Trabalho de Conclusão de Curso

SISTEMA DE DETECÇÃO E RECONHECIMENTO DE NÚMEROS BASEADO EM CASCATA DE CLASSIFICADORES E REDES NEURAIS

Matheus Clemente Bafutto
Orientador: Prof. Dr. Alexandre Ricardo Soares Romariz

Brasília, Julho de 2018

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

Agradecimentos

Eu gostaria de agradecer a Deus por ter me iluminado com conhecimento e sabedoria.

Gostaria de agradecer aos meus pais, Marcos e Fabíola, por todo o suporte prestado ao longo de minha criação, a educação que ofereceram me moldou no homem que sou hoje. Gostaria de agradecer aos meu irmãos, Victor e Marina, e à minha namorada, Tainá, por sempre terem me apoiado e encorajado durante minha jornada de graduação.

Meus agradecimentos ao professor Alexandre Romariz, por ter compartilhado seu conhecimento comigo no inicio de meu curso e por ter me agraciado com a oportunidade de realizar este trabalho sob sua supervisão. Sou grato à Universidade de Brasília e ao Departamento de Engenharia Elétrica pela educação de qualidade que me foi dada.

Um agradecimento especial ao aluno de doutorado Vinícius de Oliveira e Silva e à professora Mylene por suas contribuições a este trabalho de graduação.

Quero agradecer também a todos os amigos que me acompanharam na vida universitária, eles me deram forças para persistir e são indiretamente responsáveis pela conclusão deste trabalho.

Matheus Clemente Bafutto

Abstract

Big advancements have been happening on modern computing hardware within the last decades. Year after year processors and electronic devices have been rising their speed and capabilities, all of this at an exponencial level. This rising on processing power accelerates the execution of complex algorithms from the machine learning field, and this enables their use in new fields of robotics, surveillance and healthcare. Besides all stated about processors, the introduction and wide adoption of mobile devices, or *smartphones*, contributed the decrease in costs and improvements on accessibility to high quality video recording technology. In this scenario, there has been a surge in studies about computer vision and its applications under these conditions. Text detection is a complex problem related to computer vision, and solutions to it could enable interesting applications such as photo optical character recognition on printed documents as well as new ways to automate processes in robotics.

This work presents a system for detecting and recognizing numbers, both written and printed. Number detection was implemented by making use of computer vision techniques like cascade classifiers, image pyramids, the sliding window method along with manual feature generation method “Haar”. Number recognition was done by training two different machine learning classifiers, a multi-layer neural network and a convolutional neural network.

The digit detection system was validated with a group of hand crafted examples. A confusion matrix was extracted from the detector’s performance on the set and its f1 score was calculated, achieving a maximum value of 0.4194. Classification was validated by making use of the MNIST dataset, on which each classifier’s accuracy was registered, reaching a maximum of 97.5% for the multi-layer neural net and 99% for the convolutional model. Validation for the system as a whole was also performed by making use of a hand crafted dataset, to which the system achieved a best accuracy of 90%.

Resumo

Grandes avanços tem sido observados nos processadores modernos nas últimas décadas. Ano após ano a capacidade de processamento de computadores e dispositivos eletrônicos aumenta em ritmo exponencial. Este aumento no poder de processamento acelera a execução de algoritmos de aprendizado de máquina e viabiliza seu uso em novos campos como na segurança, robótica e saúde. Além disso, a introdução e massificação dos chamados dispositivos móveis, ou *smartphones*, contribuíram para o barateamento e acessibilidade à tecnologia fotográfica e de gravação de vídeos com qualidade. Neste contexto, estudos de visão computacional como detecção de texto passam a ser viáveis. Detecção de texto é um problema complexo, porém sua solução permite aplicações interessantes como OCR em documentos impressos e alternativas de automação para a robótica.

Este trabalho apresenta um sistema de detecção e reconhecimento exclusivo para números, tanto escritos como impressos. A implementação da detecção de números foi alcançada por meio da associação das técnicas de visão computacional "cascata de classificadores", "pirâmide de imagens", "método da janela móvel" e "geração manual de características Haar". A classificação dos números detectados foi realizada utilizando-se duas alternativas, um modelo de rede neural multi-camada e um modelo de convolução.

A detecção de números foi validada com um grupo de exemplos criados manualmente. Foram extraídos parâmetros da matriz de confusão do detector e também foi calculada sua nota F1, atingindo o máximo de 0.4194. A classificação foi validada por meio do grupo de exemplos da base de dados MNIST, em que o melhor desempenho foi registrado no valor 97.5% e 99% para o modelo de rede multi-camada e convolucional respectivamente. O sistema completo também foi validado em um grupo de exemplos criado à mão, em que o melhor desempenho teve acurácia de 90%.

Índice

Lista de Imagens

Lista de Tabelas

Lista de siglas

SVM	Máquina de Vetor Suporte
MLP	Perceptron multicamada
CNN	Rede Neural Convolucional
R-CNN	Rede Neural Convolucional baseada em Regiões
RPN	Rede Neural de proposição de regiões
YOLO	Modelo de detecção de observação única
OCR	Reconhecimento Óptico de Caracteres
AM	Aprendizado de Máquina
MNIST	Base Modificada do Instituto Americano de Padrões e Tecnologia
SGD	Descida de Gradiente Estocástica
FP	Falso Positivo
TP	Verdadeiro Positivo
FN	Falso Negativo
TN	Verdadeiro Negativo
ROC	Curva Receptora de Características Operantes
PCA	Análise de Componente Principal
SVHN	Base de Números de Casas do *street view*
CPU	Unidade de Processamento Computacional
GPU	Unidade de Processamento Gráfico

1 Introdução

A última década foi marcada por uma explosão no uso de algoritmos de aprendizado em uma vasta gama de aplicações. Entre os motivos que contribuíram para a massificação do aprendizado de máquina pode-se citar três em especial.

O primeiro é a massificação e aceleração da produção de dados digitais. Smartphones e computadores hoje registram todo tipo de informação. Conforme a Internet das coisas é adotada mais e mais dispositivos e sensores serão conectados à rede disponibilizando os dados que produzirem.

O segundo motivo para o aumento de técnicas de aprendizado se relaciona aos avanços feitos nos processadores ao longo dos anos. A *Lei de Moore*, que afirma que processadores dobram de velocidade a cada dois anos aproximadamente, segue se mostrando verdadeira há quase 50 anos, tendo tido grande impacto na viabilização de algoritmos de aprendizado [22].

Por fim, o último motivo para o aumento no uso do aprendizado de máquina se trata de uma consequência dos dois primeiros. A abundância de dados digitais disponíveis para alimentar algoritmos e a melhora nos processadores despertou o interesse um tanto adormecido até então na comunidade acadêmica pelo aprendizado de máquina. A viabilidade do uso destas técnicas levou pesquisadores a desenvolver novos métodos e arquiteturas para estes algoritmos, levando-os a aprender ainda mais rápido e com menos dados (a exemplo da transferência de aprendizado).

Uma das aplicações em destaque que faz uso dos algoritmos de aprendizado é a detecção de objetos. Detecção de objetos é um problema fundamental do campo da visão computacional, e consiste na produção de abstrações a partir em uma única imagem ou uma sequência delas. Estas abstrações normalmente são a identificação e delimitação de um objeto de interesse em alguma área da imagem bem como a designação de qual possível categoria melhor descreve o objeto (carro, árvore, pessoa). A detecção de objetos tem potencial para transformar indústrias, com suas possíveis aplicações na área de vigilância, segurança, direção autônoma de veículos e robótica.

Normalmente, a pesquisa realizada para detecção de objetos é feita com modelos de classificação de uma única imagem, já que não há componente temporal relevante como da detecção de gestos ou ações. Desse modo, a detecção em vídeos ocorre classificando-se individual e sequencialmente cada quadro a medida que estes são produzidos.

1.1 Proposta e Objetivos do Trabalho

Detecção e reconhecimento de texto a partir de uma imagem constitui um problema complexo. Sua solução, no entanto, permite surgimento de uma gama de aplicações interessantes como a realização de reconhecimento ótico de caracteres (OCR) em documentos impressos e, até mesmo, a melhoria de sistemas de navegação para automóveis [7]. Neste trabalho, objetiva-se a criação e otimização de um sistema de detecção de objetos para reconhecimento de números, tanto manuscritos como impressos. O sistema apresenta desde a busca por áreas de interesse (possível presença de números) até a tomada de decisão se há números ou não e, caso positivo, que número seria.

A abordagem escolhida para este intuito é uma associação de técnicas de visão computacional e de algoritmos de aprendizado. Para detecção, faz-se uso de abordagens de reconhecimento clássicas, como pirâmide de imagens, classificadores em cascata e geração manual de características de interesse. Para classificação, são testados dois modelos de

aprendizado distintos, a rede neural artificial multicamada e a rede de convolução. SVM's (do inglês *suporte vector machines*) e redes multicamada, ou MLP's, já são usados rotineiramente em conjunto com métodos de detecção para desenvolvimento de sistemas deste tipo. Já as redes de convolução, CNN, tiveram seu potencial exposto mais recentemente nos últimos 5 anos. A diferença entre ambas é a presença de camadas de convolução e de agrupamento na CNN antes de sua seção integralmente conectada como na SVM e o MLP.

1.2 Trabalhos Relacionados

Nesta seção, trataremos das diferentes técnicas que têm sido empregadas na literatura com o mesmo intuito da abordagem escolhida na concepção do projeto neste trabalho. A variedade de abordagens diferentes demonstra a relevância e o potencial que uma solução eficaz para detecção de objetos poderia ter no futuro.

Existem múltiplas abordagens para atingir a detecção de objetos como a que este trabalho propõe. Normalmente este tipo de aplicação exige que sejam analisadas sub-áreas da imagem inicial em múltiplas dimensões variando razão entre lados e também sua escala. A análise de todas estas possíveis regiões pode trazer muita complexidade computacional para serem classificadas por um modelo. A abordagem do sistema deste trabalho segue a linha proposta por Viola e Jones em seu trabalho [20], onde para realizar esta classificação de maneira rápida utiliza de classificadores lineares simples.

O uso de classificadores lineares simples parece ser a solução mais adequada quando se tem de classificar todas as regiões possíveis em uma imagem (método de "força bruta"). Sua simplicidade reduz a complexidade computacional por região analisada, o que economiza tempo considerável por imagem. A literatura traz, no entanto, outras abordagens classificadas por redes mais complexas viabilizadas por heurísticas de seleção de regiões, ou proposição de regiões. Estas heurísticas permitem selecionar do imenso espaço de possíveis regiões as de maior potencial para presença de objetos, simplificando assim o processamento e permitindo o uso de classificadores mais complexos como as redes de convolução [27].

Métodos independentes de proposição de regiões como [28], quando concatenados a modelos de redes de convolução especiais com capacidade classificadora e de localização são chamadas de redes convolucionais baseada em regiões, ou R-CNN. Projetos que utilizam R-CNN tem esta abordagem genérica, permitindo flexibilidade na escolha de qual rede usar bem como do método de proposição de regiões.

R-CNN's são um modelo de detecção baseado no aprendizado profundo em associação com seleção de regiões. Há na literatura um melhoramento desta abordagem chamado R-CNN rápido (*fast r-cnn*), no qual há um aumento de velocidade pela imagem de entrada passar primeiro pelas camadas convolucionais e depois ter regiões de interesse extraídas. Este processo acelerou muito o tempo de processamento por imagem comparado a seu antecessor, reduzindo em várias vezes o tempo de treinamento e teste [23]. As melhorias na rede de convolução neste método foram tão significativas que o fator limitante para o modelo passaram a ser os próprios métodos de proposição de regiões. Neste contexto surgiu uma nova abordagem, o R-CNN mais rápido (*faster r-cnn*).

O R-CNN mais rápido inovou com relação a seus antecessores no que se refere ao método de seleção de regiões. Enquanto ambos modelos anteriores de R-CNN fazem uso de um método independente da rede para a seleção de regiões, R-CNN mais rápido introduz um modelo convolucional especializado em proposição de regiões, também chamado RPN (do

inglês *region proposal network*) [24]. A incorporação da RPN no fluxo do sistema de detecção acelerou a detecção profunda para além da velocidade da R-CNN rápida.

Uma última abordagem que vem sendo pesquisada para detecção de objetos relevante para o contexto deste trabalho é o modelo regressivo chamado Yolo, do inglês *you only look once*. Diferentemente de todos os sistemas r-cnn e até da própria abordagem de força bruta utilizada no projeto, yolo divide a imagem de entrada em várias sub-regiões. Para cada sub-região, yolo produz uma nota de confiança para presença de cada classe juntamente com uma caixa delimitadora. A vantagem disso com relação aos outros modelos é a velocidade que esse sistema apresenta, sendo mais rápido inclusive que o R-CNN mais rápido [25]. O ganho de velocidade vem acompanhado, porém, de uma piora no desempenho.

1.3 Organização do Relatório

Este relatório de trabalho de graduação está dividido em cinco seções: "Introdução", “Aprendizado de máquina”, “Detecção de Objetos”, “Resultados e Análise” e “Conclusão”. A introdução deste documento consiste neste mesmo capítulo, e tem por objetivo estabelecer o contexto do trabalho e também expor os diferentes métodos presentes na literatura como uma motivação sobre a importância da detecção de objetos.

Os próximos dois capítulos deste relatório, “Aprendizado de máquina” e “Detecção de Objetos”, tem como principal objetivo revisar ou elucidar os conceitos usados durante o desenvolvimento do projeto. Estes capítulos devem estabelecer uma fundamentação teórica para melhor compreensão dos procedimentos realizados no capítulo “Resultados e Análise”. O capítulo 2 apresenta a teoria de redes neurais clássicas, cobrindo também aspectos de validação e otimização de modelos de rede. Por fim, termina elucidando as teorias de aprendizado profundo em foco neste trabalho, as redes neurais de convolução e suas estruturas. O capítulo 3 mostra as técnicas de detecção de objetos envolvidas na implementação específica deste trabalho, cobrindo métodos da pirâmide de imagens, janela móvel e cascata de classificadores.

O capítulo 4, Resultados e Análise, é o capítulo que discute as decisões tomadas durante os procedimentos experimentais do projeto, mostrando os resultados intermediários e finais do projeto. Aqui será possível ter acesso às diversas escolhas envolvidas em cada etapa do projeto bem como a justificativas que levaram a cada tomada de decisão.

Finalmente, o capítulo da conclusão fecha a discussão sobre detecção de objetos e o sistema de detecção de números desenvolvido. Ao longo da conclusão, será feita uma discussão sobre o desempenho que o sistema conseguiu atingir e que decisões poderiam ser consideradas como melhorias futuras.

2 Aprendizado de máquina

O conjunto deste capítulo e o capítulo 3 de Detecção de Objetos têm como objetivo a elaboração e fundamentação de conceitos para as técnicas tratadas durante o desenvolvimento do protótipo de sistema de OCR discutido no capítulo 4. Neste capítulo serão discutidos o funcionamento e as particularidades de modelos do aprendizado de máquina, tais como redes neurais multicamada e modelos convolucionais do aprendizado profundo. Ainda com respeito ao aprendizado de máquina, serão apresentadas técnicas da álgebra linear e estatística que trabalham bem, em conjunto com redes neurais.

No passado, todos os problemas eram resolvidos por meio da escrita de algoritmos imperativos, com instruções claras dadas ao computador ou grupo de computadores. Esta abordagem, no entanto, se mostrou ineficaz na solução de alguns problemas aparentemente simples de resolver para um atuador humano mas extremamente complexas para serem plenamente sanadas com um algoritmo explícito.

Houve tentativas de se resolver problemas deste tipo com regras, lógicas de programação e até com o auxílio de especialistas humanos. Porém as técnicas de aprendizado de máquina se mostraram um meio bastante consistente de se obter boas soluções para estes problemas [9,10]. O aprendizado de máquina, ou AM, é um ramo da área da inteligência artificial conhecido por ser capaz de gerar de maneira automática funções, ou hipóteses, que geram o comportamento desejado para o sistema. Algoritmos de AM são capazes de aprender representações de modelos complexos por meio do treinamento com dados e exemplos relacionados aos problemas. Desse modo, a busca por regras se torna desnecessária pois estas regras são aprendidas pelos próprios algoritmos de maneira automática, que a incorporam na hipótese desejada.

O aprendizado de máquina tem sua teoria com uma base multi disciplinar, contendo contribuições de campos como neurologia, física, probabilidade e estatística entre outros [2]. Exemplos de problemas resolvidos com sucesso hoje por meio de técnicas de AM são a classificação de imagens, direção automatizada de automóveis, detecção de fraudes de cartão de créditos e outras análises preditivas como diagnóstico de câncer por análises de expressão genética [1].

2.1 Classificação dos algoritmos de AM

O aprendizado de máquina como campo também tem suas próprias sub-divisões. As categorias são separadas conforme a abordagem de aprendizado escolhida, e estas são *aprendizado supervisionado*, *aprendizado não supervisionado* e *aprendizado por reforço*.

