



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

Técnicas de aprendizado de máquina para detecção de pessoas em ambiente industrial

Trabalho de Conclusão de Curso submetido ao curso de Engenharia Elétrica da Universidade Federal de Santa Catarina como requisito para aprovação da disciplina EEL7890 - Trabalho de conclusão de Curso (TCC).

Eduardo Henrique Arnold

Orientador: Danilo Silva

Florianópolis, 7 de Dezembro de 2016.

EDUARDO HENRIQUE ARNOLD

**TÉCNICAS DE APRENDIZADO DE
MÁQUINA PARA DETECÇÃO DE
PESSOAS EM AMBIENTE INDUSTRIAL**

Trabalho de Conclusão de Curso
submetido ao curso de Engenharia
Elétrica da Universidade Federal de
Santa Catarina como requisito para
aprovação da disciplina EEL7890
- Trabalho de conclusão de Curso
(TCC).

Orientador: Danilo Silva.

**FLORIANÓPOLIS
2016**

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Arnold, Eduardo Henrique
Técnicas de aprendizado de máquina para detecção de
pessoas em ambiente industrial / Eduardo Henrique Arnold ;
orientador, Danilo Silva - Florianópolis, SC, 2016.
53 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico.
Graduação em Engenharia Elétrica.

Inclui referências

1. Engenharia Elétrica. 2. Aprendizado de máquina. 3.
Visão computacional. 4. Aprendizado profundo. I. Silva,
Danilo. II. Universidade Federal de Santa Catarina.
Graduação em Engenharia Elétrica. III. Título.

Eduardo Henrique Arnold

TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA DETECÇÃO DE PESSOAS EM AMBIENTE INDUSTRIAL

Este Trabalho de Conclusão de Curso foi julgado adequado no contexto da disciplina EEL7890 - Trabalho de conclusão de Curso (TCC), e aprovado em sua forma final pelo Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina.


Florianópolis, 7 de Dezembro de 2016.

Banca examinadora:



Prof. Danilo Silva, Ph.D.

Universidade Federal de Santa Catarina



Prof. Eduardo Batista, Dr.

Universidade Federal de Santa Catarina



Prof. Marcio Costa, Dr.

Universidade Federal de Santa Catarina

Agradecimentos

Desejo expressar meu reconhecimento a todos que, de uma maneira ou outra, colaboraram na realização deste trabalho. Primeiramente meu orientador Danilo Silva por todo o suporte, dedicação e paciência em me orientar semanalmente. Agradeço também aos meus pais, Sandra e Valmir, cujo suporte emocional e financeiro foi fundamental durante toda a extensão de minha graduação. Agradeço ao Instituto SENAI de Tecnologia e Sistemas Embarcados pela oportunidade em trabalhar nesse projeto que se revelou uma aplicação interessante para as técnicas de aprendizado de máquina. Por fim, agradeço à Universidade Federal de Santa Catarina, professores e demais servidores, pela oportunidade de ensino público, gratuito e de qualidade.

RESUMO

O presente trabalho aborda o desenvolvimento de um sistema de segurança industrial que requer detecção automática de pessoas. Esse sistema deve impedir que uma estrutura se movimente enquanto houver colaboradores em uma área de risco. O foco do trabalho é na detecção automática de pessoas, que é feita com base em imagens de profundidade, visão computacional e algoritmos de aprendizado de máquina. São propostas duas soluções para tal problema. A primeira se baseia em técnicas tradicionais de aprendizado de máquina utilizando extratores de características e classificador *Support Vector Machine*. A segunda utiliza técnicas de aprendizado profundo, mais especificamente redes neurais artificiais. A análise de desempenho das soluções revelou que os métodos de aprendizado profundo apresentam desempenho superior ao das técnicas tradicionais. Além disso, observou-se que as técnicas de aprendizado profundo não se restringem a grandes conjuntos de dados (big data), mas que podem ser empregadas com sucesso em situações com volume moderado de amostras, inclusive desbalanceadas.

Palavras-chave: Visão computacional. Aprendizado de máquina. Classificação. Redes neurais artificiais. Aprendizado profundo.

ABSTRACT

This work describes the development of an industrial security system that requires automatic human detection. This system must block the movement of a certain structure while there are workers under a hazardous area. The focus is on human detection, which is carried out based on depth images, computer vision and machine learning. Two solutions are presented. The first one is based on traditional learning techniques using feature extraction and a Support Vector Machine classifier. The second solution uses deep learning methods for classification, more specifically Convolutional Neural Networks. The performance analysis of both solutions revealed that the deep learning methods outperform traditional learning techniques. Moreover, it is shown that such deep learning methods are not restricted to big data applications, but could be used to moderate-sized unbalanced datasets.

Keywords: Computer vision. Machine learning. Classification. Artificial neural networks. Deep learning.

Lista de Figuras

1.1	Exemplo de descritores: histograma de bordas e cores. . . .	3
1.2	Hiperplano ótimo para separação das classes	4
1.3	A estrutura de um perceptron	5
1.4	Exemplo de MLP com 5 camadas intermediárias. Amarelo, verde e azul correspondem, respectivamente, às camadas de entrada, intermediária e de saída.	7
1.5	Exemplo de rede convolucional com max-pooling	9
2.1	Diagrama demonstrando etapas do sistema de detecção de pessoas e seus respectivos sinais.	11
2.2	Pontos em azul são máximos locais e os quadrados vermelhos indicam os candidatos centralizados.	14
2.3	Ilustração dos descritores: grades simples e anéis concêntricos.	15
2.4	Hiperplano de separação e margem para conjuntos linearmente separáveis.	17
2.5	Exemplos de dataset em que separação perfeita não é ideal.	18
2.6	Exemplo de mapeamento para obter um classes linearmente separáveis. Espaço transformado e original, respectivamente.	18
3.1	Convecção das conexões entre unidades.	24

3.2	Gráfico mostrando o comportamento da sigmoide.	25
3.3	Gráfico mostrando o comportamento da função RELU.	26
3.4	Comportamento da função custo entropia cruzada para amostras negativas (esquerda) e positivas (direita).	28
3.5	Exemplo de rede convolucional e campos receptivos.	31
3.6	Estrutura proposta para rede MLP.	32
3.7	Estrutura proposta para rede CNN.	33
4.1	Curvas ROC comparando classificadores SVM sob descritores de anéis com diferentes dimensões.	40
4.2	Curva de validação SVM com descritor de anéis 18dim.	43
4.3	Curvas ROC comparando diferentes classificadores apresentados e zoom na região de interesse	44
4.4	Curva ROC comparando diferentes parâmetros e classificadores do sistema completo	46

Lista de Tabelas

4.1	Matriz de confusão	36
4.2	Anéis concêntricos, 8 dimensões, conjunto de treinamento reduzido	39
4.3	Anéis concêntricos, 16 dimensões, conjunto de treinamento reduzido	39
4.4	Anéis concêntricos, 18 dimensões, conjunto de treinamento reduzido	39
4.5	Anéis concêntricos com 18 dimensões, conjunto treinamento	40
4.6	Anéis concêntricos com 18 dimensões, conjunto teste . . .	40
4.7	Grades simples, conjunto treinamento	41
4.8	Grades simples, conjunto teste	41
4.9	MLP, conjunto treinamento estendido $T = 0.5$	41
4.10	MLP, conjunto treinamento estendido $T = 0.9$	41
4.11	MLP, conjunto teste $T = 0.5$	41
4.12	MLP, conjunto teste $T = 0.9$	41
4.13	CNN, $T = 0.5$, conjunto treinamento estendido	42
4.14	CNN, $T = 0.9$, conjunto treinamento estendido	42
4.15	CNN, $T = 0.5$, conjunto de teste	42
4.16	CNN, $T = 0.9$, conjunto de teste	42

Sumário

1	Introdução	1
1.1	Aprendizado de máquina	2
1.2	Métodos clássicos de detecção de objetos	3
1.3	Aprendizado profundo	5
1.3.1	Perceptron multicamadas (MLP)	6
1.3.2	Redes convolucionais	7
1.4	Estrutura do trabalho	8
2	Detecção utilizando métodos tradicionais	11
2.1	Caracterização da aplicação	12
2.2	Obtenção de candidatos	12
2.3	Descritores	14
2.4	Classificação	16
2.5	Variação para saída probabilística	20
2.6	Implementação	20
3	Classificação utilizando aprendizado profundo	23
3.1	Perceptron Multicamadas (MLP)	23
3.2	Função de ativação	25
3.2.1	Sigmóide	25
3.2.2	RELU	26

3.3	Treinamento	27
3.3.1	Função custo	27
3.3.2	<i>Stochastic Gradient Descent</i>	28
3.3.3	Backpropagation	29
3.4	Redes convolucionais	30
3.5	Implementação	33
4	Resultados	35
4.1	Medidas de avaliação	35
4.2	Desempenho dos Classificadores	38
4.2.1	SVM	39
4.2.2	MLP	40
4.2.3	CNN	41
4.2.4	Discussão	42
4.3	Sistema completo	43
5	Conclusão	49
	Referências bibliográficas	53

CAPÍTULO 1

Introdução

Na atualidade, a segurança e bem estar das pessoas é um aspecto fundamental a ser garantido, ainda mais para trabalhadores em ambientes industriais, onde os riscos são mais severos. Diversas normas e regras existem no sentido de garantir que essa condição seja atendida. Com a progressão dos meios tecnológicos novas ferramentas surgiram para facilitar as medidas de controle e proteger as pessoas.

Um caso concreto para aplicação dessas ferramentas ocorre na Wanke, indústria de eletrodomésticos de Indaial-SC, que conta com pontes rolantes, que facilitam a movimentação de peças no meio industrial. Esse instrumento permite carregar moldes de até centenas de kilogramas entre estações de trabalho. É um fator de preocupação que um desses moldes se solte ou contenha peças livres que caíam e fiam algum funcionário.

Este trabalho pretende elaborar um sistema automático de detecção de pessoas que possibilite impedir que a ponte rolante se movimente enquanto houver colaboradores sob sua área de movimentação. Essa detecção é feita com base em imagens de profundidade e algoritmos de aprendizado de máquina. As imagens são obtidas através de uma câmera stereo de maneira que o valor de cada pixel representa a distância

entre a câmera e o objeto.

Os aspectos fundamentais e um comparativo entre técnicas tradicionais e de aprendizado profundo, conjunto de métodos de aprendizado de máquina no estado da arte, é apresentado. O resultado das diferentes abordagens em figuras de mérito específicas para classificação é destacado.

1.1 Aprendizado de máquina

Segundo Arthur Samuel [1], aprendizado de máquina é o ramo da inteligência artificial que estuda técnicas que possibilitam um computador realizar uma tarefa sem ser explicitamente programado para desempenhá-la.

Mitchell [2] define: “Se diz que um computador aprende com uma experiência E com respeito a uma tarefa T e medida de desempenho P à medida que o desempenho verificado por P diante de uma tarefa T progride com a experiência E ”.

Entre as tarefas pode-se citar classificação, em que uma amostra deve ser atribuída a uma das classes apresentadas; regressão, em que uma amostra deve ser mapeada em um valor real; síntese, onde o algoritmo deve gerar novas amostras que sejam similares às de entrada.

As medidas de desempenho são específicas da tarefa em questão e seu uso deve variar com a aplicação. Para conjuntos de dados simples e balanceados (distribuição uniforme de classes) uma medida comum a classificadores é acurácia, razão do número correto de classificações pelo número total. Porém outras medidas são necessárias para avaliar conjuntos de dados desbalanceados. Para problemas de regressão podem ser adotadas as medidas de erro médio quadrático.

