EXPLORANDO LIMITAÇÕES DAS REDES CONVOLUCIONAIS NA SELEÇÃO DE TESTES BINÁRIOS

BERNARDO JANKO GONÇALVES BIESSECK

EXPLORANDO LIMITAÇÕES DAS REDES CONVOLUCIONAIS NA SELEÇÃO DE TESTES BINÁRIOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais — Departamento de Ciência da Computação. como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Erickson Rangel do Nascimento

Belo Horizonte

Agosto de 2018

Biesseck, Bernardo Janko Gonçalves

B589e Explorando Limitações das Redes Convolucionais na Seleção de Testes Binários / Bernardo Janko Gonçalves Biesseck. — Belo Horizonte, 2018
xxii, 56 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais — Departamento de Ciência da Computação.
Orientador: Erickson Rangel do Nascimento

1. Computação – Teses. 2. Visão por computador. 3. Redes neurais (Computação). I. Orientador. II. Título.

CDU 519.6*82.10(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

EXPLORANDO LIMITAÇÕES DAS REDES CONVOLUCIONAIS NA SELEÇÃO DE TESTES BINÁRIOS

BERNARDO JANKO GONÇALVES BIESSECK

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ERICKSON RANGEL DO NASCIMENTO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. MARIO FERNANDO MONTENEGRO CAMPOS
Departamento de Ciência da Computação - UFMG

PROF. FLÁVIO LUIS CARDEAL PÁDUA
Departamento de Computação - CEFET-MG

DR. RENATO JOSÉ MARTINS
Pós Doc/Departamento de Ciência da Computação - UFMG

Belo Horizonte, 10 de agosto de 2018.

Este trabalho é dedicado à minha querida mãe Lucia de Lourdes Gonçalves e à minha filha Lisa Maria Moreira Biesseck, duas estrelas presentes na minha vida para sempre!

Agradecimentos

Agradeço primeiramente à minha esposa Rosi pelo apoio durante o desenvolvimento deste trabalho. A sua compreensão pelas horas de ausência em casa para estudo e pesquisa são muito importantes para mim, de verdade. Te amo!

Meus agradecimentos ao meu orientador, professor Erickson Rangel do Nascimento, pelas suas contribuições. Tenho grande admiração pelo seu trabalho dentro do Departamento de Ciência da Computação da UFMG, foi uma honra fazer parte do VeRLab e poder absorver um pouco do conhecimento produzido ali dentro. Sou grato também ao Edson Roteia pela ajuda como estudante de iniciação científica, este trabalho em conjunto foi bastante enriquecedor.

Aos colegas do VeRLab Omar Vidal, Felipe Oliveira, Daniel Balbino, Michel Silva, Washington Ramos, Guilherme Nascimento, Bruno Teixeira, Guilherme Potje, Anderson Rocha, Héctor Azpúrua, Elerson Santos, Paulo Drews e Rafael Colares, muito obrigado pelas contribuições diretas e indiretas de cada um.

Aos meus familiares, agradeço todo o apoio que tive nos momentos alegres e tristes. Em especial agradeço à minha falecida mãe Lucia de Lourdes Gonçalves e aos meus avós Luzia Augusta Pereira Cirino e José Julio Cirino, o nosso vô Zito.

Meus sinceros agradecimentos ao amigo Marcos Paulo e seus familiares Marialva e Amanda pelo acolhimento em Belo Horizonte quando iniciei o Mestrado. São pessoas maravilhosas que tive a honra de conhecer!

Não poderia deixar de agradecer também ao IFMT pela ótima oportunidade de qualificação acadêmica, é uma instituição da qual sou fruto e tenho muito carinho.

"Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar onde a maioria não chega, faça o que a maioria não faz."

(Bill Gates)

Resumo

As Redes Neurais Convolucionais (CNN) vem sendo utilizadas com sucesso em diversas aplicações que exigem reconhecimento de padrões em imagens, como detecção e classificação de objetos, reconhecimento de cenas e recuperação de imagens. Essas redes também têm contribuído para avanços na extração local de características através do aprendizado automático de *features*. É o caso do LIFT, que gera descritores de pontos de interesse com qualidade superior à dos descritores tradicionais SIFT, HOG e SURF, projetados manualmente. Mais recentemente novas metodologias baseadas em CNN foram criadas para produzir descritores binários, como DeepBit e DBD-MQ. Apesar dos resultados superiores aos dos descritores baseados em testes binários BRIEF, ORB, BRISK e FREAK estas novas abordagens adotam a estratégia de binarizar a saída em ponto-flutuante da última camada da arquitetura VGGNet, o que mantém um custo computacional elevado.

Este trabalho investiga o problema de seleção de testes binários e apresenta algumas características que limitam a busca de soluções com CNNs quando o gradiente necessário para executar o algoritmo *Backpropagation* é calculado sobre a vizinhança local dos pixels selecionados. Foram realizados experimentos com uma Rede Siamesa treinada a partir de pares de *patches* correspondentes e não-correspondentes, cujos resultados mostram a presença de *Mínimos Locais* e um outro problema definido nesta dissertação como *Componentes Incorretas do Gradiente*. Desta forma, este trabalho contribui para a compreensão do problema de seleção de testes binários e também de algumas limitações da Redes Neurais Convolucionais, no sentido de evitar a busca de soluções em direções inviáveis.

Palavras-chave: Redes Neurais Convolucionais, limitações, seleção, testes binários.

Abstract

Convolutional Neural Networks (CNN) have been successfully used in several applications that require pattern recognition in images, such as object detection, object classification, scene recognition, and image retrieval. These networks have also contributed to advances in local features extraction by automatic feature learning. This is the case of LIFT, which generates more discriminative keypoint descriptors than the traditional handcrafted algorithms SIFT, HOG and SURF. Recently new CNN-based methodologies have been created to produce binary descriptors, such as DeepBit and DBD-MQ. Despite their results are better than BRIEF, ORB, BRISK and FREAK, which are based on binary tests, these new approaches use the strategy of binarizing the floating-point output of the VGGNet architecture, what maintains a high computational cost.

This work investigates the binary tests selection problem and presents some characteristics that limit searching solutions with CNNs when the gradient needed to execute the backpropagation algorithm is computed from local neighborhood of the selected pixels. Experiments were performed with a Siamese Network trained with corresponding and non-corresponding patch pairs, whose results show the presence of *Local Minima* and another problem defined in this dissertation as *Incorrect Gradient Components*. Therefore, this work contributes to understand the binary tests selection problem and also some limitations of Convolutional Neural Networks, what avoid searching for solutions in unviable directions.

Keywords: Convolutional Neural Network, limitations, binary tests, selection.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

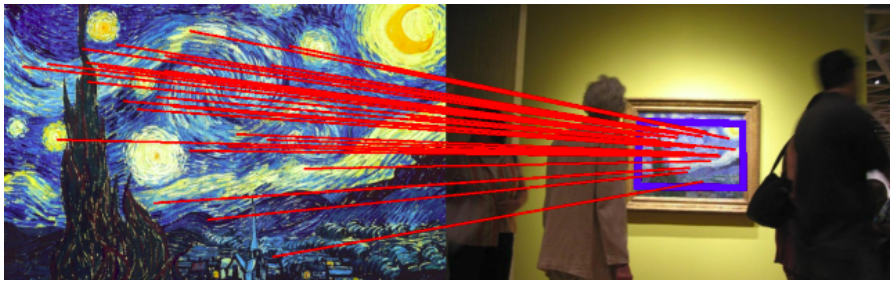
Introdução

Extrair características de imagens é uma das importantes tarefas em Visão Computacional. As informações de nível mais baixo contidas nos pixels são normalmente transformadas em novas representações e armazenadas sob a forma de vetores ou matrizes conhecidos como *feature vectors*. Isso permite filtrar variações não relevantes existentes nos dados e obter algo mais compacto e discriminativo.

A extração de características pode ser feita de maneira local, quando os vetores são gerados a partir de padrões existentes em *patches* (sub-regiões) isolados, ou global, quando um único vetor representa a imagem inteira. A escolha entre uma forma ou outra depende do problema e do padrão a ser extraído. A partir dos descritores locais é possível estabelecer correspondência entre pixels de um mesmo objeto contido em diferentes imagens, o que é fundamental em aplicações como reconhecimento de instâncias de objetos, *image stitching*, reconstrução 3D e *Simultaneous Localization and Mapping* (SLAM).

Durante muitos anos descritores locais baseados em histogramas de orientação de gradientes, como SIFT [Lowe, 2004], HOG [Dalal & Triggs, 2005], SURF [Bay et al., 2006], etc., foram os mais eficientes em diversas aplicações de visão computacional. São classificados como *handcrafted descriptors* pois seu funcionamento é baseado no conhecimento das características visuais contidas nas imagens por parte dos projetistas humanos que os criaram. Nos últimos anos esses descritores têm sido desafiados pelo aprendizado automático de *features* a partir de dados de treinamento. A arquitetura da Rede Neural Convolucional (CNN) proposta por Krizhevsky et al. [2012] foi uma das primeiras a superá-los na tarefa de reconhecimento, embora ainda não esteja totalmente claro se a rede faz um *overfitting*, beneficiando-se na verdade da enorme quantidade de dados de treinamento, ou se é de fato capaz de aprender características visuais que generalizam bem para novos dados. Esta é uma discussão feita por Zhang et al. [2017].

(a)



(b)

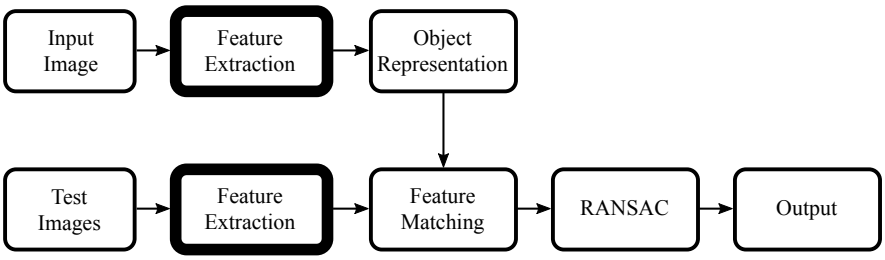


Figura 1.1: Ilustração do processo de reconhecimento de instâncias de objetos a partir da correspondência entre descritores locais de características. a) Uma mesma pintura reconhecida em imagens diferentes. As linhas vermelhas representam as associações entre os *keypoints*. b) Diagrama do processo de reconhecimento de instâncias de objetos mostrando em destaque as etapas em que os descritores locais de características são utilizados. Imagens adaptadas de Hare et al. [2011].

Apesar dos avanços importantes que as Redes Neurais Convolucionais permitiram alcançar em algumas áreas de visão computacional, os descritores locais de características continuam tendo um papel importante para várias aplicações que necessitam de informações esparsas para funcionar. É o caso do reconhecimento de instâncias de objetos, da reconstrução 3D, da *image stitching* e da SLAM. Nesses casos um único descritor global para a imagem não tem utilidade porque essas aplicações necessitam de correspondências entre pixels específicos.

No reconhecimento de instâncias de objetos o problema normalmente é, dada uma imagem de um objeto específico previamente conhecido, encontrá-lo em outras imagens. Por exemplo, um brinquedo, uma capa de livro, o rosto de uma pessoa e assim por diante. Isso pode ser feito através da correspondência dos pontos de interesse da imagem inicial com os da imagem de teste e o desafio é tratar problemas relacionados a variações de iluminação, rotação, escala, transformações perspectivas e oclusões. Os pontos de interesse de uma imagem, conhecidos como *keypoints*, são pixels cuja vizinhança local possui muitas variações de intensidade. Essas variações ao seu redor criam uma identidade para o pixel, permitindo que ele possa ser detectado e localizado em outras imagens. A Figura 1.1 mostra um exemplo e o diagrama do processo de correspondência entre *keypoints* de uma mesma pintura fotografada em

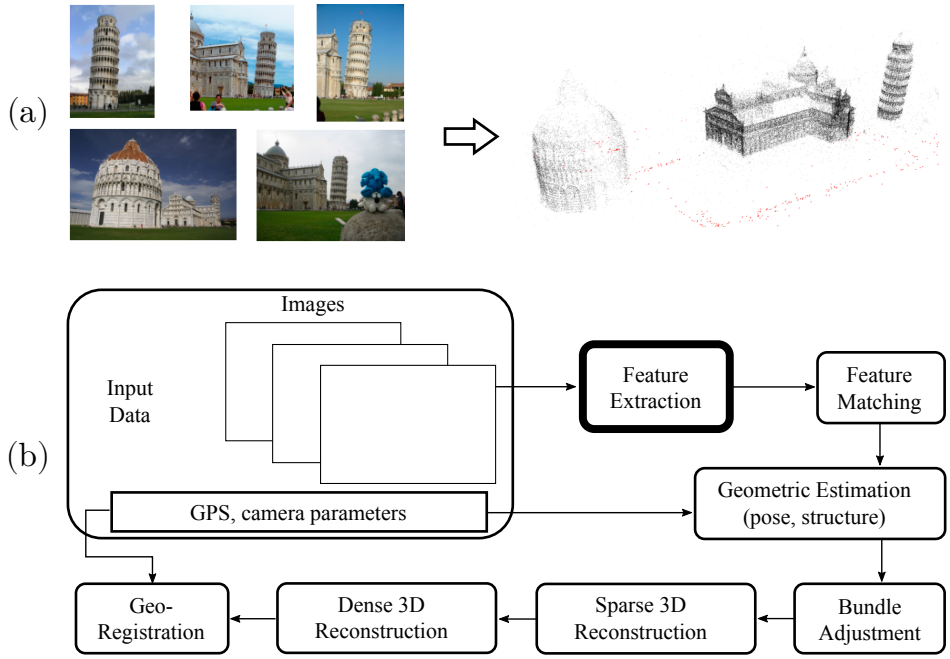


Figura 1.2: Ilustração do processo de reconstrução 3D de uma cena capturada em imagens 2D. a) Cena reconstruída a partir da correspondência entre os descritores locais de características extraídos das imagens de entrada. b) Diagrama do processo de reconstrução 3D mostrando em destaque a etapa em que os descritores locais de características são utilizados. Imagens adaptadas de Sweeney [2016].

imagens diferentes.

A reconstrução 3D, também conhecida como *Structure From Motion*, especialmente em grande escala a partir de centenas de imagens, tem atraído grande interesse na área de visão computacional. O objetivo é obter as coordenadas tridimensionais dos pontos de um objeto ou uma cena a partir de imagens 2D. O maior desafio nesta tarefa é estabelecer correspondências confiáveis entre os *pixels* das imagens. A partir dessas correspondências é possível recuperar as poses das câmeras que capturaram as imagens e as coordenadas espaciais em 3D da cena. A Figura 1.2 ilustra a reconstrução de uma cena 3D a partir de imagens 2D.

Image stitching é outra aplicação que depende de descritores locais de características e tem um papel importante na criação de imagens panorâmicas, na formação de imagens de super-resolução e na análise de imagens médicas. *Image stitching* consiste em produzir uma única imagem a partir de várias outras capturadas separadamente, como mostra a Figura 1.3. Em geral, o processo começa com a detecção e a correspondência de *keypoints* nas imagens através de descritores locais. Isso possibilita estimar as matrizes de Homografia que modelam as transformações existentes entre elas. Depois disso as imagens são distorcidas e alinhadas de modo que as regiões comuns fiquem

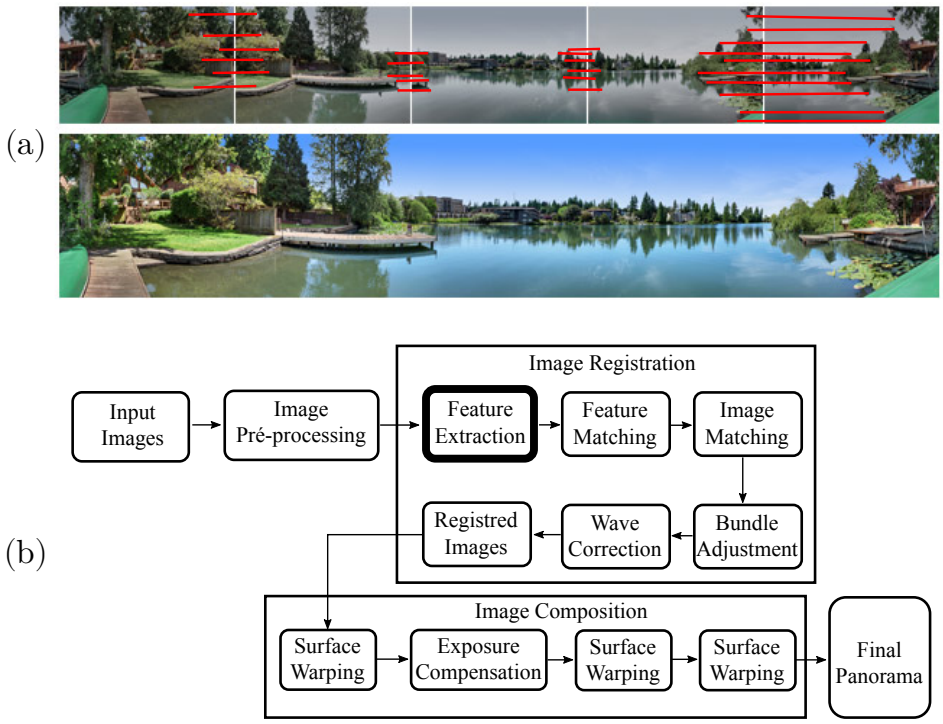


Figura 1.3: Ilustração do processo de *image stitching*. a) Imagem panorâmica criada a partir da correspondência entre os descritores locais de características extraídos das imagens de entrada. b) Diagrama do processo de casamento e alinhamento das imagens mostrando em destaque a etapa em que os descritores locais de características são utilizados. Imagens adaptadas de Brown [2018].

sobrepostas.

Descritores locais de características são utilizados também na área da robótica para fazer SLAM, cujo o objetivo é simultâneamente localizar o agente (determinar a posição e a orientação) a partir dos dados capturados à sua volta enquanto se realiza a reconstrução do ambiente em tempo real. É uma capacidade muito importante para um robô explorar um lugar desconhecido, assim como fazem os seres humanos. Existem diferentes tipos de SLAM baseados em diferentes sensores, como laser, LIDAR e câmera. O ORB-SLAM [Mur-Artal et al., 2015] é um exemplo de sistema monocular baseado em vídeo que funciona em ambientes reais, de pequeno e grande porte, *indoor* e *outdoor*. Seu nome deve-se ao uso do descritor binário ORB [Rublee et al., 2011] como um dos elementos centrais.

1.1 Motivação

Os descritores locais de características baseados em operações de ponto-flutuante são conhecidos na literatura por sua discriminância e robustez a variações de rotação, escala

e iluminação nas imagens. No entanto possuem custo computacional elevado e também são caros para serem armazenados, o que pode inviabilizar sua execução em sistemas com hardware limitado (sistemas embarcados, *smartphones*, etc.) quando a quantidade de imagens e descritores é relativamente grande. Seria o caso, por exemplo, de ter que calcular e armazenar todos os descritores de imagens capturadas 24h por dia em uma cena urbana com uma câmera de alta resolução.

