Universidade Federal de Santa Catarina
Departamento de Engenharia Elétrica e Eletrônica
Laboratório de Comunicações e Sistemas Embarcados

Reconhecimento de Gestos em Vídeos Utilizando Modelos Ocultos de Markov e Redes Neurais Convolucionais Aplicado a Libras

Vinícius Morais Breda

Florianópolis, 2018.

Vinícius Morais Breda

RECONHECIMENTO DE GESTOS EM VÍDEOS UTILIZANDO MODELOS OCULTOS DE MARKOV E REDES NEURAIS CONVOLUCIONAIS APLICADO A LIBRAS

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina para obtenção do Grau de Mestre em Engenharia Elétrica.

Orientador: Prof. Ph.D. Danilo Silva.

Florianópolis
2018

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

Breda, Vinícius Morais
Reconhecimento de gestos em vídeos utilizando modelos ocultos de Markov e redes neurais convolucionais aplicado a Libras / Vinícius Morais Breda ; orientador, Danilo Silva, 2018.
177 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Engenharia Elétrica, Florianópolis, 2018.

Inclui referências.

1. Engenharia Elétrica. 2. Reconhecimento de gestos. 3. Reconhecimento de imagens. 4. Hidden Markov Models. 5. Convolutional Neural Networks. I. Silva, Danilo. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

Vinícius Morais Breda

RECONHECIMENTO DE GESTOS EM VÍDEOS UTILIZANDO MODELOS OCULTOS DE MARKOV E REDES NEURAIS CONVOLUCIONAIS APLICADO A LIBRAS

Esta Dissertação foi julgada adequada para obtenção do Título de Mestre, Área de Concentração em Comunicações e Processamento de Sinais, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica.

Florianópolis, 10 de setembro de 2018.

Prof. Bartolomeu Ferreira Uchoa-Filho, Ph.D.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica,
Universidade Federal de Santa Catarina

Banca Examinadora:

Prof. Danilo Silva, Ph.D.
Orientador, Universidade Federal de Santa Catarina

Prof. Joceli Mayer, Ph.D.
Universidade Federal de Santa Catarina

Prof. Marcelo Ricardo Stemmer, Dr.-Ing.
Universidade Federal de Santa Catarina

Aos meus pais, os quais sempre pude contar em todos os momentos.

AGRADECIMENTOS

Gostaria de expressar meu reconhecimento a todos, que de uma maneira ou outra, colaboraram para a realização deste trabalho, mas em especial a algumas pessoas que foram essenciais para a realização do mesmo:

Ao professor Joceli Mayer que foi o meu orientador inicial, mas que infelizmente não fui capaz de finalizar o trabalho devido a problemas pessoais. Porém quando decidi retomar o trabalho, me ajudou a encontrar outro orientador disposto a dar continuidade ao trabalho.

Ao meu orientador atual, professor Danilo Silva, que topou o desafio de me ajudar a retomar e aprimorar o trabalho inicial, sempre sendo muito prestativo.

À minha namorada, Frã, que esteve comigo em momentos difíceis e continuou ao meu lado incondicionalmente.

À minha tia Rejane e a minha falecida vó Edy, que me ajudaram muito durante a minha infância e adolescência, sem as quais talvez não tivesse a chance de estar aqui hoje.

À minha família, que sempre me deu muito amor. Em especial à minha mãe, que dedicou a vida aos filhos, fazendo o impossível pelo meu irmão e por mim.

Muitas coisas não ousamos empreender por parecerem difíceis; entretanto, são difíceis porque não ousamos empreendê-las. (Sêneca)

RESUMO

A comunicação através de gestos é algo natural para os seres humanos, e por isso é muito importante que sistemas computacionais consigam reconhecer esses gestos, permitindo uma interação natural entre homem e máquina, seja para fins de acessibilidade, segurança, entretenimento, etc. Somando a importância dos gestos aos avanços tecnológicos, o interesse no reconhecimento de gestos utilizando visão computacional tem crescido cada vez mais, pois já se mostrou útil em diversas aplicações. Uma ferramenta que já conquistou o seu espaço no reconhecimento da fala são os modelos ocultos de Markov (*Hidden Markov Models* - HMMs), que também têm se mostrado úteis no reconhecimento de gestos. Outra ferramenta muito poderosa para o reconhecimento de imagens são as redes neurais convolucionais (*convolutional neural networks* - CNNs), que vêm sendo muito utilizadas nos últimos anos devido as modernas placas de processamento gráfico. Este trabalho propõe um sistema para reconhecimento de uma sequência de gestos dinâmicos em vídeo utilizando a combinação das duas ferramentas. São utilizados gestos da LIBRAS (Língua Brasileira de Sinais), que são modelados por HMMs, enquanto que uma CNN modela as formas da mão. As saídas da CNN são utilizadas como descritores para os HMMs em conjunto com descritores convencionais como a posição e inclinação das mãos. Em comparação com o uso exclusivo de descritores convencionais, o uso da CNN proporciona um aumento expressivo da acurácia da classificação no conjunto de teste.

Palavras-chave: Reconhecimento de Imagens. Reconhecimento de Gestos. Libras. CNN. Redes Neurais Convolucionais. HMM. Modelos Ocultos de Markov.

ABSTRACT

Communication through gestures is natural for humans, and so it's very important that computer systems can recognize these gestures, allowing a natural interaction between humans and machine, whether for accessibility, security, entertainment, and so on. The importance of gestures added to the technological advances has led to a growing in the interest in gesture recognition using computer vision, since it has proved usefull in several applications. One tool that has already gained space in speech recognition is the Hidden Markov Models (HMMs), which have also proved useful in gesture recognition. Another very powerful tool for image recognition is the convolutional neural networks (CNNs), which have been widely used in the last years due to modern graphics processing boards. This work proposes a system for recognition of a sequence of dynamic gestures in videos using the combination of the two tools. LIBRAS's gestures (Brazilian Sign Language) are used, which are modeled by HMMs, while a CNN models the hand shapes. CNN's outputs are used as descriptors for the HMMs in conjunction with conventional descriptors such as the position and tilt of the hands. Compared with the exclusive use of conventional descriptors, the use of CNN has showed a significant increase in the accuracy of the classification in the test set.

Keywords: Image Recognition. Gesture Recognition. ASL. CNN. Convolutional Neural Networks. HMM. Hidden Markov Models.

Lista de Figuras

1.1 Exemplo de imagens geradas por duas classes diferentes. . 30
1.2 Número de pixels versus compactação da mão para diferentes imagens. 30

3.1 Exemplo de um HMM. 47
3.2 Hmm para o exemplo dos dados. 50
3.3 Rede de gestos. 52
3.4 Rede de HMMs para quatro gestos. 52
3.5 Arquitetura de uma rede feedforward com 4 camadas, 3 entradas e uma saída. 54
3.6 Exemplo de um neurônio com duas entradas. 54
3.7 Saída de um sigmoide. 55
3.8 Função de ativação relu(z). 56
3.9 Camada de entrada correspondente a uma imagem de 28 x 28 pixels. 60
3.10 Um neurônio oculto ligado a uma região de tamanho 5 x 5 da camada de entrada. 60
3.11 Camada oculta gerada a partir do deslocamento do campo receptivo local sobre a camada de entrada. 61
3.12 Camada convolucional com 3 mapas de atributo. 62
3.13 Max-pooling com uma região 2 x 2. 63
3.14 Camada de entrada com uma camada oculta em uma CNN. 63

3.15 Uma CNN de uma camada. 64

4.1 Módulos do sistema. 75
4.2 Vizinhos-de-4 e vizinhos-de-8. 77
4.3 Classes e objetos. 77
4.4 Regiões de interesse sem intersecção. 78
4.5 Regiões de interesse com intersecção. 79
4.6 Histograma da cor da pele. 81
4.7 Histograma da cor da luva. 81
4.8 PDF da cor da pele. 82
4.9 PDF da cor da luva. 82
4.10 Tabela de probabilidades. 82
4.11 Tabela de classes. 83
4.12 Geração dos mapas de objetos. 84
4.13 Bloco de classificação da Figura 4.12. 84
4.14 Mapa de objetos filtrado para mãos separadas. 85
4.15 Mapa de objetos filtrado para mãos sobrepostas. 85
4.16 Diagrama em blocos do sistema de detecção da face e das mãos. 86
4.17 Regiões de interesse e objetos de interesse detectados em um quadro. 87
4.18 Inicialização do sistema de deteção. 88
4.19 Sistema de detecção das mãos. 89
4.20 Oclusão das mãos. 89
4.21 Intersecção entre os retângulos de menor contorno. 90
4.22 Diagrama do bloco Detecção das mãos do diagrama da Figura 4.19. 90
4.23 Diagrama do bloco Verificação de oclusão do diagrama da Figura 4.19. 92
4.24 Sistema para detecção da face. 93
4.25 Oclusão da face. 94
4.26 Rede de HMMs. 95
4.27 Rede de gestos. 95
4.28 HMM com estrutura esquerda-direita sem transições de escape. 97
4.29 HMM com estrutura esquerda-direita com transição de retorno. 97
4.30 Estrutura criada para os HMMs dos sub-gestos. 98

4.31 Construção do HMM de um gesto. 99

5.1 Tipos de configurações da mão principal. 108
5.2 Módulo Inception. 111
5.3 LeNet-5 utilizada no trabalho. 112
5.4 Variação da AlexNet utilizada no trabalho. 112
5.5 Variação da GoogLeNet utilizada no trabalho. 113
5.6 Estrutura de quatro gestos. 118
5.7 Estrutura de um a quatro gestos. 118
5.8 Estrutura livre. 118

6.1 Distância do gesticular à câmera. 125
6.2 Diferença de qualidade dos vídeos. 126

A.1 Inicialização de um HMM com estrutura esquerda-direita . 150
A.2 Inicialização de um HMMs com estrutura diferente da esquerda-direita. 151
A.3 Formato do arquivo de amostras. 157
A.4 Estrutura do arquivo do HTK que armazena as informações sobre um HMM. 160
A.5 Arquivo de dicionário. 161
A.6 Estrutura do reconhecedor. 161
A.7 Estrutura do reconhecedor para gestos isolados. 161

B.1 Distribuição do z de um neurônio com a rede inicializada com distribuição normal. 178
B.2 Distribuição do z de um neurônio com a rede inicializada com distribuição Gaussiana de média zero e desvio padrão $1/\sqrt{n_{in}}$. 179

Lista de Tabelas

3.1 Propriedades da Série de Fourier. 71
3.2 Descritores implementados. 73

4.1 Descritores utilizados. 103
4.2 Resultado para os conjuntos de descritores gerados na primeira etapa. 104
4.3 Resultado para os conjuntos de descritores gerados na segunda etapa. 104
4.4 Resultado para os conjuntos de descritores gerados na terceira etapa. 105

5.1 Parâmetros da LeNet-5 utilizada no trabalho. 112
5.2 Parâmetros da AlexNet utilizada no trabalho. 113
5.3 Parâmetros da VGG-13 utilizada no trabalho. 114
5.4 Parâmetros da GoogLeNet utilizada no trabalho. 115
5.5 Resultados Das Redes 116
5.6 Resultado para diversos descritores em redes com modelo do movimento de transição. 118
5.7 Tabela de confusão para os descritores do conjunto A - Rede 3. 119
5.8 Tabela de confusão para os descritores da CNN - Rede 3. . 120

5.9 Tabela de confusão para os descritores do conjunto B + CNN - Rede 3. 120
5.10 Resultado do pós-processamento na rede 3. 121

6.1 Características dos vídeos. 124
6.2 Sinais da LIBRAS utilizados. 124
6.3 Conjuntos de Vídeos 125
6.4 Tabela de confusão da CNN. 127
6.5 Resultado das Redes de HMMs no Conjunto de Teste. . . . 128

Lista de Abreviaturas

KNN K-Nearest Neighbours

CNN Convolutional Neural Networks

ANN Artificial Neural Networks

RNN Recurrent Neural Networks

DTW Dynamic Time Warping

fps quadros por segundo

IHM interface homem-máquina

GPU unidade de processamento gráfico

HOG histogram of oriented gradient

SIFT scale invariant feature transform

PCA Principle Component Analysis

SVM Support Vector Machine

MSE Mean Squared Error

SGD Stochastic Gradient Descent

pdf função densidade de probabilidade

pdfs funções densidade de probabilidade

ASL American Sign Language

HMM Hidden Markov Model

HTK Hidden Markov Model Toolkit

LIBRAS Língua Brasileira de Sinais

SLF HTK Standard Lattice Format

EBNF extended Backus-Naur Form

Sumário

1 Introdução **27**
1.1 Motivações 28
1.2 Reconhecimento de Padrões 29
1.3 Reconhecimento de Gestos 31
1.4 Objetivos 33
1.4.1 Objetivo Geral 33
1.4.2 Objetivos Específicos 33
1.5 Contribuições 34
1.6 Organização 34

2 Revisão Bibliográfica **37**
2.1 Segmentação das Mãos 37
2.2 Rastreamento das Mãos 39
2.3 Extração dos Descritores 39
2.4 Classificação 39
2.4.1 K-Nearest Neighbours 39
2.4.2 Support Vector Machine 40
2.4.3 Dynamic Time Warping 40
2.4.4 Artificial Neural Networks 40
2.4.5 Hidden Markov Models 42

3 Conceitos Básicos **45**
3.1 Espaço de Cores . . . 45
3.2 Modelos Ocultos de Markov . . . 46
3.2.1 Parâmetros de um HMM . . . 47
3.2.2 Exemplo . . . 48
3.2.3 Tipos de HMMs . . . 50
3.2.4 Os Três Problemas Básicos . . . 52
3.3 Redes Neurais Artificiais . . . 53
3.3.1 Arquitetura . . . 53
3.3.2 Tipos de Neurônios . . . 54
3.3.3 Aprendizagem . . . 57
3.3.4 Funções de Custo . . . 58
3.4 Redes Neurais Convolucionais (CNNs) . . . 59
3.4.1 Campo Receptivo Local . . . 59
3.4.2 Pesos Compartilhados . . . 60
3.4.3 Camadas de Pooling . . . 62
3.4.4 Uma CNN Completa . . . 63
3.5 Descritores . . . 64
3.5.1 Descritores de Movimento . . . 64
3.5.2 Descritores da Forma da Mão . . . 66
3.5.3 Resumo dos Descritores Utilizados . . . 72

4 Metodologia **75**
4.1 Módulo de Rastreamento . . . 76
4.1.1 Classe e Objeto . . . 76
4.1.2 A Região de Interesse . . . 77
4.1.3 Classificação dos Pixels . . . 80
4.1.4 Modelo da Cor da Pele e Luvas . . . 81
4.1.5 Tabelas de Probabilidades e Tabelas de Classes . . . 81
4.1.6 Mapas de Objetos . . . 83
4.1.7 O Sistema de Detecção . . . 86
4.2 Módulo de Extração dos Descritores . . . 94
4.3 Módulo Classificador . . . 94
4.3.1 Treinamento dos HMMs . . . 96
4.4 Determinação do Número de Estados e Descritores dos HMMs . . . 98
4.4.1 Determinação dos Conjuntos de Descritores . . . 100
4.4.2 Resultados dos Testes . . . 101

5 Integrando a CNN ao Sistema **107**
5.1 CNN para o Reconhecimento das Formas da Mão 107
5.1.1 Definição das Formas da Mão 108
5.1.2 Escolha da Arquitetura da CNN 109
5.1.3 CNNs Implementadas 111
5.1.4 Treinamento e Testes 114
5.2 Utilizando a CNN nos HMMs 116