Uma experiência, também chamada de exemplo ou amostra, é uma coleção de descrições obtidas através de um descritor de características, conforme ilustrado na Figura 1.1. Essa descrição é obtida através de um processo que pode mensurar alguma grandeza específica de algum objeto ou evento. Para o campo de imagens são comuns descritores de borda e histograma de cores. Em geral, um descritor busca obter uma representação compacta, reduzindo dimensionalidade, e discriminativa entre as amostras de maneira a facilitar a etapa de classificação. A etapa de extração de características pode ser vista como um pré-

processamento ou transformação para um domínio de maior relevância.

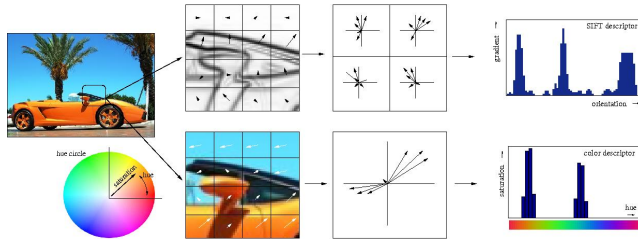


Figura 1.1: Exemplo de descritores: histograma de bordas e cores.

As experiências formam o conjunto de dados que o algoritmo utiliza para aprender, aqui também chamado de *dataset*. Esse conjunto de dados é dito supervisionado quando possui um indicador da classe que cada amostra pertence ou o valor esperado para aquela amostra, no caso de regressão. Para um conjunto não supervisionado, espera-se aprender mais sobre a distribuição dos dados. Isso permite, por exemplo, detectar amostras anômalas e dividir o conjunto de dados em grupos (*clusters*) que apresentem alguma similaridade estrutural.

Neste trabalho, deseja-se realizar a detecção de objetos, mais especificamente, pessoas. Essa tarefa é um caso particular de classificação e fará uso de um conjunto de dados supervisionados.

1.2 Métodos clássicos de detecção de objetos

O problema de detecção de objetos é um clássico no campo de visão computacional e, em geral, requer aprendizado de máquina. Uma abordagem tradicional [3] consiste nos seguintes passos.

- (i) Identificar uma região de interesse no caso de uma imagem com múltiplos objetos.
- (ii) Utilizar um extrator de características para descrição do objeto.
- (iii) Introduzir a amostra, proveniente do descritor, em um classificador simples, obtendo a classe correspondente.

Métodos tradicionais requerem que um extrator de características seja utilizado antes da classificação. Isso se deve à necessidade de reduzir a dimensionalidade das amostras e discriminar classes diferentes,

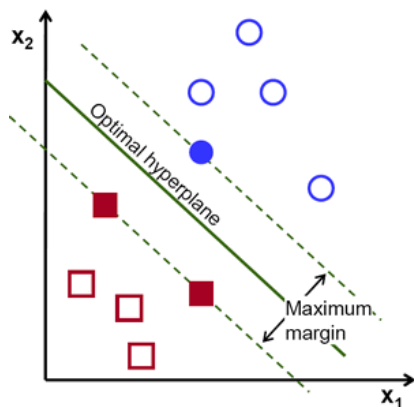


Figura 1.2: Hiperplano ótimo para separação das classes

o que possibilita ganhos de desempenho com baixa complexidade do classificador.

A escolha do descritor está diretamente ligada com a qualidade do processo de classificação. Um descritor é criado especificamente para uma aplicação e dificilmente pode ser utilizado em outros campos: um descritor para imagens é incompatível com um sinal de áudio. Assim, sua concepção demanda conhecimento e prática de especialistas da área de aplicação. Para um problema de classificação entre imagens de bananas e maçãs, por exemplo, um descritor relacionado a cor teria maior relevância do que um de bordas.

Após ter uma representação adequada das experiências escolhe-se o tipo do classificador. O modelo de Máquina de Vetores Suporte (*Support Vector Machines* – SVM), é utilizado pois possui representação e uso simplificado e demonstrou bons resultados [4] ao tratar amostras de dimensões medianas (ordem de centenas) com um conjunto de treinamento pequeno (ordem de milhares de amostras) sem ajuste fino. Seu modelo consiste em encontrar um hiperplano ótimo de separação entre classes, no sentido de maximizar a distância entre as amostras e o hiperplano. As amostras mais próximas do hiperplano são chamadas vetores suportes, como mostrado na Figura 1.2.

Para amostras com distribuição mais complexa a possibilidade de não existir um hiperplano que consiga separar o conjunto satisfatoriamente é grande. Para resolver esse problema emprega-se uma função

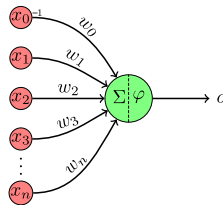


Figura 1.3: A estrutura de um perceptron

que mapeia essas amostras para um espaço de maior dimensionalidade em que as classes sejam linearmente separáveis.

1.3 Aprendizado profundo

Outro modelo de aprendizado de destaque são as redes neurais artificiais. Essas foram inspiradas no entendimento rudimentar dos neurônios biológicos ainda na década de 1950 [5], porém deixou o foco biológico para fornecer modelos robustos de aprendizado de máquina. Suas estruturas eram então compostas por um único elemento chamado de *perceptron*, neurônio artificial ou unidade, um modelo matemático descrito por

$$o(x) = \phi \left(\sum_{i=1}^n x_i w_i + b \right) = \phi \left(w^T x + b \right) \quad (1.1)$$

onde w é o vetor de pesos, b de *bias* (valor constante adicionado ao somatório de entrada) e $\phi(x)$ é chamada de função de ativação e em geral é não linear.

A estrutura do perceptron, ilustrada na Figura 1.3, permite utilizar um algoritmo de aprendizado simples e rápido. Entretanto alguns problemas impediram os avanços desse classificador, a destacar: a incapacidade de aprender uma tarefa comum como o ou-exclusivo (dado que as amostras não são linearmente separáveis) e o baixo poder de

processamento dos computadores da época.

A partir da década de 1970 com o aumento drástico da capacidade computacional essa estrutura ganhou complexidade e passou a ser integrada por múltiplos *perceptrons* organizados em camadas, o que possibilitou um aprendizado a partir de datasets mais desafiadores. Foi esse evento que iniciou a jornada para o advento das técnicas de aprendizado profundo.

1.3.1 Perceptron multicamadas (MLP)

A evolução do único perceptron para uma rede de camadas desse elemento aumentou a capacidade de aprendizado do modelo. Essa estrutura pode ser subdividida em: camada de entrada, camadas intermediárias e camada de saída, como ilustrado na Figura 1.4. As camadas de entrada e de saída são compostas por um *perceptron* para cada dimensão da entrada/saída. As camadas intermediárias são fundamentais pois correspondem justamente à extrair uma descrição da amostra inserida na camada de entrada.

Esse processo evolutivo só foi possível com o advento de algoritmos de aprendizado por propagação reversa (*backpropagation learning*) [6]. Tal algoritmo permite ajustar os parâmetros de cada unidade automaticamente baseado na influência que essa unidade exerce sobre o erro das saídas e assim minimizar o erro total. O ajuste dos parâmetros é realizado utilizando o algoritmo para calcular o gradiente da função custo, que indica o erro, em função dos parâmetros das diferentes unidades. Ajusta-se os parâmetros proporcionalmente a esse gradiente através de um método de otimização chamado de *Stochastic Gradient Descent* [7] até satisfazer alguma condição de erro mínimo num processo conhecido como treinamento.

Essa estrutura já era utilizada na década de 1980, porém era limitada pelo número de camadas intermediárias, dado o chamado problema do gradiente enfraquecido. Esse problema se resume na impossibilidade de alterar os parâmetros das camadas iniciais visto que o gradiente da função custo se enfraquece ao se propagar reversamente pela rede. Esse enfraquecimento advém da função de ativação utilizada até então, a sigmoide, conforme será visto na seção 3.2, que satura facilmente e causa uma derivada quase nula. Uma solução proposta em [8] prevê uma nova função de ativação, conhecida como RELU, des-

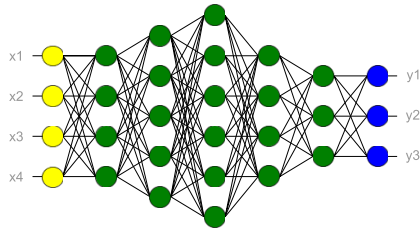


Figura 1.4: Exemplo de MLP com 5 camadas intermediárias. Amarelo, verde e azul correspondem, respectivamente, às camadas de entrada, intermediária e de saída.

crita posteriormente na seção 3.2, que resolve o problema do gradiente enfraquecido e permite criar redes com muitas camadas.

Uma rede é considerada profunda quando possui um grande número de camadas intermediárias. Essa característica, em contraposição com métodos de aprendizado superficial, permite que a amostra seja alimentada ao classificador sem pré-processamento. Isso é possível pois internamente o sistema gera um descritor ótimo para discriminação das amostras. Ou seja, o próprio classificador encontra a melhor maneira de representar as amostras para obter um bom resultado, economizando tempo e reduzindo a necessidade de especialistas em criar esses descritores.

1.3.2 Redes convolucionais

A evolução natural da rede MLP se deu ainda década de 1970 através das redes convolucionais. Uma aplicação de destaque foi a leitura de caracteres (OCR) de talão de cheques criada por LeCun em 1990 [9], utilizando a estrutura vista na Figura 1.5. Apesar disso os algoritmos de treinamento ainda não estavam completamente consolidados e acabaram sendo abandonados em detrimento de técnicas de aprendizado

superficiais, como *Support Vector Machines*.

Com o aumento drástico do poder de processamento essas redes ultrapassaram seus predecessores e ganharam destaque novamente após o trabalho de Hinton [10] em 2006, que estabeleceu uma base sólida para o treinamento de grandes redes convolucionais. Essa contribuição mudou o paradigma do aprendizado de máquina no que se refere à extração de características e se mostra ainda hoje como estado da arte.

Nessas estruturas definem-se filtros, na forma de matrizes, que são convolvidas com uma matriz (proveniente da camada anterior) e resultam em outra matriz, com algum aspecto realçado. Em geral, muitos filtros são definidos e há, portanto, muitas matrizes de saída para cada nível de convolução, o que aumenta drasticamente a dimensionalidade da rede. Nesses casos é comum utilizar um processo de *max pooling*, descrito na Seção 3.4.

A abordagem mais tradicional desse tipo de rede utiliza um número de camadas convolucionais para extrair características mais relevantes das imagens e em seguida utiliza uma rede MLP para realizar alguma operação de classificação ou regressão sobre essas descrições.

A grande vantagem dessas redes é reduzir o número de parâmetros geral do modelo e ainda assim manter uma alta capacidade de aprendizado. Elas também possibilitam que estruturas locais sejam encontradas mais facilmente de maneira a compor um descritor mais complexo e discriminativo para dados complexos como imagens e sinais de áudio.

Além disso, existe a vantagem da invariância à translação: em uma rede MLP, para encontrar faces em uma imagem, seria necessário amostrar em todas as posições possíveis da imagem. Já em uma rede convolucional esse problema é resolvido através da convolução e amostragem que garantem esse aspecto fundamental para detecção de objetos.

1.4 Estrutura do trabalho

Esse trabalho apresenta duas soluções para o problema de detecção de pessoas, cada uma descrita em um capítulo diferente. A primeira é fundamentada nos métodos tradicionais de visão computacional e aprendizado superficial. Divide-se o algoritmo em localização de candidatos à pessoas e posterior extração de características, desenvolvida

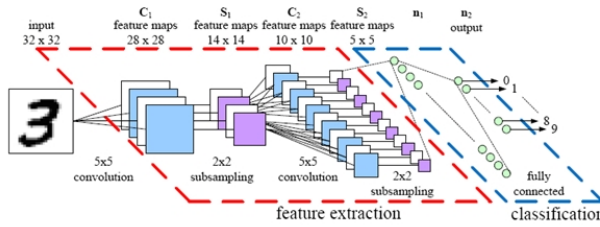


Figura 1.5: Exemplo de rede convolucional com max-pooling

manualmente, e classificação através de SVM.