2.1.1 Aprendizado supervisionado: Como o próprio nome já sugere, este aprendizado tem seu nome por conta da presença de um "supervisor" que fornece as respostas ao algoritmo durante o treinamento do modelo. Entre as atividades mais populares neste tipo de aprendizado podemos citar problemas predominantemente preditivos de regressão e classificação [1]. Modelos de redes neurais normalmente são usados neste tipo de treinamento por conta da conveniência que uma saída explícita apresenta para aprendizado nestas estruturas. Uma comparação de desempenho entre diferentes métodos de aprendizado supervisionado foi realizada por R. Caruana e A. Niculescu-Mizil da Universidade de Cornell [31], e pode fornecer uma visão geral das técnicas populares deste campo.

2.1.2 Aprendizado não-supervisionado: Nesta categoria não existe a presença de um "supervisor" informando ao algoritmo sobre a saída do problema e, consequentemente, não faz sentido em se falar de saídas certas ou erradas. Algoritmos que se enquadram neste tipo de aprendizado têm aplicações em atividades de agrupamento, problemas de associação e sumarização [1]. Exemplos de agrupamento são a divisão de um total de objetos em grupos menores como segmentos de clientes em uma loja. Em seu artigo, Quoc prova que é possível treinar um detector de faces a partir de um grupo de imagens não classificadas [32] através do uso de uma técnica de aprendizado não supervisionado.

2.1.3 Aprendizado por reforço: Durante o aprendizado de técnicas que se enquadram nesta categoria, não há respostas fornecidas por um supervisor. No entanto, o algoritmo recebe recompensas pelas ações que realiza no meio onde se encontra, ao estilo de uma "nota" por sua ação. Dado que o algoritmo visa maximizar a recompensa ou minimizar a penalização, percebe-se que existem elementos supervisionados e não supervisionados. Entre as aplicações do aprendizado por reforço pode-se citar o aprendizado de movimentos na robótica ou aplicações em jogos. Volodymyr desenvolveu um algoritmo de *Q-learning* baseado em modelos profundos que foi capaz de aprender a jogar 49 jogos distintos em nível de um jogador humano profissional [33]. Esta capacidade e flexibilidade no uso de algoritmos de reforço torna este um campo de aplicações com potencial.

Embora o aprendizado de máquina contemple estes três tipos de aprendizado, este relatório terá como foco o aprendizado supervisionado e suas contribuições para o objetivo final deste projeto, o reconhecimento ótico de dígitos. Mais especificamente, a teoria de redes neurais.

2.2 Redes Neurais

Redes neurais, por conta da universalidade de aplicações que aceitam sua utilização, são hoje as estruturas dominantes do aprendizado supervisionado. A maneira como se constrói uma rede neural influencia diretamente em seu desempenho final no grupo de exemplos de validação após concluída a etapa de treinamento. Entre arquitetura de ligações das unidades que a compõem, número de unidades e de camadas, valor da regularização e abordagem para aprendizagem, são muitas as escolhas que se deve fazer ao implementar uma rede neural. A estas escolhas que devem ser feitas pelo projetista do sistema, dá-se o nome hiper-parâmetros.

2.2.1 Uma breve Introdução histórica

Já há muito tempo o funcionamento da mente humana é objeto de fascínio de pesquisadores da área da medicina [4, 5]. No entanto, uma publicação de 1943 com autoria dos pesquisadores McCulloch e Pitts [6] tornou a neurologia um objeto de estudo interdisciplinar ao proporem representações simplificadas para neurônios fazendo uso de lógica matemática. O artigo publicado por eles teve um impacto na comunidade acadêmica da época, e iniciou uma discussão sobre modelos aproximados dos neurônios até chegar ao seu estado atual.

2.2.2 Neurônio como unidade básica

Se hoje existem redes neurais complexas, elas se devem à criação da abstração matemática do neurônio biológico conhecida como a unidade básica, cuja organização e interligação formam os grandes sistemas neurais que observamos hoje atuando do processamento de imagens à tradução de textos. A unidade básica pode ser melhor compreendida fazendo-se uma analogia desta simplificação com seu análogo biológico. Um neurônio do sistema

nervoso é uma célula especializada para realização de sinais de comunicação com seus semelhantes, seja no cérebro ou no sistema nervoso periférico.

Como é possível perceber por meio da ilustração na Figura 2.1, a célula tem várias estruturas necessárias para a realização de sinapses. Sinapses são pulsos elétricos produzidos pela célula e usados na comunicação desta com outros neurônios. Dentre estas estruturas, destacam-se em especial os dendritos de entrada, que ligam sinais de entrada à unidade central de processamento da célula (corpo celular) e por fim, a cauda alongada na região inferior da célula, chamada de axônio. A maneira simplificada desta célula se comportar pode ser vista pelo processo descrito abaixo:

- O neurônio em seu estado inicial percebe sinais de outros neurônios em seus dendritos de entrada, que começam a excitá-lo.
- O neurônio excitado recebe os sinais elétricos de suas entradas e então as processa no interior de seu corpo celular.
- Após processamento das entradas, o neurônio retransmite o sinal de saída por meio do axônio para seus dendritos de saída, propagando o sinal para as unidades neurais conectadas a ele.

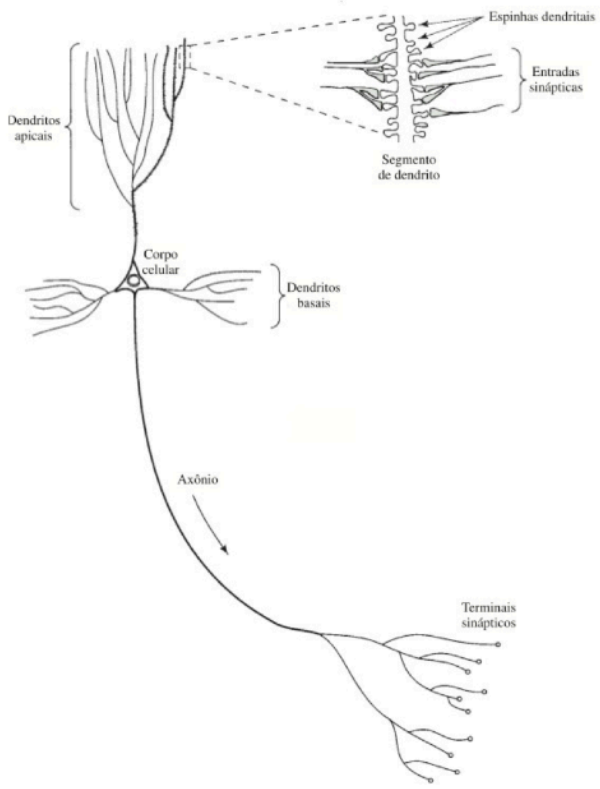


Figura 2.1 - Ilustração da estrutura física de um neurônio segundo Haykin [3]

O modelo matemático que emula o comportamento do neurônio original possui estruturas análogas às existentes no exemplo anterior. Os sinais de entrada conectados ao neurônio entram em uma unidade de processamento que calcula uma soma das entradas multiplicadas por pesos individuais acrescentado de um viés. O resultado é passado para uma função de

ativação ϕ. A saída da unidade corresponde ao resultado da função de ativação sobre a soma ponderada das entradas.

$$v_k = \sum_{i=1}^{m} w_{ki} x_i + b_k \tag{2.1}$$

$$y_k = \phi(v_k) \tag{2.2}$$

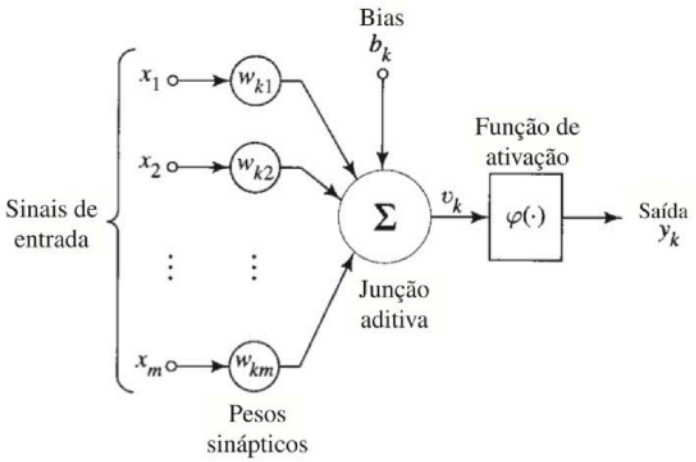


Figura 2.2 - Abstração matemática de neurônio segundo Haykin [3]

A função de ativação é o primeiro hiper-parâmetro que será discutido. Sua escolha influencia diretamente a saída da unidade neural, e existem várias opções de funções de ativação tanto lineares quanto não lineares. A escolha da função de ativação correta por vezes depende da forma do resultado desejado, e se feita de forma consciente pode inclusive acelerar o processamento das entradas bem como acelerar o treinamento do modelo.

2.2.3 Funções de Ativação

Como já mencionado, a escolha da função de ativação dita o comportamento do neurônio como unidade de processamento. Alguns tipos básicos de função de ativação são a função constante como um exemplo linear, e as funções limiar, logística e tangente hiperbólica como exemplos não lineares [3]. O uso de ativações não lineares é característica do modelo de Perceptron original concebido por Rosenblatt em 1958 [11].

A função de ativação constante mantém a soma ponderada inalterada. Este tipo de ativação tem utilidade para aplicações de regressão, onde o resultado da saída está em uma faixa contínua de valores e não restrito a um conjunto discreto de classes. Um exemplo de uso da ativação constante pode ser a análise preditiva do preço de venda de uma casa dadas informações sobre o imóvel (tamanho, endereço, número de suítes, etc) [8].

Quando nos referimos às ativações não lineares como tangente hiperbólica, logística e as outras citadas no início desta seção, normalmente estas são boas escolhas para problemas de classificação. Isto se dá pelo fato de que elas possuem um contra-domínio contido entre o intervalo (0,1), e a natureza similar a uma probabilidade é ideal para inferir se uma entrada pertence ou não a uma determinada classe.

Seguem as definições das funções não lineares citadas:

Logística: $$\phi(v) = \sigma(v) = \frac{1}{1 + e^{-v}} \tag{2.3}$$

Tangente hiperbólica: $$\phi(v) = tanh(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}} \tag{2.4}$$

Função limiar: $$\phi(v) = \begin{cases} 0 & v < 0 \\ 1 & v \geq 0 \end{cases} \tag{2.5}$$

Embora a função tangente hiperbólica tenha seu contra domínio definido no intervalo (-1,1) ao invés do usual (0,1) das demais ativações de classificação, existem maneiras de ajustá-la para esta aplicação. A tangente hiperbólica também é útil para neurônios intermediários em uma rede neural, onde o resultado não corresponde à saída definitiva do sistema. Estas questões serão tratadas mais a frente nos modelos de Perceptron multicamada.

Uma última função de ativação cujo uso foi popularizado no início da massificação do uso das redes neurais convolucionais é a função linear retificada, ou Relu. A ativação Relu foi testada e popularizada por conta do trabalho de reconhecimento de padrões de Krizhevsky [12], e se mostrou tanto vantajosa para discriminação de classes em imagens [13] como mais simples de processar, acelerando a propagação de entradas pelas redes neurais. A ativação Relu é definida pela seguinte equação:

$$\phi(v) = \begin{cases} 0 & v < 0 \\ v & v \geq 0 \end{cases} \tag{2.6}$$

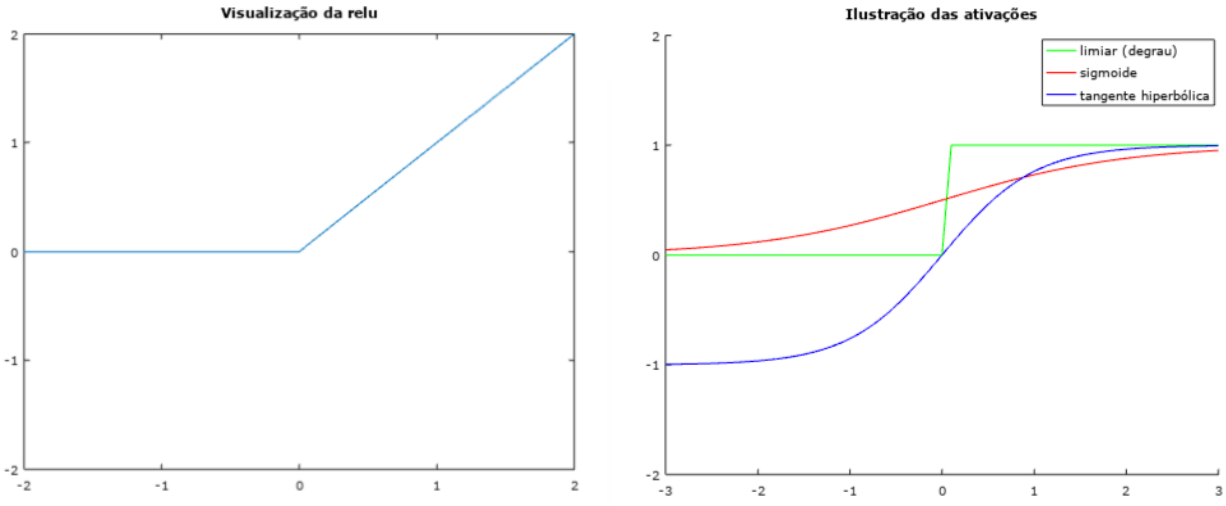


Figura 2.3 - Visualização das ativações Relu (a esq.) e demais não lineares (a dir.)

A maior parte destas ativações são utilizadas hoje, a função degrau foi a primeira a ser teorizada historicamente porém caiu em desuso por conta da popularização da descida de gradiente como algoritmo de treinamento em redes neurais. Este algoritmo requer que as funções pelo menos tenham alguma continuidade (caso da Relu).

2.2.4 Rede Neural de Camada Única

A unidade neural pode ser usada tanto para regressão quanto para problemas de classificação. Na classificação, percebe-se que a saída de uma única unidade é um valor numérico entre 0 e 1 se assemelhando a uma variável booleana. Este modelo é útil para classificações binárias, mas é limitado se o problema exigir múltiplas classificações. Felizmente é muito simples estender o uso destas unidades para problemas de várias classes agregando múltiplas unidades em uma única "camada". Desse modo, cada neurônio é designado para a classificação de uma única classe.

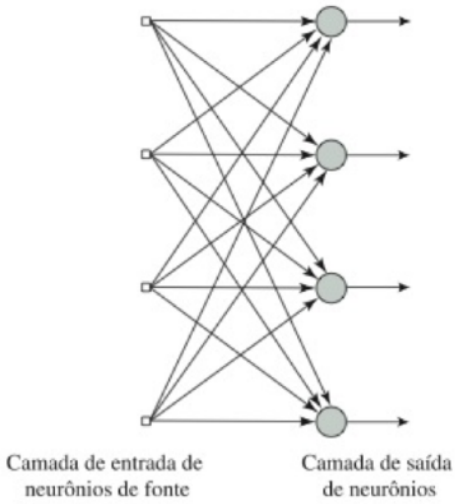


Figura 2.4 - Modelo de perceptron de camada única segundo Haykin [3]

2.2.5 Redes Neurais Multicamada

Redes neurais de camada única são capazes de aprender características e discriminar diferentes grupos de exemplos usados em seu treinamento. No entanto, a presença de uma camada única na rede neural permite apenas a separação de grupos de exemplos por um único hiper-plano. Não sendo incomum grupos de exemplos que não são bem separáveis por meio de um hiper-plano, é necessário que haja uma maneira de discriminar estes exemplos por meio de separações não lineares.

Uma rede neural é capaz de formar estas separações não lineares na arquitetura de múltiplas camadas [3]. Uma rede neural multi-camada segue um padrão similar ao padrão de ligações das redes de camada única, em que a saída contém uma quantidade de unidades correspondendo ao número de categorias possíveis da aplicação. Neste caso porém, entre entrada e saída pode haver uma ou mais camadas escondidas. O número de camadas e a quantidade de unidades por camada são hiper-parâmetros da rede.

