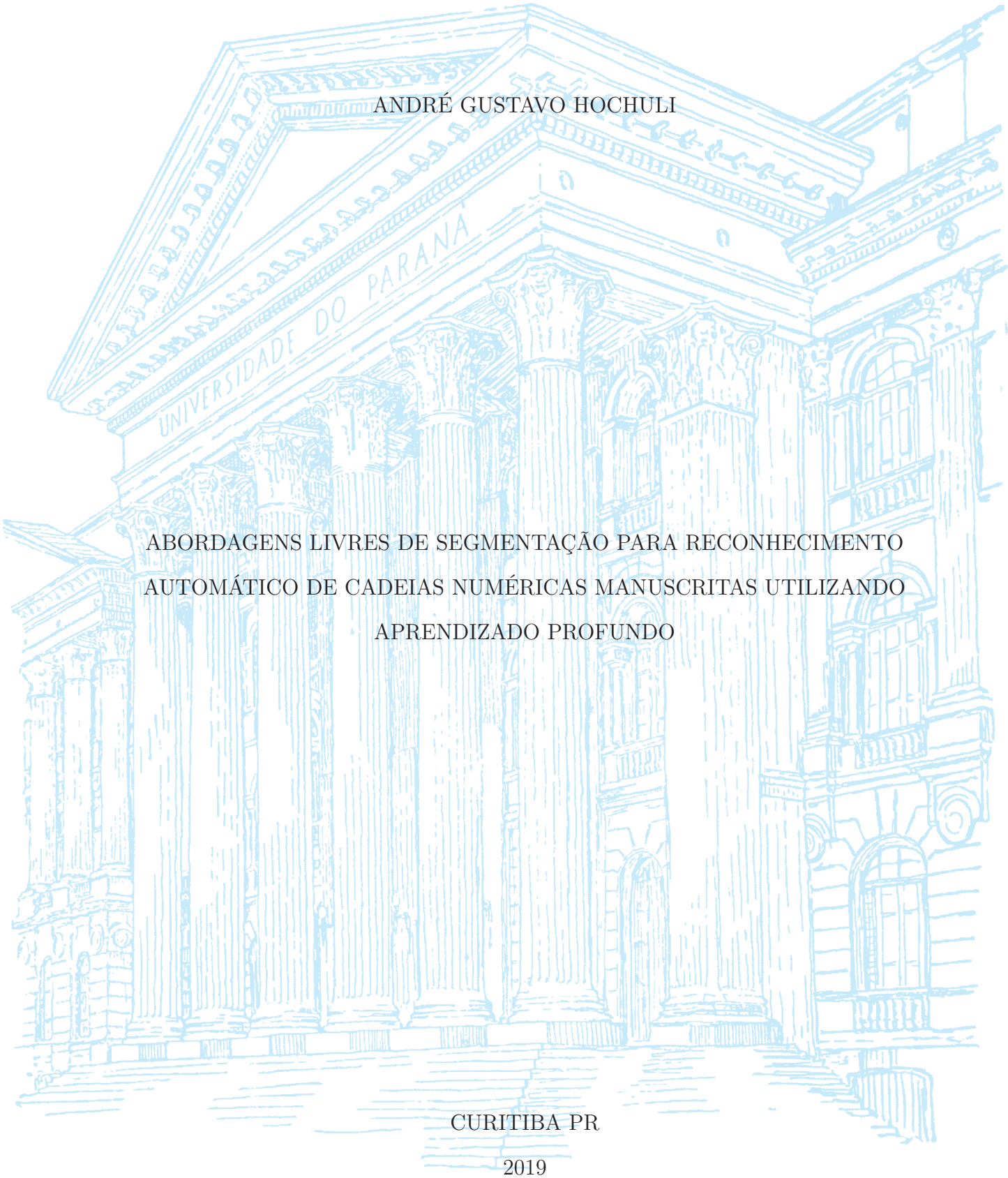
UNIVERSIDADE FEDERAL DO PARANÁ

ANDRÉ GUSTAVO HOCHULI



ABORDAGENS LIVRES DE SEGMENTAÇÃO PARA RECONHECIMENTO AUTOMÁTICO DE CADEIAS NUMÉRICAS MANUSCRITAS UTILIZANDO APRENDIZADO PROFUNDO

CURITIBA PR

2019

ANDRÉ GUSTAVO HOCHULI

ABORDAGENS LIVRES DE SEGMENTAÇÃO PARA RECONHECIMENTO AUTOMÁTICO DE CADEIAS NUMÉRICAS MANUSCRITAS UTILIZANDO APRENDIZADO PROFUNDO

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof Dr. Luiz Eduardo Soares de Oliveira.

Coorientador: Prof. Dr. Robert Sabourin.

CURITIBA PR

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

H685a Hochuli, André Gustavo
Abordagens livres de segmentação para reconhecimento automático de cadeias numéricas manuscritas utilizando aprendizado profundo [recurso eletrônico] / André Gustavo Hochuli. – Curitiba, 2019.

Tese - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2019.

Orientador: Luiz Eduardo Soares de Oliveira – Coorientador: Robert Sabourin.

1. Sistemas de reconhecimento de padrões. 2. Redes Neurais (Computação). 3. Inteligência artificial. 4. Sequências (Matemática). I. Universidade Federal do Paraná. II. Oliveira, Luiz Eduardo Soares de. III. Sabourin, Robert. IV. Título.

CDD: 006.42

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **ANDRÉ GUSTAVO HOCHULI** intitulada: **ABORDAGENS LIVRES DE SEGMENTAÇÃO PARA RECONHECIMENTO AUTOMÁTICO DE CADEIAS NUMÉRICAS MANUSCRITAS UTILIZANDO APRENDIZADO PROFUNDO**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 12 de Março de 2019.

LUIZ EDUARDO SOARES DE OLIVEIRA
Presidente da Banca Examinadora

GEORGE DARMITON DA CUNHA CAVALCANTI
Avaliador Externo (UFPE)

ROBERT SABOURIN
Avaliador Externo (ETS)

DAVID MENOTTI GOMES
Avaliador Interno (UFPR)

ALCEU DE SOUZA BRITTO JR
Avaliador Externo (PUCPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Pós-Graduação em Informática

À minha família e amigos.

AGRADECIMENTOS

Agradeço aos meus pais, Maria de Lourdes e Edenir Mário (*In memoriam*), aos meus irmãos Alexandre e Cristiano pelo apoio durante esse tempo de estudo. Também agradeço aos meus padrinhos Fernando Girardi e Celina Girardi.

Em especial, à minha namorada e companheira Kellin Demeterko, que me apoiou incondicionalmente nas decisões desse período e me motivou a essa conquista.

Aos meu orientadores Prof. Dr. Luiz S. Oliveira, Prof. Dr. Alceu Britto e Prof. Dr. Robert Sabourin pela excelente condução deste trabalho, auxiliando em todas as dúvidas dentro e fora do contexto da pesquisa, além do exemplo de profissionalismo os quais tomo como inspiração para minha carreira. Também agradeço aos professores do departamento que compartilharam com dicas valiosas durante os cafés e reuniões. Em especial ao Prof. Dr. Eduardo Cunha, ao Prof. Dr. Lucas Ferrari e ao Prof. Dr. David Menotti.

Aos membros da banca expresso aqui meus sinceros agradecimentos pelas valiosas contribuições.

Por fim, mas não menos importante, um agradecimento aos colegas do departamento pelas variadas conversas, muitas vezes descontraídas, as quais com certeza me ajudaram a enxergar diferentes pontos de vista e evoluir como ser humano.

Em uma atitude de Fé, agradeço a Deus por me conduzir nessa jornada e me dar forças para conquistar o objetivo.

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original"
(Albert Einstein)

RESUMO

Nas últimas décadas, o reconhecimento de cadeias numéricas manuscritas tem sido abordado de maneira similar por vários autores, no que se refere ao tratamento de dígitos conectados. A necessidade de segmentar esses componentes é um consenso. Dessa forma, as propostas se concentram em determinar os pontos de segmentação aplicando heurísticas sobre características extraídas do objeto, plano de fundo, contorno, entre outras. No entanto, a produção de dígitos fragmentados, ocasionando a sobre-segmentação da cadeia, é um problema comum entre essas abordagem. Assim, as metologias são categorizadas pela forma como manipulam os componentes resultantes desse processo: (a) Naquelas que produzem apenas uma segmentação possível, ou (b) naquelas que definem um conjunto de hipóteses de segmentação, além de um método de fusão para determinar a hipótese mais provável. Apesar da segunda categoria apresentar taxas de reconhecimento mais elevadas, o custo computacional torna-se um aspecto desfavorável, devido as recorrentes consultas ao classificador pelas inúmeras hipóteses produzidas. Além disso, a variabilidade dos tipos de conexão entre os dígitos e a falta de contexto, como a informação sobre a quantidade de dígitos, denotam a limitação de abordagens baseadas em processos heurísticos. Visando evitar estes problemas, evidenciamos ser possível superar os métodos tradicionais implementando modelos baseados em aprendizado profundo para classificar dígitos conectados diretamente, reduzindo a etapa de segmentação a um processo de detecção de componente conexo. Além disso, aproveitando os avanços na área de detecção de objetos, apresentamos uma nova abordagem para o problema, na qual, dígitos passam a ser compreendidos como objetos em uma imagem e neste cenário, uma sequência de dígitos é uma sequência de objetos. Para validar nossas hipóteses, experimentos realizados em bases de conhecimento geral avaliaram nossas propostas com os trabalhos presentes na literatura em termos de reconhecimento, correta segmentação e custo computacional. Os resultados atingiram taxas de reconhecimento em torno 97% quando aplicado a uma base de duplas de dígitos conectados e 95% para as amostras de cadeias da base *NIST SD19*, superando os níveis do estado da arte. Além das altas taxas de reconhecimento, também houve significativa redução de consultas ao classificador (custo computacional), principalmente em casos complexos, superando o desempenho dos trabalhos presentes no estado da arte, denotando o potencial das abordagens propostas.

Palavras-chave: Reconhecimento de Cadeias Manuscritas, Segmentação de Dígitos Manuscritos, Reconhecimento de Padrões, Aprendizado Profundo, Redes Neurais Convolucionais

ABSTRACT

Over the last decades, the recognition of handwritten digit strings has been approached in a similar way by several authors, regarding the connected digits issue. The segmentation of these components is a consensus. In this way, the approaches attempt to determining the segmentation points by applying heuristics on features extracted from the object, background, contour, etc. However, the production of fragmented digits, causing the over-segmentation of the string is a common problem among these approaches. Thus, the methodologies are categorized by the way they manipulate the components resulting from this process: (a) those ones that produce only a possible segmentation, or (b) those ones that define a set of segmentation hypotheses and a fusion method to determine the best hypothesis. Although the second category has higher recognition rates, the computational cost becomes an unfavorable aspect, due to the recurrent classifier calls to classify the hypotheses produced. Therefore, the variability of the connection types and the lack of context, such as the number of digits present in the string, denote the limitation of approaches based on heuristic processes. In order to avoid these problems, we believe that is possible to overcome traditional methods by implementing models based on deep learning to classify connected digits directly, reducing the segmentation step to a connected component detection process. In addition, taking advantage of advances of object detection field, we propose a new approach to the problem, in which, digits are understood as objects in an image and in this scenario, a sequence of digits is a sequence of objects. To validate our hypotheses, experiments were carried out in well-known datasets, evaluating our proposals against state-of-art in terms of recognition, correct segmentation and computational cost. The results achieved recognition rates of 97% when applied to a base of connected digit pairs, and 95% for the NIST SD19 samples, surpassing state-of-art levels. Besides the high recognition rates, it has a significant reduction in terms of classifier calls (computational cost), especially in complex cases, surpassing the performance of the works present in the state of the art, denoting the potential of the proposed approaches.

Keywords: Handwritten Digit String Recognition, Digit Segmentation, Pattern Recognition, Deep Learning, Convolutional Neural Network

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AG	Algoritmo Genético
CNN	Redes Neurais Convolucionais (Convolutional Neural Network)
DSM	Modelo de Cadeia Numérica (Digit String Model)
MLP	Perceptron Multi-Camadas (Multi-Layer Perceptron)
HMM	Cadeias Escondidas de Markov (Hidden Markov Models)
IA	Inteligência Artificial
LEM	Modelo de Estimativa do Tamanho (Length Estimator Model)
LSTM	Redes de Memória Longa e Curta (Long-Short Term Model)
Max	Valor máximo de um conjunto de dados
Min	Valor mínimo de um conjunto de dados
MQDF	Função Quadrática Discriminante Modificada (Modified Quadratic Discriminant Function)
N/A	Não aplicável
N/D	Não disponível
ReLu	Unidade de Retificação Linear (Rectified Linear Unit)
RNA	Rede Neural Artifical (Artificial Neural Network)
RNN	Rede Neural Recorrente (Recurrent Neural Network)
DBN	Rede de Confiança Profunda (Deep Belief Network)
RBF	Função de Base Radial (Radial Basis Function)
SVM	Máquina de Vetores de Suporte (Support Vector Machine)

LISTA DE SÍMBOLOS

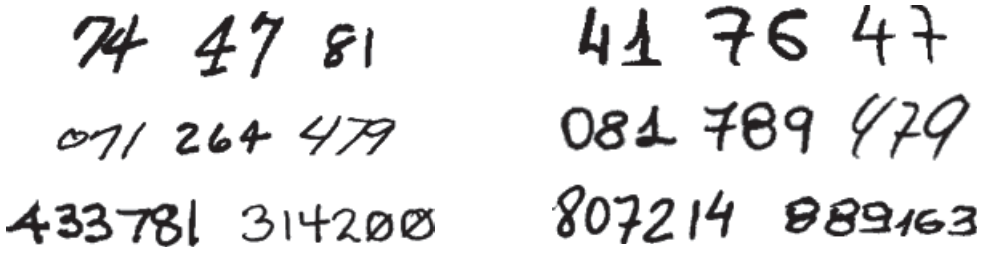
$\mathcal{C}$	Classificador
$\mathcal{C}_1$	Classificador de Dígitos Isolados
$\mathcal{C}_2$	Classificador de Duplas de Dígitos Conectados
$\mathcal{C}_3$	Classificador de Triplas de Dígitos Conectados
$\mathcal{C}_{1110}$	Classificador de 1110 Classes de Dígitos
$\mathcal{L}$	Classificador de Estimativa de Tamanho
CC	Componente Conexo.
ω	Número de classes de um problema.
p	Probabilidade *a posteriori*.
t	Limiar (*threshold*).

SUMÁRIO

1 INTRODUÇÃO

O reconhecimento de cadeias numéricas manuscritas, por meio de algoritmos computacionais, é uma área de pesquisa com grande atividade e vasta aplicação, destacando-se a leitura automática de formulários, e de maneira mais específica, códigos postais e documentos bancários. Com o crescimento do poder computacional ao longo dos anos e consequentemente, a viabilidade de algoritmos de reconhecimento de padrões mais eficazes, altas taxas de desempenho vêm sendo alcançadas.

No entanto, a interpretação automática de cadeias numéricas não é uma tarefa simples. Implementar um classificador capaz de reconhecer a cadeia como um todo torna-se, na maioria das vezes, inviável devido a gama de variáveis inerentes ao processo. Além da variabilidade de escrita e características regionais, conforme ilustra a Figura 1.1, a falta de léxico em numerais aumenta a complexidade do problema. Para exemplificar, reconhecer numerais de até quatro dígitos, consiste em um problema de dez mil combinações possíveis (0000 até 9999), isso sem contabilizar os numerais contendo dois (00-99) e três dígitos (000-999).



(a) (b)

Figura 1.1: Exemplo da variabilidade de escrita em cadeias numéricas: (a) Base Americana (*NIST*) e (b) Base Brasileira (Cheques Bancários). Nota-se em especial, diferentes grafias para os numerais '0', '1', '4' e '7'.

Para contornar este problema, as metodologias disponíveis na literatura (Ribas et al., 2012; Kulkarni e Vasambekar, 2010; Saba et al., 2010) concentram-se em dividir a solução em duas etapas[1]: segmentar a cadeia em dígitos isolados (segmentação), processo ilustrado na Figura 1.2, e então reconhecê-los (classificação). Sendo assim, na etapa de segmentação, os principais desafios consistem em definir a quantidade de dígitos que compõem a cadeia (tamanho) e a correta segmentação destes, inclusive daqueles conectados entre si. Para a etapa de classificação, de maneira geral, um classificador treinado com exemplos de dígitos isolados, define uma classe entre 0 a 9 para cada hipótese de segmentação gerada. É evidente que uma estratégia capaz de segmentar os dígitos com precisão é crucial para a eficácia da solução, pois do contrário, os ruídos remanescentes desta etapa tornarão o processo de classificação impreciso.

Neste cenário, os algoritmos de segmentação são classificados em duas grandes classes (Casey e Lecolinet, 1996): Segmentação-e-Reconhecimento; ou Segmentação-Baseada-no-Reconhecimento.

Na primeira, os pontos de segmentação são baseados em heurísticas. Comumente, os parâmetros são definidos empiricamente a partir de características explícitas como

[1] As etapas podem ser complementares, por exemplo, utilizar o resultado da classificação para avaliar uma dada hipótese de segmentação.

(a) (b)

Figura 1.2: Processo de Segmentação: (a) Cadeia Numérica e (b) Hipótese de segmentação.

tamanho, contorno, informações do plano de fundo do objeto, análises de histogramas, projeção de vales e topos, dentre outras. Por fim, os dígitos segmentados são classificados. Na segunda, os pontos de segmentação são definidos a partir da função de desempenho do classificador, tornando a decisão implícita. Primeiramente, os potenciais pontos de corte são definidos utilizando-se de características similares às citadas acima, porém as sub-regiões (dígitos segmentados) resultantes destes pontos são classificadas e atribuída uma probabilidade *a posteriori* para cada uma, definindo assim hipóteses de segmentação. Posteriormente, as múltiplas hipóteses (sobre-segmentação) são organizadas em sequências hierárquicas, as quais sugerem as possíveis classes. Enfim, de acordo com uma estratégia de combinação, é então selecionada a melhor sequência que irá definir o numeral presente na cadeia.

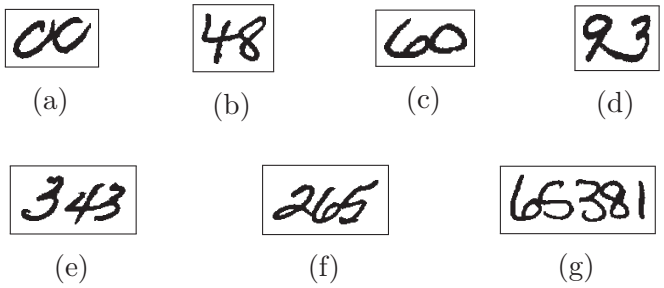
Conforme descrito na Seção 3 deste documento, a literatura apresenta melhores taxas para as estratégias de segmentação-baseada-no-reconhecimento. Contudo, a desvantagem desta solução é que, as inúmeras consultas ao classificador para validar as hipóteses aumentam o custo computacional da solução. Além disso, um processo adicional de análise e combinação dos melhores resultados é necessário.

1.1 DEFINIÇÃO DO PROBLEMA

Reconhecer cadeias numéricas, conforme exposto anteriormente, é um processo que envolve a segmentação e o reconhecimento de forma isolada dos dígitos que compõem a cadeia. Dentro deste processo, a segmentação permanece um problema em aberto.

O objetivo da etapa de segmentação é isolar os dígitos da cadeia numérica para que estes possam ser identificados entre classes de 0 a 9, diminuindo a complexidade do problema. Em um primeiro momento parece um processo simples, como por exemplo, dividir a cadeia em componentes conexos. Porém, a complexidade aumenta quando dois ou mais dígitos estão conectados entre si (Figura 1.3), ou seja, não é evidente em termos computacionais a região de separação entre eles. Estes casos são denominados de "dígitos conectados" e, se não houver um tratamento, podem ser classificados como um dígito isolado comprometendo o reconhecimento da cadeia. Para ilustrar este ponto, Wang et al. (2000) analisaram 200.000 cadeias numéricas extraídas de correspondências do sistema de correio americano (USPS). Esse estudo revela que 15,5% dos componentes apresentam dígitos conectados, dos quais, 13,1% são conexões entre duplas de dígitos e 2,4% contendo três ou mais. Composições com o dígito '0' são as mais comuns.

Podemos encontrar na literatura uma variedade de algoritmos de segmentação que exploram diferentes características do contorno para fornecer potenciais pontos de corte para uma dada cadeia de tamanho desconhecida. Uma comparação interessante entre as abordagens é apresentada em Ribas et al. (2012), na qual é possível observar que, uma alternativa para reduzir as heurísticas necessárias para encontrar os pontos de segmentação são as estratégias baseadas em sobre-segmentação. Estas segmentam a cadeia o máximo possível, produzindo componentes que podem representar dígitos ou fragmentos

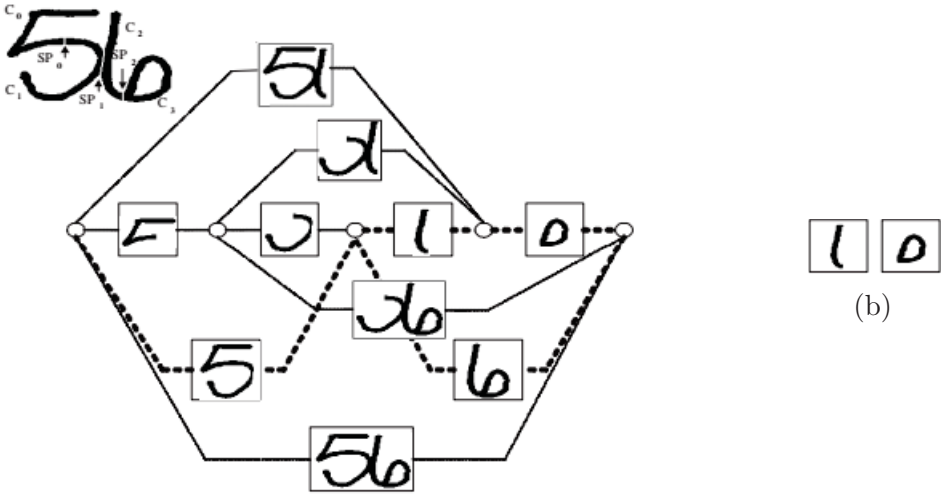


(a) (b) (c) (d)

(e) (f) (g)

Figura 1.3: Exemplo de Dígitos Conectados.

destes. Após classificar cada componente, os algoritmos buscam pela combinação que otimiza o resultado. O processo de sobre-segmentação do numeral "56" é ilustrado na Figura 1.4. De fato, a razão por trás dessa abordagem é maximizar as chances de gerar a correta segmentação, no entanto, arcando com o alto custo computacional de segmentar e reconhecer cada componente, até mesmo aqueles que são fragmentos.



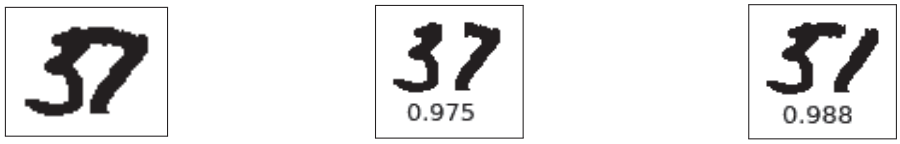
(a) (b)

Figura 1.4: (a) Grafo de segmentação para a cadeia "56" e (b) Imagens que podem ser confundidas com as classes "0" e "1" (Vellasques et al., 2008).

Nestes casos, a definição automática do tamanho da cadeia (quantidade de dígitos) torna-se uma informação importante no processo de fusão, auxiliando o algoritmo a filtrar ruídos e fragmentos de dígitos, evitando que falsos positivos sejam incorporados ao resultado final.

Além disso, ao contrário do reconhecimento de palavras manuscritas, a falta de contexto nos numerais, o que permitiria a criação de dicionários e regras semânticas, aumenta a complexidade do problema, não havendo nenhuma sugestão de qual o próximo dígito a ser analisado. Sem o contexto, uma incidência de falsos positivos provenientes de segmentações errôneas torna difícil a implementação de um método de rejeição. A Figura 1.5 ilustra esse caso. Em alguns casos, o uso de heurísticas baseadas em observações sobre os tipos de conexões tornam a solução especializada, muitas vezes não reproduzindo o mesmo desempenho para casos similares.

Para superar estes obstáculos, alguns métodos recorrem a estratégias livres-de-segmentação, nas quais os componentes conexos (CC's) são reconhecidos diretamente



(a) 37 (b) 3-7 (c) 5-1

Figura 1.5: Hipóteses de segmentação: (a) Par Conectado, (b) Hipótese Correta e (c) Falso Positivo

sem a necessidade de segmentação em dígitos isolados. Este tipo de abordagem tem recuperado a atenção da comunidade nos últimos anos motivada pela introdução das técnicas de aprendizado profundo. Enquanto, as estratégias baseadas em segmentação demandam uma heurística para gerar os potenciais cortes, um classificador robusto de dígitos isolados e uma estratégia para combinar as inúmeras hipóteses, as estratégias livres-de-segmentação demandam uma volumosa quantidade de dados para o correto aprendizado das representações.

1.2 OBJETIVOS

No que diz respeito ao reconhecimento de cadeias numéricas, notamos uma predominância de métodos baseados em sobre-segmentação (de S. Britto et al., 2003; de Oliveira et al., 2002; Liu et al., 2004; Sadri et al., 2007). Estes trabalhos implementam diferentes esquemas de pré-processamento, segmentação, classificação e fusão. No entanto, todos possuem uma forte dependência sobre o algoritmo de segmentação. Para evitar a ausência do correto ponto de segmentação, a sobre-segmentação é geralmente empregada, mesmo que com a carga do alto custo computacional.

Uma leitura mais profunda da literatura revela que os avanços apresentados pelos algoritmos de aprendizado profundo (Bengio et al., 2013; Gu et al., 2017) definiram novas fronteiras em diferentes áreas como reconhecimento de caracteres (Xiao et al., 2017; Laroca et al., 2018, 2019), reconhecimento de palavras (Roy et al., 2016; Tamen et al., 2017; Y. Wua e Liu, 2017), identificação de escrita (Ziyong et al., 2017), verificação de assinaturas (Hafemann et al., 2017), além de segmentação e reconhecimento de íris (Luz et al., 2017; Zanlorensi et al., 2018). No entanto, o reconhecimento de cadeias numéricas continua limitado pelas barreiras apresentadas pelos algoritmos de segmentação.

A respeito disso, algumas questões são pertinentes: Precisamos continuar dependendo de algoritmos de segmentação? Por que não aproveitamos os avanços que ocorreram no campo de aprendizado de máquina e tornamos o reconhecimento de dígitos manuscritos menos dependente destes algoritmos? Alguns trabalhos da literatura (Ciresan et al., 2012; Liao e Carneiro, 2017; Sabour et al., 2017) mostram que redes neurais profundas são capazes de alcançar desempenhos próximos aos dos humanos na base de referência MNIST para dígitos manuscritos e em outros problemas como o reconhecimento de objetos em uma imagem (He et al., 2016; Girshick et al., 2014; Redmon e Farhadi, 2017).

Assim sendo, o objetivo principal deste trabalho é propor abordagens livres de segmentação para o problema de reconhecimento de cadeias numéricas manuscritas. As metodologias exploram o conceito de Aprendizado Profundo (*Deep Learning*), o qual tem sido amplamente aplicado nos campos de processamento de imagens e reconhecimento de padrões (Lecun et al., 2015; Schmidhuber, 2015), atingindo elevadas taxas de reconhecimento. Sob esse conceito, acreditamos ser possível aperfeiçoar as taxas de reconhecimento de cadeias numéricas, eliminando a segmentação de dígitos conectados. Para tal, propomos

a aplicação de redes neurais convolucionais, capazes de determinarem implicitamente características e parâmetros do problema, como tamanho dos componentes e as classes dos dígitos. É importante ressaltar que, apesar de abordagens baseadas em aprendizado profundo necessitarem de grandes volumes de dados para atingirem resultados satisfatórios, a geração de dados sintéticos é viável em cadeias numéricas manuscritas (Choi e Oh, 1999; Ribas et al., 2012; Oliveira et al., 2005), possibilitando o uso deste conceito.

Em paralelo ao esforço para atingir o objetivo principal, elencamos abaixo alguns objetivos secundários:

- Definir uma metodologia capaz de estimar a quantidade de dígitos presentes em um componente conexo proveniente de uma cadeia numérica.
- Classificar componentes contendo duplas e triplas de dígitos conectados sem a necessidade de segmentá-los.
- Definir arquiteturas modulares, baseada em aprendizado de máquina, de modo que seja evitada e/ou reduzido a necessidade da aplicação de algoritmos de segmentação.
- Descrever uma metodologia eficaz para integração dos módulos da arquitetura proposta (fusão).
- Validar o uso de métodos baseados em aprendizado profundo em reconhecimento de numerais, aplicando este conceito na implementação dos módulos propostos, caracterizando de forma clara suas vantagens e desvantagens.
- Desenvolver um modelo baseado em detecção de objetos, sem restrições quanto ao tamanho da cadeia.

1.3 DESAFIOS

Desenvolver métodos de reconhecimento de cadeias numéricas eficientes, requer a habilidade de tratar os seguintes problemas:

- Dificuldade em determinar o tamanho de um numeral.
- Variabilidade de escrita de um mesmo dígito, como por exemplo, características regionais e inclinação.
- Presença de dígitos conectados entre si.
- Falta de contexto e vocabulário, como em palavras, não permitindo qualquer indicativo de qual o próximo dígito a ser reconhecido.

1.4 HIPÓTESE

Considerando o exposto até o momento, definimos como hipótese de que é possível aprender a variabilidade de dígitos conectados utilizando aprendizagem profunda e assim reconhecer cadeias numéricas de qualquer tamanho sem a necessidade de segmentação. Sendo assim, será possível desenvolver sistemas mais eficazes que aqueles baseados em heurísticas presentes no estado da arte.

1.5 CONTRIBUIÇÕES

Os esforços empregados neste trabalho resultaram nas seguintes contribuições:

- Construção de bases sintéticas de cadeias numéricas, produzidas a partir de dígitos isolados, contendo variabilidade de dígitos conectados, fornecendo dados suficientes para o aprendizado da representação pelos modelos utilizados neste trabalho.
- Um método baseado em seleção dinâmica de classificadores, composto de três modelos capazes de reconhecer dígitos isolados, duplas e triplas conectados, além de um modelo capaz de determinar a quantidade de dígitos conectados (tamanho), publicado em Hochuli et al. (2018a).
- Um método capaz de reconhecer 1110 classes de dígitos a partir de um único modelo, sendo, 10 classes de dígitos isolados (0..9), 100 classes de duplas conectadas (00..99) e 1000 classes de triplas (000..999), publicado em Hochuli et al. (2018b).
- Uma solução de ponta-a-ponta, baseada em detecção de objetos, capaz de reconhecer cadeias numéricas sem a restrição quanto ao tamanho de cadeias conectadas, reduzindo a complexidade quando comparado aos métodos acima citados. Os resultados desta abordagem foram submetidos para revisão em periódico.