6 Testes e Resultados **123**
6.1 Informações Sobre os Vídeos 123
6.2 Testes 126
6.2.1 Resultados dos Testes da CNN 127
6.2.2 Resultados dos Testes dos HMMs 127
6.3 Tempo de Processamento 129

7 Conclusão **131**

Referências bibliográficas **135**

A HMMs **143**
A.1 Estimação, Decodificação e Treinamento 143
A.1.1 Problema 1: Estimação 143
A.1.2 Problema 2: Decodificação 146
A.1.3 Problema 3: Treinamento 149
A.2 Detalhes de Implementação 153
A.3 Hidden Markov Model ToolKit (HTK) 155
A.3.1 O Arquivo de Amostras 155
A.3.2 O Arquivo de Estrutura do HMM 157
A.3.3 A Rede de Comunicação e o Dicionário 158

B ANNs **163**
B.1 Gradient Descent 163
B.1.1 Gradient Descent em Duas Dimensões 164
B.1.2 Gradient Descent para N Dimensões 165
B.1.3 Gradient Descent Aplicado a Redes Neurais 165
B.1.4 Stochastic Gradient Descent 165
B.2 O Algoritmo Backpropagation 166
B.3 Dedução das Equações do Algoritmo Backpropagation . . 168
B.3.1 Erro Devido aos Neurônios da Última Camada . . . 169

B.3.2 Erro Devido aos Neurônios das Camadas Restantes 170
B.3.3 Cálculo de $\frac{\partial C}{\partial b_j^k}$ 171
B.3.4 Cálculo de $\frac{\partial C}{\partial w_{ji}^k}$ 172
B.4 Análise das Funções de Custo 172
B.5 Derivando as Funções de Custo 174
B.5.1 Derivando $\sigma(z_j^L)$ para uma Sigmoide 175
B.5.2 Obtendo δ_j^L para o Erro Quadrático Médio e Sigmoide 175
B.5.3 Obtendo δ_j^L para Cross-Entropy e Sigmoide 176
B.5.4 Obtendo δ_j^L para Log-Likelihood e Softmax 177
B.6 Inicialização dos Pesos de uma ANN 178

CAPÍTULO 1

Introdução

Os gestos são uma forma natural de comunicação que possibilita a comunicação e expressão de emoções sem o uso de palavras. Como no mundo atual a tecnologia está presente em praticamente todos os lugares, sermos capazes de interagir com ela de forma natural é algo extremamente importante. O reconhecimento de gestos consiste justamente em fazer com que máquinas sejam capazes de reconhecer gestos feitos por seres humanos e interpretá-los a fim de realizar alguma tarefa útil.

A visão computacional é um campo muito desafiador e que cresceu muito nas últimas décadas, principalmente devido a evolução tecnológica que permitiu a captação de mais informações através de câmeras com múltiplas lentes, com percepção de profundidade, infravermelha e do sensor kinect. Adicionalmente a isso a evolução dos microprocessadores, memórias e unidade de processamento gráfico (GPU) permitiu a implementação de técnicas de reconhecimento que antes pareciam impraticáveis.

Um campo da visão computacional é o reconhecimento de gestos, que ganha cada vez mais importância e já se mostrou muito útil em interface homem-máquina (IHM), como a interação com jogos ou con-

trole de dispositivos. Além disso, existem outras aplicações com muito potencial, como o monitoramento de ambientes, ou a transcrição de gestos da língua de sinais para a língua escrita.

Expressões faciais, movimentos corporais, movimentos manuais, podem ser considerados gestos. Este trabalho tem o foco no reconhecimento de gestos manuais, que são os gestos realizados pelas mãos, e por esse motivo sempre que se referir ao reconhecimento de gestos neste trabalho, refere-se ao reconhecimento de gestos manuais. Além disso, como trabalhar-se-á com gestos da Língua Brasileira de Sinais (LIBRAS), o termo gesto e sinal são intercambiáveis no contexto desse trabalho.

Sistemas de reconhecimento de gestos podem ser divididos normalmente em dois tipos, quanto ao método utilizado para obter os dados:

- Métodos baseados em luvas: utilizam luvas para obter os dados. Essas luvas são dotadas de sensores que fornecem informações precisas sobre as mãos, porém são desconfortáveis e dificultam a execução natural dos gestos, além de normalmente possuírem um alto custo econômico.

- Métodos baseados em visão computacional: utilizam câmeras para a obtenção dos vídeos dos gestos, que são processados para detectar as mãos e extrair as informações necessárias para o reconhecimento. Esses métodos possuem a vantagem de possuírem um baixo custo e não serem invasivos, porém necessitam de um processamento complexo dos dados para tentar compensar uma série de dificuldades que surgem ao se extrair as informações dos vídeos.

1.1 Motivações

Atualmente os smartphones se tornaram uma ferramenta indispensável presente na vida de quase todas as pessoas. Esses possuem câmeras e hardwares cada vez melhores, o que possibilita a gravação de vídeos e o processamento de dados de forma cada vez mais veloz. Essa evolução tecnológica e o baixo custo de aquisição incentiva a utilização dos métodos baseados em visão computacional. De fato, os métodos baseados em visão tem sido foco da atenção dos pesquisadores nos últimos anos, onde várias abordagens foram desenvolvidas. Porém essa ainda

é uma tarefa complexa e muito desafiadora com vários problemas a serem resolvidos, como: o reconhecimento em um ambiente com um fundo complexo, com um fundo dinâmico, com variação da luminosidade, com múltiplos gesticuladores, com diferentes pontos de visão. Devido a todos esses problemas são várias as restrições que existem nesses métodos, como um ambiente com fundo e luminosidade controlados, utilização de roupas que cobrem todo o corpo, utilização de luvas coloridas, determinada distância e ângulo da câmera, entre outras, dependendo da abordagem adotada.

Outro problema que existe nos métodos baseados em visão é o fato do alto grau de liberdade das mãos e as oclusões que ocorrem, o que faz com que as mãos possam assumir uma infinidade de formas em uma imagem. Além disso, no reconhecimento de gestos contínuos, ainda existe a desafiadora tarefa de lidar com os movimentos de coarticulação, que são os movimentos de transição entre os gestos.

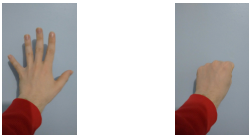
Sabendo da vasta gama de aplicações para o reconhecimento de gestos e da contínua evolução tecnológica, esse trabalho se propõem a desenvolver um sistema capaz de reconhecer gestos complexos da LIBRAS em tempo real, utilizando técnicas modernas de reconhecimento de imagens como as Convolutional Neural Networks (CNN) em conjunto com uma abordagem mais clássica como Hidden Markov Model (HMM)s, já bem estabelecida no campo do reconhecimento da fala. Sendo assim, o sistema também será capaz de reconhecer gestos mais simples e bem especificados para alguma aplicação específica.

1.2 Reconhecimento de Padrões

O reconhecimento de padrões ([1], [2]) possui aplicação nas mais variadas áreas, como medicina, biologia, música, visão computacional, reconhecimento de caracteres, auxílio no diagnóstico de doenças, reconhecimento da fala, análise financeiras, etc.

O objetivo do reconhecimento de padrões consiste em, a partir de um conjunto de dados, classificar uma amostra entre determinadas classes. Tomemos como exemplo ilustrativo a figura 1.1, onde observam-se as imagens de uma mão aberta e outra fechada. Podemos dizer que a figura 1.1 (a) foi gerada por uma determinada classe A e a figura 1.1 (b) por uma classe B, onde pode-se supor que essas duas imagens

foram retiradas de um banco de dados com várias outras imagens de mãos abertas e fechadas, as quais são definidas como diferentes padrões gerados pelas classes A e B.



(a) (b)

Figura 1.1: Exemplo de imagens geradas por duas classes diferentes.

Primeiramente é necessário identificar uma quantia mensurável que seja capaz de diferenciar os padrões das diferentes classes. A figura 1.2 mostra um gráfico hipotético contendo o número de pixels (nP) e a compactação da mão (Cpt). Cada ponto no gráfico corresponde a uma diferente imagem do banco de dados. Podemos observar que os padrões gerados pela classe A (círculos) estão distribuídos em uma área diferente da dos padrões gerados pela classe B (sinais de adição), e a linha reta parece um bom limiar de separação entre as classes. Supondo agora que tenha-se uma nova imagem, da qual não sabe-se qual classe a gerou. Seu número de pixels e compactação são representados pelo * no gráfico da figura 1.2, sendo, então, mais lógico assumir que esse padrão desconhecido seja mais provável de ter sido gerado pela classe A do que pela B.

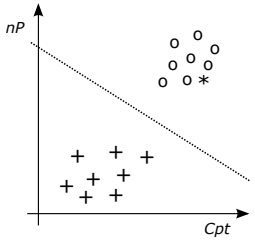


Figura 1.2: Número de pixels (nP) versus compactação (Cpt) da mão para diferentes imagens. Os círculos correspondem às imagens geradas pela classe A, e o sinal de adição às geradas pela classe B.

As medidas utilizadas para diferenciar os padrões, que no exemplo anterior foram o número de pixels e a compactação da mão, são chamadas de descritores.

O reconhecimento de padrões, ou mais especificamente, o modo como o sistema de reconhecimento é treinado pode ser dividido em três categorias: aprendizagem supervisionada, aprendizagem não supervisionada, e aprendizagem semi-supervisionada.

No treinamento supervisionado utiliza-se a informação *a priori* sobre o conjunto de dados para treinamento (dados rotulados). Como exemplo, pode-se pensar em um conjunto de dados que foram adquiridos a partir de imagens da mão fechada e outro conjunto a partir de imagens da mão aberta. O treinamento do classificador projetado neste trabalho utiliza o treinamento supervisionado.

No treinamento não supervisionado não se tem conhecimento sobre a qual classe os dados utilizados para o treinamento pertencem (dados não rotulados). Nesse caso o sistema deve ser capaz de descobrir similaridades entre os dados e agrupá-los de acordo com o nível de similaridade.

No caso semi-supervisionado, tem-se um conjunto de dados de treinamento em que se conhecem as classes que os geraram, e outro conjunto em que não se tem essa informação. Nesses casos, pode-se conseguir algumas informações sobre os dados não rotulados a partir dos dados rotulados.

1.3 Reconhecimento de Gestos

Em se tratando do reconhecimento de gestos manuais, podemos classificar os gestos em dois tipos:

- Reconhecimento de gestos estáticos: os gestos são descritos apenas pela forma da mão naquele instante, ou pela forma e sua posição.

- Reconhecimento de gestos dinâmicos: os gestos se desenvolvem no tempo, possuindo somente uma trajetória ou uma trajetória em conjunto com uma forma de mão que pode ou não variar no desenvolvimento do gesto.

O reconhecimento de gestos dinâmicos que contém tanto a informação da trajetória quanto a forma da mão é uma tarefa muito mais complexa que o reconhecimento de gestos estáticos, pois a sua natureza temporal e a infinidade de combinações entre trajetórias e formas da mão geram uma complexidade muito maior.

Os gestos da LIBRAS são realizados com uma ou com as duas mãos, sendo que a mão principal (para as pessoas destras é a mão direita e para as canhotas a esquerda) contém a principal fonte de informação do gesto. A outra mão, chamada de secundária, realiza movimentos auxiliares à mão principal. Todo gesto dinâmico da LIBRAS possui uma movimentação, bem como um formato inicial e final da mão principal. Quando a mão realiza um movimento ela parte de uma posição inicial para uma posição final. Logo os principais discriminantes entre gestos são o movimento descrito pela mão principal e o formato da mão principal nas suas posições inicial e final, pois dois gestos que descrevem a mesma trajetória mas cuja configuração da mão é diferente, também são diferentes.

Ainda podemos separar o reconhecimento de gestos quanto a gestos isolados e contínuos. O reconhecimento de gestos isolados consiste em reconhecer gestos em vídeos que contenham apenas um gesto, ou que exista algum tipo de separação explícita entre os gestos. Já o reconhecimento de gestos contínuos consiste em reconhecer uma sequência de gestos feitos continuamente em um vídeo. No reconhecimento contínuo não sabemos onde inicia nem onde termina cada gesto, e podemos determinar ou não um número fixo ou máximo de gestos no vídeo, dependendo da aplicação. Além disso, outro problema são os movimentos de transições entre os gestos, que geram inúmeras trajetórias diferentes dependendo da combinação entre as posições de início e fim da mão, bem como da sua forma.

Os sistemas de reconhecimento de gestos são normalmente divididos em quatro etapas: a segmentação das mãos, o rastreamento das mãos, a extração dos descritores e a classificação.

(i) Segmentação das mãos: para que se possa extrair descritores para identificar os gestos é necessário, primeiramente, identificar os objetos de interesse dos quais se deseja extrair os descritores. A segmentação das mãos consiste justamente em localizar e particionar uma imagem em seus objetos constituintes, como as mãos,

por exemplo. Esta é uma etapa muito importante, pois todas as etapas posteriores dependem da acurácia dessa segmentação.

(*ii*) Rastreamento das mãos: essa etapa é responsável por rastrear os objetos de interesse durante o gesto.

(*iii*) Extração dos descritores: nessa etapa são extraídos os descritores dos objetos de interesse.

(*iv*) Classificação: os descritores extraídos são utilizados por um ou mais classificadores, que devem ser capaz de interpretar as informações contidas nos descritores para reconhecer os diferentes gestos.

1.4 Objetivos

Nessa seção, o objetivo geral e os objetivos específicos perseguidos nesta dissertação serão apresentados.

1.4.1 Objetivo Geral

O objetivo geral desta dissertação é propor um sistema capaz de reconhecer, em tempo real, uma sequência de gestos dinâmicos da LIBRAS em vídeos, utilizando uma combinação de HMMs e CNN.

1.4.2 Objetivos Específicos

- Identificar os descritores mais relevantes dentre os vários propostos neste trabalho.
- Identificar um número de estados adequado para os HMMs.
- Identificar as formas de mãos relevantes para os gestos selecionados e modelá-las através de uma CNN, gerando novos descritores.
- Propor um sistema capaz de reconhecer uma sequência de gestos dinâmicos com boa acurácia e um baixo tempo de processamento.

1.5 Contribuições

Nesta dissertação é proposto um sistema para o reconhecimento de gestos dinâmicos e contínuos utilizando HMMs e CNN. As contribuições deste trabalho são as seguintes:

- Criação de um novo banco de dados com sinais dinâmicos e contínuos da LIBRAS.
- Verificação do desempenho de diversos descritores utilizados para o reconhecimento com HMMs.
- A utilização de uma CNN para gerar descritores relativos ao formato da mão.
- Foi publicado um artigo na *The 2014 International Conference on Image Processing, Computer Vision, and Pattern Recognition* com o título *Continuous Gesture Recognition Using Hidden Markov Models*. O artigo apresenta os resultados da primeira parte desta dissertação, utilizando apenas HMMs, sem as CNNs.
- A principal contribuição deste trabalho é a proposta de um sistema que integra HMMs e CNN para o reconhecimento de gestos dinâmicos e contínuos.

Os resultados da primeira parte desta dissertação, utilizando apenas HMMs, sem as CNNs, foram publicados em [3].

