onde $\vec{x} = (x_0, x_1, \dots, x_n)$, $\vec{w} = (w_0, w_1, \dots, w_n)$ e $\eta$ = taxa de aprendizagem.

Para cada exemplo os valores dos pesos são ajustados até a rede convergir em um plano que separe as classes de dados $C_1$ e $C_2$.Este modelo é bastante limitado, sendo capaz de resolver apenas problemas que sejam linearmente separáveis além de possuir um algoritmo de treinamento bastante custoso para a época em que foi criado, retardando o avanço nas pesquisas em novos algoritmos na área de redes neurais. Posteriormente, foram descobertas novas formas de contornar os problemas do modelo de McCulloch&Pitts com Redes Multicamadas (*Multilayer Perceptron*, MLP) (K. Hornik et al., 1989) (Figura 2.10), capazes de resolver problemas não lineares, com novos algoritmos de treinamento menos custosos e com melhor desempenho na modificação dos pesos, novas funções de ativações e novos modelos de neurônios.

Em redes MLP as funções de ativação utilizadas devem ser contínuas e diferenciáveis em todos os pontos. Isso se dá pelo algoritmo de aprendizado utilizado pela rede que utiliza a primeira derivada para computar o erro que será utilizado para corrigir os pesos da rede. A função Sigmoide ou logística (Figura 2.11a) é a mais popular para este tipo de arquitetura. Sua característica de ativação é semelhante a função degrau vista anteriormente para o Perceptron, mas com uma região de transição continua representando a mudança de estado do neurônio, possuindo pontos diferenciáveis em toda sua extensão.

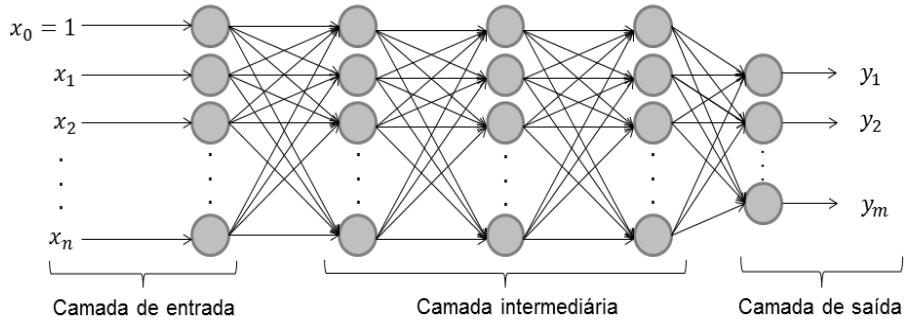


**Figura 2.10** - Rede Multilayer Perceptron (MLP). A rede consiste de uma camada de entrada do tamanho do vetor de características de entrada. Uma, várias ou nenhuma camada intermediaria com tamanho a ser determinado em cada aplicação, e uma camada de saída definida de acordo com o problema. Fonte: o autor.

## 2.3.1 Estruturas das redes neurais

Toda arquitetura de rede neural pode ser descrita de forma geral por um conjunto de características. Os elementos essenciais que definem uma rede neural são: o tipo de unidade de processamento, a topologia da rede e a estratégia de aprendizagem. Estes elementos são descritos com mais detalhes abaixo.

## Unidade de processamento

A unidade de processamento ou neurônio é o elemento básico de toda rede neural e nela são definidas a regra de propagação para a rede e a função de ativação utilizada. Cada rede neural emprega um tipo específico de neurônio para resolução de alguma tarefa. Um exemplo é a rede MLP que utiliza como neurônio o modelo de McCulloch&Pitts, o *Perceptron* simples. A regra de propagação do *Perceptron* é o somatório das entradas ponderadas por seus respectivos pesos e para a função de ativação pode ser utilizada qualquer função que tenha como característica valores que representem um estado ativo e um estado inativo alternando em um dado ponto do domínio da entrada ou em um intervalo muito curto, como visto na Figura 2.8.

Redes que utilizam o *Perceptron* como neurônio tem um alto grau de generalização na resolução dos problemas, ou seja, dado um padrão de classe não conhecida e com características totalmente diferentes das características aprendidas, o novo padrão será atribuído a classe que mais se aproximar de suas características. Como viso anteriormente, cada *Perceptron* gera um hiperplano de separação no espaço de dados, classificando o padrão de acordo com sua posição em relação à curva gerada, isso permite à rede obter uma resposta para todo domínio do problema (Figura 2.11a).

Um outro tipo de neurônio muito utilizado é o neurônio de Base Radial (*Radial Basis Function*, RBF) (J. Park; J. Sanderberg, 1991), tipo de neurônio que tende a generalizar menos o problema, criando regiões fechadas para cada tipo de classe aprendida (Figura 2.11b). Os neurônios de base radial utilizam a função de distância euclidiana como regra de propagação, cada neurônio tem um conjunto de centroides $C$, onde $C = (c_0, c_1, \ldots, c_n)$, com $n$ sendo o tamanho do vetor de entrada. A função de ativação do neurônio de base radial são funções de ativação local, como por exemplo, a função Gaussiana que pondera o resultado de acordo com a resposta local de cada neurônio, ou seja, as menores distâncias entre a entrada e o centroide do neurônio geram uma maior saída como resposta. Neurônios com função de base radial são muito utilizados para agrupamento de classes e classificação de padrões.

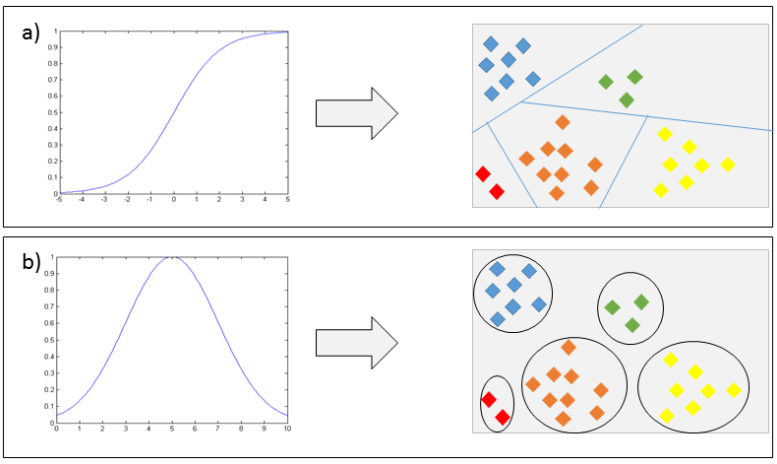


**Figura 2.11** – a) Resposta do *Perceptron* simples em uma rede MLP com função de ativação *Sigmoide* (esquerda), a área de decisão gerada pela rede (direita) é formada por vários hiperplanos formando uma região de decisão aberta (aproximador universal). b) Resposta gerada por um neurônio com base radial com função de ativação local Gaussiana (esquerda) em uma rede RBF, a área de decisão (direita) retorna grupos fechados localizados nos centroides treinados em cada neurônio. Fonte: o autor.

## Topologia da rede

Topologia da rede é um termo que se refere à organização e interação dos neurônios. Com o grande avanço no desenvolvimento de arquiteturas de redes neurais vieram também a grande quantidade de possibilidades existentes de construir um modelo para resolução de diferentes problemas. Como exemplo, a rede MLP tem uma camada específica para processar a entrada onde a quantidade de neurônios é geralmente o tamanho do vetor de entrada, uma camada para saída com quantidade de neurônios igual ao número de classes a classificar e entre essas duas camadas a rede pode conter uma, várias ou nenhuma camada intermediária com um número indeterminado de neurônios. Cada uma dessas camadas intermediarias tem a função de tentar aprender um tipo de característica do problema, com a finalidade de distribuir o problema de classificação dentro da rede neural. A definição da constituição das camadas intermediarias em uma rede MLP depende do problema e no grau de especialização que a rede precisa ter para resolução do mesmo.

Várias são as arquiteturas de redes neurais existentes atualmente, como as redes Função de Base Radial (*Radial Basis Function*, RBF) (J. Park; J. Sanderberg, 1991), Redes de Ressonância Adaptativas (*Adaptive Ressonance Theory*, ART), Redes Neurais Convolucionais (*Convolutional Neural Networks*, CNNs), Redes de Mapeamento Auto-Organizavéis (*Self Organizing Maps*, SOMs) (T. Kohonen; P. Somervuo, 1998), cada uma com seus parâmetros e topologias a serem definidas para o problema desejado.

## Algoritmo de aprendizagem

O último elemento básico de uma rede neural a ser definido é o algoritmo de aprendizagem ou estratégia de aprendizado. Algoritmo de aprendizagem são os passos a serem seguidos pela rede neural para ajustes dos seus parâmetros durante a fase de treinamento. A partir do algoritmo de aprendizagem, a rede é capaz de aprender as características do problema e predizer as respostas para o problema. As equações (2.1), (2.2), (2.3) e (2.4) definem as regras de atualização dos pesos do *Perceptron* simples; para cada iteração do problema o neurônio ajusta os seus pesos de entrada de acordo com o conjunto de equações, e os pesos caracterizam a informação relevante para cada neurônio.

Uma abordagem de treinamento mais sofisticada e eficiente utilizada em diversos tipos de redes neurais é o algoritmo de Gradiente Descente. O Gradiente Descente é o algoritmo padrão utilizado no treinamento da rede MLP.

O Gradiente Descente ou *Backpropagation* tem duas fases de operação. Na primeira fase a rede testa para cada exemplo de treinamento os parâmetros atuais, e cada exemplo resultará em um erro de classificação (passo *feedfoward*). Na segunda fase a derivada do erro de classificação é utilizada para atualizar os parâmetros da rede neural em cada camada no sentido inverso (da camada de saída para a camada de entrada), o que deriva o nome *Backpropagation*. Este processo se repete durante todo o treinamento, aperfeiçoando os pesos da rede neural a cada iteração.

Este algoritmo é o mais utilizado para treinamento das redes neurais MLP em conjunto com a utilização da função de ativação Sigmoide. O gradiente descendente é ainda bastante usado em diversos modelos de redes neurais atuais, e alguns modelos de DNNs utilizam variações deste algoritmo para treinamento das últimas camadas de classificação.

### 2.3.2 Redes neurais para reconhecimento de imagens

Técnicas de redes neurais podem ser utilizadas para construção de aplicações em reconhecimento de imagens.

A forma mais comum para descrever imagens em arquiteturas de redes neurais é utilizar os valores da intensidade de cor em cada *pixel* que compõem a imagem, onde cada *pixel* representará uma característica do vetor de entrada.

Uma imagem digital pode ser vista como um vetor $m\,x\,n$, onde $m$ e $n$ representam a largura e a altura da imagem, respectivamente. Cada posição do vetor representa uma informação contida na imagem, e essa informação é chamada de *pixel*. Dependendo da representação da imagem, cada *pixel* pode conter um ou mais valores que o descreva. Por exemplo, imagens em tons de cinza utilizam geralmente valores de 0 a 255 para caracterizar cada cor na faixa de tons de cinza existente, onde 0 representa o preto e 255 representa o branco. Imagens coloridas utilizam mais de um canal para representar a intensidade de cada tipo de cor contida no *pixel*, em uma representação RGB qualquer cor pode ser representada por uma combinação das cores vermelho (*red*), verde (*green*) e azul (*blue*). Em uma imagem RGB os *pixels* são caracterizados por um vetor tridimensional, onde cada posição deste vetor descreve a intensidade de cor vermelha, verde e azul contidas na imagem. No caso de imagens coloridas a imagem pode ser descrita como um vetor $m\,x\,n\,x\,c$, sendo $m$ e $n$ a largura e altura da imagem e $c$ o número de canal de cores por *pixel*.

O problema da utilização de redes neurais para reconhecimento está na quantidade de parâmetros que precisam ser treinados e armazenados na rede. Por exemplo, imagens de tamanho 1024x780 *pixels* são bastante utilizadas atualmente em diversas aplicações existentes. Utilizando a abordagem mencionada anteriormente e, considerando uma imagem RGB como entrada para a rede neural, a representação da imagem resultaria em uma rede com 2.396.160 pesos existentes na camada de entrada, o que torna necessário a utilização de computadores com grande poder computacional para o treinamento da rede, podendo levar entre dias a semanas o tempo de treinamento da rede neural, dependendo da quantidade de informação disponível na base de treinamento.

Uma abordagem idealizada para contornar este problema foi a utilização de redes que respondam a regiões de sobreposição local, ou seja, obtenham valores para apenas uma pequena região da imagem com algumas informações compartilhadas entre outras unidades com a finalidade de minimizar a quantidade de pré-processamento necessário. Essas redes são denominadas de Redes Neurais Convolucionais (CNNs) ou *ConvNets*, e tais redes são uma variação da rede MLP e foram inspiradas nos processos biológicos da visão animal.

### 2.3.3 Redes Neurais Convolucionais

Hubel e Wiesel mostraram que o córtex visual dos gatos e macacos contém neurônios que respondem individualmente a pequenas regiões do campo visual. A região do espaço visual dentro da faixa de estímulos é conhecida como campos receptivos (D. Hubel; T. N. Wiesel, 1968). Os neurônios vizinhos têm campos receptivos semelhantes e sobrepostos, o tamanho e a localização de cada campo variam através do córtex para formar o mapa visual completo do espaço.

Redes Neurais Convolucionais (*Convolutional Neural Networks*, CNNs ou *ConvNets*) são variações da rede MLP *feed-forward*, onde as organizações das conexões entre os neurônios são inspiradas no padrão de organização do córtex visual animal. Cada neurônio responde a um estímulo em uma região restrita do espaço, chamada de campo receptivo. Os campos receptivos de cada neurônio compartilham algumas informações entre os campos receptivos vizinhos (Campos receptivos com sobreposição). A resposta de cada neurônio da rede pode ser aproximada matematicamente pela operação de convolução. CNNs foram projetadas para usar a mínima quantidade possível de pré-processamento, sendo usada frequentemente em aplicações de reconhecimento de imagens e processamento de linguagem natural. Geralmente uma arquitetura CNN contém várias camadas convolucionais seguidas por camadas de *pooling* (camada para realização de redução dos dados*)*, e, camadas totalmente conectadas (MLP) no final do modelo para classificação (Figura 2.12). Cada uma dessas camadas será descrita posteriormente.