1.6 ESTRUTURA DO DOCUMENTO

O documento está organizado da seguinte maneira: no Capítulo 2 será descrito a base teórica relativa ao processo de segmentação e reconhecimento, como por exemplo, técnicas de aprendizado de máquina utilizado para o reconhecimento do dígito. A revisão do estado da arte, discutindo os principais trabalhos da área, serão expostos no Capítulo 3. As metodologias propostas para reconhecer cadeias numéricas serão apresentadas no Capítulo 4. Fechando o documento, os experimentos serão expostos no Capítulo 5, as conclusões e o trabalho futuro no Capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresentamos uma introdução ao conceitos de Redes Neurais Artificiais (Seção 2.1), Aprendizado Profundo (Seção 2.2), Redes Neurais Convolucionais (Seção 2.3), Redes Neurais Recorrentes (Seção 2.4) e ao *framework* YoLo (Seção 2.5), as quais são abordagens chaves para a nossa metodologia.

2.1 REDES NEURAIS ARTIFICIAIS

Na área de aprendizado de máquina, Redes Neurais Artificiais (RNA's) são bastante difundidas e com vasta aplicação. Um neurônio artificial (perceptron - Rosenblatt (1958)) é constituído pelo seus pesos e *bias*, além da função de ativação que determina se o neurônio está ativo ou não para um dado conjunto de características fornecido como entrada, funcionando assim como um classificador binário linear, conforme ilustra a Figura 2.1. Seguindo esse raciocínio, uma RNA ou "Perceptron Multi-Camadas" (*Multi-Layer Perceptron - MLP*), distribui uma certa quantidade de neurônios artificias interconectados ao longo de uma rede, formando um fluxo unidirecional (*feed-forward*) entre as camadas (Riedmiller, 1994). As camadas intermediárias, também conhecidas como camadas "escondidas", são responsáveis por mapear os dados de um vetor de entrada em um vetor para a camada de saída, através da ativação ou não de neurônios, funcionando assim como extratores de características implícitos. Os parâmetros de cada neurônio (pesos, *bias* e ativação) nas camadas intermediárias, são atualizados por um algoritmo de retro-propagação (*Back Propagation*), em função do erro calculado na camada de saída.

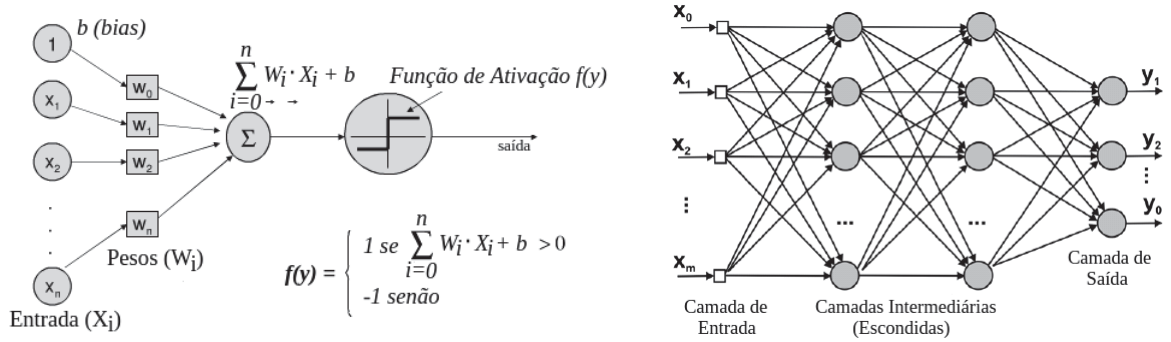


Figura 2.1: (a) Neurônio Artificial (Perceptron) e (b) Rede Neural Artificial ou Perceptron Multi-Camadas.

2.2 APRENDIZADO PROFUNDO (*DEEP LEARNING*)

Um processo de aprendizado de máquina consiste em definir um conjunto de características para representar um determinado problema, e posteriormente, aplicar um modelo capaz de classificá-las. Deste modo, a representação adequada dos dados definem uma importante etapa deste processo. Basicamente, os métodos tradicionais consistem em predefinir descritores de características desenvolvidos por um humano, ou seja, algoritmos computacionais transformam características de percepção humana

em compreensão de máquina. Assim, esforços têm sido realizados para que técnicas de extração de características representem qualitativamente os dados do problema (*Feature Engineering*), aumentando assim as taxas de reconhecimento dos modelos (Bengio et al., 2013). No entanto, esse tipo de abordagem têm limitado o nível de abstração devido a complexidade dos dados em alguns problemas, tornando pobre a representação para a máquina, e consequentemente, estagnando o avanço das taxas de reconhecimento. Além disso, esse processo é empírico e custoso. Para contornar estes problemas, viu-se a necessidade de criar modelos capazes de extrair as características automaticamente, ao compreendimento da máquina, transformando a descrição pré-definida de características em modelos de descritores treináveis (*Feature Learning*). A Figura 2.2 ilustra a diferença entre as metodologias.

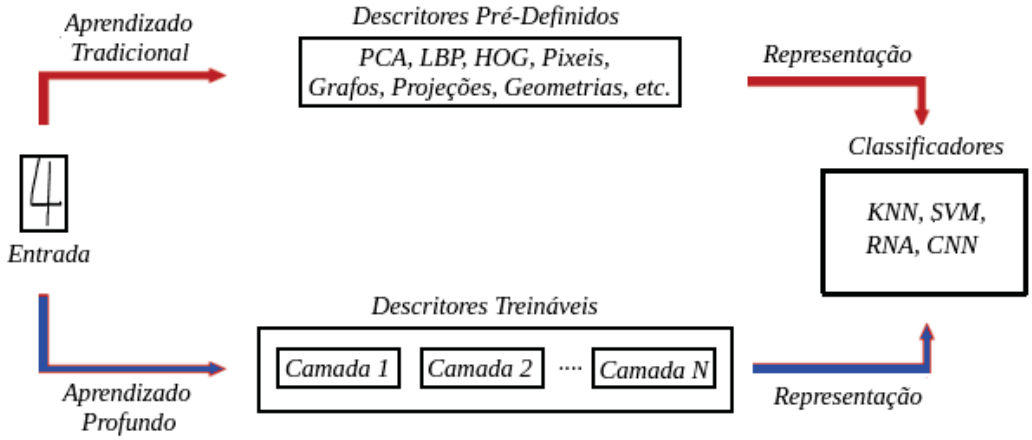


Figura 2.2: Processo de Aprendizado Tradicional e Profundo.

Sendo assim, o conceito de aprendizado profundo (Lecun et al., 2015; Schmidhuber, 2015) define arquiteturas densas, treináveis, compostas por várias camadas (Figura 2.3), de forma que a própria máquina consiga extrair características particulares do problema. O dado é repassado entre as camadas, e as camadas mais profundas da arquitetura tendem a representá-lo de forma mais abstrata, ou seja, uma representação mais próxima da máquina do que a apresentada na primeira camada. Alguns exemplos dessas arquiteturas (Bengio, 2009; Deng e Yu, 2014) são: (a) Rede Neurais e Redes Neurais Convolucionais (detalhada na seção 2.3), utilizadas em Aprendizado Supervisionado, e (b) Máquinas de Boltzmann, Rede de Confiança Profunda (*Deep Belief Network*) e Auto-Codificadores (*Auto-Encoder*), aplicadas para mapeamento das características, geralmente aplicadas em Aprendizado Não-Supervisionado. Vale ressaltar que uma abordagem deste tipo, devido a sua densa arquitetura, requer um volume grande de dados para o treinamento eficiente do algoritmo, caso contrário, a solução tenderá a ficar especializada (*overfitting*). É evidente também que essas abordagens têm um custo computacional elevado, devido à quantidade de parâmetros para ajuste e dados processados.

Neste trabalho, exploraremos o potencial de Redes Neurais e Redes Convolucionais, sob o conceito de aprendizado profundo, para o problema proposto. Um detalhamento sobre Redes Convolucionais é apresentado na Seção 2.3.

[1]Fonte: http://neuralnetworksanddeeplearning.com/chap6.html, acessada em 01/02/2017.

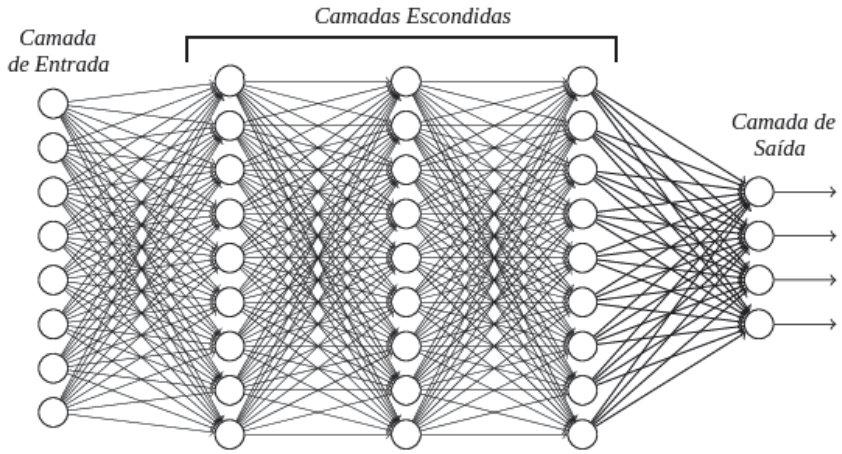


Figura 2.3: Aprendizado Profundo: Estrutura de uma Rede Neural Densa[1].

2.3 REDES NEURAIS CONVOLUCIONAIS (*CONVOLUTIONAL NEURAL NETWORK*)

Como visto na Seção 2.2, uma abordagem baseada em aprendizado profundo (arquiteturas densas) eleva os níveis de abstração dos dados de forma a se aproximar da compreensão pela máquina. Seguindo por este raciocínio, Redes Neurais Convolucionais (*Convolutional Neural Network - CNN*), introduzem o conceito de auto-aprendizagem de filtros, a fim de criar extratores de características para problemas representados de forma matricial, como por exemplo, reconhecimento de padrões em imagens e espectrogramas de áudio (LeCun et al., 2010; Lecun et al., 2015).

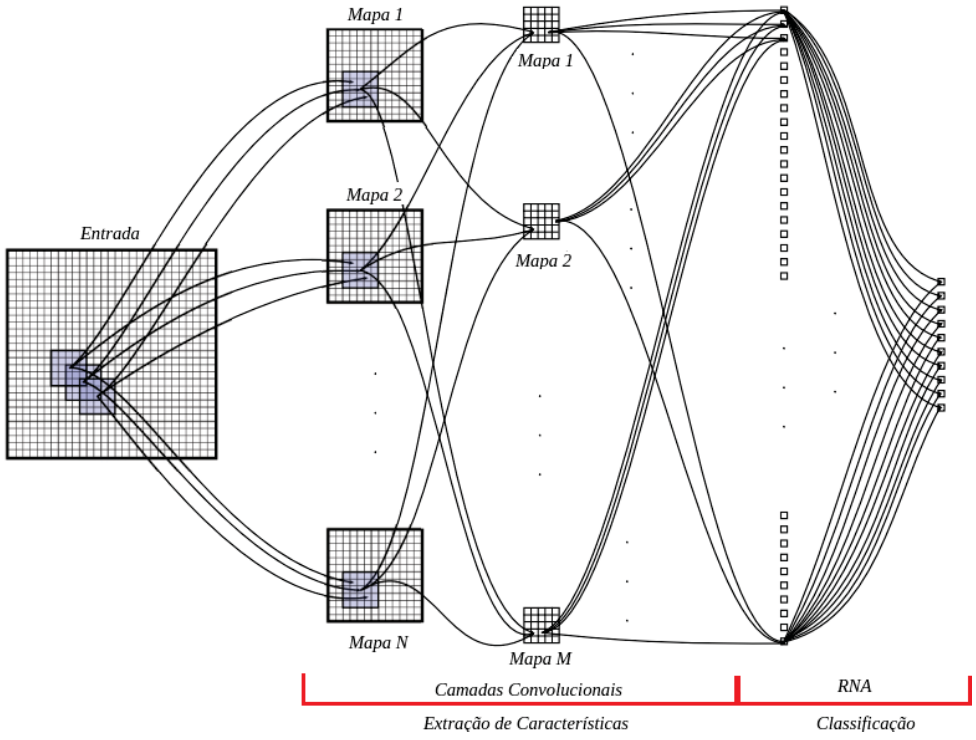


Figura 2.4: Exemplo de arquitetura de uma Rede Neural Convolucional. Os retângulos dentro da matriz representam os filtros/convoluções. (Figura adaptada de (Ciresan et al., 2011))

Basicamente, existem dois grandes estágios em uma rede convolucional, conforme ilustra a Figura 2.4. O primeiro é responsável por extrair características de uma matriz de dados, comumente uma imagem, fornecida como entrada para a rede. Sua arquitetura é composta de camadas convolucionais (Ciresan et al., 2011), as quais implementam uma sequência de filtros, ativações (ex: ReLU/Sigmoid) e agregações (*pooling's*), mapeando os dados de entrada em novos mapas de características (*feature maps*), criando assim descritores treináveis, abstraindo os dados de acordo com a compreensão da máquina. O detalhamento destes processos se dará a seguir, nas Seções 2.3.1, 2.3.2 e 2.3.3. Ao final desse estágio teremos um modelo que transforma uma matriz de entrada em um vetor de características. A Figura 2.5 apresenta um exemplo de abstração dos dados ao longo da rede.

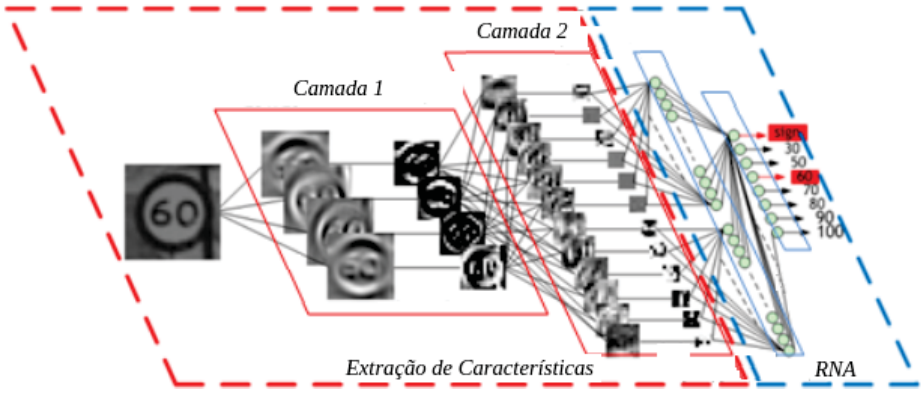


Figura 2.5: Níveis de abstração dos dados em uma rede convolucional de duas camadas[1].

No segundo estágio, o vetor de características produzido pelas camadas convolucionais é aplicado a um algoritmo de aprendizagem de máquina, como uma RNA ou um *SVM*, determinando assim uma classe para o padrão de entrada (classificação).

Na fase de testes, a imagem de entrada é fornecida à arquitetura da rede neural convolucional, com os pesos e filtros das respectivas camadas calculados durante a fase de treinamento, a qual determinará a classe final do padrão.

Este tipo de abordagem tem elevado os níveis de desempenho do estado da arte, principalmente na última década, antes estagnados pela falta de métodos para representar adequadamente o problema, criando assim um novo marco para as pesquisas nas área de Inteligência Artificial, Reconhecimento de Padrões e Visão Computacional. Dentre as arquiteturas mais conhecidas, citamos os trabalhos de LeCun et al. (1989); Lecun et al. (1998), pioneiros na aplicação de redes convolucionais para reconhecimento de dígitos manuscritos e os recentes trabalhos de Krizhevsky et al. (2012) e Szegedy et al. (2015), trabalhos vencedores nos anos de 2012 e 2014 do desafio Imagenet (ILSVRC) (Russakovsky et al., 2015a), o qual apresenta uma base contendo mais de 1,2 milhões de imagens e 1000 classes.

2.3.1 Convoluções

Um processo de convolução compreende à aplicação de um filtro de dimensões (*kernel*), pesos e passo (*stride*) parametrizáveis. A operação consiste em deslizar o filtro sobre a matriz de entrada, sendo que, a soma do produto dos pesos em cada iteração, gera

[1]Fonte: https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/, acessada em 01/02/2017.

um novo dado na matriz resultante ou um novo mapa de características (*feature maps*), conforme ilustra a Figura 2.6. As dimensões do mapa de características gerado é definido pela Equação 1.

$$M_w = \frac{I_w - K_w}{S_w + 1} + 1; \qquad M_h = \frac{I_h - K_h}{S_h + 1} + 1 \tag{1}$$

tal que M, I, K e S denotam o mapa, a entrada, o filtro e o passo, enquanto que w e h representam, os eixos de largura e a altura, respectivamente.

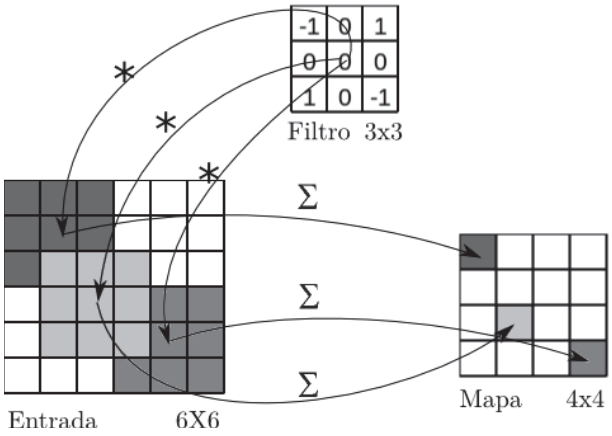


Figura 2.6: Convolução de uma entrada 6x6 utilizando um filtro 3x3, ilustrado em diferentes regiões da imagem.

Desta forma, as convoluções que compõem uma *CNN*, compreendem um conjunto de N filtros, capazes de serem treinados para descreverem automaticamente as características da camada de entrada, gerando N mapas na camada seguinte. Cada neurônio presente no mapa, está conectado à um subconjunto de neurônios do mapa/camada anterior, também chamado de Região Receptiva (*Receptive Field*), determinado pela área abrangida pelo filtro. De forma análoga aos *perceptrons*, os filtros têm seus pesos atualizados a cada iteração da rede por um algoritmo de retro-propagação, em função do erro na camada de saída. Assim, ao longo da rede, os filtros mapeiam novos mapas de caraterísticas para cada camada. Um exemplo de filtros aprendidos por uma *CNN* pode ser visualizados na Figura 2.7.

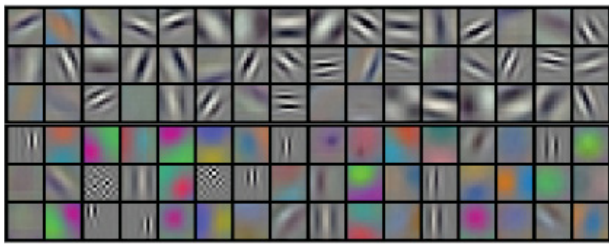


Figura 2.7: Filtros aprendidos na primeira camada da arquitetura proposta por (Krizhevsky et al., 2012).

2.3.2 Funções de Ativação

Uma função de ativação determina se um neurônio está ativo ou não, de acordo com o valor de entrada. Em arquiteturas densas, com aprendizado baseado em descida do gradiente e retro-propagação, como o caso de uma *CNN*, a saturação ou desaparecimento

do gradiente (*Vanish/Explode Gradient*) torna-se um problema. De outro modo, um neurônio saturado ou nulo, não repassa informação para as camadas mais profundas da rede, acarretando em um aprendizado mais lento ou um aprendizado fraco (mínimos locais). Este comportamento é comum no uso de funções de ativação de crescimento logístico (Schmidhuber, 2015), como as sigmoides ou tangente hiperbólica (tanH), devido ao fato destas normalizarem suas saídas, conforme ilustra a Figura 2.8.

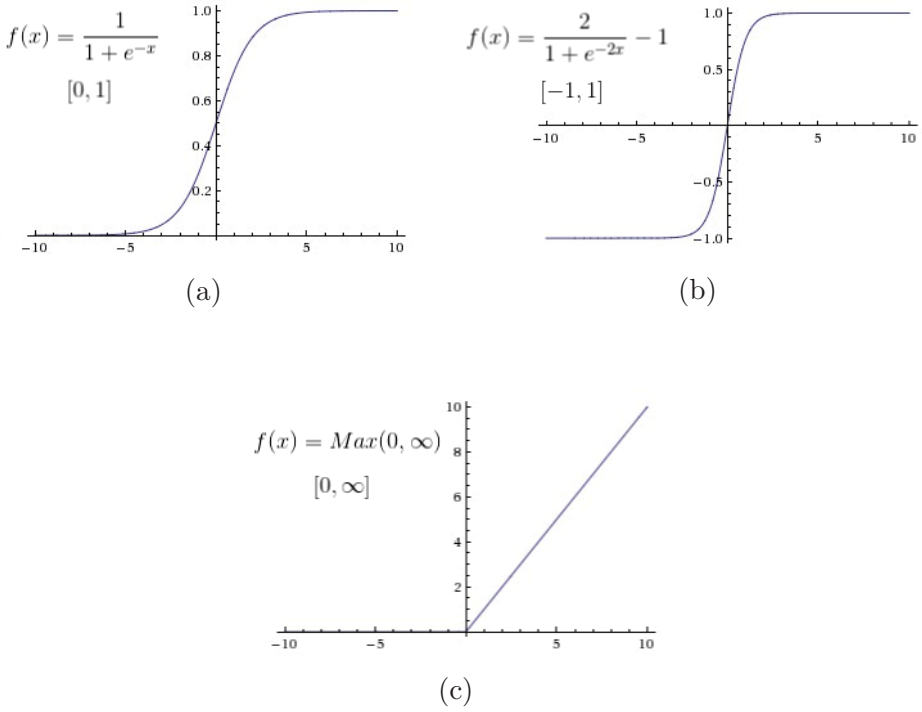


Figura 2.8: Funções de Ativação: (a) Sigmoide, (b) Tangente Hiperbólica e (c) ReLU.

Para contornar este problema, o uso da função *ReLU* (*Rectified Linear Unit* ou Unidade de Retificação Linear), mostrou-se eficiente em redes neurais convolucionais, aumentando em seis vezes a convergência da rede (Figura 2.9), tornando um marco no aprendizado de redes profundas (Krizhevsky et al., 2012). Basicamente a função determina um limiar em zero ($f(x) = MAX(0, x)$). Outro fator importante é que, devido a simplicidade da função quando comparada as funções Sigmoide/*TanH* (threshold x operações exponenciais), o custo computacional da rede torna-se menor.

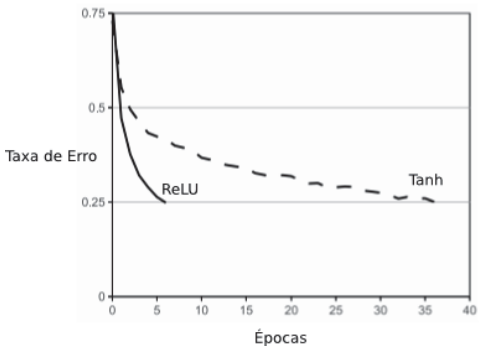


Figura 2.9: Convergência da rede para as funções *ReLU*(linha sólida) e *Tanh*(linha tracejada) (Krizhevsky et al., 2012).

2.3.3 Agregação (*Pooling*)

Uma camada de agregação tem como objetivo reduzir a dimensão do problema e selecionar características de forma invariante à translação. Outro objetivo inerente é a redução da quantidade de parâmetros (neurônios, pesos, etc), evitando a especialização da rede (*overfitting*). O processo consiste em deslizar uma janela quadrada (MxM) sobre a imagem, sem que exista sobreposição das regiões. Para isso é utilizado um passo S, tal que $S >= M$. Para cada região, extrai-se uma única característica, utilizando uma regra de agregação.

De modo geral, uma agregação é inserida após uma camada convolucional. Os parâmetros comumente utilizados são janelas de tamanho 2x2 e passo 2, reduzindo pela metade a dimensão do problema na transição das camadas. A regra de agregação mais utilizada é a do valor máximo (*Max Pooling*), a qual extrai o maior valor da região. Um exemplo desse processo é ilustrado na Figura 2.10.

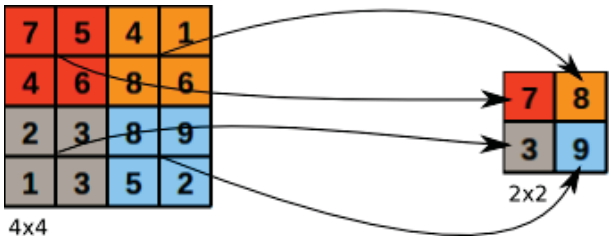


Figura 2.10: Agregação de uma matriz 4x4, utilizando uma Janela 2x2 com Passo 2 e regra do Máximo Valor, resultando em uma redução de escala de fator 2.

Vale ressaltar, que existem outras regras de agregação como média ou normalização euclidiana (L^2).

2.4 REDES NEURAIS RECORRENTES (RNN)

Quando se trata de sequências, como por exemplo, processamento de linguagem natural, séries temporais, reconhecimento de voz e sentenças manuscritas, entre outras, a informação sobre o passado é de suma importância para a criação de contexto e classificação. Embora as RNA's sejam capazes de discriminar uma enorme quantidade de classes, a sua mecânica não considera a informação produzida em um estado anterior. Uma Rede Neural Recorrente (RNN) (Elman, 1990; Rumelhart et al., 1986; Lipton, 2015), ilustrada na Figura 2.11, resolve esse problema implementando um laço, no qual a informação produzida é re-introduzida no modelo.

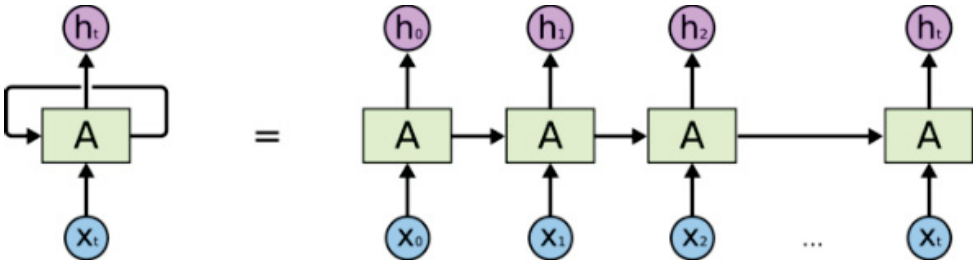


Figura 2.11: RNN: A cada passo a rede (A) recebe como entrada a informação nova (X_t) e a informação produzida no passo anterior (h_{t-1}).[1]

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Seja A uma rede neural, X_t uma sequência qualquer de tamanho t, a rede irá produzir h_t estados, considerando sempre a informação produzida no estado h_{t-1}. De outro modo, as RNN's podem ser compreendidas como múltiplas cópias da mesma rede, cada uma repassando a informação produzida para a sua sucessora, adicionando uma espécie de memória na rede.

As RNN's se mostraram bastante eficazes no tratamento de sequências, no entanto enfrentam problema quando a informação precisa persistir por um longo tempo. Esse problema é explorado em Bengio et al. (1994); Pascanu et al. (2013), revelando que funções baseadas em minimização do gradiente tendem a perder a informação ao longo das RNN's (*Vanish Gradient Problem*), fazendo com que as RNN's apresentem uma memória curta em relação ao contexto. Para solucionar este problema foram criadas as Redes de Memória Curta e Longa que será descrita na Seção 2.4.1.

2.4.1 Redes de Memória Curta e Longa (LSTM)

As Redes de Memória Curta e Longa (*Long-Short Term Memory* - LSTM) (Hochreiter e Schmidhuber, 1997; Sherstinsky, 2018) são um tipo de RNN capaz de persistir a informação por um longo período. Em uma RNN padrão, os módulos são construídos usando uma simples camada baseada em uma função tangente hiperbólica, tomando como entrada, o padrão atual (X_t) e a informação produzida no módulo anterior (h_{t-1}). Esse processo é ilustrado na Figura 2.12a.

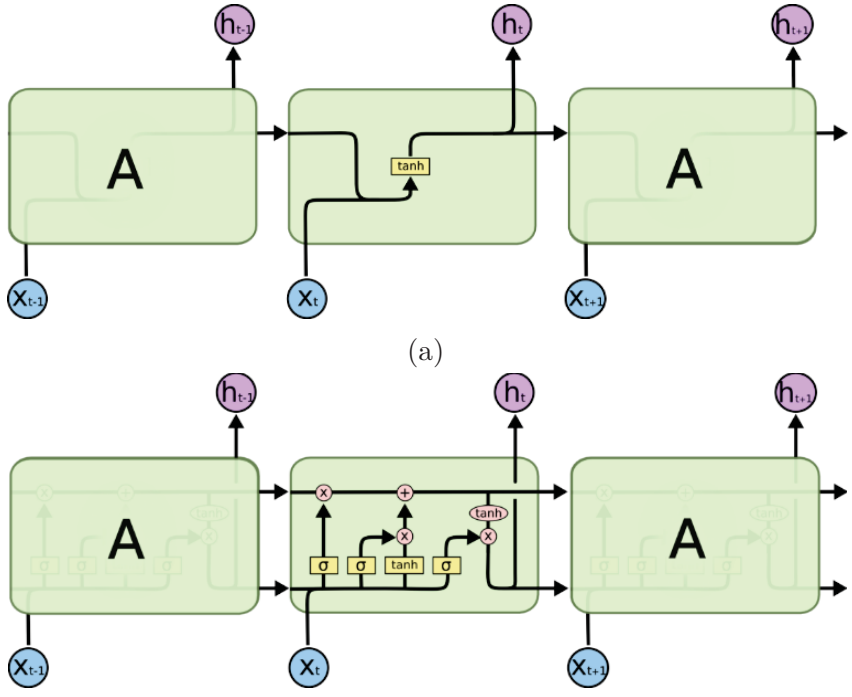


Figura 2.12: RNN x LSTM: (a) RNN apresentando apenas um fluxo de informação, ou memória curta, baseada em seu estado anterior, e (b) LSTM apresentando dois fluxos de informação, ou seja, uma memória curta e outra longa, sendo a longa representado pela linha superior, sendo atualizada pela informação atual e pela memória curta do estado anterior.[1]

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs possuem uma estrutura um pouco mais complexa (Figura 2.12b), contendo 4 funções que interagem entre si, além de um segundo fluxo de informação, denominado C_t, que representa a memória longa. Esse fluxo de informação transita entre as camadas realizando algumas poucas interações lineares, conforme ilustra a Figura 2.13.

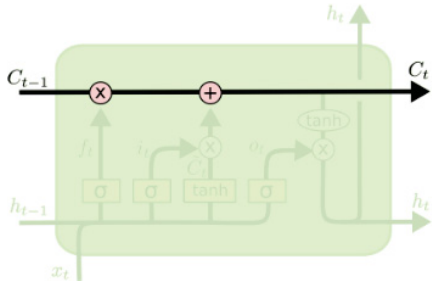


Figura 2.13: Fluxo de Informação C_t: A informação transita entre as camadas realizando poucas interações lineares, persistindo a informação por períodos mais longos.

Para determinar a persistência da informação, são utilizados três gatilhos denominados gatilho de Esquecimento (f_t), gatilho de Entrada (i_t) e gatilho de Saída (o_t). Essas funções são detalhadas a seguir e estão representadas na Figura 2.14.