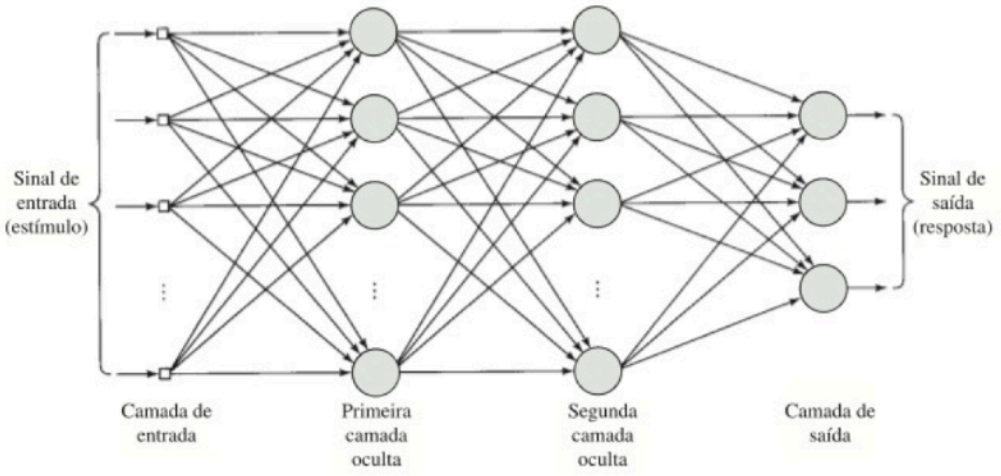


Figura 2.5 - Exemplo de rede neural multi-camada segundo Haykin [3]

A figura 2.5 ilustra a arquitetura usual de ligações entre as unidades e camadas de um modelo neural multi-camada. Existe espaço para uso de mais de uma ativação na rede, no entanto, é usual que uma mesma ativação se repita para todas as unidades pertencentes a uma mesma camada. Modelos de redes multi-camada têm capacidade de aprender soluções para uma universalidade de problemas, tendo sido extensamente testados por meio de estudos acadêmicos para reconhecimento de padrões como o clássico problema do MNIST [14].

O treinamento das redes multi-camada é realizado com o auxílio de um algoritmo de aprendizado chamado retro-propagação de erro. Este é um algoritmo que estende a técnica de aprendizado minimizando o custo total de uma rede em um grupo de treinamento de maneira a atualizar todos os pesos da rede, tanto os de entrada e saída como os das camadas escondidas. A retro-propagação do erro será tratada na seção 2.2.6.

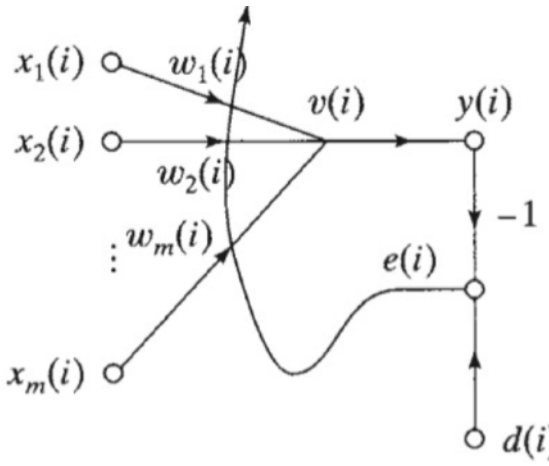


Figura 2.6 - Diagrama do fluxo de aprendizado com um exemplo segundo Haykin [3]

2.2.6 Análise de desempenho de sistemas neurais em um grupo de treinamento

Os valores dos pesos e da parcela de viés na soma ponderada das unidades de uma rede ditam como estas unidades reagem a sinais em suas entradas. Neste contexto, para uma dada rede neural, é necessário que haja valores ou métodos que quantifiquem o quão bem ajustados estão os pesos da rede à aplicação desejada. Isto é feito por meio de funções capazes de produzir uma medida de desempenho numérica. Estas funções são o custo e o erro.

A função custo é definida pela equação 2.7 abaixo:

$$C(Y, D) = \frac{1}{m} \sum_{i=t}^{m} e(Y_i, D_i) \tag{2.7}$$

- m - número de exemplos considerados a partir do valor t
- t - exemplo inicial do custo, varia conforme a técnica de otimização
- Y - matriz das saídas n-dimensionais da rede para os m exemplos considerados
- D - Matriz com os valores esperados para cada elemento de Y

O custo é normalmente usado como o parâmetro de medição do desempenho da rede neural na aplicação em que esta está inserida, sendo assim, uma quantidade fundamental para os algoritmos de aprendizado.

Os sinais de erro definidos por $e(Y_i, D_i)$ nas parcelas da equação de custo 2.7 podem ter uma gama de formatos.

$$e_{LMS}(Y, D) = \frac{1}{2} \sum_{i}^{n} (D_i - Y_i)^2 \tag{2.8}$$

A forma do sinal de erro da equação 2.8 corresponde ao erro quadrático, um sinal de erro que mantém registro das imperfeições cometidas nos exemplos durante o cálculo da função de custo C(Y, D).

Uma outra alternativa para se definir o sinal de erro é uma forma apresentada na equação (2.9) que pode ser deduzida a partir da estatística, mais especificamente pela estimação por máxima verossimilhança [8].

$$e(Y, D) = \sum_{i} - D_i \log(Y_i) - (1 - D_i) \log(1 - Y_i) \tag{2.9}$$

As equações 2.8 e 2.9 são de uso comum para sinais de erro em algoritmos de aprendizado. Com os sinais de erro devidamente apresentados, pode-se prosseguir para os métodos práticos de minimização do custo.

2.2.7 Aprendizado em redes neurais

Em uma rede neural, existem valores para seus pesos os quais maximizam o desempenho da rede para uma determinada aplicação. Algoritmos de aprendizado tem a finalidade de ajustar estes pesos de maneira automática para maximizar o desempenho em qualquer que seja a tarefa designada ao sistema.

Um algoritmo de aprendizado normalmente inicializará os pesos de forma aleatória. Os ajustes realizados nos pesos requerem um conjunto de exemplos processados como saídas y e valores “desejados” d para cada saída do neurônio. Os ajustes ocorrem por conta de análises do *custo* produzido ao se propagar um grupo de exemplos de treinamento pela rede. Os

algoritmos de aprendizado a serem tratados nesta fundamentação teórica se baseiam no problema de otimização que minimiza a função de custo produzida a cada iteração.

Descida de Gradiente - A Descida de Gradiente é uma das abordagens mais simples de minimização da função de custo de uma rede neural. Seu princípio básico consiste na averiguação do gradiente da função custo contendo todos os exemplos de treinamento. Neste caso, os valores de t e m na equação de custo 2.7 são respectivamente 1 e o tamanho do grupo de treinamento. Após o cálculo do gradiente, dá-se um passo na direção contrária ao gradiente regulado por uma constante η, chamada também de taxa de aprendizado.

$$w_{ij}^{l}(n+1) = w_{ij}^{l}(n) - \eta \frac{\partial C(Y, D)}{\partial w_{ij}^{l}(n)} \tag{2.10}$$

- w_{ij}^{l} - variável peso pertencente à j-ésima característica na i-ésima unidade da camada l
- η - taxa de aprendizado
- n - atual iteração do valor do peso em questão

A maneira mais simples de se tratar a taxa de aprendizado é manter seu valor constante ao longo do treinamento. Há, no entanto, heurísticas que podem ser adotadas ao longo do processo iterativo de atualização dos pesos na rede como o decrescimento exponencial de η ou a adição e momento às iterações.

As heurísticas são úteis porque elas permitem que seja feito o aprendizado sem tanto rigor na seleção do valor inicial da taxa de aprendizagem. Para valores fixos, deve-se tomar cuidado especial nesta seleção, visto que o sistema é retro-alimentado e pode não convergir para valores de η muito grandes. Neste contexto, o valor de η se torna um hiper-parâmetro, e deve idealmente ser escolhido com um valor pequeno o suficiente para garantir a convergência do aprendizado. Porém sendo o maior possível neste intervalo de convergência, visto que quanto menor o η, mais lento é o aprendizado.

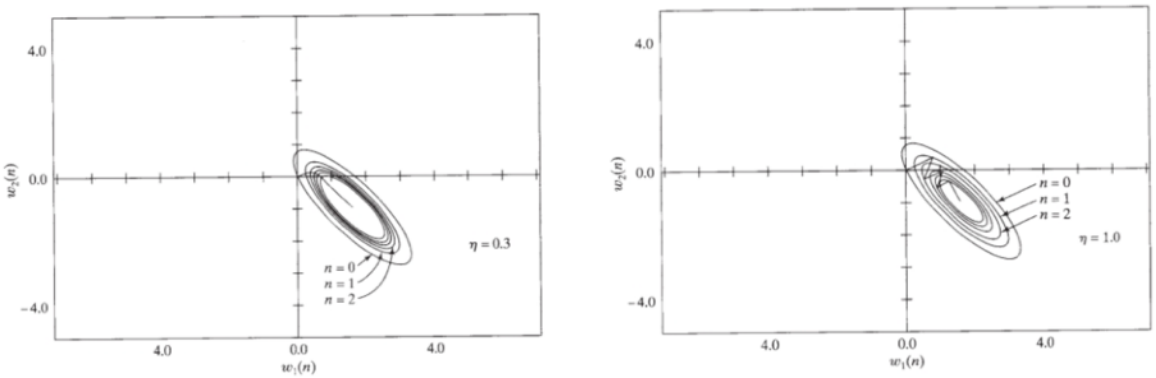


Figura 2.7 - Visualização da decida de gradiente em somente dois pesos para η sendo 0.3 (a esq.) e 1 (a dir.) [3]

Observe na figura 2.7 como a trajetória fica mais instável à medida que se aumenta a magnitude da taxa de aprendizagem. Os valores ótimos para η que garantam a convergência variam de aplicação em aplicação. Normalmente a escolha é feita observando a curva de aprendizagem do modelo para vários valores em intervalos de ordem de grandeza.

A descida de gradiente pode fazer uso de técnicas de vetorização, trocando partes dos passos iterativos por operações matriciais. Isto acelera os processos de treinamento e propagação de entradas pela rede. No entanto, o uso do conjunto inteiro de exemplos de treinamento em cada iteração de descida de gradiente pode, ainda que com vetorização, tornar-se inviável conforme o tamanho do grupo de treinamento aumenta. Outras formas de se minimizar o custo levando em conta o mesmo princípio da descida de gradiente existem, no entanto estas outras maneiras são capazes de lidar melhor com grandes volumes de exemplos no treinamento.

Descida de Gradiente Estocástica (SGD) - Variação da descida de gradiente original, a decida de gradiente estocástica reduz a função de custo de cada iteração do aprendizado a um sinal único de erro de um exemplo. Neste caso, o custo ilustrado na equação 2.7 adquire valores de t e m iguais, removendo o somatório da equação. Enquanto a convergência da decida de gradiente original se aproxima gradativamente do mínimo mais próximo da função de custo para todos os exemplos, a convergência para esta abordagem lembra uma caminhada aleatória, por esta razão, o nome estocástica. Por ter seu custo bastante simplificado, a SGD consegue realizar iterações para atualização de pesos de forma mais rápida, sendo mais vantajosa que a descida de gradiente original para grandes grupos de treinamento. Uma desvantagem deste método no entanto, é a redução do uso das técnicas de vetorização usadas para acelerar os cálculos como na descida de gradiente clássica. Existe uma segunda variação da SGD que torna operação vetoriais mais viáveis porém ainda mantendo o benefício para grandes grupos de treinamento.

Descida de Gradiente em *mini-batches* - A última variação de otimização por decida de gradiente a ser discutida neste relatório consiste na descida em *mini-batches*. Esta abordagem de minimização do custo contém elementos tanto da decida de gradiente estocástica como da decida de gradiente original. Como na SGD, a descida em *mini-batches* não utiliza todos os exemplos de treinamento em cada iteração. Porém, cada iteração é feita com uma função de custo abrangendo um sub-grupo, ou *batch*, de exemplos do grupo de treinamento. Desse modo, esta abordagem de aprendizado aumenta a frequência de operações matriciais mantendo a vantagem da SGD para grandes grupos de exemplos de treinamento.

2.2.8 A retro-propagação do erro em redes multi-camada

Como já mencionado, a correção de pesos no treinamento dos neurônios de saída é feita através na análise do gradiente da função custo total da amostra de treinamento caminhando na direção de maior descida como descrito na equação 2.10. Um obstáculo histórico para a viabilidade do treinamento de modelos de rede multi-camada no entanto, é o do problema de atribuição de crédito pela saída do sistema às unidades escondidas e sua consequente atualização de pesos. A retro-propagação do erro da saída para as camadas anteriores surgiu baseando-se no princípio matemático da regra da cadeia, e analisa a contribuição de cada neurônio por meio do gradiente local da unidade.

A equação em análise continua sendo a equação 2.10, porém no caso devemos fazer algumas adaptações para cálculo do termo $\frac{\partial C(Y,D)}{\partial w_{ij}^{l}(n)}$, onde consideramos a função de custo como sendo o sinal de erro instantâneo da camada l. Com base na regra da cadeia, a derivada parcial do custo C em relação ao peso w_{ij}^{l} fica na forma:

$$\frac{\partial C}{\partial w_{ij}^{l}} = \frac{\partial C}{\partial p_i}\frac{\partial p_i}{\partial y_i}\frac{\partial y_i}{\partial v_i}\frac{\partial v_i}{\partial w_{ij}^{l}} \tag{2.11}$$

- C - função de custo definida pelo erro instantâneo na camada do neurônio em questão
- p_i - parcela de sinal de erro na soma C correspondente ao neurônio em questão
- y_i - saída da ativação na unidade conforme ilustrada na equação 2.2
- v_i - soma ponderada das entradas do neurônio segundo a equação 2.1
- w_{ij}^{l} - peso de uma característica do neurônio em questão

Como sabemos que C é uma soma de sinais de erro, e se o erro for quadrático, temos os seguintes resultados:

$$\frac{\partial C}{\partial p_i} = p_i \tag{2.12}$$

$$\frac{\partial p_i}{\partial y_i} = -1 \tag{2.13}$$

$$\frac{\partial y_i}{\partial v_i} = \phi'(v_i) \tag{2.14}$$

$$\frac{\partial v_i}{\partial w_{ij}^{l}} = x_j \tag{2.15}$$

Por fim, agrupamos alguns destes resultados em um termo de gradiente local δ, definido pela equação 2.15:

$$\delta_i = \frac{\partial C}{\partial v_i} = \frac{\partial C}{\partial p_i}\frac{\partial p_i}{\partial y_i}\frac{\partial y_i}{\partial v_i} = -p_i\phi'(v_i) \tag{2.16}$$

Resultando em um termo de correção $\Delta w_{ij} = \eta\delta_i x_j$ [3]. Substituindo este novo valor na equação 2.10 temos uma maneira de iterar sobre todos os possíveis pesos que uma rede multi-camada pode ter.

2.2.9 Generalização

Um modelo de rede neural uni ou multi camada treinado com um determinado grupo de exemplos de treinamento precisa ter sua generalização comprovada para se ter ideia da qualidade do treinamento feito neste modelo. O método usado para ponderar esta qualidade consiste em submeter a rede plenamente treinada a um grupo de exemplos de teste, os quais a rede não foi exposta durante seu treinamento e observar a precisão da discriminação destes exemplos.

O grupo de teste é essencial para se fazer a avaliação da capacidade de generalização da rede. Mesmo que esta apresente bom desempenho no treinamento, pode ser que a hipótese aprendida tenha variância grande, tornando o desempenho da rede viciado nos exemplos de treinamento. A validação com os exemplos de teste ajudam a expor problemas como este, denominado "overfitting". Uma maneira de se resolver uma situação de *overfitting* pode ser por meio de decréscimo no tamanho da rede ou por meio de regularização. A regularização é um termo extra acrescentado na função de custo minimizada que força os pesos a terem valores pequenos [8].

$$C_{reg}(Y, D) = C(Y, D) + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{j=1}^{s_l} \sum_{i=1}^{s_{l+1}} (w_{ij}^{l})^2 \tag{2.17}$$

A maneira como os pesos são forçados a terem seus valores reduzidos é acrescentando parcelas quadradas de todos os pesos na equação de custo original da equação 2.7. O parâmetro λ é chamado constante de regularização, e também é um hiper-parâmetro importante na implementação de redes neurais.

O extremo oposto do *overfitting* é quando a rede tem uma estrutura simples demais para ser capaz de se ajustar bem aos exemplos de treinamento. O erro de treinamento permanece grande mesmo após muitas iterações de decida de gradiente. Este problema é chamado *underfitting*, e pode ser resolvido aumentando-se a complexidade da rede, acrescentando-se novas características na entrada da rede ou diminuindo a magnitude do parâmetro λ.

2.3 Técnicas para Análise da Qualidade de uma rede Treinada

Medidas clássicas para se analisar a qualidade de um modelo de rede após a conclusão do seu treinamento é a taxa de erro ou a taxa de acerto no grupo de validação. Estas medidas, no entanto, tem limitações. Para casos em que os exemplos tanto de treinamento como de validação estão suficientemente desbalanceados, ou seja, existe um desequilíbrio leve ou significativo entre a quantidade de exemplos de uma classe com relação aos outros, a taxa de erro ou a de acerto podem não ser muito informativas.