1.6 Organização

Os demais capítulos deste trabalho estão organizados da seguinte forma: o capítulo 2 traz a revisão bibliográfica. O capítulo 3 apresenta uma revisão sobre a teoria necessária para o entendimento do trabalho, bem como os descritores utilizados no mesmo. O capítulo 4 explica a metodologia utilizada pelo sistema de reconhecimento de gestos, desde o rastreamento das mãos nos vídeos até a modelagem dos gestos pelos HMMs. No capítulo 5 são apresentadas diferentes estruturas de CNNs e a sua integração no sistema. O capítulo 6 apresenta informações sobre os vídeos utilizados no trabalho, bem como os resultados finais obtidos pelo sistema. O apêndice A apresenta o equacionamento da

estimação, decodificação e treinamento dos HMMs, bem como uma introdução sobre o Hidden Markov Model Toolkit (HTK). O apêndice B traz equacionamentos e algoritmos relacionados as Artificial Neural Networks (ANN)s.

CAPÍTULO 2

Revisão Bibliográfica

Neste capítulo é feita uma breve revisão sobre alguns métodos utilizados nas etapas de um sistema de reconhecimento de gestos.

2.1 Segmentação das Mãos

Um dos principais métodos utilizados para identificar as mãos em uma imagem é através da informação da cor da pele, onde a detecção da cor da pele é normalmente uma das primeiras etapas no processo de detecção das mãos. Porém a detecção através da cor da pele não é uma tarefa trivial, pois a mesma é sensível a vários fatores, como a variação da iluminação, as características da câmera, a etnia das pessoas, sombras, características individuais, um fundo complexo ou dinâmico, mais de um gesticulador, etc [4].

Muitos dos métodos já propostos para classificar um pixel como pertencente ou não a pele se enquadram em dois grupos: métodos baseados na cor do pixel e os métodos baseados na informação da textura da região onde o pixel se encontra. Os métodos baseados na cor do pixel necessitam de uma baixa complexidade computacional em comparação com os métodos regionais, porém quando pixels que não pertencem

a pele possuem cor semelhante a da mesma, esses métodos tendem a gerar uma resposta precária [5]. Já os métodos regionais tentam contornar esse problema levando em conta a informação de uma região da imagem [6], [7], [8], [9].

Os métodos de classificação baseados na cor do pixel podem ser divididos em três principais categorias: métodos baseados em regras, métodos não paramétricos e métodos paramétricos. Os dois últimos são métodos estatísticos que se baseiam na probabilidade do pixel pertencer a pele dado que o mesmo possui a cor c, $p(\text{pele}|c)$. [10]

- **Métodos baseados em regras**: A cor do pixel é processada por diversas regras para classificá-lo como pertencente ou não a determinada classe. Esses métodos tendem a ser muito rápidos, porém sua performance é insatisfatória na maioria dos casos.

- **Métodos não-paramétricos**: Esses métodos estimam $p(\text{pele}|c)$ através de um conjunto de dados de treinamento sem considerar um modelo de probabilidade. Normalmente possuem uma taxa alta de verdadeiros positivos e uma taxa baixa de falsos positivos, ou seja, esses métodos conseguem classificar a maioria dos pixels pertencentes a pele corretamente, enquanto poucos pixels não pertencentes a pele são classificados como pixels da pele. Uma grande desvantagem desses métodos é a grande quantidade de memória necessária para armazenar $p(\text{pele}|c)$ [11].

- **Métodos paramétricos**: Nesses métodos assume-se que a função de probabilidade $p(\text{pele}|c)$ possui uma forma paramétrica, como uma distribuição gaussiana ou uma mistura de gaussianas. A principal vantagem desses métodos em comparação aos não paramétricos é que eles necessitam de uma pequena quantidade de memória para armazenar o modelo, em troca de um maior tempo de processamento. Geralmente possuem uma taxa de de verdadeiros positivos maior que os métodos não-paramétricos, porém a taxa de falsos positivos também é maior [5].

Uma desvantagem do método baseado na cor da pele é que se o fundo conter outros objetos da mesma cor da pele haverá muito ruído. Outros métodos consistem na subtração do fundo entre os quadros (*frames*) de modo a detectar regiões onde ocorrem movimentos [12], [13].

Esse método da subtração do fundo pode ser utilizado para fundos complexos, mas apresenta desvantagens quando ocorrem variações abruptas na iluminação. Já o método baseado na entropia da imagem proposto em [14] é capaz de lidar com variações na iluminação.

2.2 Rastreamento das Mãos

O rastreamento das mãos consiste em saber as posições consecutivas das mãos durante um gesto. Os rastreamento utiliza técnicas baseadas na cor e subtração do fundo para detectar regiões da cor da pele em movimento [15] ou técnicas baseados em filtros, como a utilização do filtro de Kalman para prever a localização da mão no quadro atual baseado na sua localização no quadro anterior [16].

2.3 Extração dos Descritores

Os descritores são formas de representar os gestos de modo que o classificador consiga entender. Vários tipos de descritores já foram propostos, sendo que a combinação de múltiplos descritores geralmente apresenta melhor resultado. Eles são extraídos a partir das etapas anteriores e são utilizados como entrada para o classificador.

No reconhecimento de gestos utiliza-se, normalmente, dois tipos de descritores: descritores que representam a forma da mão - como descritores geométricos, histogram of oriented gradient (HOG), scale invariant feature transform (SIFT), Principle Component Analysis (PCA) - e descritores que representam a trajetória da mão [17], [18].

2.4 Classificação

Os classificadores utilizam métodos estatísticos que têm em sua entrada os descritores e geram como saída um rótulo, que no caso corresponde ao gesto realizado. Alguns dos classificadores mais utilizados são Dynamic Time Warping (DTW), ANN e HMMs.

2.4.1 K-Nearest Neighbours

K-Nearest Neighbours (KNN) é um classificador utilizado no reconhecimento de gestos estáticos, muito simples de ser implementado e

que permite uma calibragem muito fácil para novos usuários, apenas incluindo novas amostras no conjunto de treinamento.

A classificação de uma amostra é feita medindo-se a distância (euclidiana ou outra) entre ela e as k amostras mais próximas (vizinhos) no conjunto de treinamento, pertencendo a classe que possuir mais vizinhos. O problema desse tipo de classificador é que dependendo da quantidade de amostras de treinamento, a classificação pode se tornar muito demorada, sendo necessário algum tipo de técnica para reduzir a quantidade de amostras de treinamento. Por outro lado, diminuindo-se a quantidade de amostras de treinamento implica em uma necessária redução da dimensão do conjunto de descritores.

2.4.2 Support Vector Machine

Support Vector Machine (SVM) é um classificador muito utilizado para o reconhecimento de padrões, como gestos estáticos por exemplo. Consiste em encontrar um hiperplano de decisão capaz de separar uma classe de outra. Normalmente as classes não são linearmente separáveis, mas isso pode ser alcançado através do mapeamento dos descritores para um espaço dimensional maior. Atualmente, SVM possui um desempenho inferior e um tempo de reconhecimento superior a técnicas mais modernas com as CNN.

2.4.3 Dynamic Time Warping

Consiste em medir a similaridade entre duas sequências as quais podem variar no tempo ou velocidade. Seu objetivo é alinhar duas sequências de vetores de descritores deformando o eixo do tempo iterativamente ate que uma combinação ótima entre as duas sequências seja encontrada. O ponto negativo desse método é o tempo necessário para que ocorra a classificação.

2.4.4 Artificial Neural Networks

ANN são redes compostas por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, que são conectadas através de ligações contendo pesos modificáveis. A camada de entrada representa os componentes do vetor de descritores, e a camada de saída emite valores que são utilizados para a classificação. São normalmente utilizadas

para o reconhecimento de gestos estáticos, uma vez que não se adaptam naturalmente as mudanças ao longo do tempo que são produzidas pelos gestos dinâmicos.

Existem diversos tipos de ANN, sendo a rede do tipo *feedforward perceptron* a mais simples. Essa rede pode receber em sua entrada um vetor com os pixels da imagem ou um vetor composto de descritores. Redes desse tipo foram utilizadas em [19] para reconhecer 5 diferentes direções em que a mão estava apontando ou quantos dedos estavam sendo mostrados. Já em [20] ela é utilizada para reconhecer 10 posturas da mão, obtendo uma acurácia em torno de 90% em ambos os trabalhos.

Outro tipo muito importante que vem sendo utilizado recentemente na classificação de imagens é a rede neural convolucional (convolutional neural network, CNN), devido a possuir uma estrutura bidimensional e uma quantidade menor de parâmetros que uma rede *feedforward* convencional profunda. As CNNs são redes profundas que utilizam a matriz de pixels da imagem como sua entrada. Em [21] é utilizada uma CNN para fazer o reconhecimento de 24 posturas estáticas da American Sign Language (ASL). Outros trabalhos comparam a abordagem utilizando CNN com outra mais clássica utilizando SVM, como em [22], onde são reconhecidas 6 posturas baseadas na contagem dos dedos, e em [23], onde são reconhecidas 36 posturas da ASL. Em ambos os casos a acurácia utilizando CNN é muito superior e, embora o treinamento da CNN seja muito mais lento, o reconhecimento é cerca de 15 vezes mais rápido que utilizando SVM.

As CNNs podem possuir diversas estruturas de acordo como são construídas, como a LeNet, AlexNet, VGG-Net e GoogLeNet, citadas em nível de complexidade. Em [24] é feito um comparativo entre essas estruturas, sendo que as estruturas mais simples são executadas mais rapidamente, mas geraram uma acurácia inferior.

Apesar das ANNs serem normalmente utilizadas para reconhecimento estático, em [25] são utilizadas CNNs para fazer o reconhecimento de gestos dinâmicos. Esse trabalho utiliza vídeos do ChaLearn Looking at People 2014, o qual contém um vocabulário de 20 gestos realizados por 27 pessoas, totalizando um conjunto de 10143 amostras de gestos que é dividida entre treinamento, validação e teste. Apesar de cada vídeo conter vários gestos, após cada gesto a mão volta para uma posição de repouso. Como os vídeos foram gravados com o Ki-

nect, para cada vídeo também são disponibilizados vídeos da imagem de profundidade, local do gesticulador na imagem de profundidade e a posição de pontos chaves do corpo humano. O sistema utiliza duas CNNs, uma que recebe como entrada imagens com foco na mão e a outra com foco na parte superior do corpo. As saídas das duas CNNs servem como entrada para uma ANN convencional que é responsável por fazer a classificação do gesto. Como entrada das CNNs são utilizados 32 quadros do vídeo em escala de cinza e do vídeo da imagem de profundidade. Uma janela deslizante seleciona os 32 quadros para cada intervalo do vídeo e alimenta a rede, e os segmentos classificados com a mesma classe são considerados um segmento do gesto. O classificador obteve uma acurácia de 95.68% no conjunto de validação e 91.7% no conjunto de teste.

Por fim, outro tipo de rede neural artificial que vem sendo utilizado para o reconhecimento de gestos dinâmicos são as Recurrent Neural Networks (RNN). Essas redes possuem uma memória e por isso são capazes de reconhecer sequências temporais.

2.4.5 Hidden Markov Models

HMMs são máquinas de estados estocásticas que modelam processos estocásticos não estacionários através de uma série de processos estocásticos estacionários. São ideais para processos temporais, e por isso são muito utilizados no reconhecimento da fala. Uma vez que gestos, assim como a fala, são processos estocásticos não estacionários, em [26] é feito o uso de HMMs para o reconhecimento de gestos dinâmicos. Em um ambiente controlado e com a utilização de luvas de cores diferentes, os autores se propõem a reconhecer sentenças contínuas compostas por um vocabulário de 40 sinais da ASL, que são classificadas de acordo com a sua classe gramatical. Os autores utilizam HMMs com estrutura equerda-direita e somente quatro descritores para cada mão, sendo a posição e mais três descritores geométricos. Utilizando 395 vídeos para o treinamento e 99 o teste, o sistema obteve uma acurácia de 99.2% fazendo uso de uma estrutura gramatical, o que impede inserções e exclusões, uma vez o número de sinais no vídeo é conhecido. Já sem a utilização da estrutura gramatical, a acurácia caiu para 91.2%. Importante notar que os autores não lidam com os movimentos de co-articulação e que não foi realizado nenhum teste de validação.

Os HMMs também são utilizados em [27] para fazer o reconhecimento de gestos isolados, mas com um vocabulário maior, contendo 262 gestos isolados da Língua de Sinais da Holanda. O autor utiliza luvas coloridas, sendo uma com 7 cores, uma para cada dedo, palma e parte superior da palma. A outra luva é simples. Os modelos possuem estrutura esquerda-direita, porém a quantidade de estados de cada modelo varia de acordo com o tamanho do sinal que ele modela. Utilizando um conjunto de gestos realizados por duas pessoas, foi obtido uma taxa de reconhecimento de 91.3%. Reduzindo o vocabulário para um conjunto de 43 gestos, a taxa subiu para 99.2%.

Uma abordagem diferente utilizando HMMs é feita em [28]. Nesse trabalho os autores não fazem uso de visão computacional, mas utilizam DataGloves para a extração dos descritores. O sistema reconhece sentenças com vários gestos, mas necessita de uma pausa entre eles, de modo a identificar o início e fim de cada gesto. Em vez de apenas um HMM por gesto que tem como entrada o conjunto de descritores, são utilizados quatro HMMs por gesto, onde cada modelo classifica o gesto baseado em descritores de determinada característica. Dessa forma tem-se um HMM que classifica o gesto baseado na postura da mão, outro baseado na posição, outro na orientação e outro no movimento. Também é utilizado um modelo de linguagem que relaciona os vários gestos estatisticamente. Finalmente, uma combinação linear das entropias das probabilidades geradas pelos HMMs e das probabilidades geradas pelo modelo de linguagem gera a sentença com mais alta probabilidade. Em um vocabulário contendo 71 sinais o sistema obteve uma taxa de classificação correta de 84.5% para gestos isolados, 87% para sentenças curtas (com média de 2.83 sinais por sentença) e 93.3% para sentenças longas (com média de 5.24 sinais por sentença).

Em [29] os autores comparam diferentes estrutura de HMMs no reconhecimento de gestos, demonstrando que a estrutura esquerda-direita é mais adequada que a estrutura totalmente conectada para o reconhecimento de gestos, pois focam mais na aprendizagem dos parâmetros que na da estrutura. Colocando-se limitações nas transições da estrutura esquerda-direita obteve-se um ganho na taxa de reconhecimento.

Um sistema de reconhecimento de sinais da Língua Britânica de Sinais independente do usuário é proposto em [30]. O trabalho conta com um vocabulário com uma média de 229 sinais isolados realizados

por quatro gesticuladores em ambiente controlado. O sistema possui um modelo de cor da pele independente da iluminação e da pessoa, mas mesmo assim possui as restrições do gesticulador utilizar camisas de manga comprida e roupa de cor diferente da cor da pele. Como nos trabalhos anteriores, são utilizados somente alguns descritores geométricos, como o centro de gravidade das mãos e a sua área, que alimentam os HMMs utilizados para a classificação. Os testes foram realizados da seguinte forma: os vídeos de um gesticulador foram utilizados para o teste e combinações dos vídeos dos outros gesticuladores foram utilizados para o treinamento, fazendo com vídeos do gesticulador de teste nunca estivessem no conjunto de treinamento. As taxas de reconhecimento variaram de 3.7% a 44.1% dependendo do gesticulador de teste e dos gesticuladores de treinamento escolhidos. Ao final é realizado outro teste, onde são utilizados para o treinamento os vídeos dos quatro gesticuladores e os testes são feitos em vídeos realizados por outros 6 gesticuladores em ambientes não controlados. Porém nesse teste é utilizado apenas um subconjunto do vocabulário. A média de reconhecimento desse teste foi de 95% para um vocabulário de seis gestos e 87.8% para um vocabulário de 18 gestos.