Em seguida, propõe-se um método que emprega o mesmo algoritmo tradicional de localização de candidatos porém utiliza métodos de aprendizado profundo para a etapa de classificação. Compreende-se que o classificador encontra de maneira automática um descritor de características específico para o conjunto de dados do problema nas camadas de convolução. A classificação propriamente dita se dá nas camadas finais da rede profunda.

Após a apresentação teórica das soluções, são exibidos os resultados e as figuras de mérito que permitem comparar o desempenho das propostas e determinar a melhor solução.

Finalmente, a conclusão discute os resultados e o impacto da utilização das redes profundas. São descritas as contribuições do trabalho assim como sugestões de trabalhos futuros.

Detecção utilizando métodos tradicionais

O sistema de segurança na fábrica consiste na detecção de pessoas sob a área da ponte. Esse é um problema de localização de objetos, mais especificamente, de detecção de pessoas. Tradicionalmente divide-se essa tarefa em três técnicas de visão computacional: obtenção de candidatos a objeto, obter descrição de cada candidato e então validar a amostra através de classificador. Essa solução, apresentada visualmente na Figura 2.1, foi baseada no trabalho de [11], que propõe um método de obtenção de candidatos e de extração de características. Primeiramente descreve-se aspectos técnicos da aplicação e em seguida cada uma das tarefas para a concretização da solução, por fim, pontos sobre a implementação são mencionados.

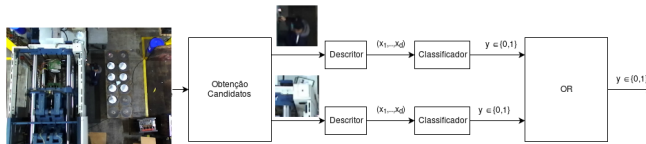


Figura 2.1: Diagrama demonstrando etapas do sistema de detecção de pessoas e seus respectivos sinais.

2.1 Caracterização da aplicação

Para essa aplicação, recomenda-se a utilização de imagens de profundidade, em que o valor dos pixels indicam a distância entre a câmera e o objeto em questão. Isso se deve ao fato de esse sensor fornecer uma medida independente de luminosidade e variações de cor ou textura se comparada à uma câmera RGB tradicional. Três câmeras foram compradas e avaliadas: *ASUS XtionPRO*, *Orbbec Astra PRO* e *Stereolabs ZED*. As duas primeiras funcionam com o princípio de laser estruturado, já a última funciona baseado em visão estéreo (duas câmeras). Através de testes determinou-se que as maiores distâncias detectadas foram de 4m, 5m e 20m respectivamente. Dado que a câmera será instalada a 6m do chão, em visão superior, optou-se por utilizar a câmera *ZED*.

As imagens recebidas da câmera precisam ser pré-processadas de forma a obter um mapa de distâncias, que é a imagem de profundidade. Isso é feito pela biblioteca fornecida pela *Stereolabs*. Em seguida, inverte-se a imagem (subtrai-se cada pixel da distância da câmera) de maneira que cada pixel forneça a altura em relação ao chão do objeto por ele representado. Além disso pixels cujas distâncias não foram corretamente estimadas são marcados com um valor específico e o resultado é uma imagem cujos valores representam a altura dos objetos em milímetros. Essa informação é relevante pois permite obter candidatos a cabeças de maneira eficiente.

2.2 Obtenção de candidatos

O primeiro passo consiste em obter candidatos a pessoas, como a vista é superior, candidatos a cabeças. Em uma aplicação tradicional, com câmeras RGB esse processo dividiria a imagem em pequenos blocos de tamanhos que se espera para uma cabeça, sem qualquer informação sobre o posicionamento, isto é, varrendo uma “janela imaginária” pela imagem.

Para aprimorar esse método, assume-se a hipótese de que cabeças serão os objetos mais altos de uma vizinhança, se destacando na cena dos demais objetos. Embora ela não seja sempre verdadeira, ao se verificar que existem máquinas altas, é um método mais eficiente de selecionar candidatos do que utilizar uma janela varrendo todo o

quadro.

Aplica-se o método de máximos locais: a imagem é dividida em quadrados de áreas iguais. Para cada quadrado percorre-se os pixels e seleciona-se o maior deles, se este pixel for único no conjunto ele é considerado um ponto candidato à cabeça. Esse processo resulta num conjunto de pontos que determinam candidados à cabeças.

O tamanho dos quadrados que dividem a imagem é fundamental nessa etapa: se forem muito grandes há alta probabilidade de se perder cabeças visto que na cena existem máquinas altas. Por outro lado, se forem muito pequenos, existirão muitos candidatos e o processo se torna lento. Esse tamanho é calculado [11] utilizando a equação

$$s_w = \frac{f}{d} \cdot s_r \quad (2.1)$$

onde f é a distância focal da câmera, d a distância média entre a câmera e as pessoas e s_r o tamanho padrão de cabeças.

Em seguida, é necessário determinar um quadrado que delimite a cabeça de forma centralizada. Para obter o tamanho do quadrado, novamente utiliza-se a equação (2.1), porém dessa vez utilizando o valor do pixel encontrado para aquele candidato como altura. Tem-se então um quadrado de tamanho apropriado com o pixel máximo no centro, porém em poucos casos esse quadrado centraliza a cabeça.

Utiliza-se um processo de centralização baseado em *mean shift*, que permite iterativamente localizar o máximo de uma função probabilística dado uma máscara. Nesse caso, uma interpretação do procedimento é mover o quadrado mantendo suas dimensões fixas para o centróide dos pixels que estão em seu interior. Intuitivamente, percebe-se que os pixels de valor mais elevado terão maior peso, portanto, espera-se que o quadrado seja centralizado sob parte central da cabeça.

Atualiza-se o centro do retângulo

$$m(p) = \frac{\sum_{p_i \in S(p)} I(p_i) p_i}{\sum_{p_i \in S(p)} I(p_i)} \quad (2.2)$$

onde I é a imagem e $S(p)$ é o conjunto dos pixels no quadrado de centro p .

Ao final dessa etapa, conta-se com uma lista de candidatos representados pelo *patch* da imagem correspondente, como ilustrado na Figura

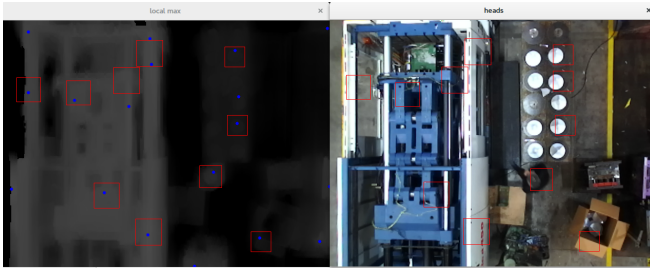


Figura 2.2: Pontos em azul são máximos locais e os quadrados vermelhos indicam os candidatos centralizados.

2.2 pelos quadrados em vermelho.

2.3 Descritores

Tendo a lista de candidatos é necessário obter uma descrição para cada um deles. No trabalho de referência [11] se destacam dois descritores: grades simples e grades circulares. Um terceiro descritor é proposto no presente trabalho para substituir o de grades circulares. Observa-se que para todos os descritores a característica importante de representação está nos desníveis da cabeça em relação ao resto do corpo, que se dá de maneira aproximadamente circular. Portanto, objetiva-se encontrar uma maneira de evidenciar quando e em que grau esse tipo de padrão ocorre.

O primeiro descritor, grades simples, divide o candidato em blocos, com uma quantidade ímpar em cada dimensão, como ilustrado na Figura 2.3. Para esse descritor em específico, fixou-se a quantidade de blocos em 7 para cada dimensão com 49 blocos no total, pois avaliou-se que para o tamanho dos candidatos esta foi a representação mais adequada. Então calcula-se a média dos pixels em cada um dos blocos, gerando uma matriz de médias 7×7 . Em seguida, subtrai-se dessa matriz o valor da média do bloco central. Por fim, gera-se um histograma da matriz resultante com número de intervalos igual a 32. Esse vetor de histograma é considerado o vetor de descrição (*feature vector*), cuja soma é 49.

Já o segundo método, grades circulares, apresentado em [11], propõe que uma série de blocos quadrados seja disposta circularmente sobre a

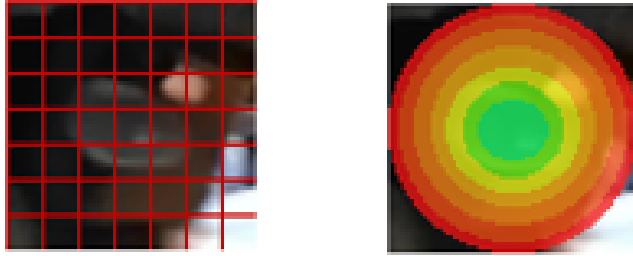


Figura 2.3: Ilustração dos descritores: grades simples e anéis concêntricos.

imagem candidata e que o mesmo procedimento do primeiro extrator seja realizado. O objetivo seria ter um descritor com maior invariância à rotação, dada a distribuição dos blocos. Porém, esse extrator é difícil de implementar devido à variação do tamanho dos candidatos.

Para facilitar a implementação, propomos, então, um descritor de anéis concêntricos. A ideia é similar às grades circulares, porém ao invés de se dispor blocos de maneira circular, utilizam-se coroas que partem desde o centro da imagem até as bordas. Primeiramente divide-se a imagem em n coroas circulares cujo centro coincide com o da imagem e cuja diferença entre os raios é constante, como visto na Figura 2.3. Então gera-se um vetor $v \in \mathbb{R}^n$ em que cada elemento corresponde à média dos pixels pertencentes à n -ésima coroa, desconsiderando os pixels inválidos. Em seguida subtrai-se desse vetor o valor da média da primeira coroa, cujo raio menor é 0 correspondendo então a um círculo. Por fim a descrição do candidato corresponde à diferenciação discreta (diferenças entre as componentes adjacentes) desse vetor, para evidenciar ainda mais as diferenças entre as alturas médias das coroas. Note que o vetor de descrição tem dimensão $d = n - 1$. Para esse descritor diferentes valores de n foram avaliados para identificar a melhor divisão, conforme descrito no Capítulo 4.

Após obter a descrição de cada candidato é possível classificá-los de maneira a validar que são cabeças de fato.

2.4 Classificação

Utiliza-se um classificador binário SVM (*Support Vector Machine*). Seja o dataset o conjunto (x_i, y_i) para $i = 1 \dots N$ com $x_i \in \mathbb{R}^d$ sendo o vetor de descrição do candidato i e $y_i \in \{-1, 1\}$ sendo a classe correspondente, não-cabeça ou cabeça, respectivamente. Deseja-se encontrar um classificador $f(x)$ tal que

$$\begin{cases} f(x_i) > 0, & \text{se } y_i = 1 \\ f(x_i) < 0, & \text{se } y_i = -1 \end{cases}$$

ou seja, $y_i f(x_i) > 0$ para uma classificação correta da amostra i .

A função de decisão do SVM pode ser descrita como $f(x) = w^T x + b$ onde $w \in \mathbb{R}^d$ é o vetor de pesos, que fornece a inclinação do hiperplano e $b \in \mathbb{R}$ um deslocamento do hiperplano em relação à origem.

No caso de um conjunto de treinamento linearmente separável, é possível que existam múltiplos hiperplanos que separem as classes corretamente. Nessa situação, pode-se escolher o hiperplano que oferece a maior margem, que em geral oferece melhor generalização. A margem é definida como a menor distância entre o hiperplano de separação e qualquer amostra.