Para melhorar o estudo da qualidade do desempenho de um modelo em circunstâncias mais gerais, existem outras medidas que podem ser levadas em consideração. Precisão e *recall* são quantidades inversamente proporcionais que fornecem informações sobre a capacidade de um modelo em produzir "falsos positivos"(FP), "falsos negativos"(FN), "positivos verdadeiros"(TP) e "falsos verdadeiros"(TN). Para compreender estas duas grandezas, é necessário observar a chamada matriz de confusão.

Tabela 2.1 - Matriz de Confusão

	Positivo	**Falso**
Previsão positiva	TP	FP
Previsão negativa	FN	TN

$$recall = r = \frac{TP}{TP + FN} \tag{2.18}$$

$$precisao = p = \frac{TP}{TP + FP} \tag{2.19}$$

A matriz de confusão ilustrada pela tabela 2.1 apresenta de maneira intuitiva a relação entre as previsões corretas e incorretas do modelo independentemente do balanceamento dos dados de treinamento e validação. As definições de TP, TN, FP e FN são a fundação para as grandezas precisão e *recall*, que além de quantidades melhores para analisar a performance de um modelo treinado por si só, também se tornam bons conceitos para construção de teorias de validação mais sofisticadas como as notas F1.

A substituição da taxa de erro ou de acerto pelas medidas de precisão e *recall* melhora a estimativa de qualidade do modelo de modo a incluir circunstâncias em que há desbalanceamento significativo das classes nos dados. Enquanto existe um ganho de visibilidade para qualidade do modelo, há uma perda no sentido de um ganho de complexidade. Uma taxa quantitativa única passa a ser representada por duas grandezas, precisão e recall. Não é tão simples comparar um modelo com outro usando dois parâmetros em vez de uma única taxa de erro por exemplo. A nota F1, nesse contexto, se apresenta como uma das possíveis soluções que combinam a precisão e o recall em um único valor, simplificando a comparação entre dois modelos. A taxa F1 aplica uma relação similar a regra da combinação de resistores em paralelo na engenharia elétrica, de modo que o resultado final é menor que o menor dos dois valores. Isso permite um valor único que incorpore precisão e recall de maneira mais real.

$$F1 = \frac{p\ r}{p + r} \tag{2.20}$$

Uma última técnica de verificação da qualidade de treinamento de um modelo a ser tratada neste relatório consiste nas curvas características de operação do receptor, ou ROC. As curvas ROC são muito utilizadas para realização de uma validação gráfica para um modelo treinado, por vezes sendo comparada a uma curva ROC de modelo aleatório. Como o gráfico na figura 2.8 sugere, uma curva ROC é uma ilustração gráfica da taxa de verdadeiros e falsos positivos de um classificador binário. Da maneira como as curvas ROC são calculadas, a área abaixo da curva representa uma medida única de desempenho como a taxa de erro. As curvas ROC traçam uma relação entre os verdadeiros positivos e os falsos positivos, e a qualidade de um modelo é verificada conforme a área abaixo da curva se aproxima de 1.

$$TPR = \frac{TP}{TP + FN} \tag{2.21}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.22}$$

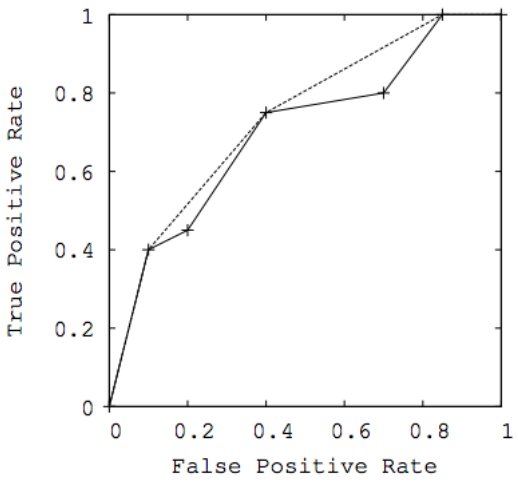


Figura 2.8 - Ilustração de uma curva ROC, figura adaptada de [15]

2.4 Aprendizado profundo

Aprendizado profundo é um sub-campo do aprendizado de máquina tradicional, e por ter ganhado popularidade recentemente, existem várias definições cabíveis para este conceito no momento. Uma boa definição para caracterizar as técnicas presentes neste campo é uma das apresentadas por Deng [16], que explica o aprendizado profundo como sendo "uma classe de técnicas de aprendizado de máquina que se vale do uso de múltiplas camadas não lineares de processamento com intuito de uso supervisionado ou não de extração de características…".

Algoritmos do aprendizado profundo têm alta capacidade de abstração formada a partir do seu treinamento. Isto se dá em parte por conta da introdução de modelos capazes de serem treinados para aprenderem que características da entrada devem extrair e levar em consideração. O aprendizado automático de que características devem ser extraídas resolve um antigo problema do aprendizado de máquina. Uma rede neural multi-camada com duas camadas escondidas por exemplo teria, em tese, capacidade para modelar a vasta maioria de aplicações que surgem para aprendizado de máquina [2]. Muitas destas aplicações exigem um número e uma variedade muito grande de exemplos no treinamento para que uma rede multi-camada seja capaz de generalizar bem. A extração das características principais dos exemplos é usada tanto para acelerar sua propagação pela rede como para auxiliar a rede a generalizar sem necessidade de tanta variedade em seu treinamento. Antes do aprendizado profundo, a extração de características era feita manualmente. Quando os modelos aprendem quais as melhores características devem ser extraídas, esta extração deixa de ser um hiper-parâmetro, e produz resultados muito bons com menos variedade de exemplos de treinamento.

Pelo fato de algoritmos profundos contarem com muitas camadas de processamento, normalmente há um número grande de parâmetros a serem ajustados durante o treinamento. Este ganho de complexidade nos modelos costuma aumentar o processamento necessário para uso e treinamento de técnicas profundas. Este tipo de aprendizado se tornou viável ao longo dos últimos anos por conta do aumento gradual do poder de processamento dos computadores disponíveis. A implementação de algoritmos de paralelismo como *mapReduce* e o uso de unidades de processamento gráfico, ou GPU's, também melhoraram o trabalho com modelos profundos. Graças à massificação da internet e da produção de dados devido à internet das coisas, foi possível a construção de vastos bancos de dados com potencial para se tornarem exemplos de treinamento em diversas aplicações.

Nossa revisão teórica no aprendizado profundo terá foco nas redes neurais famosas por sua eficácia no reconhecimento de padrões em imagens, as redes convolucionais.

2.4.1 Redes Neurais de convolução (CNN)

Redes de convolução se estabeleceram como o estado da arte para classificação de imagens nos últimos anos a partir de trabalhos como a publicação de Krizhevsky na base imageNet [12]. A base imageNet contém imagens com 1000 classes distintas, em orientações e posições diferentes de um modo que técnicas de extração de característica manuais se tornavam muito difíceis. A rede de Krizhevsky, batizada de VGG16, foi capaz de classificar as imagens da base imageNet com um erro de 17%, muito superior a outras técnicas, com erros acima de 25%.

A vantagem que as redes de convolução apresentam em relação às demais técnicas é a presença de camadas de rede dedicadas a extração de características, capazes de aprender a extração ótima a partir do treinamento assim como uma camada comum. As chamadas camadas de convolução transformaram a classificação de imagens como se entendia na época. Cada camada de convolução presente na rede processa características específicas da entrada original, aumentando-se o grau de abstração conforme a entrada se propaga pelas camadas da rede. As camadas de convolução se alternam com camadas compactadoras chamadas camadas de *pooling*, e ao final da arquitetura, todas as saídas da camadas de convolução são ligados a uma parcela integralmente conectada como em uma rede multi-camada. A saída da parcela multicamada corresponde a saída global da rede.

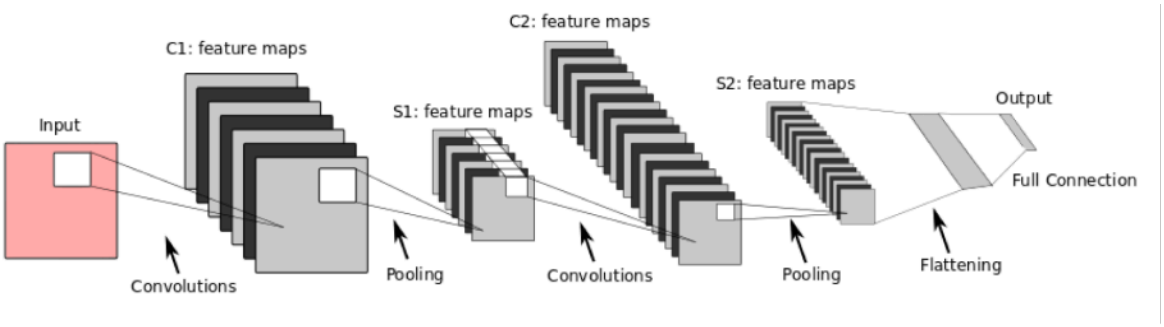


Figura 2.9 - Arquitetura de rede de convolução LeNet segundo Lecun [14]

2.4.2 Camadas de convolução

As camadas de convolução podem ter um ou mais mapas de características. Um mapa de características consiste em um um grupo bidimensional de neurônios. Neurônios de um mesmo grupo compartilham os mesmo pesos, e não se conectam a toda a camada de entrada, mas a uma pequena porção desta. A inserção das entradas nos neurônios são feitas pelo que se chama filtro de convolução, um pequeno quadrado 2D normalmente 3x3 ou 5x5 que é aplicado ao longo da saída da camada anterior. A saída de cada neurônio em um mesmo mapa de característica corresponde ao resultado de uma função de ativação σ recebendo a soma ponderada das entradas multiplicadas pelos pesos somado de um bias em diferentes pontos da entrada. Segue uma formalização matemática para o valor de cada saída individual de um mapa de características [2]:

$$y_{mn}^{l} = \sigma\left(\sum_{i=1}^{Nn}\sum_{j=1}^{Nm} y_{(m+i-1)(n+j-1)}^{l-1} w_{ij}^{l} + b^{l}\right) \tag{2.23}$$

Na equação 2.23, Nn e Nm correspondem às dimensões do filtro de convolução. Uma camada de convolução pode conter múltiplos mapas de características, cada um treinado para

perceber determinados traços dos dados advindos da camada anterior. Embora os neurônios de um único mapa compartilhem os pesos, isto não acontece para mapas distintos. A figura 2.10 ilustra um exemplo de como determinadas características são percebidas por diferentes mapas conforme uma imagem é propagada pela seção de convolução de uma CNN na classificação de imagens.

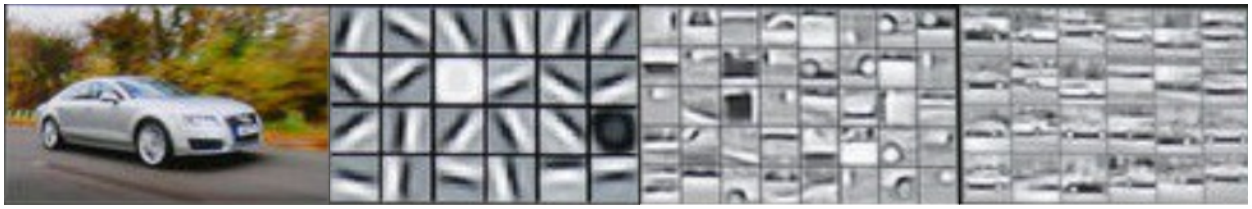


Figura 2.10 - Visualização de diferentes mapas por camada de CNN [17]

A imagem na figura 2.10 [2] esclarece como uma entrada é propagada em uma camada de convolução:

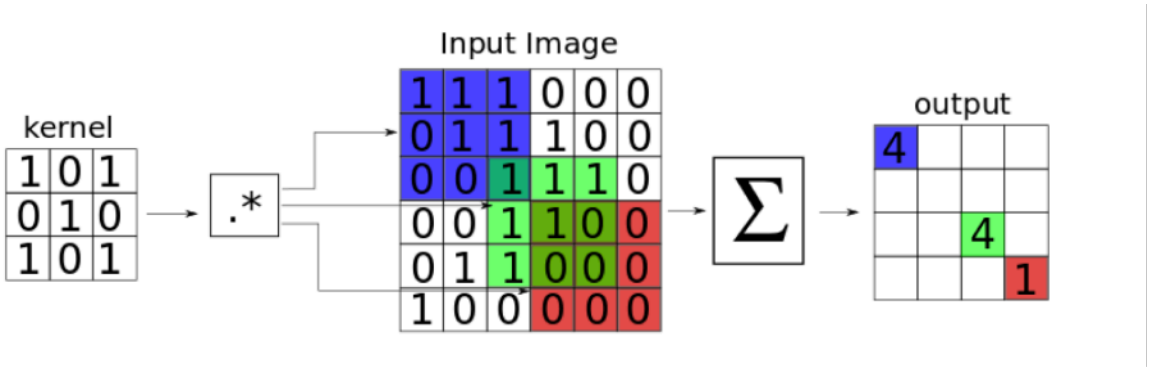


Figura 2.11 - Ilustração da propagação de uma entrada em uma camada convolucional segundo [2]

Tanto o tamanho do filtro usado na convolução como o espaçamento definido entre os filtros para cada neurônio definem a maneira como a camada de convolução vai se comportar. Outro recurso usado para estas convoluções consiste na adição de “margem" de modo a manter o tamanho da camada convolutiva igual à entrada. As três possibilidades tratadas neste parágrafo consistem em hiper parâmetros de escolha do projetista.

2.4.3 Camadas de *pooling*

Segunda e última estrutura a ser tratada no escopo deste relatório são as camadas de *pooling*. O *pooling* foi batizado assim pois esta camada tem a função de agrupar neurônios de uma camada de convolução que o antecede, condensando assim, o tamanho dos mapas de características. Há diversos critérios para decidir o valor que passa para a camada de *pooling*, os dois mais comuns são o *pooling* médio, em que o valor passado é a média das entradas, e o *pooling* máximo, onde o valor passado é a entrada de máximo valor.

Camadas de *pooling*, além de reduzir o tamanho das camadas de convolução também auxiliam na generalização de rede para reconhecimento de padrões, tornando-a menos sensível a variações espaciais e de escala na classificação de imagens por exemplo. Assim como nas camadas convolucionais, as camadas de *pooling* tem o tamanho do filtro de convolução e o espaçamento entre filtros como hiper-parâmetros da camada.

2.5 Análise de Componente Principal (ou PCA)

Análise de componente principal é um algoritmo da álgebra linear que tem muita utilidade nas áreas da ciência que lidam com informação, ou dados. Em resumo, a técnica basicamente identifica em um grupo de m amostras n-dimensionais quais as principais componentes que compõem aquele grupo. Tirando proveito das implicações do PCA, é possível selecionar quantas das componentes principais utilizar em uma transformação do grupo de amostras inicial mantendo controle sobre a variância perdida conforme se removem componentes. O resultado disso implica que a dimensionalidade de cada amostra individual pode decrescer sem muita perda na variância que caracteriza o grupo se as dimensões forem altamente correlacionadas. Em aprendizado de máquinas, o PCA tem um papel importante uma vez que a redução da dimensionalidade das amostras ou exemplos passados para redes neurais pode acelerar a propagação e o treinamento com exemplos dados ao modelo [18].

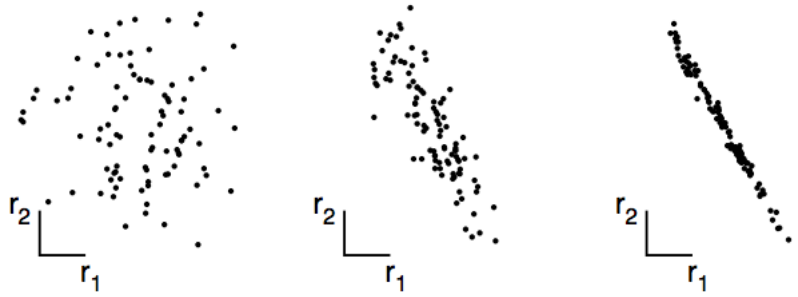


Figura 2.12 - Visualização de exemplos bidimensionais com redundância crescente da esquerda para direita [18]

A redundância em um grupo de amostras pode ser bem visualizada por meio de gráficos de amostras tri ou bi-dimensionais como os da figura 2.12. Conforme fica mais evidente que há uma correlação entre as grandezas r_1 e r_2, mais se percebe que este espaço bidimensional poderia ser bem representado em um espaço unidimensional. No qual uma única reta poderia representar sem grandes erros de aproximação o grupo da amostra mais a direita.