Em [18] são utilizados três classes de HMMs por gesto, um para cada tipo de descritor. São selecionados os três gestos com maior probabilidade, uma para cada classe de HMM, então é feita uma votação onde escolhe-se o gesto que teve maior ocorrência dentre os três selecionados. Em caso dos três gestos selecionados serem diferentes, escolhe-se o que obteve maior probabilidade na classificação pelos HMMs. É utilizado um vocabulário com 30 palavras que não ocluem a face. Dos 110 vídeos utilizados, 90 são para o treinamento, e todos os 110 para teste, obtendo uma taxa de classificação de 89.09%.

CAPÍTULO 3

Conceitos Básicos

Este capítulo apresenta uma revisão bibliográfica sobre alguns dos assuntos fundamentais para o entendimento deste trabalho. Na seção 3.1 é apresentado o espaço de cor YCrCb utilizado neste trabalho. Na seção 3.2 é feita uma revisão sobre os HMMs. A seção 3.3 apresenta as redes neurais artificiais. As CNNs são apresentadas em seguida, na seção 3.4. Por fim, a seção 3.5 apresenta os descritores utilizados neste trabalho.

3.1 Espaço de Cores

Para fazer a classificação dos pixels baseando-se em sua cor é necessário, primeiramente, escolher um espaço de cor, isto é, simplesmente um sistema de coordenadas onde cada cor é representada por um ponto. Existem vários espaços de cores desenvolvidos para diversos propósitos, sendo que neste trabalho e utilizado o espaço YCrCb.

Enquanto no espaço RGB existe uma correlação entre as informações de crominância e luminância, o espaço YCrCb separa a informação de luminância (Y) das componentes de crominância (Cr e Cb). Além disso, a transformação do espaço RGB para YCrCb é linear, tornando

este espaço um dos mais utilizados no reconhecimento baseado em cor [31], [32], [33], [34], [35].

As crominâncias Cr e Cb correspondem a quantidade de vermelho e azul na cor. O mapeamento do espaço RGB para o espaço YCrCb é feito através da seguinte transformação [36]:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cr &= (R - Y)0.713 + K \\ Cb &= (B - Y)0.564 + K \end{aligned} \tag{3.1}$$

Onde $K = 128$ para imagens de 8 *bits* por canal, e R, G e B são o valores dos canais vermelho, verde e azul, respectivamente.

3.2 Modelos Ocultos de Markov

HMM é um tipo de modelagem estocástica apropriada para sequências estocásticas não estacionárias, de forma a modelar uma sequência de observações como um conjunto de processos estacionários. O HMM é basicamente uma máquina de estados estocástica, a qual é constituída por um conjunto de estados, onde cada estado é responsável pela emissão de um símbolo de acordo com uma distribuição de probabilidade. A transição entre os estados ocorre de acordo com um conjunto de probabilidades chamado de probabilidades de transição. Os modelos são chamados de ocultos devido ao fato de termos acesso apenas as observações emitidas pelos estados, e não aos estados que emitiram as observações. De acordo com a sequência de observações pode-se estimar qual a sequência de estados mais provável de tê-las gerado [37], [38].

A figura 3.1 mostra um HMM com quatro estados, onde cada estado possui uma probabilidade de emitir determinado símbolo s. Pode-se ver que as transições entre os estados ocorrem de forma que, no instante inicial o estado ativo seja o primeiro, e a cada instante o estado ativo deve ser ele mesmo ou o próximo. Este é um exemplo de um HMM do tipo esquerda-direita, porém estas transições podem ocorrer de qualquer maneira possível, dependendo do que se deseja modelar.

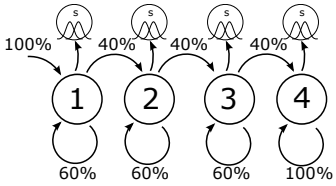


Figura 3.1: Exemplo de um HMM.

3.2.1 Parâmetros de um HMM

Denominamos aqui o conjunto de parâmetros que os estados podem emitir como símbolos, e os símbolos emitidos pelos estados em determinada realização do processo como observações. Logo uma realização do processo gera uma sequência de observações $\mathbf{X} = \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_T}$ proveniente da sequência de estados $\mathbf{y} = y_1, y_2, \ldots, y_T$.

Um HMM pode ser definido pelos seguintes parâmetros: o conjunto de distribuições de probabilidade de emissão de símbolos de cada estado $\mathbf{b}$; as probabilidades de transições entre os estados $\mathbf{A}$; a probabilidade inicial de cada estado $\boldsymbol{\pi}$. Então um HMM pode ser representado como $\lambda = (\mathbf{A}, \mathbf{b}, \boldsymbol{\pi})$, onde:

- Os N estados de um modelo são definidos por $E = \{e_1, \ldots, e_N\}$.
- Os M símbolos que cada estado é capaz de emitir são dados por $S = \{\mathbf{s_1}, \ldots, \mathbf{s_M}\}$. Se o espaço de símbolos for contínuo, então M é infinito.
- A matriz de probabilidade de transição entre os estados é definida por $\mathbf{A}$, onde seus membros a_{ij} representam a probabilidade de transição do estado e_i para o estado e_j.

$$\begin{aligned} &a_{ij} = p\{y_{t+1} = e_j | y_t = e_i\}, && 1 \leq i \text{ e } j \leq N \\ &a_{ij} \geq 0, && 1 \leq i \text{ e } j \leq N \\ &\textstyle\sum_{j=1}^{N} a_{ij} = 1, && 1 \leq i \leq N \end{aligned} \tag{3.2}$$

  Onde y_t é o estado atual e y_{t+1} é o estado no próximo instante.

- O conjunto de distribuições de probabilidade de emissão dos símbolos em cada estado é dado por $\mathbf{b} = \{b_j(\mathbf{s_k})\}$, onde $b_j(\mathbf{s_k})$ é a probabilidade de emissão do símbolo $\mathbf{s_k}$ dado que o estado emissor seja

e_j.

$$\begin{aligned} &b_j(\mathbf{s_k}) = p\{\mathbf{x_t} = \mathbf{s_k} | y_t = e_j\}, \qquad 1 \le j \le N, \quad 1 \le k \le M \\ &b_j(\mathbf{s_k}) \ge 0 \\ &\textstyle\sum_{k=1}^{M} b_j(\mathbf{s_k}) = 1, \qquad\qquad 1 \le j \le N \end{aligned} \tag{3.3}$$

onde $\mathbf{x_t}$ representa a observação no instante atual.

Caso as observações pertençam a um espaço contínuo, então devem ser usadas funções densidade de probabilidade (pdfs) para cada estado. Geralmente as pdfs são aproximadas por uma soma ponderada de distribuições Gaussianas $\mathcal{N}$:

$$b_j(\mathbf{s_k}) = \sum_{m=1}^{\kappa} c_{jm} \mathcal{N}(\boldsymbol{\mu}_{\mathbf{jm}}, \boldsymbol{\Sigma}_{\mathbf{jm}}, \mathbf{s_k}) \tag{3.4}$$

onde os índices m e j fazem referência a m-ésima gaussiana pertencente ao estado e_j, κ é a quantidade de gaussianas utilizada na soma, $\boldsymbol{\mu}_{\mathbf{jm}}$ é o vetor de média, e $\boldsymbol{\Sigma}_{\mathbf{jm}}$ a matriz de covariância. c_{jm} são os coeficientes de ponderação de cada gaussiana e

$$\begin{aligned} &c_{jm} \ge 0, \quad 1 \le j \le N, \quad 1 \le m \le \kappa \\ &\sum_{m=1}^{\kappa} c_{jm} = 1, \quad 1 \le j \le N \end{aligned} \tag{3.5}$$

- A probabilidade inicial de cada estado $\boldsymbol{\pi} = \{\pi_i\}$, onde π_i é a probabilidade de que o modelo esteja no estado e_i no instante de tempo $t = 0$.

$$\pi_i = p\{y_1 = e_i\}, \quad 1 \le i \le N \tag{3.6}$$

3.2.2 Exemplo

Como exemplo, tem-se dois dados, um de 6 lados e outro de 20 lados, que são lançados várias vezes de forma oculta a um visualizador. Este, por sua vez, só tem acesso a sequência de números gerados pelo lançamento dos dados. O que determina qual dado será lançado é o lançamento de uma moeda, onde cara implica o lançamento do dado de 6 lados e coroa o lançamento do dado de 20 lados. O primeiro lançamento sempre é feito com o dado de 6 lados.

Neste exemplo, temos um HMM com dois estados ($E = \{e_1, e_2\}$, onde o primeiro estado representa o lançamento do dado de 6 lados e o segundo estado representa o lançamento do dado de 20 lados. Sendo assim o espaço de símbolos emitidos pelos estados é discreto, unidimensional, e varia de 1 a 20.

$$S = \{1, 2, \ldots, 20\}$$

Como o primeiro estado representa o lançamento do dado de 6 lados, este possui uma probabilidade de emissão de 1/6 para os símbolos $1, 2, 3, 4, 5, 6$ e zero para os símbolos restantes. Logo

$$\begin{cases} b_1(s) = 1/6, & 1 \leq s \leq 6 \\ b_1(s) = 0, & 7 \leq s \leq 20 \end{cases}$$

Da mesma forma para o dado de 20 lados tem-se a probabilidade de emissão de 1/20 para todos os símbolos quando o processo se encontra no segundo estado, logo

$$b_2(s) = 1/20, \quad 1 \leq s \leq 20$$

Como a transição dos estados é determinada pelo lançamento de uma moeda, tem-se 50% de chance do mesmo dado ser lançado (o estado continua o mesmo) e 50% de chance de que ocorra uma mudança de dados (troca de estados), logo a matriz de transição é

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Devido ao primeiro lançamento ser feito sempre com o dado de 6 lados, então

$$\boldsymbol{\pi} = \{1, 0\}$$

A figura 3.2 ilustra o HMM do exemplo, onde tem-se dois estados, e cada estado pode realizar uma transição para ele próprio ou para o outro. Inicialmente há apenas uma transição para o primeiro estado, e cada estado emite uma observação x de acordo com a probabilidade de emissão $b_j(s_k)$).

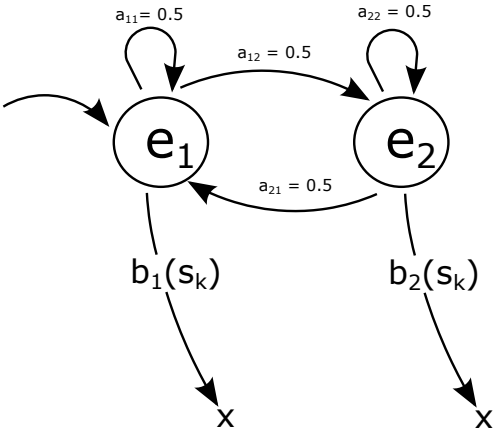


Figura 3.2: Hmm para o exemplo dos dados.

3.2.3 Tipos de HMMs

A matriz **A** de um modelo λ, ou seja, as conexões entre os estados do HMM definem a estrutura do modelo. Há dois tipos principais que possuem propriedades bem definidas: os modelos totalmente conectados e os modelos do tipo esquerda-direita. O primeiro possui a propriedade de que qualquer estado pode ser alcançado (em um único intervalo de tempo) a partir de qualquer outro estado, logo todos os coeficientes a_{ij} são positivos.

No modelo esquerda-direita ou modelo Bakis nenhuma transição é permitida para estados cujo índice seja menor que o índice do estado atual, ou seja, a transição entre os estados ocorre sempre da esquerda para a direita. Além disso a sequência de estados deve começar no estado 1 e terminar no estado N. Logo, para um modelo esquerda-direita temos:

$$a_{ij} = 0, \quad j < i \tag{3.7}$$

$$a_{NN} = 1 \tag{3.8}$$

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{3.9}$$

Geralmente algumas limitações são postas nos coeficientes de transição entre os estados dos modelos esquerda-direita, de modo a não permi-

tir transições entre estados com índices muito distantes. Uma das limitações mais utilizadas possui a forma

$$a_{ij} = 0, \quad j > i + \Delta \tag{3.10}$$

Se $\Delta = 2$, por exemplo, não é permitida transições para estados que estejam a mais de duas posições na frente do estado atual.

Além destes modelos discutidos, existem muitas outras combinações e variações possíveis. Por exemplo, modelos onde somente alguns coeficientes a_{ij} são nulos, de modo que todos os estados possam ser alcançados a partir de qualquer estado em determinado intervalo de tempo, ou modelos esquerda-direita em paralelo.

Rede de HMMs

Uma das principais aplicações dos HMMs se encontra no reconhecimento da fala, onde cada HMM é utilizado para modelar uma palavra ou um fonema, por exemplo. Neste trabalho cada HMM modela um gesto. Dessa forma os HMMs podem modelar e fazer o reconhecimento de um único gesto isolado. Uma das formas para fazer o reconhecimento de um conjunto de gestos realizados em sequência em um vídeo, por exemplo, consiste na conexão dos modelos isolados de forma a gerar uma rede de HMMs. Essa rede gerada consiste, na verdade, de um novo HMM.

A figura 3.3 representa uma rede chamada de rede de gestos, onde cada HMM modela um gesto, como os gestos barato, bonito e verdade. Cada HMM contém cinco estados emissores de símbolos (em branco) e dois estados não emissores (em cinza). Os estados não emissores funcionam como pontos de conexão entre os HMMs. Nessa rede os HMMs estão conectados em paralelo, o que significa que partindo do estado inicial até o final, somente um dos caminhos pode ser percorrido, indicando a ocorrência de apenas um gesto.

Já a figura 3.4 utiliza a rede da figura 3.3 como uma sub-rede de uma rede maior. Além dos modelos da rede de gestos, há a adição de um HMM de um estado emissor no início e no final, modelando a posição de repouso. Essa rede implica em um HMM capaz de reconhecer uma sequência 4 gestos quaisquer que estejam modelados na rede de gestos da figura 3.3.

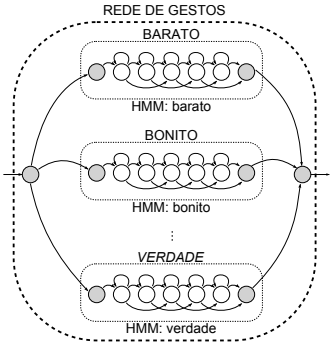


Figura 3.3: Rede de gestos.

Figura 3.4: Rede de HMMs para quatro gestos.

3.2.4 Os Três Problemas Básicos

Existem três problemas básicos que devem ser resolvidos para que os HMMs sejam úteis: a estimação, a decodificação e o treinamento.

Estimação

Dado um modelo $\lambda = (\mathbf{A}, \mathbf{b}, \boldsymbol{\pi})$ e uma sequência de observações $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_T}\}$, deve-se calcular a probabilidade de que essa sequência tenha sido gerada dado o modelo λ, ou seja, deve-se calcular $p\{\mathbf{X}|\lambda\}$. A solução deste problema permite que, por exemplo, dentre vários modelos seja escolhido o com maior $p\{\mathbf{X}|\lambda\}$.

$$p\{\mathbf{X}|\lambda\} = \sum_{\text{para todo } \mathbf{y}} \left(\prod_{t=1}^{T} b_{y_t}(\mathbf{x_t})\pi_{y_1} a_{y_1 y_2} \ldots a_{y_{T-1} y_T}\right) \tag{3.11}$$

$p\{\mathbf{X}|\lambda\}$ pode ser calculada utilizando a equação 3.11, porém isso não é computacionalmente viável. Um método prático de calcular $p\{\mathbf{X}|\lambda\}$ é através de algoritmos recursivos como os algoritmos *forward* e *backward*, apresentados no apêndice A.1.1.