Para atacar este problema surgiram os descritores binários. Projetados para serem rápidos, eles são baseados em *testes binários* que simplesmente comparam as intensidades de alguns pixels previamente selecionados ao redor do *keypoint*. Enquanto cada descritor SIFT [Lowe, 2004] ocupa 512 Bytes de memória e utiliza a distância euclidiana como medida de similaridade o descritor BRIEF [Calonder et al., 2010] ocupa apenas 32 Bytes e utiliza a distância de Hamming, que tem baixo custo computacional e pode ser implementada diretamente em hardware.

Com o objetivo de melhorar a descrição binária de pontos de interesse foram criados os algoritmos ORB [Rublee et al., 2011], BRISK [Leutenegger et al., 2011] e FREAK [Ortiz, 2012]. Cada um deles explora diferentes propriedades de imagens e define um padrão diferente para a disposição espacial dos testes binários, trazendo avanços no desempenho da descrição dos *keypoints*. No Capítulo 2 são apresentados mais detalhes sobre o funcionamento e as características de cada um deles.

A partir do conhecimento de que a distribuição espacial dos testes binários tem impacto direto na qualidade do descritor foram realizados experimentos para buscar novas soluções com a perspectiva de aprofundar o conhecimento na área. A ideia central foi utilizar o poder das Redes Neurais Convolucionais para extrair padrões ainda não observados pela comunidade científica e construir um novo descritor, mas os resultados obtidos mostraram algumas limitações deste modelo neste cenário devidos às características do problema.

Esta dissertação apresenta, portanto, algumas características do problema de seleção de testes binários e os resultados experimentais que mostram algumas limitações das Redes Convolucionais para resolvê-lo.

1.2 Definição do Problema

Um teste binário é a comparação das intesidades de dois pixels $p = I(x, y)$ e $q = I(w, z)$, sendo I uma imagem digital. Um descritor binário é uma *string* de n bits gerados a partir da aplicação de n testes binários. A escolha das coordenadas dos testes binários $(p_1, q_1), (p_2, q_2), ..., (p_n, q_n)$ tem impacto direto na qualidade do descritor em termos de

discriminância.

As perguntas que motivaram a realização deste trabalho foram: *quais são os melhores testes binários para compor um descritor? Uma abordagem baseada em CNN é capaz de encontrar uma distribuição espacial de testes binários que minimize as distâncias entre keypoints correspondentes e maximize as distâncias entre keypoints não-correspondentes?*

1.3 Objetivo Geral

O objetivo deste trabalho é explorar o problema de seleção de testes binários apresentando algumas características que limitam a busca de soluções através de Redes Convolucionais.

1.3.1 Objetivos Específicos

- Apresentar a arquitetura da Rede Convolucional e da Rede Siamesa utilizada nos experimentos;
- Descrever a modelagem de representação dos testes binários;
- Detalhar a aproximação numérica das derivadas $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$;
- Explicar o problema dos Mínimos Locais e das Componentes Incorretas do Gradiente;
- Apresentar a avaliação experimental do descritor binário aprendido pela Rede Convolucional.

1.4 Contribuições

Ao apresentar uma análise do problema de seleção de testes binários este trabalho contribui para a sua compreensão no sentido de evitar a busca de soluções em direções inviáveis.

São apresentadas também algumas limitações das Redes Convolucionais para este tipo de problema, o que contribui para a compreensão deste modelo e suas limitações.

1.5 Organização do Texto

O texto desta dissertação está organizado da seguinte maneira:

Capítulo 2 - Background: apresenta os principais conceitos que fundamentam a metodologia e os resultados apresentados nesta dissertação.

Capítulo 3 - Trabalhos Relacionados: apresenta os principais algoritmos de descrição de pontos de interesse conhecidos na literatura e também as abordagens mais recentes utilizando Redes Convolucionais.

Capítulo 4 - Metodologia: descreve algumas características do problema de seleção de testes binários.

Capítulo 5 - Experimentos: apresenta resultados de experimentos de seleção de testes binários realizados com uma Rede Convolucional.

Capítulo 6 - Conclusão: conclui o trabalho e apresenta uma visão geral sobre os resultados obtidos.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os principais conceitos que fundamentam esta dissertação. Também são apresentados os Trabalhos Relacionados utilizados como referências.

2.1 Redes Neurais

De modo geral uma Rede Neural é um algoritmo de otimização bioinspirado que funciona a partir de um conjunto de *unidades de processamento* ou *células de computação* organizadas em camadas. A rede pode ser implementada usando componentes eletrônicos ou simulada através de softwares, o que é mais comum devido à facilidade de ajustes em suas configurações. A quantidade e a maneira como as unidades de processamento são conectadas dependem da modelagem do problema e dos dados de treinamento que serão utilizados. A Figura 2.1 ilustra a estrutura genérica de uma Rede Neural com a camada de entrada, as camadas escondidas e a camada de saída.

A quantidade de unidades de processamento na camada de entrada é definida a partir da dimensionalidade dados que serão utilizados no treinamento. Se uma amostra $x_i \in \mathbb{R}^{10}$ então a primeira camada deverá ter 10 unidades de processamento. Os dados são recebidos e repassados para as camadas escondidas, que são responsáveis por deformar o espaço de representação original e criar um novo espaço de características de modo que a última camada encontre um separador linear.

O procedimento executado no treinamento de uma Rede Neural é chamado de *algoritmo de aprendizado*. Sua função é ajustar os pesos sinápticos W_i e b_i de maneira a reduzir o erro e realizar o aprendizado desejado. O processo de treinamento da rede acontece através das operações *forward propagation*, que consiste em fornecer uma amostra na camada de entrada para obter uma saída na última camada, e *back*

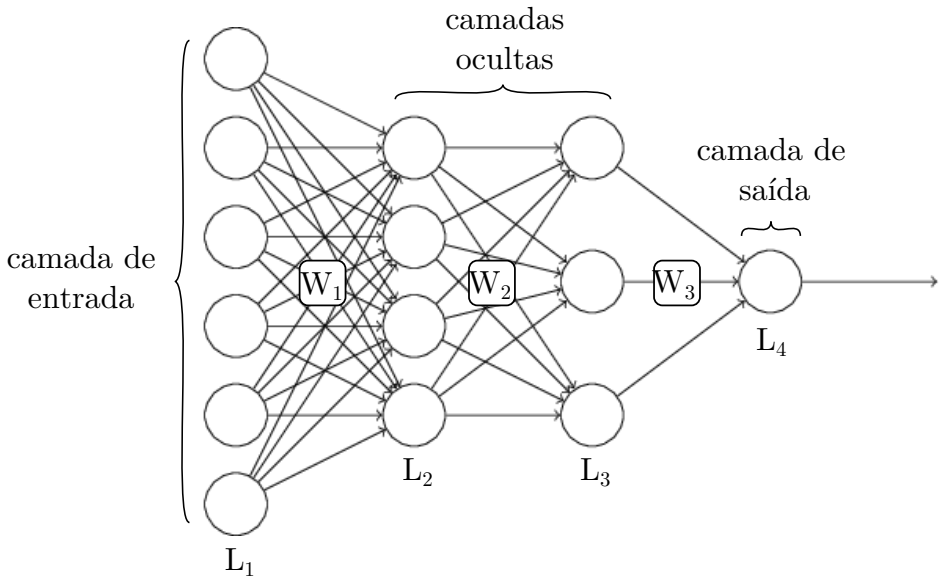


Figura 2.1: Modelo padrão de uma Rede Neural com uma camada de entrada L_1, duas camadas ocultas ou intermediárias L_2 e L_3 e uma camada de saída L_4. As ligações entre as unidades de processamento de diferentes camadas são ponderadas pelas matrizes W_i. A camada de entrada é responsável por receber as amostras de dados. As camadas ocultas realizam transformações não lineares no espaço de representação até chegar à camada de saída, que forncerá a informação sobre o padrão a ser aprendido. Imagem adaptada de [Nielsen, 2017].

propagation, na qual os pesos sinápticos são ajustados a partir do vetor gradiente da função de erro para melhorar a resposta da rede.

2.1.1 Backpropagation

A rede da Figura 2.1 pode ser escrita como uma composição de funções, i.e.,

$$L_4(X) = L4(L3(L_2(L_1, W_1), W_2), W_3), \tag{2.1}$$

sendo $L_i(X, W_i) = \varphi_i(X \cdot W_i + b_i)$ a saída da camada L_i e $\varphi_i(.)$ a sua respectiva função de ativação. Trata-se portanto de uma *função composta* em que a saída da camada L_1 servirá de entrada para L_2, que servirá de entrada para L_3, que servirá de entrada para L_4.

Para ajustar os pesos sinápticos W_i e b_i o algoritmo *Backpropagation* calcula

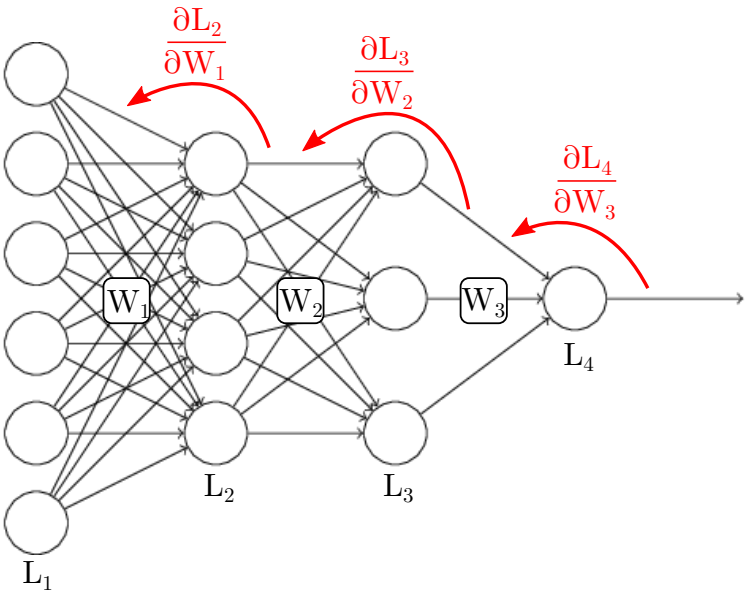


Figura 2.2: Ilustração do algoritmo *Backpropagation* utilizado no treinamento de Redes Neurais. As derivadas parciais $\frac{\partial L_4}{\partial W_3}$, $\frac{\partial L_3}{\partial W_2}$ e $\frac{\partial L_2}{\partial W_1}$ indicam a direção de redução do erro no espaço de otimização e são utilizadas para atualizar os pesos sinápticos W_i e b_i. Imagem adaptada de [Nielsen, 2017].

através da *Regra da Cadeia* os vetores gradientes

$$\nabla_3 L_4 = \frac{\partial L_4}{\partial W_3} \tag{2.2a}$$

$$\nabla_2 L_4 = \frac{\partial L_4}{\partial W_2} = \frac{\partial L_4}{\partial L_3} \cdot \frac{\partial L_3}{\partial W_2} \tag{2.2b}$$

$$\nabla_1 L_4 = \frac{\partial L_4}{\partial W_1} = \frac{\partial L_4}{\partial L_3} \cdot \frac{\partial L_3}{\partial L_2} \cdot \frac{\partial L_2}{\partial W_1}. \tag{2.2c}$$

Eles indicam em cada camada as direções em que o erro da Rede Neural é reduzido na otimização. Os pesos W_i e b_i são ajustados fazendo-se

$$W_i = W_i - \alpha \frac{\partial L_4}{\partial W_i}, \tag{2.3}$$

sendo α o hiperparâmetro conhecido como *taxa de aprendizado* (*learning rate*). A Figura 2.2 ilustra o procedimento de retropropagação.

2.2 Redes Neurais Convolucionais

Uma Rede Neural Convolucional, do inglês *Convolutional Neural Network* (CNN), é um modelo específico de Rede Neural projetado para processar imagens através de

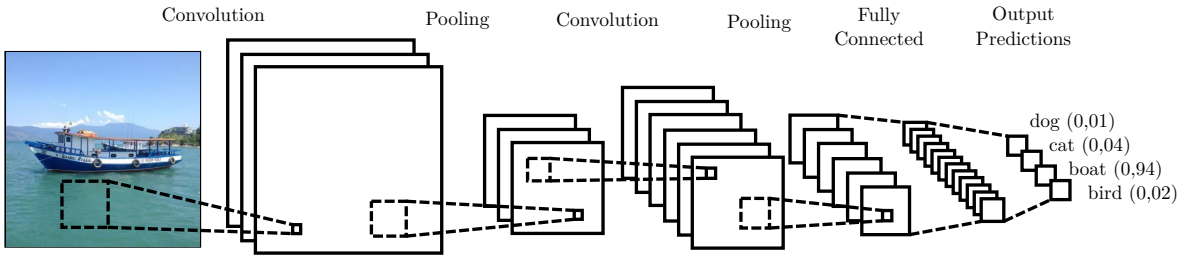


Figura 2.3: Arquitetura genérica de uma Rede Convolucional com as camadas *convolution*, *pooling*, *fully connected* e *output predictions*. Cada camada atua em diferentes níveis de abstração, as primeiras capturam features de baixo nível e as últimas de mais alto nível. Imagem adaptatda de [Karn, 2016]

sucessivas operações de convolução, *pooling* e ativação. Seu funcionamento baseia-se na consecutiva modelagem de padrões visuais existentes em diferentes níveis de abstração, combinados através das suas camadas.

As primeiras camadas operam para detectar padrões de baixo nível, como bordas, manchas e círculos. Em seguida, as próximas camadas combinam os padrões detectados para formar modelos mais elaborados correspondentes a partes de objetos em diferentes posições e com variações de iluminação, rotação e escala. As camadas finais serão ativadas por padrões mais elaborados, como rostos, carros, barcos e animais, dependendo dos dados utilizados para treinamento. Assim, as Redes Convolucionais mais profundas são capazes de reconhecer combinações complexas de padrões visuais, tendo recebido bastante atenção da comunidade científica. A Figura 2.3 mostra uma representação de uma Rede Convolucional.

O treinamento de uma Rede Convolucional é comumente realizado através do algoritmo *Backpropagation*. Seu funcionamento baseia-se na ideia de orientar o aprendizado a partir do erro calculado na última camada. A primeira premissa para que esse algoritmo funcione corretamente é que a função de erro L possa ser escrita como uma média $L = \frac{1}{n}\sum_{i=1}^{n} L(x_i)$ dos erros de cada amostra de treinamento x_i. A segunda premissa é que a função de erro possa ser derivada em relação aos pesos W_i e b_i, permitindo assim calcular os gradientes $\nabla_i L = \left\{\frac{\partial L}{\partial W_i}, \frac{\partial L}{\partial b_i}\right\}$ para a propagação do erro.

Nos últimos anos, resultados interessantes tem sido publicados utilizando Redes Convolucionais, tais como detecção de objetos [Girshick et al., 2014], reconhecimento de cena [Zhou et al., 2014] e reconhecimento de faces [Parkhi et al., 2015]. As CNNs também têm sido utilizadas para resolver problemas mais específicos, como aprender um descritor com ponto flutuante [Simo-Serra et al., 2015]. Em outro trabalho Yi et al. [2016b] treinaram uma rede siamesa para computar a orientação canônica de *keypoints* ao invés de utilizar diretamente a direção predominante dos gradientes da imagem,

como é feito no SIFT [Lowe, 2004]. Nesse trabalho, eles rotacionaram cada *patch* de treinamento de 5 em 5 graus e computaram o respectivo descritor de cada direção, podendo calcular algumas derivadas parciais numericamente.

2.2.1 Convolução

A convolução é uma operação realizada entre duas funções ou sinais de entrada que resulta em uma terceira função ou sinal de saída. É uma operação linear muito semelhante à correlação e tem diversas aplicações em processamento de sinais. A convolução discreta entre duas funções bidimensionais $f(x, y)$ e $g(x, y)$ é definida como

$$f(x, y) * g(x, y) = h(x, y) = \sum_{m=0}^{M} \sum_{n=0}^{N} f(m, n) \cdot g(x - m, y - n). \tag{2.4}$$

No contexto das Redes Convolucionais a convolução é utilizada para detectar em uma imagem de entrada $f(x, y)$ a presença do padrão definido pelo *kernel* $g(x, y)$, o que resulta em uma nova imagem $h(x, y)$, também conhecida como *feature map*, cujos valores medem a semelhança entre $f(x, y)$ e $g(x, y)$. A Figura 2.4 apresenta o resultado da convolução entre duas imagens.

No artigo "*Understanding Neural Networks Through Deep Visualization*" Yosinski et al. [2015] apresentam os *kernels* de convolução aprendidos por uma rede com 8 camadas e discutem o seu funcionamento a partir deles. A Figura 2.5 mostra alguns destes *kernels*, onde é possível observar o aumento gradativo do nível de abstração.

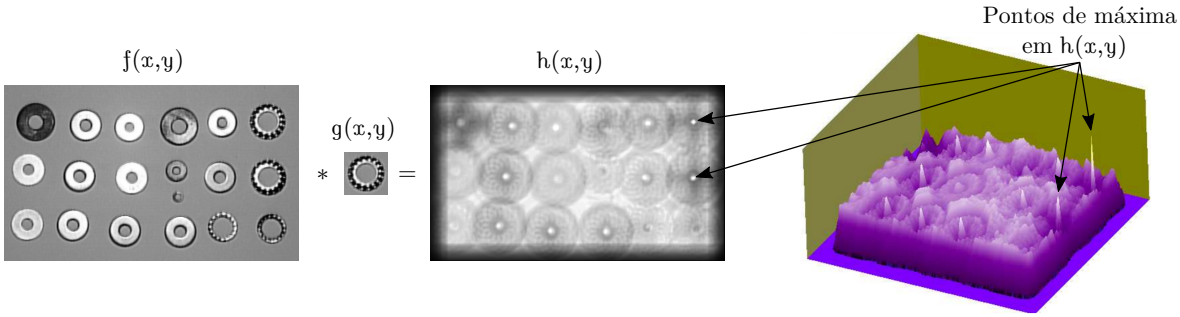


Figura 2.4: Resultado da convolução entre duas imagens $f(x, y)$ e $g(x, y)$, resultando em uma nova imagem $h(x, y)$, também conhecida como *feature map*. A imagem à direita apresenta uma visualização 3D de $h(x, y)$, onde é possível perceber que os picos estão localizados exatamente onde $g(x, y)$ aparece em $f(x, y)$.