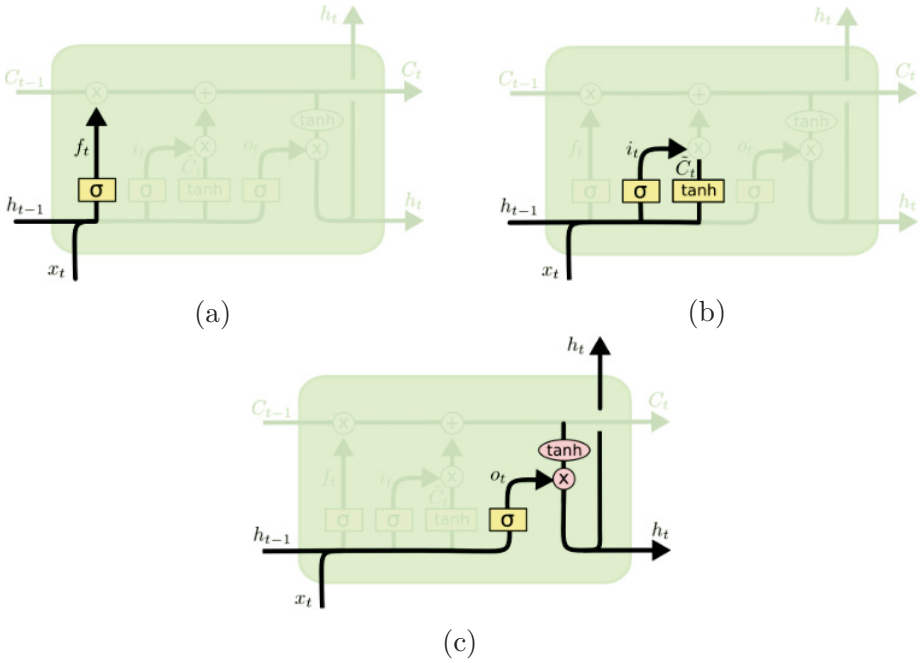


Figura 2.14: Gatilhos LSTM: (a) Esquecimento (f_t), (b) Entrada (i_t) e (C) Saída (o_t).[1]

O primeiro passo é decidir qual informação será descartada da memória longa (C_{t-1}), o qual é determinado pelo gatilho de esquecimento f_t (Figura 2.14a). Este leva em consideração os dados fornecidos por h_{t-1} e X_t, que aplicados a função sigmoide determina um valor entre 0 e 1, no qual, 0 representa "total esquecimento". Posteriormente, o gatilho de entrada i_t (Figura 2.14b) determina o quanto da nova informação será adicionada a C_t. Então, C_t é atualizado pelo produto de f_t e a adição dos dados produzidos por i_t.

Por fim, o gatilho de saída o_t decide quais dados vão produzir a saída do módulo, baseado nos dados fornecidos por h_{t-1} e X_t e na memória C_t atualizada.

[1]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

2.5 YOLO *FRAMEWORK*

O YoLo (Redmon et al., 2016; Redmon e Farhadi, 2017) é um *framework* de detecção de objetos de propósito geral que pode ser treinado de ponta-a-ponta. Utilizando uma única rede (Darknet) e considerando a imagem como um todo, este modelo estima regiões (localização) e classes com um único transpasse ao invés de aplicar o modelo em cada região como os tradicionais métodos de janela deslizante ou baseados em região (Girshick, 2015; Ren et al., 2015). O framework é ilustrado na Figura 2.15.

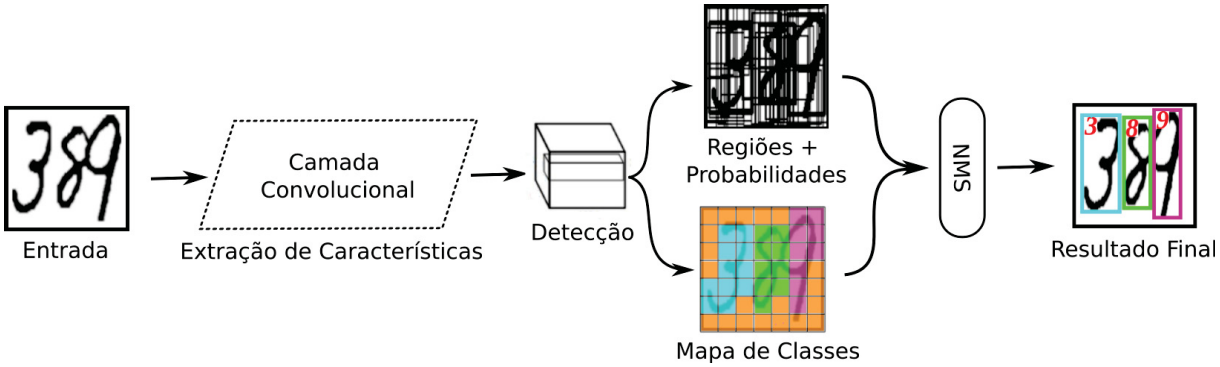


Figura 2.15: YoLo *Framework*: O sistema divide a imagem em uma matriz e cada célula estima localização e classes.

Primeiro, as camadas convolucionais Seção (2.5.1) extraem características da imagem inteira e posteriormente a camada de detecção divide a imagem em células (matriz). Para cada célula, o modelo estima as coordenadas (x, y, largura e altura) e uma probabilidade desta região compreender um objeto. Para auxiliar o aprendizado da rede na estimativa da localização, caixas âncoras (Seção 2.5.2) são previamente definidas. Além disso, são fornecidas classes para aquelas células que compreendem objetos. Por fim, o algoritmo de supressão não-máxima (Seção 2.5.3) é utilizado para mitigar algumas confusões entre regiões sobrepostas.

2.5.1 Camada Convolucional

Redmon et al. (2016) definiu a arquitetura de rede (Darknet) apresentada na Tabela 2.1 para classificar 1000 classes de objetos. A estrutura é composta por 19 camadas convolucionais e 5 camadas de agregação pelo máximo. Para desempenhar a detecção de objetos, foi suprimida a última camada convolucional e foram adicionado três camadas 3x4 com 1024 filtros.

2.5.2 Caixas-Âncoras

Conforme dito anteriormente, as caixas âncoras tem por função auxiliar na detecção de objetos. Sendo assim, essas devem possuir dimensões similares as classes disponíveis na base de treinamento. A Figura 2.16 apresenta a dimensão das caixas âncoras definidas pelos autores utilizando exemplos da base de dados Imagenet (Russakovsky et al., 2015b) a qual é composta por mais de 1000 classes de objetos reais.

Tabela 2.1: Arquitetura da Darknet

Camada	Tipo	Filtros	Tamanho/Passo	Saída
#1	Convolucional	32	3x3 /1	224x224
#2	Agregação-Máxima		2x2 / 2	112x112
#3	Convolucional	64	3x3 / 1	112x112
#4	Agregação-Máxima		2x2 / 2	56x56
#5	Convolucional	128	3x3 / 1	56x56
#6	Convolucional	64	1x1 / 1	56x56
#7	Convolucional	128	3x3 /1	56x56
#8	Agregação-Máxima		2x2 / 2	28x28
#9	Convolucional	256	3x3 / 1	28x28
#10	Convolucional	128	1x1 / 1	28x28
#11	Convolucional	256	3x3 / 1	28x28
#12	Agregação-Máxima		2x2 / 2	14x14
#13	Convolucional	512	3x3 / 1	14x14
#14	Convolucional	256	1x1 / 1	14x14
#15	Convolucional	512	3x3 / 1	14x14
#16	Convolucional	256	1x1 / 1	14x14
#17	Convolucional	512	3x3 / 1	14x14
#18	Agregação-Máxima		2x2 / 2	7x7
#19	Convolucional	1024	3x3 / 1	7x7
#20	Convolucional	512	1x1 / 1	7x7
#21	Convolucional	1024	3x3 / 1	7x7
#22	Convolucional	512	1x1 / 1	7x7
#23	Convolucional	1024	3x3 / 1	7x7
#24	Convolucional	1000	1x1	7x7
#25	Agregação-Média		Global	1000
#26	Softmax			

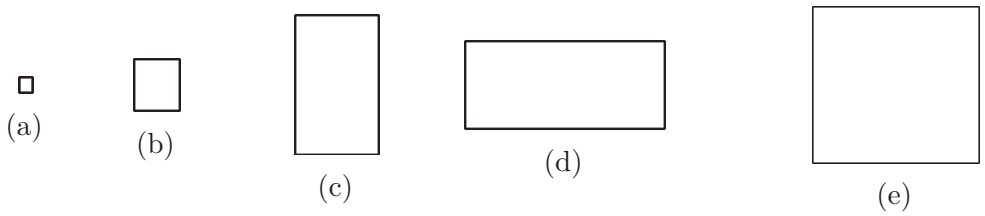


Figura 2.16: Caixas-âncoras definidas a partir da base Imagenet.

2.5.3 Algoritmo de Supressão Não-Máxima (NMS)

Para lidar com a múltipla detecção de objetos, quando um ou mais regiões estão sobrepostas, o algoritmo de supressão não-máxima é aplicado. Primeiro, as detecções com probabilidades *a posteriori* (confiança) abaixo de um limiar ($\mathcal{T}_{cnf}$) são rejeitadas. Posteriormente, aquelas regiões que se sobrepõem entre si, com uma razão de interseção sobre a união (IoU) maior que um limiar ($\mathcal{T}_{IoU}$), o algoritmo considerará aquela com maior confiança. Sejam A e B duas detecções, a interseção sobre a união é definida pela Equação 2 e é ilustrada na Figura 2.17.

$$IoU(A, B) = A \cap B / A \cup B \tag{2}$$

Figura 2.17: Interseção sobre a União (IoU).

Alguns exemplos de taxas de IoU são ilustrados na Figura 2.18 e o impacto do limiar $\mathcal{T}_{IoU}$ é apresentado na Figura 2.19. Da perspectiva de segmentação, este limiar é diretamente similar a sobre-segmentação.

Figura 2.18: Exemplos de taxas de Interseção sobre a União (IoU): (a) 5%, (b) 15%, (c) 40% e (d) 85%.

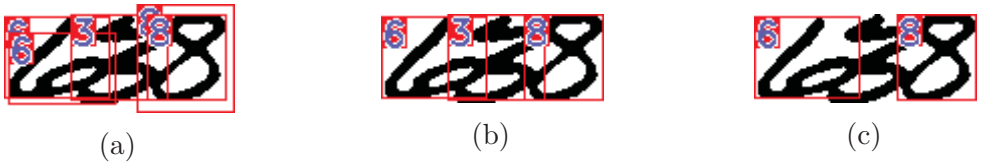


Figura 2.19: Impacto de $\mathcal{T}_{IoU}$ no algoritmo NMS: (a) Sem supressão, (b) $\mathcal{T}_{IoU} = 30\%$ e (c) $\mathcal{T}_{IoU} = 10\%$.

2.6 CONSIDERAÇÕES FINAIS

Conforme veremos no decorrer deste trabalho, os conceitos apresentados neste capítulo embasam alguns trabalhos apresentado no estado da arte (Capítulo 3) e as propostas apresentadas no Capítulo 4.

3 ESTADO DA ARTE

Este capítulo apresenta uma revisão do estado da arte. Conforme exposto na Introdução (Capítulo 1), as metodologias baseadas em segmentação serão classificadas de acordo com a taxonomia proposta por Casey e Lecolinet (1996), na qual os algoritmos são divididos entre Segmentação-e-Reconhecimento (Seção 3.1) e Segmentação-Baseada-no-Reconhecimento (Seção 3.2). O esquema é ilustrado na Figura 3.1. A Seção 3.3 apresenta algumas abordagens livres de segmentação, os quais os componentes são classificados diretamente.

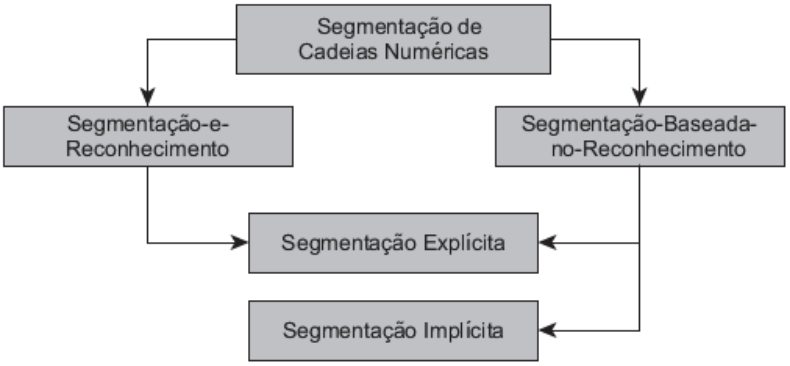


Figura 3.1: Classificação dos Métodos de Reconhecimento de Cadeias Numéricas baseados em segmentação.

Com o propósito de nortear o nosso levantamento, utilizamos os trabalhos de Kulkarni e Vasambekar (2010), Saba et al. (2010) e Ribas et al. (2012) os quais revisam as principais contribuições das duas últimas décadas. O trabalho de Ribas et al. (2012) em específico, propõe uma base sintética (Oliveira et al., 2005) com quase 80.000 amostras de duplas conectadas, balanceadas entre as classes 00 a 99 e diferentes tipos de conexão. Deste modo o trabalho consegue normalizar a comparação de desempenho entre os algoritmos, apresentando análises interessantes das abordagens.

Ao final deste capítulo, na Seção 3.4, apresentamos um comparativo entre os métodos mais relevantes, definindo suas principais características, além de bases utilizadas e se o objetivo principal era a segmentação e/ou reconhecer cadeias numéricas. Por fim, a Seção 3.5 apresenta as contribuições na área de Redes Neurais Recorrentes (RNN's) e detecção de objetos, conceitos estes que serão aplicado para o reconhecimento de dígitos em uma imagem.

3.1 SEGMENTAÇÃO-E-RECONHECIMENTO

Nessa estratégia as definições sobre os pontos de segmentação são baseados em informações explícitas tais como contorno e características geométricas, além de informações implícitas, como análises de histogramas, por exemplo. Também algumas heurísticas são aplicadas para definir os melhores pontos. O resultado final é uma única sequência de sub-imagens segmentadas, contendo hipoteticamente os dígitos isolados da cadeia (hipóteses). A Figura 3.2 ilustra o processo. O reconhecimento da cadeia é dado pela classificação individual das hipóteses de segmentação geradas.

Shi e Govindaraju (1997) analisam o contorno do objeto, avaliando as oito direções do código de *freeman* que apresentem curvas acentuadas para a direita, em relação ao

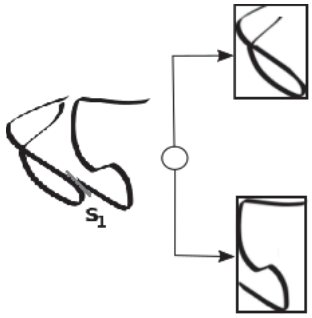


Figura 3.2: Segmentação-e-Reconhecimento: (a) Numeral “85” conectado. (b) Uma única hipótese de segmentação, denotada por S_1, define os dígitos isolados. A classificação de cada componente determina a classe da cadeia.

seu contorno oposto. A determinação se a curva é acentuada ou não é definida por um limiar baseado no ângulo calculado. Posteriormente, um histograma de projeção vertical da imagem é gerado. Dentre os pontos obtidos, analisando curvas acentuadas à direita, a segmentação será sobre aquele que mais se aproximar de uma linha média vertical obtida a partir do vale projetado pelo histograma. Os autores revelam uma taxa de acerto de 80,5% na segmentação de 1966 pares conectados presentes na base *CEDAR*. A simplicidade do método torna-o muito particular para casos em que os dígitos estão conectados por ligamentos, ou seja, mesmo que haja conexão existe um espaçamento entre os contornos, permitindo segmentá-los claramente. No entanto, o método tende a falhar quando não houverem vales bem definidos ou esses não condizerem com a região de toque, como em casos onde os dígitos estiverem sobrepostos e/ou existirem mais de um ponto de toque, conforme ilustra Figura 3.3.

Figura 3.3: Shi e Govindaraju (1997): (a) Segmentação baseada na linha média do vale, (b) e (c) são exemplos de casos em que o algoritmo tenderá a falha devido a inexistência de um vale.

Chen e Wang (2000) utilizou um conjunto de funções Gaussianas para determinar a segmentação de pares de dígitos conectados. A partir de esqueletos superiores e inferiores extraídos do contorno, algumas heurísticas são aplicadas para determinar hipóteses de segmentação (Figura 3.4). Características a respeito da geometria dos componentes segmentados são extraídas, formando um vetor de características para cada hipótese. A etapa de aprendizado da arquitetura, consiste em selecionar manualmente o vetor que representa a correta segmentação de cada amostra de treinamento. Posteriormente, a partir destes vetores, o algoritmo *K-Means* categorizará 20 tipos de toques (classes). A ideia é que, para cada tipo toque, exista uma segmentação adequada. Por fim, funções Gaussianas são computadas para cada categoria. Na fase de teste, a segmentação de um

par de dígito conectado, é determinada pela função Gaussiana que apresentar a maior probabilidade dentre as hipóteses de segmentação geradas para este. O autor revela uma taxa de acerto de 96%, aplicando o método sobre 4178 amostras da base *NIST SD19* e 322 amostras particulares. Apesar da elevada taxa de acerto, a etapa de treinamento envolve um processo manual de seleção dos melhores vetores de características, tornando-a custosa e subjetiva. Com relação ao protocolo experimental, o autor define que, dentre as 20 classes de toques categorizadas, 5 representam casos com múltiplos toques, quando é necessário mais de um corte para separar os dígitos, e o restante, toque simples. Assim, é provável a existência de um *viés* em relação a base utilizada. Se amostras com múltiplos toques forem mais frequentes que as de toque simples, essa categorização não deverá atender.

(a) (b) (c)

Figura 3.4: Chen e Wang (2000): (a) Esqueletos definidos sobre a imagem original (linhas mais finas), (b) e (c) são hipóteses de segmentação a partir de heurísticas definidas sobre os esqueletos.

Wang et al. (2000) apresentam uma segmentação baseada em uma matriz (*grid*) aplicada sobre a imagem. É definida uma matriz contendo 4 linhas e 6 colunas (4x6), suprimindo as duas colunas centrais por estatisticamente conter as regiões de toque e foram consideradas sem informação relevante. O processo de segmentação é demonstrado na Figura 3.5. Cada célula da matriz forma um vetor de características binário de 512 bits, extraindo informações da sua região como Gradiente (192 bits), Estrutura (192 bits) e Concavidade (128 bits), as quais segundo o autor descrevem respectivamente, forma local, relação entre os traços e forma global do dígito. A classificação é realizada por um *KNN*, computando os vetores extraídos por cada célula. O testes foram aplicados em pares conectados das bases *CEDAR* e do correio americano *USPS*, atingindo respectivamente 83,5% e 85,1%. A presunção de que as colunas centrais não apresentam informações relevantes é a principal falha do método, pois ao suprimi-las, além de alterar a forma dos dígitos, e adicionando assim ruídos, o centro de massa do objeto não garante a divisão correta do componente, visto que os dígitos podem ter larguras distintas.

(a) (b) (c)

Figura 3.5: Wang et al. (2000): (a) Imagem Original, (b) Colunas da matriz de segmentação, (c) Resultado da supressão das colunas centrais.

Suwa e Naoi (2004) geram um grafo sobre o contorno, definindo os pontos de corte analisando as arestas e vértices, de acordo com o ilustrado na Figura 3.6. Primeiramente um algoritmo de afinamento é aplicado, transformando o contorno em uma linha de apenas um pixel de largura. A definição dos vértices que irão compor o grafo é baseada na vizinhança de cada pixel. De acordo com o grau do pixel (número de vizinhos), o vértice é definido dentro das seguintes condições: Final (grau 1), Junção T (grau 3) e Cruzamento (grau 4+). Outros vértices são definidos quando houverem mudanças bruscas de direção

do contorno. Também são gerados histogramas de projeção superior e inferior. Ainda são delimitados dois pontos de toque do contorno, ambos localizados nos picos das projeções superior e inferior. Posteriormente, os vértices do grafo que estiverem mais próximos a esses picos definem a região de corte. Algumas heurísticas são aplicadas para identificar o tipo de toque (simples, múltiplo ou ligamento), e dessa forma definir se a aresta, ou um conjunto de arestas, entre os vértices devem ser eliminadas ou irá compor um dos sub-grafos resultantes. O protocolo experimental utilizou 2000 amostras de pares conectados da base *NIST SD19* e a proposta obteve um desempenho de 88,7% na segmentação destes. A definição dos pontos de corte baseados em grafos e projeções, cria um modelo dinâmico. No entanto, ao final do processo, algumas decisões empíricas para determinar as arestas candidatas a segmentação, refletem a necessidade de um afinamento manual de acordo com características singulares à base, limitando assim o desempenho do algoritmo.

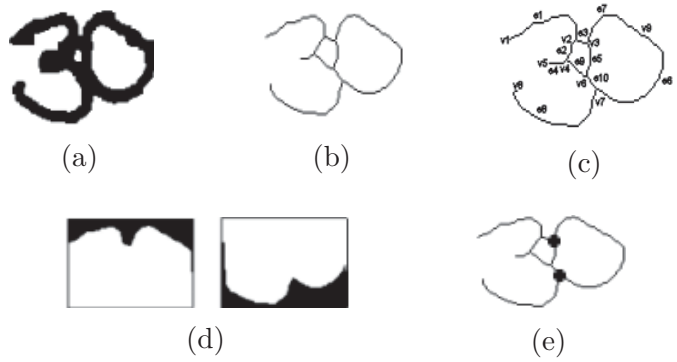


Figura 3.6: Suwa e Naoi (2004): (a) Imagem Original, (b) Afinamento, (c) Grafo, (d) Projeção vertical inferior e superior e (e) Vértices candidatos.

Chuang et al. (2005) utiliza uma representação em forma de ondas para segmentar dígitos conectados (Figura 3.7). Dado uma cadeia de tamanho desconhecido, componentes conexos são encontrados analisando o histograma de projeção vertical dos pixeis. Para definir se o componente contém um dígito isolado ou dígitos conectados, é calculada uma linha contínua sobre a respectiva projeção vertical deste componente, tangendo os picos e vales. O resultado é algo que se assemelha a gráfico de frequência de sinal ou onda. O número de dígitos é definido pelo número de cristas de onda, enquanto que os possíveis pontos de segmentação para componentes com mais de um dígito, são determinados combinando parâmetros da crista, como comprimento e altura, com pontos extraídos ao longo do objeto (*foreground*). Uma taxa de acerto de 83,2% foi obtida, segmentando 685 cadeias numéricas extraídas a partir de uma base particular de cheque bancários chineses. Conta como ponto positivo para o trabalho, a definição automática do tamanho da cadeia. No entanto, o fraco protocolo experimental apresentado, levanta dúvidas a respeito do desempenho. Além de utilizar uma base relativamente pequena, o autor informa apenas a taxa de acerto global e não fornece maiores detalhes sobre as amostras presentes na base.

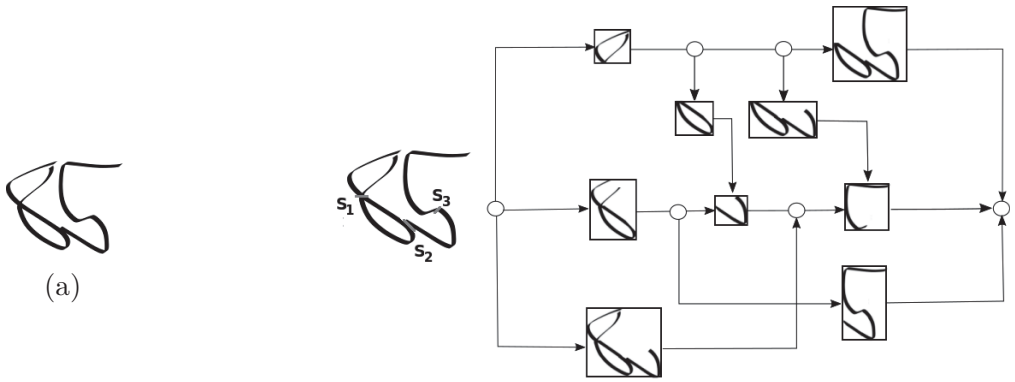
Conforme apresentado nesta seção, as abordagens buscam encontrar o melhor ponto de segmentação baseando-se apenas pelas heurísticas implementadas. Evidente que uma heurística pode resolver um caso ou outro, mas devido a grande variabilidade de padrões de conexões entre os dígitos, definir uma única heurística global, que contemple todos os casos é um procedimento complexo.

(a) (b)

Figura 3.7: Chuang et al. (2005): (a) Determinação dos componentes conexos pela projeção do histograma e (b) Determinação da quantidade de dígitos de um componente pelo número de cristas.

3.2 SEGMENTAÇÃO-BASEADA-NO-RECONHECIMENTO

A segmentação nessa abordagem é auxiliada pelo desempenho do classificador. Utilizando características explícitas e implícitas análogas às descritas na seção anterior (3.1), vários pontos de segmentação são definidos, gerando um conjunto finito de hipóteses. As hipóteses deste conjunto são classificadas individualmente. O reconhecimento da cadeia, é dado por um método de combinação desses resultados, geralmente baseados em um modelo probabilístico e/ou hierárquico, por exemplo, um grafo ou uma árvore. A Figura 3.8 ilustra este processo. De maneira geral, devido a inúmeras consultas ao classificador e a implementação de métodos para seleção das melhores hipóteses, as abordagens baseadas no reconhecimento apresentam um custo computacional mais alto em relação as abordagens do tipo Segmentação-e-Reconhecimento, porém apresentam melhores resultados.



(a) (b)

Figura 3.8: Segmentação-Baseada-no-Reconhecimento: (a) Par de dígitos conectados, “85”, e (b) as hipóteses de segmentação organizadas em um grafo, geradas a partir dos pontos de corte S_1, S_2 e S_3.

Ha et al. (1998) definiram uma arquitetura modular, ilustrada na Figura 3.9. Primeiramente, o módulo de pré-segmentação determina os componentes conexos e, através de algumas heurísticas, reconstrói alguns dígitos fragmentados. Em seguida, um classificador determina se o componente é um dígito ou uma cadeia conectada. No caso de dígito isolado, este é classificado diretamente, caso contrário, a cadeia é segmentada. Esse processo, validação de dígito isolado e segmentação é repetido até que a cadeia esteja segmentada. O módulo de decisão é responsável por determinar uma classe para cadeia ou rejeitá-la O classificador implementado é uma rede neural treinada com características extraídas de projeções e do contorno (Ha e Bunke, 1997). A taxa de reconhecimento atingida sobre a *NIST SD3*, com 4925 amostras entre 2- e 6-dígitos, foi de 92,7%. O ponto positivo da abordagem é a definição automática se o é um dígito isolado ou dígitos conectados. No

entanto, não estima a quantidade de elementos naqueles conectados, o que auxiliaria a rejeição nos casos de sub- e sobre-segmentação.

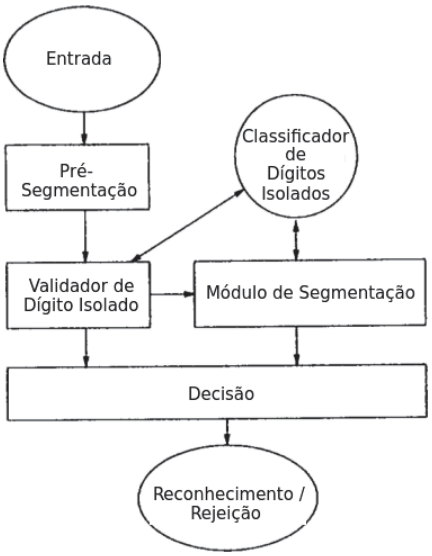


Figura 3.9: Arquitetura proposta por Ha et al. (1998).

Kim et al. (2002) utilizam um conjunto de pontos de segmentação para definir o tipo de toque e consequentemente a região de corte. Os pontos extraídos do contorno (Figura 3.10), formam conjuntos de pontos superiores e inferiores sobre o mesmo eixo X da imagem, dos quais, uma projeção da diferença vertical entre ambos os conjuntos (eixo Y) é calculada. Heurísticas aplicadas sobre essa projeção, resultam na definição de pontos de transição e pontos de vales e topos. Esses pontos caracterizam o tipo de toque existente entre os dígitos, dentre seis possíveis tipos de toques previamente definidos de forma empírica. De acordo com o tipo de toque identificado, novas heurísticas são aplicadas, combinando os pontos para determinar possíveis regiões de corte (sub-imagens). As sub-imagens de cada região são classificadas por um MLP, e a região de corte que apresentar o melhor resultado é escolhida. Uma taxa de 92,5% foi alcançada em 3500 de pares de dígitos conectados disponíveis na base *NIST SD19*. O ponto negativo dessa abordagem, é que apesar de ser baseada em reconhecimento, utiliza alguns parâmetros definidos empiricamente para elencar os pontos de segmentação e tipo de toque existente, o que acaba especializando a solução e limitando seu desempenho para a determinada base.

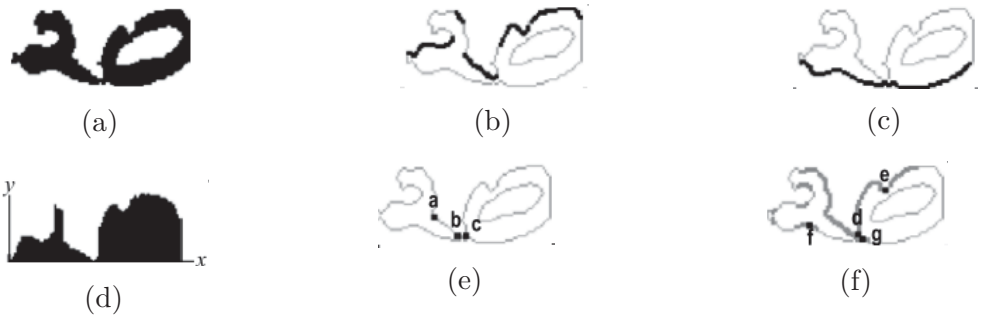


Figura 3.10: Kim et al. (2002): (a) Imagem Original, (b) Pontos Superiores (PS), (c) Pontos Inferiores (PI), (d) Diferença Vertical ($DV = PS - PI$), (e) Pontos de Transição, (f) Vales e Topos (Max/Min).

O método proposto por Oliveira et al. (2002) adiciona duas redes neurais na etapa de classificação para avaliar se houve sobre-segmentação ou sub-segmentação da cadeia. Na etapa de segmentação, os pontos são definidos a partir de interseções e bases do contorno e do esqueleto do dígito (Oliveira et al., 2000). Um grafo interconectando esses pontos é gerado para representar o numeral conectado, e os sub-grafos possíveis, geram as hipóteses de segmentação do numeral. Na etapa de reconhecimento, uma matriz de tamanho 3x6 é aplicada sobre cada provável dígito, extraindo de cada célula características de concavidade, direção do contorno (códigos de *Freeman*) e contagem de pixels. As células formam um vetor de característica que fornece as informações para que uma rede neural defina a classe do provável dígito. A contribuição do método consiste em paralelamente validar se houve sobre-segmentação ou sub-segmentação, extraindo distintas informações de concavidades e aplicando em redes neurais específicas para cada caso. Por fim, as três probabilidades (dígito, sobre e sub) são combinadas a fim de fornecer uma classe para a hipótese. As hipóteses válidas são organizadas em um grafo para posterior análise. O autor reporta 93,88% de acerto em cadeias de dois a dez dígitos da base *NIST SD19*. Posteriormente, em (Oliveira e Sabourin, 2004), o autor compara o desempenho de SVM's nessa metodologia, os quais elevam em torno de um ponto percentual as taxas de acerto, superando as RN's. Neste método, os dois modelos treinados para validar sobre e sub-segmentação atuam como uma espécie de um filtro implícito ao processo, minimizando que ruídos provenientes da segmentação incorreta acarretem em falso positivo. O amplo número de exemplos e classes utilizados nos testes, torna consistente o resultado obtido.