Decodificação

Dado um modelo λ e uma sequência de observações $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_T}\}$, deseja-se encontrar a sequência de estados $\mathbf{y} = \{y_1, \ldots, y_T\}$ ótima de acordo com algum critério. O problema aqui está na definição do que é uma sequência ótima, sendo utilizado normalmente dois critérios: no primeiro determina-se os estados mais prováveis em cada instante de tempo; no segundo utiliza-se o algoritmo de Viterbi para determinar a sequência de estados mais provável.

Outro algoritmo utilizado para fazer a decodificação de redes de HMMs é uma variação do algoritmo de Viterbi chamada de *Token Passing Model*, através do qual pode-se obter, além da sequência de estados mais provável, a sequência de HMMs mais provável e os instantes em que ocorreram as transições de um HMM para outro.

Detalhes sobre a decodificação são apresentados no apêndice A.1.2.

Treinamento

Consiste em otimizar os parâmetros λ do modelo de modo a maximizar $p\{\mathbf{X}|\lambda\}$, sendo $\mathbf{X}$ uma sequência de treinamento. Não é conhecido nenhum método analítico para resolver esse problema, no entanto, podemos escolher λ de modo que $p(\mathbf{X}|\lambda)$ seja localmente maximizado através de um procedimento iterativo como a re-estimação de Baum-Welch (apêndice A.1.3).

3.3 Redes Neurais Artificiais

ANN é uma ferramenta matemática inspirada no funcionamento do cérebro, muito utilizada no reconhecimento de padrões. Consiste de múltiplas unidades, chamadas de neurônios, que são conectadas através de pesos variáveis. A aprendizagem ocorre pela variação desses parâmetros. [39].

3.3.1 Arquitetura

Os neurônios são as unidades das quais as redes neurais são compostas. Podem ser de diversos tipos, de acordo com a função que modela o seu comportamento, também chamada de função de ativação. Os

neurônios são agrupados em camadas e conectados entre si. A arquitetura da rede indica a forma como a rede é construída, como o número de camadas, número de neurônios por camada e a ligação entre os neurônios.

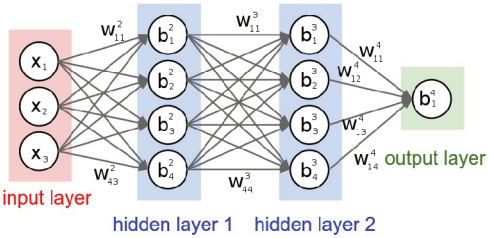


Figura 3.5: Arquitetura de uma rede feedforward com 4 camadas, 3 entradas e uma saída.

Na figura 3.5 b_j^k é o bias do neurônio j da camada k. Da mesma forma, w_{ji}^k é o peso dado a entrada i do neurônio j da camada k. A camada de entrada possuí neurônios especiais que não são modelados por uma função, eles simplesmente representam o valor do parâmetro de entrada. A camada de saída é a responsável por fazer a classificação. As camadas intermediárias são chamadas de camadas ocultas.

Quando a saída de uma camada é usada como entrada para a próxima camada, temos uma rede do tipo *feedforward*. Quando há realimentação temos uma rede do tipo *recurrent*.

3.3.2 Tipos de Neurônios

Os neurônios são unidades com múltiplas entradas x, cada uma associada a um peso w, e somente uma saída, como ilustrado na figura 3.6. Existem vários tipos de neurônios, que são classificados de acordo com a sua função de ativação, ou seja, o modo como eles processam os dados das suas entradas. Alguns dos tipos mais importantes são os sigmoides, RELU (*rectified linear unit*) e *softmax*.

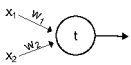


Figura 3.6: Exemplo de um neurônio com duas entradas.

A variável z armazena a soma das entradas do neurônio ponderada pelo seu respectivo peso, e é dada por:

$$z = \mathbf{w} \cdot \mathbf{x} + b \tag{3.12}$$

Onde $\mathbf{w}$, $\mathbf{x}$ e b são os vetores dos pesos, das entradas e o *bias* do respectivo neurônio.

Sigmoides

Esses neurônios possuem uma saída diferenciável e limitada entre zero e um (figura 3.7), dada por 3.13:

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.13}$$

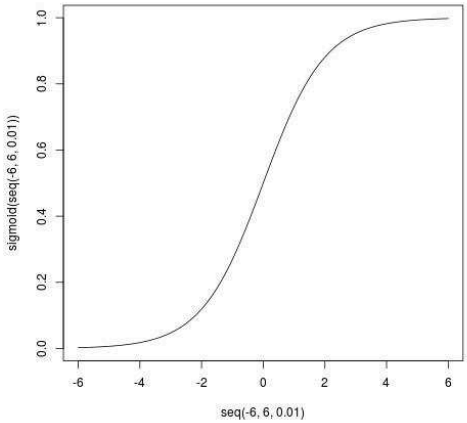


Figura 3.7: Saída de um sigmoide.

RELU

A saída da função de ativação RELU é mostrada na figura 3.8, e é dada por

$$\text{relu}(z) = \max(0, z) \tag{3.14}$$

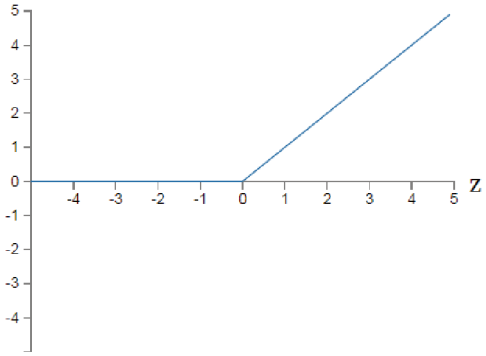


Figura 3.8: Função de ativação relu(z).

Trabalhos recentes em reconhecimento de imagens ([40], [41], [42]) relataram aprimoramento do reconhecimento com a utilização da função relu em substituição a sigmoide, porém a causa desse aprimoramento ainda não é clara.

Softmax

São neurônios com uma função de ativação chamada *softmax*. Esses neurônios são geralmente utilizados na camada de saída pelo fato de que a sua saída pode ser interpretada como uma distribuição de probabilidade. A função de ativação é dada pela equação 3.15, onde L é a camada de saída.

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \tag{3.15}$$

De acordo com 3.15, a_j^L é sempre positiva e a soma das saídas de uma camada de neurônios *softmax* é sempre 1:

$$\sum_j a_j^L = 1 \tag{3.16}$$

Essa função é útil sempre que for necessário interpretar a saída da rede neural como probabilidades. Os neurônios com função de ativação do tipo softmax geralmente são utilizados em conjunto com a função de custo log-likelihood.

3.3.3 Aprendizagem

Sendo x uma amostra de entrada utilizada para treinamento e $\mathbf{y}(x) = \mathbf{y}$ o vetor de saída esperado para a amostra x (supondo que a camada de saída possua mais de um neurônio).

Precisamos achar os valores dos pesos w e dos biases b de modo que a saída da rede se aproxime da saída desejada $\mathbf{y}$ para todas as amostras x. Para calcular essa semelhança devemos utilizar algum tipo de métrica de erro entre a saída desejada e obtida, também chamada de função de custo. Uma das mais comuns é o Mean Squared Error (MSE), dado por 3.17:

$$C = \frac{1}{2n} \sum_{x} \|\mathbf{y} - \mathbf{a}\|^2 \tag{3.17}$$

Onde C é a função de custo, $\mathbf{y}$ é o vetor de saída desejado para a amostra x, $\mathbf{a}$ é o vetor de saída gerado pela rede para a amostra x, $\|\mathbf{v}\|$ é o módulo do vetor $\mathbf{v}$, n é o número de amostras x utilizadas. O somatório é sobre todas as amostras x.

O que a função 3.17 faz é calcular a média quadrática da diferença entre a saída desejada e a obtida para todas as amostras utilizadas. O ideal seria que essa função retornasse zero, pois isso indica que não existe diferença entre a saída desejada e a obtida.

Dessa forma, devemos encontrar um conjunto de pesos e biases que minimizem o erro da rede, ou seja, devemos achar um conjunto de parâmetros que minimizem C. Isso pode ser obtido através de um algoritmo chamado *Gradient Descent* (dedução no apêndice B.1):

$$w_k \leftarrow w_k - \eta \frac{\partial C}{\partial w_k} \tag{3.18}$$

$$b_l \leftarrow b_l - \eta \frac{\partial C}{\partial b_l} \tag{3.19}$$

A equação 3.18 nos diz que devemos atualizar o valor de determinado peso w_k pelo seu próprio valor subtraído da variação da função de custo C em relação a uma variação em w_k multiplicada por um fator η chamado de taxa de aprendizagem. O mesmo vale para a equação 3.19. A taxa de aprendizagem pode ser vista como um atenuador sobre a variação dos parâmetros. Se for muito baixa, a aprendizagem da rede será muito lenta. Por outro lado, se for muito alta, o algoritmo pode

divergir do mínimo, levando a valores de erro cada vez maiores, não gerando aprendizado. Um método eficiente para se calcular as derivadas das equações 3.18 e 3.19 é através do algoritmo *backpropagation*, apresentado no apêndice B.2.

Embora o *Gradient Descent* possa calcular os valores dos pesos e biases, o tempo de treinamento pode ser muito longo se a quantidade de amostras de treinamento for grande. Na prática é utilizada uma variação chamada de Stochastic Gradient Descent (SGD) (descrição no apêndice B.1.4) que diminui o tempo de treinamento consideravelmente. O SGD diz que deve-se escolher aleatoriamente um subconjunto do conjunto de amostras de treinamento, chamado de *mini-batch*, e utilizá-lo para o treinamento. Então deve-se escolher outro mini-batch e repetir o processo, até que se esgotem as amostras de treinamento. Quando todas as amostras são utilizadas para o treinamento completa-se uma época.

3.3.4 Funções de Custo

Apesar do MSE ser uma função de custo utilizada na prática, ele apresenta problemas na velocidade de aprendizagem da rede quando se utilizam neurônios do tipo sigmoide, pois a aprendizagem é proporcional a derivada da saída do neurônio $\sigma'(z)$, que é muito baixa na região de saturação do neurônio, como pode ser visto na figura 3.7. Outra função de custo muito utilizada é a *cross-entropy*, pois gera uma aprendizagem mais rápida que o MSE para neurônios sigmoides. Uma análise detalhada é dada no apêndice B.4.

Log-Likelihood

Essa função de custo é utilizada em conjunto com neurônios softmax, e é dada por:

$$C = \frac{-1}{n} \sum_x \sum_y y_j \ln a_j^L \tag{3.20}$$

Onde o somatório é sobre todas as amostras x e todos os neurônios da última camada, e a_j^L é a saída do neurônio j da última camada (camada L) com função de ativação do tipo softmax.

Para entender o funcionamento dessa função, imagine que a rede deve classificar uma determinada amostra x pertencente a classe y_c.

Logo $y_j = 0$ se $j \neq c$ e $y_c = 1$. Então a função se resume a

$$C = -\ln a_c^L$$

Importante observar que a_c^L é a probabilidade que a rede calcula de que a saída seja y_c, então a parcela do custo relativa aos outros neurônios já está implícita em a_c^L. Por isso o custo total pode ser calculado utilizando apenas a saída do neurônio a_c^L. Se a rede gerar um valor alto a_c^L quando a saída é y_c, então a função de custo gerará um valor baixo. Caso a rede produza um valor baixo de a_c^L quando a saída é y_c, então o custo será alto.

3.4 Redes Neurais Convolucionais (CNNs)

São redes neurais com uma arquitetura especialmente adaptada para o reconhecimento de imagens. Devido a sua estrutura, CNNs também facilitam o treinamento de redes com várias camadas, também chamadas de redes neurais profundas. Essas redes baseiam-se em três ideias básicas: campos receptivos locais, pesos compartilhados e camadas de pooling.

3.4.1 Campo Receptivo Local

Nas redes neurais convencionais os neurônios de entrada são organizados em uma coluna e são conectados a todos os neurônios da primeira camada oculta. Em uma CNN os neurônios de entrada são organizados em uma matriz, correspondendo ao nível de cinza de uma imagem, por exemplo. Uma imagem de 28 x 28 pixels seria representada por uma camada de entrada de 28 x 28 neurônios, como ilustrado na figura 3.9.

Como feito na redes neurais convencionais, essa camada de entrada é conectada a primeira camada oculta. Porém a diferença é que agora os neurônios de entrada não são conectados a todos os neurônios da camada oculta, mas as ligações ocorrem apenas em pequenas regiões. Ou seja, cada neurônio da primeira camada oculta será conectado a uma pequena região da camada de entrada. Como exemplo, a figura 3.10 mostra um determinado neurônio oculto ligado a uma região de tamanho 5 x 5 da camada de entrada.

Essa região na qual o neurônio oculto está conectado é chamada

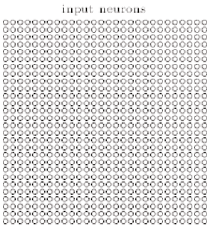


Figura 3.9: Camada de entrada correspondente a uma imagem de 28 x 28 pixels.

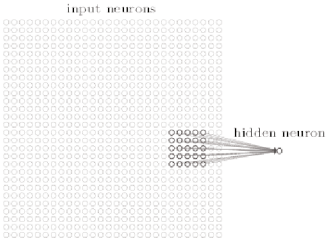


Figura 3.10: Um neurônio oculto ligado a uma região de tamanho 5 x 5 da camada de entrada.

de seu campo receptivo local, e pode ser interpretada como a região que determinado neurônio deve aprender a analisar. Cada uma dessas conexões possuem um peso e o neurônio também possui um bias, como nas redes convencionais.

Devemos, então, deslocar o campo receptivo local por toda a camada de entrada, possuindo um neurônio oculto para cada deslocamento, como ilustrado na figura 3.11. O tamanho do deslocamento do campo receptivo local é chamado de *stride length* e pode variar, indicando o tamanho da sobreposição da região que cada neurônio oculto deve analisar.

3.4.2 Pesos Compartilhados

Cada neurônio da camada oculta possuí um bias e os pesos correspondentes as ligações do seu campo receptivo local. Porém cada um dos neurônios dessa camada oculta compartilharão os mesmos valores para os pesos e biases, reduzindo enormemente a quantidade de parâmetros envolvidos em uma CNN. Dessa forma todos os neurônios dessa ca-

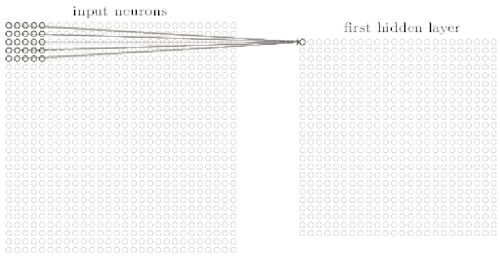


Figura 3.11: Camada oculta gerada a partir do deslocamento do campo receptivo local sobre a camada de entrada.

mada possuirão os mesmos parâmetros, e a saída do neurônio oculto na posição j, k será:

$$\sigma\left(b + \sum_{l=0}^{d-1}\sum_{m=0}^{d-1} w_{l,m} a_{j+l,k+m}\right) \tag{3.21}$$

Onde σ é a função de ativação do neurônio, d é o tamanho do campo local receptivo, b é o bias compartilhado, $w_{l,m}$ é uma matriz d x d contendo os pesos compartilhados, e $a_{x,y}$ é a entrada na posição x, y.