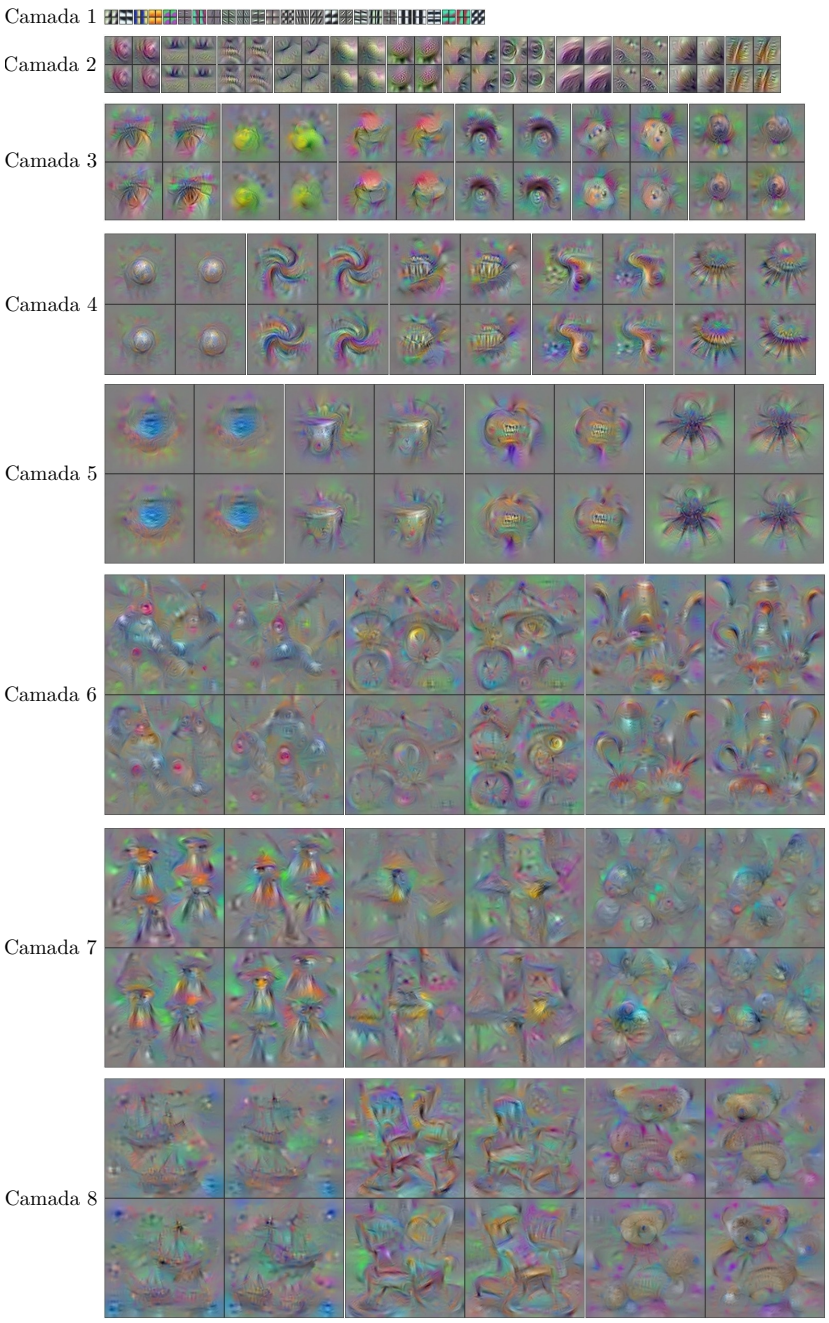


Figura 2.5: *Kernels* de convolução aprendidos por uma Rede Convolucional de 8 camadas. Nas camadas 1 e 2 os *kernels* detectam padrões visuais de baixo nível. Nas camadas 3 e 4 são detectadas partes de objetos e nas camadas 5, 6, 7 e 8 observa-se objetos inteiros. Imagens apresentadas em [Yosinski et al., 2015].

2.2.2 Pooling

Outro tipo de camada bastante importante para o funcionamento das Redes Convolucionais é a camada de *pooling*, também conhecida como *downsampling* ou *subsampling*. Sua principal tarefa é realizar uma redução do tamanho dos *feature maps* gerados pelos

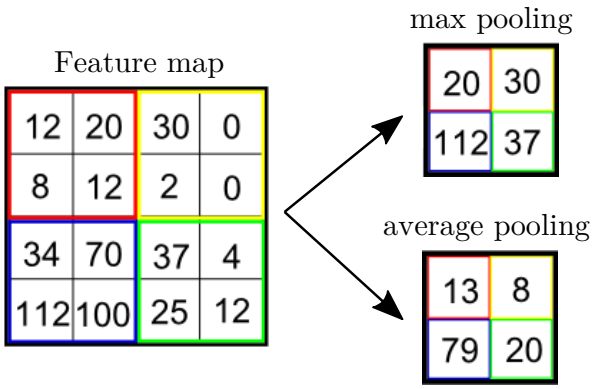


Figura 2.6: Operação de *max pooling* e *average pooling* realizadas sobre um *feature map* fictício de tamanho 4×4, com bloco 2×2 e deslocamento 2. Imagem adaptada de [Salomon, 2018]

kernels de convolução. É um processo de filtragem que tem por finalidade reduzir a quantidade de dados e extraindo somente as partes mais relevantes, dando à rede maior robustez e invariância a ruídos.

Cada camada de *pooling* recebe n imagens de entrada e fornece n de saída, mas com tamanho reduzido. As imagens são divididas em blocos de tamanho variado que são sumarizados através de uma operação de *max pooling* ou *average pooling*, como mostra a Figura 2.6.

2.2.3 Funções de ativação

A função de ativação é o mecanismo matemático implementado nas unidades de processamento para tomar a decisão de passar ou não o sinal adiante. O aprendizado de novos padrões, implica, portanto, no ajuste dos pesos sinápticos que vão modificar a resposta de uma unidade de processamento a uma determinada entrada. A seguir são apresentadas as funções de ativação mais comuns.

2.2.3.1 Sigmoid

Também é conhecida como *função logística*. Ele recebe um número real e o escala para o intervalo entre 0 e 1. Ela é bastante utilizada na camada de saída da rede, quando o objetivo é calcular a probabilidade de uma amostra pertencer a uma determinada classe. A função sigmoid converte números negativos grandes para 0 e números positivos grandes para 1. É representada como

$$sigmoid(x) = \frac{1}{1 + e^{-\lambda x}}, \tag{2.5}$$

sendo λ o parâmetro responsável pela suavidade da curva. A Figura 2.7 mostra a curva da sigmoid.

2.2.3.2 Tangente Hiperbólica (Tahn)

A tangente hiperbólica, conhecida simplesmente como *Tanh* é semelhante à sigmoid, mas as suas saídas estão entre -1 e 1. A curva é centralizada em zero e normalmente tem inclinação mais acentuada que a sigmoid. Na prática a *Tanh* é preferível à sigmoid nas camadas intermediárias porque as entradas negativas não anulam os sinais transmitidos às próximas unidades de processamento. Sua formulação é definida como

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.6}$$

Assim como a sigmoid a função *tanh* é susceptível ao *vanishing gradient problem*, que acontece quando a unidade de processamento é saturada por valores muito pequenos (negativos) ou muito grandes (positivos). Em âmbos os casos a derivada será 0 e a rede deixa de aprender. A Figura 2.7 mostra a curva da tangente hiperbólica.

2.2.3.3 Rectified Linear Unit (ReLU)

A *ReLU* tornou-se muito popular nos últimos anos com as Redes Convolucionais. Ela é retificada na parte negativa, como mostra a Figura 2.7, e é definida pela expressão

$$ReLU(x) = max(0, x). \tag{2.7}$$

Isso significa que quando $x < 0$ a saída é 0 e quando $x > 0$ a saída é x. De acordo com Krizhevsky et al. [2012] essa simples função faz a rede convergir até 6 vezes mais rápido do que a *tanh*. Ele não satura na parte positiva e por isso é mais resistente ao *vanishing gradient problem*. Além disso a *ReLU* é computacionalmente muito eficiente porque pode ser implementada usando uma limiarização simples.

2.2.3.4 Parametric Rectified Linear Unit (PReLU)

Esta função foi criada como uma alternativa à *ReLU* pelo fato dela saturar quando recebe valores negativos de entrada. A *PReLU* é definida como

$$PReLU(x) = max(0, x) + \alpha\ min(0, x), \tag{2.8}$$

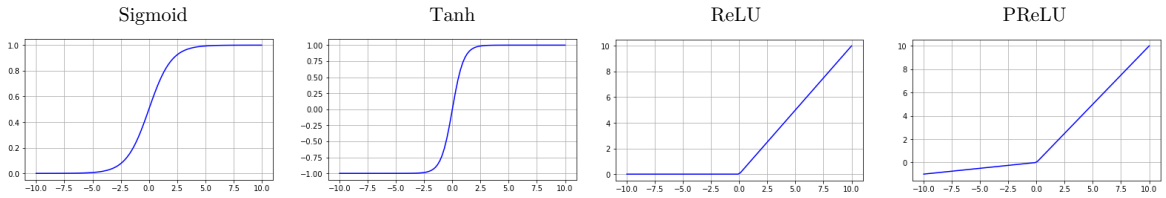


Figura 2.7: Gráficos das funções de ativação *Sigmoid*, *Tanh*, *ReLU* e *PReLU*. Imagem adaptada de [Sharma, 2017].

sendo α o parâmetro responsável pelo controle da inclinação da parte negativa da função, o que permite o seu ajuste durante o treinamento, conforme apresentado por He et al. [2015]. Quando $\alpha = 0.01$ a função é conhecida como *Leaky ReLU*.

2.2.3.5 Limitações das Redes Neurais Convolucionais

Recentemente alguns trabalhos tem sido publicados apresentando algumas limitações das Redes Convolucionais, como [Nguyen et al., 2015], [Sabour et al., 2017], [Su et al., 2017]. Isso é importante para ampliar a compreensão do que essas redes estão de fato aprendendo e possibilitar a criação de novas metodologias para problemas ainda não solucionados.

Hosseini et al. [2017] avaliam a performance de CNNs em *imagens negativas* (geradas através da inversão da intensidade de cada pixel) por ser uma transformação comumente não utilizada na fase de treinamento. Apesar de causar grandes perturbações na aparência da imagem as estruturas são mantidas (bordas, disposição dos objetos, etc.) e sua semântica continua sendo facilmente reconhecida por humanos. Os resultados deles mostram que o desempenho da rede é reduzido quando imagens negativas são apresentadas a ela, mesmo utilizando imagens classificadas corretamente

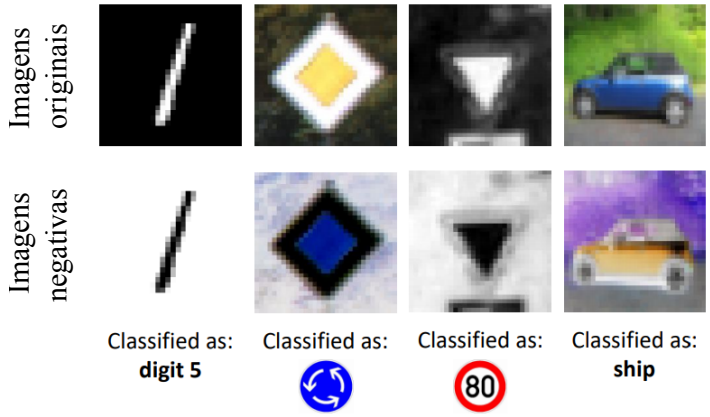


Figura 2.8: Imagens coloridas e negativas dos datasets MNIST, GTSRB e CIFAR que as CNNs avaliadas em [Hosseini et al., 2017] erraram.

em sua forma original. A Figura 2.8 mostra algumas imagens coloridas e negativas dos datasets MNIST, GTSRB e CIFAR que as CNNs avaliadas erraram. Segundo os autores isso significa que as metodologias atuais estão forçando a memorização das imagens de treinamento ao invés de extrair características genéricas.

2.3 Trabalhos Relacionados

2.3.1 Descritores de Pontos de Interesse

Uma imagem digital monocromática I é uma função bidimensional $I(x, y)$ cujas coordenadas (x, y) definem a sua intensidade naquele ponto (pixel). Interpretar o conteúdo de uma imagem depende, portanto, do processamento dos seus pixels, o que pode ser bastante custoso computacionalmente dependendo das dimensões dela. Por exemplo, uma imagem com resolução 4000×4000 pode ser representada em $\mathbb{R}^{4000^2}$ de 16.000.000! maneiras diferentes se permutarmos seus pixels.

É importante notar que nem todos os pixels têm a mesma importância quando o objetivo é analisar o conteúdo de uma imagem. Alguns são mais relevantes do que outros e têm maior carga de informações devido à sua vizinhança e o que representam na cena. Pixels localizados em regiões homogêneas têm pouca informação, normalmente são gerados por partes da cena que não contém objetos ou são partes planas deles.

Para localizar automaticamente os pixels mais relevantes de uma imagem digital foram criados os algoritmos *detectores de pontos de interesse* (*keypoint detectors*). Eles têm o papel de processar todos os pixels da imagem e gerar uma lista contendo apenas os mais relevantes de acordo com algum critério definido previamente. Alguns algoritmos detectores famosos na literatura são Scale-Invariant Feature Transform (SIFT) [Lowe, 2004], Speeded Up Robust Features (SURF) [Bay et al., 2006] e Features from Accelerated Segment Test (FAST) [Rosten & Drummond, 2006]. Eles permitem filtrar informações desnecessárias, o que reduz o custo computacional e traz maior robustez aos sitemas de visão.

Apesar dos avanços os detectores de pontos de interesse sozinhos não são capazes de extrair muita informação. É necessário algum método que descreva a vizinhança local e gere uma assinatura para cada *keypoint*, tornando possível localizá-los novamente em outras imagens. Esses algoritmos são conhecidos como *descritores de pontos de interesse* (*keypoint descriptors*) e geram o que se chama de *vetores de características* (*feature vectors*).

Vetores de características são representações criadas para descrever pixels ou regiões de imagens afim de torná-las discriminativas e robustas a transformações, como

variações de escala, rotação e iluminação. A discriminância refere-se à capacidade do algoritmo gerar vetores diferentes para pontos de interesse diferentes. Já a robustez está relacionada à capacidade do algoritmo gerar um vetor igual ainda que um mesmo ponto de interesse sofra transformações. Desenvolver um bom descritor não é uma tarefa simples e depende da captura de características visuais relevantes.

2.3.1.1 Handcrafted Descriptors

A abordagem clássica para detectar e descrever pontos de interesse é utilizar o conhecimento humano sobre imagens. Durante vários anos pesquisadores da área desenvolveram algoritmos categorizados como *handcrafted descriptors* que capturam características consideradas relevantes por eles. A seguir são apresentados alguns descritores em ponto-flutuante e também binários.

2.3.1.2 Descritores em Ponto-Flutuante

O descritor *Scale-Invariant Feature Transform* (SIFT) [Lowe, 2004] é um dos algoritmos mais conhecidos pela comunidade científica devido à sua robustez a transformações de escala, rotação e iluminação. Sua construção é baseada na distribuição dos gradientes ao redor do *keypoint*, uma característica bastante discriminativa. Lowe utilizou ainda informações de escala e rotação, o que permite obter vetores de características semelhantes ainda que o objeto esteja próximo ou distante da câmera. O descritor *Speeded Up Robust Features* (SURF) [Bay et al., 2006] surgiu como uma alternativa ao SIFT [Lowe, 2004]. Nele Bay et al. utilizaram o cálculo de *imagem integral* para reduzir o custo computacional e conseguir resultados semelhantes. As principais desvantangens do SIFT e do SURF é não serem robustos a variações não-lineares de iluminação e a transformações perspectivas porque elas modificam a distribuição dos gradientes da imagem.

O descritor *Histogram of Oriented Gradients* (HOG) [Dalal & Triggs, 2005] é utilizado como descritor de forma e captura a distribuição dos gradientes em um grid denso. Sua principal aplicação é a detecção de pedestres mas como foi projetado a partir de imagens frontais de pessoas em pé ele não funciona bem em imagens aéreas, quando há oclusão dos membros superiores e inferiores.

2.3.1.3 Descritores Binários

Apesar da boa performance os descritores com ponto-flutuante possuem custo computacional elevado por utilizarem operações de convolução no seu cálculo. Isso motivou

a criação dos *descritores binários*, que surgiram como alternativas para aplicações executadas em sistemas com poucos recursos computacionais, como sistemas embarcados, *smartphones*, etc.

Descritores binários são baseados em *testes binários* [Calonder et al., 2010], definidos como

$$\tau(I, x, y, w, z) = \begin{cases} 1 & \text{se } I(x, y) < I(w, z) \\ 0 & \text{caso contrário} \end{cases}, \tag{2.9}$$

sendo $I(x, y)$ a intensidade de uma imagem digital no pixel (x, y). Cada teste binário compara dois pixels e um conjunto de n testes binários forma o descritor.

Um *patch* de tamanho 31×31 possui $N = 961$ pixels, com os quais é possível formar $M = \binom{N}{2} = 461.280$ diferentes testes binários. Utilizar todos eles torna-se inviável em termos computacionais, seriam necessários $B = M/8 = 57.660$ Bytes para armazenar um único descritor. A escolha de um conjunto reduzido de testes binários é, portanto, fundamental para obter algo compacto que possa ser processado rapidamente.

Um descritor binário pode ser visto como uma *string* de n bits e é definido como

$$d = \sum_{i=1}^{n} 2^{i-1} \, \tau(I, x_i, y_i, w_i, z_i). \tag{2.10}$$

A similaridade entre dois descritores binários d_1 e d_2 é comumente calculada através da *Distância de Hamming* $H(d_1, d_2)$. É uma função discreta $f : F_2^{(n)} \to \mathbb{N}$ cujas entradas são vetores $d_1, d_2 \in F_2^{(n)}$, sendo $F_2^{(n)}$ um corpo finito de 2 elementos e dimensão n. Sua formulação é definida como

$$H(d_1, d_2) = \sum_{1}^{n} (d_{1,i} \oplus d_{2,i}), \tag{2.11}$$

sendo $\oplus$ o operador lógico *Exclusive OR* (XOR). A equação 2.11 contabiliza quantos bits diferentes existem entre em d_1 e d_2. Por exemplo, a distância entre os vetores $d_1 = (1, 0, 1, 1, 0, 0, 1, 1)$ e $d_2 = (1, 1, 1, 0, 0, 0, 0, 1)$ será

$$\begin{aligned} H(d_1, d_2) &= \sum[\ (1, 0, 1, 1, 0, 0, 1, 1) \oplus (1, 1, 1, 0, 0, 0, 0, 1)\ ] \\ &= \sum[\ (0, 1, 0, 1, 0, 0, 1, 0)\ ] \\ &= 3. \end{aligned} \tag{2.12}$$

Na prática, o ideal é que descritores de pontos de interesse iguais tenham distância de Hamming pequena, quanto mais próximo de 0 melhor. Assim, também é desejado que descritores de pontos de interesse diferentes tenham distância grande, quanto mais

próximo de n melhor. As principais vantagens dos descritores binários são:

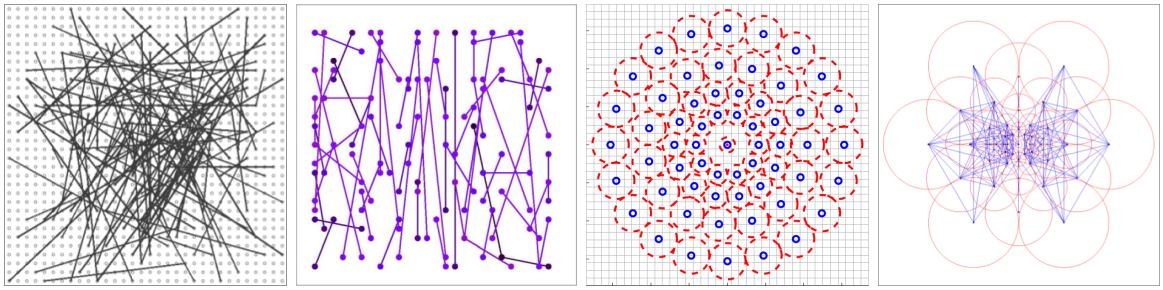
- Baixo custo computacional - a comparação de intensidades de pixels é uma operação atômica, não depende da execução de convoluções ou derivações de imagem;
- Baixo custo de espaço para armazenamento - é possível armazenar os resultados de 8 testes binários em uma única variável de 1 Byte;
- Velocidade da métrica de similaridade - a operação XOR, que fundamenta a distância de Hamming, é nativa nos processadores e bastante rápida de ser executada.