Arbitrariamente, define-se que $f(x_+) = 1$ e $f(x_-) = -1$, onde x_+ e x_- são as amostras positiva e negativa mais próximas do hiperplano de separação, conforme Figura 2.4, portanto $y_i f(x_i) \geq 1$ para todo i . Com essa formulação obtém-se que a margem é dada por $\frac{2}{\|w\|}$ onde

$$\|w\| = \sqrt{w_1^2 + \dots + w_n^2}. \quad (2.3)$$

O aprendizado consiste em encontrar w e b que maximizem a a margem sujeitos à restrição $y_i(w^T x_i + b) \geq 1$. Esse é um problema de otimização convexa, em que qualquer mínimo local é também global. No entanto, o conjunto pode não ser linearmente separável, portanto a condição $y_i(w^T x_i + b) \geq 1$ não é atendida para todo i e existem erros de treinamento. Ou, mesmo sendo o conjunto linearmente separável, o hiperplano que separa corretamente todas as amostras pode não ser o que oferece maior generalização, como o conjunto apresentado na Figura 2.5.

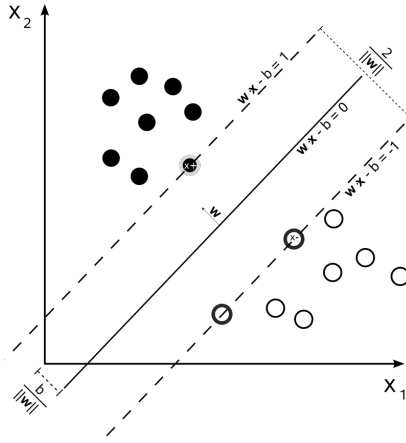


Figura 2.4: Hiperplano de separação e margem para conjuntos linearmente separáveis.

Dessa forma, para maior flexibilidade, minimiza-se a função custo [12]

$$\min P(w, b) = \frac{1}{2} \|w\|^2 + C \sum_i H_1(y_i f(x_i)), \quad (2.4)$$

onde a função

$$H_1(z) = \max(0, 1 - z)$$

é utilizada para dar um peso negativo quanto maior for o erro do classificador (quando $y_i f(x_i) < 0$).

O primeiro termo da equação (2.4) está associado à maximização da margem e o segundo à minimização dos erros de treinamento, quando uma amostra é classificada erroneamente. O parâmetro C controla quanto se penaliza os erros de classificação diante da maximização da margem, visto existir uma relação ideal entre eles.

O modelo de SVM apresentado até aqui é conhecido como linear. Mesmo com o método de penalização de erros de classificação, datasets complexos podem não ter um desempenho satisfatório. Dessa maneira, é introduzido um modelo não linear para o classificador: transforma-se as amostras x_i em um novo espaço obtendo amostras $\Phi(x_i)$ que são, no caso ideal, linearmente separáveis, como mostrado na Figura 2.6.

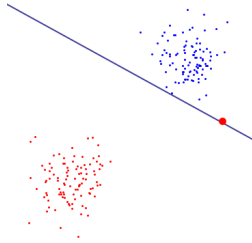


Figura 2.5: Exemplos de dataset em que separação perfeita não é ideal.

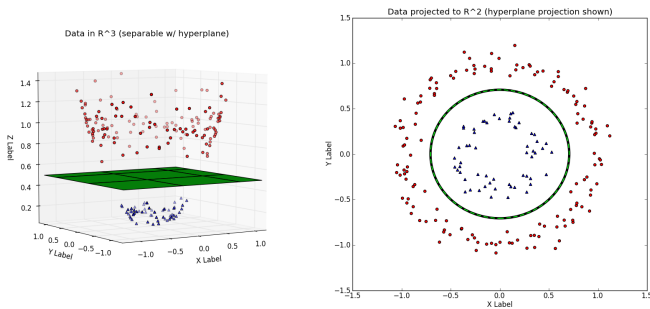


Figura 2.6: Exemplo de mapeamento para obter um classes linearmente separáveis. Espaço transformado e original, respectivamente.

Formalmente tem-se a função decisão modificada como

$$f(x) = w^T \Phi(x) + b.$$

Na prática, entretanto, não se calcula as coordenadas da amostra transformada no novo espaço. Ao invés disso utiliza-se o chamado *kernel trick* [12] em que o produto interno entre duas amostras transformadas é obtido diretamente através da avaliação da função $\Phi(x)$ sob as amostras originais.

O problema de minimizar $\|w\|$ sujeito à $y_i(w^T x_i + b) \geq 1$ pode ser visto como um problema de otimização com restrição. Nesse caso, pode-se utilizar o método de multiplicadores de Lagrange [12] obtendo que o vetor de pesos pode ser representado por

$$w = \sum_{i=1}^N \alpha_i \Phi(x_i) \quad (2.5)$$

para alguma variável α . Portanto, ao invés de minimizar w diretamente, podemos minimizar α e a função decisão se torna

$$f(x) = \sum_{i=1}^N \alpha_i \Phi(x_i) \Phi(x) + b. \quad (2.6)$$

Chamamos $K(x_i, x) = \Phi(x_i) \cdot \Phi(x)$ de *kernel* ou função núcleo. Um exemplo bastante utilizado é a *Radial Basis Function* (RBF), também conhecida como núcleo gaussiano, dada por

$$K(x, x_i) = \exp \left(-\frac{\|x - x_i\|^2}{2\sigma^2} \right). \quad (2.7)$$

Nota-se que quando $\|x - x_i\|$ é muito maior que σ a função retorna um valor que tende a zero, o que demonstra que a amostra em questão tem pouca importância para o hiperplano de decisão. Assim, quanto menor o valor de σ , mais local tende a ser o classificador, o que faz com que o desempenho no conjunto de treinamento seja excelente. Porém, ao mesmo tempo proporciona menor generalização do modelo, o que significa que esse desempenho não se replica no conjunto de teste. Portanto, é necessário escolher apropriadamente o parâmetro σ de forma a ter um

bom compromisso entre desempenho no treinamento e generalização.

2.5 Variação para saída probabilística

A formulação tradicional do SVM possui saída categórica, isto é, a classe à qual a amostra pertence. Entretanto, para uma classificação binária, é possível adicionar um componente que permite estimar uma saída probabilística, isto é, a probabilidade da amostra ser positiva.

A função de decisão (2.4) é proporcional à distância (com sinal) da amostra até o hiperplano de separação. Redimensionando essa medida através da função de Platt [13] é possível obter a probabilidade da amostra pertencer à determinada classe $P(y_i = 1|x_i)$. A função de Platt é simplesmente uma regressão logística, cujos parâmetros são encontrados por otimização após o treino do SVM tradicional. Permitindo, então, obter uma saída probabilística de um classificador SVM.

Essa formulação é interessante pois dá flexibilidade do rigor da classificação através da escolha do limiar de probabilidade a partir do qual uma amostra é considerada positiva após o modelo ter sido treinado.

2.6 Implementação

Para os algoritmos de processamento de imagens e extração de características, fez-se uso da biblioteca *OpenCV* que implementa grande quantidade de operações básicas em imagens e fornece um ambiente rico de trabalho para criar novas funcionalidades.

No que se refere a aprendizado de máquina, utilizou-se a biblioteca *Scikit-learn* [14] que implementa o algoritmo de treinamento SVM. Além de fornecer as amostras (neste caso, a descrição dos candidatos), é necessário configurar os parâmetros C , para penalidade dos erros de treinamento e o parâmetro σ do *kernel RBF*.

A abordagem utilizada foi o método de treinamento automático fornecido pela biblioteca: indica-se um conjunto de elementos para cada um dos parâmetros. Para cada escolha de C e σ a avaliação de desempenho é medida através de um processo de validação cruzada. Nesse processo, divide-se o conjunto de treinamento em 5 conjuntos. Numa próxima etapa, 4 desses conjuntos são utilizados para o treinamento e um para validação. Essa etapa se repete até cada um dos 5 conjun-

tos ter sido escolhido para validação. Ao término, faz-se a média do desempenho dos cinco testes, que é o resultado para uma escolha dos parâmetros.

Ao final do procedimento de treinamento automático as estatísticas de cada escolha dos parâmetros é exibida e a que tiver melhor desempenho, segundo métrica estabelecida, é escolhida. Tendo obtido esses parâmetros um novo processo de treinamento é efetuado, agora utilizando todo o conjunto de treinamento, sem validação.

Nota-se ainda que o uso dessa biblioteca possibilitou paralelizar o processo de treinamento através da utilização de todos os núcleos do processador, o que traz um impacto bastante positivo na performance temporal do algoritmo.

Tendo treinado o modelo é possível obter o resultado da classificação para cada candidato e remover os que não forem reconhecidos como cabeças. O resultado final do processo é um vetor de quadrados contendo todas as cabeças encontradas no quadro.

CAPÍTULO 3

Classificação utilizando aprendizado profundo

O Capítulo 2 tratou de uma solução baseada em seleção de candidatos, extração de características e posterior classificação, o que caracteriza uma técnica de aprendizado tradicional. Neste capítulo, propomos uma solução híbrida, que utiliza a seleção de candidatos idêntica à anterior, porém utiliza técnicas de aprendizado profundo para classificação, de modo que os candidatos são classificados sem pré-processamento. Remove-se o bloco de descrição dos candidatos e correspondentemente adiciona-se complexidade ao classificador. Inicialmente introduz-se as redes MLP e convolucionais, fornecendo as bases teóricas, funções de ativação e processo treinamento. Questões relativas à implementação também são abordadas.

3.1 Perceptron Multicamadas (MLP)

Segundo [7], uma rede perceptron multicamadas (MLP) também pode ser chamada de rede profunda sem realimentação (*deep feed-forward networks*) ou de rede neural. Essa estrutura consiste na organização de múltiplos nós organizados em camadas, como ilustra a Figura 1.4.

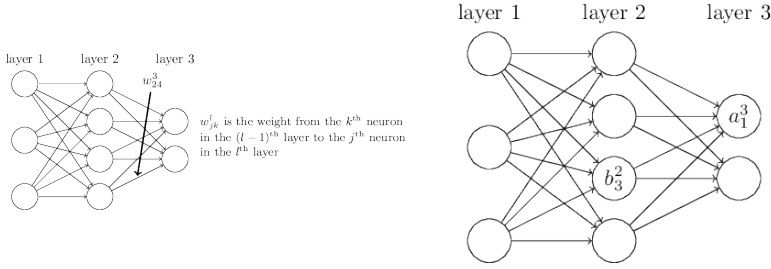


Figura 3.1: Convecção das conexões entre unidades.

Convencionam-se w_{jk}^l como o peso da conexão entre o nó j da camada l e o nó k da camada $(l-1)$, como ilustrado na Figura 3.1. De maneira semelhante, b_j^l e a_j^l indicam o *bias* e a ativação (saída) do nó j da camada l dados por

$$a_j^l = \phi(z_j^l) = \phi \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right) \quad (3.1)$$

onde $\phi(z)$ é a função de ativação da rede. Denota-se por θ o vetor de parâmetros da rede, composto por todos os parâmetros de pesos w e de *bias* b da rede.