O compartilhamento dos parâmetros para todos os neurônios dessa camada oculta significa que todos os neurônios dessa camada são treinados para detectar exatamente o mesmo atributo, porém em regiões diferentes da imagem, o que faz com que as CNNs sejam naturalmente invariantes a translação. Devido a essa característica, chama-se esse conjunto de neurônios ocultos de mapa de atributo (*feature map*) ou canal. Aos pesos e a bias compartilhados que definem esse mapa chamam-se de kernel ou filtro.

Como visto até agora, a camada oculta é composta de neurônios compartilhando os mesmos parâmetros, e é capaz de identificar apenas um tipo de atributo. Para que possa ser feito um reconhecimento de uma imagem é necessário que se possa fazer o reconhecimento de vários atributos, portanto uma camada convolucional é composta por vários mapas de atributo ou vários canais. A figura 3.12 mostra uma camada convolucional composta por 3 mapas de atributo compostos por 24 x 24 neurônios cada. Cada mapa possui um kernel composto de 5 pesos compartilhados e um bias compartilhado, e foram gerados com um *stride length* igual a 1. O resultado é uma rede capaz de detectar 3

tipos de características em toda a imagem.

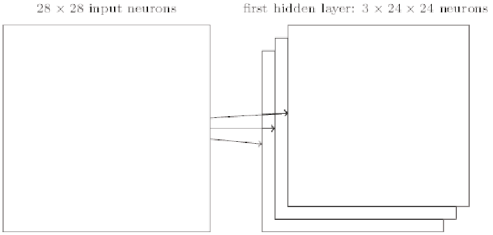


Figura 3.12: Camada convolucional com 3 mapas de atributo.

Importante observar na figura 3.12 que a camada convolucional tem como entrada uma camada de um único canal, e a equação 3.21 pode ser usada. Porém quando a entrada da camada convolucional possuir mais de um canal, como uma imagem rgb ou uma camada convolucional com vários mapas de atributo, seu campo local receptivo será uma matriz tridimensional, com profundidade igual ao número de canais de sua camada de entrada. Logo, para uma camada de entrada com múltiplos canais, a equação de 3.21 torna-se:

$$\sigma\left(b + \sum_{g=1}^{h}\sum_{l=0}^{d-1}\sum_{m=0}^{d-1} w_{g,l,m}a_{g,j+l,k+m}\right) \tag{3.22}$$

Onde σ é a função de ativação do neurônio, d é o tamanho do campo local receptivo, h é o número de canais ou mapas de atributos da camada de entrada, b é o bias compartilhado, $w_{g,l,m}$ é uma matriz h x d x d contendo os pesos compartilhados, e $a_{z,x,y}$ é a entrada na posição x, y do canal ou mapa de atributo z.

3.4.3 Camadas de Pooling

Após uma camada convolucional geralmente existe uma camada de pooling para compactar a informação da camada convolucional. Cada unidade da camada de pooling compacta uma região de tamanho n x n, sem sobreposição, da camada convolucional anterior. Um dos métodos utilizados é o chamado *max-pooling*, em que cada unidade da camada de pooling retém apenas o valor máximo entre as saídas em uma região n x n da camada convolucional.

Na figura 3.13 a camada convolucional possuí 24 x 24 neurônios. Utiliza-se o método *max-pooling* em uma região 2 x 2 para gerar a camada de pooling com tamanho 12 x 12.

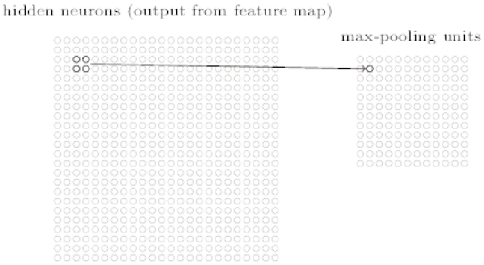


Figura 3.13: Max-pooling com uma região 2 x 2.

Como a camada convolucional possuí vários mapas de atributo, a camada de pooling possuirá um número igual de mapas, como mostrado na figura 3.14.

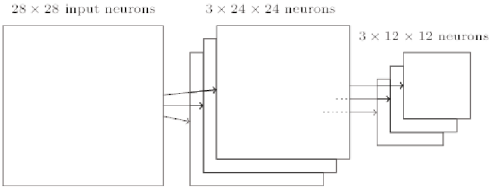


Figura 3.14: Camada de entrada com uma camada oculta em uma CNN.

3.4.4 Uma CNN Completa

A figura 3.15 mostra uma CNN com 28 x 28 neurônios na camada de entrada, uma camada convolucional com 3 mapas de atributo seguida de uma camada de pooling. No final da rede temos uma camada de saída como utilizada nos neurônios convencionais, onde cada neurônio dessa camada é conectado a todos os neurônios da camada de pooling anterior. Mais informações sobre as estruturas das CNNs são dadas na seção 5.1.2.

O treinamento de uma CNN pode ser feito utilizando-se os algoritmos SGD e *backpropagation* (apêndice B), devidamente modificados para as CNNs.

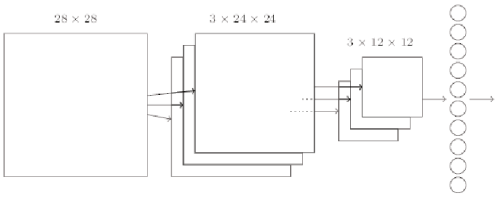


Figura 3.15: Uma CNN de uma camada.

3.5 Descritores

Os descritores são informações, extraídas dos vídeos, que podem ser utilizadas pelo sistema para fazer a discriminação entre os diferentes tipos de gestos. Esta seção apresenta os descritores que foram implementados no sistema, e foram escolhidos por serem utilizados em outros trabalhos e por serem descritores que podem ser extraídos rapidamente, viabilizando a implementação do sistema de reconhecimento em tempo real. Embora todos os descritores possam ser utilizados pelo sistema de reconhecimento, o número limitado de amostras para o treinamento dos HMMs pode diminuir o desempenho do sistema, pois a pequena quantidade de amostras de treinamento pode ser insuficiente para treinar corretamente os modelos para a grande variação de parâmetros existente devido a utilização de tantos descritores. Por esse motivo, no capítulo 4.4 foram feitos testes com diferentes combinações desses descritores para revelar quais geram a melhor taxa de reconhecimento. Os descritores foram divididos em dois grupos principais, os descritores de movimento, apresentados na seção 3.5.1, e os descritores de forma da mão, apresentados na seção 3.5.2.

3.5.1 Descritores de Movimento

Os descritores de movimento são aqueles que armazenam informação sobre o movimento das mãos, como sua posição, velocidade, aceleração, direção, etc, e são descritos a seguir.

Posição das Mãos em Relação ao Centro da Face

A posição das mãos carrega a informação do lugar onde as mãos se encontram em determinado momento do gesto, se as mãos estão na

barriga ou acima da cabeça, se estão lado a lado ou uma está acima da outra, etc. Para extrair a posição das mãos é necessário um sistema de coordenadas, aqui escolhido como sendo um sistema radial com origem no centro de massa da face. O centro de massa da face foi escolhido como a origem do sistema de coordenadas devido a sua baixa variação de posição. Além da origem do sistema de coordenadas também é necessário fazer uma normalização desse sistema, pois a distância entre as mãos e a face varia com a distância que o usuário se posiciona em relação a câmera. O fator de normalização escolhido foi a largura da face L_{face}. Então a posição das mãos é descrita pela sua distância $r_{\text{mão,face}}$ e pelo seu ângulo $\alpha_{\text{mão,face}}$ em relação ao centro de massa da face, dados por:

$$r_{\text{mão,face}} = \frac{\sqrt{(C_{x,\text{mão}} - C_{x,\text{face}})^2 + (C_{y,\text{mão}} - C_{y,\text{face}})^2}}{L_{\text{face}}} \tag{3.23}$$

$$\alpha_{\text{mão,face}} = arctan(\frac{C_{y,\text{mão}} - C_{y,\text{face}}}{C_{x,\text{mão}} - C_{x,\text{face}}}) \tag{3.24}$$

Onde $C_{x,\text{mão}}$, $C_{y,\text{mão}}$, $C_{x,\text{face}}$ e $C_{y,\text{face}}$ são a posição x e y do centro de massa da mão e a posição x e y do centro de massa da face, respectivamente.

Direção do Movimento

A direção do movimento é dada pelo ângulo formado pela posição da mão no quadro atual e no quadro anterior. Esse descritor armazena informação sobre a forma da trajetória que a mão descreve durante um gesto e é dado por:

$$\theta_t = arctan(\frac{C_{t,y,\text{mão}} - C_{t-1,y,\text{mão}}}{C_{t,x,\text{mão}} - C_{t-1,x,\text{mão}}}) \tag{3.25}$$

Onde $C_{t,y,\text{mão}}$, $C_{t-1,y,\text{mão}}$, $C_{t,x,\text{mão}}$ e $C_{t-1,x,\text{mão}}$ são a posição y do centro de massa da mão no quadro atual e no anterior, e a posição x do centro de massa da mão no quadro atual e no quadro anterior, respectivamente.

Deslocamento e sua Primeira Diferença

O deslocamento mede a variação da posição da mão entre um quadro e outro, e representa a velocidade da mão tendo como base de tempo o quadro. O deslocamento é calculado por

$$D_t = \frac{\sqrt{(C_{t,x,\text{mão}} - C_{t-1,x,\text{mão}})^2 + (C_{t,y,\text{mão}} - C_{t-1,y,\text{mão}})^2}}{L_{\text{face}}} \tag{3.26}$$

Onde L_{face} é o fator de normalização (largura da face).

A primeira diferença do deslocamento representa a aceleração da mão entre dois quadros e é dada por

$$A_t = D_t - D_{t-1} \tag{3.27}$$

O deslocamento e sua primeira diferença são descritores úteis, pois trazem a informação dos momentos de paradas da mão durante um sinal. Por exemplo, um sinal com uma trajetória circular tende a manter seu deslocamento constante e sua aceleração nula, diferente de um sinal com uma trajetória retangular. Além disso, esses descritores também são usados no cálculo da posição do centro de massa do próximo quadro, utilizado para mover a região de interesse na etapa de classificação dos pixels (seção 4.1).

3.5.2 Descritores da Forma da Mão

Os descritores da forma da mão não fornecem informações sobre o movimento da mão durante o gesto, mas sim informações sobre o formato da mão. Se utilizados em conjunto com os descritores de movimento obtém-se a informação do formato da mão em determinado momento da trajetória do gesto.

Alongamento

O alongamento Alg_t é um descritor invariante à escala, translação e rotação, e é definido com sendo a taxa entre o maior e o menor lado do menor retângulo de contorno da mão.

Retangularidade

A área da mão não é um descritor invariante à escala, porém ao dividir-se a área da mão pela área do menor retângulo de contorno obtém-se a retangularidade Ret_t, que é um descritor invariante à escala e está no intervalo (0,1], sendo 1 quando o o objeto for um retângulo.

Compactação

A compactação é um descritor independente de transformações lineares e é dado pelo quadrado da perímetro de uma região dividido pela sua área. A região mais compacta no espaço euclidiano é o círculo.

$$Cpt_t = \frac{\text{perímetro}^2}{\text{área}} \tag{3.28}$$

Direção

A direção é medida como sendo o ângulo do maior lado do menor retângulo de contorno da mão e pode ser calculada como

$$\phi_t = \frac{1}{2} arctg\left(\frac{2\mu_{11}}{\mu_{20} - \mu 02}\right) \tag{3.29}$$

μ_{pq} são os momentos centrais da região da mão, dados por

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i,j) \tag{3.30}$$

Onde x_c e y_c são as coordenadas do centro de massa do objeto, i e j são as coordenadas dos pixels na imagem, e $f(i,j)$ o valor do pixel nas coordenadas i, j.

Momentos de uma Região

Considerando uma imagem como sendo uma probabilidade de massa de uma variável aleatória 2D, $f(i,j)$ (onde i, j são as posições dos pixels na imagem), pode-se extrair propriedades dessa variável aleatória através de características estatísticas chamadas **momentos**. Assumindo que os pixels de uma imagem com valores diferentes de zero representem regiões, pode-se utilizar os momentos para descrever essas

regiões. Os momentos são dependentes de transformações lineares do nível de cinza das regiões. Para eliminar essa dependência trabalha-se com a versão binária da imagem, onde a função probabilidade de massa $f(i,j) = 1$ nos pixels de uma região e $f(i,j) = 0$ nos pixels fora. [43]

Um momento de ordem $(p+q)$ é dependente da escala, translação e rotação,e para imagens digitalizadas é dado por

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i,j) \tag{3.31}$$

onde i,j são as coordenadas dos pixels e $f(i,j)$ é o valor do pixel na coordenada i,j.

Momentos Centrais não Escalados

São momentos invariantes à traslação e escala. A invariância à translação pode ser obtida através dos momentos centrais, dados por

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i,j) \tag{3.32}$$

onde x_c, y_c são as coordenadas do centro de massa da região, que são calculadas da seguinte forma

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}} \tag{3.33}$$

A invariância à escala é obtida através dos momentos centrais não escalados ϑ_{pq}, dados por

$$\vartheta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma}, \quad \text{onde} \quad \gamma = \frac{p+q}{2} + 1 \tag{3.34}$$

Momentos de Hu

Através dos momentos centrais não escalados é possível obter um conjunto de sete momentos invariantes à rotação, traslação e escala,

chamados de **momentos de Hu** [44] e dados por

$$\begin{aligned}
\varphi_1 &= \vartheta_{20} + \vartheta_{02},\\
\varphi_2 &= (\vartheta_{20} - \vartheta_{02})^2 + 4\vartheta_{11}^2,\\
\varphi_3 &= (\vartheta_{30} - 3\vartheta_{12})^2 + (3\vartheta_{21} - \vartheta_{03})^2,\\
\varphi_4 &= (\vartheta_{30} + \vartheta_{12})^2 + (\vartheta_{21} + \vartheta_{03})^2,\\
\varphi_5 &= (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{30} + \vartheta_{12})((\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2)\\
&\quad +(3\vartheta_{21} - \vartheta_{03})(\vartheta_{21} + \vartheta_{03})(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2),\\
\varphi_6 &= (\vartheta_{20} - \vartheta_{02})((\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2) + 4\vartheta_{11}(\vartheta_{30} + \vartheta_{12})(\vartheta_{21} + \vartheta_{03}),\\
\varphi_7 &= (3\vartheta_{21} - \vartheta_{03})(\vartheta_{30} + \vartheta_{12})((\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2)\\
&\quad -(\vartheta_{30} - 3\vartheta_{12})(\vartheta_{21} + \vartheta_{03})(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2).
\end{aligned} \tag{3.35}$$

Momentos Invariantes a Transformações Afins

Os momentos descritos anteriormente são invariantes apenas à translação, escala e rotação, porém não são invariantes à transformações afins. Um conjunto de quatro momentos invariantes à transformações afins é apresentado em [45]

$$\begin{aligned}
I_1 &= \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4},\\
I_2 &= \frac{\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{21}^3\mu_{03} - 3\mu_{21}^2\mu_{12}^2}{\mu_{00}^{10}},\\
I_3 &= \frac{\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)}{\mu_{00}^7},\\
I_4 &= \frac{A + B + C + D}{\mu_{00}^{11}},
\end{aligned} \tag{3.36}$$

onde

$$\begin{aligned}
A &= \mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2,\\
B &= 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12},\\
C &= -8\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21}^2,\\
D &= 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} + \mu_{02}^3\mu_{30}^2.
\end{aligned} \tag{3.37}$$

Momentos do Contorno de uma Região

Os momentos também podem ser extraídos a partir do contorno de uma região. Uma região representada por um contorno fechado é caracterizada por uma sequência ordenada $z(i)$ que representa a distância Euclidiana entre o centro de massa da região e seus N pontos de seu contorno. O r-ésimo momento m_r e o r-ésimo momento central μ_r do contorno da região podem ser estimados através de

$$m_r = \frac{1}{N}\sum_{i=1}^{N} z(i)^r, \tag{3.38}$$

$$\mu_r = \frac{1}{N}\sum_{i=1}^{N} (z(i) - m_1)^r. \tag{3.39}$$

Invariância à translação, rotação e escala são obtidos através do r-ésimo momento central $\overline{\mu}_r$

$$\overline{\mu}_r = \frac{\mu_r}{\mu_2^{r/2}} \tag{3.40}$$

Embora o conjunto de μ_r possa ser usado para representação de formas, os seguintes descritores são menos sensíveis ao ruído [46]

$$F_1 = \frac{\mu_2^{1/2}}{m_1}, \quad F_2 = \frac{\mu_3}{\mu_2^{3/2}}, \quad F_3 = \frac{\mu_4}{\mu_2^2}, \quad F_4 = \overline{\mu}_5. \tag{3.41}$$

Descritores de Fourier

Sendo C uma curva fechada (o contorno de uma região na imagem), deslocando-se pela curva no sentido anti-horário pode-se obter duas funções periódicas, $x(k)$ e $y(k)$, onde a primeira contém a posição x e a segunda a posição y do ponto da curva em função da distância percorrida k. Como $x(k)$ e $y(k)$ são funções periódicas, estas podem ser representadas na forma de série de Fourier [43] [47]

$$x(k) = \sum_n a_n e^{\frac{j2\pi kn}{L}}, \tag{3.42}$$

$$y(k) = \sum_n b_n e^{\frac{j2\pi kn}{L}}. \tag{3.43}$$

Onde a_n e b_n são os coeficientes da série, L é o comprimento da curva e $j = \sqrt{-1}$.