2.3.1.4 Seleção de Testes Binários

O primeiro descritor binário publicado na literatura foi o *Local Binary Pattern* (LBP) [Ojala et al., 1996]. Baseado no conceito de *Unidade de Textura* [Wang & He, 1990] uma janela de tamanho 3×3 é deslizada sobre a imagem, comparando cada pixel com seus 8 vizinhos, o que resulta em 1 Byte de informação por pixel. Após a execução desse procedimento uma nova imagem é gerada e, a partir dela, são extraídas características de distribuição das intensidades. É um descritor robusto a mudanças monotônicas de iluminação mas não a variações não-lineares de iluminação. O LPB [Ojala et al., 1996] é bastante utilizado para descrição de texturas, tendo sido aplicado no reconhecimento de faces quando o ambiente é mais controlado.

Mesmo extraindo boas características o LBP [Ojala et al., 1996] funciona bem para descrever imagens inteiras, não sendo aplicável para realizar casamento entre *keypoints* de diferentes imagens. Essa é a proposta do *Binary Robust Independent Elementary Features* (BRIEF) [Calonder et al., 2010], cuja seleção dos pixels de cada teste binário foi realizada aleatoriamente. Os autores utilizaram uma distribuição Normal $N(\mu = 0, \sigma^2 = \frac{1}{25}S^2)$ ao redor do ponto de interesse, sendo $S \times S$ o tamanho do *patch*. Foram realizados experimentos com $n = 128, 256$ e 512 bits, sendo 256 o tamanho padrão. A Figura 2.9a mostra uma visualização dos testes binários do selecionados. A principal desvantagem do BRIEF é não ser invariante a rotação então para que ele funcione adequadamente é necessário calcular a orientação canônica dos *keypoint* e rotacioná-los através de algum outro método.

O descritor *Oriented FAST and rotated BRIEF* (ORB) [Rublee et al., 2011] é uma extensão do BRIEF [Calonder et al., 2010] e foi desenvolvido baseando-se em propriedades estatísticas dos testes binários. Ao invés de uma seleção aleatória Rublee et al. criaram um conjunto com 205.590 possíveis testes binários dentro de um *patch* de tamanho 31×31 e selecionaram através de uma busca gulosa os 256 com maior variância



(a) BRIEF (b) ORB (c) BRISK (d) FREAK

Figura 2.9: Distribuição espacial dos testes binários dos descritores BRIEF [Calonder et al., 2010], ORB [Rublee et al., 2011], BRISK [Leutenegger et al., 2011] e FREAK [Ortiz, 2012]. Imagens obtidas dos seus respectivos artigos originais.

e menor correlação entre si. A variância está relacionada ao poder do algoritmo em representar *keypoints* diferentes de maneiras distintas e a correlação tem impacto na redundância das características que serão armazenadas. Devido à seleção dos testes binários o algoritmo ORB [Rublee et al., 2011] gera descritores mais discriminativos que o BRIEF [Calonder et al., 2010]. A Figura 2.9b mostra uma visualização dos testes binários do ORB [Rublee et al., 2011].

Ainda em 2011 foi apresentado o descritor *Binary Robust Invariant Scalable Keypoints* (BRISK) [Leutenegger et al., 2011], cuja distribuição espacial dos pixels é simétrica. São 60 pontos organizados de forma concêntrica, conforme mostra a Figura 2.9c. O raio de cada círculo representa o desvio padrão de uma função Gaussiana utilizada para calcular a intensidade do ponto central. A partir dos $\binom{60}{2} = 1.770$ pares possíveis, foram criados os conjuntos S (*short-distance*) e L (*long-distance*) de testes binários, respeitando os limiares $\delta_{max} = 9,75t$ e $\delta_{min} = 13,67t$, sendo t a escala do *keypoint*. O conjunto L é utilizado para calcular a orientação canônica do *patch*, já o conjunto S, formado por 512 testes binários, é aplicado para gerar o descritor.

O *Fast Retina Keypoint* (FREAK) [Ortiz, 2012] foi criado com base na visão humana, mais especificamente na retina do olho, onde a concentração de células receptoras de luz é maior na região central. A Figura 2.9d mostra os pontos selecionados para gerar os testes binários. São 43 pontos que formam $\binom{40}{2} = 903$ pares, dos quais 512 são selecionados com a mesma abordagem do ORB [Rublee et al., 2011] para formar o descritor. Outra contribuição desse trabalho está na forma de realizar *matchings*. Cada descritor é dividido em 4 grupos de 128 bits, utilizados separadamente em 4 estágios de comparação. Ao utilizar o primeiro grupo 90% dos candidatos são eliminados, o que resulta em maior velocidade de processamento se compararmos com outros descritores binários.

Diferente da estratégia de utilizar apenas a intensidade dos pixels o descritor OSRI [Xu et al., 2014] é gerado através da comparação de subregiões invariantes a rotação e iluminação. De modo geral, seu cálculo envolve três etapas principais: dividir o *patch* em subregiões a partir da ordenação das intensidades e orientações dos gradientes; computar algumas *Invariantes Regionais* para cada subregião; comparar as Invariantes Regionais com alguns pares de subregiões pré-definidas para obter o descritor binário. As Invariantes Regionais e os pares de subregiões pré-definidas são determinadas por um algoritmo de aprendizado de máquina.

A distribuição espacial dos testes binários dentro do *patch* influencia diretamente a qualidade do descritor, conforme discutido por Balntas et al. [2015]. Para demonstrar isso eles geraram aleatoriamente 5 diferentes conjuntos de 256 testes binários com a mesma distribuição Gaussiana utilizada no descritor BRIEF [Calonder et al., 2010] e cada um deles apresentou uma performance diferente. A partir dessa análise foi proposto o descritor BOLD [Balntas et al., 2015], construído por dois processos de otimização, um global e outro local. A otimização global é realizada *offline*, nela são identificados os testes binários que apresentam alta variância e baixa correlação em um conjunto total de N *patches*. Na otimização local cada *patch* é considerado uma classe separada e novas instâncias são geradas *online*, de maneira sintética, para estimar a variância intra-classe. Nesta etapa são selecionados os testes binários que minimizam a variância.

2.3.1.5 Descritores Baseados em Aprendizado de Máquina

A ideia de utilizar abordagens de aprendizado para construir novos descritores é baseada principalmente na possibilidade de capturar características visuais ainda não percebidas pelos projetistas humanos. Neste tipo de abordagem o desafio passa a ser a modelagem dos dados de treinamento e a análise dos padrões aprendidos quanto à resolução do problema em si.

O aprendizado de descritores é normalmente modelado como um problema de *aprendizado supervisionado* a partir de um conjunto P de *patches* positivos (correspondentes) e outro conjunto N de *patches* negativos (não-correspondentes). Neste caso o objetivo é obter através de algoritmos de aprendizado de máquina uma representação que faça os descritores de P terem distâncias próximas de zero e os descritores de N distâncias grandes de acordo com a métrica de distância. Os valores que definem se uma distância é pequena ou grande dependem do contexto e da modelagem do problema.

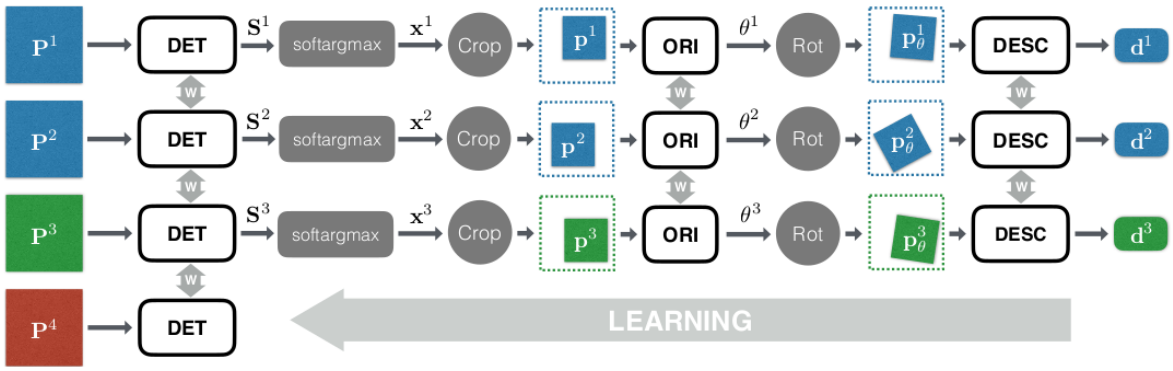


Figura 2.10: Arquitetura com 4 ramos construída para realizar o treinamento do LIFT [Yi et al., 2016a]. Os *patches* P^1 e P^2 correspondem ao mesmo ponto 3D capturados em diferentes imagens e são utilizados como amostras positivas para o descritor. O *patch* P^3 é um *patch* diferente de P^1 e P^2 e serve como amostra negativa para o descritor. O *patch* P^4 é uma imagem que não contém um *keypoint* e é utilizado para treinar o detector.

2.3.1.6 Descritores Baseados em Redes Convolucionais

Com uma proposta mais ampla o *Learned Invariant Feature Transform* (LIFT) [Yi et al., 2016a] foi criado a partir de uma Rede Siamesa com quatro ramos, cada um contendo três Redes Convolucionais treinadas para detectar *keypoints*, computar a orientação canônica e calcular o descritor. O treinamento foi realizado com quádruplas formadas por um par de *patches* correspondentes P^1 e P^2, um *patch* P^3 diferente de P^1 e P^2, e um *patch* P^4 que não contém um ponto de interesse. De modo geral, o objetivo é fazer com a distância entre os descritores de P^1 e P^2 seja minimizada enquanto se aumenta a distância deles para o descritor de P^3. O *patch* P^4 é utilizado como exemplo negativo para treinar o detector. A Figura 2.10 ilustra a arquitetura construída para fazer o treinamento.

Outro trabalho baseado em CNN é chamado de DeepBit [Lin et al., 2016]. Nele os autores utilizaram as 16 camadas pré-treinadas da VGGNet [Simonyan & Zisserman, 2014] e continuaram o treinamento para criar um novo descritor binário. Para isso binarizaram a saída da última camada *fully-connected* através da função $b = 0.5 \times (sign(F(x;W)) + 1)$, onde x representa a imagem de entrada, $F(x;W)$ a Rede Convolucional e b o descritor binário resultante. Neste caso não há uma definição específica das coordenadas dos testes binários, a ideia geral consiste em aprender pesos W que minimizem o erro de quantização entre b e o vetor de saída $F(x;W)$.

No *Deep Binary Descriptor with Multi-Quantization* (DBD-MQ) [Duan et al., 2017] a binarização das componentes da última camada da Rede Convolucional é tratada como uma tarefa de multi-quantização para reduzir ainda mais o erro de quan-

tização. Para isso são acoplados K Auto Encoders à última camada da VGGNet [Simonyan & Zisserman, 2015], contendo 16 camadas pré-treinadas. O treinamento do modelo consiste em aprender os pesos dos Auto Encoders que minimizam o erro de reconstrução entre o vetor de ponto-flutuante da VGGNet [Simonyan & Zisserman, 2014] e o descritor binário de saída.

Apesar da qualidade dos resultados dos descritores binários baseados em CNN todos eles adotam a estratégia de binarizar a saída Real da última camada da rede, o que mantém um custo computacional elevado. Nesse trabalho foi avaliada a capacidade das Redes Convolucionais na selecão de testes binários com a perspectiva de descobrir novas distribuições espaciais para criar um novo descritor. É uma junção das abordagens dos descritores binários clássicos BRIEF [Calonder et al., 2010], ORB [Rublee et al., 2011], BRISK [Leutenegger et al., 2011] e FREAK [Ortiz, 2012] com o poder de extração de características das CNNs.

Capítulo 3

Metodologia

Este capítulo descreve a metodologia proposta para avaliar a capacidade de um modelo baseado em Rede Convolucional aprender novos padrões de distribuição espacial de testes binários. São descritas as arquiteturas da Rede Convolucional e da Rede Siamesa, além da modelagem dos testes binários, da aproximação numérica para calcular o gradiente a partir de funções não-deriváveis e o treinamento multiescala.

3.1 Arquitetura da Rede Convolucional

Baseando-se em [Simo-Serra et al., 2015] foi construída uma pequena Rede Convolucional com apenas 4 camadas, sendo 3 de convolução, *pooling* e ativação e a última *fully connected* que fornece as coordenadas θ_i dos pixels dos testes binários do *patch* x_i. Esta arquitetura foi escolhida por ter sido utilizada por Simo-Serra et al. [2015] para aprendizado de um descritor de ponto-flutuante, assim o conhecimento experimental deles foi utilizado como ponto de partida.

Na camada 1 foram utilizados 32 kernels de convolução de tamanho 7×7 seguidos de uma janela 2×2 de *max pooling* e função de ativação *tanh*. A camada 2 é formada por 64 kernels de convolução 6×6, seguidos de uma janela 3×3 de *max pooling* e *tanh* como ativação. A camada 3 é formada por 128 kernels de convolução 5×5, seguidos de uma janela 4×4 de *max pooling* e ativação *tanh*. A camada 4 não tem convolução nem *pooling*, apenas os pesos associados a cada unidade de processamento e a função de ativação *sigmoid*. Ela foi aplicada por estar limitada ao intervalo $[0, 1]$, o que permite reescalar facilmente para o intervalo $[0, S]$, sendo $S \times S$ o tamanho do *patch* x_i de treinamento. A Figura 3.1 apresenta uma visualização e a Tabela 3.1 um sumário da arquitetura da Rede Convolucional com um total de 805.632 parâmetros.

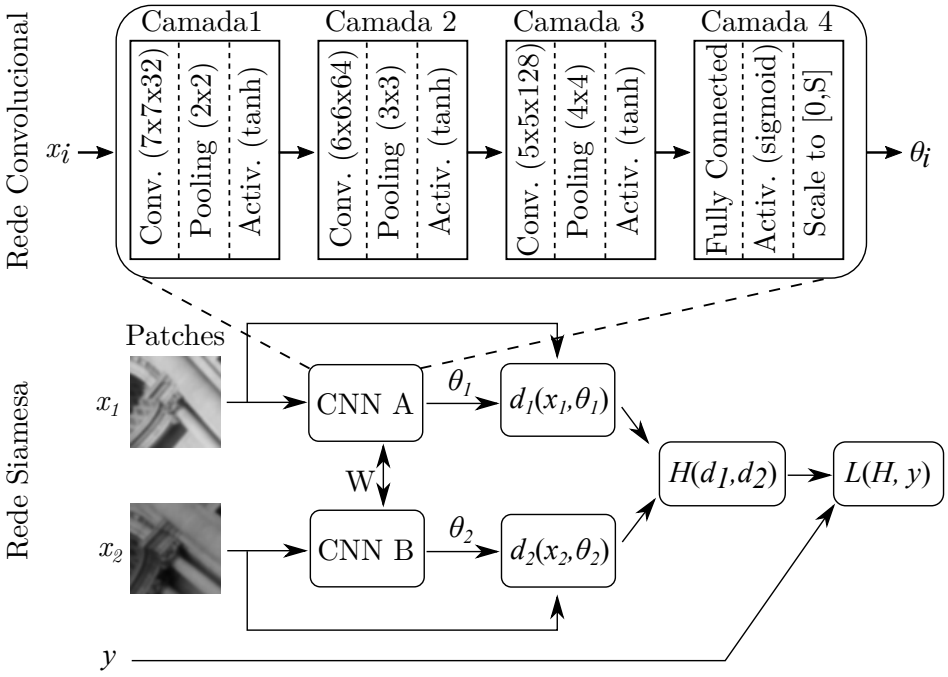


Figura 3.1: Arquitetura da Rede Convolucional e da Rede Siamesa utilizada para realizar experimentos de seleção de testes binários. A Rede Siamesa é composta de 2 Redes Convolucionais CNN A e CNN B que compartilham os pesos W_i e b_i e permitem o processamento de pares de *patches*. Para cada *patch* x_i é gerado um vetor de coordenadas θ_i.

3.2 Rede Siamesa

Para a realização de experimentos foi construída uma Rede Siamesa com 2 Redes Convolucionais idênticas à descrição anterior. Elas compartilham todos os pesos W_i e b_i, permitindo que o treinamento seja realizado com pares de amostras, muito útil em problemas de regressão. A Figura 3.1 ilustra a arquitetura e a ligação entre as camadas.

O objetivo é fazer com que a Rede Siamesa minimize a distância entre descritores de *patches* correspondentes e maximize as distâncias entre os não-correspondentes. Para isso foram adicionadas as camadas $d_i(x_i, \theta_i)$, $H(d_1, d_2)$ e $L(H, y)$. A camada $d_i(x_i, \theta_i)$ calcula o descritor binário a partir das coordenadas θ_i geradas pela rede a partir do *patch* x_i. A camada $H(d_1, d_2)$ calcula a distância de Hamming entre os descritores d_1 e d_2. Por fim, $L(H, y)$ computa o erro utilizando a função $ContrastiveLoss$ [Simo-Serra et al., 2015]

$$L(H, y) = \begin{cases} H^2 & \text{se } y = 1 \\ max(0, C - H)^2 & \text{se } y = 0 \end{cases}, \tag{3.1}$$

Tabela 3.1: Sumário da arquitetura da Rede Convolucional para realizar experimentos de seleção de testes binários. As camadas 1, 2 e 3 são compostas de convolução, *pooling* e ativação. A camada 4 é *fully connected*, sendo responsável por fornecer as coordenadas θ_i dos pixels dos testes binários do *patch* de entrada x_i. No total esta Rede Convolucional possui 805.632 parâmetros.

Camada	*1*	*2*	*3*	*4*
Tipo	Conv.+Pool.	Conv.+Pool.	Conv.+Pool.	Fully Connected
Entrada	64×64	29×29	8×8	1×1
Kernels de Conv.	32	64	128	—
Tam. dos Kernels	7×7	6×6	5×5	—
Pooling	2×2	3×3	4×4	—
Saída	29×29	8×8	1×1	$4\times$ $\#bits$
Ativação	tanh	tanh	tanh	Sigmoid

sendo $y \in \{1, 0\}$ uma variável binária que define se o par de *patches* é correspondente (1) ou não-correspondente (0). A Equação 3.1 foi modificada para facilitar a implementação da última camada, sendo definida como

$$L(H, y) = y \cdot H^2 + (1 - y) \cdot max(0, C - H)^2, \tag{3.2}$$

Quando a rede é alimentada com um par correspondente a variável $y = 1$, o que cancela o segundo termo da Equação 3.2 e faz com que o erro seja calculado como $L(H, y) = 1 \cdot H^2 + (1 - 1) \cdot max(0, C - H)^2 = H^2$. Quando um par não-correspondente é utilizado a variável $y = 0$, cancelando o primeiro termo da Equação 3.2 de forma que o erro seja calculado como $L(H, y) = 0 \cdot H^2 + (1-0) \cdot max(0, C-H)^2 = max(0, C-H)^2$.

A constante C define a distância H mínima que os descritores de *patches* não-correspondentes devem ter, penalizando valores que estiverem abaixo dela. Em contrapartida as distâncias entre descritores de *patches* correspondentes devem ser minimizadas até 0.