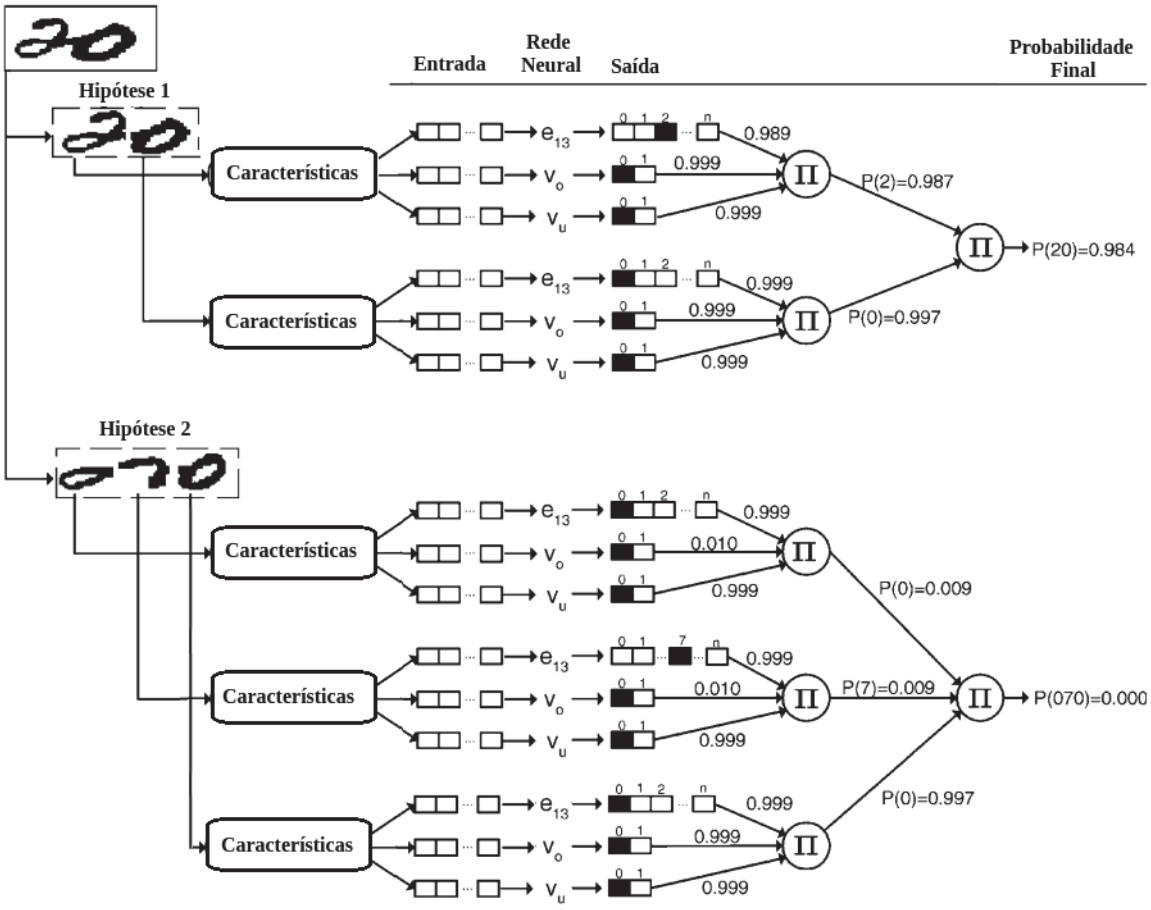


Figura 3.11: Oliveira et al. (2002): $e13$, V_o, V_u representam, respectivamente, as redes neurais para dígitos isolados, sobre-segmentação e sub-segmentação.

Britto et al. (2003) propõe um método composto de dois módulos baseados em Cadeias Escondidas de Markov (HMM). No primeiro módulo cada coluna é processada individualmente, extraindo-se informações sobre os pixeis nas regiões de transição entre o fundo e o objeto, e vice-versa, tais como direção do pixel, posição relativa e projeção vertical. Cada coluna gera um vetor de característica, os quais são mapeados em símbolos discretos previamente definidos, formando uma sequência com o número de elementos igual ao número de colunas. A sequência de símbolos discretos é processada por dez HMM's treinados sobre dígitos isolados (0-9), fornecendo as hipóteses de segmentação e seus respectivos *scores*. No segundo módulo, para cada hipótese, novas características são extraídas, porém combinando as colunas e as linhas, e re-mapeadas em uma sequência símbolos discretos, análogo ao processo do primeiro módulo. Por fim, novos HMM's classificam a cadeia combinando as informações produzidas no primeiro e no segundo módulo. Os experimentos realizados na base *NIST SD19* alcançaram um desempenho geral de 90,6% em cadeias de dois a dez dígitos. Apesar da taxa de acerto apresentar níveis ligeiramente abaixo ao estado da arte, a contribuição da proposta é uma solução de segmentação implícita, sem necessitar de conhecimento prévio da base. Também vale ressaltar o protocolo de teste, utilizando um vasto número de exemplos, tornando promissor o uso de HMM's em reconhecimento de numerais.

Em Lei et al. (2004), linhas de segmentação são definidas a partir do contorno. A primeira etapa consiste verificar se o componente conexo contém um dígito isolado ou um numeral, definindo-se um limiar sobre a probabilidade *a posteriori* de um classificador treinado com dígitos isolados. Uma baixa probabilidade assume-se que o componente é um numeral. Sendo assim, define-se as extremidades horizontais do componente, ou seja, os pontos mais a esquerda e a direita. O contorno presente entre as extremidades, da esquerda para a direita, no sentido horário, é considerado a parte superior do dígito, enquanto que o contrário será parte inferior. Este processo é ilustrado na Figura 3.12. Então, em ambos os contornos, pontos de curva ou interseção do são extraídos e, heurísticas são aplicadas para elencar os pontos mais promissores ou candidatos. Posteriormente, através de análises sobre as projeções horizontais, são definidos fragmentos do contorno (linhas) que não apresentem qualquer interseção entre dois pontos candidatos. Finalmente, esses linhas são segmentadas gerando sub-linhas, que por sua vez são denominadas "Linhas de Segmentação" (LS). Então, dado o conjunto formado por "LS's", a subtração de uma ou mais linhas desse conjunto geram as hipóteses de segmentação do numeral. A definição sobre a melhor hipótese é dada pela maior probabilidade obtida pelo classificador, ou seja, voto majoritário.

Os experimentos foram realizados utilizando cadeias com dois e três dígitos, havendo região de toque entre todos os elementos. A taxa de acerto para esta metodologia foi de 97,72% sobre 3359 imagens de pares e, 93,33% em 525 imagens de três dígitos, ambas extraídas da base *NIST SD19*, utilizando o classificador *Nested-Subset*. A proposta, combinando a subtração das linhas de segmentação, ao invés de um único ponto, visa eliminar o ligamento que define região de toque permitindo que o dígito isolado seja melhor definido. No entanto o contrário também é válido, a eliminação de uma dessas linhas de maneira que altere o formato do dígito, pode acarretar em um falso positivo. Além disso, a definição de sub-linhas a partir de uma linha de segmentação, contribui para o aumento do número de hipóteses. Com relação ao protocolo experimental, apesar de uma boa taxa em pares de dígitos, foram utilizadas poucas amostras nas cadeias com três dígitos, além de não haver testes com cadeias maiores.

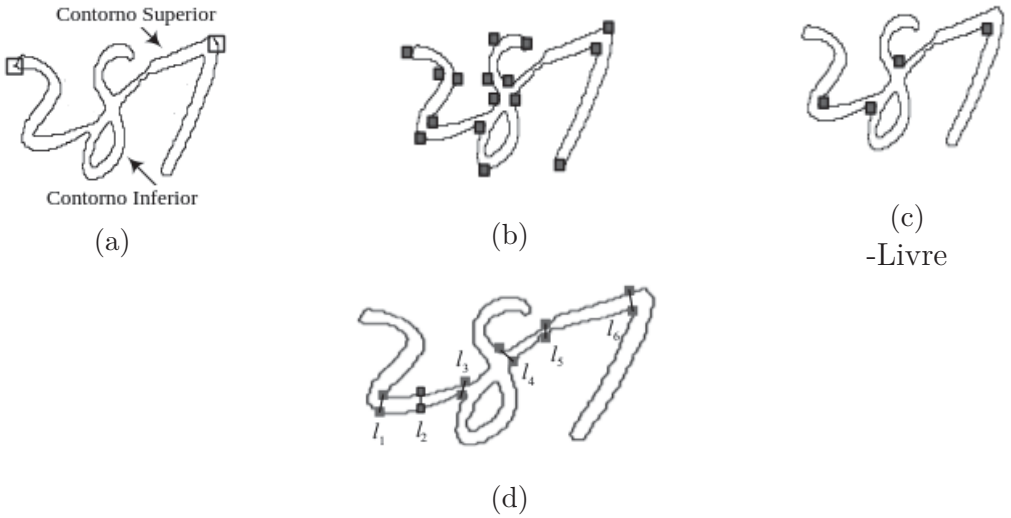


Figura 3.12: Lei et al. (2004): (a) Contornos Inferior e Superior determinado entre extremidades (□), (b) Definição dos pontos de interseção, (c) Pontos Candidatos e (D) Linhas de Segmentação.

O trabalho de Liu et al. (2004) compara o desempenho de distintos modelos tais como RNA's, SVM's, entre outros, tanto para o reconhecimento de dígitos isolados, quanto para cadeias numéricas manuscritas. Os modelos foram treinados em dois cenários: (a) apenas com amostras de dígitos e (b) amostras de dígitos e de ruídos. Seu propósito foi verificar se os modelos conseguem atribuir baixas probabilidades quando ruídos provenientes da etapa de segmentação são apresentados, como dígitos fracionados ou mal formados. Um sistema contendo as fases de segmentação, classificação e verificação foi implementado para efeito de comparação com outros trabalhos. Seus experimentos utilizando apenas dígitos isolados revelam que, a inserção de amostras ruidosas no treinamento não apresenta alterações significativas na taxas de acerto para classificação de dígitos bem formados. No entanto, para cadeias numéricas, as quais o processo de segmentação gera componentes ruidosos, os modelos treinados com uma classe para ruído possuem significativo impacto nas taxas de acerto, principalmente nos modelos utilizando redes neurais. Para exemplificar, o reconhecimento de 1476 cadeias de 3-dígitos provenientes da base *NIST SD19*, obtiveram taxas de melhoria de 13,08% utilizando uma RN, sendo 82,52% a taxa de reconhecimento no modelo treinado sem amostras ruidosas. Já em 1471 cadeias de 6-dígitos desta mesma base, a melhoria foi de 31,88%, sendo 63,70% de acerto naquele treinado sem amostras de ruído. O modelo SVM com *kernel* RBF foi o que apresentou melhores resultados em ambos ambientes. Nas amostras de três dígitos, este obteve 94,65% quando treinado somente com dígitos isolados e 96,82% com amostras ruidosas Para 6-dígitos as taxas foram de 93,07% e 96,74% respectivamente. Testes nas base *CEDAR* com 5-dígitos, o SVM obteve uma melhora de 5,51%, sendo 85,32% para treinamento apenas com dígitos isolados. O autor investigou o fato de que uma segmentação sempre produzirá ruídos, e que, um modelo robusto pode melhorar as taxas de acerto, evitando que falsos positivos sejam classificados com altas probabilidades, facilitando a fusão na etapa de verificação. A analise de erro é bastante interessante, uma vez que o autor indica quais erros foram provenientes de falsos positivos, ou seja, ruídos classificados com altas taxas, e consequentemente, causando sobre-segmentação, e quais foram provenientes de classificações errôneas de dígitos bem formados. Segundo os dados apresentados pelo autor, o SVM é um dos modelos mais robustos, obtendo as melhores taxas em todos os casos de teste. O protocolo experimental,

no entanto, poderia ser expandido para variados tamanhos de cadeias e amostras da base *NIST SD19* e *CEDAR*, o que tornaria a análise mais contundente.

Uma abordagem baseada em algoritmos genéticos (AG) foi proposta por Sadri et al. (2007) visando reduzir a complexidade de avaliação entre as inúmeras hipóteses de segmentação. Inicialmente, os pontos de segmentação são gerados a partir de uma heurística combinando a análise do esqueleto dos dígitos e histogramas de projeção superior e inferior do contorno (Sadri et al., 2004). A Figura 3.13 ilustra o processo. Primeiro, pontos de interseção são encontrados analisando o contorno superior e inferior do esqueleto da cadeia. Posteriormente, esqueletos das projeções superiores e inferiores são extraídos. Os pontos de interseção e do esqueleto das projeções são então combinados gerando as regiões de segmentação. As hipóteses são então classificadas e organizadas em um grafo, gerando um conjunto de possíveis sequências para a cadeia. Para encontrar a melhor sequência, um algoritmo genético é empregado, otimizando assim o espaço de busca. O autor revela uma taxa de acerto de 96,4%, utilizando um SVM, para reconhecimento de numerais de até dez dígitos extraídos da base *NIST SD19*. Neste trabalho, a segmentação foi tratada como um problema de otimização, o que torna a abordagem interessante além de atingir um bom resultado utilizando uma base consistente. No entanto, a solução proposta acarreta em um aumento de complexidade, inserindo uma etapa de pós-processamento utilizando AG's para elencar a melhor hipótese. Outro ponto a ser questionado é o fato de que um AG é um método estocástico, ou seja, não é garantido um resultado ótimo, nem mesmo a reprodução para uma mesma entrada.

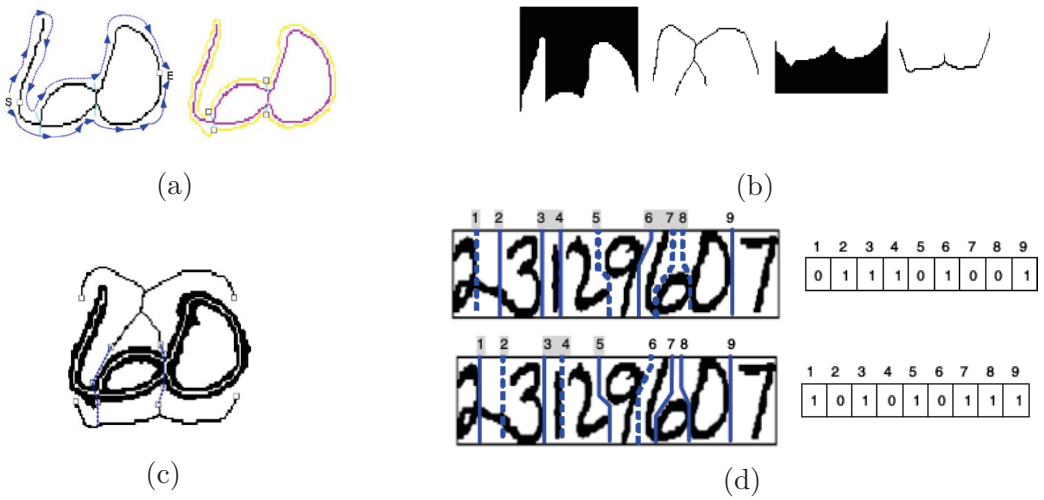


Figura 3.13: Sadri et al. (2004): (a) Contornos Inferiores/Superiores do Esqueleto e Pontos de Interseção, (b) Esqueleto das Projeções dos Contornos Superiores e Inferiores, (c) Definição dos pontos de corte e (d) hipóteses de segmentação da cadeia, sendo as linhas contínuas aquelas selecionadas pelo algoritmo genético.

Vellasques et al. (2008) propõem reduzir o número de hipóteses de segmentação, introduzindo um filtro baseado em aprendizagem de máquina entre a etapa de segmentação e reconhecimento. Na fase de treinamento, características sobre a vizinhança dos pixeis do contorno, obtidas em exemplos de dígitos isolados e de ruídos, alimentam um SVM. O objetivo é determinar se a hipótese resultante de um processo de segmentação qualquer, contém um dígito parcial (ruído) ou um dígito bem definido. Em caso de ruído a hipótese é ignorada, caso contrário, a hipótese é enviada para um classificador de dígitos isolados. O autor revela uma redução de 83% no número de hipóteses de segmentações incorretas, aplicadas em uma base sintética de dígitos pares (Oliveira et al., 2005). O foco deste

trabalho foi a redução de número de hipóteses para melhorar o processo de reconhecimento e verificação. A proposta de usar aprendizado de máquina ao invés de heurísticas para avaliar as hipóteses de segmentação, se mostrou bastante eficiente. As propostas de Vellasques et al. (2008) e Oliveira et al. (2002), apesar de similares, diferem pelo fato do segundo implementar dois classificadores, um para sobre-segmentação e outro para sub-segmentação, que na prática também atuam como filtros, porém em paralelo a etapa de classificação, não reduzindo assim o envio de ruídos ao classificador de dígitos isolados, tornando a proposta do primeiro menos custosa.

Ding et al. (2008) apresenta um método auto-ajustável para encontrar os melhores pontos de segmentação. Na primeira etapa, o contorno é percorrido encontrando pontos de descontinuidade, gerando uma hipótese de segmentação entre dois desses pontos. Ao invés de definir o exato ponto de segmentação baseado em heurísticas, o autor propõe um algoritmo recursivo aonde alguns ajustes de parâmetros são realizados e os pontos são selecionados e avaliados a cada iteração. Enquanto a classificação não atingir um resultado aceitável, o algoritmo repete a operação de ajuste. A operação é direcionada da esquerda para a direita, até que se encontre o dígito mais a esquerda primeiro. Posteriormente, o processo é repetido para restante da cadeia. O autor também propõe o uso de uma rede convolucional, para classificar os numerais "00" e "000", o qual apresenta maior dificuldade de segmentação. Os experimentos foram realizados em 12985 cheques, da base *Quebec Bell*, atingindo 74,3% de acerto no reconhecimento da cadeia. A proposta é interessante por tentar definir uma segmentação implícita, porém o fato de determinar o ponto de segmentação apenas por um limiar não é uma boa estratégia. Uma segmentação incorreta no início da cadeia, proveniente de um falso positivo, compromete toda a análise. O ideal é utilizar-se de um modelo para combinar as segmentações possíveis, como um grafo por exemplo, ao invés de um limiar global. Pesa como ponto negativo, o fraco detalhamento das etapas, principalmente das métricas utilizadas para formar o conjunto dos pontos de segmentação ou como identificar um numeral composto por zeros, além do detalhamento das amostras utilizadas na base, como quantidades dígitos, número de classes, etc.

Os trabalhos de Gattal e Chibani (2012); Gattal et al. (2015); Gattal e Chibani (2015); Gattal et al. (2017) propõem o uso de uma janela deslizante sobre determinados pontos para definir a melhor segmentação. Primeiro, pontos de interseção dos contornos são definidos baseados em algumas heurísticas, similar ao método proposto por Oliveira et al. (2000). Então, para cada um destes pontos é aplicado uma janela vertical (NxM), a qual rotaciona sobre seu eixo em ângulos pré-definidos, conforme ilustra a Figura 3.14. Dessa forma são geradas diferentes hipóteses de segmentação sobre um mesmo ponto. Afim de reduzir o número de hipóteses, em Gattal et al. (2017) os autores aplicaram a transformada de Radon (Ludwig, 1966; Toft, 1996) para definir o ângulo da janela deslizante. Para a etapa de reconhecimento, em cada um de seus trabalhos, os autores utilizaram conjuntos distintos de características, variando entre análise do esqueleto, concavidades, projeções, entre outras. Para cada hipótese, o conjunto de característica utilizado é classificado por um SVM, definindo uma classe e uma probabilidade. A seleção da melhor hipótese será através do melhor escore (voto majoritário), formado pelo produto das probabilidades de cada classe, dos prováveis dígitos que compõe a cadeia. Os experimentos foram realizados em pares de dígitos e atingiram taxas de 92,46% (Gattal e Chibani, 2012), 90,57% (Gattal et al., 2015), 93,24% (Gattal e Chibani, 2015) e 96,91% (Gattal et al., 2017), utilizando uma base sintética (Ribas et al., 2012) e a base *NIST SD19*. A proposta de uma janela rotacionando sobre seu eixo é a principal contribuição dos autores. O ponto positivo é que este método trata mais facilmente a segmentação de dígitos inclinados, no entanto, o

protocolo experimental com poucas amostras da vasta base e utilizando apenas cadeias de dois dígitos (vide dados na Tabela 3.1), torna difícil a avaliação mais detalhada do método.

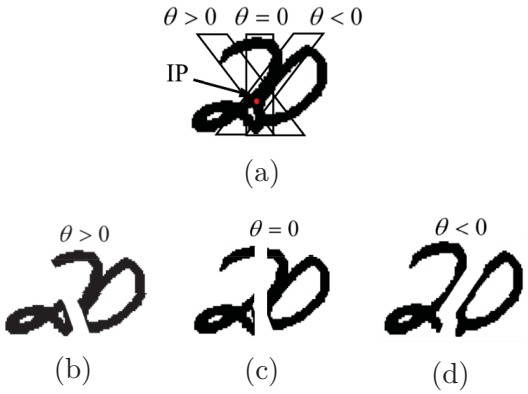


Figura 3.14: Gattal et al. (2015): (a) Exemplos de posicionamento da janela sobre o ponto de interseção (IP), de acordo com o ângulo θ. (b),(c) e (d) Representam as hipóteses de segmentação.

Os autores Wu et al. (2014) definem características geométricas para o problema. Primeiramente uma etapa de detecção de componente conexo e segmentação baseada em projeção de histograma é realizada, gerando assim algumas hipóteses de segmentação e organizando-as em um grafo. As hipóteses são então classificadas utilizando uma função quadrática (MQDF), atribuindo uma probabilidade *a posteriori*. O autor define mais algumas características geométricas dos componentes, como aspectos do *bounding box*, perfil vertical, semelhança com outros dígitos e ruídos, dentre outras. Para essas características, elenca dois novos classificadores quadráticos e dois baseados em *SVM*. Esses quatro novos classificadores definem novas probabilidades ao componente, com o propósito de elencar as melhores hipóteses, filtrando assim prováveis ruídos. Os experimentos foram focados no desempenho dos classificadores geométricos, não reportando taxas de acerto para a cadeia numérica.

3.3 LIVRE-DE-SEGMENTAÇÃO

Para evitar os problemas provenientes da etapa de segmentação, como fragmentação dos dígitos e a avaliação de inúmeras hipóteses, alguns autores apresentaram abordagens que classificam diretamente os componentes.

Lecun et al. (1998) apresentaram uma abordagem baseada em redes convolucionais e deslocamento espacial (*Space Displacement*). Sem a necessidade de definir explicitamente pontos de segmentação ou limiares, uma cadeia numérica é processada recorrentemente pela rede convolucional, varrendo a entrada da esquerda para a direita, análogo à um processo temporal. Uma versão aprimorada da sua rede convolucional (LeCun et al., 1989) é apresentada, a *Lenet-5*. O final deste processo é a concatenação dos mapas de características, produzidos pela última camada de cada instância da rede. A concatenação do mapas segue a arquitetura da rede, porém estes com uma maior dimensão horizontal devido a quantidade de instâncias. A lógica em torno desta metodologia é que os mapas produzem ativações diferentes para cada região da imagem, sendo possível determinar hipóteses de segmentação do numeral. Deste modo, define-se as respectivas probabilidades *a posteriori* e as hipóteses são dispostas em um grafo, sendo essas analisadas pelo algoritmo de *Viterbi*, o qual determina as classes do numeral. Uma ilustração do processo pode ser visualizado na Figura 3.15.

Outro modo de analisar as hipóteses, segundo o autor, seria aplicá-las à uma Máquina de Estados ou Modelos Escondidos de Markov (HMM). O autor utilizou a base MNIST para avaliar a metologia de reconhecimento em cadeias numéricas, no entanto não reportou as taxas obtidas, fazendo apenas menções sobre custos computacionais, casos de sucesso e falha. Em um trabalho anterior, utilizando este conceito e a arquitetura *Lenet-1*, Matan et al. (1992) reportaram 66,3% em cadeias de 5 dígitos extraídas de códigos postais. A principal contribuição é a proposta de método de segmentação, determinada de forma implícita. O processo é realizado sobre os mapas de características produzidos pela última camada convolucional da rede, diferentemente de um processo que determina heurísticas sobre a entrada original. Outra vantagem é que os filtros da rede definem os parâmetros do problema, como absorção de ruídos, pontos de segmentação ou dígitos parciais. O ponto negativo, como citaram os próprios autores, é que a proposta ficou abaixo as taxas do estado da arte, porém a estratégia foi pioneira na área, apresentando uma arquitetura mais rápida e paralelizável. Além disso deve-se relevar que a mesma foi proposta e testada em uma época de diferentes limitações computacionais e com certa dificuldade em adquirir bases volumosas, crucial para o treinamento de uma rede convolucional.

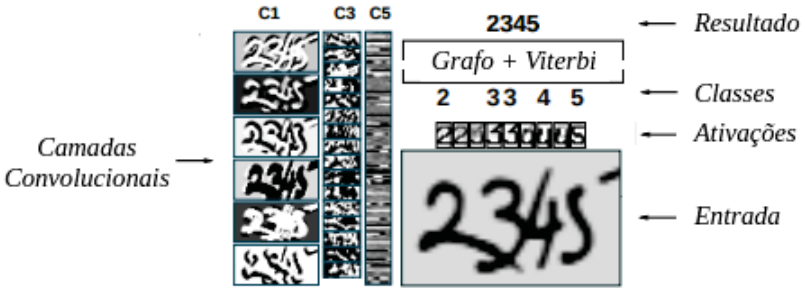


Figura 3.15: Lecun et al. (1998): Reconhecimento de Numerais utilizando uma rede convolucional e deslocamento espacial.

Choi e Oh (1999) propõem o reconhecimento de pares de dígitos conectados, utilizando uma arquitetura formada por 100 RNA's. Ao invés de criar um único modelo que estime probabilidades *a posteriori* para as classes entre 00 a 99, o propósito é que cada rede seja especializada em uma determinada classe. Sendo assim, cada rede é desenhada para classificar uma entrada em um problema de duas classes: positiva e negativa. Deste modo, para um par de dígitos apresentado como entrada, a arquitetura determinará uma classe selecionando a rede que prover, dentre as classes positivas, aquela com maior probabilidade *a posteriori*. Em um cenário ideal, apenas a rede especialista para uma determinada entrada classificaria como positiva, e as demais, determinariam a classe negativa. A Figura 3.16 ilustra o método. Para suprir a falta de amostras de pares de dígitos conectados, tanto em quantidade, quanto em variabilidade e balanceamento entre as classes, os autores construíram uma base sintética, produzindo assim dados suficientes para o treinamento das redes. Os experimentos, a partir de 1374 amostras extraídas da base *NIST*, atingiram 77,8% de taxa de acerto. O método é bastante interessante por não apresentar heurísticas no processo e evitar a segmentação, estando alinhado a nossa proposta. No entanto, seu conjunto de características, denotado apenas pela distância entre os pixeis da vizinhança, pode ser considerado o ponto fraco, sendo uma das prováveis causas para um desempenho inferior as demais abordagens.

Redes Neurais Convolucionais foram utilizadas por Ciresan (2008); Ciresan e Pescaru (2008) para classificar pares de dígitos sem a necessidade de segmentá-los. Após detectar os componentes conexos de uma cadeia, o autor aplica uma heurística de agru-

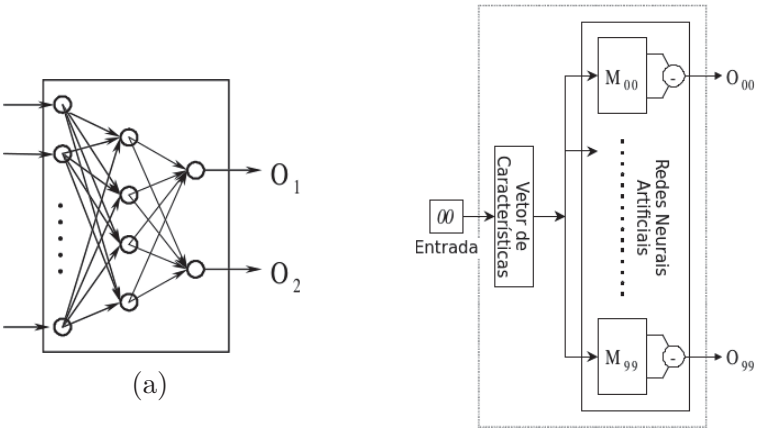


Figura 3.16: Choi e Oh (1999): (a) Rede especialista e (b) arquitetura composta por 100 RNA's, fornecendo cada qual uma probabilidade O para entrada.

pamento (*clustering*) para reconstruir dígitos que apresentam ruídos em sua composição, como por exemplo a parte superior do dígito '5' ilustrado na Figura 3.17. O autor define duas redes convolucionais, uma com 10 classes para dígitos isolados e outra com 100 classes para pares de dígitos conectados. Os componentes conexos da cadeia são classificados por ambas as redes e uma combinação das probabilidades *a posteriori*, determina a classe vencedora. Os testes realizados em 1476 cadeias contendo três dígitos da base *NIST SD19*, atingiram uma taxa de reconhecimento de 93,49%. A proposta denota o potencial das redes convolucionais dentro do problema de reconhecimento de dígitos, além de demostrar futuro para a abordagem de classificar pares de dígitos conectados, como sendo um único componente. O ponto falho da proposta é utilizar apenas as probabilidades *a posteriori* para definir se o componente é contido por um ou dois dígitos, visto que falsos positivos podem introduzir erros a análise, conforme exposto por Oliveira et al. (2002) e Sadri et al. (2007), sendo adequado determinar antecipadamente o tamanho do componente e utilizar o classificador especialista no problema. O protocolo experimental também não aborda cadeias maiores que três dígitos. Este trabalho suporta duas de nossas hipóteses de pesquisa, a de que classificar dígitos conectados utilizando aprendizado profundo, sem a necessidade de segmentá-los é uma abordagem promissora, e também, a definição da quantidade de dígitos é uma informação valiosa.

Figura 3.17: Ciresan (2008): (a) Dígito parcial e (b) dígito reconstruído.