PCA permite uma maneira de generalizar estas simplificações para um número arbitrário de dimensões por amostra. A formulação matemática por trás do algoritmo será tratada a seguir. Uma simples definição da matriz de covariância para uma dada matriz segue na equação 2.24:

$$C_x = \frac{1}{n} X X^T \tag{2.24}$$

O objetivo do PCA como algoritmo é determinar uma matriz ortonormal P tal que $Y = PX$ fazendo com que a matriz de covariância C_y seja uma matriz diagonal. X representa o grupo de amostras inicial. Pela definição de matriz covariância da equação 2.24 aplicada a C_y, temos que:

$$C_y = \frac{1}{n}(PX)(PX)^T \tag{2.25}$$

$$C_y = \frac{1}{n} P X X^T P^T \tag{2.26}$$

$$C_y = P(\frac{1}{n} X X^T) P^T \tag{2.27}$$

$$C_y = PC_xP^T \tag{2.28}$$

Para uma matriz quadrada, vale a relação $A = EDE^T$, em que E é a matriz de autovetores da matriz A e D é uma matriz diagonal. Portanto, se fizermos P = E^T, temos que:

$$C_y = P(EDE^T)P^T \tag{2.39}$$

$$C_y = (PP^T)D(PP^T) \tag{2.30}$$

$$C_y = D \tag{2.31}$$

Portanto, se fizermos P um operador de transformação igual a matriz de autovetores correspondente ao conjunto inicial X, conseguimos visualizar as amostras em suas componentes principais [18].

2.6 Considerações Finais

Redes neurais são ferramentas muito flexíveis, capazes de resolver uma vasta gama de problemas assumindo que haja exemplos suficientes no treinamento e uma arquitetura bem elaborada. O campo do aprendizado profundo foi apresentado como um nicho do aprendizado de máquina mais amplo. Dele, foram discutidas arquiteturas de rede de convolução, com camadas convolutivas e de agrupamento (*pooling*) usadas na automação de extração de características.

O capítulo 3 a seguir consiste na apresentação de conceitos teóricos relacionados ao campo da Detecção de Objetos. Nele, será feita uma discussão sobre as técnicas e métodos para detecção de objetos que tiveram utilidade na implementação do sistema de reconhecimento de números conforme especificado no Capítulo 4, de “Resultados e Análise”.

3 Detecção de Objetos

A associação deste capítulo com o capítulo 2 fornece a totalidade dos conceitos necessários para compreender os procedimentos explícitos no capítulo 4. Neste capítulo serão tratados conceitos da Detecção de Objetos incluindo processamento básico de imagens e técnicas que auxiliam a detecção de objetos como pirâmide de imagens, método da janela móvel e métodos manuais de extração de características.

Detecção de objetos feita por computadores é um problema de dificuldade formidável. Isso se dá porque a variedade de cenários, escalas, posições e até iluminação dos objetos de interesse contribuem para a ineficácia das abordagens analíticas, que observam formas geométricas, textura, coloração entres outras características. Até por volta dos anos 2000, os algoritmos criados para detecção de objetos não eram capazes de superar o desempenho de um humano de 2 anos de idade [19]. Em 2002 o cenário começou a mudar com a publicação do sistema de reconhecimento de rostos baseado em aprendizado de máquina com autoria dos pesquisadores Viola e Jones [20]. A abordagem proposta por eles foi um grande sucesso na época, e serviu de base para softwares de uso comercial em câmeras nos dias de hoje. A solução de [20] foi usada neste trabalho para a seção de detecção de números tratado no Capítulo 4 adiante. Os detalhes técnicos da abordagem serão discutidos nesta seção.

3.1 O Método da janela móvel

Dada uma imagem de entrada, como saber se há presença de objetos de interesse e, caso afirmativo, onde estes se encontram na figura ou vídeo? O método da janela móvel é uma técnica de busca de objetos consolidada, e é conhecido por sua simplicidade de visualização. A busca por objetos por meio do método da janela móvel é feita escolhendo-se um retângulo de dimensões normalmente inferiores às da imagem de entrada original. O tamanho das dimensões deste retângulo são parâmetros de escolha do próprio projetista. Este retângulo então é aplicado a um dos extremos da imagem. O algoritmo dos classificadores em cascata então recebe a sub-região delimitada na imagem pelo retângulo e classifica a presença ou não de um objeto de interesse naquela sub-região. O retângulo então é movido um número pequeno de pixels para sua lateral e repete-se a classificação. E assim, o retângulo vai se movendo pela imagem como se fosse um filtro de convolução, sempre passando a sub-região da imagem de entrada à cascata de classificadores. O método é batizado de janela móvel pois o retângulo em questão se move pela imagem inteira em busca de se alinhar com um padrão que indique à cascata de classificadores que existe um objeto de interesse naquela sub-região específica.

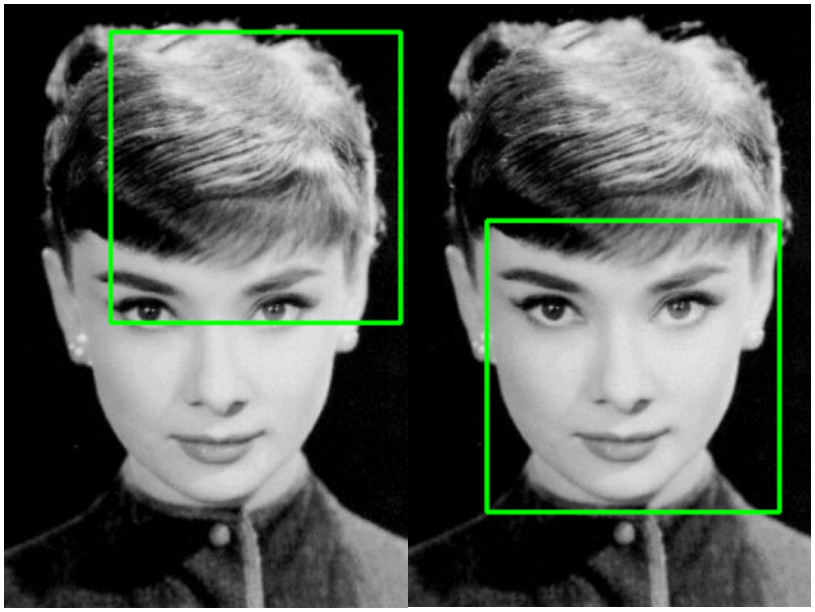


Figura 3.1 - Exemplo ilustrativo do método da janela móvel para detecção de rostos [21]

Este método por si só, no entanto, apresenta dois problemas claros. O primeiro sendo que a detecção pode ficar ruim se a razão entre dimensões do retângulo estiver em desacordo com a dimensão usual do objeto de interesse. Para este problema, deve-se escolher uma razão entre as dimensões compatível com a maior parte das aparições dos objetos. A limitação desta técnica é que a razão fixa do retângulo não generaliza bem para objetos com grande variância na razão de suas dimensões. Embora existam algoritmos mais recentes que consigam contornar este problema sua complexidade e custo computacional não justificam seus usos neste projeto. A janela móvel é uma alternativa aceitável para esta detecção de dígitos em específico, visto que os números em questão tem razões similares.

Um segundo problema presente na detecção por meio da janela móvel é a escala desconhecida na qual podem aparecer os objetos de interesse. O retângulo ou janela tem um tamanho fixo enquanto objetos podem aparecer mais próximos ou mais distantes na imagem. Para mitigar a influência da escala na busca de objetos por janela móvel, existe uma técnica chamada pirâmide de imagens. Esta técnica visa re-ajustar o tamanho da imagem de diversas formas de modo que uma única janela de tamanho fixo seja capaz de identificar objetos em várias escalas.

3.2 Pirâmide de Imagens

Como mencionado anteriormente, a técnica da janela móvel tem eficácia otimizada se usada em conjunto com a técnica da pirâmide de imagens. Esta técnica para detecção de objetos toma como entrada a imagem de interesse no tamanho original. Então se aplicam transformações de tamanho na imagem de modo que se criem várias outras imagens a partir da original, todas em diferentes resoluções. Uma imagem de entrada pode ser transformada de maneira a aumentar sua resolução ou diminuí-la. Estas operações denominam-se *interpolação* para aumento de tamanho e *decimação* para diminuição. São muitas as maneiras de se realizar estas transformações de tamanho, por exemplo métodos lineares, bilineares e cúbicos. No entanto estes métodos estão além do escopo desta fundamentação teórica e podem ser aprofundados com a leitura em [19].

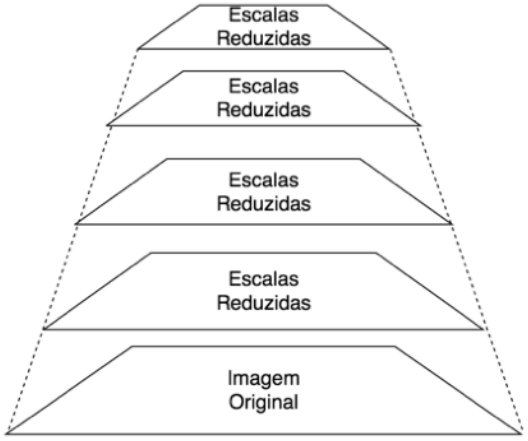


Figura 3.2 - Diagrama de uma pirâmide de imagens

Como ilustrado na figura 3.2, a janela móvel pode passar pelas diferentes resoluções da imagem inicial com um retângulo de tamanho fixo. Desse modo, um mesmo tamanho de janela é capaz de cobrir diferentes escalas possíveis para aparições dos objetos desejados desde que estes sigam uma razão de dimensões relativamente constante. O número de redimensionamentos que a imagem sofre na aplicação de uma pirâmide de imagens bem como o fator de escala entre cada transformação são parâmetros de escolha do projetista do sistema. A detecção de objetos por meio da janela móvel melhora conforme se aumenta o número de imagens geradas através de redimensionamentos e se decresce o fator de redimensionamento. O ganho de precisão no aumento de imagens é compensado por um gasto computacional extra, advindo do aumento no número de janelas que devem ser processadas. Para detecção de objetos, é preciso analisar cada caso específico para que haja uma precisão aceitável sem muita demora na conclusão da busca.

3.3 Cascatas de classificadores

Até o momento foi discutido como se quebra uma mesma imagem de entrada em sub-regiões de diferentes escalas e mesma razão entre dimensões. Isso é feito para que classificadores sejam capazes de determinar a presença ou não de objetos de interesse em diferentes pontos da imagem. Os classificadores em si são algoritmos de aprendizado de máquina, e as aplicações podem utilizar um ou múltiplos modelos de classificação. Uma abordagem mais simples seria acoplamento de uma rede neural multi-camada para classificação de cada janela individual, como descrito pelo diagrama da figura 3.3. A abordagem utilizada na implementação discutida no capítulo da **Discussão** é baseada na técnica proposta por Viola e Jones em sua publicação [20]. Este método faz uso de múltiplos modelos de aprendizado bem simples, tomando uma decisão baseada na maioria dos classificadores (técnica de nome *adaboost*).

A figura 3.3 ilustra a passagem das janelas diretamente ao classificador com somente alguns ajustes de processamento. No entanto, dependendo do tamanho da imagem e das escolhas feitas na construção da pirâmide, o processo de busca por objetos de interesse ainda pode ficar muito oneroso do ponto de vista computacional. Seria interessante que houvesse algum modo de reduzir o número de entradas passadas aos modelos classificadores para acelerar a conclusão da busca. Felizmente, existem técnicas manuais para extração de características de imagens que podem ser usadas nas janelas. A aplicação da extração de características em uma imagem retira traços que definem a imagem e diminuem a

dimensionalidade das entradas passadas aos modelos de classificação, acelerando sua propagação. Trataremos duas destas técnicas nesta fundamentação teórica, Haar e Hog.

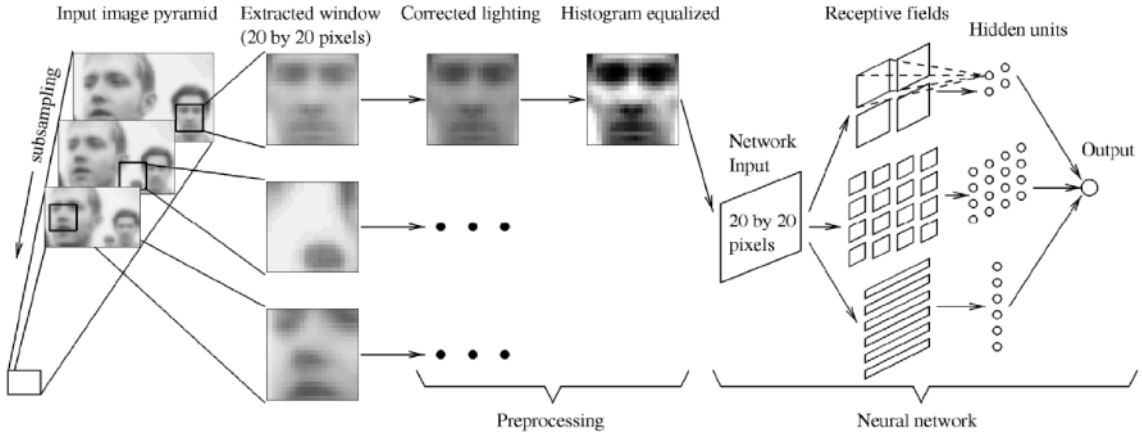


Figura 3.3 - Exemplo de sistema de detecção de objetos sem extração de características [19]

Haar é a técnica de extração de característica utilizada no detector de números durante os procedimentos deste trabalho, e tem seu nome porque a técnica se baseia no wavelet de Haar. Haar extrai características por meio da subtração de pixels em áreas mais claras pelas áreas mais escuras na imagem em diferentes pontos da imagem [19]. Esta subtração é executada em vários pontos de uma imagem e então as informações são então passadas adiante ao classificador em forma de vetor. Normalmente as sub-regiões usadas são obtidas por meio de um algoritmo de aprendizado de máquina chamado boost.

A segunda técnica para extração de características se chama Hog, sigla para Histograma de orientação de gradientes da imagem. Hog extrai os gradientes entre os pixels em toda a imagem, passando esta versão "simplificada" da imagem adiante para classificação. A extração de gradientes é feita em várias sub-regiões da imagem onde se identifica a orientação e a magnitude do gradiente resultante de cada sub-região. O vetor simplificado consiste em um histograma dos gradientes extraídos de cada sub-região ponderado por suas magnitudes.

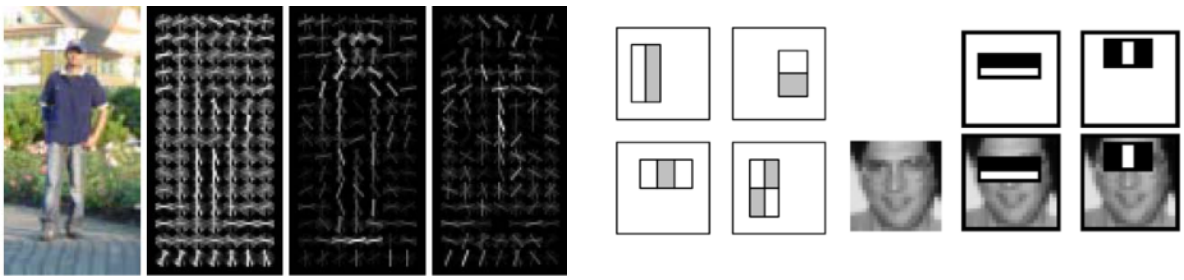


Figura 3.4 - Ilustrações das técnicas Hog (a esq.) e Haar (a dir.) para extração de características adaptado de [19]

O treinamento dos classificadores é feito como foi discutido na seção de aprendizado de máquinas. Junta-se um grupo de imagens com resolução das janelas contendo uma instância do objeto de interesse e um grupo de "imagens de fundo", sem a presença destes objetos. O sistema então é treinado por várias épocas atualizando seus pesos por meio da retro-propagação do erro.

3.4 Considerações Finais

Neste capítulo, foi feita uma revisão de conceitos dos campos da Detecção de Objetos, fundamentando conceitos e principais algoritmos importantes para a compreensão dos procedimentos realizados no desenvolvimento deste projeto.

A Detecção de Objetos fornece algoritmos complexos porém úteis para diversas aplicações comerciais hoje como a detecção de texto em documentos impressos (OCR). Existe uma variedade de técnicas para detecção de objetos em imagens e quadros de vídeo. Foi apresentada aqui uma das maneiras pelas quais se pode detectar objetos, fazendo uso de cascatas de classificadores simples. As técnicas de pirâmide de imagens e o método da janela móvel foram também apresentados aqui. Seu uso, no entanto não se restringe à cascata de classificadores, tendo seu uso explícito em outras técnicas de detecção de objetos.

O capítulo 4 a seguir consiste na apresentação de resultados e análises por trás do projeto. Nele, será feita uma discussão sobre as escolhas que tiveram de ser feitas conforme se aproximava do objetivo do projeto. Serão também justificadas as ações tomadas e apresentados os resultados do projeto após a sua conclusão.