3.3 Testes Binários

Dado um *patch* p de tamanho $l \times c$ extraído de uma imagem digital I, um teste binário $\tau(p, x, y, w, z)$ é uma função

$$\tau(p, x, y, w, z) = \begin{cases} 1 & \text{se } p(x, y) < p(w, z) \\ 0 & \text{caso contrário} \end{cases}, \tag{3.3}$$

que compara as intensidades dos pixels $p(x, y)$ e $p(w, z)$. São necessárias, portanto, 4

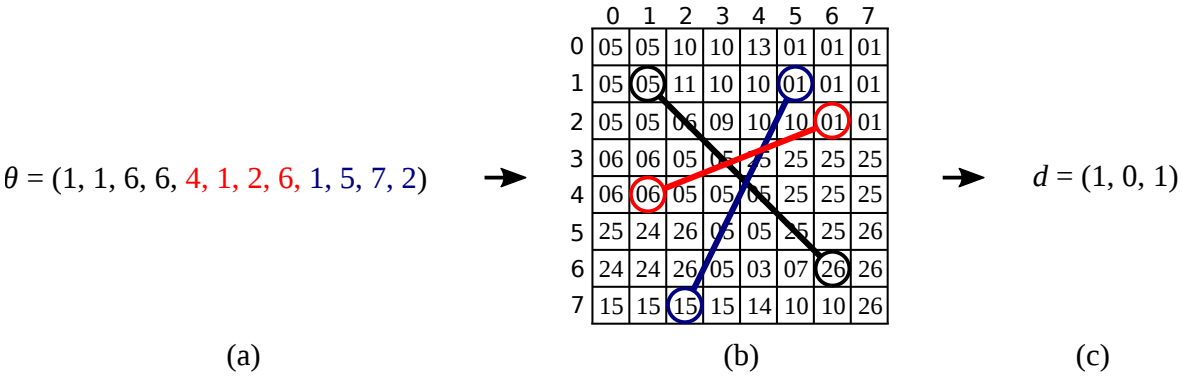


Figura 3.2: Ilustração do vetor θ para um descritor binário d de 3 bits. a) Uma instância do vetor $\theta \in \mathbb{N}^{12}$. b) Visualização dos testes binários (melhor visto em cores) sobre um *patch* 8×8. c) Descritor binário d resultante da comparação das intensidades dos *pixels* dos testes binários.

coordenadas para cada teste binário. Para aprender um descritor binário de n bits a camada 4 de cada Rede Convolucional foi configurada para fornecer o vetor $\theta_i \in \mathbb{N}^{4n}$, sendo $0 \leq \theta_i \leq S$. A Figura 3.2 ilustra uma instância do vetor θ para um descritor binário d de 3 bits. As coordenadas $(1, 1, 6, 6)$ formam o primeiro teste binário, cujo resultado será 1 porque a intensidade do pixel $p(1, 1) = 05$ é menor do que a do pixel $p(6, 6) = 26$. Da mesma forma as coordenadas do segundo teste binário $(4, 1, 2, 6)$ resultam no bit 0 porque $p(4, 1) = 06$ não é menor do que $p(2, 6) = 01$ e as coordenadas do terceiro teste binário $(1, 5, 7, 2)$ resultam no bit 1 porque $p(1, 5) = 01$ é menor que do $p(7, 2) = 15$.

3.4 Gradiente

A atualização dos pesos W_i e b_i da Rede Siamesa através do algoritmo *Backpropagation* depende dos gradientes $\nabla_i L = \left\{ \frac{\partial L}{\partial W_i}, \frac{\partial L}{\partial b_i} \right\}$ da função de erro. O gradiente é um vetor que indica o sentido e a direção na qual a função L tem o maior incremento possível a partir do ponto atual. Assim, para problemas de otimização cujo objetivo seja minimizar uma função segue-se na direção do gradiente com sentido oposto.

Dado que uma Rede Neural é uma função composta é possível calcular os gradientes utilizando a *regra da cadeia* através das equações

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial H} \left(\frac{\partial H}{\partial d_1} \frac{\partial d_1}{\partial \theta_1} \frac{\partial \theta_1}{\partial W_i} + \frac{\partial H}{\partial d_2} \frac{\partial d_2}{\partial \theta_2} \frac{\partial \theta_2}{\partial W_i} \right) \text{ e} \tag{3.4}$$

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial H} \left(\frac{\partial H}{\partial d_1} \frac{\partial d_1}{\partial \theta_1} \frac{\partial \theta_1}{\partial b_i} + \frac{\partial H}{\partial d_2} \frac{\partial d_2}{\partial \theta_2} \frac{\partial \theta_2}{\partial b_i} \right). \tag{3.5}$$

Para cada par de *patches* de treinamento a derivada parcial $\frac{\partial L}{\partial H}$ indica a direção que a distância de Hamming deve seguir. Para os pares correspondentes a derivada $\frac{\partial L}{\partial H} > 0$ quando $H(d_1, d_2) > 0$, o que significa que a distância daquele par deve ser reduzida. Já para os pares não-correspondentes a derivada $\frac{\partial L}{\partial H} < 0$ quando $H(d_1, d_2) < C$, indicando que a distância de Hamming deve aumentar além da margem C. Na camada anterior as derivadas $\frac{\partial H}{\partial d_1}$ e $\frac{\partial H}{\partial d_2}$ indicam quais bits dos descritores d_1 e d_2 devem ser modificados para seguir a direção indicada por $\frac{\partial L}{\partial H}$. Do mesmo modo, as derivadas $\frac{\partial d_1}{\partial \theta_1}$ e $\frac{\partial d_2}{\partial \theta_2}$ indicam as direções de seleção dos pixels a partir das coordenadas atuais θ_1 e θ_2 para que os bits indicados por $\frac{\partial H}{\partial d_1}$ e $\frac{\partial H}{\partial d_2}$ sejam modificados. Mais atrás, as derivadas $\frac{\partial \theta_1}{\partial W_i}$, $\frac{\partial \theta_2}{\partial W_i}$, $\frac{\partial \theta_1}{\partial b_i}$ e $\frac{\partial \theta_2}{\partial b_i}$ indicam as direções de atualização dos pesos W_i e b_i para que os pixels indicados por $\frac{\partial d_1}{\partial \theta_1}$ e $\frac{\partial d_2}{\partial \theta_2}$ sejam selecionados.

As derivadas parciais $\frac{\partial L}{\partial H}$ e $\frac{\partial \theta_i}{\partial W_i}$ podem ser obtidas analiticamente e resultam em

$$\frac{\partial L}{\partial H} = \begin{cases} 2Hy & \text{se } C \leq H, \\ 2(C(y-1)+H) & \text{caso contrário} \end{cases} \text{ e} \tag{3.6}$$

$$\frac{\partial \theta_i}{\partial W_i} = \frac{\partial}{\partial W_i}\varphi_i(W_i X_{i-1} + b_i), \tag{3.7}$$

sendo $\varphi_i(.)$ a função de ativação da camada i e X_{i-1} a saída da camada anterior da Rede Convolucional. A camada 4, cuja função de ativação é a *sigmoid*, tem derivada

$$\frac{\partial \theta_i}{\partial W_4} = \sigma(W_4 X_3 + b_4)(1 - \sigma(W_4 X_3 + b_4)). \tag{3.8}$$

As camadas 3, 2 e 1, que possuem a função de ativação *tanh*, tem derivadas

$$\frac{\partial \theta_i}{\partial W_3} = 1 - tanh(W_3 X_2 + b_3)^2, \tag{3.9}$$

$$\frac{\partial \theta_i}{\partial W_2} = 1 - tanh(W_2 X_1 + b_2)^2 \text{e} \tag{3.10}$$

$$\frac{\partial \theta_i}{\partial W_1} = 1 - tanh(W_1 X_0 + b_1)^2, \tag{3.11}$$

sendo X_0 o *patch* de entrada da rede.

As demais derivadas, $\frac{\partial H}{\partial d_i}$ e $\frac{\partial d_i}{\partial \theta_i}$, não podem ser obtidas analiticamente, pois dependem de operações não-deriváveis, como a *Exclusive OR* (XOR) e a comparação das intensidades dos pixels para computar os testes binários. A solução encontrada foi aproximar numericamente a derivada $\frac{\partial L}{\partial \theta_i}$ por diferenças finitas. Através da fórmula

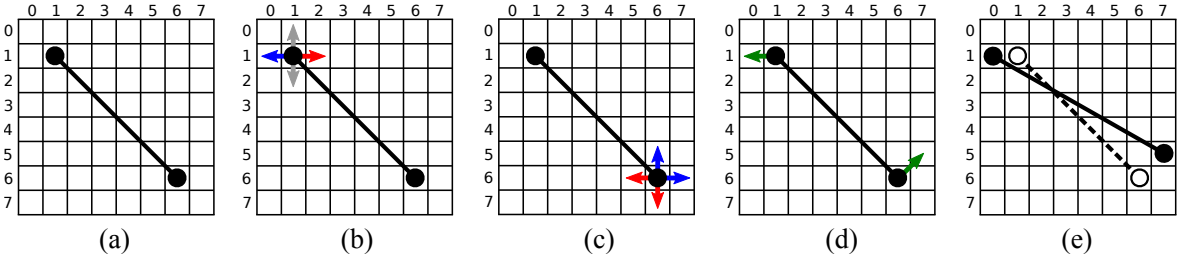


Figura 3.3: Ilustração do processo de seleção de testes binários. a) Um teste binário $\theta = (x_1, y_1, x_2, y_2)$ sobre um *patch* fictício de tamanho 8×8; b) Variação das coordenadas x_1 e y_1 nas direções horizontal e vertical. A seta azul indica que o erro diminui naquela direção e sentido, a seta vermelha indica aumento do erro naquela direção e sentido e as setas cinzas indicam que o erro não sofre alteração naquela direção e sentido; c) Variação das coordenadas x_2 e y_2 nas direções horizontal e vertical. A seta azul indica que o erro diminui naquela direção e sentido, a seta vermelha indica aumento do erro naquela direção e sentido e as setas cinzas indicam que o erro não sofre alteração naquela direção e sentido; d) Vetores resultantes em verde apontando para as novas coordenadas do teste binário; e) Teste binário resultante após a contabilização do erro, sendo $x_1 = x_1 - 1$, $x_2 = x_2 + 1$ e $y_2 = y_2 - 1$.

centralizada obtêm-se

$$\frac{\partial L}{\partial \theta_1} = \frac{L(H(d_1(x_1, \theta_1 + \Delta), d_2(x_2, \theta_2))) - L(H(d_1(x_1, \theta_1 - \Delta), d_2(x_2, \theta_2)))}{2\delta} \text{ e} \quad (3.12)$$

$$\frac{\partial L}{\partial \theta_2} = \frac{L(H(d_1(x_1, \theta_1), d_2(x_2, \theta_2 + \Delta))) - L(H(d_1(x_1, \theta_1), d_2(x_2, \theta_2 - \Delta)))}{2\delta}. \quad (3.13)$$

Em se tratando de pixels foi definido que $\Delta = [\delta_1, \delta_2, ..., \delta_N]$ para a resolução original da imagem, sendo $\delta_i = 1$ e $N = 4 \cdot \#bits$. Desta forma os gradientes $\frac{\partial L}{\partial W_i}$ e $\frac{\partial L}{\partial b_i}$ foram calculados como

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial \theta_1}\frac{\partial \theta_1}{\partial W_i} + \frac{\partial L}{\partial \theta_2}\frac{\partial \theta_2}{\partial W_i} \text{ e} \quad (3.14)$$

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial \theta_1}\frac{\partial \theta_1}{\partial b_i} + \frac{\partial L}{\partial \theta_2}\frac{\partial \theta_2}{\partial b_i}. \quad (3.15)$$

Com esta formulação o que a Rede Siamesa faz é variar as coordenadas dos testes binários nas direções horizontal e vertical para escolher os pixels que diminuem o erro. Para os pares de *patches* correspondentes são selecionados os pixels que reduzem a distância de Hamming entre os seus descritores. Já no caso dos não-correspondentes

são escolhidos aqueles que aumentam essa distância. A Figura 3.3 ilustra a variação das coordenadas e o teste binário resultante.

3.5 Treinamento Multiescala

A partir dos experimentos realizados com a modelagem dos testes binários descrita na seção 3.3 e a aproximação numérica do gradiente descrita na seção 3.4, foi descoberta a presença de *Mínimos Locais* durante o treinamento da Rede Siamesa. Isso fazia ela parar de aprender antes de alcançar um erro satisfatoriamente baixo. Maiores detalhes sobre a análise dos Mínimos Locais são apresentados no capítulo 4.

A solução proposta para diminuir o impacto dos Mínimos Locais foi realizar o treinamento em uma pirâmide de escalas, da menor resolução para a maior, permitindo à Rede Siamesa ter uma visão global dos *patches* e escolher os testes binários baseando-se primeiramente em regiões discriminativas para depois analisar pixels individualmente. A redução de escala foi realizada aplicando-se convoluções com filtro Gaussiano

$$G(x, y; k, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.16}$$

sendo $k \times k$ as dimensões do *kernel*, calculadas como $k = \left(\frac{3S}{r} + 1\right)$, sendo $S \times S$ as dimensões do *patch*, r a resolução espacial para a qual se deseja reduzir o *patch* e $\sigma = \frac{k}{f}$. Para os experimentos foi definido $f = 2,5$ para obter uma cobertura de $98,76\%$ sob o *kernel* Gaussiano.

Para garantir uma boa distribuição espacial dos testes binários sobre os *patches* reescalados as coordenadas foram ajustadas dentro de um *grid* de tamanho $r \times r$, correspondendo à resolução atual. A Figura 3.4 mostra um *patch* nas resoluções 8×8, 16×16 e 32×32 obtidos com esta abordagem a partir da resolução orginal 64×64.

Além disso, uma restrição para que a Rede Siamesa selecione somente um pixel por célula do *grid* para formar os testes binários foi adicionada. Isso foi implementado através das *coordenadas virtuais* $\hat{\theta}_i \in \{0, \cdots, r-1\}$, definidas neste trabalho. Na resolução 8×8 as coordenadas vituais $\hat{\theta}_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ pois o *grid* é formado por 8 linhas e 8 colunas. Nas resoluções 16×16 e 32×32 as coordenadas virtuais $\hat{\theta}_i \in \{0, \cdots, 15\}$ e $\hat{\theta}_i \in \{0, \cdots, 31\}$, respectivamente. Para haver sobreposição dos pixels selecionados em diferentes resoluções as coordenadas foram ajustadas de acordo

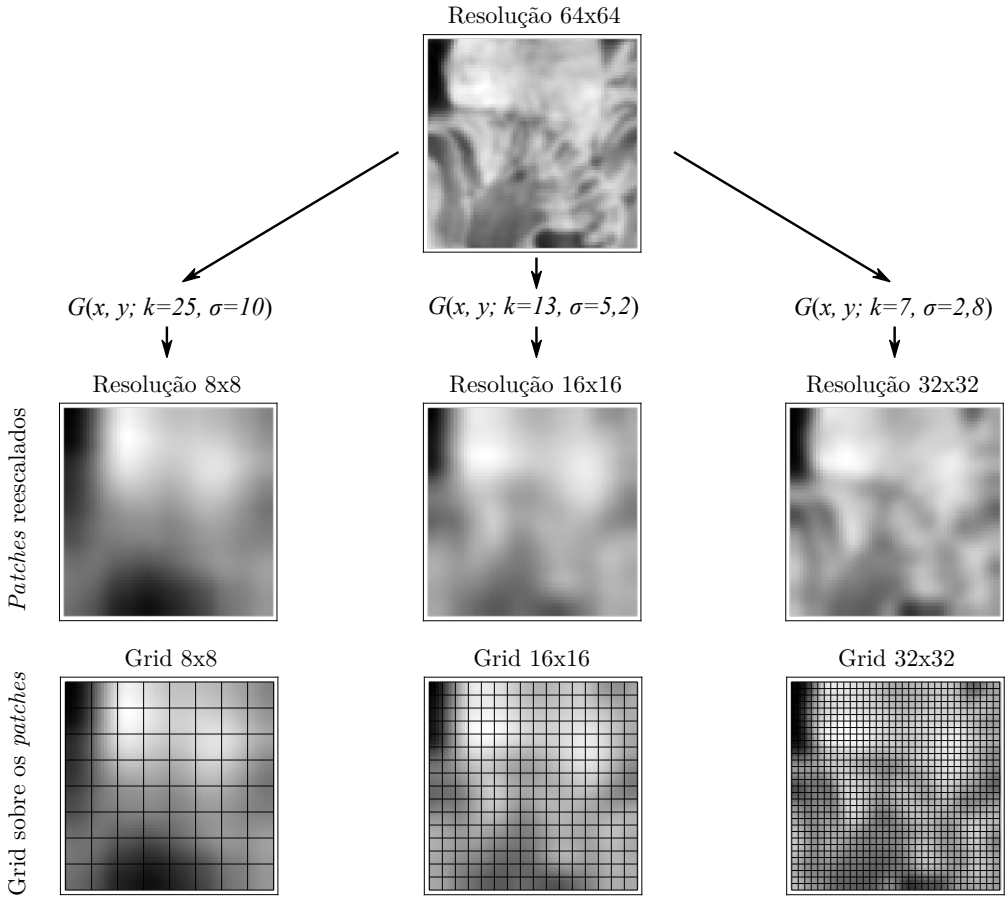


Figura 3.4: *Patch* 64×64 reescalado com filtros Gaussianos $G(k, \mu_x, \mu_y, \sigma)$ para as resoluções 8×8, 16×16 e 32×32. Para garantir uma boa distribuição espacial dos testes biários sobre os *patches* reescalados foi aplicado um grid $r \times r$ correspondente à resolução atual. Neste cenário os testes binários só podem ser compostos por um pixel de cada célula do *grid*.

com a função polinomial de grau 3

$$\alpha(r) = Round(0.000139508929r^3 - 0.0130208333r^2 + 0.25r + 0.761904762), \quad (3.17)$$

aproximada por Mínimos Quadrados a partir das resoluções de entrada $(8, 16, 32, 64)$ e saídas desejadas $(2, 2, 0, 0)$. A compatibilidade de dimensões com o vetor $\hat{\theta}_i$ foi realizada definindo-se o vetor $\hat{\alpha} = [\alpha_1(r), \alpha_2(r), \cdots, \alpha_N(r)]$, sendo $\alpha_i(r) = \alpha(r)$ e $N = 4 \cdot \#bits$. Assim, a conversão das coordenadas virtuais $\hat{\theta}_i$ para as coordenadas reais θ_i é feita calculando-se

$$\theta_i = \left(\hat{\theta}_i \cdot \frac{64}{r}\right) + \hat{\alpha}. \quad (3.18)$$

A simulação da resolução espacial 8×8 foi feita convoluindo os *patches* originais com o kernel Gaussiano $G(x, y; k = 25, \sigma = 10)$ antes de iniciar o treinamento, o que foi definido neste trabalho como época 0. Após algumas épocas foram mantidos os pesos W_i e b_i aprendidos pela rede e passou-se para a resolução espacial simulada 16×16 convoluindo os *patches* originais com o kernel Gaussiano $G(x, y; k = 13, \sigma = 5, 2)$. O treinamento prosseguiu por mais algumas épocas, permitindo que a rede pudesse escolher pixels mais próximos. Em seguida foi feito o mesmo ao simular a resolução espacial 32×32, mantendo os pesos W_i e b_i e convoluindo os *patches* originais com o kernel Gaussiano $G(x, y; k = 7, \sigma = 2, 8)$.