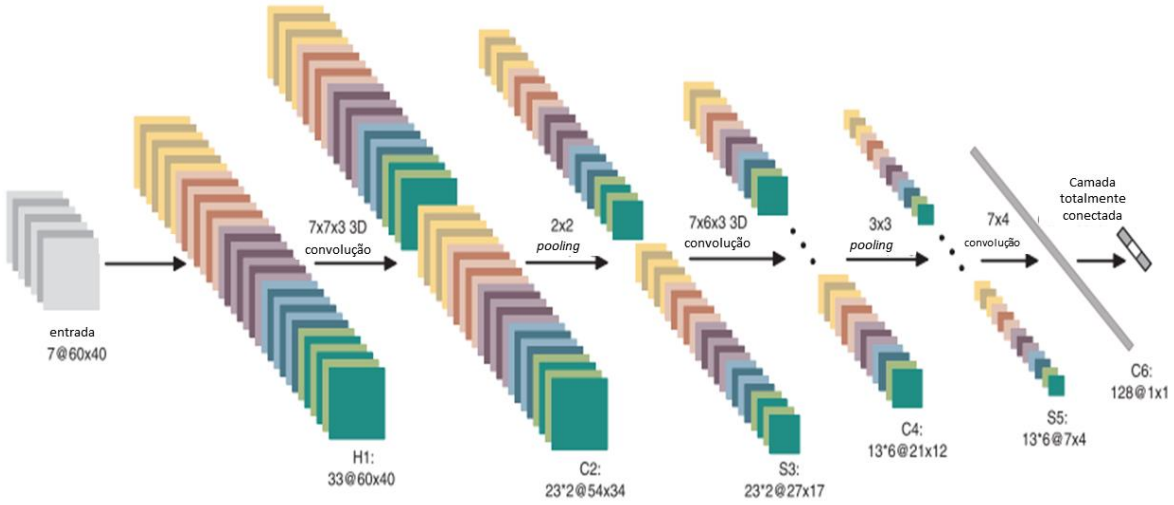


**Figura 2.12** – Exemplo de uma arquitetura de rede neural convolucional. A rede consiste de um conjunto de camadas de convolução seguidas por camadas de *pooling* utilizadas para filtrar as informações dos dados. Ao final do processo, geralmente uma camada MLP é utilizada para realizar a classificação dos dados. Fonte: S. Ji et al., 2013.

Em reconhecimento de imagens, cada neurônio tem um conjunto de conexões correspondentes a uma região da imagem de entrada. As saídas destes neurônios em conjunto correspondem a imagem de entrada com regiões de sobreposição entre as bordas de cada região receptiva. Cada camada aplica um filtro à sua entrada com a finalidade de distinguir alguma característica específica da imagem passando como

entrada sua resposta às camadas seguintes; esse processo é repetido como forma de obter a melhor representação da imagem de entrada. Abaixo são descritos os principais componentes que compõem as arquiteturas CNNs.

### Camadas convolucionais

A camada convolucional é o principal elemento que caracteriza uma CNN. Os parâmetros dos neurônios consistem em pesos para os filtros (*Kernels*) que podem ser aprendidos durante o treinamento. Durante a fase de *feed-forward*, cada filtro convolve com o vetor de entrada (Figura 2.13a, b e c). Durante a convolução, o filtro computa o produto de cada ponto da entrada na sua região produzindo um mapa de ativação. Como resultados desta operação a rede gera filtros que ativam quando encontram algum tipo específico de característica em alguma posição da entrada. As camadas convolucionais tendem a se especializar em características específicas de certos objetos.

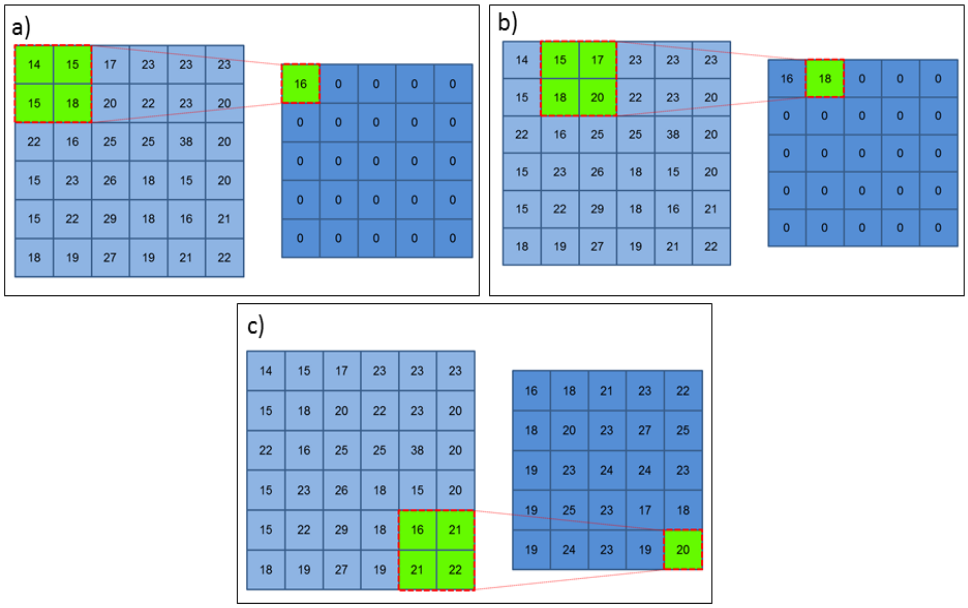


**Figura 2.13** – Operação de convolução com um filtro de média 4x4. O filtro desliza por todas as posições do vetor de entrada computando a operação de média para cada janela formada. Na letra a) a janela computa a primeira iteração da operação de convolução, na letra b) é computada a segunda iteração deslizando a janela para esquerda uma posição, na letra c) a ultima iteração é computada após serem calculados os valores de cada janela. Em redes CNNs essa operação é simulada através das conexões em pequenas regiões da entrada, onde cada conjunto de conexão representa uma posição da janela de convolução. Fonte: o autor.

Na rede convolucional, nem todos os neurônios são conectados entre si durante a transição de camadas como na rede MLP. CNNs exploram a correlação espacialmente existente localmente e os padrões de conectividade fazem com que cada neurônio seja conectado a apenas uma pequena região da entrada. As conexões locais no espaço são locais em relação à altura e largura, mas sempre se estendem

para todo o volume de entrada (canais de cores da imagem), desta forma os filtros produzidos geram respostas mais fortes para padrões de entrada local.

### Camadas de *Pooling*

As camadas de *pooling* são usadas para diminuir o número de amostras de saída (*downsampling*). A função de *pooling* mais usada em modelos CNN é a *max pooling*, onde a imagem de entrada é dividida em um conjunto de regiões igualmente espaçadas e não sobrepostas dentro da imagem e para cada região o valor máximo é recuperado e passado a frente (Figura 2.14). A função desta camada é reduzir a quantidade de parâmetros necessários para representação dos dados, controlando também a sobreposição de cada resposta e a redundância espacial gerada durante a convolução do filtro treinado. Este tipo de operação pode obter para a arquitetura invariância a translação, gerando uma nova representação onde apenas a relação entre os principais pontos de cada região na imagem são analisados. A camada de *pooling* é comumente aplicada após conjuntos de camadas convolucionais em uma CNN.

Camadas de *pooling* são independentes dos parâmetros da rede, recebendo uma entrada e redimensionando-a espacialmente. Outras funções de *pooling* podem ser utilizadas como funções de média ou mediana, além de diferentes tamanhos de filtros para realização da operação de *downsampling*. A tendência de uso de camadas de *pooling* é em aplicar filtros pequenos em cada etapa da arquitetura (B. Graham, 2014). Reduções maiores ajudam a controlar *overfitting*, mas são mais eficientes em conjuntos de dados pequenos.

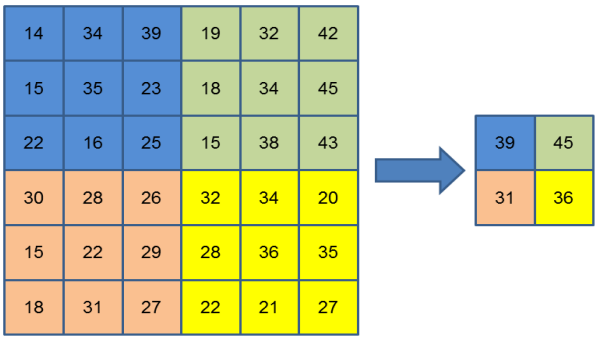


**Figura 2.14** - Operação de *max pooling* realizada na camada de *pooling*. Na figura, é feita uma operação de *max pool* com tamanho *downsampling* em janelas 6x6. A imagem é dividida em regiões semelhantes do tamanho definido pela janela utilizada, os valores máximos de cada região são armazenados e passados a frente, gerando uma nova representação da imagem escalonada. Fonte: o autor

### Camadas totalmente conectadas (*fully-connected*)

Camadas totalmente conectadas ou camadas *fully-connected* são camadas de neurônios com conexões comuns como na rede MLP (Figura 2.10), onde cada neurônio está conectado com todo neurônio existente na camada posterior. As

camadas totalmente conectadas aparecem geralmente após as camadas convolucionais e de *max pooling*, elas têm a função de realizar o raciocínio de alto nível, ou seja, são as camadas que utilizam as características intermediarias para retornar o resultado desejado.

### Unidades retificadoras lineares (*Rectified Linear Units*)

*Rectified Linear Units* (ReLUs) é uma camada de neurônios que aplica neurônios com funções de ativação não saturantes. A função utilizada em unidades ReLU é a função *rectifier* ou retificadora. Diferente das funções mais utilizadas para ativação dos neurônios, como por exemplo a função Sigmoide, a função retificadora não contém um limite de resposta definido (Figura 2.5). Na prática, esta função alcança melhores taxas de desempenho de tempo durante o treinamento, contendo uma grande variação de valores durante a inicialização e atualização dos pesos da rede neural. Alguns problemas podem ocorrer devido a característica não linear no ponto de ativação da função como a falta de fronteira para os valores possíveis, problemas de aprendizado com altas taxas de aprendizagem (capítulo 5) e a existência de pontos não diferenciáveis.

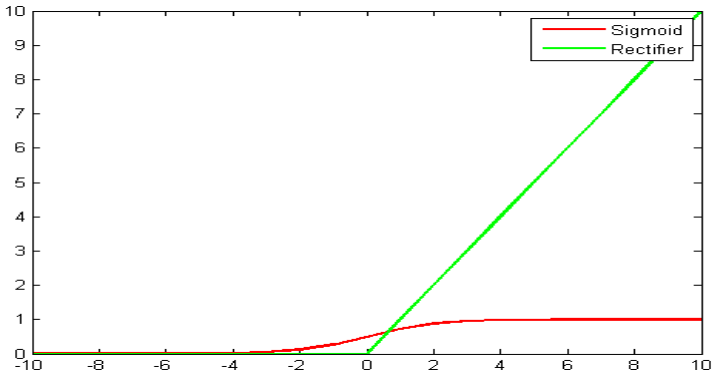


**Figura 2.15** – Função de ativação retificadora vs sigmoide. A retificadora aumenta a eficiência de treinamento em redes neurais convolucionais e profundas por ser uma função não-saturante, diferente da sigmoide e a maioria das funções de ativações utilizadas em neurônios artificiais. Fonte: o autor.

Proposta inicialmente por Hahnloser (R. Hahnloser et al., 2000) a função retificadora é bastante utilizada em CNNs para aumentar a resposta às propriedades não-lineares da função de decisão sem afetar os campos receptivos da camada de convolução. Unidades ReLU resultam em um treinamento da rede mais rápido sem ocasionar perdas na precisão do modelo (A. Krizhevsky et al., 2012).

### Unidades de classificação *Softmax*

Estas unidades são utilizadas nas camadas onde é computado o erro entre a resposta predita pela rede e a resposta real do exemplo analisado. Estas unidades utilizam a função que computa perdas *softmax*. Geralmente essas unidades estão

inseridas nas camadas finais da rede neural e usadas para modelos de classificação, onde será computado o valor da classe mais provável ao qual pertença algum exemplo de teste.

A função *softmax* funciona como uma função de probabilidade acumulada. Dado um exemplo de entrada, para cada classe existente no domínio da aplicação, a função tem como resultado uma lista de valores prováveis de pertinência deste exemplo. Nesta lista, a classe que tem o maior valor é a mais provável de ser a classe à qual pertence o exemplo analisado.

### 2.3.4 Métodos de ajustes

Os métodos de ajustes são utilizados para modificação dos parâmetros do modelo durante o treinamento, com o objetivo de melhorar o desempenho do modelo, o tempo de treinamento, a precisão e problemas de *overfitting*.

Os métodos mais utilizados em modelos de CNN para ajustes de parâmetros são descritos abaixo:

*Dropout*: O *dropout* é uma técnica que tem como objetivo reajustar os parâmetros da rede para evitar o *overfitting*. Em modelos DCNN, a alta quantidade de parâmetros (pesos da rede) torna a identificação e ajuste de *overfitting* um processo complexo, principalmente nas camadas de classificação totalmente conectadas, onde o número de conexões é alto. Com o *dropout*, em cada etapa do treinamento alguns pesos da rede são desprezados com probabilidade $p$, de modo a o processo resultar em uma rede reduzida. Na próxima fase, a rede é treinada apenas com os pesos ainda ativos, e após a iteração, os pesos são reinseridos na rede para treinamento. A probabilidade de cada peso pode ser definida pelo valor que ele assume durante a iteração.

Esta técnica reduz o tempo de treinamento total para a rede, até mesmo para arquiteturas CNNs. O ajuste de um subconjunto dos pesos torna a combinação de interações complexas bem definidas e o modelo pode ser ajustado de forma a obter maior generalização das características dos dados, evitando o *overfitting* para o conjunto de treinamento.

Decaimento de pesos: É uma forma de regularização dos valores dos pesos da rede. Com este método, um erro adicional proporcional a alguma métrica é inserido em cada peso. Esta técnica pode ser utilizada para aumentar a influência de alguma característica durante o treinamento ou inserir maior complexidade na adaptação dos pesos aos exemplos de treinamento. As métricas mais utilizadas são a soma total dos pesos por camada, ou o quadrado da soma dos pesos.

## 2.4 Redes Neurais Profundas

Redes Neurais Profundas ou *Deep Neural Networks* (DNNs) é um ramo de pesquisa em redes neurais que tem como objetivo a construção de algoritmos para modelagem de dados com alto nível de abstração. Uma rede profunda consiste de

uma rede com muitas camadas intermediarias entre a camada de entrada e de saída do modelo (uma arquitetura pode ser considerada profunda com pelo menos 5 camadas entre a entrada e saída). Estas redes geralmente utilizam métodos de transformações não-lineares e se baseiam em gerar múltiplas representações para aprendizagem dos dados de exemplo (L. Deng; D. Yu, 2014; Y. Bengio et al., 2013).

Os modelos de DNN tem como objetivo principal a criação de modelos para aprendizado das características e representações de dados em grande escala. Várias arquiteturas DNN são amplamente aplicadas para resolução de problemas em visão computacional. Entre elas, as mais utilizadas são redes convolucionais, redes recorrentes e *belief networks*. Tais arquiteturas podem ser utilizadas para reconhecimento facial, processamento de linguagem natural, reconhecimento de objetos, entre outros.

As camadas da rede são construídas em cascata onde a resposta computada em cada uma é usada na camada seguinte. A resposta é propagada sucessivamente entre os neurônios até a camada de saída e as unidades de processamento utilizadas assumem funções e regras não-lineares para extração das características dos dados de exemplo. Cada camada tem a função de gerar uma representação intermediaria dos dados que serão interpretados pelas camadas subsequentes. Os algoritmos utilizados em arquiteturas DNN podem ser supervisionados ou não, sendo na maioria das vezes utilizado o aprendizado supervisionado para classificação.