4 Resultados e Análise

Podemos dividir o problema de reconhecimento de dígitos em dois módulos: para se realizar o reconhecimento, o sistema desenvolvido deverá inicialmente detectar a presença de números em uma imagem de entrada. A seguir, dado que houve sucesso na localização dos dígitos na imagem, deve-se encaminhar os dígitos detectados como janelas da imagem original para um outro classificador cuja responsabilidade é classificar de fato o número presente nas sub-imagens em sua entrada. O conjunto representaria a solução final do problema maior de detecção e reconhecimento.

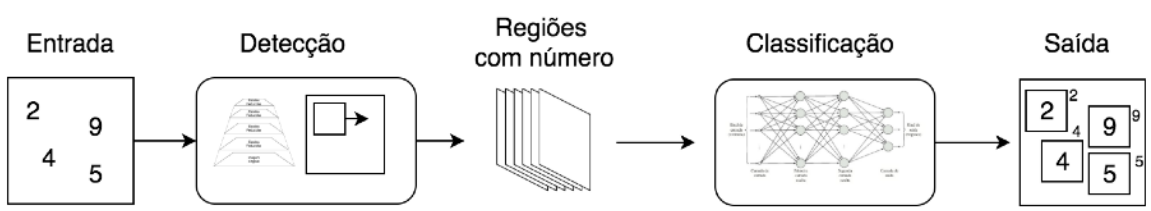


Figura 4.1 - Diagrama de blocos do sistema de detecção e reconhecimento completo

4.1 Classificação de dígitos

Optou-se por dar início à implementação do sistema pela concepção da seção classificadora de números. Isto se deu por conta da conveniência de uma extensa gama de soluções em reconhecimento de dígitos ilustrado pela página da base de dados MNIST de dígitos manuscritos [14]. A base de dados MNIST, sigla do inglês *modified national institute of standards and technology*, é uma versão modificada de uma coleção de dados do Instituto NIST, que foi produzida pelo pesquisador Yan Lecun e sua equipe.

A base MNIST contém aproximadamente 60000 imagens para treinamento e 10000 para validação com resolução 28x28 de números manuscritos extraídos de cartas do correio norte-americano. Todos os dígitos extraídos para compor a base modificada passaram por um processo de enquadramento em uma caixa delimitadora de tamanho 20x20 e foram centralizados na figura conforme o centróide do número. A coloração foi outro critério usado para padronizar os números do MNIST, todos os números da base estão escritos em branco em fundo preto.

O MNIST é uma das bases de números mais famosas do campo do aprendizado de máquina porque entre os anos 1990 e 2000 Lecun e sua equipe não só montaram a base como também realizaram um estudo minucioso de diferentes modelos de classificadores comparando suas performances.

4.1.1 Classificação de dígitos com MLP

Como uma primeira solução simples, implementou-se um modelo de rede neural multicamada clássico (MLP) com duas camadas escondidas, a primeira com 300 unidades e a segunda com 100 respectivamente. Usando uma taxa de aprendizado e um parâmetro de regularização ambos de valor 0.01, o modelo foi treinado com exemplos de treinamento da própria base MNIST por meio do método de otimização Decida de Gradiente Estocástica em mini-batches de 100 exemplos de cada vez.

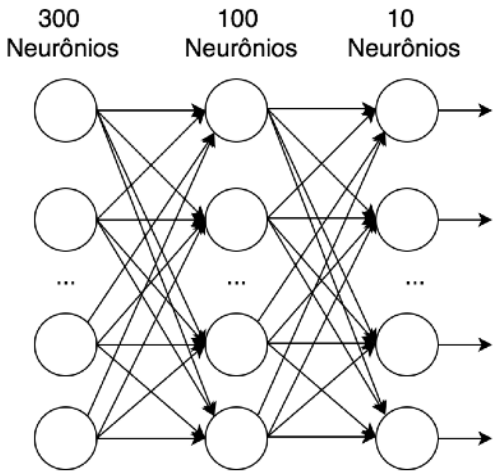


Figura 4.2 - Modelo de rede multi-camada MLP proposta para classificação

Os exemplos foram importados da base MNIST no formato e divisão padrões, em que são selecionados 60000 exemplos para treinamento e 10000 para a validação. Como mencionado anteriormente, cada exemplo consiste em uma imagem de resolução 28x28 que, ao ser passado para o modelo classificador é apresentado como um vetor com 28x28 = 784 elementos. Cada elemento destes é tratado como uma característica de entrada para o modelo classificador.

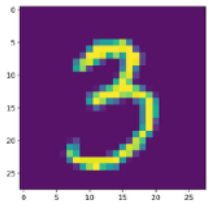
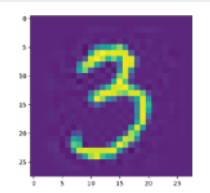
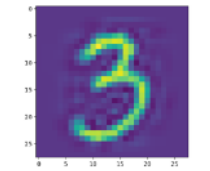
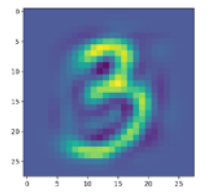
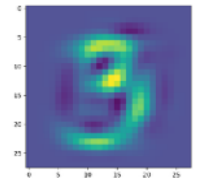
O primeiro treinamento teve duração de aproximadamente 20 minutos, resultando em uma acurácia de 97.2% no conjunto de exemplos de validação do MNIST. Cogitou-se realizar uma análise de arquitetura de rede para determinar o modelo de melhor desempenho para a aplicação, mas a própria página da base MNIST já oferece uma tabela comparativa com desempenho de vários modelos aplicados a estes exemplos.

Aproveitando-se que o projeto estava em fase de treinamento do classificador, foi feito um estudo dos ganhos de velocidade no treinamento do modelo em conjunto com uma técnica chamada análise de componente principal (PCA). Com seus aspectos teóricos discutidos na seção 2.5 do capítulo Aprendizado de Máquina, o vetor da imagem tem seu comprimento reduzido mantendo controle sobre variância específica da base de dados pelo uso do PCA. Isto diminui a quantidade final de pesos a serem ajustados no treinamento e também a serem considerados na propagação pela rede, simplificando cálculos e acelerando a rede. A tabela 4.1 apresenta variados treinamentos do mesmo modelo aplicando PCA com retenção de diferentes variâncias do modelo.

A última coluna da tabela 4.1 corresponde à visualização dos efeitos que as diferentes taxas de retenção de variância do PCA tem sobre um número. A visualização é feita aplicando-se a transformação PCA inversa à saída do PCA aplicado ao número. Ao retornar o número à sua

dimensionalidade original, é possível observar os efeitos que a perda de variância tem sobre o número.

Tabela 4.1 - Efeito da redução de dimensionalidade no modelo

Porcentagem de variância retida no PCA	Características de entrada	Tempo de treinamento (segundos)	Acurácia de validação do modelo	Exemplo
100%	784	697	97.53%	
99%	538	408.8	97.52%	
95%	329	314	97.66%	
85%	182	210	97.53%	
70%	96	169	97.62%	
30%	12	174	93.97%	N/A

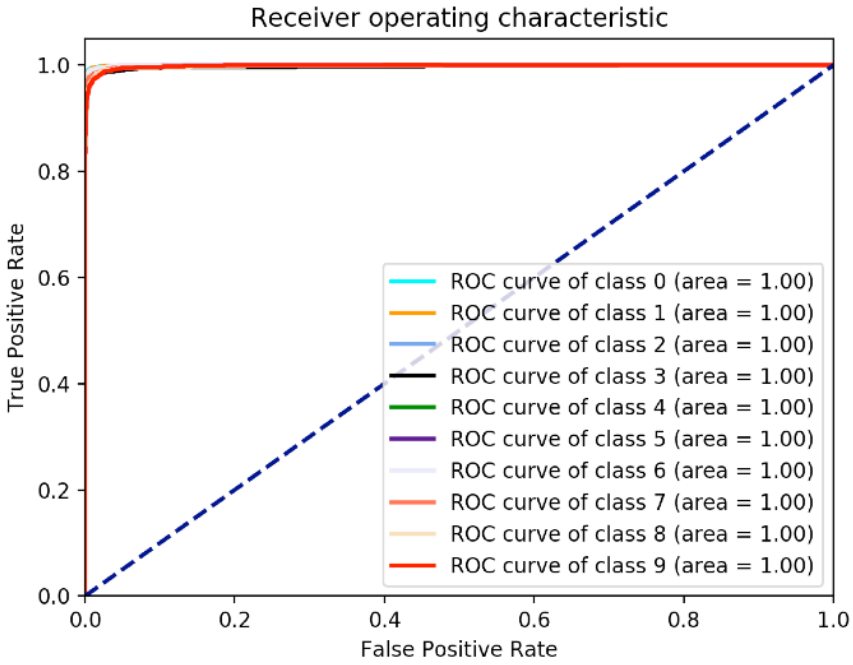


Figura 4.3 - Curva ROC por classe do modelo de classificação multi-camada
(Áreas iguais a 1 por conta de limites de precisão de cálculo existentes no software de desenho de gráfico)

Pôde-se perceber por meio da aplicação desta técnica, que métodos como PCA, que reduzem a dimensionalidade da entrada removendo redundâncias, são capazes de acelerar muito o treinamento e a classificação de novos exemplos por conta da queda no custo computacional de se processar um exemplo no modelo. A última coluna da Tabela 4.1 ilustra as imagens após passarem pela redução de dimensionalidade e a maneira como o exemplo é afetado. Visando averiguar melhor o desempenho do modelo treinado, optou-se pela extração das curvas características de operação do receptor, ou ROC. As curvas ROC são boas maneiras de se estudar a adequação de um classificador binário após treinamento. Como descrito na seção 2.3, estas curvas relacionam as taxas de falsos e verdadeiros positivos, de tal maneira que a área abaixo de uma curva ROC sinalize sobre o desempenho de um modelo específico. A área abaixo de uma curva ROC pode, em tese variar entre 0 e 1, sendo 1 o valor máximo teórico que um classificador poderia atingir caso aprenda um hiper-plano para separar a classe em análise das demais perfeitamente.

Uma adaptação necessária para o desenho da curva de ROC para um classificador de múltiplas classes em vez de binário foi o uso do critério “todos contra um”. Este critério estabelece cada classe individual como uma classificação binária, quebrando o gráfico em 10 curvas ROC diferentes para cada número. Neste contexto, uma curva para cada classe individual do problema (de 0 até 9). O resultado se dá na Figura 4.3.

De fato, as áreas das curvas ROC para todas as 10 classes do problema de classificação se aproximaram de tal maneira do valor ideal 1 que o software usado para desenho do gráfico arredondou as áreas para 1. Na verdade, pelo bom resultado de validação (acurácia de ~97.5% sobre os 10000 exemplos de validação do MNIST), todas as áreas teriam seu valor bem próximo de 1. Com a qualidade de desempenho e generalização asseguradas pela ilustração nas curvas ROC e pela magnitude das porcentagens de acerto da validação, concluiu-se que o treinamento do modelo discriminador de números estava satisfatório. O último passo a ser tomado para tomar esta como sendo a primeira sub-solução do problema de detecção e reconhecimento é a inserção e classificação de exemplos tirados de imagens exteriores ao MNIST.

4.1.2 Classificação de números exteriores ao MNIST

Apesar do desempenho muito bom do modelo treinado nos exemplos de validação do MNIST, este não aparentou ter a mesma capacidade de generalização para exemplos tirados de imagens de câmera de celular desprovidas do processamento especificado no MNIST. Para o bom funcionamento dos classificadores, alguns passos foram necessários para adequar exemplos exteriores ao MNIST para classificação com os modelos treinados a partir da base. A seguir, seguem os passos iterativos dados até que se atingisse desempenho satisfatório para o modelo multi-camada.

Conforme é explicado na página do MNIST, todas as imagens da base de dados são algarismos brancos em fundo preto. Portanto, as imagens foram convertidas para escalas de cinza e foi realizada uma limiarização (ou *thresholding*) das imagens tiradas do celular para se encaixar melhor na situação de um exemplo nativo do MNIST.

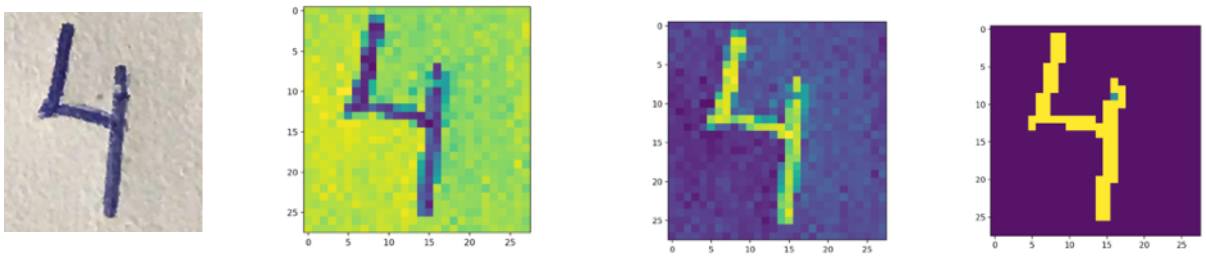


Figura 4.4 - Primeira iteração de pré-processamento de imagens exteriores ao MNIST

Percebe-se que houve necessidade de inversão de cores pelo fato de que o modelo havia sido treinado somente com exemplos do MNIST, com dígitos brancos em fundo preto. Isso foi feito para maximizar a precisão da classificação do modelo. Apesar deste pré-processamento, ainda assim o modelo ainda não se mostrou capaz de classificar dígitos processados desse modo. Um retorno à referência do MNIST indicou a necessidade de redimensionar o dígito em uma caixa 20x20, e por fim, de centralizar o "centroide" da imagem.

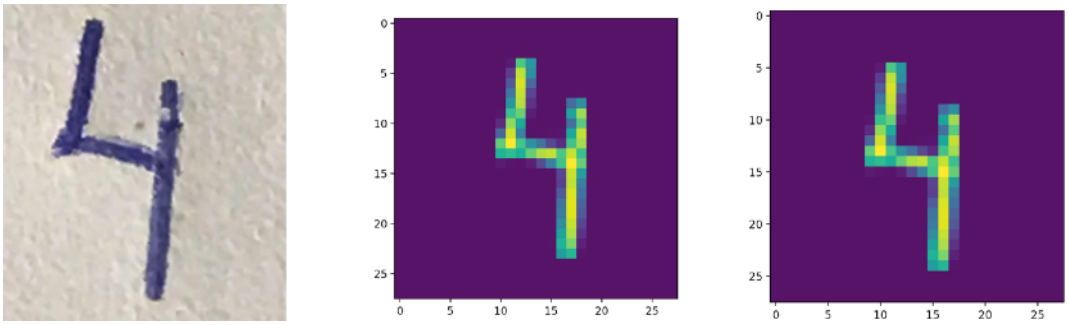


Figura 4.5 - Segunda iteração de pre-processamento de dígitos para classificador multi-camada

A realização dos procedimentos de pré-processamento indicados acima permitiram que o modelo classificador de dígitos discriminasse corretamente dígitos inseridos desta maneira. Após classificação correta de alguns dígitos inseridos de teste, tomou-se por concluída a etapa de classificação com o modelo de rede multi-camada.

4.1.3 Classificação de dígitos com rede neural de convolução

Para finalizar a seção de reconhecimento de dígitos, implementou-se um modelo convolucional simples alternativo à rede multicamada. O modelo convolucional a substituir o MLP tem uma arquitetura relativamente pequena, composta por duas camadas de convolução, uma com 20 mapas e outra com 50 mapas respectivamente. Ambas as camadas são formadas por meio de um filtro 5x5 em intervalos de 1 pixel por deslocamento. As convoluções foram realizadas sem adição de margem às entradas, e terminavam conectadas a uma seção com unidades plenamente conectadas. Esta seção constitui um modelo análogo ao MLP, com uma camada escondida, com 500 unidades e uma camada de saída. Seu treinamento foi feito na mesma base do modelo multi-camada, com 60000 exemplos do MNIST. Com respeito ao desempenho do novo modelo, observou-se acurácia de 99% para o melhor treinamento da rede durante teste com os 10000 exemplos de validação também da própria base MNIST. Com isso, concluiu-se a etapa de implementação do modelo classificador de números treinado com a base MNIST, e foi possível prosseguir com a construção do sistema de detecção e reconhecimento de números.