Para o problema de classificação, pode-se utilizar uma rede neural de L camadas: a camada de entrada recebendo a amostra x sem pré-processamento, camadas intermediárias responsáveis pela descrição da amostra e a camada final, que no caso de classificação binária possui uma única unidade, cuja saída a_0^L é a probabilidade da amostra ser positiva. Pode-se denotar, portanto, a saída da rede como a função

$$f(x, \theta) = a_0^L(x, \theta). \quad (3.2)$$

Os hiper-parâmetros da rede consistem no número de camadas, e na quantidade de unidades em cada camada que se deseja utilizar. Para problemas complexos, incluindo imagens de alta resolução com múltiplas classes de objetos pode-se ter mais de vinte camadas intermediárias. Entretanto, para o presente problema, por se tratar de amostras de baixa resolução e reduzida complexidade, utilizam-se apenas duas camadas intermediárias. Inclue-se, ainda nos hiper-parâmetros, a fun-

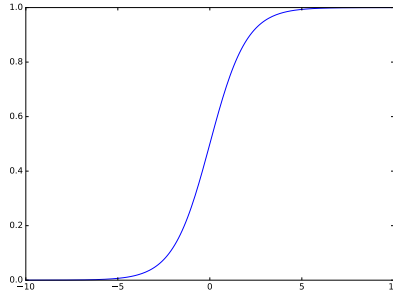


Figura 3.2: Gráfico mostrando o comportamento da sigmoide.

ção de ativação $\phi(z)$ que se escolhe para cada camada da rede.

3.2 Função de ativação

Uma função de ativação $\phi(z)$ é responsável por introduzir não-linearidades na rede, um requisito fundamental para ter bom resultado com conjuntos de dados mais complexos. A escolha dessa função está associada a diversos fatores. Um deles é o desempenho da rede, especialmente durante treinamento. Outro fator é a interpretação que se dá à saída de uma unidade. Duas funções, escolhidas segundo um desses objetivos, são utilizadas nesse trabalho e são introduzidas a seguir.

3.2.1 Sigmóide

Para classificadores, o valor das unidades da camada de saída podem ser interpretados como probabilidades da amostra pertencer a uma determinada classe. A escolha da função de ativação, nesse caso, se dá justamente pela interpretação da camada de saída. Nesse caso espera-se que os valores estejam no intervalo $[0, 1]$. Uma função que assegura essa condição é a sigmoide, ilustrada na Figura 3.2 e dada por

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \quad (3.3)$$

Historicamente essa função foi utilizada para as camadas intermediárias. No entanto, observa-se que tal função satura facilmente com entradas próximas de 1 ou 0, ocasionando uma derivada praticamente

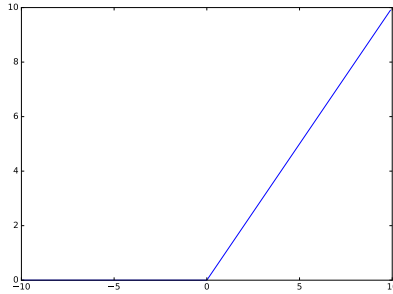


Figura 3.3: Gráfico mostrando o comportamento da função RELU.

nula nessas regiões. Como destacado na Subseção 3.3.3, isso é um problema ao se considerar que a derivada dessa função definirá a atualização dos pesos desses nós durante o processo de treinamento. Ao propagar esse processo camada a camada, verificam-se derivadas cada vez menores, o que causa o problema do gradiente enfraquecido, em que as camadas iniciais não conseguem aprender pois o gradiente é muito reduzido.

3.2.2 RELU

Para resolver o problema do gradiente enfraquecido, obtendo uma melhora no desempenho da rede, a função RELU (*Rectified Linear Unit*), ilustrada na Figura 3.3, foi proposta [8] como

$$\text{RELU}(z) = \max\{0, z\}. \quad (3.4)$$

Observa-se que para valores positivos essa função tem uma derivada constante e igual a um, proporcionando um gradiente considerável sempre que o nó estiver ativo. Por outro lado, para valores negativos considera-se o nó inativo e os pesos não são atualizados. Essas características garantem boa propagação do gradiente, portanto essa função de ativação é usualmente utilizada para as camadas intermediárias da rede, mantendo, ainda, a ativação sigmóide para a camada de saída.

3.3 Treinamento

Uma maneira de avaliar o desempenho de uma rede é através da avaliação da chamada função custo. O processo de treinamento de uma rede neural é justamente o problema de otimização dessa função. Porém, diferentemente do caso do SVM, trata-se de uma otimização não convexa, isto é, existem múltiplos mínimos locais para essa função e diversas soluções podem ser encontradas, dependendo dos valores iniciais dos parâmetros. Esse fato advém do caráter não linear das funções de ativação nas diferentes camadas da rede.

Durante o treinamento otimiza-se os parâmetros θ de forma que $y_i \approx f(x_i, \theta)$ para o conjunto de treinamento $\{(x_i, y_i) | i = 1, \dots, N\}$. Isso é feito através da minimização de uma função custo $C(\theta)$ por um algoritmo de otimização chamado SGD (*Stochastic Gradient Descent*) utilizando o gradiente de C obtido por um algoritmo de propagação reversa de erros. Todos esses passos são descritos a seguir.

3.3.1 Função custo

A função custo quantifica a qualidade de representação que o modelo oferece para o conjunto de treinamento, isto é, qual a medida de erro que o classificador comete dentro do próprio conjunto de treinamento.

Em geral pode-se escrever a função custo como

$$C(\theta) = \sum_i^N C_i(\theta) \quad (3.5)$$

de forma que se pode calcular o custo de cada amostra de treinamento individualmente, correspondente ao erro equivalente àquela amostra.

Introduz-se a função custo de entropia cruzada binária (*binary cross entropy*) [12], utilizada no presente trabalho e descrita por

$$C(\theta) = -\frac{1}{N} \sum_i^N [y_i \ln f(x_i, \theta) + (1 - y_i) \ln(1 - f(x_i, \theta))] \quad (3.6)$$

cuja grande contribuição é permitir o uso de unidades sigmoidais na camada de saída (mesmo utilizando ativação sigmóide nas camadas intermediárias) sem enfraquecer o gradiente, visto que a função exponencial da ativação é revertida ao se utilizar o logaritmo natural imposto pela

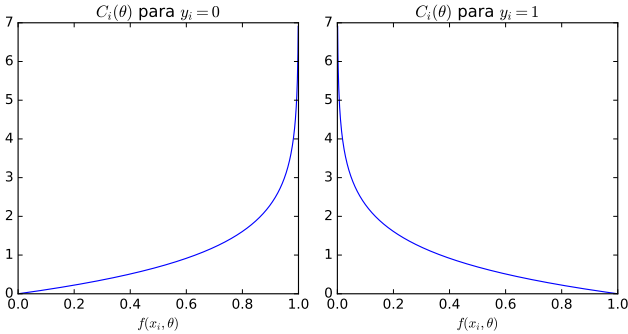


Figura 3.4: Comportamento da função custo entropia cruzada para amostras negativas (esquerda) e positivas (direita).

entropia cruzada. Nota-se ainda, no caso em que $f(x, \theta) = y \in \{0, 1\}$ tem-se um custo nulo dessa função.

É interessante notar, ainda, como essa função penaliza amostras cuja saída da rede $f(x_i, \theta)$ difere do valor especificado no treinamento y_i conforme ilustra a Figura 3.4.

Tendo uma métrica para avaliar a qualidade do classificador em relação ao conjunto de treinamento utiliza-se um algoritmo de otimização para encontrar parâmetros que melhoram o desempenho desse mecanismo.

3.3.2 Stochastic Gradient Descent

Um método tradicional de minimização é o do gradiente descendente, que utiliza a aproximação

$$\Delta C \approx \nabla C \cdot \Delta \theta \quad (3.7)$$

para verificar a variação ΔC na função custo em função do gradiente ∇C e na variação nos parâmetros θ . Basicamente escolhe-se os parâmetros inicialmente aleatórios e então busca-se encontrar o vetor de variação dos parâmetros $\Delta \theta$ que faça C diminuir. Ao escolher $\Delta \theta = -\eta \nabla C$, tem-se

$$\Delta C = -\eta |\nabla C|^2 \quad (3.8)$$

que será negativo ao escolher $\eta > 0$, chamada de taxa de aprendizado (*learning rate*). Esse processo é executado iterativamente, atualizando os parâmetros com $\theta \leftarrow \theta + \Delta\theta$ até se encontrar um mínimo local.

Em geral, redes profundas necessitam de um grande número de amostras para serem treinadas. O cálculo do gradiente ∇C é custoso ao se considerar a quantidade amostras, visto que $\nabla C = \sum_i^N \nabla C_i$. Utiliza-se, então, uma variação do método chamada de *Stochastic Gradient Descent* – SGD. Essa variação particiona o conjunto de treinamento aleatoriamente em M subconjuntos, chamados *mini batches*. Os índices das amostras do m -ésimo mini batch compõem o conjunto de índices I_m , de maneira que $I_1, I_2, \dots \subseteq \{1, \dots, N\}$. Então, estima-se o gradiente total fazendo a soma dos gradientes individuais como

$$\nabla C \approx \sum_{i \in I_m} \nabla C_i \quad (3.9)$$

ou seja, avaliando o gradiente apenas nas amostras percentences ao m -ésimo *mini batch* a cada iteração do método, o que reduz drasticamente o tempo de execução do algoritmo.

Nota-se ainda que a nomenclatura *Stochastic Gradient Descent* se refere ao caso particular quando $M = 1$, porém será utilizada nesse trabalho mesmo para valores de M maiores que um.

3.3.3 Backpropagation

O método de otimização visto na Seção 3.3.2 faz uso do gradiente de função custo ∇C , porém não mostra como calcular essa quantidade. Para isso é necessário encontrar as derivadas da função custo em relação aos parâmetros individuais de cada elemento da rede, w e b . De fato, utilizar as ferramentas do cálculo se mostra um desafio na prática, visto a complexidade da função custo em termos da função da rede $f(x, \theta)$ que combina todas as não linearidades encadeadas nas camadas.

Para contornar esse problema, uma solução foi o algoritmo de propagação reversa de erros (*backpropagation*), já existente na década de 1979 porém proposto mais recentemente em [6]. Esse algoritmo fornece um método eficiente de calcular as derivadas da função custo em relação a quaisquer parâmetros w e b da rede.

Primeiramente define-se uma grandeza intermediária chamada de erro da camada l , δ^l , interpretada como a contribuição da camada l

para o erro total, através do qual serão calculadas as grandezas de interesse.

Esse algoritmo possui uma formulação complexa que não está no escopo deste trabalho. Entretanto, alguns aspectos serão enfatizados pela análise simples das equações fornecidas por esse método, disponíveis com demonstração em [15] e reproduzidas a seguir:

$$\delta^L = (a^L - y) \odot \phi'(z^L) \quad (3.10)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \phi'(z^l) \quad (3.11)$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (3.12)$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (3.13)$$

onde \odot é o operador de multiplicação elemento por elemento.

Pode-se entender o algoritmo como calculando o erro na camada de saída, δ^L com (3.10) (simplificada para um caso específico da função custo) e propagando esse erro para as outras camadas, com a contribuição individual de cada unidade, obtendo os correspondentes δ^l conforme (3.11). O erro das camadas intermediárias permite obter as demais derivadas através de (3.12) e (3.13), e finalmente, obter o gradiente da função custo.

Um aspecto a destacar é o papel fundamental da derivada da função de ativação $\phi(z)$ em (3.11). Se ela saturar, essa derivada terá um valor baixo, o que provoca um gradiente pequeno e a atualização dos parâmetros fica estagnada, impedindo o aprendizado. Por isso a importância de funções de ativação que impedem a saturação, como a RELU, vista anteriormente.

Nota-se ainda que a equação (3.13) mostra que a derivada da função custo em relação ao pesos de um determinado nó é proporcional à entrada correspondente ao nó de entrada. Portanto, caso a entrada seja nula, essa componente do gradiente será também nula, o que significa que o nó só tem seu peso atualizado quando estiver ativo.