O aprendizado profundo de uma DNN explora a ideia de que os dados podem ser representados e analisados através de uma hierarquia de conceitos, onde a arquitetura mantém controle do nível de abstração interpretado da camada superior para a mais inferior. O aprendizado de uma DNN é construído de forma incremental, camada por camada. Cada camada reconhece e interpreta um nível diferente de abstração e as representações intermediarias tem como objetivo a extração dos componentes principais existentes nos dados de entrada.

Estruturas DNN que utilizam redes convolucionais são chamadas de Redes Neurais Convolucionais Profundas (*Deep Convolutional Neural Networks*, DCNNs). Há um grande esforço na literatura no desenvolvimento de novas arquiteturas DCNNs para resolução de problemas de imagens, estas redes conseguem obter um alto grau de precisão quando treinadas em bases de dados com grandes quantidades de exemplos.

### 2.4.1 Big Data

Atualmente, dados são obtidos em vários lugares e de diversas formas distintas. *Smartphones*, relógios, computadores e qualquer tipo de dispositivo eletrônico é capaz de disponibilizar uma grande quantidade de informação. As informações em grande escala podem ser armazenadas em banco de dados em nuvem e milhões de *bytes* de usuários são armazenados diariamente. A grande quantidade de fluxo de dados e informação obtida pode ser definida como *Big Data*.

Ao mesmo tempo em que o nível de informação cresce, cresce também a necessidade de modelos computacionais eficientes e robustos para análise da grande quantidade de dados disponíveis. A avaliação destes dados em larga escala por humanos se tornou impraticável e, o esforço atual está no desenvolvimento de aplicações capazes de gerenciar e avaliar tais dados.

O uso de técnicas de aprendizagem de máquina é uma das escolhas obvias para resolução deste problema uma vez que tais técnicas requerem uma grande quantidade de dados para aprimoramento do sistema, obtendo maior precisão e robustez de predição.

A era do *Big Data* é um dos fatores capazes de impulsionar as pesquisas da área de aprendizagem de máquina, incluindo principalmente o uso de métodos de DNN, que requerem uma grande quantidade de informação para conseguir um bom desempenho de classificação.

### 2.4.2 Uso de GPU

Outro fator importante no uso de DNNs é o rápido desenvolvimento de novas arquiteturas e ferramentas de *hardware*, principalmente com a disponibilidade de novas e robustas unidades gráficas de processamento (GPUs).

As GPUs são microprocessadores que manipulam gráficos computadorizados. Diversas bibliotecas gráficas existem para uso em GPU que otimizam funcionalidades normalmente custosas de serem construídas em CPU. GPUs usam arquiteturas massivamente paralelas e possuem capacidade de cálculo superior a unidades de processamento convencionais.

Com a utilização de GPUs o processo de treinamento de DNNs pode ser bastante otimizado pois o treinamento pode ser dividido entre os núcleos da unidade de processamento, sendo reduzido em várias horas e até dias. As operações matriciais necessárias para os ajustes dos parâmetros da rede são implementadas de forma mais eficiente em GPUs.

### 2.4.3 Frameworks para deep learning

Com a popularidade no uso de aprendizagem profunda e redes de aprendizagem profundas, vários *frameworks* foram construídos ou adaptados para implementação de modelos DNN.

Um *framework* é um conjunto de ferramentas de funcionalidade genérica disponíveis para solução de um conjunto de problemas semelhantes.

*Frameworks* para métodos numéricos e de aprendizagem provêm os modelos e arquiteturas necessárias para construção da aplicação desejada. No caso de DNNs, os *frameworks* podem fornecer implementação das camadas utilizadas (como camadas recorrentes ou convolucionais), métodos de otimização e ajustes de erros, suporte para paralelismo e treino, métricas para avalição de treinamento, etc.

Os principais *frameworks* utilizados em aplicações de redes profundas são mostrados abaixo:

**Tabela 2.1 -** Descrição dos principais *frameworks* para construção de aplicações que empregam modelos DNN.

| ***Framework*** | **Responsável** | **Licença** | **Código aberto** | **Plataformas** | **Linguagem** |
|---|---|---|---|---|---|
| Caffe | Berkeley Vision and Learning Center | BSD2 | Sim | Ubuntu, OS X, Android, Windows | C++ |
| CNTK | Microsoft | MIT | Sim | Windows, Linux | C++ |
| TensorFlow | Google | Apache 2.0 | Sim | Linux, Mac OS X, Windows | Python, C++ |
| Theano | Theano Development Team | BSD | Sim | Cross-Platform | Python |
| Torch | Ronan Collobert, Clement Farabet, Koray Kavukcuoglu, Soumith Chintala | BSD | Sim | Linux, Android, Mac OS X, iOS, Windows | C, Lua |

Todos eles implementam as principais redes utilizadas, disponibilizam suporte para paralelismo e contém modelos pré-treinados para reuso.

### 2.4.5 ImageNet

O ImageNet (J. Deng *et al*., 2009) é uma base de dados de imagens, organizado em uma hierarquia de nós, onde cada nó é representado por milhares de imagens. A base do ImageNet recebe continuamente contribuições de pesquisadores em todo o mundo, com o objetivo de gerar e disponibilizar uma base de dados de grande escala de fácil acesso. O projeto é um esforço crescente para contribuição no desenvolvimento do campo de processamento de imagens e visão computacional. As imagens podem ser obtidas sem restrições autorais para pesquisadores e estudantes dentro do ambiente acadêmico (sem fins comerciais). Atualmente a base conta com um conjunto de 15 bilhões de imagens categorizadas em 1000 classes distintas.

Anualmente é realizada uma competição, a *ImageNet Large Scale Visual Recognition Competition* (ILSVRC), que tem como objetivo centralizar os avanços obtidos no desenvolvimento de modelos e técnicas para reconhecimento de objetos. Vários pesquisadores submetem seus modelos em diferentes categorias da competição, com a finalidade de obter o melhor resultado para reconhecimento de objetos nas classes da base ImageNet.

Técnicas e arquiteturas DNN precisam de uma grande quantidade de dados para aprendizado e ajuste dos modelos. Por este motivo, a grande maioria das pesquisas com redes DNN de grande escala utilizam a base de dados do ImageNet para realização do treinamento dos modelos.

Nos últimos anos os vencedores da categoria de detecção e localização de objetos têm sido propostas de arquiteturas de redes neurais profundas convolucionais

(*Deep convolutional neural network*, DCNN). Arquiteturas DCNN que tenham uma grande quantidade de exemplos disponíveis para treinamento (milhões de imagens) conseguem obter uma grande precisão para classificação de objetos de forma genérica (reconhecer e classificar a maior quantidade de objetos possíveis em uma cena).

Recentemente, essas arquiteturas DCNN podem ser reimplementadas e seus parâmetros treinados na base do ImageNet podem ser recuperados. Essa estratégia vem tornando a utilização de DCNN mais acessível, deixando de lado um grande esforço no treinamento completo destas redes.

## 2.5 Reconhecimento de logotipos

Reconhecimento de logotipos é um subproblema da área de reconhecimento de objetos. O estudo de modelos para reconhecimento de logotipos está em rápido avanço atualmente na literatura e diversas soluções e estudos vêm sendo propostos para resolução do problema.

Este é um problema chave na *web* para aplicações que fazem o reconhecimento de logotipos em anúncios e proteção de propriedade intelectual. Atualmente, com o avanço do *big data*, uma grande quantidade de informação se encontra disponível para análise. Devido a isso, se torna necessário o desenvolvimento de sistemas autônomos que recebam esta grande quantidade de informação e classifiquem a maior quantidade possível de dados de forma rápida e precisa.

Diferentes aplicações e sistemas utilizam ferramentas para reconhecimento de logotipos, como por exemplo a Ford, que utiliza um sistema baseado em SIFT e redes probabilísticas (*Probabilistic Neural Networks*, PNN) para classificação de logotipos em carros (A. Psyllos et al., 2011), a Figura 2.16 mostra a arquitetura desenvolvida para reconhecimento de logotipos de carros e a Figura 2.17 os resultados que foram obtidos.

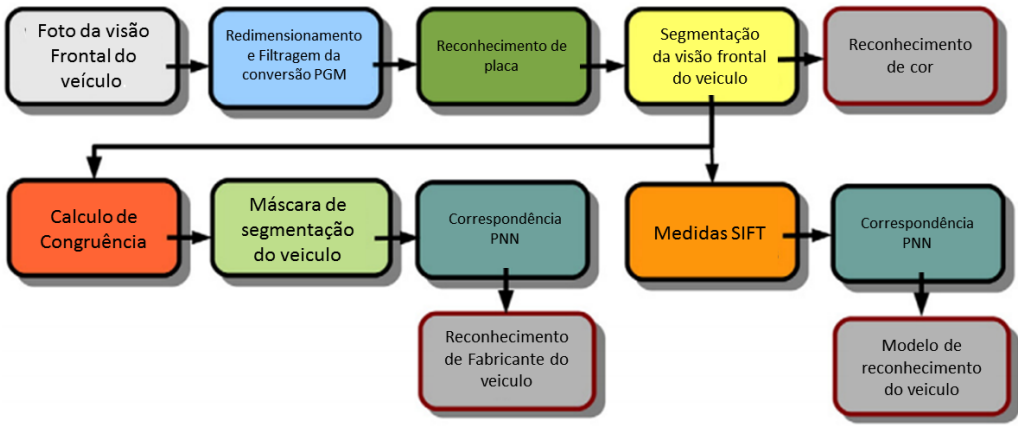


**Figura 2.16** – Sistema de reconhecimento de logotipos de carros utilizando características SIFT e redes neurais probabilísticas. Fonte: Adaptado de A. Psyllos et al., 2011.

Métodos utilizados para reconhecimento de logotipos devem ser bastante flexíveis e com alto poder de generalização. Sistemas para reconhecer imagens contendo logotipos na maioria das vezes sofrem com alguns problemas como a variação do logotipo ao longo do tempo (Figura 2.18) a escala do logotipo na imagem muito pequena em relação ao ambiente e as diferentes condições de iluminação e oclusão do logotipo na imagem.

| | Taxa de reconhecimento | | |
|---|---|---|---|
| Fabricante | Corretos | Errados | Não reconhecidos |
| Alfa Romeo | 9 | 0 | 1 |
| Audi | 7 | 3 | 0 |
| Bmw | 10 | 0 | 0 |
| Citroen | 10 | 0 | 0 |
| Fiat | 9 | 1 | 0 |
| Peugeot | 7 | 2 | 1 |
| Renault | 7 | 2 | 1 |
| Seat | 9 | 0 | 1 |
| Toyota | 8 | 1 | 1 |
| Volkswagen | 9 | 1 | 0 |
| Unknown/other | 8 | 2 | 0 |
| Total | 93 [85%] | 12 [11%] | 5 [4%] |

**Figura 2.17** – Resultados de reconhecimento de logotipos em carros com arquitetura de classificação baseada em PNN. Fonte: Adaptado de A. Pysllos et al., 2011.


**Figura 2.18** - Exemplo de variação de logotipos ao longo dos anos. Fonte: I. Chih, 2011.

Devido a tais problemas, técnicas mais robustas são necessárias. Atualmente estudos indicam que o uso de redes profundas pode gerar um bom resultado na classificação destas imagens (F. Landola et al., 2015). Com a possibilidade de se obter arquiteturas pré-treinadas sem qualquer restrição a utilização destes modelos, eles se tornam mais acessíveis, deixando de lado um grande esforço no treinamento de uma grande quantidade de parâmetros e a necessidade de se ter uma base de dados de larga escala.

F.Landola e colegas mostram que os principais problemas de reconhecimento de logotipos podem ser classificados genericamente da seguinte forma:

**Classificação de logotipo:** O interesse é apenas em saber se existe um logotipo específico na imagem, não interessando sua localização ou a quantidade existente.

**Detecção de logotipo:** Se existe um logotipo na imagem, o interesse é apenas em saber quantos tipos diferentes de logotipo existem na imagem e quais são eles.

**Localização de logotipos:** Neste tipo de problema o objetivo é saber onde está localizado o logotipo na imagem (todas as ocorrências do logotipo na imagem), não interessando qual tipo está presente. Uma aplicação deste tipo de problema é a identificação e remoção de logotipos transparentes em programas de TV, como feito em (S. Duffner; C. Garcia, 2006).

**Detecção e localização de logotipos:** Se existe um logotipo na imagem, o objetivo é saber qual tipo e onde ele está localizado na imagem. Neste tipo de abordagem, se considera todas as ocorrências do logotipo na imagem observada.

Este trabalho é focado em resolver os problemas de classificação de logotipos e localização de logotipos.

# 03. Trabalhos Relacionados

Atualmente o método de representação de características *Scale Invariant Feature Transform* (SIFT) (D. Lowe, 1999), utilizado em conjunto com a técnica de *Bag-of-Words* (F. Li; P. Perona, 2005), vem sendo a principal técnica utilizada na construção de aplicações para reconhecimento de logotipos. Com esta técnica, as características obtidas com o SIFT são quantificadas em um vocabulário de imagens, para realização das correspondências entre os descritores dos objetos observados na imagem e os descritores armazenados no sistema.

O SIFT produz um novo espaço de imagens escalonadas e suavizadas em níveis de cinza, encontrando os pontos de interesse através de máximos locais utilizando uma aproximação da função laplaciano da Gaussiana. Escolhendo a região ao redor do ponto de interesse, é criado um histograma de gradientes. Cada descritor consiste de um vetor de histogramas. O uso de um conjunto de gradientes ao redor da região de interesse possibilita a obtenção de invariância à rotação e escala na detecção.

Variações destas técnicas são extensivamente propostas na literatura para melhorar os resultados de classificação e o desempenho computacional do sistema, como em (R. Boia et al., 2014), que aplica *Complete Rank Transform* (CRT) com o SIFT+*Bag-of-Words* para obter invariância à iluminação e mudanças de intensidade.

Em (A. Psyllos et al., 2012) é proposto um algoritmo para reconhecimento de logotipo em veículos, utilizando uma variação do SIFT, chamada de *Multi-Spectral Scale Invariant Features Transform* (M-SIFT). Com esta técnica é possível melhorar a taxa de classificações positivas corretas, diminuindo o tempo de processamento do algoritmo SIFT tradicional. No M-SIFT os descritores obtidos são baseados nos valores dos vizinhos mais próximos (*Nearest Neighbors*); os pontos obtidos são armazenados utilizando a transformação de Hough, ajustada posteriormente com o método RANSAC (*Random Sample Consensus*).

Como visto no capítulo anterior, o reconhecimento de logotipos pode ser visto como um subcaso do problema de reconhecimento de objetos em imagens. Técnicas de DNN estão se tornando cada vez mais acessíveis com a construção de novas arquiteturas DCNN que requerem menos requisitos de *hardware*, e com a possibilidade de recuperação de modelos pré-treinados.

Em 2012, o vencedor do ILSVRC foi a DCNN AlexNet, baseada na arquitetura proposta em (A. Krizhevsky et al., 2012). A rede contém cerca de 60 milhões de parâmetros treináveis e um conjunto de 650 mil neurônios, divididos entre cinco camadas convolucionais seguidas de camadas de *max-pooling* e três camadas totalmente conectadas com saída de 1000 neurônios para classificação de cada classe presente no ImageNet (Figura 3.1).