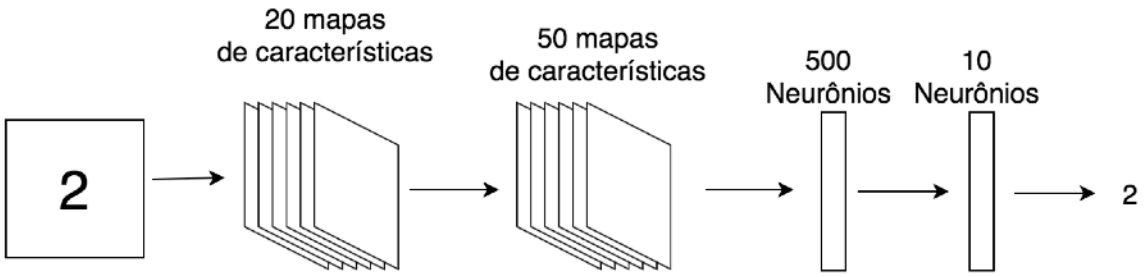


Figura 4.6 - Modelo de rede convolucional para classificação

4.2 Detecção de dígitos

O problema de detecção de dígitos se mostrou um grande obstáculo para a implementação do sistema de detecção de números. Para se obter um sistema que seja capaz de reconhecer áreas com presença de números em uma imagem maior, é necessário redimensionar a imagem para uma pirâmide com várias resoluções da imagem original, como especificado na seção 3.2 deste relatório. Através desta imagem de entrada e suas análogas de dimensões variadas, verificam-se sub-regiões de cada uma por meio de janelas móveis que escaneiam a imagem como em uma convolução.

Cada deslocamento da janela em uma determinada resolução da pirâmide irá produzir uma candidata a região com número. Com este método, uma única imagem poderia produzir centenas a milhares de sub-regiões a espera de classificação (se há objeto nela ou não). A grande quantidade de janelas a serem processadas impossibilitam o uso de classificadores muito complexos para discernir a presença ou não de números em cada uma. Isto nos leva a abordagem escolhida para o projeto.

Em tese, existem heurísticas para proposição de regiões de modo que o espaço de regiões a serem consideradas na detecção, a abordagem mais famosa é a busca seletiva [28]. A busca seletiva funde pixels de coloração e textura similares de maneira a fechar “caixas” ao redor de regiões de interesse. Técnicas de proposição de regiões como esta foram, de fato usadas para implementação de R-CNN e suas variantes, porém, elas tomam um tempo considerável de

processamento por imagem, principalmente quando não se dispõe de equipamento adequado (processadores de baixa velocidade ou GPU's). Uma técnica alternativa à proposição de regiões é denominada método dos classificadores em cascata, e esta foi escolhida como abordagem a ser seguida para solucionar o problema de detecção neste trabalho.

O método da cascata de classificadores considera que não há maneira simples de se reduzir o espaço de janelas a serem consideradas no problema. Esta suposição é tal que voltamos ao problema inicial, em que há um número muito grande de janelas a serem analisadas por imagem. Neste contexto, cada janela será classificada por uma cascata de classificadores lineares. A simplicidade dos classificadores da cascata otimiza o número de operações por imagem na realização da detecção. Desse modo, é possível reconhecer a presença de dígitos em todas as janelas de uma imagem sem necessidade de técnicas de proposição de regiões e em tempo mínimo. A seguir, serão discutidas as decisões e o raciocínio por trás da implementação da detecção de dígitos no projeto.

4.2.1 Detector de números pré-treinado

Em um primeiro momento, por razão da dificuldade computacional de se treinar um classificador em cascata, pesquisou-se na internet por um detector já pré-treinado para se implementar o fluxo necessário para se realizar a detecção de números. Felizmente, um detector treinado para uma aplicação bem similar ao projeto proposto neste trabalho estava disponibilizado por seu autor na rede [30].

O detector em cascata fazia uso da técnica de extração de características Haar bem como dos métodos auxiliares da pirâmide de imagens e janela móvel. O método de extração de características baseado na *wavelet* de Haar é uma técnica que auxilia a simplificar a janela como entrada dos classificadores em cascata. Por vezes uma entrada que consiste nos pixels da janela em questão, ao passar por um método manual de extração de características acaba por reduzir sua dimensionalidade, como no uso do PCA. Por este motivo, técnicas de extração manual de características são muito úteis para sistemas de detecção sem base nas técnicas do aprendizado profundo.

Tomando os parâmetros do classificador pré-treinado, foi possível implementar um fluxo de OCR preliminar como uma prova de conceito. Como proposto inicialmente, o detector de dígitos passava seções da imagem de entrada com uma chance grande de conter números para o sistema classificador, que discriminava os valores presentes. A imagem 20 ilustra como era feita a detecção e o reconhecimento dos números na primeira iteração do projeto.

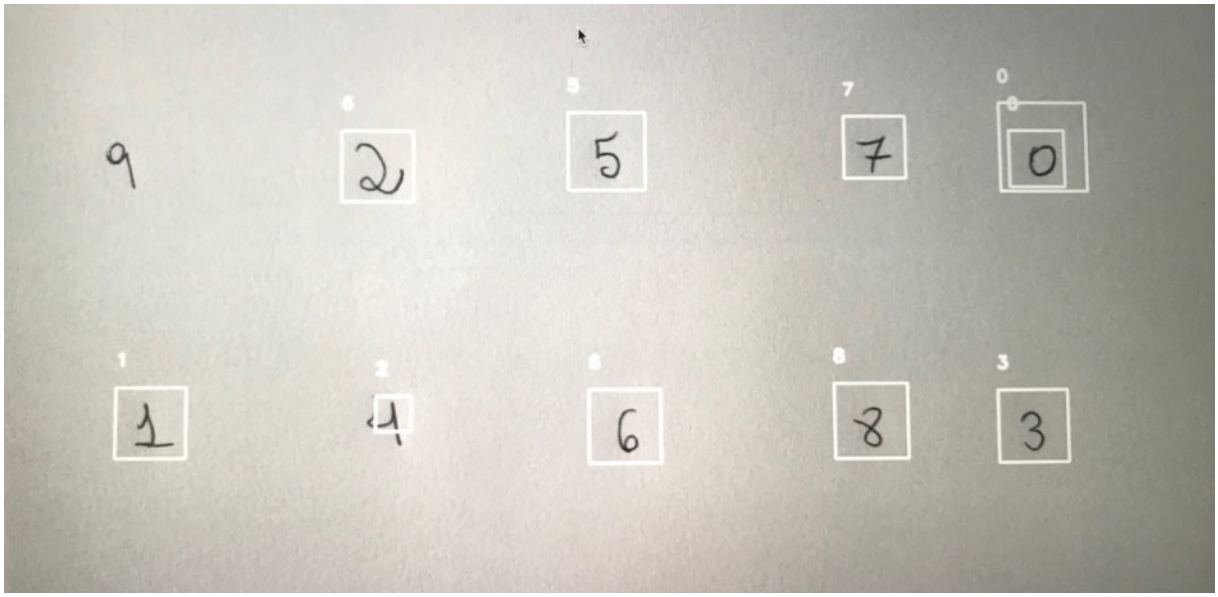


Figura 4.7 - Primeiro resultado do sistema com detector pré-treinado

No entanto, o sistema apresentava muitos falsos positivos e detectava em janelas de tamanhos incompatíveis com uma caixa envolvendo do dígito. O sistema também falhava em detectar alguns valores mesmo em condições ideais de detecção (algarismos pretos em fundo branco com pouco ruído de fundo). Neste contexto, foi tomada a decisão de fazer o treinamento de um detector em cascata próprio para o projeto. Por simplicidade e para facilitar a comparação com esta primeira implementação, selecionou a técnica de extração de características Haar para treinamento.

4.2.2 Detector treinado para o projeto

Em posse da teoria de detecção de objetos simplificada dos classificadores em cascata para uso em CPU's, foi realizada uma revisão na literatura acadêmica em busca de uma base de dados similar ao MNIST porém com exemplos na presença de ruído de fundo e não necessariamente de números brancos em fundo preto.

Foi encontrada uma base de números de casas obtidas com o uso do programa *street view* da Google, usada em uma publicação e disponibilizada gratuitamente pela Universidade de Stanford [7]. A base contém mais de 100 mil exemplos de números de casas oferecidos tanto no formato original e uma versão similar ao MNIST. O formato original eram imagens de múltiplos números em variadas resoluções da forma como foram fotografadas no programa *street view* da Google. A versão da base SVHN similar ao MNIST fazia um pré-processamento das imagens originais da outra versão, cortando cada número individualmente, redimensionando a imagem cortada para resolução 32x32 e centralizando o número na imagem. Esta foi a versão utilizada neste trabalho para treinamento do detector.

A base de números de casas tem nome SVHN, do inglês *Street View House Numbers dataset*, e se adequou aos objetivos finais do treinamento da cascata de detectores porque possuía números com uma variedade grande de tons, formatos, colorações e qualidades. Tal base de dados aparentava ser capaz de fornecer exemplos suficientes para aumentar a robustez de detecção para a cascata de classificadores a ser treinada.

Figura 4.8 - base de números de Stanford no formato MNIST [7]

A biblioteca python opencv para visão computacional e processamento de imagens oferece uma implementação aceitável para treinamento de um detector Haar em cascata. Selecionou-se uma fração da base de números de Stanford contendo 2000 exemplos positivos do grupo similar ao MNIST, com imagens de resolução 32x32 e algarismo centralizado.

Para exemplos negativos, foram baixadas imagens de fundo da plataforma web ImageNet para salas e pássaros. A ideia foi acumular um conjunto de imagens que certamente não conteriam números em seu interior. As imagens de fundo foram padronizadas em uma resolução de 200x200 pixels. De um total aproximado de 2400 imagens de fundo coletadas, 1000 foram efetivamente usadas no treinamento do detector.

Foram produzidos dois arquivos texto contendo as respostas de cada imagem negativa e positiva, onde as linhas do arquivo das negativas continham somente o caminho relativo para cada imagem de fundo. O arquivo com as respostas dos exemplos positivos era de formato similar ao negativo, no sentido que cada linha indicava o caminho relativo para cada imagem positiva porém com o adicional das coordenadas e dimensões das janelas contendo números. Provou-se ter sido uma boa escolha a seleção da base de Stanford nos padrões do MNIST, visto que simplificava muito o processo de marcação das janelas positivas. Para todos os exemplos positivos, a janela começava na coordenada (0,0) e sua dimensão era um o retângulo da forma 32x32.

O treinamento do classificador foi realizado em um total de 10 épocas passando pelos 2000 exemplos positivos e 1000 exemplos negativos. O processo se mostrou um tanto demorado, o treinamento completo levou aproximadamente 16 horas para ser concluído e consumiu muito

poder de processamento da unidade utilizada. Foi estudada a possibilidade de se fazer este treinamento localmente em um computador pessoal, porém a exposição prolongada a uma condição acima de 95% de uso de CPU levou o treinamento a ser efetuado em um servidor de um serviço de rede terceirizado para previnir danos à máquinas pessoais.

4.2.3 Validação e análise comparativa dos dois detectores

Após a execução do treinamento, o modelo de cascata resultante foi comparado com o modelo extraído da internet em um estudo de desempenho envolvendo um grupo de exemplos de validação criado à mão. Foram utilizados 40 exemplos de números divididos em 4 imagens. Ambos foram submetidos às imagens e foi observado o comportamento deles em uma perspectiva de *recall* e precisão, ilustrado na Tabela 4.3.

Table 4.2 - Observações da detecção no grupo de 40 números

	Modelo pré-treinado	Modelo treinado neste trabalho
Positivos Verdadeiros (TP)	32	39
Positivos Falsos (FP)	3	14
Negativos Falsos (FN)	11	1
Negativos Verdadeiros (TN)	N/A	N/A

Repare como o conceito de Negativos Verdadeiros não se aplica ao contexto de detecção de objetos. Um positivo verdadeiro consiste na detecção correta de uma janela que de fato continha um número. Positivo falso se trata da detecção em uma janela que na verdade não continha números. Já o negativo falso pode ser visto como a não detecção de um número em uma janela onde havia presença de número. Não faz sentido contar todas as áreas onde de fato não há números, portanto, o conceito de negativos verdadeiros não se aplica para a aplicação deste trabalho.

Felizmente, não é necessário conhecimento dos negativos verdadeiros para cálculo das grandezas precisão, recall e nota f1 quando se está em posse das demais medidas (TP, FP e FN). Os valores citados foram calculados para ambos classificadores.

Table 4.3 - Métricas de desempenho para detecção de números

	Detector pré-treinado	Detector treinado no projeto
$recall = \frac{TP}{TP + FN}$	0.744	0.975
$precisao = \frac{TP}{TP + FP}$	0.914	0.736
$F1 = \frac{Prec * recall}{prec + recall}$	0.4101	0.4194

A tabela 4.3 indica ganhos de recall advindos do treinamento de uma nova cascata de classificadores. Apesar de se ter tido perdas de precisão durante a substituição de detectores, observa-se que a troca causou um leve aumento na métrica de qualidade única f1. Embora o detector treinado tenha apresentado melhora no desempenho final do sistema de detecção, é

complexo analisar comparativamente os dois modelos devido ao montante pequeno de exemplos usados na validação.

Uma vez em posse dos pesos do detector treinado e validada a sua capacidade de detecção em relação ao detector anterior, este foi inserido no fluxo do sistema.

4.3 Análise do Sistema Completo

Como já mencionado, uma vez com desempenho satisfatório, ambos sistemas com detector pré-treinado e com detector criado foram avaliados em um conjunto de validação compatível com a aplicação a qual estes devem ser empregados. As imagens continham dez seções positivas uma de cada classe totalizando um montante de 50 exemplos para a validação. Na tabela 4.4, são ilustrados os resultados da performance de cada sistema:

Tabela 4.4 - Avaliação dos sistemas em conjunto de validação (melhor desempenho)

Sistema	OCR com detector pre-treinado(com MLP)	OCR com detector treinado(com MLP)
Desempenho no conjunto de validação	75%	90%

Como ilustrado na tabela 4.4, é perceptível a melhora de desempenho do OCR realizado com o novo detector treinado com a base de dados de Stanford em comparação ao modelo pre-treinado baixado da rede. Embora a descrição do modelo baixado mencionasse que este foi treinado com um conjunto de 7000 exemplos negativos e 9000 exemplos positivos, ainda assim sua performance foi 15% inferior a sua contraparte criada. Esta observação pode sugerir dois fatores distintos que contribuem na ocorrência desta queda de desempenho. Primeiro, que aparentemente a base utilizada no treinamento do modelo treinado para o projeto oferece uma maior variedade de números que a base usada no modelo externo, e portanto, melhor generalização. Ou segundo, que os exemplos negativos utilizados no modelo pré-treinado continham exemplos com presença de elementos similares aos números que a detecção busca, limitando a capacidade do detector em discernir números de imagens de fundo. Ao que tudo indica, a resposta parece estar no meio termo entre estas duas suposições. Visto que o modelo nativo é superior na detecção de dígitos acima porém aparenta ter dificuldade em não sinalizar falsos positivos para imagens não otimizadas para a aplicação em questão. Diferentemente do modelo externo, que é melhor na não detecção de falsos positivos mas apresenta deficiências na detecção de positivos verdadeiros.

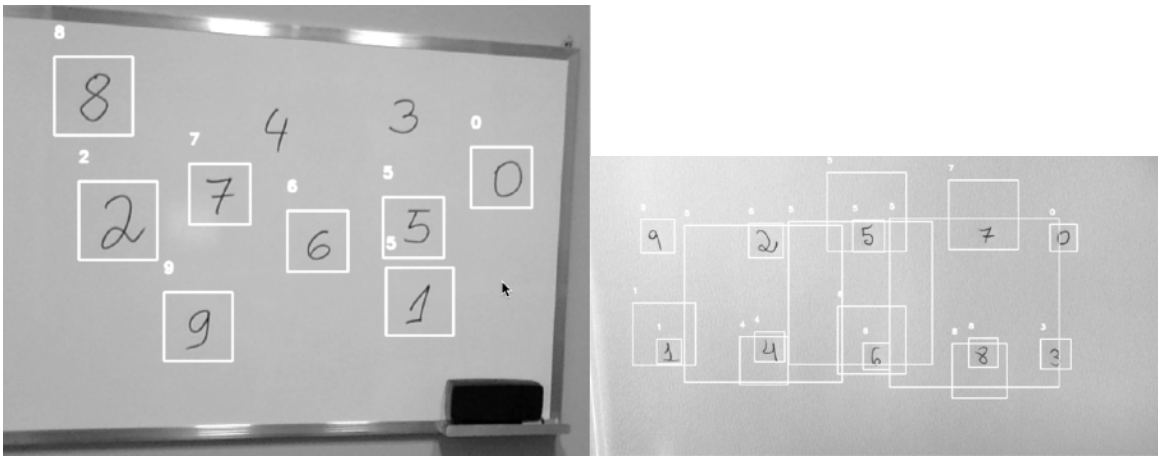


Figura 4.9 - OCR com detector externo (a esq.) e visualização do problema de escala (a dir.)

Estes resultados da tabela 4.4 correspondem ao melhor desempenho observado de cada classificador no conjunto no qual eram avaliados. As condições de avaliação foram fundo branco, números escritos em tinta escura e envoltos em caixas delimitadoras quadradas com dimensões entre 60x60 e 100x100 pixels. A limitação de tamanho para as janelas analisadas pelos classificadores contribuíram para diminuir o número de falsos positivos devido a regiões de interseção que, anteriormente acusavam presença de números indesejada nos resultados. A melhora decorrida da limitação de tamanho é ilustrada pelas imagens da Figura 4.9.