Capítulo 4

Experimentos

Este capítulo descreve os experimentos de seleção de testes binários realizados a partir da modelagem apresentada no Capítulo 3. A Rede Siamesa foi alimentada com pares de *patches* correspondentes e pares não-correspondentes, tendo o objetivo de selecionar os testes binários que reduzem as distâncias de Hamming dos correspondentes e aumentam as dos não-correspondentes acima da margem C definida na *Contrastive Loss*.

O primeiro problema encontrado foi a existência de *Mínimos Locais* na função objetivo. Eles impedem a seleção de pixels espacialmente distantes porque os vizinhos imediatos dos testes binários atuais não reduzem o erro depois de algumas épocas de treinamento. O outro problema é a existência de *Componentes Incorretas do Gradiente*, que aparece quando os gradientes indicam de maneira errada a direção e o sentido que reduzem o erro.

4.1 Dataset de Treino

Os experimentos foram realizados com *patches* de tamanho 64×64 extraídos do *dataset* Trevi Fountain [Winder & Brown, 2007]. Ele contém mais de 30.000 *keypoints* diferentes, cada um contendo entre 5 e 50 instâncias capturadas com variações de rotação, escala e iluminação. Os *patches* foram organizados aleatoriamente em pares correspondentes e pares não-correspondentes. A Figura 4.1 ilustra a composição dos pares e apresenta algumas instâncias.

Para melhorar a capacidade de generalização da Rede Siamesa foi atribuída uma cota de pares correspondentes e não-correspondentes a cada *keypoint* do *dataset*, garantindo assim maior uniformidade na seleção das amostras. O procedimento se baseia na quantidade k de *keypoints* disponíveis, na quantidade T de pares desejados e na porcentagem $P \in [0, 1]$ desejada de pares correspondentes. A porcentagem N de pares

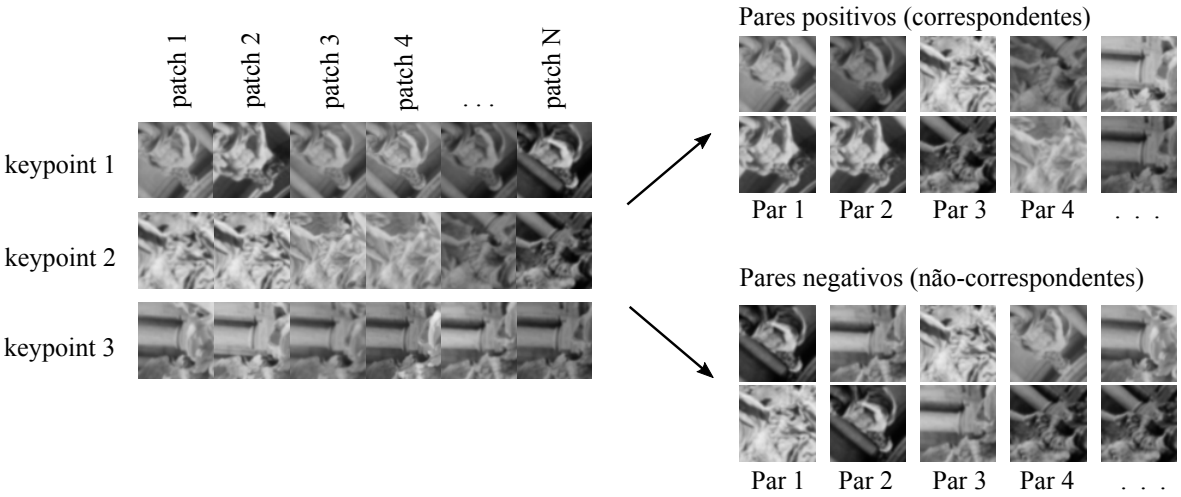


Figura 4.1: Ilustração dos pares de *patches* gerados para treinamento da Rede Siamesa. Cada *patch* corresponde a um *keypoints* detectado e extraído de uma cena registrada em imagens com diferentes condições de iluminação, rotação e escala. A partir deles são organizados os conjuntos P de pares positivos (*patches* correspondentes) e N de pares negativos (*patches* não-correspondentes). Imagens extraídas do *dataset* Trevi Fountain [Winder & Brown, 2007].

não-correspondentes é calculada como $N = 1 - P$. A cada *keypoint* é atribuída uma cota $C_{pc} = \left\lceil P \cdot \frac{T}{k} \right\rceil$ de pares correspondentes e outra $C_{pn} = \left\lfloor (1 - P) \cdot \frac{T}{k} \right\rfloor$ de pares não-correspondentes.

4.2 Ajuste de Hiperparâmetros

A Rede Siamesa criada para aprendizado de testes binários foi treinada utilizando o protocolo *K-Fold*, com $K = 5$, sendo três *folds* para treino, um para validação e outro para teste. Em cada *fold* foram criados $T = 10.000$ pares de *patches*, sendo 50% correspondente e 50% não-correspondente. Dessa forma, a rede foi treinada com 30.000 pares de *patches*. O aumento dessa quantidade não resultou em melhoria na qualidade do descritor devido aos problemas apresentados nas seções 4.4 e 4.5.

Para a otimização foi utilizado o algoritmo *Stochastic Gradient Descent* (SGD), sendo os melhores resultados obtidos com *batch size* 32, *learning rate* 1×10^{-9}, *momentum* 0.9 e *decaimento* 1×10^{-3}. Foram executadas apenas 30 épocas de treinamento porque a partir dessa quantidade a taxa de redução do erro tornava-se muito pequena e a qualidade do descritor estagnava.

Quanto à função objetivo, *Contrastive Loss*, os melhores resultados foram obtidos utilizando a margem $C = 150$. A implementação da Rede Siamesa foi feita em Python com Keras [Chollet et al., 2015] e Theano [Theano Development Team, 2016].

4.3 Distribuição de Distâncias

A distribuição de distância entre os *patches* de treinamento indica se o aprendizado dos testes binários está ocorrendo da maneira esperada. O ideal é que descritores de *patches* correspondentes tenham distâncias pequenas, próximas de zero, e que descritores de *patches* não-correspondentes tenham distâncias grandes de acordo com a medida de similaridade aplicada.

Para fazer essa análise as distâncias foram observadas a cada 10 épocas de treinamento, sendo a época 0 referente ao estado inicial da Rede Siamesa, com pesos W_i e b_i inicializados randomicamente. A Figura 4.2 apresenta os histogramas de distâncias que demonstram o comportamento esperado pois as distâncias entre os pares correspondentes diminuíram enquanto as distâncias entre os não-correspondentes aumentaram ao longo do treinamento. Na primeira linha de histogramas observa-se que época 0 as distâncias entre os pares correspondentes estavam concentradas em torno de 75 e ao final caíram para 20. Já as distâncias entre os pares não-correspondentes estavam concentradas em torno de 125 e cresceram para 150 no final da época 30.

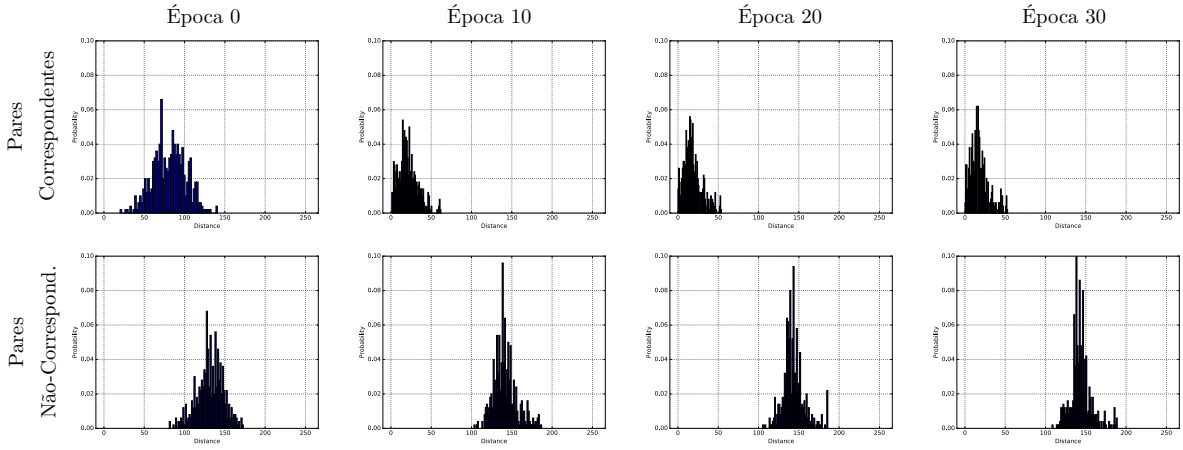


Figura 4.2: Distribuições de distâncias entre pares de *patches* correspondentes e não-correspondentes durante o treinamento da Rede Siamesa. Os resultados da Época 0 foram obtidos antes do treino, com pesos W_i e b_i randômicos, para fins de comparação e análise. Observa-se na primeira linha de histogramas que as distâncias entre os pares correspondentes diminuem durante o treinamento enquanto a linha de baixo mostra que as distâncias entre os pares não-correspondentes vão se concentrando em torno de 125. Estes comportamentos são esperados e indicam que o aprendizado dos testes binários acontece corretamente.

4.4 Mínimos Locais

A seleção de testes binários baseada na vizinhança local apresenta um problema de *Mínimos Locais* devido à distribuição espacial das intensidades dos pixels. Na fase inicial do treinamento as variações horizontais e verticais das coordenadas θ_i reduzem o erro, mas após algumas épocas verifica-se que os vizinhos imediatos $\theta_i + \Delta$ e $\theta_i - \Delta$ não alteram a relação de ordem entre os pixels dos testes binários. Ou seja, ao deslocá-los para a direita, para a esquerda, para cima e para baixo o resultado do teste binário continua sendo o mesmo.

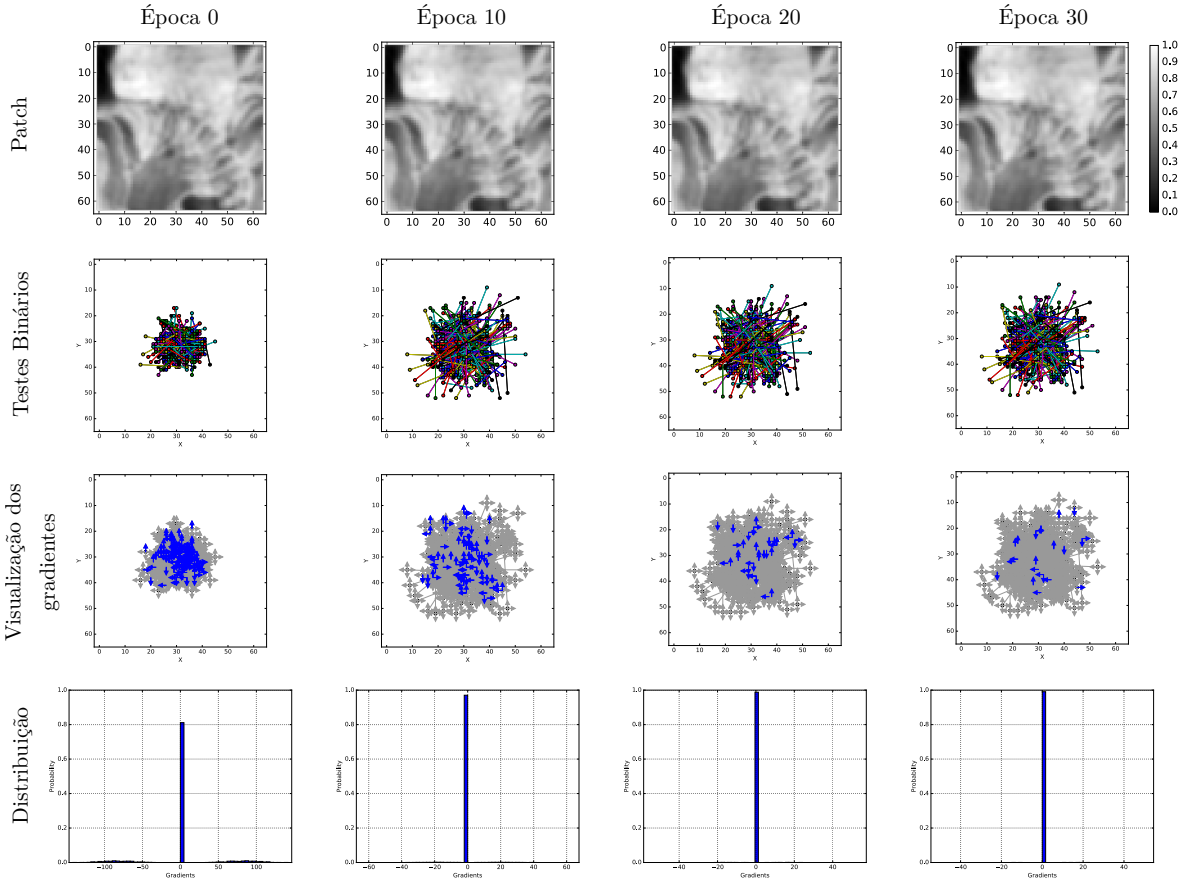


Figura 4.3: Distribuições das componentes dos vetores $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$ durante o treinamento da Rede Siamesa que demonstram a existência de mínimos locais. Os resultados da Época 0 foram obtidos antes do treino, com pesos W_i e b_i randômicos, para fins de comparação e análise. Na primeira linha são apresentados os testes binários de um *patch*. Na segunda linha estão as visualizações das componentes dos vetores $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$ referente a cada teste binário. As setas *azuis* representam valores diferentes de 0 (positivos ou negativos)e indicam que a variação naquela direção e sentido reduz o erro. As setas *cinzas* representam valores iguais a 0 e indicam que não há variação do erro naquela direção. Os histogramas da linha de baixo mostram que a concentração em 0 aumenta durante o treinamento.

A detecção e a análise desse problema foi realizada primeiramente através da observação da distribuição das componentes dos vetores $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$ durante o treinamento. Elas apontam para as direções que reduzem o erro no espaço de otimização. Componentes com valores positivos indicam que o erro diminui deslocando o pixel para a esquerda ou para cima e componentes com valores negativos indicam que o erro diminui deslocando o pixel para a direita ou para baixo. Componentes iguais a 0 indicam que não há variação do erro em qualquer sentido naquela direção.

A Figura 4.3 apresenta os testes binários de um *patch* juntamente com as visualizações e distribuições das componentes dos vetores $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$ durante 30 épocas de treinamento. Os resultados da época 0 são referentes ao estado inicial da Rede Siamesa, com os pesos W_i e b_i inicializados aleatoriamente, para fins de comparação e análise. Observa-se na primeira linha um espalhamento acentuado dos testes binários até a época 10 devido à existência de componentes diferentes de 0 (positivas ou negativas) nos gradientes apresentados na segunda linha de imagens. Enquanto isso, nos histogramas da terceira linha, é possível notar o aumento da concentração das componentes em 0, o que significa que o aprendizado da Rede Siamesa estagnou. Depois da época 30 os pesos W_i e b_i sofrem atualizações tão pequenas que as coordenadas dos testes binários já não são mais alterados.

4.5 Impacto do Treinamento Multiescala

O problema dos mínimos locais acontece porque a Rede Siamesa faz a seleção dos testes binários analisando apenas os vizinhos imediatos dos pixels atuais. Esta visão localizada impede que pixels mais distantes sejam selecionados e estagna o seu aprendizado após algumas épocas de treinamento.

A solução proposta para diminuir esse problema foi realizar o treinamento multiescala, começando da menor resolução até a maior. Os detalhes dessa abordagem são apresentados no Capítulo 3, seção 3.5.

Para simular a resolução espacial 8×8 os *patches* foram convoluídos com um kernel Gaussiano $G(x, y; k = 25, \sigma = 10)$ e divididos em um *grid* regular 8×8 formado por células de tamanho 8×8. A resolução 16×16 foi simulada com um kernel Gaussiano $G(x, y; k = 13, \sigma = 5, 2)$ e um grid 16×16 formado por células de tamanho 4×4. Já a resolução 32×32 foi simulada com um kernel Gaussiano $G(x, y; k = 7, \sigma = 2, 8)$ e um grid 32×32 formado por células de tamanho 2×2.

Com a restrição de selecionar somente um pixel em cada célula do *grid* da respectiva resolução houve maior espalhamento dos testes binários, como mostra a Figura

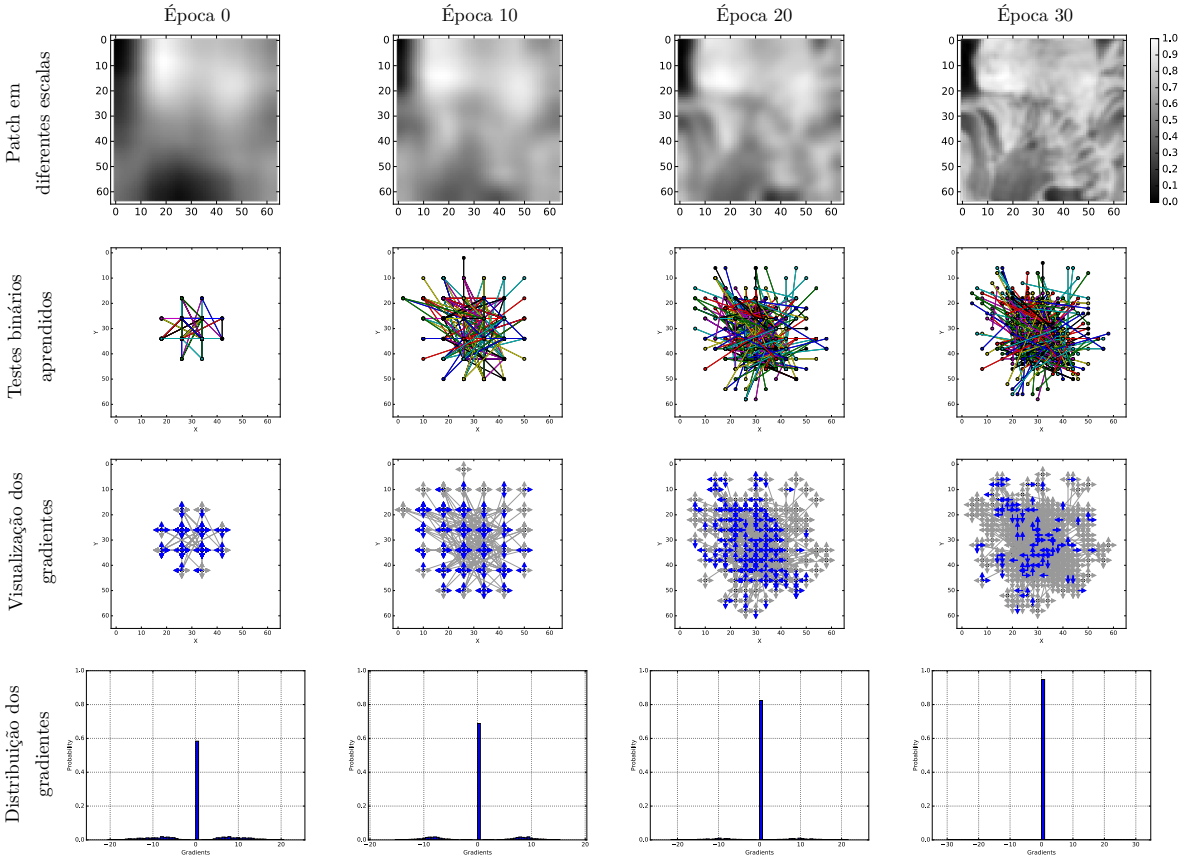


Figura 4.4: Resultados obtidos durante o treinamento com *patches* em diferentes escalas. Os resultados da Época 0 foram gerados antes do treino, com pesos W_i e b_i randômicos, para fins de comparação e análise. A primeira linha mostra um *patch* variando da menor escala até o tamanho original. A segunda linha mostra os testes binários aprendidos ao final da respectiva época. A terceira linha apresenta visualizações dos gradientes de cada teste binário sobre um *patch* de treinamento. Na última linha estão as distribuições das componentes dos gradientes $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$.