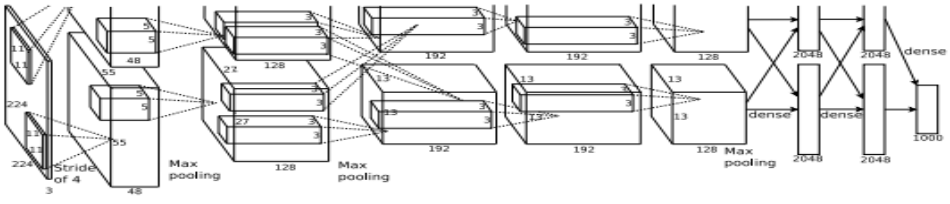


**Figura 3.1** – Arquitetura do AlexNet vencedora do ILSVRC 2012, baseada na proposta publicada em (A. Krizhevsky et al., 2012). Fonte: A. Krizhevsky et al., 2012.

A rede utiliza *data augmentation* para ajustes dos dados de treinamento com o objetivo de obter invariância a escala, rotação, translação e intensidade de *pixels*. O algoritmo de ajuste utilizado para evitar *overfitting* das redes é o *dropout*. A rede tem como principal característica o uso da função de ativação retificadora utilizando neurônios ReLU e, realizando normalização local de cada resposta, possibilitando o uso de diferentes *kernels* nos neurônios para computar os dados. A rede é capaz de realizar a operação de *pooling* tradicional ou *pooling* com sobreposição, e o uso da sobreposição é capaz de melhorar a taxa de erros em até 0,4%.

O modelo vencedor do ILSVRC 2015 foi o GoogleNet (C. Szegedy et al., 2015). A arquitetura do GoogleNet é composta por 22 camadas, entre camadas convolucionais, *max pooling*, *softmax*, *average pool* e concatenação de filtros (Figura 3.2). A principal ideia da rede é utilizar não apenas filtros de tamanho fixo divididos de forma padrão, mas também aprender além dos parâmetros, os filtros utilizados para extração das características que podem identificar objetos em diferentes escalas. Nesta rede são definidas novas camadas, chamadas de camadas de *inception*, onde a rede aprende e define o filtro utilizado na rede. Isto é feito empregando diversos filtros de tamanhos diferentes que recebem a entrada da camada anterior e processam o resultado que será concatenado com a resposta dos outros filtros ao fim da operação (Figura 3.3). O modelo utiliza ajuste de *overfitting* com *dropout* e utiliza funções de classificação linear e *softmax*.

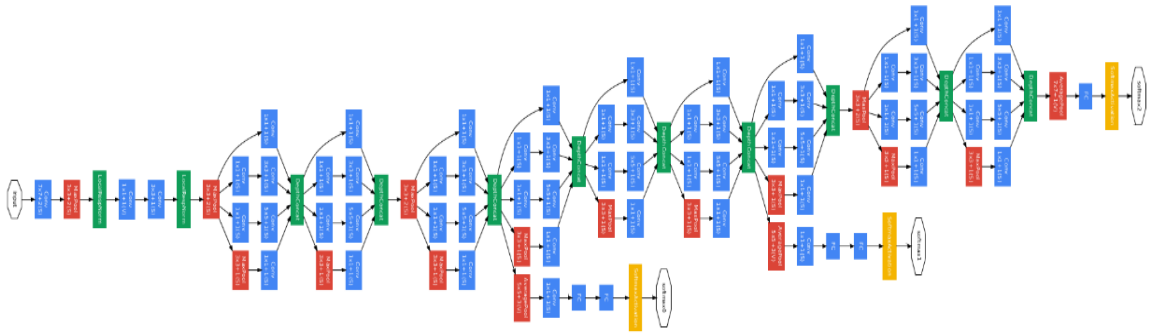

**Figura 3.2** - Arquitetura da rede GoogleNet vencedora do ILSVRC 2015. Fonte: C. Szegedy et al., 2015.

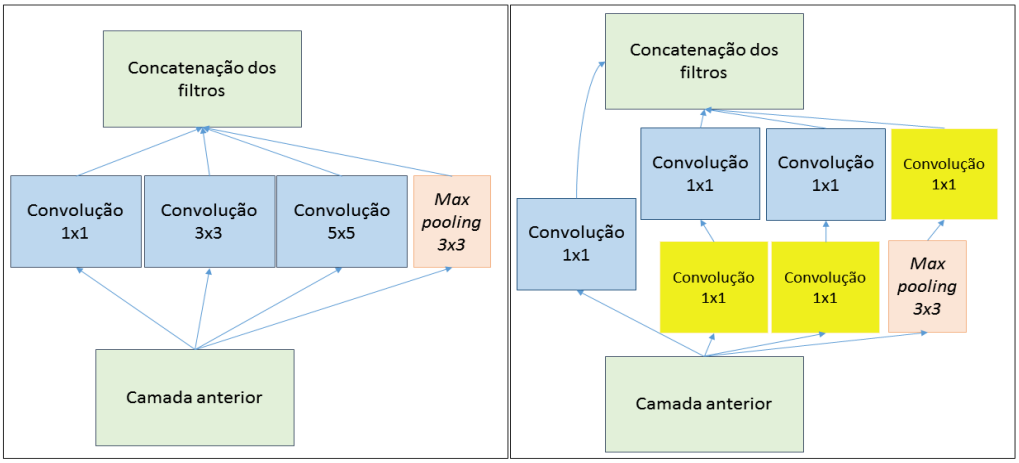


**Figura 3.3** - Camadas de *inception* da rede GoogleNet. A camada de inception utiliza diferentes tipos de camadas convolucionais, a fim de identificar características das imagens em diferentes escalas. Fonte: Adaptado de C. Szegedy et al., 2015.

Novos modelos baseados nas redes submetidas no ILSVRC vêm sendo propostas para resolução de problemas em algum domínio específico. A ideia destes modelos é utilizar os parâmetros da rede DCNN já treinado na base de dados de larga escala do ImageNet e aprimorar o treinamento utilizando uma base de dados específica menor, atualizando os parâmetros da rede apenas nas últimas camadas de classificação. Desta forma, a rede mantém os parâmetros treinados nas primeiras camadas, onde extrai características mais genéricas dos dados, e sofre mudanças nas últimas camadas onde faz uma separação mais restrita dos dados, diminuindo a quantidade de parâmetros a serem treinados e mantendo a flexibilidade de classificação para diversos tipos de categorias.

Seguindo esta ideia, um novo conjunto de modelos de DCNN para reconhecimento de logotipos foi proposto em (F. Landola et al., 2015). Os modelos são baseados na arquitetura descrita em (C. Szegedy et al., 2015), implementada como GoogleNet. Três variações são propostas onde a maior parte da arquitetura original é preservada, com algumas adições de operações extras entre algumas camadas da rede.

Na primeira variação, Googlenet-GP, é adicionada a operação de *pooling* antes de cada camada de classificação totalmente conectada (*fully connected layer*). A intenção desta operação é permitir a rede processar imagens de qualquer tamanho de entrada, mantendo a taxa de classificação.

A segunda variação, Googlenet-FullyConnected, visa diminuir o erro de treinamento do gradiente descendente na camada de classificação, inserindo uma camada de classificação com função *softmax* após cada camada de *inception* da rede.

A terceira variação, *Full-Inception*, propõe melhorar o desempenho de classificação da rede com imagens contendo logotipos de baixa resolução, utilizando uma arquitetura com todas as camadas da rede, exceto a de saída, sendo camadas de *inception*. A ideia é utilizar diversas variações de filtros na entrada para extrair o máximo de características possíveis para melhorar o reconhecimento de logotipos em qualquer tipo de imagem.

A rede foi treinada na base de dados ImageNet e posteriormente realizado o treinamento na base Flickrlogos-32, que contém 100 imagens para 32 classes de logotipos diferentes. A arquitetura proposta foi utilizada para resolução do problema de classificação de logotipos e obteve uma taxa de acertos de 89% na base de dados analisada. A arquitetura falha na maioria das vezes para o problema de variação de logotipo e para logotipos muito pequenos em relação ao ambiente na imagem.

Em (S. Hoi et al., 2015) é proposto um modelo de classificação e localização de logotipos em imagens baseado em modelos *Rapid Convolutional neural networks* (R-CNNs) (R. Girshick et al., 2014). A variação proposta é denominada *Deep Logo Detection Region-Based Convolutional Networks* (DRCN) e utiliza uma implementação de um algoritmo de segmentação, *Selective search* (SS) (K. Van de Sanden et al., 2011), para encontrar objetos dentro da imagem, limitando a região que precisa ser analisada pela rede. O diferencial da arquitetura é a utilização de dois tipos de camada *fully-connected* na última etapa de classificação, uma camada de regressão que retorna o valor da crença de que há um logotipo desejado dentro da imagem e uma camada que retorna as coordenadas do *bounding box* que delimita este logotipo se ele estiver presente (Figura 3.4).

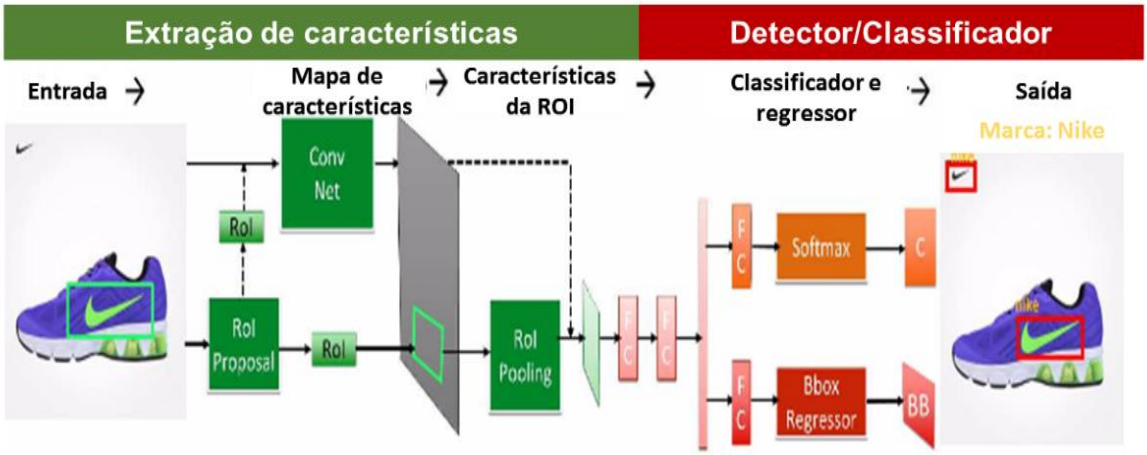


**Figura 3.4** - Arquitetura DRCN para classificação e localização de logotipos. A extração da região de interesse é realizada com a utilização do método de segmentação SS; o modelo retorna o nível de confiança da presença do logotipo e sua localização na imagem. Fonte: Adaptado de S. Hoi et al., 2015.

Uma das maiores dificuldades em reconhecimento de logotipos está na grande quantidade de variação que um logotipo pode ter ao longo do tempo, como visto na (Figura 2.17). Logotipos podem sofrer alterações como variações do estilo da fonte, *design* da logo, oclusões na imagem, mudança de escala, etc.

O sistema de reconhecimento de logotipos deve ser robusto a tais variações, conseguindo classificar os logotipos das imagens independente destas condições. O sistema para reconhecimento de logotipo deve ser capaz de realizar generalização no espaço de classificação e deve ter um sistema de extração de características e classificação robusto o suficiente para identificar o logotipo em variadas situações.

Em (P. Sermanet et al., 2014) é proposto um modelo multi-escala com deslizamento de janela para reconhecimento de objetos que visa melhorar o desempenho de reconhecimento mantendo a eficiência durante a execução do modelo.

A rede que implementa o modelo é chamada de *overfeat*, vencedora do ILSVRC 2013. É mostrado que diferentemente de outras técnicas onde a implementação da janela deslizante geralmente é um limitante para o sistema, o uso deste método em redes neurais convolucionais pode ser feito de forma eficiente sem causar grande impacto no desempenho do sistema.

As redes convolucionais compartilham naturalmente a computação dos dados em regiões sobrepostas. A resposta da operação de convolução é um mapeamento espacial de cada classe detectada em cada região da imagem. Quando aplicadas imagens de tamanhos diferentes, a computação extra necessária para fornecer uma resposta é limitada ao quão diferente é o novo tamanho, como visto na (Figura 3.5).

A arquitetura de extração de características do *overfeat* é similar à utilizada pela rede AlexNet, com a diferença de que não há normalização de contraste, as operações de *pooling* não são sobrepostas e a arquitetura do *overfeat* tem camadas de mapeamento de características maiores.

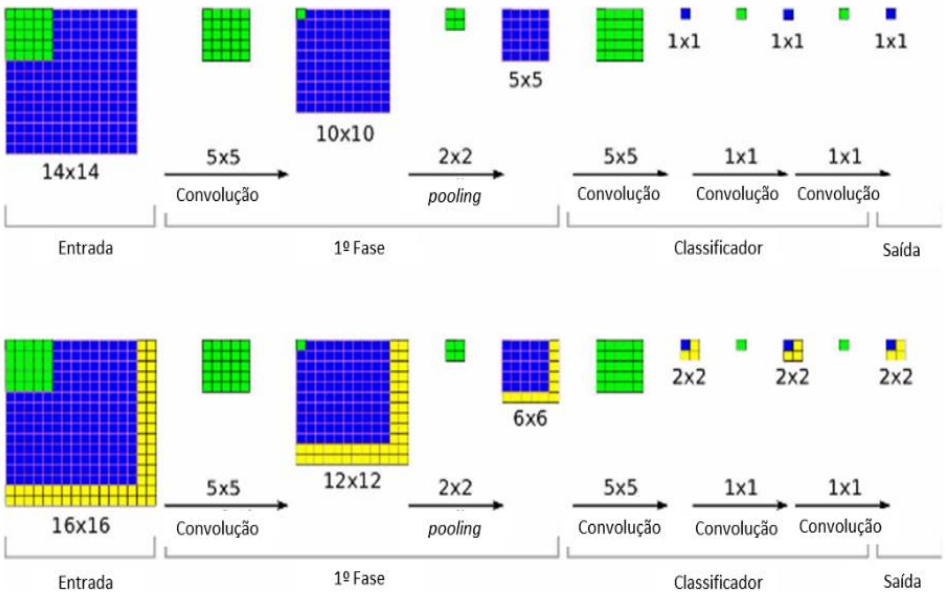


**Figura 3.5** – Operação de convolução aplicada em redes neurais convolucionais para classificação de diferentes regiões da imagem. A convolução digital em imagens segue o mesmo principio da operação de deslizamento de janelas para gerar uma representação intermediaria dos dados. A técnica de deslizamento para avaliar cada região da imagem pode ser adicionada sem custo adicional, pois a computação extra requerida é limitada somente ao tamanho extra da imagem em relação ao tamanho padrão utilizado no treinamento da rede neural. Fonte: Adaptado de P. Sermanet et al., 2014.