3.4 RESUMO DOS MÉTODOS

Na Tabela 3.1 apresentamos um breve comparativo entre os métodos apresentados em nosso levantamento do estado da arte. Os trabalhos são comparados de acordo com os seguintes atributos:

- **Ano [Ref]:** Ano e Referência do trabalho.
- **Tipo**: Informa se o algoritmo processa pares de dígitos ou cadeias numéricas.
- **Pré-Proc:** Informa se é aplicado alguma técnica de pré-processamento da imagem, como redução de ruídos, detecção da área de interesse (*Bounding Box*), algoritmos afinamento ou esqueletização, dentre outros.
- **Estimativa de Tamanho**: Informa se o algoritmo é capaz de determinar a quantidade de dígitos e a técnica utilizada.
- **Características**: Informa quais as principais informações extraídas sobre a imagem, como pixeis do fundo e do objeto (BF), Contorno (Cont.), Concavidades (Conc.), Projeções (Proj.), dentre outras, ou até mesmo se é realizada de forma implícita.
- **Sobre-Seg**: Informa se o processo gera sobre-segmentação ou não.
- **Classificador:** Informa o algoritmo de classificação utilizado.
- **Verificação:** Informa qual o método implementado para combinar as hipóteses.
- **Base:** Informa a Base de Dados utilizada.
- **Amostras / Desempenho (%):** Informa, respectivamente, a quantidade de exemplos utilizado na fase de teste e a taxa de acerto obtida.

Tabela 3.1: Comparativos entre os Métodos por Tipo de Abordagem

Ano [Ref]	Tipo	Pré-Proc	Estimativa de Tamanho	Características	Sobre-Seg	Classificador	Verificação	Base	Amostras / Desemp. (%)
Segmentação-e-Reconhecimento									
Shi e Govindaraju (1997)	Pares	Sim	Não	BF / Cont. / Proj.	Não	N/D	N/A	CEDAR	1966 / 80,5[1]
Chen e Wang (2000)	Pares	Não	Não	BF / Cont.	Sim	Gaussianas	N/A	NIST SD19 CEDAR	4178 / 96,0 322 / 96,0
Wang et al. (2000)	Pares	Sim	Não	BF / Cont. / GSC	Não	KNN	N/A	USPS CEDAR	2778 / 85,1 628 / 83,5
Suwa e Naoi (2004)	Pares	Sim	Não	Proj. / Grafo	Não	N/D	N/A	NIST SD19	2000 / 88,7[1]
Chuang et al. (2005)	Cadeia	Sim	Cristas de Onda	BF/ Proj. / Cristas de Onda	Não	N/D	N/A	PARTICULAR	683 / 83,2[1]
Segmentação-Baseada-no-Reconhecimento									
Ha et al. (1998)	Cadeia	Sim	Proj. Vertical	BF/ Cont. / Proj.	Sim	KNN e MLP	Prob.	NIST SD3	4925 / 92,7
Kim et al. (2002)	Pares	Sim	Não	Cont. / Proj.	Sim	MLP	Limiar Prob.	NIST SD19	3500 / 92,5
Oliveira et al. (2002)	Cadeia	Sim	Não	BF / Cont. / Conc. / Freeman	Sim	03 MLP's	Viterbi	NIST SD19	12802 / 93,8
Britto et al. (2003)	Cadeia	Sim	L.B.A	BF / Conc.	Não	HMM	HMM	NIST SD19	12802 / 90,6
Lei et al. (2004)	Pares	Sim	Prob. Classe	Cont. / Proj.	Sim	NESTED-SUBSET	Voto	NIST SD19	3359 / 97,7
Liu et al. (2004)	Cadeia	Não	Dimensões	Cont. / Proj.	Sim	SVM	Busca em Feixe	NIST SD19 CEDAR	2947 / 97,7 436 / 90,83
Sadri et al. (2007)	Cadeia	Sim	Dimensões	BF / Cont. / Proj.	Sim	SVM	AG	NIST SD19	12802 / 96,4
Vellasques et al. (2008)	Pares	Sim	Não	BF / Cont.	Sim	SVM	Limiar Prob.	Sintética	10000 / 83,0[2]
Gattal e Chibani (2012)	Pares	Não	Não	BF / Cont. / Freeman	Sim	SVM	Voto	NIST SD19	600 / 92,4
Gattal e Chibani (2015)	Pares	Não	Não	BF / Cont. / Proj.	Sim	SVM	Voto	Sintética	79466 / 93,2
Gattal et al. (2017)	Cadeia	Não	Não	BF / Cont. / Proj / Radon.	Sim	SVM	Voto	NIST SD19	12802 / 96,91
Livre-de-Segmentação									
Lecun et al. (1998)	Cadeia	Sim	Não	Implícita / Desl. Espacial	N/A	CNN	Viterbi	MNIST	- / 66,3[3]
Choi e Oh (1999)	Pares	Sim	Não	BF / Distâncias	N/A	MPL	Prob. Máx	Nist SD3	1476 / 77,8
Ciresan (2008)	Cadeia	Sim	Prob. Classe	Implicíta	N/A	02 CNN	Voto / Prob.	NIST SD19	1476 / 93,4

[1]Desempenho em termos de correta segmentação.
[2]A taxa representa a redução do número de hipóteses ruidosas ou incorretas.
[3]Taxa extraída de um trabalho correlacionado (Matan et al., 1992).

3.5 ABORDAGENS PONTA-A-PONTA

Nesta seção apresentaremos alguns métodos baseados em modelos de ponta-a-ponta. A vantagem desses modelos sobre os métodos tradicionais é que estes apresentam a etapa de segmentação, representação dos dados e treinamento em um único fluxo (*pipeline*), não necessitando de informações heurísticas na elaboração dos modelos. Basicamente estão organizados em duas abordagens, aqueles baseados em sequência-a-sequência (Seção 3.5.1) e Detecção de Objetos (Seção 3.5.2).

3.5.1 Sequência-a-Sequência

Recentemente, as redes neurais convolucionais tem superado inúmeros modelos de classificação, elevando o estado da arte para um novo patamar. Apesar de poderosa, essas redes requerem que as entradas e saídas sejam conhecidas e de tamanho fixo. Esta é uma limitação importante visto que alguns problemas requerem a interpretação de sequências, como por exemplo, séries temporais e reconhecimento de cadeias manuscritas, nos quais seus tamanhos são variáveis e desconhecidos *a priori*. Também, nestes casos, a persistência do conhecimento é essencial para a classificação da sequência. Para superar este problema, modelos baseados em Redes Neurais Recorrentes (Seção 2.4) estão sendo aplicados para reconhecer sequências. Esses modelos são denominados sequência-a-sequência, conforme ilustra a Figura 3.18. Basicamente, a informação produzida em um estágio anterior da rede é repassado ao seu estágio sucessor, produzido assim informação de contexto referente ao passado.

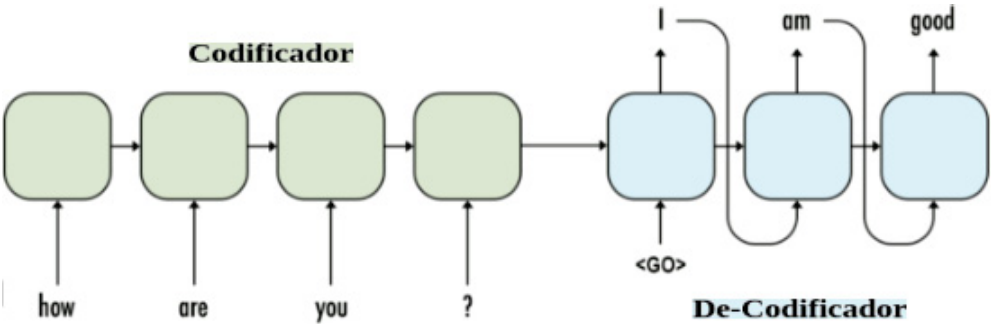


Figura 3.18: Um exemplo de um modelo de sequência-a-sequência para interpretação de linguagem natural[1].

Recentemente, abordagens utilizaram esse conceito para reconhecimento de cadeias de palavras extraídas de imagens com sucesso (Shi et al., 2017; Voigtlaender et al., 2016; Dutta et al., 2018). O *pipeline* dessas soluções é apresentado na Figura 3.19. Primeiro uma CNN extrai as características de uma imagem. Em um segundo momento, os mapas de características produzem uma sequência de vetores. Estes vetores são aplicados a uma rede LSTM produzindo uma classe para a sequência.

3.5.2 Detecção de Objetos

Como exposto no Capítulo 1, neste trabalho argumentamos que o reconhecimento de cadeias numéricas pode se beneficiar dos recentes avanços na área de detecção de objetos, na qual o objetivo é localizar e reconhecer um conjunto de classes em um dada

[1]https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts

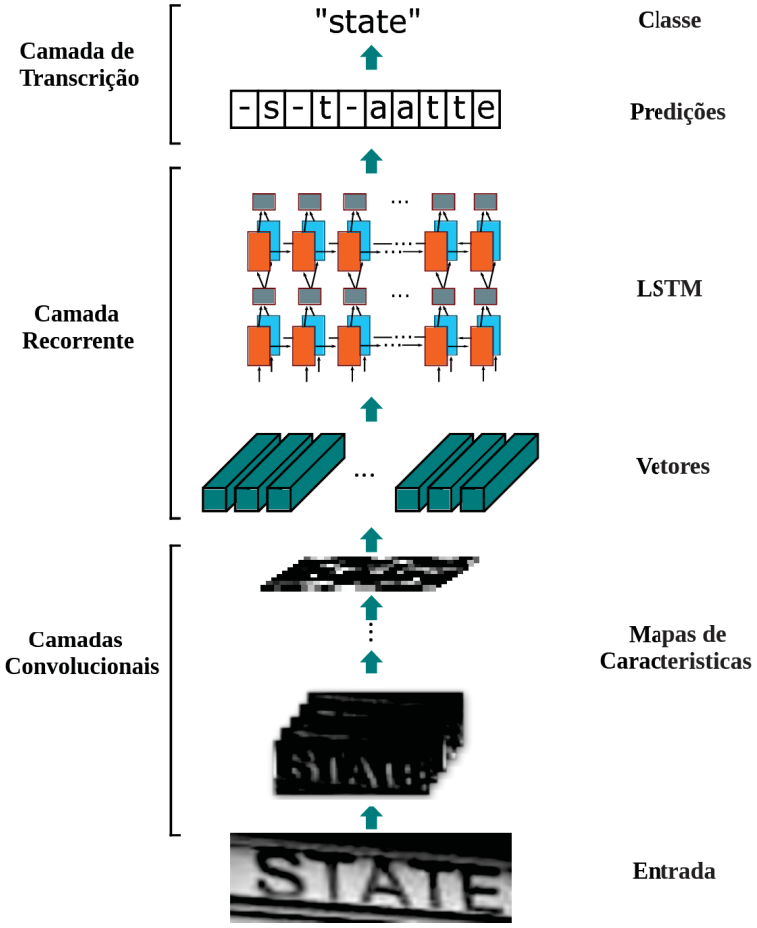


Figura 3.19: Modelo proposto por Shi et al. (2017): Rede Neural Convolucional combinada com uma Rede LSTM para o reconhecimento de palavras contidas em imagens.

imagem. Até a última década, as abordagem clássicas consistiam em métodos baseados em janelas deslizantes e suas variantes (Felzenszwalb et al., 2008, 2010; Lampert et al., 2008). Este tipo de abordagem reutiliza um classificador treinado com características explícitas em várias posições da imagem. Um gargalo dessa abordagem é o elevado número de janelas necessárias para cobrir inúmeras escalas e proporções. Além disso, nessa estratégia exaustiva, o custo computacional cresce rapidamente.

Um grande avanço ocorreu devido ao surgimento de conjuntos de dados em grande escala (Russakovsky et al., 2015b; Lin et al., 2014), popularização dos *hardwares* (GPU's) e o surgimento de redes profundas em ILSVRC 2012 (Russakovsky et al., 2015b). Neste momento, esta área de pesquisa retomou a atenção da comunidade e inúmeros métodos baseados em aprendizado profundo foram propostos elevando o estado da arte para novas fronteiras (Han et al., 2018).

Uma das primeiras abordagens que obtiveram sucesso foi a Rede Convolucional baseada em Regiões (R-CNN) proposta por (Girshick et al., 2014). Primeiro, a arquitetura extrai as regiões candidatas da imagem utilizando o algoritmo de busca seletiva (Uijlings et al., 2013). Então, cada região é redimensionada para o tamanho de entrada da rede convolucional (CNN), a qual extrai as características. Por fim, um classificador SVM determina a classe e um regressor de objetos refina a localização. Essa abordagem é ilustrada na Figura 3.20a. A necessidade de extrair características de cada região é a principal desvantagem do método, uma vez que é um processo computacionalmente custoso.

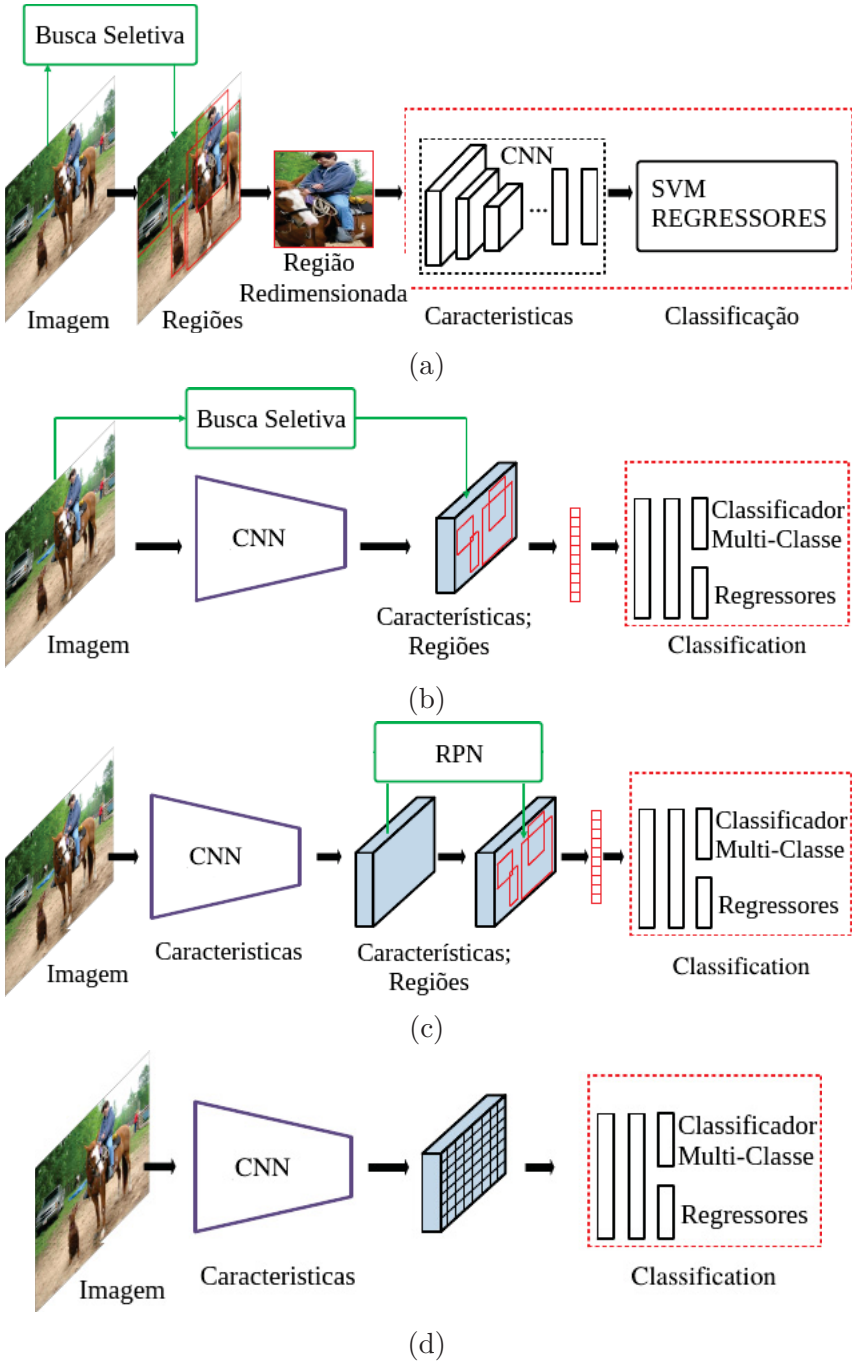


Figura 3.20: Abordagens de Detecção de Objetos: (a) R-CNN, (b) *Fast* R-CNN, (c) *Faster* R-CNN e (d) YoLo

Para superar este obstáculo, as estratégias SPPnet (He et al., 2015) e Fast-RCNN (Girshick, 2015) (Figura 3.20b) foram propostas. Estas abordagens propõe regiões candidatas diretamente sobre mapas de características. Uma camada de agregação é introduzida para produzir representações de tamanho fixo (redimensionamento em nível de característica). Embora essas abordagens aceleram o processo, elas continuam utilizando um método explícito para propor regiões. Para superar esse problema, em (Ren et al., 2015; He et al., 2017) os autores introduziram uma rede para propor regiões (RPN), a qual implicitamente produz regiões candidatas (Figura 3.20c). Neste caso, as características

produzidas pela última camada convolucional são utilizadas nas duas tarefas: (a) propor regiões e (b) classificar regiões.

As estratégias acima continuam apresentando um *pipeline* de dois estágios, uma vez que requerem uma estratégia de estimativa de regiões, não importando se está implícito ou não. Uma alternativa foi proposta por Redmon et al. (2016), a arquitetura YoLo, na qual os autores propõe uma abordagem baseada em regressão que encapsula todos os estágios em um única rede. Com um simples transpasse na rede, localizações e classes são estimados. No gráfico de precisão ilustrado na Figura 3.21, a YoLo apresenta a melhor relação entre precisão e tempo. Nota-se em especial que, quando o tempo não é restrição, mensurado em quadros por segundo, a YoLo supera todas as abordagens propostas. A variação no tempo de processamento é dado pela resolução de entrada da rede. Uma resolução mais baixa tende a processar mais rápido, no entanto, prejudicando a precisão da rede.

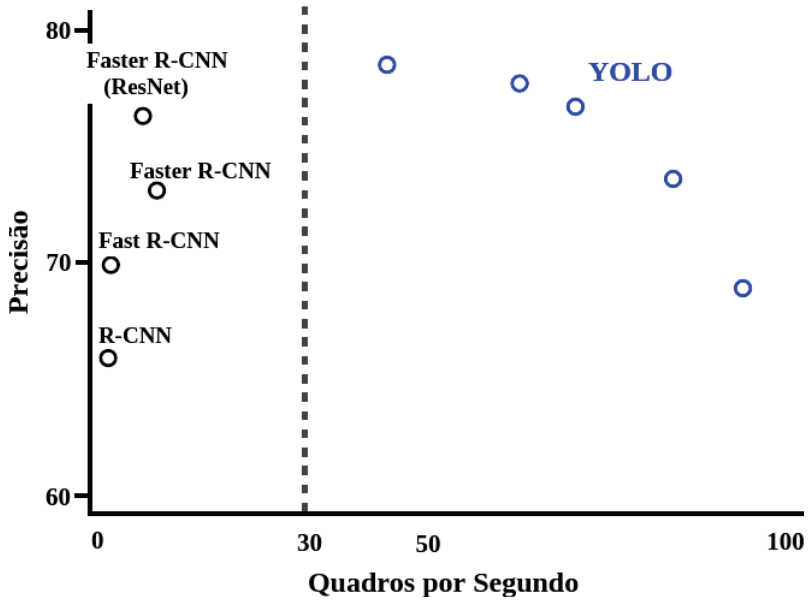


Figura 3.21: Comparativo entre as abordagens de Detecção de Objetos em termos de precisão por tempo quando aplicado a base ImageNet. (Adaptado de Redmon e Farhadi (2017))

Outro importante aspecto da YoLo é a codificação implícita do contexto e aparência da vizinhança dos objetos, o que é uma importante característica para segmentação implícita dos dígitos. Sendo assim, selecionamos este modelo para uma de nossas abordagens, a qual discutiremos melhor na Seção 4.4.

3.6 CONSIDERAÇÕES FINAIS

Apesar de um grande avanço nas técnicas de aprendizado profundo, podemos observar neste capítulo que as abordagens relevantes para o reconhecimento de cadeias numéricas ainda enfrentam barreiras provenientes da aplicação de processos heurísticos de segmentação. Nem mesmo o recente surgimento dos modelos ponta-a-ponta (Seção 3.5.1 e Seção 3.5.2), foram capazes de produzir novas abordagens para este problema. No Capítulo 4, apresentaremos os métodos resultantes deste trabalho, utilizando abordagens recentes em direção a soluções livres de segmentação para o reconhecimento de cadeias numéricas manuscritas.

4 MÉTODOS PROPOSTOS

Este capítulo descreve abordagens livre-de-segmentação para reconhecimento de cadeias numéricas manuscritas. Conforme descrito anteriormente (Seção 1.2), as abordagens resultantes tem como objetivo superar os desafios inerentes ao processo, como por exemplo, dígitos conectados e cadeias numéricas de tamanho desconhecido. Além disso, um requisito é que essas propostas tenham reduzidos parâmetros e reduzidos métodos heurísticos quando comparados ao trabalhos do estado da arte, mantendo elevados níveis de acerto.

Para suportar a tese de que a segmentação de dígitos não é mais necessária utilizando aprendizado profundo, no decorrer deste capítulo apresentamos três diferentes abordagens. A primeira consistem em um método baseado em seleção dinâmica (Seção 4.2) capaz de reconhecer dígitos isolados (10 classes), duplas conectadas (100 classes) e triplas (1000 classes) sem segmentá-los. Dado o fato que manipular múltiplos classificadores não é uma tarefa simples, a segunda abordagem (Seção 4.3) implementa um único classificador capaz de reconhecer estas 1110 classes (0...9, 00...99, 000...999).

As estratégias acima mencionadas utilizam métodos heurísticos para agrupar pixeis em componentes conexos. Para minimizar isso, a terceira estratégia utiliza uma abordagem ponta-a-ponta baseada em detecção de objetos. (Seção 4.4).

Uma vez que essas propostas são baseadas em redes profundas, as mesmas requerem uma quantidade considerável de amostras para ajustar os hiper-parâmetros. Felizmente, é possível criar cadeias numéricas sintéticas para o aprendizado correto das representações. Na seção seguinte discutimos em detalhes os dados criados para este propósito (Seção 4.1).

4.1 BASES DE DADOS SINTÉTICAS

Para aprender com eficiência a representação dos dados é necessário um número considerável de amostras. Sendo assim, criamos bases sintéticas[1] compostas de cadeias numéricas (Seção 4.1.1 e Seção 4.1.2). As cadeias são construídas concatenando dígitos isolados da base NIST SD19 (Grother, 2016) utilizando o algoritmo descrito por Ribas et al. (2012). A base SD19 é fornecida pelo Instituto Americano de Padrões e Tecnologia (NIST), contendo 3699 formulários (autores) e 814.255 dígitos e caracteres manuscritos extraídos destes formulários.

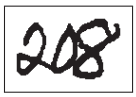
Para evitar um viés na construção das bases, utilizamos a informação disponível a respeito dos autores, de tal forma que estes foram separados exclusivamente para treinamento, validação e teste.

4.1.1 Base de Cadeias Conectadas

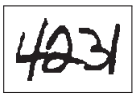
Este conjunto de dados tem como objetivo fornecer uma representação de classes de numerais conectados que variam de 2 a 4 dígitos. A Figura 4.1 ilustra alguns exemplos. A Tabela 4.1 mostra os conjuntos de treinamento, validação e teste, bem como o número de amostras disponíveis.

[1] As bases de dados estão disponíveis, mediante solicitação, para fins de pesquisa em https://web.inf.ufpr.br/vri/databases-software/touching-digits/

(a) 67



(b) 208



(c) 4231

Figura 4.1: Dados sintéticos representando cadeias numéricas conectadas de (a) 2-dígitos, (b) 3-dígitos e (c) 4-dígitos.

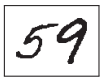
Tabela 4.1: Distribuição da base de cadeias conectadas para treinamento, validação e teste dos classificadores

Tamanho/Classes	Amostras	Autores	Propósito
2-Dígitos	161,563	1000-1599	Treinamento
100 classes	53,907	1600-1799	Validação
	55,091	1800-1999	Teste
3-Dígitos	1,448,680	1000-1599	Treinamento
1000 classes	484,346	1600-1799	Validação
	491,749	1800-1999	Teste
4-Dígitos	100,000	1000-1599	Treinamento
*	20,000	1600-1799	Validação
	20,000	1800-1999	Teste

*Dados usados para treinar o classificador de tamanho ($\mathcal{L}$).

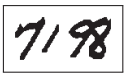
4.1.2 Base de Cadeias Sintéticas

As amostras neste conjunto de dados são numerais variando de 2- a 6-dígitos contendo dígitos isolados e/ou componentes conectados. A justificativa nesse caso é criar um conjunto de dados com informações contextuais sobre a vizinhança de dígitos isolados e conectados. A Figura 4.2 ilustra alguns casos.



(a) 59

(b) 536



(c) 7198

(d) 36125

(e) 050622

Figura 4.2: Dados sintéticos representando cadeias numéricas compostas de (a) 2- a (e) 6-dígitos.

Para criar os exemplos, introduzimos um espaçamento aleatório no algoritmo proposto por Ribas et al. (2012). O algoritmo resultante é bastante simples: Para um dado autor e um tamanho de cadeia, um numeral é definido randomicamente. Posteriormente, as amostras de dígitos isolados são concatenadas, uma a uma, alinhadas pelo eixo horizontal com um espaçamento randômico (*gap*) entre seus pontos mais próximos do contorno. O Algoritmo 1 descreve o procedimento:

Uma informação detalhada sobre a quantidade de amostras criadas e a distribuição entre treinamento, validação e teste é descrita na Tabela 4.2.

Outro aspecto que levamos em consideração ao criar este conjunto de dados, foi a distribuição de dígitos isolados e conectados nos numerais. Quando foram analisados conjuntos de dados reais, podemos observar uma distribuição exponencial de sub-cadeias, dominada por dígitos isolados. A Figura 4.3a exemplifica a distribuição enquanto que a Figura 4.3b retrata a distribuição das 10 classes de dígitos neste conjunto de dados. O dígito "1" é o menos representado, uma vez que esta classe tem menos ocorrência em numerais conectados (Ribas et al., 2012).

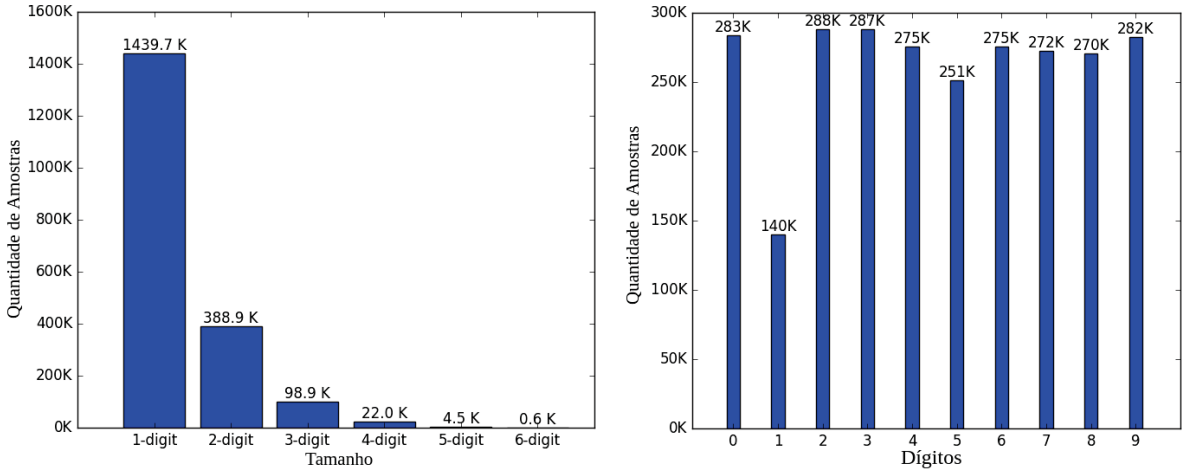
Algoritmo 1: Criação de numerais sintéticos a partir de dígitos isolados

entrada : AUTOR, TAMANHO_DA_CADEIA
saída : IMAGEM CONTENDO UM NUMERAL (IMAGEM)

```
1 NUMERAL ← GERARNUMERALALEATÓRIO(TAMANHO_DA_CADEIA)
2 IMAGEM ← OBTEMAMOSTRADOAUTOR(AUTOR, NUMERAL[0])
3 for i ← 1 to TAMANHO_DA_CADEIA do
4 |   GAP ← RANDOMPIXELGAP(-15, 15)
5 |   AMOSTRA ← OBTEMAMOSTRADOAUTOR(AUTOR_ID, NUMERAL[i])
6 |   IMAGEM ← CONCATENAAMOSTRAS(IMAGEM, AMOSTRA, GAP)
```

Tabela 4.2: Distribuição da base de numerais sintéticos para treinamento, validação e teste dos classificadores.

Tamanho/Classes	Amostras	Autores	Propósito
2-Dígitos	42,614	1000-1599	Treinamento
	14,202	1600-1799	Validação
	14,838	1800-1999	Teste
3-Dígitos	76,890	1000-1599	Treinamento
	25,507	1600-1799	Validação
	27,025	1800-1999	Teste
4-Dígitos	82,625	1000-1599	Treinamento
	27,487	1600-1799	Validação
	29,166	1800-1999	Teste
5-Dígitos	82,944	1000-1599	Treinamento
	27,663	1600-1799	Validação
	29,371	1800-1999	Teste
6-Dígitos	82,926	1000-1599	Treinamento
	27,609	1600-1799	Validação
	29,396	1800-1999	Teste



(a) (b)

Figura 4.3: Distribuição dos dados: (a) Distribuição quanto a dígitos isolados e sub-cadeias conectadas, e (b) Distribuição das 10 classes de dígitos

4.2 ABORDAGEM BASEADA EM SELEÇÃO DINÂMICA

O sistema discutido nesta seção adota uma abordagem livre de segmentação baseada em três módulos principais: Pré-processamento, Classificador de tamanho e Classificadores de Dígitos. Inicialmente, uma imagem I passa por um módulo de pré-processamento (Seção 4.2.1) que identifica todos os componentes conexos (CC's) presentes. Cada CC é classificado pelo classificador de tamanho $\mathcal{L}$ (Seção 4.2.2.1) o qual determinará a probabilidade deste ser composto por 1, 2, 3 ou 4 dígitos conectados. Como dito anteriormente, a maioria dos toques ocorre entre dois dígitos e algumas vezes entre três dígitos. Cadeias conectadas compostas por mais de três dígitos são muito raras. Por exemplo, ao analisar a base NIST SD19, encontramos poucas amostras deste tipo. Sendo assim, se o classificador de tamanho $\mathcal{L}$ atribuir a classe 4 para o CC, a cadeia será rejeitada.

O módulo de classificação (Seção 4.2.2.2), compreende três classificadores de dígitos ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$), projetados para discriminar 10 [0 . . . 9], 100 [00 . . . 99], e 1000 [000 . . . 999] classes. O classificador que será utilizado para um dado CC depende da saída do classificador de tamanho ($\mathcal{L}$). A seleção é feita por um método de fusão descrito em detalhes na Seção 4.2.3. De acordo com a confiança de $\mathcal{L}$, mais de um classificador de dígito pode ser invocado para mitigar possíveis confusões. Então, a decisão final é feita combinando as probabilidades *a posteriori* produzidas para todos os CC's encontrados na imagem I. A Figura 4.4 ilustra a arquitetura proposta, a qual será detalhada nas seguintes sub-seções.