Sendo o par de coordenadas (x_k, y_k) (para $x_k = x(k)$ e $y_k = y(k)$), onde $(x_1, y_1) = (x_L, y_L)$, pode-se calcular os coeficientes a_n e b_n de acordo com

$$a_n = \frac{1}{L-1} \sum_{k=1}^{L-1} x_k e^{-j\frac{2\pi nk}{L-1}}, \tag{3.44}$$

$$b_n = \frac{1}{L-1} \sum_{k=1}^{L-1} y_k e^{-j\frac{2\pi nk}{L-1}} \tag{3.45}$$

Os coeficientes a_n e b_n não são invariantes à translação, rotação, escala e escolha do ponto inicial da curva, porém é possível torná-los. Sendo s_k o contorno de uma região, na tabela 3.1 são mostrados os coeficientes de Fourier para algumas transformações. [48]

Transformação	Contorno	Coeficientes de Fourier
Identidade	s_k	a_n
Rotação	$s'_k = s_k e^{j\theta}$	$a'_n = a_n e^{j\theta}$
Translação	$s''_k = s_k + \Delta_{xy}$	$a''_n = a_n + \Delta_{xy}\delta(n)$
Escalamento	$s'''_k = \alpha s_k$	$a'''_n = \alpha a_n$
Ponto inicial	$s''''_k = s(k - k_o)$	$a''''_n = a_n e^{-j2\pi k_o n/L}$

Tabela 3.1: Propriedades da Série de Fourier.

Onde $\delta(n) = 1$ para $n = 0$ e $\delta(n) = 0$ para $n \neq 0$.

Se a curva s_k com coeficientes de Fourier a_n sofrer uma translação de Δ_{xy}, um escalamento de α, uma rotação de θ e uma mudança do ponto inicial de k_o pontos tem-se que, de acordo com a tabela 3.1, os coeficientes da curva transformada será

$$a'_n = \alpha e^{-j(2\pi k_o n/L+\theta)}(a_n + \Delta_{xy}\delta(n)) \tag{3.46}$$

Desconsiderando o coeficiente a_0 consegue-se invariância à translação, obtendo-se

$$a'_n = \alpha a_n e^{-j(2\pi k_o n/L+\theta)}, \quad n \neq 0 \tag{3.47}$$

A invariância à rotação e à mudança do ponto inicial pode ser obtida

fazendo-se

$$|a'_n|^2 = \alpha^2 |a_n|^2 |e^{-j(2\pi k_o n/L+\theta)}|^2 = \alpha^2 |a_n|^2, \quad n \neq 0 \tag{3.48}$$

Finalmente, a invariância ao escalamento é obtida através de

$$\frac{|a'_n|^2}{|a'_1|^2} = \frac{\alpha^2 |a_n|^2}{\alpha^2 |a_1|^2} = \frac{|a_n|^2}{|a_1|^2}, \quad n \neq 0 \tag{3.49}$$

As transformações das equações 3.47, 3.48 e 3.49 podem ser usadas nos coeficientes de Fourier a_n e b_n obtidos nas equações 3.44 e 3.45. Em [47] o descritor r_n (invariante à translação, rotação e mudança do ponto inicial) é obtido a partir de a_n e b_n

$$r_n = \sqrt{|a_n|^2 + |b_n|^2}, \quad n \neq 0 \tag{3.50}$$

E o descritor ω_n (invariante à translação, rotação, mudança do ponto inicial e escala) é obtido a partir de r_n

$$\omega_n = \frac{r_n}{r_1}, \quad n \neq \begin{cases} 0 \\ 1 \end{cases} \tag{3.51}$$

Como $\omega_1 = 1$ não é necessário calculá-lo. Em [47] os descritores ω_n foram utilizados para o reconhecimento de caracteres e foi observado que os primeiros 10-15 descritores são suficientes para a representação dos caracteres.

3.5.3 Resumo dos Descritores Utilizados

A tabela 3.2 lista todos os descritores utilizados neste trabalho, que são divididos em dois grupos: os que carregam informações sobre o movimento realizado pelas mãos e os que carregam informações sobre o formato da mão.

Símbolo	Descritor
$r_{\text{mão,face}}$	Distância da mão em relação ao centro da face
$\alpha_{\text{mão,face}}$	Ângulo da mão em relação ao centro da face
θ_t	Direção do movimento da mão
D_t	Deslocamento da mão
A_t	Primeira diferença do deslocamento da mão
Alg_t	Alongamento da mão
Ret_t	Retangularidade da mão
Cpt_t	Compactação da mão
ϕ_t	Direção da mão
ϑ	Momentos centrais não escalados
φ	Momentos de HU
I	Momentos invariantes a transformações afins
F	Momentos do contorno de uma região
ω	Descritores de Fourier

Tabela 3.2: Descritores implementados.

CAPÍTULO 4

Metodologia

Um sistema de reconhecimento de gestos envolve várias etapas, sendo que o sistema desenvolvido nesse projeto foi dividido em três módulos: rastreamento, extração dos descritores e classificador (figura 4.1). O módulo de rastreamento é responsável por detectar, segmentar e rastrear as mãos e a face durante todos os frames do vídeo. O módulo de extração dos descritores extraí vários tipos de descritores das imagens segmentadas geradas pelo módulo anterior. Finalmente o módulo classificador, composto por HMMs, utiliza os descritores gerados para classificar os gestos realizados no vídeo.

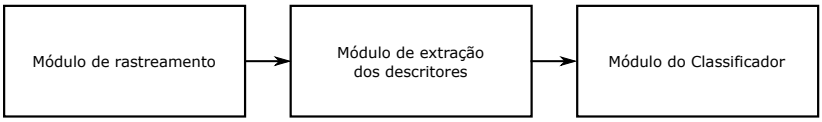


Figura 4.1: Módulos do sistema.

Na seção 4.1 é apresentado o módulo de rastreamento. Na seção 4.2 é discutido brevemente o módulo de extração dos descritores. Na seção 4.3 é apresentado o classificador e como os gestos são modelados por HMMs. Na seção 4.4 são realizados diversos testes para deter-

minar o número de estados e os descritores mais promissores para o reconhecimento dos gestos.

4.1 Módulo de Rastreamento

O módulo de rastreamento é responsável por fazer a detecção, segmentação e rastreamento das mãos e da face em cada quadro. Para facilitar a detecção são utilizadas luvas amarelas, e controla-se o fundo da imagem para que não contenha cores similares à cor da pele ou das luvas. O usuário também deve usar camiseta de manga comprida e cor diferente da cor da luva e da pele.

Este módulo foi desenvolvido de forma a ser executado em tempo real tirando vantagens das limitações impostas acima. O módulo primeiramente classifica os pixels da imagem de acordo com a sua cor, e depois detecta os pixels com a mesma cor que estejam conectados e os classifica como objetos. Então, dentre os objetos detectados, algumas regras são utilizadas para selecionar quais correspondem às mãos e à face.

Primeiramente vamos definir o conceito de **classe**, **objeto** e **região de interesse** no contexto deste trabalho.

4.1.1 Classe e Objeto

É importante deixar claro o que definimos como classe e objeto. A **classe** refere-se a semelhança entre a cor dos pixels. Por exemplo, pixels da face possuem cor diferente dos das mãos (luvas) e do fundo, pertencendo a classes diferentes. Logo, determinada classe está relacionada a determinada gama de cores. As classes utilizadas nesse trabalho representam a cor do fundo, da pele e da luva, e são dadas por $\mathcal{C} = \{i_{fundo}, i_{pele}, i_{luva}\}$.

Já um **objeto** refere-se a área formada pela conexão dos pixels (com pelo menos um de seus vizinhos-de-4, ver Figura 4.2) da mesma classe, como ilustrado na Figura 4.3. Quando os objetos são detectados, eles são inicialmente rotulados como objeto $k_1, k_2, ...$, e então, para cada objeto do conjunto, é verificado se ele é um dos objetos de interesse ou é ruído (objetos que surgem devido ao erro de classificação dos pixels). Se for considerado ruído o objeto é eliminado, caso contrário o mesmo é classificado como um dos objetos de interesse. Os objetos de interesse

neste trabalho são a face, a mão direita, a mão esquerda e as mãos sobrepostas, representados respectivamente por $\mathcal{O} = \{k_f, k_{md}, k_{me}, k_{ms}\}$. As mãos sobrepostas refere-se ao caso em que as mãos estão se tocando ou ocluindo uma a outra.

Cada objeto de interesse é composto de pixels de apenas uma classe:

- k_f é composto por pixels da classe i_{pele};
- k_{md}, k_{me} e k_{ms} são compostos por pixels da classe i_{luva}.

Figura 4.2: Vizinhos-de-4 e vizinhos-de-8. Considerando cada quadrado como um pixel, quando nos referimos aos vizinhos do pixel central devemos especificar o tipo de conexão entre eles. Os vizinhos-de-8 são os 8 pixels ao seu redor, enquanto que os vizinhos-de-4 são os 4 pixels em preto.

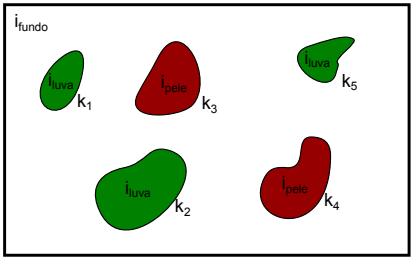


Figura 4.3: Classes e objetos. Pixels com a mesma cor pertencem à mesma classe, $i_{fundo}, i_{pele}, i_{luva}$. O conjunto de pixels com a mesma cor que estão conectados formam objetos, k_1, k_2, k_3, k_4, k_5.

4.1.2 A Região de Interesse

Em uma imagem, a probabilidade de ocorrência de pixels de determinada classe $p(i_n)$ pode ser estimada através do número de ocorrência de pixels da classe i_n sobre o número total de pixels na imagem. Portanto, como as mãos e a face ocupam uma região muito menor que o fundo da imagem, $p(i_{fundo}) >> p(i_{pele})$ e $p(i_{luva})$.

Porém, o algoritmo de classificação implementado não utiliza a imagem completa, ao invés disso, faz-se a classificação apenas em uma região onde o objeto deve estar contido, chamada de **região de interesse** $\mathcal{R}_k$, onde k é o objeto ao qual a região está relacionada. Desse modo a quantidade de pixels que devem ser classificados é muito menor, tornando a classificação mais rápida e diminuindo a probabilidade de erro, pois pixels semelhantes aos da classe do objeto, mas que não pertencem a ele, são ignorados se estiverem fora da $\mathcal{R}_k$.

A região de interesse $\mathcal{R}_k$ é uma região da imagem onde os pixels com classe i_k correspondentes ao objeto k estão encerrados. Na Figura 4.4 são mostrados 3 objetos, k_f, k_{md} e k_{me}, e suas respectivas regiões de interesse, $\mathcal{R}_f$, $\mathcal{R}_{md}$ e $\mathcal{R}_{me}$. Nota-se que como não há intersecção entre as regiões de interesse, dentro de determinada $\mathcal{R}_k$ só há pixels do fundo (i_{fundo}) e da classe i_k correspondente ao objeto k. Sendo assim, quando não há intersecção entre regiões de interesse de diferentes objetos, considerou-se que em $\mathcal{R}_k$ $p(i_{fundo}) = p(i_k)$ e $p(i_n) = 0$ para $n \neq i_{fundo}, i_k$.

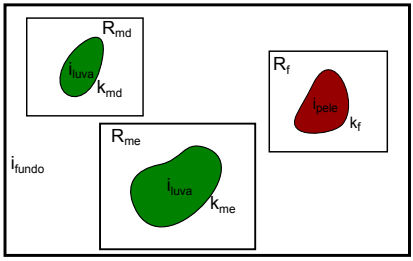


Figura 4.4: Regiões de interesse sem intersecção. Quando não há intersecção entre as regiões de interesse, dentro da região de interesse $\mathcal{R}_f$, por exemplo, devem existir somente pixels da classe i_{fundo} e da classe i_{pele}, que é a classe do objeto k_f correspondente a $\mathcal{R}_f$.

Quando há intersecção entre diferentes regiões de interesse, foi considerado $p(i_{fundo}) = p(i_n)$ para todos as classes i_n correspondentes as regiões interseccionadas. Embora essa suposição não seja verdadeira, ela garante que dentro de determinada região de interesse possam ocorrer pixels de todas as classes i_n. Por exemplo, na Figura 4.5, $\mathcal{R}_{md}$ está interseccionada com $\mathcal{R}_{me}$, logo dentro de $\mathcal{R}_{md}$ podem ocorrer pixels pertencentes às classes i_{fundo} e i_{luva}, e considera-se que dentro de $\mathcal{R}_{md}$

a $p(i_{fundo}) = p(i_{luva})$. Já $\mathcal{R}_{me}$ está interseccionada com $\mathcal{R}_{md}$ e com $\mathcal{R}_f$, logo dentro de $\mathcal{R}_{me}$ podem haver pixels das classes i_{fundo}, i_{luva} e i_{pele}, e considera-se que $p(i_{fundo}) = p(i_{luva}) = p(i_{pele})$ dentro de $\mathcal{R}_{me}$ e $\mathcal{R}_f$.