# 04. Modelo de Reconhecimento de Logotipos

Com base nas observações dos resultados em trabalhos relacionados ao problema de reconhecimento de objetos e reconhecimento de logotipos e, devido ao recente esforço de pesquisadores no desenvolvimento de novas arquiteturas de Redes Neurais Profundas (*Deep Neural Networks*, DNNs) para reconhecimento e detecção de objetos em larga escala, que vêm atingindo um alto grau de precisão em classificação e detecção de objetos, o modelo escolhido para ser construído durante o desenvolvimento deste trabalho foi um modelo de Rede Neural Convolucional Profunda (*Deep Convolutional Neural Network*, DCNN).

Para alcançar o objetivo principal deste trabalho, o modelo utilizado será inspirado em uma das arquiteturas propostas por F. Landola e colegas, em que a rede DCNN GoogleNet (C. Szegedy et al., 2015) é adaptada para resolução do problema de reconhecimento de logotipos. Neste trabalho a arquitetura proposta em (F. Landola et al., 2015) será implementada com algumas adaptações. Além da classificação já realizada, a rede será adaptada com um novo passo para localização baseado na proposta de arquitetura vista em (C. Szegedy et al., 2013) utilizando a mesma estrutura de resposta da arquitetura DRCN proposta em (S. Hoi et al., 2015). O modelo deve ser capaz de resolver o problema de classificação e localização de logotipos em imagens em grande escala de informações, ou seja, o modelo deve ser preciso e rápido, realizando uma grande quantidade de classificações em diversas imagens com baixa taxa de erros e no menor intervalo de tempo possível de avaliação.

Ao final do desenvolvimento é esperado obter uma um modelo que obtenha uma taxa de precisão semelhante a proposta das arquiteturas em (F. Landola et al., 2015). Também será realizada uma avaliação da influência do uso de diferentes parâmetros de treinamento e métodos de ajustes no desempenho de classificação do modelo, como o tempo de treinamento necessário para o modelo, a influência de diferentes taxas de aprendizagem e algoritmos de treinamento, a precisão obtida pelo modelo e o tempo de classificação do modelo para os exemplos de teste.

O modelo desenvolvido utiliza a ideia de composição de arquiteturas e reutilização de um modelo de grande escala pré-treinado na base de dados do ImageNet apresentado em (F. Landola et al., 2015). A arquitetura pré-treinada utilizada é baseada na proposta descrita em (C. Szeged et al., 2015), onde a rede construída com base nesta arquitetura (GoogleNet) obteve o melhor resultado em relação a outros modelos DNNs submetidos na competição ILSVRC. Como visto em (F. Landola et al., 2015), a rede GoogleNet consegue ter um bom desempenho de classificação quando realizado o pós-treino em uma base de dados específica. Também é mostrado que é possível melhorar a precisão da arquitetura original com a aplicação de algumas operações e estruturas extras sem grandes interferências no custo computacional do modelo.

## 4.1 Descrição do modelo

O modelo construído pode ser visto como uma composição de módulos. Cada módulo constitui uma funcionalidade existente para um sistema de reconhecimento como vista na seção 2.1.2. Desconsiderando a etapa de aquisição da imagem, o sistema aplica as etapas de pré-processamento dos dados de entrada, a extração das características dos objetos alvos (logotipos) e a interpretação e classificação destas características (Figura 4.1).

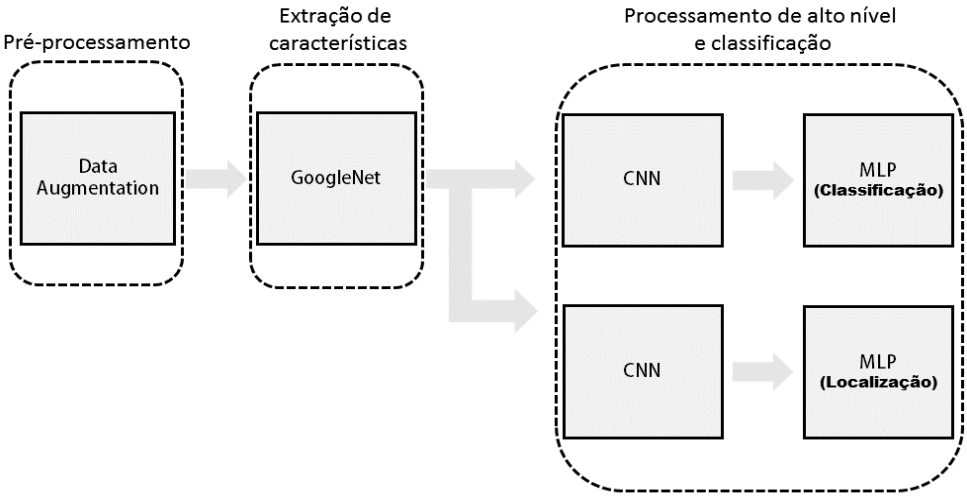


**Figura 4.1** – Representação esquemática do modelo de reconhecimento de logotipos construído. Como visto na seção 2.1.2, todo sistema de reconhecimento de imagens deve implementar os passos de pré-processamento, extração de características e classificação. A figura mostra o mapeamento de capa uma destas etapas com as estruturas existentes no modelo proposto. Fonte: o autor.

### 4.1.1 Pré-processamento

O primeiro módulo representa a etapa de pré-processamento. Como visto anteriormente, o pré-processamento dos dados é utilizado para modificar o conjunto de exemplos de forma a melhorar os resultados finais ou padronizar o conjunto de teste para utilização do modelo. Estas modificações podem ser operações básicas de manipulação de imagens (translação, rotação, mudança de escala), adição de ruídos, redimensionamento dos dados, entre outros.

No modelo, cada imagem de exemplo é redimensionada para o tamanho fixo de 256x256 através da utilização de interpolação bi cúbica e, devido à alta variação de tamanho entre as imagens de exemplos, essa transformação é necessária para aplicação da imagem no treino do modelo da rede neural que possui tamanho de entrada fixo. Em seguida, o processo de *data augmentation* é aplicado. Neste processo o conjunto de treinamento é expandido com novos exemplos derivados do conjunto de exemplos original.

Durante a realização do *data augmentation*, cada imagem recebe uma série de transformações, como rotação, translação e escala. Com isto, o conjunto de treino é ampliado, contendo novas situações e variações da ocorrência do logotipo.

Em (A. Krizhevsky et al., 2012) foi mostrado que a utilização de *data augmentation* auxilia no controle de *overfitting* de modelos DNNs. A aplicação da operação de translação e rotação na imagem é suficiente para diminuir o grau de especialização da rede durante o treinamento do modelo.

Sabendo que no reconhecimento de logotipos os problemas de variação são recorrentes (seção 2.5), neste modelo também são aplicadas as operações de rotação da imagem e alteração da intensidade de cores da imagem. Outra operação utilizada é a adição de ruído aleatório.

### 4.1.2 Extração de características

O segundo módulo do modelo consiste da etapa de extração de características. A arquitetura da rede GoogleNet (Figura 3.2) foi a arquitetura escolhida para realização desta fase, esta arquitetura é capaz de definir e retornar um conjunto de características extraídas em várias regiões da imagem, estas características vão ser utilizadas posteriormente em um modelo de classificação, de acordo com o objetivo desejado (R. Stewart; M. Andriluka, 2015). Como visto anteriormente, esta arquitetura foi escolhida por conter o melhor resultado atualmente no ILSVRC, além de conter 15 vezes menos parâmetros treináveis em sua estrutura (C. Szegedy et al., 2015) comparado aos outros modelos DCNN, sendo uma arquitetura de uso acessível para treinamento em *hardwares* menos robustos.

A rede original GoogleNet é inicialmente treinada no conjunto de dados da base do ImageNet, compondo uma rede de detecção de objetos capaz de classificar um conjunto de 1000 classes distintas.

As primeiras camadas convolucionais pertencentes à rede são utilizadas para geração de uma representação intermediaria dos dados de entrada que tende a extrair características mais genéricas da imagem (regiões de cores, objetos, bordas). Cada representação gerada por uma camada é passada adiante e interpretada pela camada seguinte. A cada camada a rede tende a se especializar em extrair características mais específicas e representativas dos dados. A função das camadas convolucionais é focada na geração dos filtros extratores de característica; os valores que definem o filtro e sua operação são os parâmetros a serem treinados nas camadas convolucionais.

Durante o treino do modelo, esta arquitetura passa por mais uma etapa de treinamento, o pós-treino. Nesta etapa o treinamento é realizado novamente, dessa vez com o conjunto de treino específico e com a inserção da camada de classificação que será utilizada no modelo. O pós-treino é necessário para atualizar os valores dos filtros de características em cada camada, aprendendo quais características são mais relevantes para a resolução do problema. Muitas vezes, é suficiente atualizar apenas as últimas camadas convolucionais da rede (camadas de extração das características especificas). Durante este processo, as camadas de perdas utilizadas na arquitetura original do GoogleNet são descartadas, inserindo novas camadas que serão treinadas apenas na segunda fase do treinado do modelo.

Ao final do processo a rede retorna como resultado um vetor de características extraídas da imagem. O vetor de características corresponde a um conjunto de regiões na imagem uniformemente divididas; cada região contém um conjunto de características extraídas que serão retornadas de acordo com o mapa de ativação especificados nos filtros de treinados.

### 4.1.3 Processamento de alto nível e classificação

Uma rede de classificação é utilizada para aprendizado de quais respostas deve gerar para cada uma das características observadas.

Neste trabalho a estrutura de classificação contém duas ramificações, como construído na proposta de S. Hoi e colegas (S. Hoi et al., 2015), o primeiro ramo tem como objetivo a classificação das imagens e o outro a localização. Esta estrutura representa o ultimo modulo descrito na figura 4.1.

A primeira rede é mostrada na figura 4.2, representando a estrutura que realiza o processamento de alto-nível das características e retorna à predição da presença de logotipo na imagem, ou seja, a rede que realiza a classificação dos logotipos.

Para interpretar as características retornadas pela rede GoogleNet, a arquitetura de classificação utilizada foi baseada no trabalho de (F. Stark et al., 2015) e é representada por um conjunto de camadas convolucionais com ativação ReLU, seguidas por duas camadas fully-connected utilizando unidades de ativação linear e uma camada com ativação *softmax*.

Essas camadas vão aprender como responder a cada conjunto de características recebido.

A última camada de neurônios é composta por dois neurônios que correspondem à resposta de classificação da rede. Idealmente a resposta da rede para exemplos negativos deve ser 1 para o primeiro neurônio e 0 para o segundo, e para exemplos positivos deve ser 0 para o primeiro neurônio e 1 para o segundo. Na prática um certo nível de incerteza é adicionado na resposta final da rede; caso a rede faça a predição da resposta com alto nível de certeza, o valor da resposta deve se aproximar do valor ideal. Como o nível de incerteza está associada a maioria dos exemplos não triviais existentes, a abordagem de escolha da resposta é dada pelo neurônio de maior ativação com a utilização, função *softmax*, sendo a resposta dada pelo $max(n1, n2)$. Se o neurônio 1 $(n1)$ tiver a maior resposta de ativação o exemplo é considerado negativo, e se caso o valor máximo corresponder ao neurônio 2 $(n2)$, então o exemplo é considerado positivo.

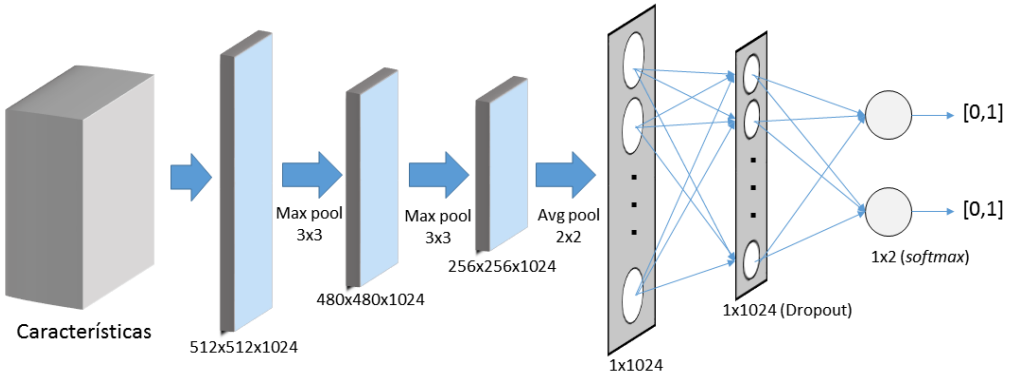


**Figura 4.2** – Arquitetura da rede neural para classificação das características extraídas da imagem de entrada. Fonte: o autor.

A segunda rede é focada na resolução do problema de localização, baseada no trabalho publicado em (Szegedy et al., 2013). A rede implementa uma abordagem simples, utilizando camadas fully-connected. Esta rede tem a mesma dimensão do vetor de características retornado pelo GoogleNet. Cada neurônio da rede representa uma região da imagem, cada região tem seu conjunto de características extraídas. A rede então aprende a identificar as características desejadas e, para cada região será verificado se as características correspondem ao valor das características do logotipo. Em caso positivo, o neurônio de saída correspondente aquela região é ativada. O conjunto de neurônios vizinhos ativados na camada de saída da rede representa a região que delimita o logotipo.

O treinamento é realizado através da comparação da saída da rede com o vetor da resposta real. O vetor da resposta real é construído a partir da informação das posições do logotipo na imagem, esta informação está disponível na base de dados utilizada no treinamento. Cada logotipo tem uma região delimitadora retangular dentro da imagem, essa região retangular é descrita pelo ponto inicial do retângulo na imagem (*pixel* $x, y$), a largura $w$ e a altura $h$. Esses parâmetros são então mapeados para 4 pontos (os 4 pontos que formam a região retangular).

A imagem é então indexada em regiões, estas regiões correspondem ao mesmo conjunto de regiões que é retornado pela rede GoogleNet. Com isso, é possível fazer a correspondência destes pontos dentro de cada região com o vetor de resposta. Se o ponto está contido dentro de uma certa região o valor relativo a ela será 1, enquanto os outros serão atribuídos com 0. Cada posição deste vetor será mapeada em um neurônio de saída.

As duas redes descritas funcionam de forma independente, cada uma contém um objetivo diferente na arquitetura. O modelo implementa ambos os casos, sendo possível também realizar apenas classificação ou localização das imagens.

O treinamento do modelo é supervisionado, ou seja, cada exemplo da base de dados possui um conjunto de respostas que representam a resposta ideal que o sistema deve fornecer. Desta forma, é possível realizar o treinamento através de algoritmos de aprendizagem que ajustam os pesos dos neurônios através do valor do erro de predição do modelo, como por exemplo, o Gradiente Descendente.

## 4.2 Implementação

A ferramenta utilizada para a implementação do modelo de reconhecimento de logotipos foi o *tensorflow (www.tensorflow.org)*. O *tensorflow* é um *framework* de aprendizagem de máquina e *deep learning*. Ele contém uma biblioteca de código aberto desenvolvida por pesquisadores e engenheiros do *Google Brain*. A API está disponível para sistemas Linux e Mac OS X e pode ser acessada através de uma interface em Python. A execução de uma aplicação com a API do *tensorflow* pode ser realizada com processamento em GPU ou CPU, podendo o código ser utilizado para os tipos de unidade de processamento sem grandes necessidades de adaptação; o *tensorflow* realiza as otimizações e adaptações automaticamente.