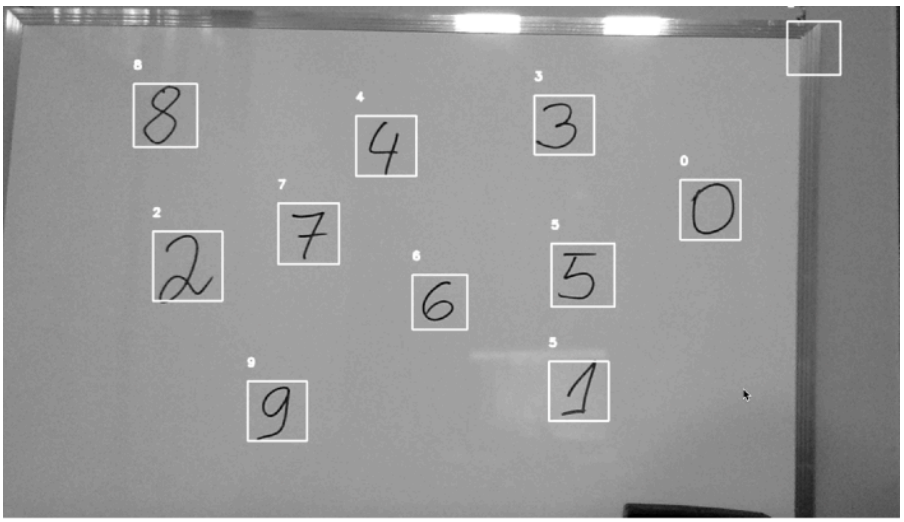


Figura 4.10 - Exemplo de Detecção pelo classificador nativo do projeto

4.4 Justificativas e Considerações Finais

Uma questão a ser colocada antes de finalizar os procedimentos realizados neste projeto é o porquê de se ter dado ênfase primeiro na seção de detecção de números no sistema em vez de investir na melhora da classificação dos números. Uma pergunta muito válida inclusive, que poderia ser reformulada para qual a ação a ser tomada que tem maior potencial de melhorar o sistema de OCR em questão. E a resposta a este questionamento pode ser encontrada em uma técnica chamada análise de limiar. A análise de limiar é uma técnica muito conhecida para otimizar a produtividade em projetos que envolvem fluxos de aprendizado de máquina [29]. Nosso sistema de OCR corresponde a um sistema com um fluxo deste tipo. A detecção de dígitos é realizada por meio de uma variedade de processos, incluindo pré-processamento de imagens e classificação de janelas em cascata. Já a classificação de dígitos envolve também pré-processamento de imagens em conjunto com outro classificador discriminador de números. A performance total do sistema é uma propagação do rendimento individual de cada uma destas duas estruturas. A presença de múltiplas estruturas relativamente complexas neste projeto torna a decisão de onde gastar a maior parte dos recursos (tempo e esforço) dos pesquisadores uma tarefa não trivial. Por meio do auxílio da análise de limiar, foi possível fazer uma escolha racional no sentido de otimizar o sistema da maneira mais eficiente.

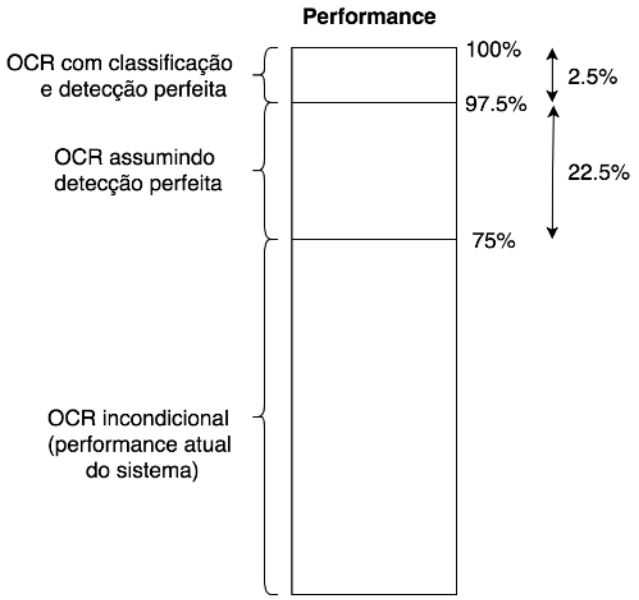


Figura 4.11 - Análises de limiar do sistema com detector treinado no projeto e MLP

Após o término da detecção feita com auxílio do detector baixado da internet, um protótipo preliminar foi estabelecido como prova de conceito. Observou-se, como ilustrado na análise do sistema completo que sua performance média no conjunto de validação orbitava em 75%. Tendo esta informação e cruzando-a com o desempenho individual do classificador treinado com a base MNIST assumindo detecção perfeita, pôde-se traçar o gráfico da Figura 4.11. Nota-se que a inclusão da detecção implementada reduziu performance do OCR em 22.5% comparado à situação de detecção perfeita. Enquanto isso, a detecção caiu somente 2.5% na inserção da seção de classificação implementada em relação à classificação perfeita. Tendo em vista estas informações, a decisão de onde atuar para melhorar a performance global do sistema de OCR se simplifica bastante. Uma vez que melhorias na classificação de dígitos pode contribuir no máximo com 2.5% para o desempenho do OCR. Enquanto melhorias na detecção de dígitos podem elevar o desempenho global do OCR em até 22.5%. A utilidade da análise de limiar foi comprovada no projeto de OCR descrito neste relatório, visto que a primeira ação de otimização tomada foi a melhoria na detecção de dígitos por meio do treinamento de um detector com melhor capacidade de generalização. Esta melhoria reduziu a queda de desempenho do sistema de 22.5% para 7.5%, resultando em um ganho de 15%. O gráfico da figura 4.12 corresponde à nova análise de limiar feita agora com a detecção de dígitos melhorada.

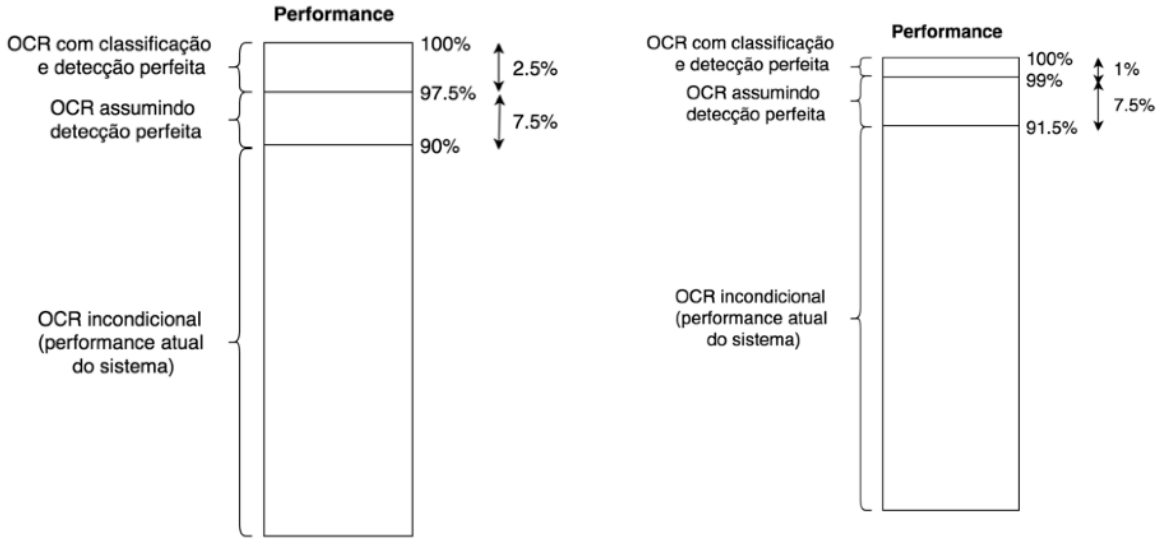


Figura 4.12 - Análises de limiar com detector nativo no MLP (a esq.) e em uma CNN (a dir.)

Apesar de a análise de limiar ainda sugerir que a detecção é a melhor maneira de se gastar tempo e esforço na melhoria do sistema, voltamos à classificação de números. A motivação por trás desta contradição é a existência de algoritmos de classificação superiores ao MLP. Algoritmos com potencial para, além de aumentar a performance do sistema, também reduzem sua complexidade. Como última medida tomada para melhoria deste sistema de OCR, substituiu-se a rede neural multicamada por um modelo de rede de convolução simples. As redes convolucionais contam com camadas extratoras de características, e tornam a classificação pouco sensível a escala, rotação e posição do objeto na figura. Portanto, o uso de um modelo desse tipo pouparia alguns dos passos de pré-processamento das imagens a serem classificadas como centralização de centroide da figura e a adequação de tamanho para os números. A substituição resultou no novo diagrama de limiar à direita na Figura 4.12.

5 Conclusão

Este trabalho apresentou o fluxo de um sistema de detecção de números composto por uma seção classificadora e uma seção de proposição de regiões. A arquitetura seguida segue em acordo com as propostas que vem sendo estudadas na literatura para detecção de objetos de maneira geral.

Dentre as possíveis alternativas para propor regiões para classificação, foi utilizada a abordagem de "força bruta" com pirâmide de imagens e janela móvel em uma cascata de classificadores lineares. Experimentara uma cascata de classificadores pré-treinada da internet e uma cascata treinada especificamente para este trabalho. O treinamento da cascata nativa do projeto foi realizado com exemplos da base SVHN [7] para exemplos positivos e imagens de fundo negativas da base ImageNet.

Duas redes foram treinadas para classificar as regiões propostas pelo detector de números, uma rede neural multi-camada e um modelo de rede similar à CNN LeNet [14]. Ambos modelos foram treinados com a base de dados MNIST. O modelo de rede multi-camada clássico apresentou melhor resultado de classificação de 97.5% enquanto a rede de convolução superou esta marca acertando 99% dos exemplos do grupo de validação. A rede de convolução ainda dispensou certos passos de pré-processamento dos exemplos comparado à rede multi-camada.

Foram combinadas as redes de detecção e classificação em um único sistema para detecção de números. O sistema atualmente atua em condições de um nicho bem específico, sendo detecção de números escritos em tinta preta em fundo branco de um determinado tamanho e em tempo real. Apesar destas limitações, a arquitetura proposta para o projeto se mostrou eficaz na tarefa designada, chegando a uma taxa de acerto global de 90% do sistema como um todo.

Mesmo com as limitações atuais do sistema, foi possível atingir resultados interessantes para o objetivo deste trabalho. Aplicações de detecção de objetos, apesar de comuns em estudos e pesquisas hoje, tem seu foco voltado para detecção de pessoas, veículos entre outros elementos visuais. Não foram encontradas muitas referências para detecção de números em específico.

Como trabalhos futuros, tem-se intenção de fazer ajustes no projeto atual de maneira a minimizar as limitações que hoje são necessárias para o bom funcionamento da detecção de números. Uma segunda meta para trabalhos futuros consiste em explorar outras arquiteturas de detecção de objetos baseadas no aprendizado profundo como R-CNN mais rápida e modelos yolo comparando seu desempenho com relação à primeira implementação deste trabalho. O estudo de arquiteturas baseadas em aprendizado profundo para detecção e classificação demandará equipamento especializado para viabilizar o tempo de treinamento e a propagação individual de exemplos pelo sistema. A aquisição ou possibilidade de uso de uma GPU contribuirá para estudos com arquiteturas não exploradas neste trabalho.

Referências Bibliográficas

[1] LORENA, Ana Carolina; GAMA, João; FACELI, Katti. Inteligência Artificial: Uma abordagem de aprendizado de máquina. Grupo Gen-LTC, 2000.

[2] SILVA, Vinicius de Oliveira; Human Action Recognition in Image Sequences Based on a Two-Stream Convolutional Neural Network Classifier. Universidade de Brasilia, 2017.

[3] HAYKIN, Simon. Redes neurais: princípios e prática. Bookman Editora, 2007.

[4] RAMACHANDRAN, Vilayanur S.; BLAKESLEE, Sandra; SHAH, Neil. Phantoms in the brain: Probing the mysteries of the human mind. New York: William Morrow, 1998.

[5] SPERRY, Roger W. Neurology and the mind-brain problem. **American scientist**, v. 40, n. 2, p. 291-312, 1952.

[6] MCCULLOCH W.S., PITTS W.,"A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[7] NETZER Yuval, WANG Tao, COATES Adam, BISSACCO Alessandro, WU Bo , Y. Ng Andrew. Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*

[8] Y. Ng Andrew. "Aprendizagem Automática" 2018. Disponível em < https://www.coursera.org/learn/machine-learning/home/info>. Acessado em 11/05/2018

[9] CHAPELLE, Olivier; HAFFNER, Patrick; VAPNIK, Vladimir N. Support vector machines for histogram-based image classification. **IEEE transactions on Neural Networks**, v. 10, n. 5, p. 1055-1064, 1999.

[10] COLLOBERT, Ronan; WESTON, Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: **Proceedings of the 25th international conference on Machine learning**. ACM, 2008. p. 160-167.

[11] ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, v. 65, n. 6, p. 386, 1958.

[12] KRIZHEVSKY A., SUSTSKEVER I., and HINTON G. E., "Imagenet classification with deep convolu- tional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[13] JARRETT K., KAVUKCUOGLU K., LECUN Y. *et al*., "What is the best multi-stage architecture for object recognition?" in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2146–2153.

[14] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998

[15] DAVIS, Jesse; GOADRICH, Mark. The relationship between Precision-Recall and ROC curves. In: **Proceedings of the 23rd international conference on Machine learning**. ACM, 2006. p. 233-240.

[16] DENG, Li et al. Deep learning: methods and applications. **Foundations and Trends® in Signal Processing**, v. 7, n. 3–4, p. 197-387, 2014.

[17] SMART J. "Your Personal AI (PAI): Pt 4 — Deep Agents (Deep Learning and Natural Intelligence)" 2018. Disponível em < https://medium.com/@johnsmart/your-personal-sim-pt-4-deep-agents-understanding-natural-intelligence-7040ae074b71 >. Acessado em 24/05/2018

[18] SHLENS, Jonathon. A tutorial on principal component analysis. **arXiv preprint arXiv: 1404.1100**, 2014.

[19] SZELISKI, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[20] JONES, Michael; VIOLA, Paul. Fast multi-view face detection. **Mitsubishi Electric Research Lab TR-20003-96**, v. 3, n. 14, p. 2, 2003.

[21] ROSENBROCK Adrian. "Sliding Windows for Object Detection with Python and OpenCV" 2015. Disponível em < https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/ >. Acessado em 28/05/2018

[22] ERIK BRYNJOLFSSON AND ANDREW MCAFEE. "What is driving the machine learning explosion?" 2017. Disponível em < https://hbr.org/2017/07/whats-driving-the-machine-learning-explosion >. Acessado em 08/06/2018

[23] GIRSHICK, Ross. Fast r-cnn. arXiv preprint arXiv:1504.08083, 2015.

[24] REN, Shaoqing et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: **Advances in neural information processing systems**. 2015. p. 91-99.

[25] REDMON, Joseph et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 779-788.

[26] GKIOXARI, Georgia et al. R-cnns for pose estimation and action detection. **arXiv preprint arXiv:1406.5212**, 2014.

[27] KARPATHY Andrej. "CS231n Winter 2016: Lecture 8: Localization and Detection" 2016. Disponível em < https://www.youtube.com/watch?v=GxZrEKZfW2o >. Acessado em 11/06/2018

[28] UIJLINGS, Jasper RR et al. Selective search for object recognition. **International journal of computer vision**, v. 104, n. 2, p. 154-171, 2013.

[29] RONCANCIO, Henry; HERNANDES, André Carmona; BECKER, Marcelo. Ceiling analysis of pedestrian recognition pipeline for an autonomous car application. In: **Robot Vision (WORV), 2013 IEEE Workshop on**. IEEE, 2013. p. 215-220.

[30] CHEUNG Joyee. "Digit Detection & Recognition" 2015. Disponível em < https://github.com/joyeecheung/digit-detection-recognition >. Acessado em 21/06/2018

[31] CARUANA, Rich; NICULESCU-MIZIL, Alexandru. An empirical comparison of supervised learning algorithms. In: **Proceedings of the 23rd international conference on Machine learning**. ACM, 2006. p. 161-168.

[32] LE, Quoc V. Building high-level features using large scale unsupervised learning. In: **Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference**. IEEE, 2013. p. 8595-8598.

[33] MNIH, Volodymyr et al. Human-level control through deep reinforcement learning. **Nature**, v. 518, n. 7540, p. 529, 2015.

[34] Data Science Academy. "O que é visão computacional?" 2017. Disponível em < http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/ >. Acessado em 27/06/2018