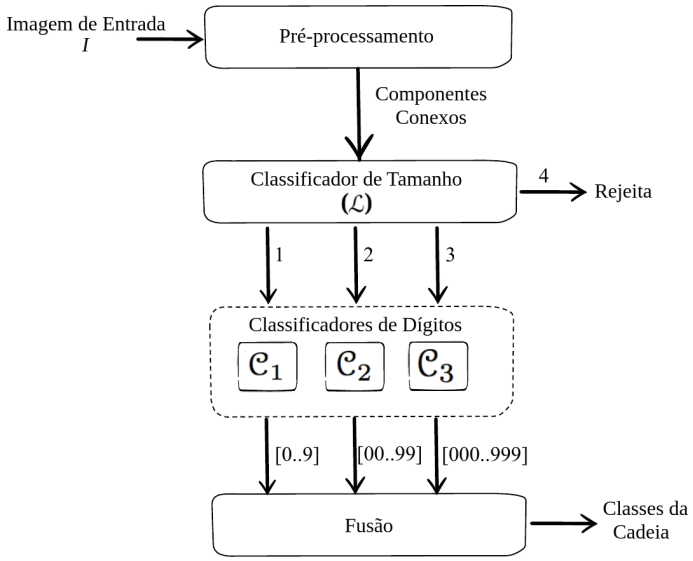


Figura 4.4: Arquitetura baseada em seleção dinâmica.

4.2.1 Pré-Processamento

Na etapa de pré-processamento, a entrada é processada, minimizando ruídos que porventura possam existir e normalizando-a para as próximas etapas da arquitetura. Primeiramente o processo consiste na eliminação de ruídos utilizando um filtro Gaussiano, o qual tem a propriedade de não alterar forma e evitar o borramento da imagem e, posteriormente, determina-se os componentes conexos presentes na imagem. Um dos problemas inerentes ao reconhecimento de dígitos manuscritos, é quando estes encontram-se fragmentados, ou seja, quando durante a escrita ou no processo de aquisição houve uma descontinuidade do contorno. É evidente que se não houver o tratamento

destes casos, estes fragmentos serão detectados como componentes e, consequentemente, tornarão a classificação da cadeia imprecisa. Para minimizar o impacto deste problema, implementamos o método de reconstrução proposto por Ha et al. (1998) e validado por de Oliveira et al. (2002).

Dado os parâmetros definidos na Figura 4.5, um CC é determinado como fragmento se ao menos uma das condições abaixo forem satisfeitas:

I. CC não intercepta a linha média.

II. $\frac{max(CC_{superior}, CC_{inferior})}{min(CC_{superior}, CC_{inferior})} > T_{frag}$

na qual $CC_{superior}$ e $CC_{inferior}$ denotam, respectivamente, a altura da fração superior e inferior de um componente em relação a linha média.

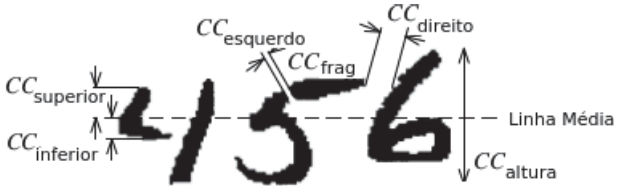


Figura 4.5: Parâmetros para detecção e reconstrução de dígitos fragmentados.

A razão determinada por T_{frag}, na segunda condição, caracteriza que, mesmo havendo interseção com a linha média, um CC é considerado fragmento se o ponto de interseção ocorrer próximo a base ou ao topo do componente, acarretando que uma das partes tenha dimensões superior a outra. A decisão para reconstruir o fragmento, CC_{frag}, ao vizinho da esquerda, $CC_{esquerdo}$, ou ao da direita, $CC_{direita}$, é definido por aquele que estiver mais próximo. Se o fragmento estiver no início ou no final da cadeia, evidente que só haverá um componente válido para reconstruí-lo. O processo é ilustrado na Figura 4.6. Determinamos $T_{frag} = 5$ a partir de imagens capturadas a 300dpi.

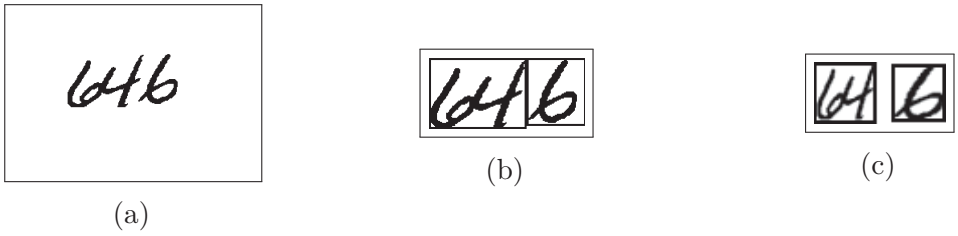
(a) (b)

Figura 4.6: Reconstrução: (a) dígitos '5' e '4' fragmentados, sendo os retângulos denotando os fragmentos, e (b) dígitos reconstruídos.

Por fim, uma normalização de tamanho é realizada, alterando a dimensão da imagem de acordo com aquela necessária para o próximo módulo da arquitetura. Neste caso, a imagem é normalizada para 64x64 pixeis. O resultado do pré-processamento é ilustrado na Figura 4.7. A determinação da quantidade de dígitos presentes em cada componente detectado na cadeia será dada no módulo posterior, detalhado na Seção 4.2.2.1.

4.2.2 Classificadores

Seja x um CC o qual deve ser atribuído a uma das ω classes. $\mathcal{C}$ é o classificador e $\mathcal{C}(x) = p^i(x) | \forall_i (1 \leqslant i \leqslant \omega)$ define que o classificador $\mathcal{C}$, atribuirá para a entrada x, probabilidades $p^i(x)$ para cada classe i de ω. Esta definição é utilizada para todos os classificadores desta arquitetura. As taxas de reconhecimento, quando representadas por *Top-1* e *Top-2* denotam, respectivamente, aquelas obtidas utilizando as primeiras e as segundas probabilidades mais altas.



(a) (b) (c)

Figura 4.7: Pre-Processamento: (a) entrada original, (b) filtro gaussiano, detecção de componentes conexos e reconstrução, se houver fragmentos, e (c) normalização de tamanho.

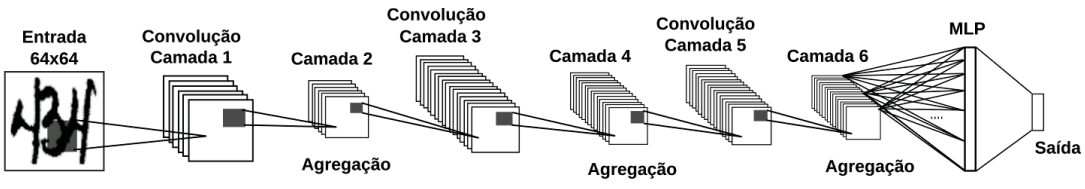
Todos os classificadores utilizados nesta estratégia são CNN's construídas utilizando múltiplas camadas considerando as seguintes operações: convoluções, agrupamento e camadas totalmente conectadas, na qual, camadas convolucionais e totalmente conectadas têm parâmetros auto-ajustáveis que são otimizados durante o treinamento.

O treinamento é realizado com Gradiente Descendente Estocástico usando retro-propagação com mini-lotes de 256 instâncias, fator de *momentum* de 0,9 e decaimento do peso de 5×10^{-4}. A taxa de aprendizado é de 10^{-2} no início para permitir um rápido ajuste dos pesos produzindo curvas abruptas. Posteriormente a taxa é reduzida gradualmente até 10^{-4} produzindo curvas suavizadas. A rede faz uso da conhecida função de perda de entropia cruzada. A regularização foi implementada via parada antecipada, o que previne o sobre-ajuste, interrompendo o treinamento quando a performance da rede na base de validação deteriora-se.

Para implementar os modelos utilizamos a arquitetura Caffe (Jia et al., 2014) e placas GPU NVidia GeForce GTX Titan Black e NVidia GeForce Titan Xp.[1].

4.2.2.1 Classificador de Tamanho ($\mathcal{L}$)

O classificador de tamanho ($\mathcal{L}$) foi projetado para definir a quantidade de dígitos presentes em um CC. Foram avaliadas diversas arquiteturas de rede para este classificador, contudo, a que atingiu os melhores resultados foi aquela baseada na conhecida arquitetura LeNet-5 (LeCun et al., 1989; Lecun et al., 1998). A arquitetura final contém três camadas convolucionais seguidas por camadas de agregação. Esta arquitetura foi definida empiricamente na base de validação e é ilustrada na Figura 4.8.



Modelo	Entrada	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5	Camada 6	MLP	Saída
$\mathcal{L}$	64x64 Binária	Convolução 5x5@1@6	Agregação 2x2@2@6	Convolução 5x5@1@32	Agregação 2x2@2@32	Convolução 3x3@1@20	Agregação 2x2@2@20	72	4

Figura 4.8: Arquitetura do classificador $\mathcal{L}$. Os parâmetros de cada camada estão representados por Dimensão@Passo@Mapas.

O classificador foi treinado conforme o protocolo descrito na Seção 4.2.2 e foram utilizadas 400.000, 79.157 e 79.742 amostras de cadeias conectadas descritas na Seção

[1]Todos os classificadores estão disponíveis, mediante solicitação, para fins de pesquisa em https://web.inf.ufpr.br/vri/databases-software/touching-digits/

4.1.1, para treinamento, validação e teste, respectivamente. O tempo para treinar 30,000 iterações foi de 90 minutos, enquanto que a classificação consome um tempo médio de 0.4 milissegundos (ms) por CC. A taxa de reconhecimento na base de testes foi de 98,4% e 99,9% para Top-1 e Top-2, respectivamente. A Tabela 4.3 apresenta a matriz de confusão.

Tabela 4.3: Matriz de confusão (%) para $\mathcal{L}$ na base de teste.

	(1)	(2)	(3)	(4)
(1)	99,9	0,01		
(2)	0,02	99,2	0,07	
(3)		0,9	96,9	2,3
(4)			2,3	97,7

Analisando as confusões resultantes de $\mathcal{L}$, pode-se concluir que a quantidade e a localização dos traços verticais parece fornecer informações importantes para determinar o tamanho da cadeia. Por exemplo, dígitos isolados que são classificados como 2-dígitos são em geral zeros com traços ou com partes faltantes, como os exemplos ilustrados na Figura 4.9a e 4.9b. Dígitos altamente sobrepostos, como as amostras "3" e "9" na Figura 4.9c e cadeias com diversos traços verticais, como o caso "544" (Figura 4.9d) também são fontes de confusão.

(a) (b) (c) (d)

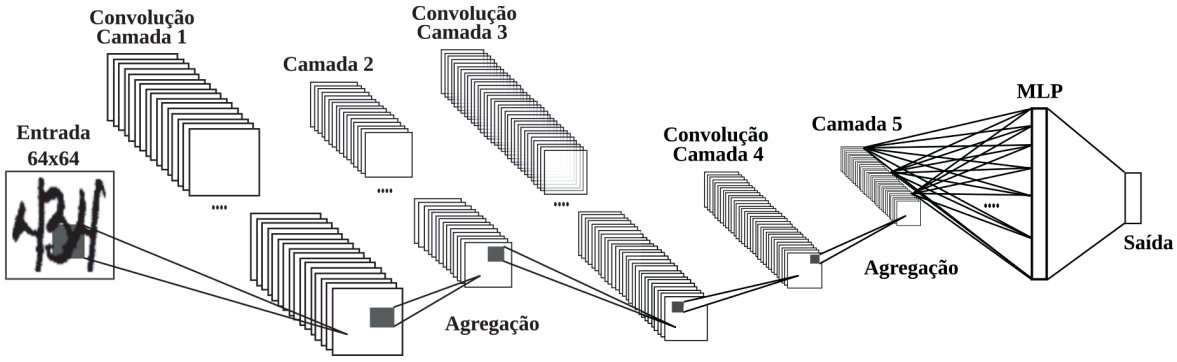
Figura 4.9: Classificações errôneas produzidas por $\mathcal{L}$: (a) dígito isolado classificado como 2-dígitos, (b) 2-dígitos classificado como 3-dígitos, (c) 3-dígitos classificado como 2-dígitos, e (d) 3-dígitos classificado como 4-dígitos.

4.2.2.2 Classificadores de Dígitos ($\mathcal{C}_1$,$\mathcal{C}_2$,$\mathcal{C}_3$)

Para reconhecer dígitos isolados e cadeias de 2- e 3-dígitos conectadas, utilizamos a arquitetura de rede ilustrada na Figura 4.10. As três CNN's ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$), as quais também são baseadas na LeNet-5 (LeCun et al., 1989; Lecun et al., 1998), compartilham a mesma estrutura porém com diferentes parâmetros como quantidade de filtros, tamanho de *kernel* e passo. A Figura 4.10 sumariza esses parâmetros, os quais foram definidos empiricamente na base de validação.

A Tabela 4.4 apresenta a quantidade de dados utilizados para treinamento, validação e teste dos modelos. Além disso, disponibiliza os tempos de treinamento e teste utilizando a arquitetura Caffe e o *hardware* mencionado na Seção 4.2.2.

Todos os três classificadores foram treinados utilizando o protocolo descrito na Seção 4.2.2 e atingiram as taxas de reconhecimento reportadas na Tabela 4.5. Como podemos observar, os classificadores obtiveram alto desempenho na base de teste, demonstrando que a CNN é capaz de aprender boas representações dos dados em três diferentes classes de problemas.



Modelos	Entrada	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5	MLP	Saída
$\mathcal{C}_1$	64x64 Binária	Convolução 7x7@3@64	Agregação 3x3@2@64	Convolução 3x3@1@128	Convolução 3x3@1@32	Agregação 3x3@2@32	128	10
$\mathcal{C}_2$	64x64 Binária	Convolução 7x7@3@72	Agregação 3x3@2@72	Convolução 3x3@1@192	Convolução 3x3@1@64	Agregação 3x3@2@64	1024	100
$\mathcal{C}_3$	64x64 Binária	Convolução 7x7@1@24	Agregação 2x2@2@24	Convolução 5x5@1@42	Convolução 5x5@1@32	Agregação 2x2@2@32	1200	1000
$\mathcal{C}_{1110}$	64x64 Binária	Convolução 7x7@1@24	Agregação 2x2@2@24	Convolução 5x5@1@42	Convolução 5x5@1@32	Agregação 2x2@2@32	1200	1110

Figura 4.10: Arquitetura dos classificadores de dígitos. Os parâmetros de cada camada estão representados por Dimensão@Passo@Mapas.

Tabela 4.4: Dados utilizados para treinar os classificadores de dígitos

Classificadores	Classes	Quantidade de Amostras (×1000)			Fonte	Treinamento	Classificação
		Treinamento	Validação	Teste		Tempo (min)	Tempo (ms)
$\mathcal{C}_1$	10	197	23	23	NIST SD 19	70[2]	0.57[2]
$\mathcal{C}_2$	100	161	53	55	Dados Sintéticos[1]	90[2]	0.60[2]
$\mathcal{C}_3$	1000	1448	484	491	Dados Sintéticos[1]	200[2]	0.63[2]
$\mathcal{C}_{1110}$	1110	1808	561	570	Dados Sintéticos[1]	183[3]	1.10[3]

(1) Cadeias Tocantes (Seção 4.1.1)
(2) NVIDIA Titan Black GPU and (3) NVIDIA Titan Xp GPU

Tabela 4.5: Taxa de reconhecimento dos classificadores de dígitos na base de teste.

Classificador	Top 1	Top 2
$\mathcal{C}_1$	99,6	99,9
$\mathcal{C}_2$	99,7	100,0
$\mathcal{C}_3$	97,7	98,9
$\mathcal{C}_{1110}$	95,9	98,6

4.2.3 Fusão

A matriz de confusão apresentada na Tabela 4.3 denota que alguns erros provenientes pelo classificador $\mathcal{L}$ podem ser recuperados utilizando também a segunda hipótese de classificação (*Top-2*). Assim, propomos um método de fusão combinando as saídas *Top-1* e *Top-2* apresentadas por $\mathcal{L}$.

Seja $\mathcal{L}^i(x) = p^i(x)$ a probabilidade de um componente x ser composto por i, tal que $i = \{1, 2, 3, 4\}$ dígitos. Seja $\mathcal{C}_1(x) = \max_{0 \leq i \leq 9} p^i(x)$, $\mathcal{C}_2(x) = \max_{0 \leq i \leq 99} p^i(x)$, e $\mathcal{C}_3(x) = \max_{0 \leq i \leq 999} p^i(x)$ serem as probabilidades sobre 10-classes, 100-classes, 1000-classes para o componente x e, seja $Top1(\mathcal{C})$ e $Top2(\mathcal{C})$ as funções que retornam as classes com a primeira e a segunda maior probabilidade, respectivamente, de um classificador $\mathcal{C}$.

Dessa forma, o método de fusão atribuirá uma classe ω_j para o componente x de acordo com a Equação 1,

$$P(\omega_j|x) \begin{cases} \text{Se } \mathcal{L}^i(x) < T, & \max(\mathcal{C}_{Top1(\mathcal{L})}(x), \mathcal{C}_{Top2(\mathcal{L})}(x)) \\ \text{Senão}, & \mathcal{C}_{Top1(\mathcal{L})}(x) \end{cases} \tag{1}$$

tal que $T = 0,95$, definido empiricamente na base de validação.

Dado que uma imagem I contém n componentes, tal que $n > 0$, a determinação das classes da cadeia numérica N é dada pela Equação 2.

$$P(N|I) = \prod_{i=1}^{n} P(\omega_j|x_i) \tag{2}$$

O processo é ilustrado na Figura 4.11. Dado a imagem I como entrada, o pré-processamento determina os componentes ($CC's$). Para cada CC ("62","5","837") são determinadas hipóteses de tamanho pelo classificador $\mathcal{L}$. Se $Top1(\mathcal{L}(x))$ obtiver uma probabilidade abaixo de T, a segunda hipótese, $Top2(\mathcal{L}(x))$, é considerada. Nesse caso, a fusão determinará o provável dígito, encontrando a maior probabilidade entre os classificadores numéricos $\mathcal{C}_{Top1(\mathcal{L})}(x)$ e $\mathcal{C}_{Top2(\mathcal{L})}(x)$. Perceba que o componente CC_1 ("62") foi erroneamente classificado como 1-dígito, utilizando o *Top-1* de $\mathcal{L}$. Mas, por apresentar uma probabilidade abaixo do limiar T, a fusão determinou a classe correta através de $max(\mathcal{C}_{Top1(\mathcal{L})}(x), \mathcal{C}_{Top2(\mathcal{L})}(x))$.

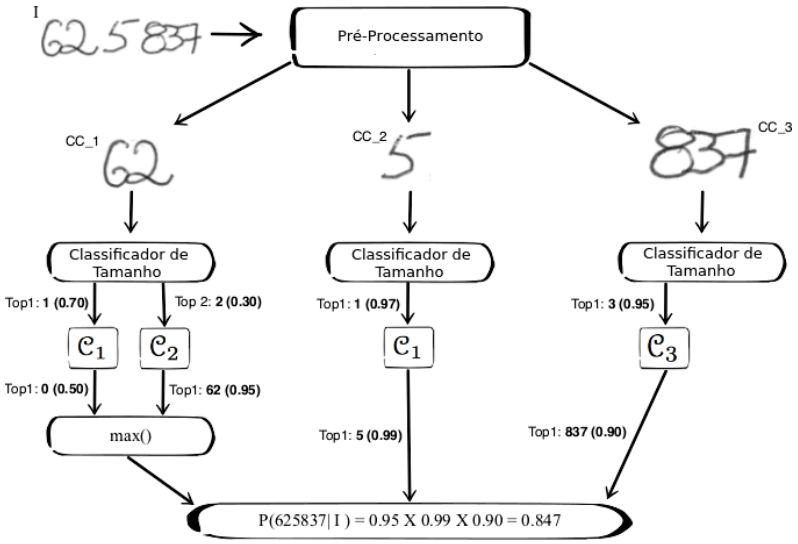


Figura 4.11: Método de Fusão

Nas hipóteses em que $\mathcal{L}$ estimar a classe 4-dígitos para um componente, o método de fusão rejeitará a cadeia, conforme descreve a Seção 4.2.3.1.

4.2.3.1 Rejeição

A rejeição de uma cadeia é dada quando para um de seus componentes for estimado a classe 4-dígitos. A Figura 4.12 ilustra o caso. É evidente que, quando apresentar probabilidades abaixo do limar T, a fusão determinará uma classe a partir do *TOP-2*.

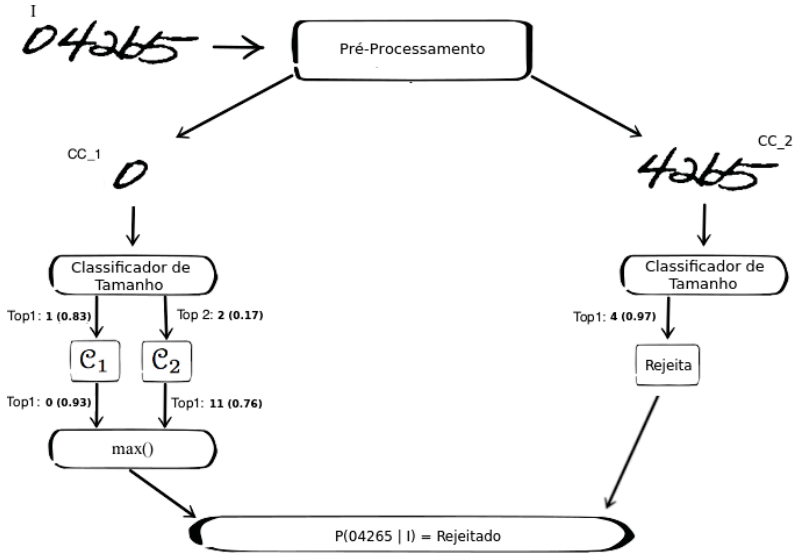


Figura 4.12: Rejeição de uma cadeia contendo componente com 4 dígitos conectados.

Apesar das amostras com 4-dígitos estarem presentes na base sintética e utilizadas no treinamento do classificador $\mathcal{L}$, implementar um classificador capaz de reconhecer dez mil classes (0000-9999) é computacionalmente custoso.

4.3 ABORDAGEM BASEADA EM 1110 CLASSES

Veremos no Capítulo 5 que a abordagem baseada em seleção dinâmica (Seção 4.2) mostrou-se bastante eficiente, superando os resultados alcançados pelas estratégias baseadas em segmentação reportadas na literatura. No entanto, em direção a uma solução de ponta-a-ponta, podemos argumentar que uma solução reduzida com um único classificador ($\mathcal{C}_{1110}$) capaz de reconhecer essas 1110 classes (10 isolados, 100 pares e 1000 trios), como aquela ilustrada na Figura 4.13, é mais elegante e fácil de implementar. Nesse caso o classificador deve codificar não só a classe, mas também o tamanho da cadeia.

Figura 4.13: Uma solução de ponta-a-ponta para dígitos conectados.

O classificador foi implementado compartilhando a mesma arquitetura do classificador de 3-dígitos ($\mathcal{C}_3$), porém com diferente número de classes. A parametrização é

apresentada na Figura 4.10. Assim como os demais classificadores, foi utilizado o protocolo de treinamento discutido na Seção 4.2.2 e utilizado a quantidade de amostras descritas na Tabela 4.4. De acordo com o exposto na Tabela 4.5, o classificador atingiu acurácias de 95,9% e 98,6% para Top-1 e Top-2, respectivamente.

De fato, a estratégia acima mencionada é simples de implementar, uma vez que é baseada em um único classificador. No entanto, algumas confusões desta solução podem ser resolvidas tendo em posse a informação a respeito do tamanho da cadeia. Com isso em mente, avaliamos a estratégia disposta na Figura 4.14, na qual combinamos a saída do classificador $\mathcal{C}_{1110}$ com a saída do classificador de tamanho ($\mathcal{L}$). Esta abordagem apresenta a mesma regra de fusão definida anteriormente pela Equação 1 na Seção 4.2.3. A diferença é que essa probabilidade é produzida por um classificador ao invés de três.

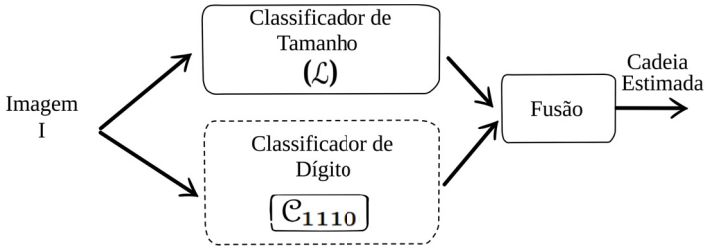
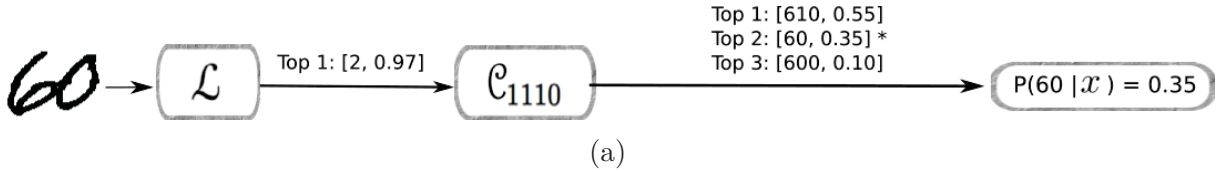
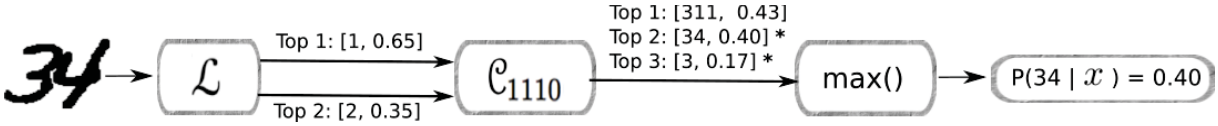


Figura 4.14: $\mathcal{C}_{1110}$ combinado com ($\mathcal{L}$).

A Figura 4.15a exemplifica o processo de fusão. Neste caso, $\mathcal{C}_{1110}$ classificou erroneamente a entrada '60' atribuindo a classe '610'. No entanto, utilizando o Top-1 de $\mathcal{L}$, a classe correta pode ser recuperada. No caso ilustrado na Figura 4.15b, devido ao fato de $\mathcal{L}$ produzir uma probabilidade baixa, as duas saídas (Top-1 e Top-2) de $\mathcal{L}$ foram utilizadas para solucionar a confusão.



(a)



(b)

Figura 4.15: Confusões realizadas por $\mathcal{C}_{1110}$: (a) '60' e (b) '34' foram corrigidas fazendo o uso da informação de tamanho provida pelo classificador $\mathcal{L}$.

4.4 ABORDAGEM PONTA-A-PONTA

Como exposto anteriormente, as abordagens tradicionais resolvem o problema agrupando pixeis em componentes conexos, e por fim, classificando cada um destes. A questão quando extraímos um grupo de pixeis da imagem é que isso representa uma visão local do problema, eliminando informações de contexto. Sem essa valiosa informação os algoritmos tendem a sofrer com a presença de ruídos e dígitos conectados.

A solução de ponta-a-ponta aborda o problema de forma global. Os modelos de aprendizagem profunda são capazes de aprender a interação entre os dígitos no contexto da imagem, a qual contém ruído, toques, sobreposição e fragmentação de dígitos. Sendo assim, a solução de ponta-a-ponta apresenta um *pipeline* muito curto: o detector de objetos recebe como entrada uma imagem I contendo n dígitos (objetos) e produz como saída a localização e classes $[0, \ldots, 9]$ associadas à uma probabilidade *a posteriori*. Considerando que a imagem de entrada I pode conter n componentes, a interpretação mais provável da quantia escrita M é dada pela Equação 3.

$$P(M|I) = \prod_{i=1}^{n} P(\omega_j|x_i) \tag{3}$$

tal que $\omega_i = \{0 \ldots 9\}$ e x_i representa os dígitos candidatos.

A Figura 4.16 compara a abordagem proposta com aquela baseada em seleção dinâmica descrita na Seção 4.2. A simplicidade do método é evidente quando comparamos os dois sistemas. Primeiro, a solução de ponta-a-ponta elimina o módulo de pré-processamento para detectar componentes conexos. Segundo, não há a necessidade de treinar múltiplos classificadores, nem elaborar um método de fusão para combiná-los. Sendo que a cadeia de dígitos é considerada uma cadeia de objetos, a abordagem não apresenta restrição com relação ao número de dígitos conectados. Por outro lado, este tipo de abordagem requer uma considerável massa de dados para ajustar os milhares de parâmetros da rede. No entanto, conforme exposto na Seção 4.1, podemos criar inúmeros dados sintéticos para solucionar essa questão.