Um dos principais motivos para a escolha do *tensorflow* no desenvolvimento do modelo é a representação utilizada por aplicações desta API. Uma aplicação

*tensorflow* pode ser construída através de um grafo que representa as operações matemáticas do modelo. A representação em grafo utilizada é denominada como um *dataflow*.

No *dataflow*, cada operação matemática é representada por um nó do grafo de computação. As relações entre as operações são representadas pelas arestas que conectam cada nó, e cada aresta possui um conjunto de parâmetros que podem ser armazenados também em nós e que serão propagados como informação útil para outros nós a frente.

Percebe-se que essa representação é análoga a forma de representação utilizada em modelos de redes neurais, onde cada nó do grafo pode ser visto como um neurônio e suas arestas as conexões entre cada neurônio com seus respectivos pesos.

O *tensorflow* também disponibiliza os principais métodos de ajustes e treinamento, camadas de redes neurais implementadas e ferramentas de controle de treinamento e avaliação de modelos.

A aplicação construída para execução do modelo é dividida em duas partes. A primeira parte é relativa ao carregamento e treinamento do modelo. A segunda parte é a avaliação do modelo, que utiliza os parâmetros treinados da rede para realizar o reconhecimento dos logotipos.

O treinamento do modelo é realizado de acordo com o pseudocódigo abaixo.

**1.** $FUNCTION\ Train(examples, par)$
**2.** $examples \leftarrow DataAugmentation(examples)$
**3.** $confidences \leftarrow GetConfidencesValues(examples)$
**4.** $SET\ batches\ to\ EMPTY\ LIST$
**5.** $FOR\ i = 1\ to\ par.num\_batches$
**6.** $batch \leftarrow CreateBatch(examples, confidences)$
**7.** $batches.append(batch)$
**8.** $googlenet \leftarrow LoadFeaturesExtractor(par.weightslocation)$
**9.** $classifier \leftarrow LoadClassifier(par.num\_classes, par.activation\_function)$
**10.** $learning\_algorithm \leftarrow tensorflow.optimizer(par.train\_name, par.learning\_rate, par.momentum, par.weights\_decay)$
**11.** $model \leftarrow tensorflow.Graph(googlenet, classifier, learning\_algorithm)$
**12.** $FOR\ i = 1\ to\ par.num\_itermax\ or\ converge$
**13.** $erros, converge \leftarrow fowardpass(model)$
**14.** $model \leftarrow adjustbackwardpass(errors, learning\_algorithm, confidences)$
**15.** $tensorflow.SaveModel(model)$

A função $Train$ é responsável por gerar e armazenar os parâmetros treinados do modelo. Como o treinamento do modelo é realizado através do aprendizado supervisionado, são necessários um conjunto de imagens de treino $(examples)$ e as respostas reais de cada exemplo e que se espera obter através do modelo. As respostas são definidas por um conjunto de regiões que delimitam os logotipos na imagem (*boundingbox*) e o valor de confiança que define a presença ou não do logotipo.

Também devem ser fornecidos os parâmetros de treinamento que serão utilizados como a taxa de aprendizagem, o tamanho do batch de treinamento, o número de neurônios na camada de classificação, a função de ativação utilizada e o algoritmo de aprendizagem utilizado. Estes parâmetros são representados no

pseudocódigo através da estrutura $par$, que deve armazenar todos os valores mencionados.

O conjunto de imagens deve conter imagens com logotipos (exemplos positivos) e imagens que não contenham o logotipo (exemplos negativos), com cada imagem rotulada adequadamente.

O primeiro passo realizado é a aplicação do *data augmentation* no conjunto de treinamento. A função $DataAugmentation$ recebe como entrada o conjunto de treinamento original e retorna um novo conjunto de treino aplicando o procedimento explicado na seção anterior. As operações de rotação, intensidade de cor e adição de ruído são aplicadas nos exemplos de treino.

Após isso, o vetor de valores de confiança é gerado. Esse vetor representa a resposta que o modelo deve obter para cada exemplo de treino (1 para imagem com o logotipo, 0 para a imagem sem logotipo). Na sessão anterior foi visto que o GoogleNet retorna um conjunto de características para cada região da imagem. Então, cada uma destas regiões deve conter um valor de confiança indicando ou não a presença do logotipo. Para gerar o vetor de confiança da imagem, devemos dividir a imagem em uma grade de tamanho fixo. Este tamanho é definido pelo volume de saída que o GoogleNet irá retornar, para cada região (posição da grade) verificamos se esta posição corresponde ao valor conhecido da posição do logotipo, comparando se a posição está dentro da região que delimita o logotipo. Se a posição estiver contida o valor de confiança é definido como 1, senão o valor é definido como 0.

O próximo passo é carregar a rede GoogleNet, para isto é necessário carregar a rede camada por camada, gerando ao final do processo o grafo do *tensorflow* que processará as etapas da rede neural. O *tensorflow* fornece implementações para as camadas convolucionais, camadas de *pooling*, camadas de perda e camadas totalmente conectadas. Durante essa construção, os pesos da arquitetura são carregados atribuindo para cada nó da rede os valores dos pesos pré-treinados.

O último modulo de construção da rede neural é definir a estrutura de classificação. Atribuímos cada camada de classificação a arquitetura do GoogleNet, utilizando as camadas da API do *tensorflow* e descartando as camadas de perda da arquitetura pré-treinada original, os pesos desta camada são então reiniciados aleatoriamente.

Como mencionado na seção 2.3.4, é importante definir um método de ajuste e regularização. O *tensorflow* disponibiliza uma camada de ajuste com *dropout*, inserimos esta camada antes das camadas totalmente conectadas, definindo um método de ajuste de *overfitting* para o modelo.

Definimos por fim o algoritmo de aprendizagem que será utilizado no modelo, assim como os parâmetros de treinamento. O algoritmo de aprendizagem é disponibilizado pelo *tensorflow*, que realiza o ajuste dos parâmetros automaticamente durante o treinamento.

Com a arquitetura da rede construída e os parâmetros de treino definidos, podemos gerar o grafo que realizará a execução do treinamento. O treinamento segue o fluxo do aprendizado supervisionado de uma rede neural, como visto na seção 2.3.1, contendo em cada iteração de treino o passo de predição *feed-foward* e passo de correção *backward*. No primeiro passo, representado pela função $fowardpass$ no pseudocódigo, os pesos atuais da rede são testados e geram uma resposta para um conjunto de exemplos. Estas respostas geradas são comparadas com as respostas conhecidas de cada exemplo e o erro entre elas é computado. Com base neste erro os valores dos pesos da rede neurais são ajustados, aproximando a resposta da rede neural a resposta conhecida.

O processo de treinamento é repetido em várias iterações, até a rede neural atingir o valor máximo de iterações permitidas ou convergir dentro do valor esperado.

O treinamento do modelo é realizado em *batches*, ou seja, em cada passo um conjunto de exemplos é selecionado. Esse conjunto é utilizado para computar o erro acumulado da rede neural para cada um dos exemplos pertencentes ao conjunto. O conjunto de erros é então utilizado para computar o erro médio que será utilizado para atualizar os pesos da rede neural.

Durante o treinamento o *tensorflow* fornece estatísticas do estado atual da rede neural, como erro do conjunto de treino e validação, acurácia do modelo, tempo de progresso para cada passo e variação dos pesos que utilizaremos para realizar algumas das medições de avaliação do modelo.

# 05. Avaliação do Modelo

Neste capítulo serão analisados os resultados de avaliação do modelo construído. Será mostrado o processo de construção da base de dados utilizada no treinamento, o conjunto de métricas utilizadas para avaliação, uma descrição da aplicação construída para comparação dos resultados e, por fim, os resultados obtidos pelo modelo desenvolvido neste trabalho.

Para análise dos resultados finais, o problema de reconhecimento de logotipos foi restringido a um caso base onde, analisamos a presença de um logotipo específico, sendo este o logotipo da Coca-Cola.

## 5.1 Base de dados

A base de dados mais utilizada na literatura para reconhecimento de logotipos é a flickrlogos-32 (S. Romberg et al., 2011), esta base contém 32 classes de logotipos distintos, sendo a maior quantidade categorizada entre as bases disponíveis atualmente (F. Landola et al., 2015). Cada classe contém um conjunto de imagens de exemplo onde cada imagem é rotulada com um conjunto de áreas retangulares (*bounding box*) delimitando a região onde se encontra o logotipo na imagem. Cada área é definida através de 4 valores, que são a posição inicial do retângulo que representa a área na imagem e a largura e altura desta área $(x, y, w, h)$. A base flickrlogos-32 contém exemplos com casos controlados de ocorrência dos logotipos (Figura 5.1), sendo necessária a construção de uma nova base de dados com exemplos mais complexos para treinamento do modelo, para melhorar o grau de especialização do modelo, sendo capaz de lidar com situações realísticas de casos presentes em aplicações comerciais.

Por este motivo criamos uma nova base de dados com imagens da Coca-Cola durante o desenvolvimento deste trabalho, extraindo um conjunto de 2899 imagens do Instagram com domínio da Coca-Cola e rotulando cada um deste para o problema. A nova base foi construída para prover um conjunto de treinamento mais robusto com uma maior quantidade de exemplos mais complexos (Figura 5.2) para realizar a especialização do modelo, isto se deve à falta de disponibilidade de base de dados que contenham uma quantidade significativa de exemplos para treinamento de uma arquitetura DCNN.

Como visto no capítulo anterior, a fase de treinamento conta com dois processos de treinamento, o primeiro realizado com a base do ImageNet para a arquitetura DCNN de extração de características e o segundo com uma base de dados especifica para o problema em questão.Para a fase de treinamento específica uma utilizamos a base de dados criada com imagens do instagram.

O conjunto de treino contém o mesmo formato dos exemplos presentes na base flickrlogos-32. Para cada imagem, todas as ocorrências do logotipo foram delimitadas por regiões retangulares e salvas através dos parâmetros $x, y, w, h$ representando a posição do pixel superior esquerdo, a largura e altura da região delimitadora.

O modelo deve identificar a presença do logotipo da Coca-Cola em imagens, ou seja, a avaliação será simplificada para um problema de classificação binária, onde

o modelo deverá responder se há ou não a presença do logotipo da Coca-Cola em cada imagem de exemplo, e a também realizar a localização dos logotipos existentes na imagem.

A base flickrlogos-32 será utilizada para os testes, o conjunto de imagens da Coca-Cola na flickrlogos-32 contém um total de 70 exemplos. A base flickrlogos-32 contém apenas casos de exemplos positivos, por isso foram adicionados mais 80 imagens com caso negativos, para validação de algumas métricas utilizadas na avaliação do modelo.


**Figura 5.1** – Exemplos do logotipo da Coca-Cola presentes na base de dados flickrlogos-32.

**Figura 5.2** – Exemplos mais complexos do logotipo da Coca-Cola presentes na nova base de dados.

Outro teste foi realizado com um conjunto de 505 exemplos positivos e 403 exemplos negativos com imagens de casos mais complexos, também retiradas do Instagram.

## 5.2 Métricas de desempenho

Além disto, para realização da avaliação dos resultados, algumas métricas de desempenho foram utilizadas.

Durante o teste, os resultados de classificação do sistema são recuperados e avaliados. Para cada exemplo de teste são analisados a quantidade de exemplos classificados corretamente ou não. Os acertos e erros do sistema podem ser categorizados em quatro classes, mostradas na Tabela 5.1.

No caso de resultados verdadeiros positivos, por exemplo, a imagem avaliada corresponde a um exemplo positivo (imagem com logotipo) e o sistema retorna como resultado o valor positivo, indicando um acerto de classificação.

Estes valores são normalmente utilizados para definição das métricas de desempenho. Cada métrica é relativa a uma característica específica da capacidade e desempenho do sistema. Estas métricas são descritas abaixo:

Acurácia: A acurácia mede a proporção total de resultados corretos do modelo, representando a frequência em que o modelo acerta, tanto exemplos positivos quanto negativos.

Sensibilidade: Proporção dos resultados verdadeiramente positivos classificados corretamente pelo modelo. Sensibilidade alta significa que o modelo tem menores chances de classificar um exemplo positivo como negativo.

Especificidade: Proporção dos resultados verdadeiramente negativos classificados corretamente pelo modelo. Alta especificidade significa que o modelo tem menores chances de classificar um exemplo negativo como positivo.

Eficiência: A eficiência mede a ausência de erro na predição do modelo, considerando os acertos negativos e positivos obtidos na classificação.

Valor Preditivo Negativo (VPN): É a probabilidade de um resultado classificado como negativo não ter o logotipo. O valor preditivo negativo representa a taxa de acerto para as imagens que foram rejeitadas pelo modelo, ou seja, que o modelo classificou como não contendo logotipos.

Valor Preditivo Positivo (VPP): É a probabilidade de um resultado classificado como positivo conter realmente o logotipo. O valor preditivo positivo representa a taxa de acerto do modelo para afirmações de que a imagem contém o objeto de interesse.

*Mean Squared Error* (MSE): O MSE representa o erro total de predições realizado pelo modelo. O uso do quadrado da diferença dos pesos faz essa métrica valorizar as maiores diferenças entre as respostas, dando menor importância a pequenas diferenças.

**Tabela 5.1** – Categorização dos resultados de predição para um modelo de aprendizagem de máquina.

| Classe | Valor real do exemplo | Valor retornado pelo sistema |
|---|---|---|
| Verdadeiro Positivo | Positivo | Positivo |
| Verdadeiro Negativo | Negativo | Negativo |
| Falso Positivo | Negativo | Positivo |
| Falso Negativo | Positivo | Negativo |

Abaixo segue a configuração do computador utilizado nos testes das aplicações:

- CPU: Intel Core i7 4700MQ
- RAM: 8GB
- GPU: NVIDIA GeForce GTX 950 M
- OS: Ubuntu 16.06 64 bit

## 5.3 Google Cloud Vision API

O Google Cloud Vision API é um *framework* com aplicações para visão computacional disponibilizado publicamente pela Google. A API contém métodos para construção de aplicações de reconhecimento de faces, reconhecimento de caracteres,

detecção de classes, reconhecimento de atributos na imagem (cores, formas, quantidade de objetos), reconhecimento de logotipos, etc.

Uma aplicação foi construída com a utilização desta ferramenta para validação comparativa dos resultados com os obtidos pelo sistema construído neste trabalho. Esta aplicação foi construída para comparação do desempenho do sistema construído com um sistema existente e em uso no mercado.

A API é disponível por acesso remoto e utiliza uma interface de acesso em Python. Para cada teste (exemplo avaliado) realizado uma requisição de acesso ao servidor é necessária. Esta requisição limita o desempenho computacional da aplicação.

## 5.4 Resultados

O primeiro passo foi avaliar o desempenho do modelo para diferentes parâmetros de treinamento, observando quais características melhor se adaptam à proposta.