3.4 Redes convolucionais

A estrutura MLP vista anteriormente é chamada de completamente conectada, onde cada unidade em uma camada está conectada à todas as demais da próxima camada. Uma variante desse modelo são as

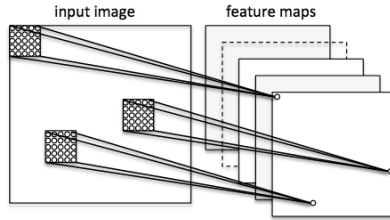


Figura 3.5: Exemplo de rede convolucional e campos receptivos.

chamadas redes convolucionais, que introduzem um aspecto espacial no processamento das amostras de forma que as camadas possuam uma estrutura de conexão local. São especialmente úteis para amostras de maior dimensão que têm uma estrutura matricial, como imagens, e se destacam em relação à MLP pois a relação de estrutura não precisa ser incorporada ao modelo via treinamento mas já advém diretamente do modelo utilizado.

Apesar das redes convolucionais já existirem na década de 1970, considera-se que [16] foi fundamental para estabelecer a teoria moderna dessas redes. Tais redes continuam organizadas em camadas, porém não são completamente conectadas. Conectam-se as camadas através dos chamados campos receptivos locais (*local receptive fields*) que ligam uma série de nós da camada anterior à próxima, como ilustra a Figura 3.5. Os campos receptivos são transladados para obter as conexões dos próximos nós na camada intermediária.

Além do aspecto estrutural introduzido pelos campos receptivos, tem-se que os pesos e *bias* são compartilhados pelos diferentes campos receptivos. Isso é, para cada nó da camada intermediária embora as ligações ocorram com diferentes nós da camada anterior, elas compartilham os mesmo pesos e isso reduz drasticamente a quantidade de parâmetros da rede, tornando o processo de treinamento mais rápido. Portanto, tem-se que os diferentes campos receptivos são sensíveis à um mesmo padrão ou característica (*feature*) e isso garante uma invariância à translação. Os pesos compartilhados são também chamados de núcleo (*kernel*). De fato, observa-se que essa operação é resumida como uma convolução entre a camada anterior e o núcleo:

$$a^1 = \phi(w * a^0 + b) \quad (3.14)$$

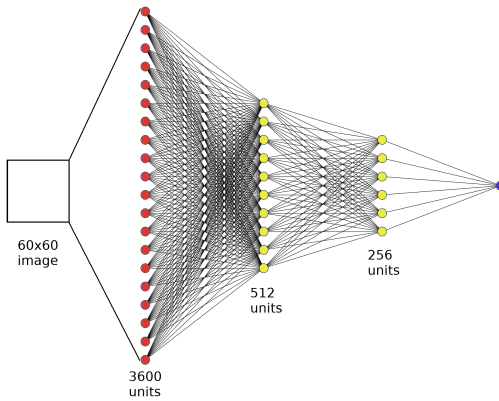


Figura 3.6: Estrutura proposta para rede MLP.

onde a^0 é a camada anterior e a^1 o primeiro mapa de características, sendo $\phi(z)$ a função de ativação e w o núcleo.

Na prática, entretanto, utilizam-se múltiplos núcleos de maneira a obter um mapa de características mais rico. Isso significa aumentar drasticamente a dimensionalidade da rede e implica necessidade de utilizar camadas de *pooling* que reduzam a dimensionalidade da rede através de uma amostragem. Uma das possibilidades é utilizar *max-pooling*, uma técnica de amostragem não linear. Esse método divide o mapa em blocos de tamanhos iguais especificados e, para cada bloco, seleciona o maior valor, descartando os demais e portanto reduzindo dimensionalidade. Basicamente condensa-se a informação sobre a característica que se deseja encontrar reduzindo o impacto da sua localização exata.

Por fim, após camadas convolucionais e de amostragem, tem-se mapas de características densos, que podem em seguida serem utilizados por uma camada densamente conectada para realizar o processo final de classificação, como ilustrado na Figura 1.5.

O processo de treinamento é similar ao das MLPs, porém com pequenas modificações dado o caráter estrutural e convolucional da rede.

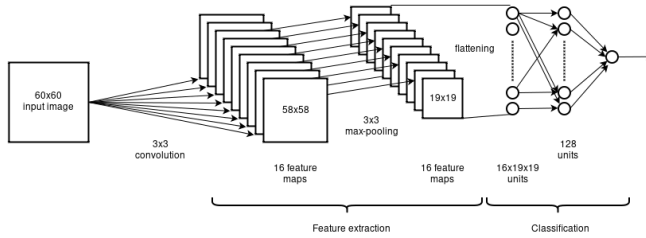


Figura 3.7: Estrutura proposta para rede CNN.

3.5 Implementação

A solução híbrida proposta neste capítulo assume que os candidatos são obtidos através do algoritmo tradicional visto na Seção 2.2 e utiliza um classificador profundo sem que haja pré-processamento das amostras. Nesse contexto, dois tipos de estruturas foram utilizados, primeiramente uma rede MLP e em seguida uma rede convolucional, como ilustram as Figuras 3.6 e 3.7, respectivamente.

Para ambas as estruturas é necessário especificar qual a topologia da rede: número de camadas e de nós em cada camada, tamanho dos filtros convolutivos e de amostragem, funções de ativação e método de inicialização de pesos e de otimização. A seleção desses parâmetros pressupõe o entendimento do funcionamento da rede e um ajuste fino baseado em experimentos empíricos pode melhorar o desempenho do classificador.

Existem diversas plataformas para se implementar redes neurais. Entre essas, destacam-se a biblioteca de código aberto THEANO [17], que permite a definição e avaliação de expressões matemáticas multidimensionais, diferenciação simbólica, otimização para velocidade e estabilidade, além de uso transparente da GPU. Essa biblioteca, entretanto, não fornece funções específicas para a criação de redes neurais, ou seja, é necessário que o usuário crie as matrizes de pesos, defina função custo e de ativação e implemente o algoritmo de propagação reversa.

Para facilitar a utilização das técnicas de aprendizado profundo e tornar o processo de desenvolvimento mais rápido, utilizou-se a biblioteca KERAS [18], que fornece uma abstração acima do THEANO. Isso permite criar redes com poucas linhas de código, definindo camadas de

maneira simples e escolhendo as funções de ativação e custo dentre as já definidas. Ainda assim, permite flexibilidade para o usuário criar suas próprias funções e ajustar parâmetros da rede, assim como processos de inicialização dos parâmetros e diferentes tipos de otimizadores.

Ambas as bibliotecas são desenvolvidas para PYTHON e portanto houve a necessidade de integrar esse código ao já existente em C++. Destaca-se a dificuldade em integrar o funcionamento paralelo dos algoritmos sob a plataforma CUDA. Isso se deve ao fato do código em C++ referente ao processamento das imagens de profundidade e ao THEANO utilizarem simultaneamente os recursos da GPU. Esse conflito foi resolvido utilizando um sistema de troca de contextos disponível na plataforma CUDA.

Outro problema enfrentado ocorreu durante a etapa de treinamento. O conjunto de dados utilizado é fortemente desbalanceado, onde mais de 89% das amostras pertence à classe negativa. Após a primeira etapa de treinamento observou-se uma acurácia de aproximadamente 89%, o que era aparentemente um ótimo resultado. Entretanto faltou a consideração da distribuição desses resultados: o classificador indicava todas as amostras como sendo negativas, pois o conjunto de dados é altamente enviesado. Para corrigir esse problema listou-se duas opções: equilibrar o conjunto de dados ou modificar a função custo para ponderar os pesos para as classes. Verificou-se que a distribuição de classes pôde ser artificialmente equilibrada replicando as amostras positivas até seu número se equiparar ao das negativas, o que de fato demonstrou um resultado positivo na prática.

Encontrou-se também uma dificuldade para convergência do método utilizando apenas o otimizador SGD. A solução foi utilizar uma variante chamada Adam [19] cuja formulação inclui considerações sobre momento, que consiste em adaptativamente corrigir a taxa de aprendizado η , o que impacta positivamente sobre a convergência da otimização.

CAPÍTULO 4

Resultados

Neste capítulo, os resultados das diferentes soluções propostas no trabalho são apresentados. Primeiramente introduzem-se as medidas de avaliação que serão utilizadas. Em seguida compara-se as etapas de classificação do método tradicional e híbrido, cujo processo de seleção de candidatos é idêntico, portanto a diferença em desempenho se dá apenas no estágio de classificação. Por fim avalia-se o desempenho do sistema completo das diferentes soluções apresentadas.

4.1 Medidas de avaliação

Para avaliar o desempenho das diferentes soluções apresentadas neste trabalho, faz-se necessário a utilização de medidas que permitam comparar os resultados objetivamente e indicar facilmente os aspectos positivos e negativos relevantes de cada método. Neste trabalho tratamos de classificação binária, em que amostras são positivas (representando cabeças e também indicadas por 1) ou negativas (representando não-cabeças e também indicadas por 0).

Em geral, em um problema de classificação, o indicador mais comum é a acurácia (*accuracy*), definida [20] como a razão das classificações

Tabela 4.1: Matriz de confusão

		Classificado	
		0	1
Real	0	TN	FP
	1	FN	TP

corretas pelo total de classificações. É fundamental notar, entretanto, que essa medida sozinha não é suficiente para avaliar um classificador, especialmente se o dataset for desbalanceado. Por exemplo, ao considerar um dataset com duas amostras negativas entre 20 e utilizar um classificador trivial que classifica qualquer amostra como sendo positiva, obtém-se uma acurácia de 90%, demonstrando o problema dessa métrica.

Uma maneira compacta de representar o desempenho de um classificador é a **matriz de confusão** [20], apresentada na tabela 4.1. Para uma classificação binária é uma matriz 2x2 em que a primeira linha indica as amostras negativas e a segunda as amostras positivas. De forma semelhante, a primeira coluna indica amostras que foram classificadas como negativas e a segunda que foram classificadas como positivas. Combinando-se essas linhas e colunas, obtém-se o número de verdadeiros negativos TN, falsos positivos FP, falsos negativos FN e verdadeiros positivos TP, respectivamente. Essa estrutura supera os problemas da acurácia pois fornece a distribuição de acertos e erros de classificação. Adota-se a descrição dos valores em percentuais relativos ao conjunto de análise.

Outro aspecto a destacar na avaliação de classificadores é que não basta avaliar os indicadores apresentados apenas no conjunto de treinamento. Embora essas medidas forneçam um limite máximo para o desempenho do classificador e permitam detectar *underfitting*, quando o modelo não tem capacidade de modelar o conjunto de treinamento, não se pode avaliar o desempenho do classificador em um caso geral, isto é, não incluso no conjunto de treinamento. Por outro lado, se o modelo for demasiadamente complexo, ele pode ter se tornado muito específico para algumas características do conjunto de treinamento, situação conhecida como *overfitting*. Assim, para medir o desempenho do classificador em uma situação real, utiliza-se outro conjunto de dados

supervisionados chamado de conjunto de teste (*test set*), independente do conjunto de treinamento. Afirma-se que o modelo tem boa generalização bem se o desempenho no conjunto de treinamento se equiparar ao de teste.

Graficamente pode-se avaliar classificadores através do gráfico ROC (*Receiver Operating Characteristic*) [20], que relaciona a taxa de verdadeiros positivos

$$TVP = \frac{TP}{FN + TP} \quad (4.1)$$

com a taxa de falsos positivos

$$TFP = \frac{FP}{TN + FP}. \quad (4.2)$$

Dessa forma, o classificador ideal indicaria uma TVP unitária com TFP nula, o que caracteriza o ponto (0, 1).

A princípio cada classificador avaliado em um conjunto de dados corresponde a um ponto no espaço ROC. Entretanto existem classificadores cuja saída é a probabilidade da amostra ser positiva. Nesses casos é possível escolher um limiar de probabilidade T acima do qual se considera uma amostra como positiva. Cada escolha desse limiar corresponderá a um ponto no espaço ROC e variando esse parâmetro é possível obter uma curva nesse espaço. Para facilitar a distinção entre diferentes classificadores um subconjunto de pontos dessa curva são destacados com marcadores.