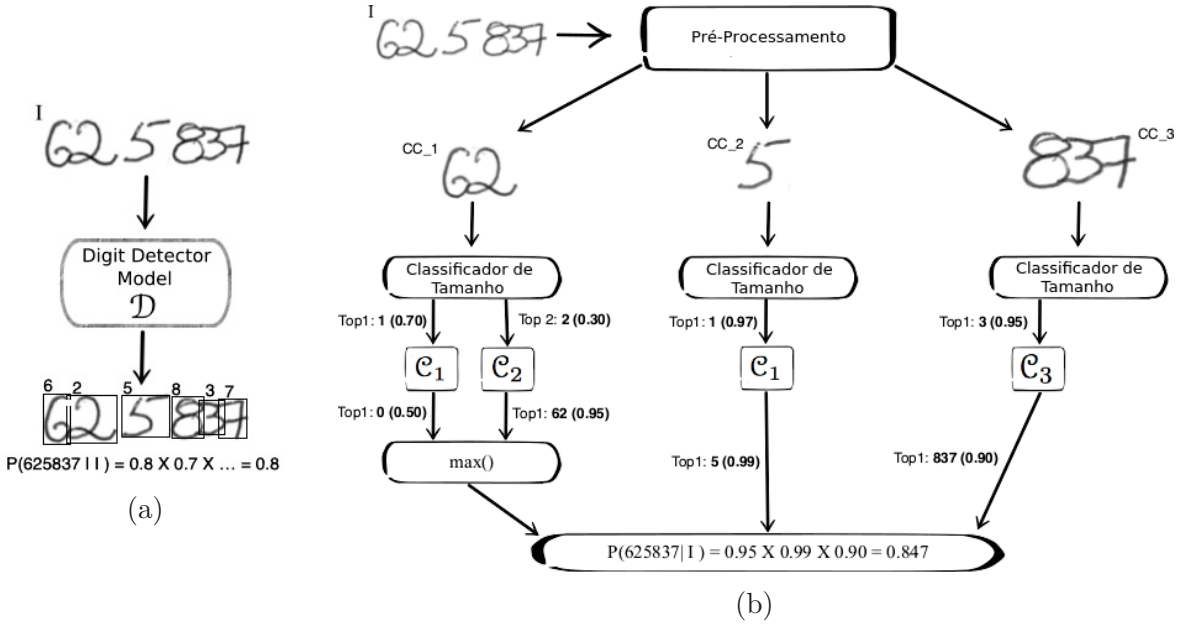


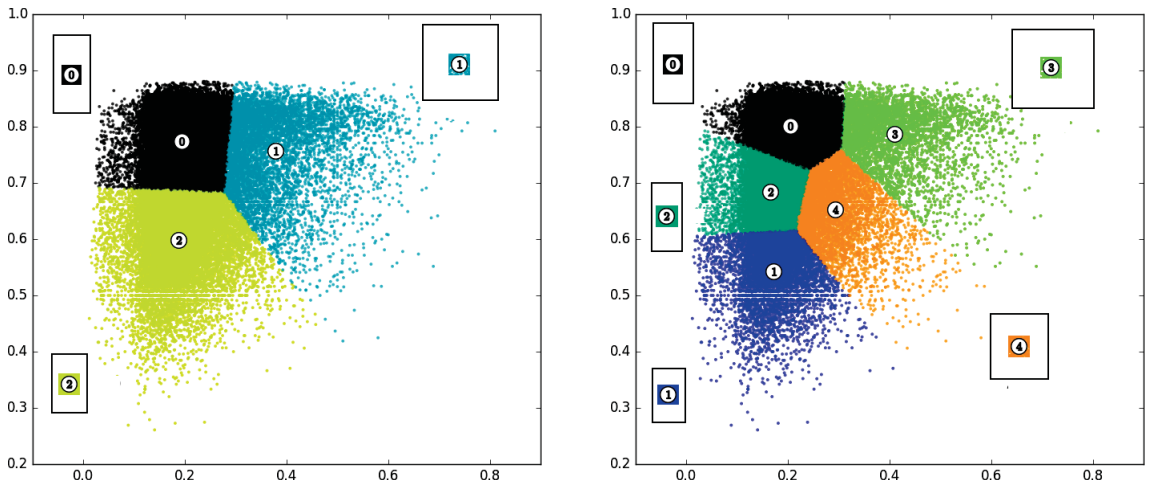
Figura 4.16: (a) Arquitetura proposta (ponta-a-ponta) e (b) Abordagem baseada em seleção dinâmica

4.4.1 Modelo Detector de Objetos

Nesta abordagem utilizaremos o *framework* YoLo (Redmon et al., 2016; Redmon e Farhadi, 2017). Conforme descrito na seção 2.5, o modelo YoLo compreende a imagem como um todo, e sendo assim, codifica implicitamente informação sobre a vizinhança dos objetos e suas interações (contexto), o que é um importante aspecto para o reconhecimento de dígitos, principalmente daqueles conectados entre si.

A resolução de entrada reportada em Redmon et al. (2016) é 416×416. No entanto, dado que cadeias numéricas são mais compridas do que altas, utilizamos o valor de entrada inicialmente em 128×256 ($altura \times largura$) para treinar o modelo. Vale ressaltar que, essa arquitetura não considera apenas uma resolução de entrada. Esse valor é alterado durante o treinamento, randomicamente, forçando a rede a aprender uma série de resoluções distintas. Na Seção 5.4 apresentamos através de alguns experimentos que durante o reconhecimento, o tamanho de entrada pode ser definido de acordo com o tamanho da imagem de teste.

Para definir as caixas-âncoras de acordo com as dimensões dos dígitos, realizamos um agrupamento (k-means) de 10.000 amostras de dígitos da base de treinamento. Para avaliar a qualidade dos grupos foi implementado o índice *Silhouette* (Rousseeuw, 1987). A Figura 4.17 ilustra a visualização do agrupamento dos dados para 3 e 5 grupos, na qual os pontos centrais representam as dimensões das respectivas âncoras. A largura e a altura são representadas pelos eixos x e y. Os valores são relativos ao tamanho da imagem de entrada. O melhor índice *Silhouette* foi encontrado para três grupos. Sendo assim, as âncoras ilustradas na Figura 4.17a foram selecionadas como parâmetros da YoLo.



(a) (b)

Figura 4.17: Visualização do agrupamento dos dados. (a) três grupos (b) cinco grupos

4.4.1.1 Treinamento

Seguindo o protocolo sugerido pelos autores em (Redmon et al., 2016; Redmon e Farhadi, 2017), primeiro treinamos as camadas convolucionais utilizando 197.784 dígitos isolados da pasta *hsf_0123* da base NIST SD19. Os pesos são ajustados para as tarefas de extração de características e classificação, não importando neste momento a localização do objeto. Para desempenhar a tarefa de detecção, o ajuste fino da arquitetura é realizado utilizando 365.000 amostras de cadeias sintéticas (Seção 4.1.2). A ideia de prover cadeias

para a tarefa de detecção é que esta possa apreender a informação contextual que não está presente em dígitos isolados. Uma discussão mais profunda sobre isso é apresentada na Seção 4.4.1.2.

O treinamento é realizado com Gradiente Descendente Estocástico usando retro-propagação com mini-lotes de 64 instâncias, fator de *momentum* de 0,9 e decaimento do peso de 5×10^{-4}. A taxa de aprendizado é de 10^{-2} no início para permitir um rápido ajuste do peso produzindo curvas abruptas. Posteriormente a taxa é reduzida gradualmente até 10^{-4} produzindo curvas suavizadas. A regularização foi implementada via parada antecipada, o que previne o sobre-ajuste, interrompendo o treinamento quando a performance da rede na base de validação deteriora-se.

4.4.1.2 A importância da informação contextual

Nesta abordagem defendemos a importância do contexto para reconhecer dígitos como objetos, e portanto, utilizar cadeias sintéticas para o treinamento. Neste momento, alguém pode questionar a necessidade disso. Por que não utilizar somente dígitos isolados para o treinamento do modelo, visto que a maioria das cadeias numéricas são compostas deste tipo de componente. Além disso, esta é a estratégia que tem sido usada no treinamento dos modelos do estado da arte.

Para responder esta questão, treinamos o modelo utilizando 197k amostras de dígitos isolados da base NIST SD19. Como era esperado, o modelo desempenhou muito bem a tarefa de localizar e reconhecer os dígitos isolados, atingindo uma taxa próxima de 99% (com 90% de IoU médio) em 23.621 dígitos isolados da pasta *hsf_7* da NIST SD19.

No entanto, quando cadeias numéricas foram apresentadas, o modelo colapsa em termos de localização e reconhecimento. O modelo tende a encontrar um único componente por imagem, uma vez que foi treinado com objetos isolados. Alguns exemplos de detecção estão ilustrados na Figura 4.18.

Além disso, no caso de cadeias numéricas, a forma dos dígitos são severamente afetados por seus vizinhos. Considere o exemplo ilustrado na Figura 4.19a. Observando a rotulação dos dígitos, a forma destes é bem diferente quando observados de forma isolada, especialmente para aqueles posicionados no meio da cadeia. Isso explica porque treinar a partir de cadeias numéricas é importante.

(a) (b) (c) (d)

Figura 4.18: Detecções do modelo treinado a partir de dígito isolados (apenas): (a) e (b) representa a correta detecção de dígitos isolados, (c) e (d) são classificações errôneas causadas por aprendizado errôneo do contexto.

(a) (b)

Figura 4.19: (a) Rotulação para a cadeia de 4-dígitos (0256) e (b) forma dos dígitos impactados pela vizinhança.

4.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os métodos resultantes deste trabalho. Como podemos observar as metologias minimizam o uso de processos heurísticos e evitam a segmentação de dígitos conectados, exposto anteriormente como a principal barreira das abordagens do estado da arte. Para validar nossas abordagens, no Capítulo 5 definimos alguns cenários para avaliar os limites, bem como compará-las com o estado da arte.

5 EXPERIMENTOS

Neste capítulo apresentamos os experimentos realizados a fim de avaliar os limites das abordagens propostas. Na Seção 5.1 avaliamos os modelos contra algoritmos de segmentação em uma base de pares conectados proposta em Ribas et al. (2012). A fim de analisar o desempenho em tarefas difíceis, como o reconhecimento de cadeias compostas por 3- e 4-dígitos conectados, amostras sintéticas dessa classe são apresentadas na Seção 5.2. Na Seção 5.3, uma base de cadeias numéricas reais, contendo diferentes formas de escrita, bem como ruídos e dígitos fragmentados é apresentada. Fechando os experimentos, realizamos uma análise com cadeias numéricas longas contendo de 2- a 20-dígitos na Seção 5.4.

5.1 SEGMENTAÇÃO DE PARES CONECTADOS

Ao analisar o estado arte, percebe-se uma grande concentração de trabalhos abordando a segmentação de pares de dígitos conectados, visto que esta classe é a que ocorre com mais frequência. Quando avaliaram algoritmos de segmentação, os autores em Ribas et al. (2012) estavam interessados em saber se os fragmentos gerados pelos algoritmos eram corretos ou não, independente da sua quantidade. Para tal, um classificador avalia os fragmentos gerados, atribuindo a estes classes entre 0 e 9 e uma probabilidade a *posteriori*. Se a classificação produzir o rótulo correto, a segmentação é considerada correta. Para aquelas abordagens que são baseadas em segmentação-e-reconhecimento a avaliação é direta, uma vez que apenas um corte é produzido. Para aquelas baseadas em sobre-segmentação, múltiplos fragmentos precisam ser avaliados. Para este último caso a estratégia é a seguinte: Se ao menos dois componentes dentre todas as hipóteses, foram classificados corretamente, a segmentação é considerada correta. É evidente que esta estratégia considera o melhor cenário, dado que falsos positivos e sobre-segmentações não estão sendo considerados.

A Tabela 5.1 resume os resultados reportados em Ribas et al. (2012) e Gattal et al. (2015) na qual os autores comparam inúmeros algoritmos em termos de correta segmentação na base de Pares Conectados (Ribas et al., 2012). Além da taxa de reconhecimento global, a tabela também revela a taxa de acordo com o tipo de conexão ilustrado na Figura 5.1. A Tabela 5.1 nos permite algumas conclusões. Algoritmos baseados em um único ponto de segmentação comumente falham em casos mais complexos, como por exemplo, conexões do tipo V, a qual dificilmente é resolvida por um único corte. Por outro lado, algoritmos baseados em múltiplos pontos de segmentação alcançam um desempenho melhor, mas com um custo elevado de produzir inúmeras hipóteses para serem avaliadas.

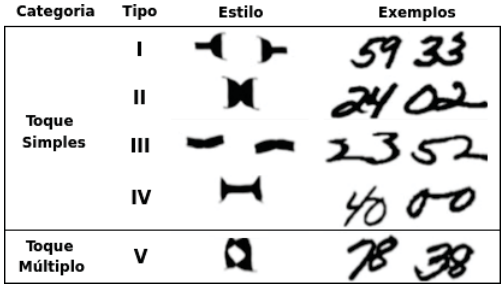
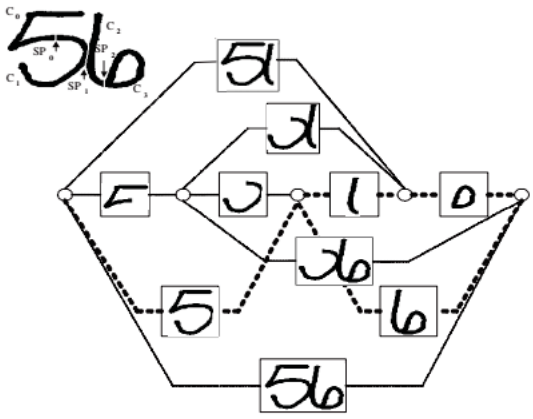


Figura 5.1: Tipos de conexões entre os numerais (extraído de (Ribas et al., 2012)).

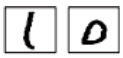
Tabela 5.1: Desempenho dos algoritmos (reportados em Ribas et al. (2012) e Gattal et al. (2015)), em termos de correta segmentação, na base de Pares Conectados (Ribas et al., 2012).

Algoritmo	Reconhecimento	Tipo de Conexão (%)				Pontos de
	%	I	II	III	V	Segmentação
Shi e Govindaraju (1997)	59,30	68,31	59,72	60,35	25,44	1
Congedo et al. (1995)	63,07	62,88	67,51	59,40	40,45	1
Lacerda e Mello (2013)	65,79	71,75	71,21	63,64	56,57	1
Elnagar e Alhajj (2003)	67,34	63,88	71,51	56,40	58,73	1
Pal et al. (2003)	71,21	73,96	74,69	80,09	41,52	1
de Oliveira et al. (2000)	88,03	90,40	90,78	89,01	64,88	1
Fujisawa et al. (1992)	89,85	95,45	91,27	83,57	63,72	3,66
Fujisawa e Krishnamoorth (1990)	92,37	97,54	93,79	99,45	65,57	4,07
Gattal et al. (2015)	93,24	96,67	93,75	99,68	77,58	24,11
Chen e Wang (2000)	93,80	97,87	94,23	97,55	76,76	45,40
$\mathcal{C}_{1110}$	94,37	94,33	95,13	96,13	91,90	0
$\mathcal{C}_{1110}$+$\mathcal{L}$	96,05	95,95	96,87	98,03	93,35	0
Ponta-a-Ponta	96,53	96,98	97,64	98,97	92,55	0
Seleção Dinâmica	97,12	97,02	97,89	98,97	93,03	0

Este problema é exemplificado na Figura 5.2. Neste exemplo, o algoritmo produziu três pontos de segmentação, resultando em 10 diferente hipóteses a serem avaliadas, para então, definir aquele conjunto que maximiza a saída no grafo de segmentação. O que acontece comumente é que fragmentos, como aqueles ilustrados na Figura 5.2b, podem ser confundidos com dígitos com altas probabilidades. Neste caso, o caminho "510" pode produzir um melhor resultado que "56". Para evitar este problema, alguns autores propõe o uso de heurísticas e filtros para reduzir o número de hipóteses (Vellasques et al., 2008). Em resumo, fragmentar demasiadamente os componentes pode até produzir boas segmentações, porém, com elevado custo computacional.



(a)



(b)

Figura 5.2: (a) Grafo de segmentação para a cadeia "56" e (b) Imagens que podem ser confundidas com as classes "0" e "1" (Vellasques et al., 2008).

Neste contexto, as abordagens livres-de-segmentação apresentadas neste trabalho comparam-se favoravelmente aos tradicionais algoritmos de segmentação. No caso de $\mathcal{C}_{1110}$, o processo custoso de encontrar pontos de segmentação, filtrar hipóteses e classificá-las é substituído por apenas uma chamada ao classificador. Como podemos observar na Tabela

5.1, esta abordagem simples atingiu 94,37% de correta classificação, o que se compara aos melhores métodos do estado da arte, Chen e Wang (2000) e Gattal et al. (2015). No entanto, esses dois métodos produzem uma grande quantidade de hipóteses, o que os torna inviáveis para aplicações reais devido ao alto custo computacional.

A Figura 5.3 ilustra algumas confusões realizadas pelo classificador $\mathcal{C}_{1110}$. A maioria dos erros estão relacionados com a errônea atribuição à classes referentes a dígitos isolados ou cadeias de 3-dígitos. A luz disso, $\mathcal{C}_{1110}$ pode se beneficiar da informação de tamanho fornecida por $\mathcal{L}$ para resolver essas confusões, a qual é a estratégia definida na Figura 4.14.

(a) (b)

Figura 5.3: Confusões realizadas por $\mathcal{C}_{1110}$ - [Classe, Probabilidade]: (a) '26' como '216' e (b) '14' com '4' podem ser resolvidas utilizando a informação de tamanho fornecida por $\mathcal{L}$

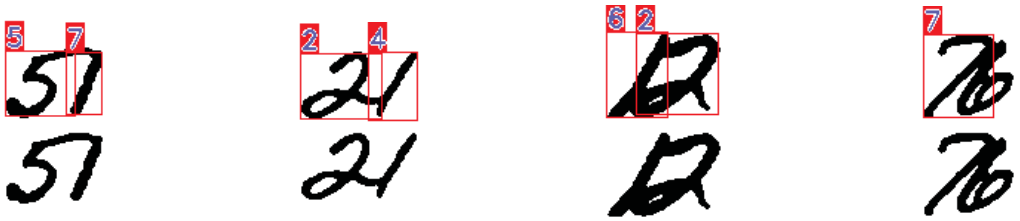
Combinando $\mathcal{C}_{1110}$+$\mathcal{L}$, algumas dessas confusões foram resolvidas aumentando a taxa de reconhecimento em quase dois porcento (96,05%). Do erro total (3,95%), 1,76% é causado por $\mathcal{L}$ e 2,79% por $\mathcal{C}_{1110}$. Em termos de custo computacional, temos um pequeno acréscimo devido a consulta a classificador $\mathcal{L}$ e a regra de fusão. No entanto, podemos desconsiderar este custo se compararmos aos métodos tradicionais de segmentação.

A abordagem baseada em seleção dinâmica apresentada na Figura 4.4 soluciona algumas confusões causadas por $\mathcal{C}_{1110}$. Ao invés de aplicar um único classificador para dígitos isolados e cadeias de 2- e 3-dígitos conectados, este divide a tarefa em três partes, criando para isso classificadores de propósito específico ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$). A decisão de qual classificador será designado para uma dada entrada é baseado na informação de tamanho fornecida por $\mathcal{L}$. Assumindo que a entrada é desconhecida neste experimento, o classificador $\mathcal{C}_2$ será invocado quando $\mathcal{L}$ atribuir a classe 2-dígitos ou o método de fusão mitigar alguma confusão. Essa estratégia alcançou o melhor desempenho (97,12%). Comparando com $\mathcal{C}_{1110}$+$\mathcal{L}$, o erro de classificação foi reduzido de 2,79% para 1,10%. A Figura 5.4 ilustra algumas imagens que foram classificadas erroneamente por $\mathcal{C}_2$. Conforme reportado na Tabela 5.1, o pior desempenho foi aquele alcançado para o tipo V (toque múltiplo), o qual apresenta grande variabilidade. No entanto, quando comparado aos métodos de segmentação, o desempenho é excepcional.

(a) (b)

Figura 5.4: Confusões realizadas por $\mathcal{C}_2$ - [Classe, Probabilidade]: (a)'15' estimado como '16', (b) '51' estimado como '57'.

A abordagem de ponta-a-ponta possui uma mecânica totalmente diferente. Primeiro, não contém módulos de pré-processamento e fusão, o que reduz a solução a uma segmentação implícita. Segundo, codifica em um único classificador a localização e a classe do componente. Deste modo, não necessita de um segundo classificador ($\mathcal{L}$) para produzir informação contextual. A solução proposta apresentou desempenho semelhante a $\mathcal{C}_{1110}$+$\mathcal{L}$ e aquela baseada em seleção dinâmica, em termos de taxa de reconhecimento, para a maioria dos tipos de conexão, exceto pelo tipo V. Neste caso, o classificador designado para dois dígitos ($\mathcal{C}_2$) consegue lidar de maneira mais eficiente com imagens inclinadas e altamente conectadas. A Figura 5.5 ilustra algumas confusões.



(a) (b) (c) (d)

Figura 5.5: Detecções errôneas da abordagem Ponta-a-Ponta: (a) ‘51’ como ‘57’, (b) ‘21’ como ‘24’, (c) ‘12’ como ‘62’ e (d) ‘76’ como ‘7’.

5.2 BASE DE CADEIAS SINTÉTICAS CONECTADAS

O objetivo deste experimento é avaliar os limites das abordagens livres de segmentação em tarefas difíceis, como por exemplo, cadeias compostas por dígitos conectados. Como apontado anteriormente, isso não é muito comum em casos reais, especialmente aqueles casos conectados por 3- e 4- dígitos, mas é útil para avaliar os limites de cada abordagem.

Além disso, alguém pode argumentar que o reconhecimento de dígitos isolados é um problema resolvido visto que a literatura apresenta taxas de reconhecimento próximas a 100% (Ciresan et al., 2012; Sabour et al., 2017). É importante lembrar, no entanto, que a falta de contexto em dígitos aumenta a complexidade do problema, uma vez que uma cadeia pode conter dígitos isolados ou conectados. Para lidar com este problema, algoritmos de segmentação confiam em estratégias de sobre-segmentação para maximizar as chances de encontrar a classe correta, mesmo quando a segmentação não é necessária. Essa estratégia tem um problema, dígitos isolados que não precisam ser segmentados geram fragmentos que são reconhecidos erroneamente com alta probabilidade. Isso é ilustrado na Figura 5.6, na qual o dígito “9” é fragmentado em “0” e “1” com alta probabilidade.

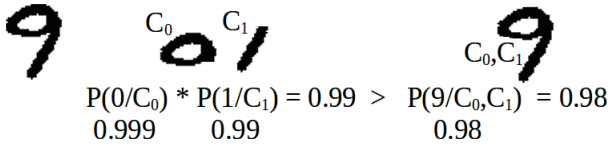


Figura 5.6: Classificação errônea causada por sobre-segmentação (extraído de (de Oliveira et al., 2002)).

Nesta seção avaliamos as abordagens livre-de-segmentação em 23,621 dígitos isolados extraídos da pasta *hsf_7* da base NIST SD 19, além de 560,000 amostras de imagens sintéticas descritas na Seção 4.1.1, a qual contém cadeias conectadas por 2-, 3- e 4-dígitos. A Tabela 5.2 apresenta as taxas de reconhecimento.

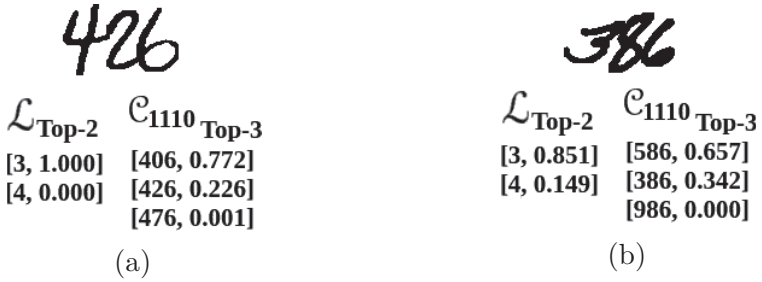
Tabela 5.2: Taxa de reconhecimento das abordagens na base sintética.

Método	1-dígitos	2-dígitos	3-dígitos	4-dígitos
$\mathcal{C}_{1110}$	97,68	94,09	96,05	-
$\mathcal{C}_{1110}$+$\mathcal{L}$	98,73	96,82	95,50	-
Ponta-a-Ponta	99,42	98,68	96,89	94,88
Seleção Dinâmica	99,56	99,00	94,88	-

Apesar do fato que manipular quatro classificadores ($\mathcal{L}$, $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$) é uma tarefa complexa, a abordagem baseada em seleção dinâmica apresenta o melhor resultado. Uma vez que $\mathcal{C}_{1110}$ discrimina essas 1110 classes em um único classificador, esse produz mais confusões pois tem mais informações para classificar do que os classificadores especializados da primeira estratégia. Para ilustrar, se um dígito isolado estiver presente, o classificador $\mathcal{C}_1$ irá discriminar entre 10 classes, enquanto que o classificador $\mathcal{C}_{1110}$ continuará discriminando 1110 classes.

A informação contextual fornecida por $\mathcal{L}$ corrobora na melhora da taxa de acerto de $\mathcal{C}_{1110}$. Inúmeras confusões realizadas entre dígitos isolados e cadeias tocantes são resolvidas utilizando a informação sobre a quantidade de dígitos presente na cadeia. Vale ressaltar que essa informação é também utilizada na seleção dinâmica para enumerar os classificadores específicos.

No caso de cadeias contendo 3-dígitos, o que não é muito frequente em base de dados reais, a maioria das confusões ocorre intra-classe, por exemplo, “426” confundida com “406” conforme ilustra a Figura 5.7. Uma vez que cadeias com 3-dígitos contem mais informação para codificar, o tamanho da cadeia fornecido pelo classificador $\mathcal{L}$ não contribuí para o aumento da taxa de reconhecimento. Por outro lado, estratégias baseadas em segmentação irão sofrer com o alto número de hipóteses a serem avaliadas.



(a) (b)

Figura 5.7: Confusões realizadas por $\mathcal{C}_{1110}$ - [Classe, Probabilidade]: (a) ‘426’ discriminado como ‘406’ e (b) ‘386’ discriminado como ‘586’.

A abordagem de ponta-a-ponta atingiu um desempenho muito próximo a abordagem baseada em seleção dinâmica e $\mathcal{C}_{1110}$+$\mathcal{L}$, no entanto, superando dois obstáculos: (a) produz um *pipeline* curto quando comparada as outras abordagens e (b) não há restrição a respeito da quantidade de dígitos tocantes.

5.3 CADEIAS NUMÉRICAS REAIS

Os experimentos utilizando cadeias numéricas são baseados em 11.585 numerais extraídos da pasta hsf_7 da base de dados reais NIST SD19 e estão divididas em cinco

classes: 2.370 (2-dígitos), 2.385 (3-dígitos), 2.345 (4-dígitos), 2.316 (5-dígitos) e 2.169 (6-dígitos). Esses numerais apresentam em sua composição tanto dígitos isolados, quanto componentes conectados. Também, a presença de fragmentos de dígitos e variabilidade de escrita são problemas inerentes ao processo. Esse protocolo é praticado por pelos autores (Ha et al., 1998; de Oliveira et al., 2002; Oliveira e Sabourin, 2004; de S. Britto et al., 2003; Sadri et al., 2007; Lee e Kim, 1999; Liu et al., 2004), além disso, é importante mencionar que as amostras da pasta hsf_7 não foram utilizadas em momento algum para os treinamento dos modelos, nem mesmo para a criação de dados sintéticos.

Vale ressaltar que a abordagem baseada em 1110 classes ($\mathcal{C}_{1110}$) não é contemplado neste experimento, pois a mesma é apresentada como solução comparativa aquela baseada em seleção dinâmica para dígitos conectados apenas, não possuindo módulo de detecção de componentes conexos.

A Tabela 5.3 apresenta as taxas de acerto para a abordagem baseada em seleção dinâmica. Além disso é apresentado a distribuição do erro de acordo com cada módulo: (a) pré-processamento, (b) estimativa do tamanho ($\mathcal{L}$) e (c) classificadores de dígitos ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$). A última coluna apresenta a porcentagem de cadeias rejeitadas por conter componentes com 4-dígitos conectados.

Tabela 5.3: Taxa de reconhecimento para a abordagem baseada em seleção dinâmica na base NIST SD19

Tamanho	Amostras	Reconhecimento	Erros (%)			(4-dígitos)
		(%)	Pré-Processamento	Tamanho($\mathcal{L}$)	Classificação	(Rejeição)
2	2370	97,6	0,2	0,5	1,7	0,0
3	2385	96,2	0,5	0,6	2,7	0,0
4	2345	94,6	0,8	0,8	3,5	0,3
5	2316	94,1	1,1	1,5	3,2	0,1
6	2169	93,3	1,3	1,5	3,6	0,3
	Média	95,2	0,8	1,0	2,9	0,1

Como podemos observar as principais fonte de erros são a estimativa na classe do dígito (2,9%), estimativa do tamanho (1,0%) e erros causados durante o pré-processamento (0,8). A presença de cadeias contendo 4-dígito é muito pequena, conforme exposto anteriormente. A Figura 5.8 ilustra alguns exemplos. Na parte superior denota-se a amostra original, o rótulo (*Input*), a classe e a probabilidade para a cadeia (*Pred*). Na parte inferior, os componentes conexos e suas respectivas classes e probabilidades estimadas para tamanho (L) e dígito (D). Os retângulos em vermelhos denotam o componente classificado erroneamente. Por exemplo, a Figura 5.8(a), o primeiro componente, "34", foi estimado corretamente por $\mathcal{L}$, 2-dígitos, mas $\mathcal{C}_2$ atribuiu a este a classe "31" ao invés "34"

No que se refere a abordagem de Ponta-a-Ponta os resultados são apresentados na Tabela 5.4. Neste caso as fontes de erro são: (a) detecção, quando o dígito não é encontrado, e (b) classificação, quando o dígito é detectado mas estimado a classe incorreta. Algumas amostras estão ilustradas na Figura 5.9.

A Tabela 5.5 compara as taxas de reconhecimento de inúmeras propostas publicadas sobre a base NIST SD19. Na primeira parte da tabela nos agrupamos aqueles trabalhos que utilizam o mesmo protocolo. Britto et al. (2003) apresenta um estratégia baseada em segmentação, a qual utiliza dois HMM's para processar as informações extraídas da imagem. Oliveira et al. (2002) propõe um sistema baseado sobre segmentação com módulos que implementam perceptrons multi-camadas (MLP) para reduzir o número de hipotiposes

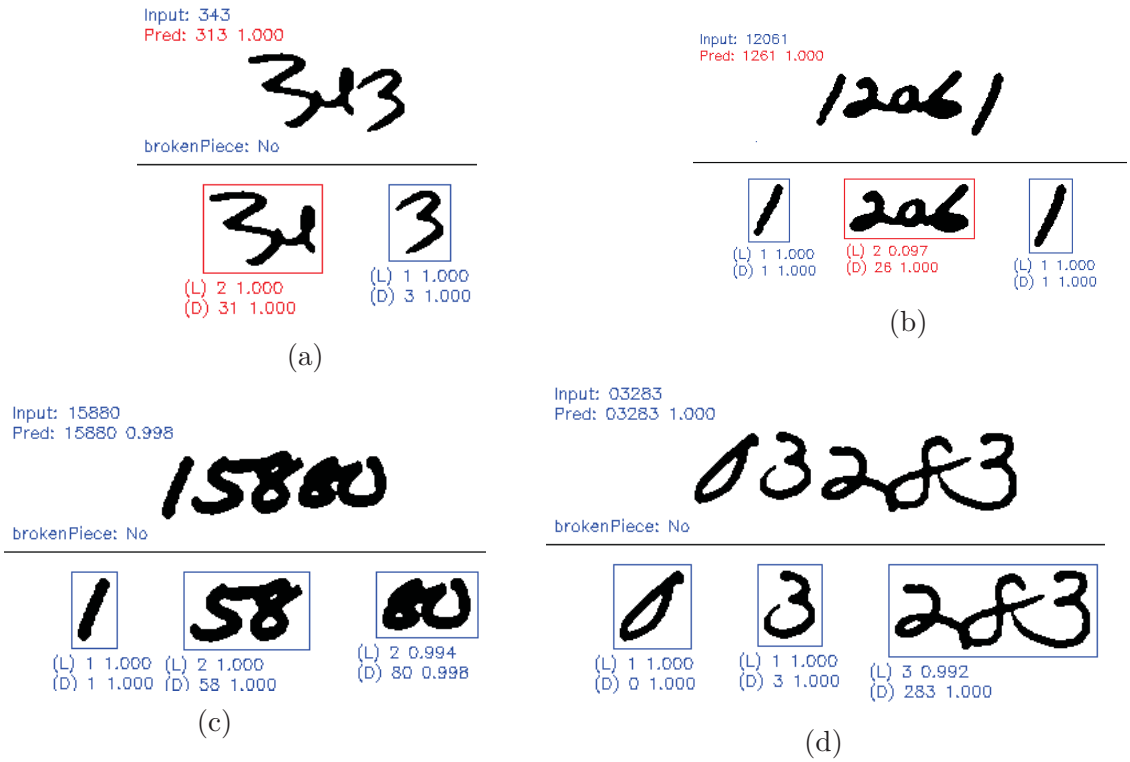


Figura 5.8: Amostras da base *NIST SD19* avaliadas pela abordagem baseada em seleção dinâmica: (a) e (b) não foram reconhecidas enquanto que, (d) e (e) foram reconhecidas.