Um passo importante no desenvolvimento de modelos que utilizam arquiteturas de redes neurais é a definição dos parâmetros de treinamento. Apesar do rápido avanço nas pesquisas desta área, a escolha dos parâmetros corretos para cada modelo continua sendo um ponto crítico na modelagem destas arquiteturas. Algumas fontes citam conjuntos de parâmetros testados empiricamente para certos modelos de redes neurais disponibilizando um ponto de partida na escolha destes parâmetros, mas não há nenhuma fonte atualmente que descreva um procedimento capaz de identificar os parâmetros ideais para cada modelo. Por este motivo, a primeira parte dos testes realizados foi focada em variar os parâmetros de treinamento do modelo construído, observando a influência de cada escolha no resultado final que o modelo é capaz de obter.

Os parâmetros variados foram os algoritmos de treinamento, as taxas de aprendizagem e a função de ativação utilizada na camada de classificação.

Para o primeiro conjunto de testes foi utilizado o mesmo conjunto de treinamento da Tabela 5.2, fixando a taxa de aprendizagem em $1e-3$ e realizando a variação do algoritmo de treinamento para *Root Mean Square* (RMS), Gradiente Descendente Estocástico (GDE) e Ada delta. Os resultados obtidos para estas configurações podem ser vistos na Figura 5.3 e 5.4.

A partir dos resultados é possível observar que o RMS obteve melhor desempenho em relação aos outros métodos analisados.

O gráfico 5.3 mostra o valor de acurácia obtido a cada iteração (época) durante o progresso de treinamento da rede neural. A figura 5.4 mostra o erro de regressão do modelo, que representa o erro MSE dos *batches* de treinamento em cada iteração.

O algoritmo RMS consegue se manter com a melhor taxa de acurácia e menor erro de classificação durante todo o intervalo de treinamento observado, mantendo a estabilidade com uma baixa taxa de variação dos resultados.

Apesar do gradiente descendente ser o algoritmo de aprendizagem mais utilizado para modelos de redes neurais simples, ele não é eficiente quando utilizado para treino de uma arquitetura de larga escala. O tempo necessário para treinamento (Tabela 5.6) utilizando este método é maior em relação aos outros métodos analisados, não adicionando nenhuma melhora significante nos resultados finais de classificação.

O algoritmo Ada Delta consegue ser mais rápidos em relação aos outros algoritmos durante o avanço do treinamento, porém perde em precisão, obtendo o maior erro de classificação entre os métodos observados

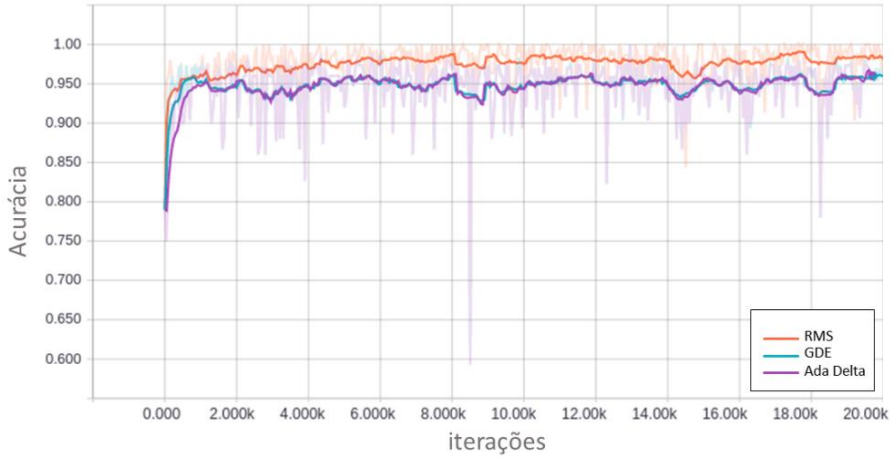


**Figura 5.3** – Resultados da acurácia obtida pelo modelo durante o treinamento variando o algoritmo de treinamento entre RMS, GDE e Ada delta.

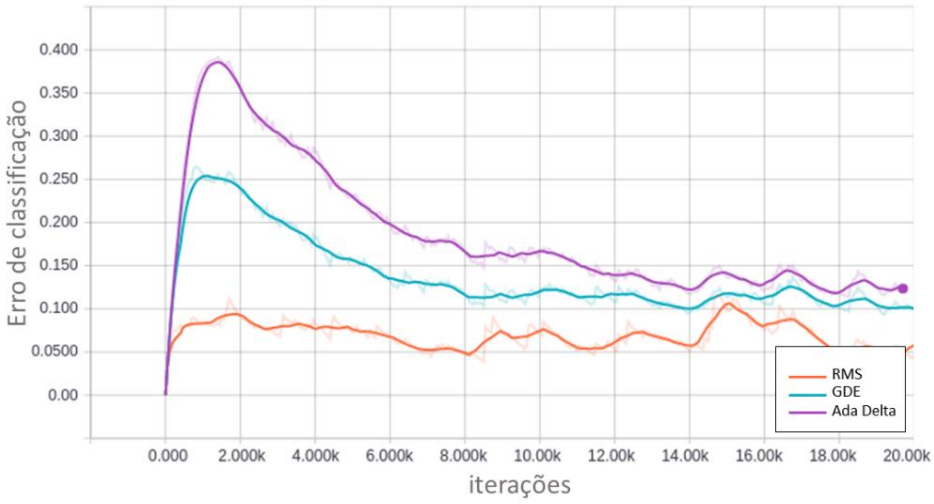


**Figura 5.4** – Resultados do erro de classificação durante o treinamento do modelo variando o algoritmo de treinamento entre RMS, GDE e Ada delta.

**Tabela 5.2** - Intervalo de tempo necessário para cada método de treinamento utilizado atingir a época (iteração) descrita.

| | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 |
|---|---|---|---|---|---|---|
| RMS | 58min42s | 1h22min31s | 3h3min29s | 6h43mim49s | 7h18min51s | 16h28min17s |
| GDE | 1h32min18s | 1h58min48s | 4h22min41s | 8h21min35s | 9h7min25s | 18h7min51s |
| Ada Delta | 1h0min8s | 1h25min45s | 3h13min41s | 6h6min18s | 6h48min21s | 15h37min4s |

No passo seguinte, foi analisada a influência da escolha da taxa de aprendizagem no treinamento do modelo, com a variação da taxa de aprendizagem entre $1e-3$, $1e-2$ com ajuste adaptativo durante o treino e $0{,}2$.

Pela figura 5.5 e 5.6 observamos que o melhor resultado foi obtido com o uso do RMS com taxa de aprendizagem adaptativa. O treinamento da rede com taxa variante torna o progresso da rede mais lento durante as iterações (Tabela 5.7), mas diminui o erro de classificação em relação ao uso da taxa de aprendizagem fixa.

O pior resultado foi obtido com a taxa de aprendizagem mais alta, isso se deve a limitação do uso da função retificadora e sua característica não saturante. Quando uma taxa de aprendizagem maior é utilizada, alguns neurônios ficam desativados na rede durante uma série de iterações, impedindo o progresso do treinamento. Apesar da diminuição do tempo de progresso da rede durante o treinamento, os valores de predição não melhoram, o que deve ser evitado no treinamento de redes neurais.


**Figura 5.5** – Resultados da acurácia obtida pelo modelo durante o treinamento variando a taxa de aprendizagem entre três variações do RMS.

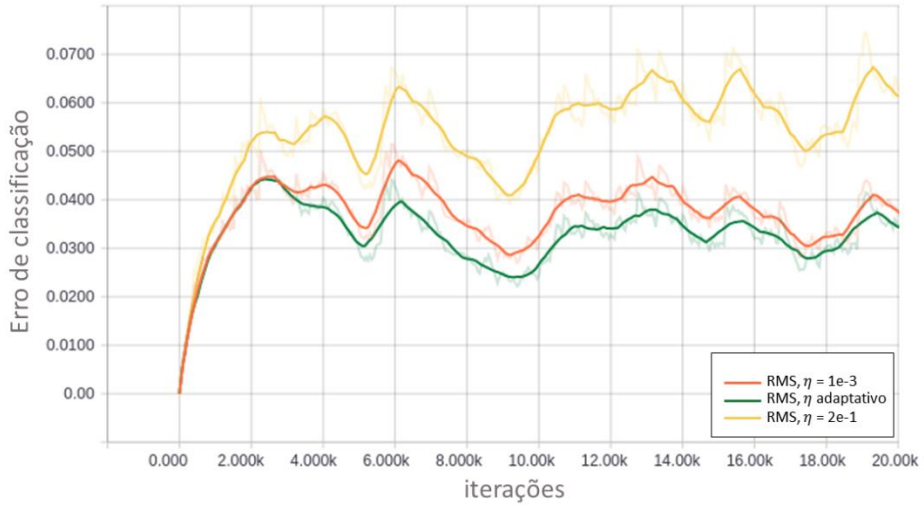


**Figura 5.6** – Resultados do erro de classificação obtida pelo modelo durante o treinamento variando a taxa de aprendizagem entre três variações do RMS.

**Tabela 5.3** - Intervalo de tempo necessário para cada variação do RMS atingir a época (iteração) descrita.

| | 1000 | 2000 | 4000 | 8000 | 10000 | 20000 |
|---|---|---|---|---|---|---|
| RMS 1 | 58min42s | 1h22min31s | 3h3min29s | 6h43mim49s | 7h18min51s | 16h28min17s |
| RMS 2 | 57min32s | 1h49min33s | 3h42min26s | 7h11min30s | 8h34min56s | 18h9min50s |
| RMS 3 | 46min55s | 1h33min31s | 3h3min17s | 5h54min35s | 7h17min51s | 15h21min1s |

Com isto, escolhemos para os testes, o algoritmo e taxa de aprendizagem que obtiveram melhor comportamento durante treino. Outros parâmetros foram definidos, como ajuste de overfitting com Dropout, decaimento para os pesos e ativação ReLU, escolhido com base em sua eficiência em modelos DCNN analisados em (A. Krizhevsky *et al*., 2012).

**Tabela 5.4** – Conjunto de treinamento utilizado no modelo proposto.

| Parâmetro | Valor |
|---|---|
| **Taxa de Aprendizagem** | $1e-2$ Adaptativo |
| **Algoritmo de treinamento** | RMS |
| **Função de ativação** | Retificadora |
| **Momentum** | 0.9 |
| **Valor de decaimento** | $1e-5$ |
| **Ajuste de Overfitting** | Dropout (50%) |

**Tabela 5.5** – Resultados de classificação dos modelos para a base flickrlogos-32.

| | VP | VN | FP | FN | Acurácia | Sensibilidade | Especificidade | Eficiência | VPN | VPP | MSE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Google API** | 60 | 80 | 0 | 10 | 93,33% | 85,71% | 100,00% | 92,85% | 88,88% | 100,00% | 0,0667 |
| **Nosso Modelo** | 69 | 80 | 0 | 1 | 99,33% | 98,57% | 100,00% | 99,28% | 98,76% | 100,00% | 0,000667 |

**Tabela 5.6** – Resultados de classificação dos modelos para de imagens do Instagram.

| | VP | VN | FP | FN | Acurácia | Sensibilidade | Especificidade | Eficiência | VPN | VPP | MSE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Google API** | 256 | 403 | 0 | 249 | 72,57% | 50,69% | 100,00% | 75,34% | 61,80% | 100,00% | 0,27422 |
| **Nosso Modelo** | 481 | 402 | 1 | 24 | 97,24% | 95,24% | 99,75% | 97,49% | 94,36% | 99,79% | 0,02533 |

A Tabela 5.3 mostra os resultados com os testes da base flickrlogos-32. As taxas obtidas por ambos os modelos foram similares. A diferença de classificação entre o Google Cloud Vision e o nosso modelo foi na taxa de sensibilidade, na qual o modelo conseguiu predizer mais exemplos positivos corretamente. Essa diferença pode ser dada pela base de treinamento especializada do modelo, com uma grande variedade de exemplos simples e complexos. Ambos os modelos obtiveram uma boa taxa de especificidade, não classificando nenhum exemplo negativo como positivo. Em geral os resultados de acurácia, VPN e VPP foram condizentes com os resultados existentes obtidos pelos modelos DCNN testados na base flickrlogos-32, como visto em (F. Landola et al., 2015).

A Tabela 5.4 mostra os resultados obtidos com os testes na base de dados complexos. A taxa de ambos os modelos diminui em relação aos resultados anteriores. O Google Cloud Vision teve uma queda considerável da acurácia e sensibilidade nesta base, contudo sua taxa de especificidade continuou a mesma. Com esta informação é possível perceber que o modelo do Google Cloud Vision tem um alto limiar para realizar classificação positiva, fornecendo como resposta positiva apenas os exemplos em que é obtido um alto grau de certeza de que existe a presença de logotipo. Esta abordagem faz com que o modelo não retorne casos negativos para avaliação, mas a alta taxa de falsos negativos influencia negativamente no valor de acurácia do modelo.

O modelo construído continuou com altas taxas de classificação positivas e negativas, mantendo altos os valores de acurácia, VPN e VPP. A taxa de especificidade diminuiu em relação aos testes anteriores, tendo o nosso modelo menos robustez em relação ao Google Cloud Vision para falsas classificações positivas, porém conseguindo manter sua proporção de resultados corretos alta.

A grande dificuldade de classificação dos modelos foi para casos onde a escala do logotipo na imagem é muito menor em relação ao ambiente em que está inserido, outro caso que dificulta a classificação dos modelos são as variações que o logotipo

contém para cada caso, diminuindo a certeza em que a rede retorna o resultado da existência do logotipo.

No geral, casos de rotação e oclusão não afetam o desempenho das arquiteturas avaliadas.

Analisando o desempenho de localização do nosso sistema, obtivemos os resultados mostrado na tabela abaixo.

**Tabela 5.7** - Resultado da localização nas duas bases de teste obtidos pelo nosso modelo.

| | **Flickrlogos-32** | **Instagram** |
|---|---|---|
| **Acerto (*Hit*)** | 131 | 715 |
| **Perda (*Miss*)** | 10 | 112 |
| **Erro (*Error*)** | 9 | 38 |

Na Tabela 5.5 é possível observar os resultados do nosso modelo para a tarefa de localização de logotipos. Nesta tabela, acerto representa os objetos que foram localizados corretamente dentro da imagem, gerando uma região que possui coordenadas delimitando o logotipo. Perda representa os logotipos na imagem que não foram localizados, e Erro é relativo as localizações falsas, onde o modelo delimita uma região sem logotipo.

O modelo obteve uma boa taxa de Acerto em relação aos valores de Perda e Erro, na maioria dos casos oclusão, rotação e alterações de iluminação não tiveram grande impacto na localização dos logotipos.

Assim como em classificação, os maiores casos de Perda são dados por logotipos que tem uma escala muito pequena em relação à toda a imagem. Outro motivo para gerar esse tipo de erro na localização é a sobreposição de várias logomarcas em locais extremamente próximos.

Os principais causadores de erros são os objetos que possuem alta similaridade com as características do logotipo analisado.

Outra métrica avaliada foi o tempo de processamento do modelo para cada exemplo, a aplicação do Google Cloud Vision possui um tempo de resposta médio de 4.2 segundos. Enquanto que o nosso modelo obteve um tempo de resposta médio de 1.5 segundos.

# 06. Conclusão

A partir dos resultados obtidos é possível ver que a utilização de arquiteturas DCNN consegue obter bons resultados de classificação e localização de logotipos.