Comparar diferentes classificadores não é trivial pois é uma tarefa multiobjetivo que envolve simultaneamente maximizar a TVP e minimizar TFP. Portanto só é possível afirmar que um classificador é melhor do que outro se sua TVP é maior e sua TFP é menor que a do outro, nesse caso diz-se que o primeiro domina o segundo. Na prática, entretanto, pode-se utilizar a métrica *Area Under Curve*, AUC, que mede a área sob a curva ROC, para facilitar essa comparação, especialmente quando a região de operação não é especificada.

Um requerimento específico para a aplicação em questão é que a linha de produção se mantenha em funcionamento o maior tempo possível, para tanto deve-se minimizar a quantidade de falsos positivos FP, que significam cabeças incorretamente identificadas que interrompam a produção sem necessidade. Assim, demanda-se uma TFP menor que

10%. Quando especificada, a AUC será correspondente à área sob a curva ROC com TFP inferior à esse limite.

4.2 Desempenho dos Classificadores

O classificador SVM originalmente possui saída categórica, porém pode ser adaptado para fornecer uma saída probabilística que indica a probabilidade da amostra ser positiva conforme visto na Seção 2.5. Nota-se entretanto que internamente a formulação é ainda categórica. Já os métodos profundos possuem saídas probabilísticas por definição. Essa estrutura da saída como probabilidade é vantajosa pois permite ajustar o compromisso entre falso positivos e falso negativos do classificador, após treinamento, através da escolha do limiar de probabilidade T acima do qual uma amostra é considerada positiva. Escolher um limiar mais alto significa exigir mais segurança do classificador e, portanto, reduzir o número de falsos positivos ao preço de aumentar o número de falsos negativos.

No caso de saídas probabilísticas, para representar a matriz de confusão se faz necessário escolher valores para T . Nesse trabalho escolhem-se dois valores (0.5 e 0.9) para esse parâmetro de maneira a demonstrar sua influência sob a distribuição das amostras na classificação. Nota-se que a estrutura de classificação é a mesma, apenas o limiar é trocado.

Para realizar o treinamento dos classificadores foi gerado um conjunto de dados a partir de gravações realizadas no ambiente industrial onde o sistema deverá ser instalado. A câmera foi posicionada num ponto próximo de onde deve ocorrer a sua instalação final. Diversos vídeos foram gravados e posteriormente analisados. Utilizando o algoritmo de detecção de candidatos visto na Seção 2.2, selecionou-se manualmente em cada quadro quais dos candidatos eram verdadeiramente cabeças.

Repetindo esse processo em cada quadro formou-se um conjunto de treinamento, cuja amostra é o recorte do candidato, reduzido de 2459 amostras negativas (não cabeças) e 667 positivas (cabeças). Em seguida estendeu-se esse conjunto reduzido para 9894 amostras negativas e 1222 positivas, formando o conjunto de treinamento completo. Para averiguar o desempenho estendeu-se esse conjunto para 16898 amostras

Tabela 4.2: Anéis concêntricos, 8 dimensões, conjunto de treinamento reduzido

	0	1
0	76.357	2.099
1	5.975	15.568

Tabela 4.3: Anéis concêntricos, 16 dimensões, conjunto de treinamento reduzido

	0	1
0	77.035	1.421
1	3.585	17.959

Tabela 4.4: Anéis concêntricos, 18 dimensões, conjunto de treinamento reduzido

	0	1
0	77.842	0.614
1	1.938	19.606

das quais 14966 são negativas. Criou-se, ainda, um conjunto de teste independente de 2738 amostras negativas e 235 amostras positivas.

4.2.1 SVM

Avaliou-se o resultado dos diferentes descritores mencionados em 2.3 utilizando um processo de treinamento automático do SVM [14]: varreu-se o espaço dos parâmetros C entre 0.0001 e 0.01 com passos de 0.001 e σ entre 1 e 100 com passos de 5. Selecionou-se os melhores resultados através da medida conhecida como precisão (*precision*) definida como

$$P = \frac{TP}{FP + TP} \quad (4.3)$$

comumente utilizada e disponível no *toolkit* utilizado.

Primeiramente avaliou-se qual a dimensão do descritor de anéis concêntricos que apresenta melhor resultado. Para tanto treinou-se classificadores SVM de saída probabilística com as descrições em 8, 16 e 18 dimensões sob o conjunto de treinamento reduzido, obtendo as matrizes de confusão das tabelas 4.2, 4.3 e 4.4. Obtiveram-se as curvas ROC desses classificadores avaliados no conjunto de teste, vistas na Figura 4.1.

Tendo visto que o classificador de 18 dimensões domina sob os ou-

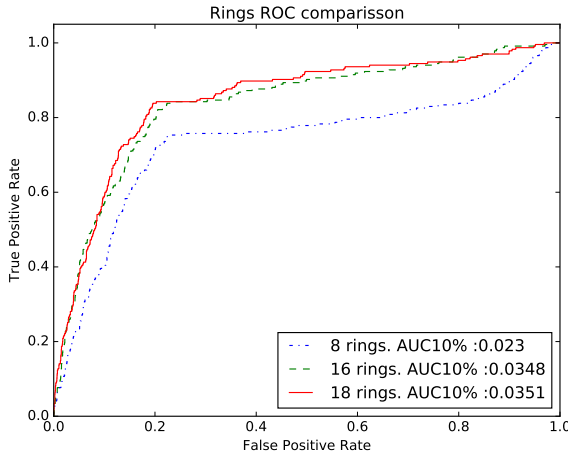


Figura 4.1: Curvas ROC comparando classificadores SVM sob descritores de anéis com diferentes dimensões.

Tabela 4.5: Anéis concêntricos com 18 dimensões, conjunto treinamento

	0	1
0	88.971	0.036
1	0.711	10.282

Tabela 4.6: Anéis concêntricos com 18 dimensões, conjunto teste

	0	1
0	86.243	5.853
1	5.314	2.590

tros na região de operação com menor TFP, decidiu-se selecionar esse descritor para seguir com a análise. Para tanto utilizou-se o conjunto de treinamento completo para o treinamento e posterior avaliação através do conjunto de teste, como visto nas Tabelas 4.5 e 4.6.

Outro descritor que também foi avaliado foi o de grades simples com dimensão 7x7, como demonstrado nas Tabelas 4.7 e 4.8.

4.2.2 MLP

Na estrutura MLP foram realizados testes com duas e três camadas intermediárias, com ativação RELU e um único perceptron de saída com ativação sigmoide indicando a probabilidade da amostra ser uma cabeça. Na configuração de duas camadas foram utilizados 512 e 256

Tabela 4.7: Grades simples, conjunto treinamento

	0	1
0	88.890	0.117
1	0.324	10.669

Tabela 4.8: Grades simples, conjunto teste

	0	1
0	82.240	9.855
1	6.088	1.816

Tabela 4.9: MLP, conjunto treinamento extendido $T = 0.5$

	0	1
0	86.004	2.562
1	0.615	10.818

Tabela 4.10: MLP, conjunto treinamento extendido $T = 0.9$

	0	1
0	87.549	1.018
1	1.491	9.942

Tabela 4.11: MLP, conjunto teste $T = 0.5$

	0	1
0	83.720	8.375
1	0.336	7.568

Tabela 4.12: MLP, conjunto teste $T = 0.9$

	0	1
0	88.631	3.465
1	0.841	7.064

perceptrons, respectivamente, conforme ilustra a Figura 3.6. Os resultados obtidos para o conjunto de treinamento são apresentados nas Tabelas 4.9 e 4.10. O resultado do conjunto de teste são apresentados nas Tabela 4.11 e 4.12.

Para três camadas foram utilizados 1024, 512 e 256 perceptrons respectivamente, entretanto os resultados foram semelhantes aos obtidos com duas camadas, portanto o aumento em complexidade não é justificado visto que o desempenho permaneceu o mesmo.

4.2.3 CNN

Para redes convolucionais utilizou-se uma única camada convolucional RELU com 16 núcleos 3x3 seguida por max-pooling (3x3) e uma camada densamente conectada de 128 perceptrons RELU e finalmente saída única com ativação sigmóide. Essa topologia, ilustrada na Figura 3.7, foi a que teve o melhor desempenho, obtendo 99% de acurácia no

Tabela 4.13: CNN, $T = 0.5$, conjunto treinamento estendido

	0	1
0	88.377	0.189
1	0.136	11.297

Tabela 4.14: CNN, $T = 0.9$, conjunto treinamento estendido

	0	1
0	88.472	0.095
1	0.669	10.765

Tabela 4.15: CNN, $T = 0.5$, conjunto de teste

	0	1
0	88.643	3.094
1	0.563	7.700

Tabela 4.16: CNN, $T = 0.9$, conjunto de teste

	0	1
0	90.515	1.581
1	1.144	6.761

conjunto de treinamento, demonstrados nas Tabelas 4.13 e 4.14. Também teve 96% de acurácia no conjunto de teste, como observado nas Tabelas 4.15 e 4.16.

4.2.4 Discussão

Verifica-se as curvas ROC dos classificadores correspondentes avaliados no conjunto de teste na Figura 4.3. Observando a região de interesse (TFP menor que 10%) fica claro que os métodos profundos, com CNN e MLP, dominam os superficiais. Em mais detalhes, MLP domina entre TFP de 0.1% a 1.3%, a partir de onde CNN passa a dominar. Para especificar qual o melhor classificador entre os profundos é necessário saber exatamente a região de operação em que se deseja atuar.

Observa-se que muitos classificadores obtiveram um desempenho elevado no conjunto de treinamento, em especial as redes convolucionais que obtiveram acurácia quase unitária (sob dataset balanceado) conforme tabelas 4.13 e 4.13, demonstrando a alta capacidade do modelo. Nota-se, entretanto, que há grande discrepância entre o desempenho dos classificadores no conjunto de treinamento e de teste nos métodos superficiais, o que caracteriza *overfitting*, como apontam as tabelas 4.5 e 4.6.

Para esclarecer a presença de *overfitting*, utiliza-se uma curva de validação da Figura 4.2, que mostra como o desempenho do conjunto

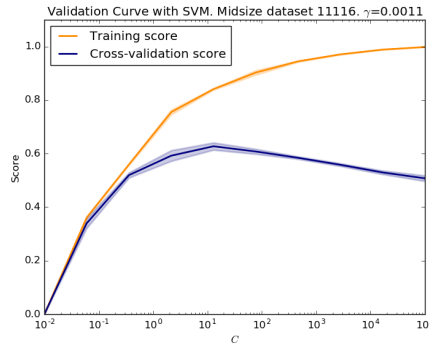


Figura 4.2: Curva de validação SVM com descritor de anéis 18dim.

de treinamento e de validação (utilizado para variar hiper-parâmetros do modelo) se relaciona com algum parâmetro do modelo, nesse caso, C , que modela o compromisso de maximização da margem versus erros do treinamento, também entendida como capacidade do modelo. Verifica-se que para C acima de 10 o modelo começa a se tornar muito específico para as características do conjunto de treinamento, perdendo capacidade de generalização, e a performance do conjunto de validação passa a decrescer, configurando *overfitting*. Especificamente nesse caso conclui-se que o maior problema está no extrator de características que não consegue discriminar as amostras suficientemente.

Os métodos profundos, em contraste, apresentam uma boa generalização, como pode ser visto na tabela 4.16. Isso é explicado porque o modelo em questão foi corretamente especificado através no número de camadas e tamanho de filtros, o que possibilita que a melhor representação (extração de características) para as amostras seja encontrada automaticamente.