4.4. A primeira linha da figura apresenta um *patch* variando de escala a cada 10 épocas de treinamento, começando da resolução 8×8, passando para 16×16, depois 32×32 e finalmente 64×64 (a resolução original). A segunda linha de imagens mostra os testes binários do *patch* exibido, onde é possível observar a regularidade dos pixels selecionados por causa do *grid*. Na terceira linha estão as visualizações dos gradientes a partir dos testes binários selecionados em cada resolução e, por último, a quarta linha traz as distribuições das componentes dos gradientes.

Comparando os gradientes e as distribuições apresentadas na Figura 4.4 com as da Figura 4.3 percebe-se que houve uma redução significativa de componentes iguais a 0 desde o começo do treinamento. Isso quer dizer que a Rede Siamesa encontrou mais pixels que reduziam o erro. Na época 0 da Figura 4.3 havia mais de 80% das

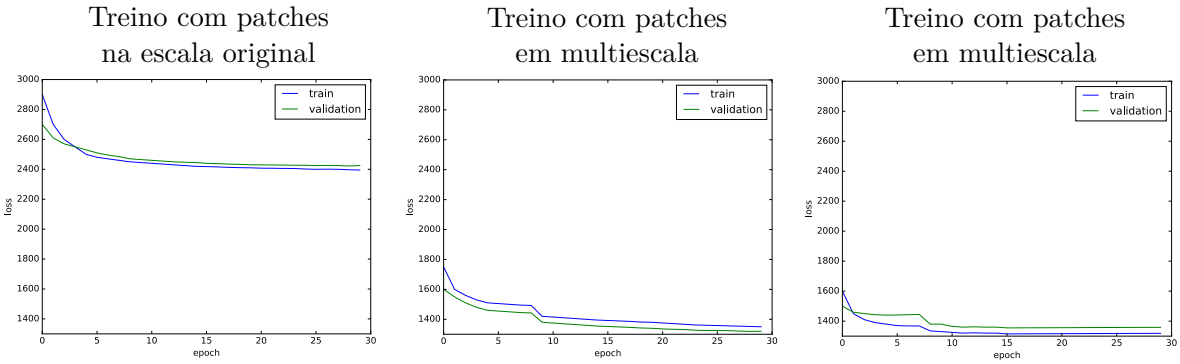


Figura 4.5: Curvas de erro em diferentes experimentos de aprendizado de testes binários, quanto menor melhor. a) Treinamento realizado com *patches* na escala original 64×64. b) Treinamento realizado com *patches* nas escalas 8 e 16, com mudança na época 10 e *learning rate*=1×10^{-9}. c) Treinamento realizado com *patches* nas escalas 8×8 e 16×16, com mudança na época 10 e *learning rate*=1×10^{-8}. Observa-se nas curvas b e c uma queda significativa do erro, principalmente ao variar a escala dos *patches*, o que demonstra que a abordagem proposta reduziu o problema dos Mínimos Locais.

componentes do gradiente com valor igual a 0 enquanto a Figura 4.4 mostra uma concentração de 60% na mesma época. Na época 10 da Figura 4.3 a concentração das componentes iguais a 0 já ultrapassava os 90% enquanto que na mesma época da Figura 4.4 a concentração ainda estava abaixo de 70%.

Essa abordagem permitou à rede ampliar a área de cobertura sobre os *patches* e possibilitou a seleção de pixels mais distantes. Os melhores resultados foram obtidos começando o treinamento com os *patches* na resolução espacial 8×8 e, depois de 10 épocas, fazendo a mudança para a resolução 16×16. A Figura 4.5 mostra três curvas de erro comparando o treinamento sem e com variação de escala. Na primeira curva o erro caiu de 2.900 na época 0 e estabilizou próximo de 2.400 após 30 épocas. Nas outras duas curvas o erro começou próximo de 1.700 e chegou perto de 1.300 depois de 30 épocas. Isso mostra que a abordagem proposta reduziu o impacto dos mínimos locais.

4.6 Componentes Incorretas do Gradiente

Além dos mínimos locais a seleção de testes binários baseada nos vizinhos imediatos apresenta um problema com algumas componentes dos vetores gradientes $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$. Por definição o vetor gradiente $\nabla f(x)$ indica a direção e o sentido de maior crescimento possível da função $f(x)$ a partir de um ponto específico x_i. Para minimizar $f(x)$ o

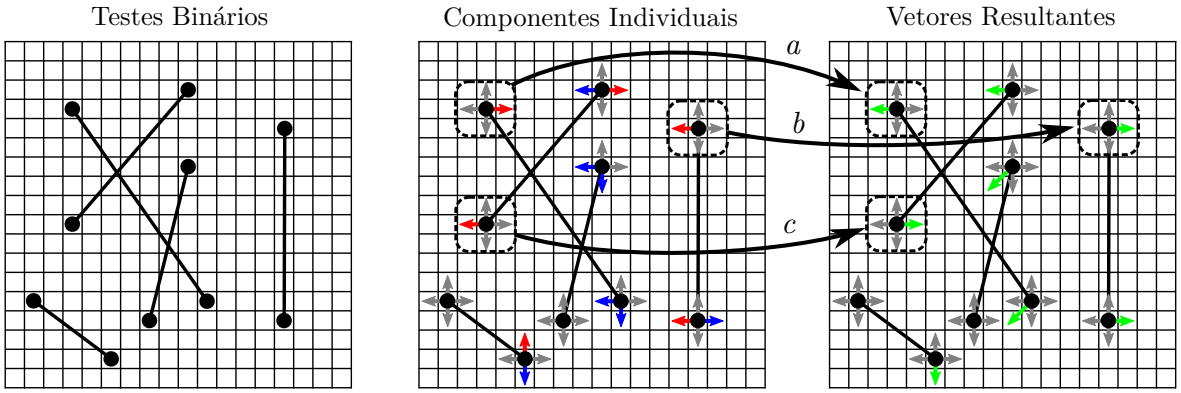


Figura 4.6: Ilustração das componentes incorretas dos gradientes $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$. Ao lado esquerdo estão os testes binários de um *patch* fictício em uma determinada época de treinamento da Rede Siamesa. No centro cada seta indica o comportamento do erro ao variar as coordenadas θ_i dos testes binários nas direções horizontal e vertical. As setas *azuis* indicam que o erro diminui naquela direção, as setas *vermelhas* indicam aumento do erro e as setas *cinzas* indicam que o erro não sofre alteração. No lado direito as setas *verdes* representam os vetores resultantes que indicam as direções de atualização dos pesos W_i e b_i da Rede Siamesa. Os vetores a, b e c são *Componentes Incorretas*, eles não apontam para direções que de fato reduzem o erro porque foram obtidos a partir de componentes individuais que aumentam ou não variam o erro.

algoritmo *Gradiente Descendente*, utilizado no treinamento de Redes Neurais, atualiza os pesos W_i e b_i na mesma direção do gradiente $\nabla f(x)$, mas com sentido oposto. Isso é válido quando $f(x)$ é uma função convexa e o sentido oposto àquele em que obtém-se o maior incremento é justamente o que proporciona o maior decremento de $f(x)$ a partir de x_i, o que não é o caso da seleção dos testes binários.

Em algumas etapas verificou-se que a Rede Siamesa estava mudando os pixels de alguns testes binários em determinado sentido não porque aquele vizinho de fato reduzia o erro, mas simplesmente porque os gradientes $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$ indicavam que a função $L(H)$ crescia no sentido oposto. Este problema é definido neste trabalho como *Problema das Componentes Incorretas do Gradiente* e faz com que o treinamento aconteça de maneira errada.

O gradiente $\frac{\partial L}{\partial \theta_1}$ terá componentes incorretas quando as desigualdades

$$L(H(d_1(x_1, \theta_1 + \Delta), d_2(x_2, \theta_2))) \geq L(H(d_1(x_1, \theta_1), d_2(x_2, \theta_2))) \quad \text{e} \tag{4.1}$$

$$L(H(d_1(x_1, \theta_1 - \Delta), d_2(x_2, \theta_2))) \geq L(H(d_1(x_1, \theta_1), d_2(x_2, \theta_2))) \tag{4.2}$$

forem verdadeiras simultaneamente. Da mesma forma o gradiente $\frac{\partial L}{\partial \theta_2}$ terá componentes

incorretas quando

$$L(H(d_1(x_1,\theta_1),d_2(x_2,\theta_2+\Delta))) \geq L(H(d_1(x_1,\theta_1),d_2(x_2,\theta_2))) \quad \text{e} \tag{4.3}$$

$$L(H(d_1(x_1,\theta_1),d_2(x_2,\theta_2-\Delta))) \geq L(H(d_1(x_1,\theta_1),d_2(x_2,\theta_2))) \ . \tag{4.4}$$

A Figura 4.6 ilustra as componentes dos gradientes $\frac{\partial L}{\partial \theta_1}$ e $\frac{\partial L}{\partial \theta_2}$ de um *patch*, onde é possível visualizar os vetores resultantes que serão utilizados pela Rede Siamesa para atualizar os pesos W_i e b_i.

4.7 Avaliação do Descritor

A avaliação da metodologia de seleção de testes binários proposta nesta dissertação foi realizada através de experimentos de *casamento (matching) de keypoints* entre diferentes imagens obtidas a partir de uma mesma cena. Para isso foram utilizados os

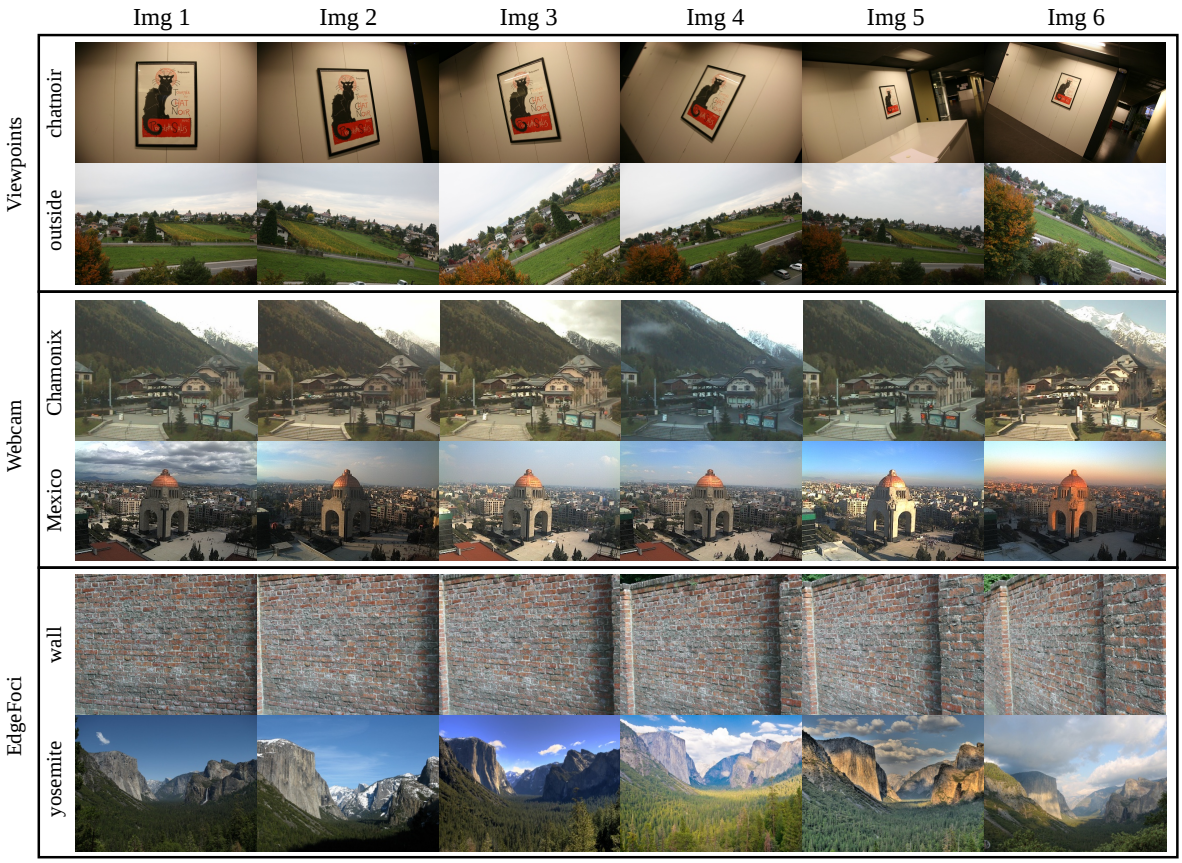


Figura 4.7: Amostras de imagens dos datasets Viewpoints [Yi et al., 2016b], Webcam [Yi et al., 2016b] e EdgeFoci [Ramnath & Zitnick, 2011].

datasets Viewpoints [Yi et al., 2016b], Webcam [Yi et al., 2016b] e EdgeFoci [Ramnath & Zitnick, 2011], cada um deles contendo imagens diferentes de várias cenas. A Figura 4.7 apresenta algumas sequências de imagens dos *datasets* para visualização.

O *dataset* Viewpoints [Yi et al., 2016b] é formado por 30 imagens no total, organizadas em 5 sequências distintas. Cada sequência foi criada a partir de imagens capturadas com variações de pose. Algumas possuem também variações de escala. O dataset EdgeFoci [Ramnath & Zitnick, 2011] contém 38 imagens divididas em 5 sequências, que possuem variações de pose e de iluminação. Já o *dataset* Webcam [Yi et al., 2016b] possui 120 imagens divididas em 6 sequências com variações de iluminação. Elas foram adquiridas com uma câmera em posição fixa ao longo do dia em diferentes estações do ano.

A qualidade do descritor aprendido pela Rede Siamesa foi avaliada primeiramente com a métrica *Matching Score*, definida como a fração de *keypoints* corretamente casados entre duas imagens. O primeiro modelo avaliado, chamado de *BinDescCNN*, foi treinado com os *patches* do dataset Trevi Fountain [Winder & Brown, 2007] no tamanho original 64×64, conforme descrito na Seção 4.2. Já o segundo modelo, chamado de *BinDescCNN MS*, foi treinado com os *patches* em multiescala, conforme descrito na Seção 4.5. Ele apresentou resultados melhores do que o primeiro devido à redução do impacto dos mínimos locais, o que possibilitou uma escolha melhor dos pixels para compor os testes binários. Os resultados obtidos foram comparados com os descritores em ponto-flutuante SURF e SIFT e com os descritores binários BRIEF, ORB, BRISK, FREAK. Para evitar qualquer enviesamento todos os descritores foram avaliados utilizando o ORB como detector de *keypoints*.

A segunda métrica de avaliação utilizada foi a *Area Under the Curve* (AUC) do gráfico *1-Precision* $\times$ *Recall*, calculados como $Precision = \frac{TP}{TP+FP}$ e $Recall = \frac{TP}{TP+FN}$, sendo TP = *True Positive*, FP = *False Positive*, TN = *True Negative* e FN = *False Negative*. *Precision* é uma medida da probabilidade de uma amostra ser classificada corretamente quando um modelo diz que ela pertence a uma determinada classe. *Recall* avalia quantas amostras corretas foram classificadas como corretas. A junção destas medidas permite analisar a qualidade de um conjunto de *features* ao separar duas classes de interesse.

As Figuras 4.8, 4.9 e 4.10 apresentam os resultados obtidos nos *datasets* Viewpoints [Yi et al., 2016b], Webcam [Yi et al., 2016b] e EdgeFoci [Ramnath & Zitnick, 2011], respectivamente. Nelas observa-se que, em geral, o *Matching Score* o modelo baseado em CNN ficou próximo ao dos descritores BRIEF e ORB. Nessa mesma métrica descritor SIFT teve desempenho bem abaixo dos outros e isso se deve à utilização do ORB como detector de *keypoints*. Possivelmente os pontos detectados não sejam

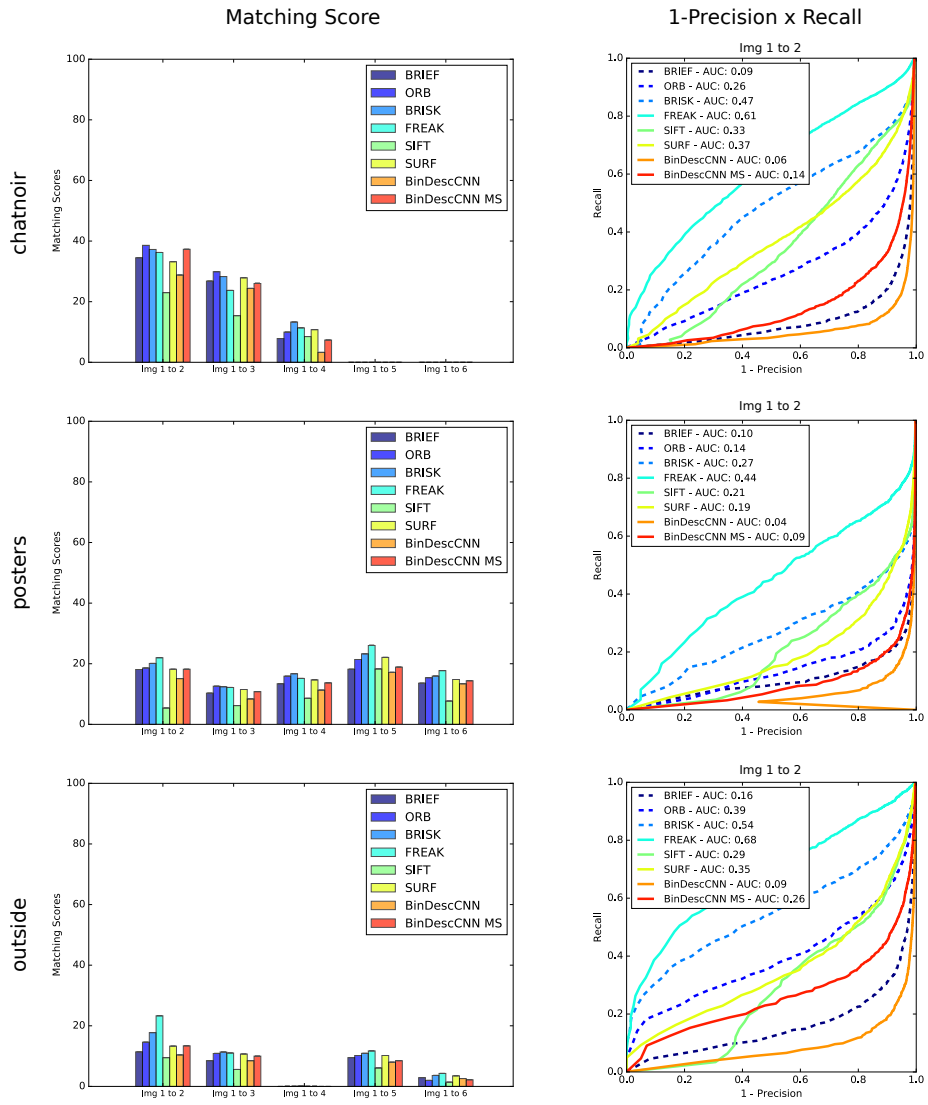


Figura 4.8: Resultados obtidos no dataset Viewpoints [Yi et al., 2016b]. O descritor *BinDescCNN* representa o modelo treinado pela nossa Rede Siamesa no dataset Trevi Fountain [Winder & Brown, 2007] com todos os *patches* no tamanho original 64×64. O descritor *BinDescCNN MS* representa outro modelo treinado no dataset Trevi Fountain com os *patches* em multi-escala. A abordagem de redução de escala apresentou melhores resultados devido à diminuição dos mínimos locais durante o treinamento.

adequados com o SIFT porque não têm distribuição de gradientes discriminativas entre si. Os resultados exibidos nas curvas *1-Precision* $\times$ *Recall* mostram que, em geral, o descritor aprendido pela Rede Convolucional foi melhor apenas que o BRIEF e apresenta bastante confusão ao realizar os *matchings*. Apesar da quantidade razoável de *matchings* corretos foram gerados muitos Falso Positivos e Falso Negativos.