A rede utilizada neste trabalho foi treinada para reconhecer apenas um tipo específico de logotipo. Para uma avaliação de n logotipos, uma adaptação seria necessária, aumentando o número de neurônios para cada classe de logotipo adicionada e realizando novamente o treinamento do modelo.

É esperado que a medida que mais logotipos sejam adicionados, o grau de generalização do modelo aumente também, diminuindo o grau de precisão e acurácia que o modelo é capaz de obter.

O reuso de arquiteturas pré-treinadas em uma base de dados de larga escala funcionou bem na adaptação do problema analisado neste trabalho. A rede neural de extração de características conseguiu manter sua taxa de eficiência de classificação apresentada durante na competição do ImageNet. Esta abordagem remove a necessidade de realizar o treinamento da rede DCNN completa sempre que alterações forem realizadas, mantendo um ponto de controle do estado da rede com os parâmetros treinados previamente e um estado após o ajuste da arquitetura.

O modelo construído neste trabalho obteve taxas de classificação condizentes com a aplicação gerada na Google Cloud Vision API, obtendo resultados melhores na etapa de teste com a base complexa. Isso se deve ao treinamento especializado que a rede contém, com diversas situações e imagens complexas do objetivo.

Apesar de grandes avanços ocorrerem nos estudos de redes neurais e DNNs, ainda não há um método ou procedimento que defina que conjunto de parâmetros é o mais adequado para construção dos modelos na aplicação de certos tipos de problemas, por este motivo é importante empregar um tempo na escolha dos parâmetros de treinamento durante o projeto da arquitetura, definindo um pequeno conjunto de variações que possam ser testadas. A mudança dos métodos e das taxas de aprendizagem utilizados pela rede influenciam diretamente no tempo necessário para treinamento do modelo e no resultado de classificação final que o modelo é capaz de obter.

O método de treinamento gradiente descendente estocástico é o método mais utilizado para treinamento da maioria das redes neurais existentes na literatura, porém sua utilização em modelos DCNN pode não ser a melhor escolha, produzindo um treinamento mais lento e sem mudanças significativas nos resultados finais.

O uso da função de ativação retificadora melhora o desempenho de treino da rede neural, porém limita o uso de taxas de aprendizagem baixas, devido à característica não saturante da função, podendo ocasionar dificuldade de aprendizagem da rede com neurônios inativos.

Por fim, notamos que a rede obteve uma boa taxa de localização de imagens em ambas bases de teste utilizada, tornando a adaptação realizada uma boa escolha, e um ponto de partida para construção de novos modelos de localização utilizando DCNNs sem métodos de segmentação externos.

# 07. Referências


[A. Davison *et al*., 2007] Davison, A. J.; Reid, I. D.; Molton, N. D.; Stasse, O. **MonoSLAM: Real-Time Single Camera SLAM**. IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume: 29, Issue: 6, 23 April 2007, Pages 1052-1067.

[A. Krizhevsky *et al*., 2012] Krizhevsky, A.; Sutskever, I.; Hinton, G. E**. Imagenet classification with deep convolutional neural networks**. Advances in Neural Information Processing Systems (NIPS). 2012, Pages 1106-1114.

[A. Psyllos et al., 2011] Psyllos, A.; Anagnostopoulos, C. N.; Kayafas, E. **Vehicle model recognition from frontal view image measurements**. Computer Standards and Interfaces. Volume: 33, Issue: 2, February 2011, Pages 142-151.

[A. Psyllos *et al*., 2012] Psyllos, A.; Anagnostopoulos, C. N.; Kayafas, E. **M-SIFT: A new method for Vehicle Logo Recognition**. IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2012.

[A. Samuel, 1967] Samuel, A. L. **Some Studies in Machine Learning Using the Game of Checkers**. IBM Journal of Research and Development. Volume:11, Issues:1-2, 1967, Pages 601-6017.

[A. Wagner *et al*., 2011] Wagner, A.; Wright, J.; Ganesh, A.; Zhou, Z.; Mobahi, H.; Ma, Y. **Toward a Practical Face Recognition System: Robust Alignment and Illumination by Sparse Representation**. IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume: 34, Issue: 2, 09 June 2011, Pages 372-386.

[B. Graham, 2014] Graham, B. **Fractional Max-Pooling**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[B. Leibe *et al*., 2005] Leibe, B.; Seemann, E.; Schiele, B. **Pedestrian detection in crowded scenes**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005.

[C. Szegedy et al., 2013] Szegedy, C.; Toshev, A.; Erhan, D. **Deep Neural Networks for Object Detection**. Advances in Neural Information Processing Systems 26 (NIPS).

[C. Szegedy *et al*., 2015] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. **Going Deep With Convolutions**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[D. Ballard; C. Brown, 1982] Ballard, D. H.; Brown, C. M. **Computer Vision**. Prentice Hall, 1982, 539 Pages.

[D. Doermann *et al*., 1993] Doermann, D. S.; Rivlin, E.; Weiss, I. **Logo recognition using geometric invariants**. Proceedings of the Second International Conference on Document Analysis and Recognition, 1993, Pages 894-897.

[D. Hubel; T. N. Wiesel, 1968] Hubel, D.; Wiesel, T. N. **Receptive fields and functional architecture of monkey striate cortex**. The Journal of Physiology. Volume: 195 Issue:1, March 1968, Pages 215–243.

[D. Lowe, 1999] Lowe, D. G. **Object recognition from local scale-invariant features**. The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999.

[E. Dickmanns; A. Zapp, 1998] Dickmanns, E D; Zapp, A. **Autonomous high speed road vehicle guidance by computer vision**. International Federation of Automatic Control, 1998, Pages 221-226.

[E. Francesconi *et al*., 1998] Francesconi, E.; Frasconi, P.; Gori, M.; Marinai, S.; Shen J.; Soda, G.; Sperduti, A. **Logo recognition by recursive neural networks**. In Graphics Recognition Algorithms and Systems. Springer. Volume: 1389, 1998, Pages 104-117.

[F. Landola *et al*., 2015] Landola, F. N.; Shen, A.; Gao, P.; Keutzer, K. **DeepLogo: Hitting Logo Recognition with the Deep Neural Network Hammer** IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[F. Li; P. Perona, 2005] Li, F.; Perona, P. **A Bayesian Hierarchical Model for Learning Natural Scene Categories**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005.

[F. Simões, 2016] Simões, F.; **Object detection and pose estimation from natural features for augmented reality in complex scenes**. PhD Thesis. 16 February 2016.

[F. Stark *et al*., 2015] Stark, F.; Hazirbas, C.; Triebel, R.; Cremers, D. **CAPTCHA Recognition with Active Deep Learning**. Workshop on New Challenges in Neural Computation (ECCV), 2015.

[F. Xu; K. Mueller, 2007] Xu, F.; Mueller, K. **Real-time 3D computed tomographic reconstruction using commodity graphics hardware**. Physics in Medicine and Biology. Volume: 52, Issue: 12, 17 May 2007.

[G. Hinton, 1989] Hinton, G. E. **Connectionist learning procedures. Artificial Intelligence**. Volume: 40, Issues: 1-3, September 1989, Pages 185-234.

[G. Zhu; D. Doermann, 2007] Zhu, G.; Doermann, D. **Automatic document logo detection**. International Conference on Document Analysis and Recognition (ICDAR). Volume: 2, 2007, Pages 864-868.

[I. Chin, 2011] Ike Chin. **Brand Recognition**. Disponível em <http://talk-about-branding.blogspot.com.br/2011/02/brand-recognition.html>, February 2011. Acessado em 06/02/2017.

[J. Aggarwal; Q. Cai, 1997] Aggarwal, J. K.; Cai, Q. **Human motion analysis: a review**. IEEE Proceedings Nonrigid and Articulated Motion Workshop, 16 June 1997.

[J. Daugman, 2004] Daugman, J. **How iris recognition Works**. IEEE Transactions on Circuits and Systems for Video Technology. Volume: 14, Issue: 1. January 2004, Pages 21-30.

[J. Deng *et al*., 2009] Deng, J.; Dong, W.; Socher, R.; Li, L. J.; Li, K.; Fei-Fei, L. **ImageNet: A Large-Scale Hierarchical Image Database**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[J. Kuntz, 2015] Kunt, J. **Gartner: No more apps by 2020, and robots will create robots**. Disponível em <http://gatewaysun.com/2015/10/06/gartner-no-more-apps-by-2020-and-robots-will-create-robots/>, October 2015. Acessado em 06/02/2017.

[J. Park; J. Sanderberg, 1991] Park, J.; Sanderberg, J. W. **Universal aproximation using radial basis functions networks**. Neural Computation. Volume 3, 1991, Pages 246-257.

[K.Hornik *et al*., 1989] Hornik, K.; Stinchcombe, M.; White, H. **Multilayer feedforward networks are universal approximators**. Neural Networks. Volume 2, 1989, Pages 359-366.

[K. Simonyan; A. Zisserman, 2014] Simonyan, K.; Zisserman, A. **Very deep Convolutional Networks for Large-Scale Image Recongnition**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[K. Van de Sanden *et al*., 2011] Van de Sanden, K.; Uijlings, J. R. R.; Gevers, T.; Smeulders, A. M. W. **Segmentation as selective search for object recognition**. IEEE International Conference on Computer Vision (ICCV), 2011.

[L. Deng; D. Yu, 2014] Deng, L.; Yu, D. **Deep Learning: Methods and Applications**. Foundations and Trends in Signal Processing. Volume: 7 Issues: 3-4, 2014, Pages 197–387.

[L. Meier *et al*., 2012] Meier, L.; Tanskanen, P.; Heng, L.; Lee, G. H.; Fraundorfer, F.; Pollefeys, M. **PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision**. Autonomous Robots, Springer. Volume: 33, Issue: 1, August 2012, Pages 21-39.

[L. Shapiro; G. Stockman, 2001] Shapiro, L. G.; Stockman, G. C. **Computer Vision**. Prentice Hall, 1 ed., 2001, 580 Pages.

[M. Turk *et al*., 1988] Turk, M. A.; Morgenthaler, D. G.; Gremban, K. D.; Marra, M. **VITS-a vision system for autonomous land vehicle navigation**. IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume: 10, Issue: 3, May 1988, Pages 342-361.

[O. Trie *et al*., 1996] Trie, O. D.; Jain, A. K.; Taxt, T. **Feature extraction methods for character recognition-A survey**. Pattern Recognition. Volume: 29, Issue: 4, April 1996, Pages 641-662.

[P. Sermanet *et al*., 2014] Sermanet, P.; Eigen, D.; Zhang, X.; Mathiel, M.; Fergus, R.; LeCun, Y. **Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[P. Sermanet; Y. LeCun, 2011] Sermanet, P.; LeCun, Y. **Traffic sign recognition with multi-scale convolutional networks**. Proceedings of International Joint Conference on Neural Networks, 2011.

[R. Boia *et al*., 2014] Boia, R.; Bandrabur, A.; Florea, C. **Local description using multi-scale complete rank transform for improved logo recognition**. IEEE International Conference on Communications (COMM), 2014.

[R. Girshick *et al*., 2014] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. **Rich feature hierarchies for accurate object detection and semantic segmentation**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[R. Gonzales; R. Woods, 2008] Gonzales, R. C.; Woods, R. E. **Digital Image Processing**. Prentice Hall, 2 ed., 2008, 954 Pages.

[R. Hahnloser *et al*., 2000] Hahnloser, R.; Sarpeshkar, R.; Mahowald, M. A.; Douglas R. J.; Seung, H. S. **Digital selection and analogue amplification coesist in a cortex-inspired silicon circuit**. *Nature*. Volume: 405, 2000, Pages 947-951.

[R. Klettle, 2014] Klette, R. **Concise Computer Vision**: **An Introduction into Theory and Algorithms.** Springer, 2014, 429 Pages.

[R. Stewart; M. Andriluka, 2015] Stewart, R.; Andriluka, M. **End-to-End people detection in crowded scenes**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[R. Szesliski, 2011] Szeliski, R. **Computer Vision: algorithms and applications**. Springer, 2 ed., 2011.

[S. Belongie *et al*., 2002] Belongie, S.; Malik, J.; Puzicha, J. **Shape matching and object recognition using shape contexts**. IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume: 24, Issue: 4, Apr 2002, Pages 509-522.

[S. Duffner; C. Garcia, 2006] Duffner, S.; Garcia, C. **A neural scheme for robust detection of trans-parent logos in tv programs**. International Conference on Artificial Neural Networks (ICANN), 2006.

[S. Hoi *et al*., 2015] Hoi, S. C. H.; Wu, X.; Liu, H.; Wu, Y.; Wang, H.; Xue, H.; Wu, Q.**; LOGO-net: Large-Scale Deep Logo Detection and Brand Recognition with Deep**

**Region-based Convolutional Networks**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[S. Izadi *et al*., 2011] Izadi, S.; Kim, D.; Hilliges, O.; Molyneaux, D.; Newcombe, R.; Kohli, P.; Shotton, J.; Hodges, S.; Freeman, D.; Davison, A.; Fitzgibbon, A. **KinectFusion: real-time 3D reconstruction and interaction using a moving depth câmera**. Proceedings of the 24th annual ACM symposium on User interface software and technolog (UIST), 2011.

[S. Ji et al., 2013] Ji, S.; Xu, W.; Yang, M.; Yu, K. **3D Convolutional Neural Networks for Human Action Recognition**. IEEE Transactions on Pattern and Machine Intelligence. Volume: 35, Issue: 1, January 2013, Pages 221-231.

[S. Romberg *et al*., 2011] Romberg, S.; Pueyo, L. G.; Lienhart, R.; Van Zwol, R. **Scalable logo recognition in real-world images**. In Proceedings of the 1st ACM International Conference on Multimedia Retrieval (ICMR), 2011.

[T. Kohonen; P. Somervuo, 1998] Kohonen, T.; Somervuo, P. **Self-organizing maps of symbol strings**. Neurocomputing. Volume: 21, Issues: 1-3, 6 November 1998, Pages 19-30.

[T. Mitchell, 1997] Mitchell, T. **Machine Learning**. McGraw Hill, 1997, 414 Pages.

[W. McCulloch; W. Pitts, 1943] McCulloch, W.; Pitts, W. **A logical calculus of the ideas immanent in nervous activity**. Bulletin of Mathematical Biophysics, Volume: 5, 1943, Pages 115-133.

[W. Ouyang; X. Wang, 2013] Ouyang, W.; Wang, X. **Single-pedestrian detection aided by multi-pedestrian detection**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[W. Wolf, 1996] Wolf, W. **Key frame selection by motion analysis. Conference Proceedings**. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1996.

[Y. Bengio *et al*., 2013] Bengio Y.; Courville, A.; Vincent, P. **Representation Learning: A Review and New Perspectives**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Volume: 35, Issue: 8, 2013, Pages 1798–1828.

[Y. LeCun *et al*., 1989] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. **Backpropagation applied to handwritten zip code recognition**. Neural Computation. Volume: 1, Issue: 4, December 1989, Pages 541-551.

[Y. LeCun *et al*., 1998] LeCun Y.; Bottou, L.; Bengio, Y.; Haffner, P. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE. Volume: 86 Issue:11, November 1998, Pages 2278-2324.