Tabela 5.4: Taxa de reconhecimento da abordagem Ponta-a-Ponta em cadeias da NIST SD19

Tamanho	Amostras	Reconhecimento (%)	Erros (%)	
			Classificação	Detecção
2	2370	98,57	1,39	0,04
3	2385	97,61	2,32	0,08
4	2345	97,10	2,56	0,34
5	2316	96,50	2,59	0,91
6	2169	95,80	3,14	1,06
	Média	97,12	2,40	0,49

geradas. Em Oliveira e Sabourin (2004) os MLP's foram substituídos por máquinas de vetores de suporte (SVM) obtendo um incremento na taxa de reconhecimento.

Sadri et al. (2007) também apresentaram um método baseado em sobre-segmentação, no qual é aplicado um algoritmo genético para encontrar o melhor o ponto de segmentação, dentre as múltiplas hipóteses, adicionando dessa forma mais complexidade ao problema. Neste mesmo trabalho, apresentaram um segundo experimentos (denotados pela marca '*' na Tabela 5.5) no qual definem um conjunto de heurísticas para melhorar a busca implementada pelo algoritmo genético. Apesar de melhora nas taxas de reconhecimento, a proposta apresenta viés uma vez que essas heurísticas foram definidas sobre um sub-conjunto da base de teste.

Outro algoritmo custoso baseado em sobre-segmentação foi proposto recentemente por Gattal et al. (2017). O algoritmo combina informações do contorno, esqueleto, além de aplicar uma janela deslizante. Os autores reportaram resultados bastante interessantes,

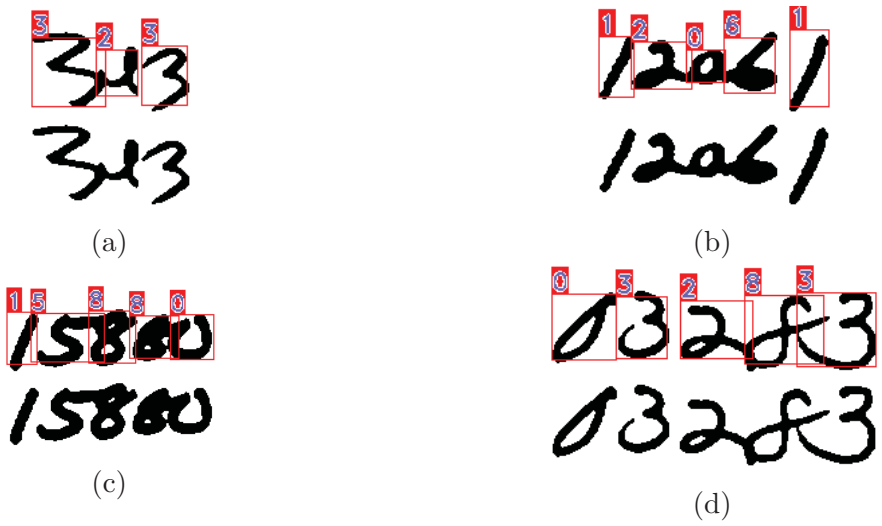


Figura 5.9: Amostras da base *NIST SD19* avaliadas pela abordagem Ponta-a-Ponta: (a) classificação incorreta estimando o componente "343" como "323", enquanto que (b), (c) e (d) são classificações corretas. A parte superior denota-se as detecções e suas respectivas classes, e na parte inferior a amostra original.

Tabela 5.5: Comparação dos métodos em termos de reconhecimento na base NIST SD19

Tamanho	Amostras	Britto et al. (2003)	Oliveira et al. (2002)	Oliveira e Sabourin (2004)	Sadri et al. (2007)	* Sadri et al. (2007)	Gattal et al. (2017)	Shi et al. (2017)	**Sistema Modular**	**Ponta-a-Ponta**	Amostras	Liu et al. (2004)	Ciresan (2008)
2	2370	94,8	96,8	97,6	95,5	98,9	99,0	70,3	97,6	98,6			
3	2385	91,6	95,3	96,2	91,4	97,2	97,3	84,4	96,2	97,6	1476	96,8	93,4
4	2345	91,3	93,3	94,2	91,0	96,1	96,5	86,8	94,6	97,1			
5	2316	88,3	92,4	94,0	88,0	95,8	95,9	83,8	94,1	96,5			
6	2169	89,0	93,1	93,8	88,6	96,1	96,6	76,3	93,3	95,8	1471	96,7	
Média		91,0	94,2	95,2	90,9	96,8	97,1	80,3	95,2	97,1		96,7	93,4

no entanto, muitos parâmetros foram ajustados na base de teste, o que torna difícil avaliar os resultados. A base de validação foi utilizada apenas para treinar os classificadores.

Uma análise mais profunda dos sistemas acima mencionados pode mostrar que todos eles enfrentam dois gargalos, tais como, a criação de inúmeras hipóteses de segmentação e então a implementação de alguma estratégia para reduzi-las para permitir o reconhecimento da cadeia em um tempo razoável. Ambos os problemas são sub-produtos da segmentação, os quais são evitados pelos nossos métodos. A não dependência da segmentação, associada à aprendizagem de representação, simplifica o design do sistema, pois não há necessidade de definir características manualmente para segmentação ou

reconhecimento. Além disso, o desempenho das nossas abordagens é comparável ao obtido na literatura.

Alguém pode argumentar que os recentes modelos baseados em redes neurais recorrentes tem obtidos resultados excepcionais no reconhecimento de palavras contidas em imagens, e então, porque não replicar esta abordagem em dígitos. Para validar tal questionamento, implementamos o modelo proposto por Shi et al. (2017) (Seção 3.5.1) utilizando para treinamento 365.000 amostras da base sintética apresentada na Seção 4.1.2.

Quando aplicado em palavras os autores reportaram taxas de 97% com auxílio do léxico e 78% sem léxico. Evidente que a introdução de contexto (léxico) contribui para mitigar algumas confusões. No entanto, essa abordagem não é possível em cadeias numéricas. A lógica proposta pelo o autor para casos sem léxico é eliminar na saída do modelo as classes adjacentes que sejam idênticas, conforme ilustra a Figura 5.10. Sendo assim, a abordagem alcançou apenas 80,3% no reconhecimento de cadeias numéricas, um taxa muito próxima aquela reportada para palavras.

Figura 5.10: Saídas de uma Rede Neural Recorrente (Shi et al., 2017) aplicada a (a) palavras e (b) cadeia numérica.

5.4 CADEIAS NUMÉRICAS LONGAS

Neste experimento, avaliamos o tamanho da entrada da rede (retina) em relação ao tamanho da cadeia para a abordagem ponta-a-ponta. Como mencionamos na Seção 2.5, as imagens foram redimensionadas para 128×256 ($altura \times largura$) para treinamento. No entanto, a YoLo altera o tamanho da entrada aleatoriamente durante o treinamento, uma vez que essa rede é capaz de reconhecer as imagens em diferentes escalas. A questão é como redimensionar corretamente a entrada da rede durante o teste a fim de maximizar o desempenho. Esse é um problema relevante, pois a largura da imagem pode variar consideravelmente de acordo com o número de dígitos da cadeia. Uma cadeia de 20-dígitos é consideravelmente maior que uma de 2-dígitos, por exemplo. Redimensionar ambas para 128×256 não é uma boa alternativa. Evidente, que para as abordagens baseadas em componente conexo ($\mathcal{L}$, $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_{1110}$) as proporções dos dígitos não são afetadas pelo tamanho da cadeia.

Para atacar este problema, realizamos um experimento sobre 5.000 cadeias numéricas de 2 à 20 dígitos, as quais foram sinteticamente criadas concatenando dígitos isolados da NIST SD 19. Para cada comprimento de cadeia, testamos o redimensionamento da entrada nas seguintes larguras $[128, 256, 384, 512, 640, 768, 896, 1024, 1152, 1280]$. A altura foi sempre mantida em 128. A Tabela 5.6 resume o tamanho da entrada (retina) que maximiza a taxa de reconhecimento em função do comprimento da cadeia.

A partir da Tabela 5.6 podemos notar que existe uma relação quase linear entre a largura média da imagem de teste e o melhor redimensionamento para a YoLo. Sendo

Tabela 5.6: Tamanho de entrada da rede (retina) que maximiza a taxa de reconhecimento em função do comprimento da cadeia

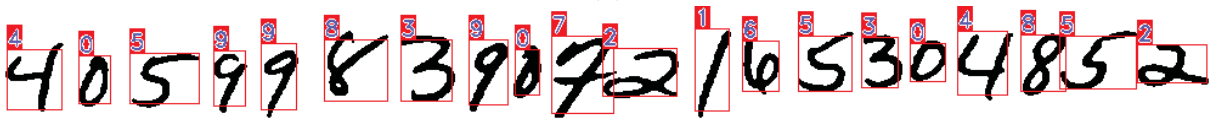
Comprimento Cadeia	Largura Média da Cadeia (S_w)	Tamanho de Entrada (II_w) ($128 \times w$)	Taxa de Reconhecimento (%) (2-6) / (2-20)
2	75	128	98,6 / 99.0
4	150	256	97,6 / 98.2
6	228	384	97,6 / 98.0
8	306	512	96,4 / 97.0
10	381	640	94,8 / 96.4
12	448	768	94,2 / 95.4
14	524	896	91,0 / 92.8
16	596	1024	90,6 / 93.2
18	666	1152	88,8 / 93.2
20	750	1280	91,2 / 95.6

assim, propomos uma regra (Equação 1) para definir o tamanho de entrada em função do tamanho da imagem de teste. Esta regra é utilizada em todos os experimentos reportados.

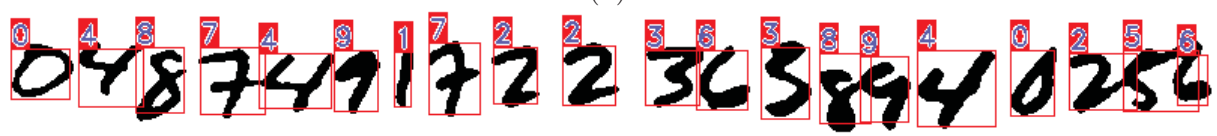
$$II_w = \begin{cases} 128 & \text{se } S_w \leqslant 75 \\ S_w \times 1,70 & \text{senão} \end{cases} \tag{1}$$

A Figura 5.11 mostra alguns exemplos de cadeias com 20-dígitos reconhecidas utilizando a regra mencionada acima. Isso confirma a eficiência da estratégia de redimensionamento adotada e mostrando que a abordagem proposta pode tem um bom desempenho mesmo que com cadeias longas, compostas por dígitos fragmentados e conectados.

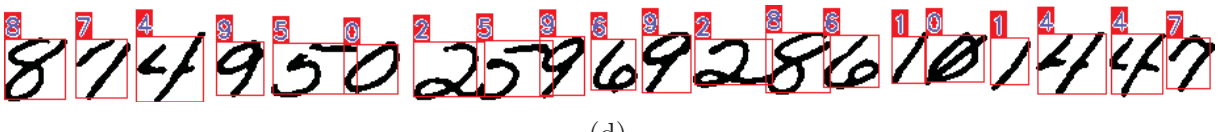
(a)



(b)



(c)



(d)

Figura 5.11: Cadeias de 20-dígitos corretamente classificadas pela abordagem ponta-a-ponta.

6 CONCLUSÕES

Dentre as metodologias propostas para reconhecimento de cadeias numéricas manuscritas, principalmente no que tange dígitos conectados, a segmentação destes é um consenso na literatura (Ribas et al., 2012; Kulkarni e Vasambekar, 2010; Saba et al., 2010). Sendo esse processo uma fonte de ruído, por exemplo, a produção de dígitos fragmentados, alguns autores apresentaram abordagens livres-de-segmentação (Choi e Oh, 1999; Lecun et al., 1998; Ciresan e Pescaru, 2008). No entanto, a literatura mostra que as abordagens de segmentação-baseadas-no-reconhecimento denotam taxas de reconhecimento superiores, todavia, com elevado custo computacional devido ao número de consultas ao classificador pelas múltiplas hipóteses geradas. Acerca deste, a falta de contexto, como a determinação do tamanho da cadeia, torna o processo ainda mais complexo.

Motivados pelo avanço dos algoritmos de reconhecimento de padrões, em especial, aqueles que implementam técnicas de aprendizado profundo, tais como as Redes Neurais Convolucionais (CNN's), postulamos neste trabalho que é possível aprender a variabilidade de dígitos conectados e assim reconhecer cadeias numéricas sem a necessidade de segmentá-las.

Através da implementação de classificadores para estimar o tamanho de um componente numérico e para reconhecer componentes conectados por até três dígitos (0..9, 00..99 e 000...999), demonstramos o potencial das nossas abordagens a partir de experimentos realizados em bases conhecidas. Quando comparamos as duas abordagens provenientes desta estratégia, aquela baseada em seleção dinâmica (Seção 4.2) se mostrou bastante eficiente, enquanto que aquela baseada em um único classificador de 1110 classes (Seção 4.3), apresenta uma menor complexidade com taxa de reconhecimento muito próxima.

Também, ao compreender dígitos como objetos em uma imagem, apresentamos uma nova abordagem para o problema. Um modelo de detecção de dígitos, criado a partir do *framework* YoLo, é capaz de codificar a vizinhança dos dígitos e suas interações, reconhecendo com sucesso uma cadeia numérica, contendo tanto dígitos isolados e conectados. Nesse caso, uma sequência de dígitos é compreendida como uma sequência de objetos. Esse tipo de abordagem entrega uma solução de ponta-a-ponta, sem restrição quanto ao tamanho da cadeia e dígitos conectados.

Outra contribuição importante do nosso trabalho diz respeito a utilização dos dados sintéticos para o treinamento dos modelos. Uma das desvantagens ao utilizar modelos baseados em aprendizado profundo é a quantidade de dados necessária para o correto ajuste do parâmetros destes. Neste caso, a utilização de cadeias numéricas sintéticas apresentou-se como uma alternativa viável para o processo, provendo representação suficiente para o aprendizado.

6.1 TRABALHOS FUTUROS

Os relevantes resultados alcançados pelas abordagens propostas neste trabalho, nos direcionam para novos questionamentos.

No contexto deste trabalho, comparamos as abordagens baseadas em aprendizado profundo contra abordagens heurísticas, utilizamos para isso bases do estado da arte à época destes trabalhos (NIST SD19/Pares Conectados). Apesar das bases apresentarem

inúmeros casos de dígitos conectados e fragmentados, não está presente maiores desafios relacionados com o fundo ou ruídos provenientes da aquisição. Recentemente, novas e mais desafiadoras bases reais foram propostas, como CVL e ORAND-CAR (Diem et al., 2014), as quais são compostas de valores numéricos extraídos de cheques bancários contendo, além dos próprios dígitos, texturas e símbolos comumente usados em montantes financeiros. A Figura 6.1 ilustra alguns exemplos. A motivação aqui é avaliar nossas abordagens nessas bases, identificando possíveis limitações e como superá-las.

Figura 6.1: Exemplos da Base ORAND-CAR: Além de conter dígitos, estão presentes símbolos e texturas.

Outro experimento plausível para as nossas abordagens é o desempenho em reconhecimento de cadeias de caracteres (palavras). Em um primeiro momento os desafios parecem similares, no entanto, a introdução de léxico, além de um número maior de classes para os modelos, são pontos a serem avaliados.

Recentemente, abordagens denominadas sequência-a-sequência (Shi et al., 2017; Voigtlaender et al., 2016; Dutta et al., 2018) tem alcançado o estado da arte ao reconhecer cadeias de caracteres presentes em imagens do cotidiano, tais como placas de propaganda, estampas de camisetas, entre outros. Algumas dessas imagens são ilustradas na Figura 6.2. Essas abordagens utilizam modelos de aprendizagem profunda para reconhecer padrões de sequências (cadeias, por exemplo) e contexto. No entanto, esses modelos não foram avaliados mais profundamente em cadeias numéricas, e acreditamos ser admissível a aplicação destes para tal caso.

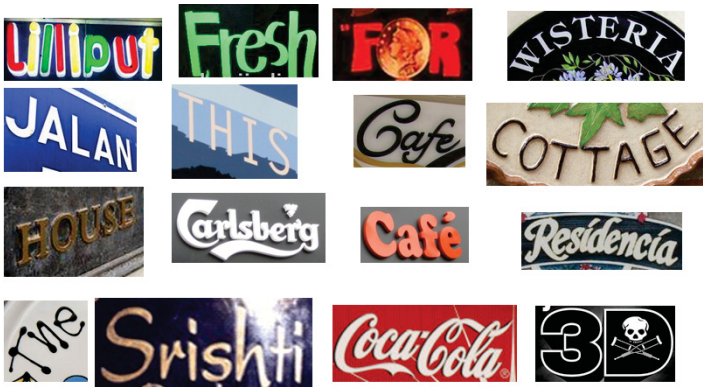


Figura 6.2: Amostras da Base IIIT5K (Mishra et al., 2012) contendo cadeias de caracteres extraídas a partir imagens.

REFERÊNCIAS

Bengio, Y. (2009). Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127.

Bengio, Y., Courville, A. e Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

Bengio, Y., Simard, P. e Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Casey, R. G. e Lecolinet, E. (1996). A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):690–706.

Chen, Y.-K. e Wang, J.-F. (2000). Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1304–1317.

Choi, S.-M. e Oh, I.-S. (1999). A segmentation-free recognition of two touching numerals using neural network. Em *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, páginas 253–256.

Chuang, Z., Zhiqing, L. e Jun, G. (2005). Study on segmentation algorithm for unconstrained handwritten numeral strings. Em *Neural Networks and Brain, 2005. ICNN B '05. International Conference on*, volume 2, páginas 1242–1247.

Ciresan, D., Meier, U., Masci, J. e Schmidhuber, J. (2011). A committee of neural networks for traffic sign classification. Em *Neural Networks (IJCNN), The 2011 International Joint Conference on*, páginas 1918–1921.

Ciresan, D., Meier, U. e Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. Em *2012 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 3642–3649.

Ciresan, D. C. (2008). Avoiding segmentation in multi-digit numeral string recognition by combining single and two-digit classifiers trained without negative examples. Em *SYNASC 2008, 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 26-29 September 2008*, páginas 225–230.

Ciresan, D. C. e Pescaru, D. (2008). Off-line recognition of handwritten numeral strings composed from two-digits partially overlapped using convolutional neural networks. Em *IEEE Proceeding ICCP 2008*, Cluj-Napoca, Romania.

Congedo, G., Dimauro, G., Impedovo, S. e Pirlo, G. (1995). Segmentation of numeric strings. Em *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, páginas 1038–1041. IEEE.

de Oliveira, L. E. S., Lethelier, E., Bortolozzi, F. e Sabourin, R. (2000). A new segmentation approach for handwritten digits. Em *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, páginas 323–326 vol.2.

de Oliveira, L. E. S., Sabourin, R., Bortolozzi, F. e Suen, C. Y. (2002). Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(11):1438–1454.

de S. Britto, A., Sabourin, R., Bortolozzi, F. e Suen, Y. C. (2003). The recognition of handwritten numeral strings using a two-stage hmm-based method. *International Journal on Document Analysis and Recognition*, 5(2):102–117.

Deng, L. e Yu, D. (2014). Deep learning: Methods and applications. *Found. Trends Signal Process.*, 7(3–4):197–387.

Diem, M., Fiel, S., Kleber, F., Sablatnig, R., Saavedra, J. M., Contreras, D., Barrios, J. M. e Oliveira, L. S. (2014). Icfhr 2014 competition on handwritten digit string recognition in challenging datasets (hdsrc 2014). Em *2014 14th International Conference on Frontiers in Handwriting Recognition*, páginas 779–784.

Ding, W., Suen, C. Y. e Krzyzak, A. (2008). A new courtesy amount recognition module of a check reading system. Em *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, páginas 1–4.

Dutta, K., Krishnan, P., Mathew, M. e Jawahar, C. V. (2018). Improving cnn-rnn hybrid networks for handwriting recognition. Em *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, páginas 80–85.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Elnagar, A. e Alhajj, R. (2003). Segmentation of connected handwritten numeral strings. *Pattern Recognition*, 36(3):625–634.

Felzenszwalb, P., McAllester, D. e Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. Em *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, páginas 1–8. IEEE.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D. e Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

Fujisawa, H., Nakano, Y. e Kurino, K. (1992). Segmentation methods for character recognition: from segmentation to document structure analysis. *Proceedings of the IEEE*, 80(7):1079–1092.

Fujisawa, R. e Krishnamoorth, S. (1990). Segmenting diverse quality handwritten digit strings in near real-time. *5nd USPS Advanced Technology Conference*, páginas 523–537.

Gattal, A. e Chibani, Y. (2012). Segmentation and recognition strategy of handwritten connected digits based on the oriented sliding window. Em *ICFHR*, páginas 297–301.

Gattal, A. e Chibani, Y. (2015). Svm-based segmentation-verification of handwritten connected digits using the oriented sliding window. *International Journal of Computational Intelligence and Applications*, 14(1).

Gattal, A., Chibani, Y. e Hadjadji, B. (2017). Segmentation and recognition system for unknown-length handwritten digit strings. *Pattern Analysis and Applications*, 20(2):307–323.

Gattal, A., Chibani, Y., Hadjadji, B., Nemmour, H., Siddiqi, I. e Djeddi, C. (2015). Segmentation-verification based on fuzzy integral for connected handwritten digit recognition. Em *2015 International Conference on Image Processing Theory, Tools and Applications, IPTA 2015, Orleans, France, November 10-13, 2015*, páginas 588–591.

Girshick, R. (2015). Fast r-cnn. Em *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, páginas 1440–1448, Washington, DC, USA. IEEE Computer Society.

Girshick, R., Donahue, J., Darrell, T. e Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 580–587.

Grother, P. J. (2016). *NIST Special Database 19 - Handprinted forms and characters database*. NIST.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. e Shen, T. (2017). Recent advances in convolutional neural networks. *Pattern Recognition*.

Ha, T. M. e Bunke, H. (1997). Off-line, handwritten numeral recognition by perturbation method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):535–539.

Ha, T. M., Zimmermann, M. e Bunke, H. (1998). Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. *Pattern Recognition*, 31(3):257 – 272.

Hafemann, L. G., Sabourin, R. e Oliveira, L. S. (2017). Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recogn.*, 70(C):163–176.

Han, J., Zhang, D., Cheng, G., Liu, N. e Xu, D. (2018). Advanced deep-learning techniques for salient and category-specific object detection: A survey. *IEEE Signal Processing Magazine*, 35(1):84–100.

He, K., Gkioxari, G., Dollár, P. e Girshick, R. (2017). Mask r-cnn. Em *Computer Vision (ICCV), 2017 IEEE International Conference on*, páginas 2980–2988. IEEE.

He, K., Zhang, X., Ren, S. e Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.

He, K., Zhang, X., Ren, S. e Sun, J. (2016). Deep residual learning for image recognition. Em *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 770–778.

Hochreiter, S. e Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hochuli, A. G., Oliveira, L. S., Britto, A. S. e Sabourin, R. (2018a). Handwritten digit segmentation: Is it still necessary? *Pattern Recognition*, 78:1 – 11.

Hochuli, A. G., Oliveira, L. S., d. Souza Britto, A. e Sabourin, R. (2018b). Segmentation-free approaches for handwritten numeral string recognition. Em *2018 International Joint Conference on Neural Networks (IJCNN)*, páginas 1–8.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. e Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Kim, K. K., Kim, J. H. e Suen, C. Y. (2002). Segmentation-based recognition of handwritten touching pairs of digits using structural features. *Pattern Recognition Letters*, 23(1–3):13 – 24.

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

Kulkarni, R. V. e Vasambekar, P. N. (2010). An overview of segmentation techniques for handwritten connected digits. Em *Signal and Image Processing (ICSIP), 2010 International Conference on*, páginas 479–482.

Lacerda, E. B. e Mello, C. A. (2013). Segmentation of connected handwritten digits using self-organizing maps. *Expert systems with applications*, 40(15):5867–5877.

Lampert, C. H., Blaschko, M. B. e Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. Em *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, páginas 1–8. IEEE.

Laroca, R., Barroso, V., Diniz, M. A., Gonçalves, G. R., Schwartz, W. R. e Menotti, D. (2019). Convolutional neural networks for automatic meter reading. *Journal of Electronic Imaging*, 28:1–14.

Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R. e Menotti, D. (2018). A robust real-time automatic license plate recognition based on the yolo detector. Em *2018 International Joint Conference on Neural Networks (IJCNN)*, páginas 1–10.

Lecun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *Nature*, 521:434 – 444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. e Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Lecun, Y., Bottou, L., Bengio, Y. e Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Kavukvuoglu, K. e Farabet, C. (2010). Convolutional networks and applications in vision. Em *Proc. International Symposium on Circuits and Systems (ISCAS'10)*. IEEE.

Lee, S.-W. e Kim, S.-Y. (1999). Integrated segmentation and recognition of handwritten numerals with cascade neural network. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 29(2):285–290.

Lei, Y., Liu, C. S., Ding, X. Q. e Fu, Q. (2004). A recognition based system for segmentation of touching handwritten numeral strings. Em *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, páginas 294–299.

Liao, Z. e Carneiro, G. (2017). A deep convolutional neural network module that promotes competition of multiple-size filters. *Pattern Recognition*, 71:94 – 105.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. e Zitnick, C. L. (2014). Microsoft coco: Common objects in context. Em Fleet, D., Pajdla, T., Schiele, B. e Tuytelaars, T., editores, *Computer Vision – ECCV 2014*, páginas 740–755, Cham. Springer International Publishing.

Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019.

Liu, C.-L., Sako, H. e Fujisawa, H. (2004). Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1395–1407.

Ludwig, D. (1966). The radon transform on euclidean space. *Communications on Pure and Applied Mathematics*, 19(1):49–81.

Luz, E., Moreira, G., Junior, L. A. Z. e Menotti, D. (2017). Deep periocular representation aiming video surveillance. *Pattern Recognition Letters*.

Matan, O., Burges, C. J. C., LeCun, Y. e Denker, J. S. (1992). Multi-digit recognition using a space displacement neural network. Em Moody, J. M., Hanson, S. J. e Lippman, R. P., editores, *Neural Information Processing Systems (NIPS 1991)*, volume 4. Morgan Kaufmann Publishers, San Mateo, CA.

Mishra, A., Alahari, K. e Jawahar, C. V. (2012). Scene text recognition using higher order language priors. Em *BMVC*.

Oliveira, L. S., Britto, A. S. J. e Sabourin, R. (2005). A synthetic database to assess segmentation algorithms. Em *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ICDAR '05, páginas 207–211, Washington, DC, USA. IEEE Computer Society.

Oliveira, L. S. e Sabourin, R. (2004). Support vector machines for handwritten numerical string recognition. Em *Ninth International Workshop on Frontiers in Handwriting Recognition*, páginas 39–44.

Pal, U., Belaıd, A. e Choisy, C. (2003). Touching numeral segmentation using water reservoir concept. *Pattern Recognition Letters*, 24(1):261–272.

Pascanu, R., Mikolov, T. e Bengio, Y. (2013). On the difficulty of training recurrent neural networks. Em *International Conference on Machine Learning*, páginas 1310–1318.

Redmon, J., Divvala, S., Girshick, R. e Farhadi, A. (2016). You only look once: Unified, real-time object detection. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 779–788.

Redmon, J. e Farhadi, A. (2017). Yolo9000: Better, faster, stronger. Em *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, páginas 6517–6525. IEEE.

Ren, S., He, K., Girshick, R. e Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Em *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, páginas 91–99, Cambridge, MA, USA. MIT Press.

Ribas, F. C., Oliveira, L. S., Britto, A. S. e Sabourin, R. (2012). Handwritten digit segmentation: a comparative study. *International Journal on Document Analysis and Recognition (IJDAR)*, 16(2):127–137.

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons from backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16(3):265–278.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, páginas 65–386.

Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Roy, P., Bhunia, A., Das, A., Dey, P. e Pal, U. (2016). HMM-based Indic handwritten word recognition using zone segmentation. *Pattern Recognition*, 60:1057–1075.

Rumelhart, D. E., Hinton, G. E. e Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. e Fei-Fei, L. (2015a). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. e Fei-Fei, L. (2015b). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Saba, T., Sulong, G. e Rehman, A. (2010). A survey on methods and strategies on touched characters segmentation. *International Journal of Research and Reviews in Computer Science*, 1(2):103–114.

Sabour, S., Frosst, N. e Hinton, G. (2017). Dynamic routing between capsules. Em *Advances in Neural Information Processing Systems 30 (NIPS 2017)*.

Sadri, J., Suen, C. Y. e Bui, T. D. (2004). Automatic segmentation of unconstrained handwritten numeral strings. Em *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, páginas 317–322.

Sadri, J., Suen, C. Y. e Bui, T. D. (2007). A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. *Pattern Recognition*, 40(3):898–919.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117.

Sherstinsky, A. (2018). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314.

Shi, B., Bai, X. e Yao, C. (2017). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304.

Shi, Z. e Govindaraju, V. (1997). Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognition*, 30(9):1501–1504.

Suwa, M. e Naoi, S. (2004). Segmentation of handwritten numerals by graph representation. Em *Ninth International Workshop on Frontiers in Handwriting Recognition, IWFHR-9 2004, Kokubunji, Tokyo, Japan, October 26-29, 2004*, páginas 334–339.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. e Rabinovich, A. (2015). Going deeper with convolutions. Em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tamen, Z., Drias, H. e Boughaci, D. (2017). An efficient multiple classifier system for arabic handwritten words recognition. *Pattern Recognition Letters*, 93(123-132).

Toft, P. (1996). The radon transform. theory and implementation.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T. e Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

Vellasques, E., Oliveira, L. S., Britto, A., Koerich, A. L. e Sabourin, R. (2008). Filtering segmentation cuts for digit string recognition. *Pattern Recognition*, 41(10):3044–3053.

Voigtlaender, P., Doetsch, P. e Ney, H. (2016). Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. Em *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, páginas 228–233.

Wang, X., Govindaraju, V. e Srihari, S. N. (2000). Holistic recognition of handwritten character pairs. *Pattern Recognition*, 33(12):1967–1973.

Wu, Y. C., Yin, F. e Liu, C. L. (2014). Evaluation of geometric context models for handwritten numeral string recognition. Em *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, páginas 193–198.

Xiao, X., Jin, L., Y.Yang, Yang, W., Sun, J. e Chang, T. (2017). Building fast and compact convolutional neural networks for offline handwritten Chinese character recognition. *Pattern Recognition*, 72-81.

Y. Wua, F. Y. e Liu, C. L. (2017). Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognition*, 65:251–264.

Zanlorensi, L. A., Luz, E., Laroca, R., Britto Jr., A. S., Oliveira, L. S. e Menotti, D. (2018). The impact of preprocessing on deep representations for iris recognition on unconstrained environments. *CoRR*, abs/1808.10032.

Ziyong, F., Zhaoyang, Y., Shuanping, J. L. H. e Jun, S. (2017). Robust shared feature learning for script and handwritten/machine-printed identification. *Pattern Recognition Letters*, 100(6-13).