4.3 Sistema completo

Na seção 4.2 assumiu-se que os candidatos já haviam sido extraídos de forma a comparar apenas os classificadores. Nesta seção serão comparados os resultados do sistema completo, incluindo a extração de candidatos, vista na seção 2.2, e posterior classificação via MLP e CNN, que demonstram os melhores resultados na seção 4.2. Como

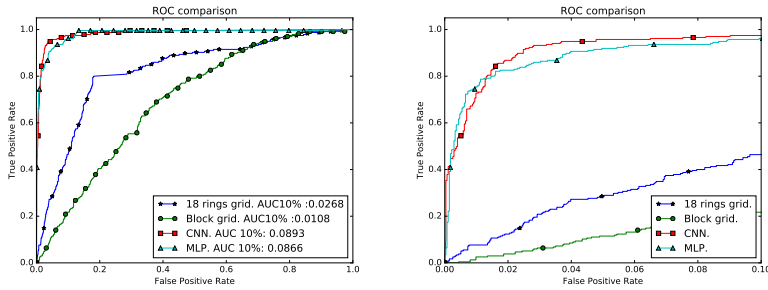


Figura 4.3: Curvas ROC comparando diferentes classificadores apresentados e zoom na região de interesse

agora é possível que candidatos deixem de ser selecionados, o sistema está limitado em performance pelos classificadores.

O sistema toma como entrada um quadro e a saída é binária detectando ou não a presença de pessoas nesse quadro. Isto é, a saída do sistema y deve ser 1 se houver pelo menos uma cabeça e 0 caso contrário.

Após selecionar os candidatos, cada um deles é introduzido no classificador e recebe uma probabilidade correspondente à amostra ser cabeça. É necessário agora converter esse conjunto de probabilidades em uma única probabilidade correspondente ao quadro conter uma ou mais cabeças. Uma possível abordagem é assumir que as classificações dos candidatos são variáveis independentes. Nesse caso a probabilidade de o quadro conter cabeças pode ser calculada como

$$P[y = 1] = 1 - \prod_i^n (1 - p_i) \quad (4.4)$$

onde n é o número de candidatos no quadro e p_i é a probabilidade do candidato i ser cabeça conforme fornecido pelo classificador. Segundo testes realizados, essa formulação não é satisfatória pois não leva em consideração a não idealidade do classificador, de forma que erros na estimativa da probabilidade são sucessivamente propagados. Outra possível abordagem é considerar que a probabilidade de o quadro conter

cabeças pode ser aproximada por

$$P[y = 1] = \max\{p_1, \dots, p_n\}. \quad (4.5)$$

Essa formulação é simples, evita propagar erros, fornece um resultado satisfatório para o sistema e foi a maneira adotada para estimar essa probabilidade.

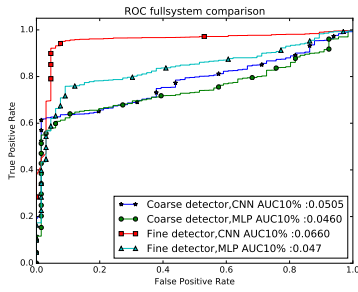
Avalia-se também o efeito da variação dos parâmetros do algoritmo de extração de candidatos e do uso de diferentes classificadores. Mais especificamente, para o extrator de candidatos varia-se o tamanho da janela de procura dos máximos locais, conforme visto na seção 2.2, em dois patamares: grosso (*coarse*), cujo tamanho da janela utilizado foi de 48 pixels, e fino (*fine*), cujo tamanho da janela é a metade: 24 pixels.

Ter uma janela menor para localização dos máximos locais permite maior segurança em encontrar cabeças, pois diminui o risco de máquinas maiores, comuns no ambiente, mascararem a presença de pessoas. Por outro lado, elas introduzem um número maior de candidatos por quadro no sistema, o que pode ser desafiador tanto em performance temporal como no desempenho do classificador, que deve conseguir descartar candidatos que não são cabeças.

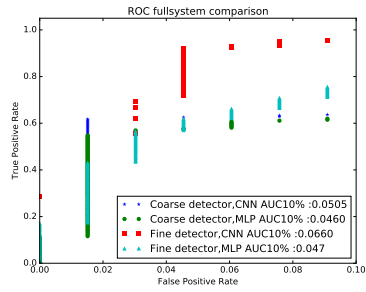
O sistema resulta, no fim, em uma probabilidade do quadro conter uma ou mais cabeças. É possível escolher um limiar ótimo para garantir uma TFP ou TVP especificada. O resultado é exibido na Figura 4.4 e compara as curvas ROC correspondentes à cada escolha dos parâmetros e classificadores num trecho de vídeo de teste.

Primeiramente nota-se que o classificador CNN domina o MLP na maior parte das regiões quando se comparara a mesma escala de extração de candidatos. Pode-se verificar também na Figura 4.4b que existem múltiplas possibilidades de TVP para uma mesma TFP. Esperaria-se que com o aumento do limiar de probabilidade T , acima do qual considera-se que o quadro possui cabeças, a TFP diminuísse. Entretanto, isso não ocorre pelo fato de existirem poucos quadros sem cabeças.

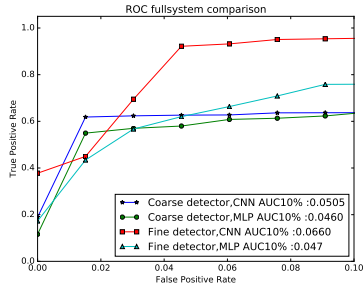
De fato, o vídeo em questão apresenta forte desbalanceamento: mais de 90% dos quadros apresentam cabeças, portanto são poucas as possibilidades de ocorrências de falsos positivos. Para facilitar a comparação, na Figura 4.4c filtrou-se os pontos de forma que para os pontos com mesma TFP apenas o com maior TVP é mantido.



(a)



(b)



(c)

Figura 4.4: Curva ROC comparando diferentes parâmetros e classificadores do sistema completo

Finalmente, conclui-se que para o sistema em questão os melhores parâmetros se dão através do classificador CNN. Dependendo da região de operação, pode-se escolher um extrator de candidatos fino (obtendo 5% de TFP e mais de 90% de TVP) ou grosso (obtendo menos de 2% de TFP com aproximadamente 60% de TVP).

Caso a TFP ainda seja muito elevada pode-se utilizar um sistema que só permita a interrupção da produção após um limite mínimo de quadros detectados com cabeça em sequência.

CAPÍTULO 5

Conclusão

Neste trabalho foi desenvolvido um sistema de detecção de pessoas em um ambiente industrial cujo objetivo é interromper o funcionamento de equipamentos de transporte se uma pessoa for detectada, e assim, aumentar a segurança dos colaboradores na linha de produção. Duas propostas de implementação desse sistema foram avaliadas: a primeira baseada em métodos tradicionais de localização de objetos, vista no Capítulo 2, e a segunda com uma abordagem híbrida, contando com classificadores profundos, vista no Capítulo 3.

Cada solução apresenta suas especificidades e parâmetros, que precisam ser ajustados de maneira individual. O resultado apresentado no Capítulo 4 demonstrou que o sistema com melhor desempenho foi o que utilizou um classificador CNN. Esse fato corrobora a tendência de avanço do uso das técnicas profundas que tem sido observada na comunidade de aprendizado de máquina nos últimos anos.

Embora o sistema ainda não tenha sido implantado na empresa, por desafios técnicos de montagem dos painéis de controle, os ensaios realizados utilizando vídeos de teste sugerem que o sistema funcionará de maneira satisfatória.

Das contribuições desse trabalho podem ser destacadas as compa-

rações de classificadores superficiais (SVM) em conjunto com diferentes descritores aos classificadores profundos, nas diferentes topologias de rede CNN e MLP. O trabalho também demonstra que as técnicas profundas não se restringem a grandes conjuntos de dados (*big data*), mas que podem ser empregadas em situações com volume moderado de amostras, inclusive desbalanceadas.

Durante a implementação, alguns desafios foram encontrados. Primeiramente na escolha dos descritores cuja escolha requer conhecimento específico da aplicação. Embora tenha sido utilizado um descritor recomendado na literatura, assim como uma variação proposta nesse trabalho, os resultados não foram satisfatórios. Também houve dificuldade nas primeiras etapas de treinamento das redes profundas, dado que o conjunto de treinamento era desbalanceado. Por fim, durante a fase de testes houve um conflito na unidade de processamento de vídeo que era utilizada simultaneamente pelo *driver* da câmera stereo e pelo *framework Keras*, mas que foi resolvido utilizando alocação de recursos da *GPU*.

Trabalhos futuros

Duas propostas de trabalhos futuros são sugeridas. Primeiramente a realização de um ajuste fino no classificador atual e em seguida outra arquitetura de detecção integralmente baseada em redes profundas.

Os vídeos utilizados para gerar o dataset deste trabalho só puderam ser gravados em uma ocasião. Isso proporcionou um conjunto de dados moderadamente limitado no sentido de ser desbalanceado e ter um padrão de peças que seriam encontradas na produção. É possível que após a instalação do sistema na fábrica ocorram mais falsos positivos do que previsto no vídeo de teste, justamente pela mudança no padrão de peças.

Poderia-se resolver esse problema gravando todas as amostras classificadas como positivas e manualmente selecionando as que foram falsos positivos. Numa próxima etapa poderia-se fazer um treinamento de ajuste, variando parâmetros apenas das últimas camadas, responsáveis pela classificação de maneira a ter um melhor desempenho real.

Tendo em vista o grande potencial das técnicas de aprendizado profundo, tanto os resultados aqui apresentados como os do estado da arte,

idealizou-se uma terceira solução integralmente baseada em estruturas profundas. Nesta proposta não existiria seleção de candidatos: o quadro inteiro seria tido como entrada do classificador, que deveria então retornar uma saída indicando a existência ou não de pessoas na cena.

A topologia de rede proposta seria convolucional com uma saída bidimensional que representa um mapa de probabilidades de existir uma cabeça em cada posição deste mapa. Essa topologia permitiria maior generalização do problema de forma que houvesse invariância à translação das cabeças no quadro. Além disso, essa saída em mapa permitiria que a rede fosse treinada mais facilmente se comparada à uma única saída que representa a probabilidade do quadro conter cabeça. Isso advém do fato de uma única saída não permitir especificar qual é o objeto de interesse, portanto o treinamento iria exigir um número muito maior de amostras até conseguir modelar esse conhecimento.

Por limitação de tempo essa solução não pôde ser implementada até a apresentação desse trabalho, porém deve seguir em desenvolvimento.

Referências bibliográficas

- [1] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM JOURNAL OF RESEARCH AND DEVELOPMENT*, pages 71–105, 1959.
- [2] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.
- [3] Kevin Murphy, Antonio Torralba, Daniel Eaton, and William Freeman. *Object Detection and Localization Using Local and Global Features*, pages 382–400. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
- [4] B Smith and R Gosine. Support vector machines for object recognition, 2001.
- [5] Frank ROSENBLATT. *The Perceptron: A Perceiving and Recognizing Automaton (Project PARA)*. Report No. 85-460-1. Cornell Aeronautical Laboratory, New York, first edition, jan 1957.
- [6] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

- [7] Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.
- [8] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.
- [9] B Boser Le Cun, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.
- [10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [11] Michael Rauter. Reliable human detection and tracking in top-view depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 529–534, 2013.
- [12] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.
- [13] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.
- [14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [15] Michael A. Nielsen. Neural networks and deep learning. <http://neuralnetworksanddeeplearning.com>, 2015.

- [16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [17] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.
- [18] François Chollet. Keras. <https://github.com/fchollet/keras>, 2015.
- [19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [20] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.
- [21] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc., 2013.
- [22] Andrey Kurenkovs. A 'brief' history of neural nets and deep learning. <http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/>, Dec 2015.