Um aspecto observado é que os descritores gerados com a CNN tiveram comportamentos semelhantes a outros trabalhos diante dos desafios de cada *dataset*. Os

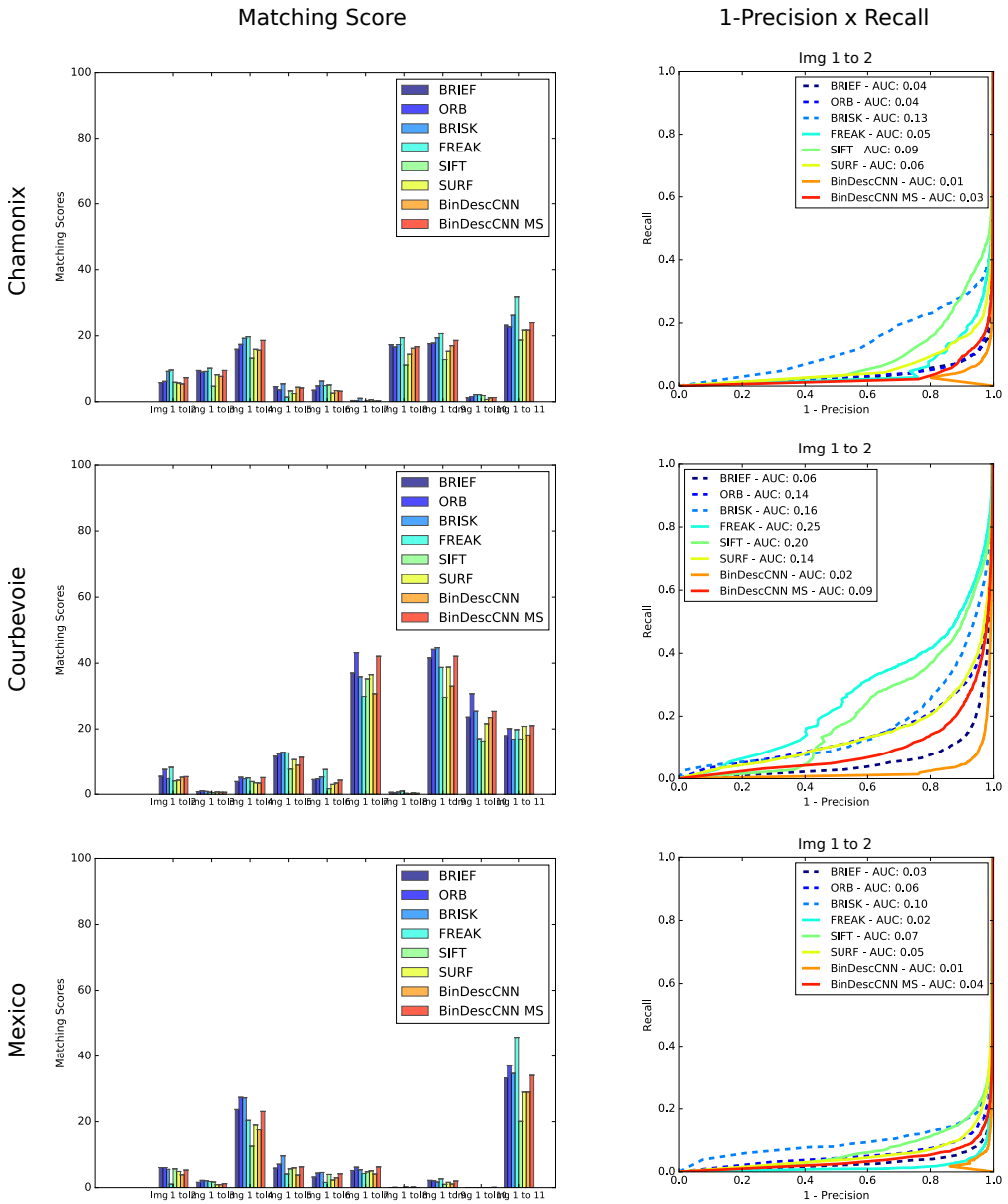


Figura 4.9: Resultados obtidos no dataset Webcam [Yi et al., 2016b]. O descritor *BinDescCNN* representa o modelo treinado pela nossa Rede Siamesa no dataset Trevi Fountain [Winder & Brown, 2007] com todos os *patches* no tamanho original 64×64. O descritor *BinDescCNN MS* representa outro modelo treinado no dataset Trevi Fountain com os *patches* em multi-escala. A abordagem de redução de escala apresentou melhores resultados devido à diminuição dos mínimos locais durante o treinamento.

resultados de todos cresceram ou diminuíram conjuntamente de uma imagem para outra, conforme mostram os *Matching Scores* das Figuras 4.8, 4.9 e 4.10.

A partir de todos esses resultados observa-se que os testes binários gerados pela Rede Convolucional não são suficientemente discriminativos. A existência de Mínimos Locais e de Componentes Incorretas do Gradiente fazem a Rede Siamesa parar de

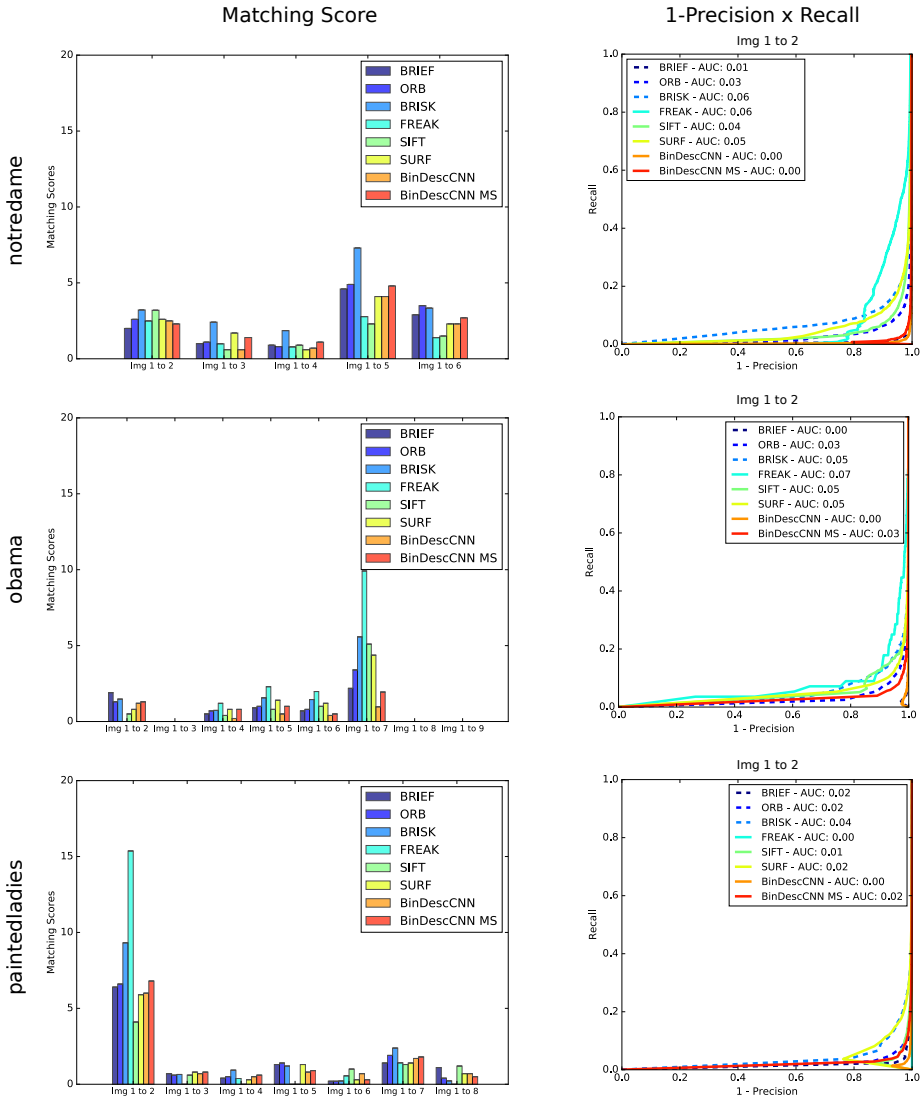


Figura 4.10: Resultados obtidos no dataset EdgeFoci [Ramnath & Zitnick, 2011]. O descritor *BinDescCNN* representa o modelo treinado pela nossa Rede Siamesa no dataset Trevi Fountain [Winder & Brown, 2007] com todos os *patches* no tamanho original 64×64. O descritor *BinDescCNN MS* representa outro modelo treinado no dataset Trevi Fountain com os *patches* em multi-escala. A abordagem de redução de escala apresentou melhores resultados devido à diminuição dos mínimos locais durante o treinamento.

aprender a partir de um determinado ponto, dificultando uma separação melhor entre pares de *patches* correspondentes e pares não-correspondentes. Isso gera descritores binários com bits correlacionados e com baixa variância, impactando diretamente a sua qualidade. Ter bits correlacionados e pouco variados significa que o descritor possui muita redundância e não utiliza todo o poder de representação que possui de acordo com o seu comprimento. Haverá, portanto, muitos erros na realização de *matchings*.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 Conclusão

O desenvolvimento deste trabalho foi motivado pela busca de melhorias na seleção de testes binários para descrever pontos de interesse de imagens. A partir dos avanços obtidos nesta década utilizando Redes Neurais Convolucionais (CNN) para resolver diversos problemas de Visão Computacional decidiu-se aplicá-las na busca de uma nova distribuição de testes binários para criar um novo descritor.

Para avaliar a capacidade dessas redes selecionarem testes binários mais discriminativos do que os descritores BRIEF [Calonder et al., 2010], ORB [Rublee et al., 2011], BRISK [Leutenegger et al., 2011] e FREAK [Ortiz, 2012] foi construída uma Rede Siamesa treinada com pares de patches correspondentes e pares não-correspondentes.

A modelagem proposta foi capaz de reduzir as distâncias entre descritores de patches correspondentes e aumentar as distâncias entre descritores de patches não-correspondentes, mas durante a realização de experimentos foram encontrados problemas que limitam a utilização de CNNs nesse contexto quando o gradiente é calculado a partir da vizinhança local dos pixels. Os problemas são descritos detalhadamente na dissertação como contribuições à comunidade científica.

Além de avaliar a capacidade das CNNs no problema de seleção de testes binários consideramos relevante a aproximação numérica proposta para treinar a Rede Siamesa utilizando a distância de Hamming como métrica de similaridade. É uma função não-derivável analiticamente e a formulação apresentada pode ser aplicada em outros contextos.

O treinamento multiescala mostrou ser capaz de reduzir o impacto dos Mínimos Locais devido à ampliação da visão da Rede Siamesa sobre os *patches*. Essa abordagem também pode ser utilizada no aprendizado de outros descritores para explorar

propriedades relacionadas ao tamanho do *keypoint*.

A conclusão é que nem sempre uma abordagem baseada em CNN será capaz de encontrar uma distribuição espacial de testes binários que minimize as distâncias entre *keypoints* correspondentes e maximize as distâncias entre *keypoints* não-correspondentes. Isso depende da formulação e da modelagem do problema. Os resultados apresentados não provam a impossibilidade do uso de Redes Convolucionais no problema de seleção de testes binários, apenas esclarecem algumas limitações quando são utilizadas somente informações da vizinhança local dos pixels.

5.2 Trabalhos Futuros

Para contornar o problema dos Mínimos Locais uma estratégia possível seria realizar um pré-processamento nos *patches* de modo a extrair regiões invariantes e computar os testes binários a partir delas ao invés de comparar pixels individualmente. Isso é feito no descritor OSRI [Xu et al., 2014] e poderia permitir a criação de uma formulação convexa para fazer a seleção de testes binários com uma CNN.

Outra estratégia possível seria utilizar uma abordagem probabilística para fazer a seleção dos pixels quando o erro da Rede Siamesa estabilizar durante o treinamento. A partir de uma distribuição Normal bivariada podem ser escolhidas direções randômicas x e y que possibilitem a escolha de pontos mais distantes.

Com um custo computacional mais elevado, poderia ser adotada também a estratégia de computar todos os testes binários possíveis dentro de cada *patch* e fazer a seleção dos mais discriminativos independente da vizinhança local. O maior desafio neste caso seria definir uma métrica de avaliação para decidir qual teste binário é melhor quando vários deles produzirem a mesma redução do erro.

Referências Bibliográficas

Balntas, V.; Tang, L. & Mikolajczyk, K. (2015). Bold - binary online learned descriptor for efficient image matching. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bay, H.; Tuytelaars, T. & Gool, L. V. (2006). Surf: Speeded up robust features. Em *In ECCV*, pp. 404--417.

Brown, M. (2018). Autostitch: a new dimension in automatic image stitching.

Calonder, M.; Lepetit, V.; Strecha, C. & Fua, P. (2010). Brief: Binary robust independent elementary features. Em *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pp. 778--792, Berlin, Heidelberg. Springer-Verlag.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. Em *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pp. 886--893, Washington, DC, USA. IEEE Computer Society.

Duan, Y.; Lu, J.; Wang, Z.; Feng, J. & Zhou, J. (2017). Learning deep binary descriptor with multi-quantization. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Girshick, R.; Donahue, J.; Darrell, T. & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Em *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pp. 580--587, Washington, DC, USA. IEEE Computer Society.

Hare, J. S.; Samangooei, S. & Dupplaw, D. P. (2011). Openimaj and imageterrier: Java libraries and tools for scalable multimedia analysis and indexing of images. Em *Proceedings of the 19th ACM international conference on Multimedia*.

He, K.; Zhang, X.; Ren, S. & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

Hosseini, H.; Xiao, B.; Jaiswal, M. & Poovendran, R. (2017). Deep neural networks do not recognize negative images. *CoRR*, abs/1703.06857.

Karn, U. (2016). An intuitive explanation of convolutional neural networks.

Krizhevsky, A.; Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em Pereira, F.; Burges, C. J. C.; Bottou, L. & Weinberger, K. Q., editores, *Advances in Neural Information Processing Systems 25*, pp. 1097--1105. Curran Associates, Inc.

Leutenegger, S.; Chli, M. & Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. Em *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pp. 2548--2555, Washington, DC, USA. IEEE Computer Society.

Lin, K.; Lu, J.; Chen, C.-S. & Zhou, J. (2016). Learning compact binary descriptors with unsupervised deep neural networks. Em *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91--110. ISSN 0920-5691.

Mur-Artal, R.; Montiel, J. M. M. & Tardós, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956.

Nguyen, A. M.; Yosinski, J. & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. Em *CVPR*, pp. 427–436. IEEE Computer Society.

Nielsen, M. (2017). Using neural nets to recognize handwritten digits.

Ojala, T.; Pietikäinen, M. & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59. ISSN 0031-3203.

Ortiz, R. (2012). Freak: Fast retina keypoint. Em *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pp. 510--517, Washington, DC, USA. IEEE Computer Society.

Parkhi, O. M.; Vedaldi, A. & Zisserman, A. (2015). Deep face recognition. Em *British Machine Vision Conference*.

Ramnath, K. & Zitnick, C. L. (2011). Edge foci interest points. Em *2011 IEEE International Conference on Computer Vision (ICCV 2011)(ICCV)*, volume 00, pp. 359–366. ISSN .

Rosten, E. & Drummond, T. (2006). Machine learning for high-speed corner detection. Em *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pp. 430--443, Berlin, Heidelberg. Springer-Verlag.

Rublee, E.; Rabaud, V.; Konolige, K. & Bradski, G. (2011). Orb: An efficient alternative to sift or surf. Em *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pp. 2564--2571, Washington, DC, USA. IEEE Computer Society.

Sabour, S.; Frosst, N. & Hinton, G. E. (2017). Dynamic routing between capsules. *CoRR*, abs/1710.09829.

Salomon, J. (2018). *Lung Cancer Detection using Deep Convolutional Networks Final Year Project Report*. Tese de doutorado.

Sharma, A. (2017). Understanding activation functions in deep learning.

Simo-Serra, E.; Trulls, E.; Ferraz, L.; Kokkinos, I.; Fua, P. & Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. Em *2015 IEEE International Conference on Computer Vision (ICCV)*, volume 00, pp. 118–126. ISSN 2380-7504.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Em *In Proc. of International Conference on Learning Representations (ICLR)*.

Su, J.; Vargas, D. V. & Sakurai, K. (2017). One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864.

Sweeney, C. (2016). Theia multiview geometry library: Tutorial and reference.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions.

Wang, L. & He, D.-C. (1990). Texture classification using texture spectrum. *Pattern Recognition*, 23(8):905 – 910. ISSN 0031-3203.

Winder, S. & Brown, M. (2007). Learning local image descriptors. Em *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR07)*, Minneapolis.

Xu, X.; Tian, L.; Feng, J. & Zhou, J. (2014). OSRI: A rotationally invariant binary descriptor. *IEEE Trans. Image Processing*, 23(7):2983--2995.

Yi, K. M.; Trulls, E.; Lepetit, V. & Fua, P. (2016a). LIFT: Learned Invariant Feature Transform. Em *Proceedings of the European Conference on Computer Vision*.

Yi, K. M.; Verdie, Y.; Fua, P. & Lepetit, V. (2016b). Learning to Assign Orientations to Feature Points. Em *Proceedings of the Computer Vision and Pattern Recognition*.

Yosinski, J.; Clune, J.; Nguyen, A.; Fuchs, T. & Lipson, H. (2015). Understanding neural networks through deep visualization. Em *Deep Learning Workshop, International Conference on Machine Learning (ICML)*.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B. & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization.

Zhou, B.; Lapedriza, A.; Xiao, J.; Torralba, A. & Oliva, A. (2014). Learning deep features for scene recognition using places database. Em Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D. & Weinberger, K. Q., editores, *Advances in Neural Information Processing Systems 27*, pp. 487--495. Curran Associates, Inc.