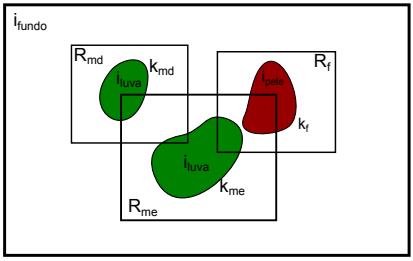


Figura 4.5: Regiões de interesse com intersecção. Quando existe intersecção entre regiões de interesse, dentro da região de interesse $\mathcal{R}_f$, por exemplo, podem existir pixels das classes i_{fundo}, i_{luva} e i_{pele}. Logo para $\mathcal{R}_f$ foi considerado que $p(i_{fundo}) = p(i_{pele}) = p(i_{luva})$.

Sendo C_{t+1} o centro de massa do objeto k no próximo quadro, a região de interesse no próximo quadro $\mathcal{R}_{k,t+1}$ é um retângulo centrado em C_{t+1} e com dimensões dadas por

$$\mathcal{R}_{k,t+1} = 2Rect_{k,t} \tag{4.1}$$

onde $Rect_{k,t}$ é o menor retângulo não rotacionado que engloba o objeto k no quadro atual. No primeiro quadro do vídeo, a $\mathcal{R}_k$ de cada objeto é igual ao tamanho da própria imagem.

C_{t+1} é expresso em termos de suas coordenadas x e y na imagem, representadas por $C_{t+1,x}$ e $C_{t+1,y}$. Essas coordenadas são calculadas, aproximadamente, utilizando os descritores θ_t, D_t e A_t, apresentados na seção 3.5.1, como:

$$\begin{aligned} C_{t+1,x} &= C_{t,x} + (D_t + \frac{A_t}{2})\cos(\theta_t)L_{\text{face}} \\ C_{t+1,y} &= C_{t,y} + (D_t + \frac{A_t}{2})\text{sen}(\theta_t)L_{\text{face}} \end{aligned} \tag{4.2}$$

4.1.3 Classificação dos Pixels

Os pixels são classificados de acordo com a sua cor. Por exemplo, um pixel amarelo tem uma grande probabilidade de pertencer a classe i_{luva}. Sendo $p(i_n|Cr, Cb)$ a probabilidade do pixel pertencer a classe i_n dado que foram observados seus valores de crominância, este deve ser classificado de acordo com a classe a qual tem a maior probabilidade de pertencer, ou seja, o pixel pertence a classe i_n que gerar maior $p(i_n|Cr, Cb)$.

Podemos escrever $p(i_n|Cr, Cb)$ como:

$$\begin{aligned} p(i_n|Cr, Cb) &= \frac{p(i_n, Cr, Cb)}{p(Cr, Cb)} \\ &= \frac{p(Cr, Cb|i_n)p(i_n)}{p(Cr, Cb)} \end{aligned} \tag{4.3}$$

Onde $p(Cr, Cb|i_n)$ é a probabilidade de se observar as crominâncias Cr, Cb dada a classe i_n, $p(i_n)$ é a probabilidade de ocorrência de pixels da classe i_n, e $p(Cr, Cb)$ é apenas um fator de normalização, e pode ser ignorado, pois aparece em $p(i_n|Cr, Cb)\ \forall\ n$. Então chega-se a:

$$p(i_n|Cr, Cb) \propto p(Cr, Cb|i_n)p(i_n) \tag{4.4}$$

Como foi explicado na seção 4.1.2, dentro de uma região de interesse $\mathcal{R}_k$, $p(i_n)$ foi considerado igual para todas classes dos objetos envolvidos naquela região, então:

$$p(i_n|Cr, Cb)_{\mathcal{R}_k} \sim \begin{cases} p(Cr, Cb|i_n) \text{ para as classes } i_n \text{ dos objetos contidos em } \mathcal{R}_k \\ 0 \text{ para outras classes} \end{cases} \tag{4.5}$$

Onde $p(i_n|Cr, Cb)_{\mathcal{R}_k}$ é a probabilidade de um pixel pertencer à classe i_n dado que foram observados seus valores de crominância, e a classificação está sendo feita em $\mathcal{R}_k$.

Finalmente, cada pixel é classificado como pertencente à classe i_n que possui a maior probabilidade de tê-lo gerado na região $\mathcal{R}_k$, ou seja:

$$\mathcal{P}_{x,y} = \arg\max_{i_n}[p(i_n|Cr, Cb)_{\mathcal{R}_k}] \tag{4.6}$$

Onde $\mathcal{P}_{x,y}$ é a classe i_n do pixel com coordenadas x, y.

4.1.4 Modelo da Cor da Pele e Luvas

As Figuras 4.6 e 4.7 mostram os histogramas da cor da luva e da pele (clara) no espaço $CrCb$, onde esses foram obtidos a partir de várias imagens em diferentes condições de iluminação. As funções densidade de probabilidade (pdfs) de $p(Cr, Cb|i_n)$ foram modeladas por uma mistura de pdfs gaussianas e estimadas utilizando um conjunto de imagens de treinamento. As Figuras 4.8 e 4.9 mostram as função densidade de probabilidade (pdf)s $p(Cr, Cb|i_{pele})$ e $p(Cr, Cb|i_{luva})$ estimadas para uma mistura de 5 gaussianas. A $p(Cr, Cb|i_{fundo})$ foi estimada do mesmo modo.

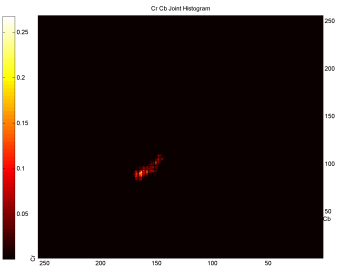


Figura 4.6: Histograma da cor da pele.

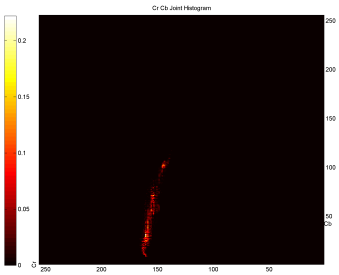


Figura 4.7: Histograma da cor da luva.

4.1.5 Tabelas de Probabilidades e Tabelas de Classes

Calcular $\mathcal{P}_{x,y}$ para cada pixel em uma imagem é um desperdício de processamento, e por esse motivo, algumas otimizações foram feitas para uma implementação mais eficiente.

Como os valores da crominâncias são discretos, as pdfs $p(Cr, Cr|i_n)$ são avaliadas para todas as combinações Cr, Cb, e o resultado é armazenado em uma tabela chamada de **tabela de probabilidades**. É gerada uma tabela de probabilidades para cada classe i_n, como ilustrado na Figura 4.10. Essas tabelas são geradas apenas uma vez e então armazenadas para uso posterior.

Ao iniciar o sistema de detecção, as tabelas de probabilidades são carregadas e utilizadas para gerar as **tabelas de classes**, que armaze-

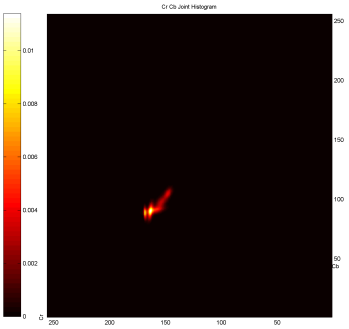


Figura 4.8: PDF da cor da pele.

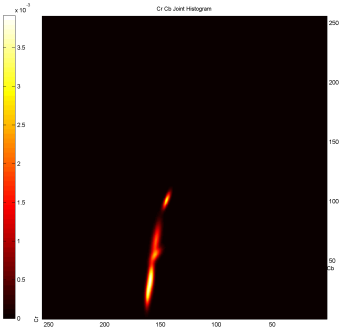


Figura 4.9: PDF da cor da luva.

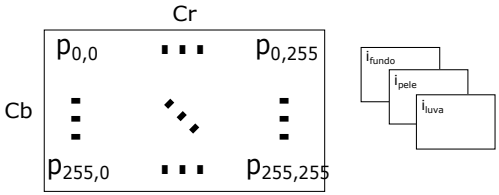


Figura 4.10: Tabela de probabilidades. À esquerda a tabela de probabilidade para determinada classe i_n, onde $p_{Cb,Cr} = p(Cr, Cb|i_n)$. À direita as três tabelas de probabilidades deste trabalho: i_{fundo}, i_{pele} e i_{luva}.

nam a classe i_n de cada combinação Cr, Cb.

Cada combinação Cr, Cb é classificada em uma das classes i_n de acordo com a equação 4.6, porém aqui classificam-se as combinações Cr, Cb no lugar dos pixels. Além disso, no lugar de se calcular $p(i_n|Cr, Cb)_{\mathcal{R}_k}$ para fazer a classificação, utiliza-se as tabelas de probabilidades correspondentes as classes i_n dos objetos contidos em $\mathcal{R}_k$. Dessa forma, é necessário uma tabela de classes para cada combinação de classes possível em $\mathcal{R}_k$. A Figura 4.11 mostra a estrutura de uma tabela de classes (esquerda) e as 3 tabelas de classes (direita) possíveis no sistema atual.

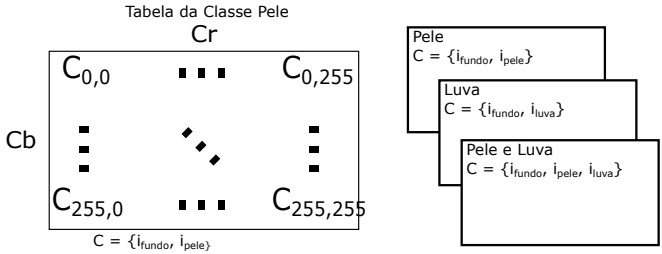


Figura 4.11: Tabela de classes. À esquerda a tabela de classes da classe i_{pele}, onde cada valor $C_{Cr,Cb}$ armazena a classe i_n que a crominância Cr, Cb foi classificada, que nesse caso são as classes i_{fundo} ou i_{pele}. À direita as diferentes tabelas de classes: a primeira correspondente à classe i_{pele} (utilizada na face), onde as cores podem ser classificadas como pertencentes à classe i_{fundo} ou i_{pele}; na segunda entre i_{fundo} e i_{luva} (utilizada nas mãos); a terceira é para a combinação das classes i_{pele} e i_{luva} (quando a mão está sobre a face).

4.1.6 Mapas de Objetos

Um mapa de objetos é uma imagem binária de uma região de interesse, onde os pixels com classe correspondente ao objeto k de $\mathcal{R}_k$ possuem valor 1 e os pixels restantes possuem valor 0. Por exemplo, na região de interesse da mão esquerda $\mathcal{R}_{me}$ todos os pixels da classe i_{luva} são classificados com valor 1 no mapa de objetos.

Na figura 4.12 podemos ver a representação de uma imagem com três objetos, k_f, k_{md} e k_{me}. Cada região de interesse é passada para um bloco responsável por classificar seus pixels como pertencentes ou não ao objeto de interesse daquela região, de acordo com a tabela de classes necessária. A saída do classificador é uma imagem binária, denominada de mapa de objetos. Finalmente os mapas são filtrados antes de serem disponibilizados para o sistema. A seguir são explicados os blocos de classificação e filtragem.

Classificação

A Figura 4.13 ilustra o processo de classificação de uma região $\mathcal{R}_k$, onde há o objeto k com pixels da classe i_{luva}, um pedaço do objeto k_f com pixels da classe i_{pele}, e ainda alguns outros objetos de ruído com pixels da classe i_{luva}. O classificador utiliza a tabela de classes correspondente a interseção de $\mathcal{R}_k$ como $\mathcal{R}_f$ para classificar os pixels

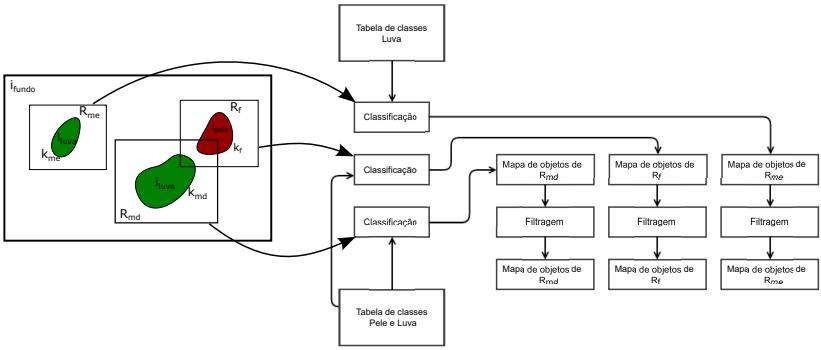


Figura 4.12: Geração dos mapas de objetos.

da região de interesse. Basicamente é verificado na tabela de classes a qual classe o pixel pertence, de acordo com sua crominância Cr, Cb. Se a classe do pixel for a mesma classe do objeto da região de interesse, então o pixel recebe valor 1, caso contrário recebe valor 0. No final obtém-se uma imagem binária com vários objetos, onde esses objetos são posteriormente detectados, utilizando o algoritmo de rastreamento de contorno apresentado em [49], o qual retorna o contorno externo de todos os objetos. Dentre os objetos detectados é necessário identificar qual deles corresponde ao objeto de interesse.

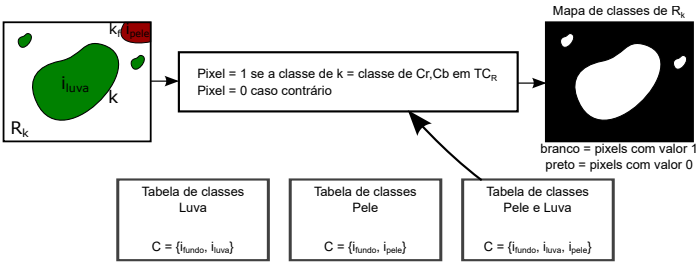


Figura 4.13: Bloco de classificação da Figura 4.12. TC_R é a tabela de classes selecionada, que neste caso é a tabela das classes Pele e Luva.

Filtragem

Para diminuir o ruído causado pelo erro de classificação dos pixels, os mapas de objetos são filtrados da seguinte forma:

(*i*) Para cada pixel do mapa de objetos, seus vizinhos-de-8 são verificados.

(*ii*) O pixel analisado é classificado como pertencendo a classe que aparece mais vezes entre os nove pixels verificados na etapa anterior (o pixel analisado e seus oito vizinhos).

As Figuras 4.14 e 4.15 apresentam uma imagem e ao lado seus mapas de objetos antes e após a filtragem. É gerado um mapa de objetos para a face, um para a mão direita e um para a mão esquerda, onde a cor branca representa os pixels atribuídos ao mapa.

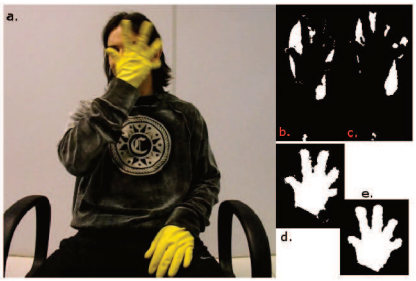


Figura 4.14: Mapa de objetos filtrado para mãos separadas. a. Imagem analisada. b. Mapa de objetos da face. c. Mapa de objetos filtrado. d. Mapa de objetos da mão direita. e. Mapa de objetos filtrado.

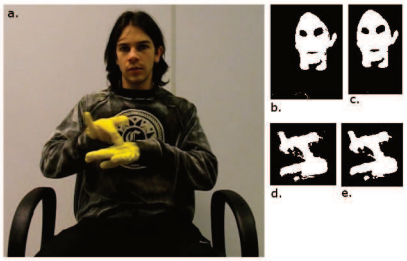


Figura 4.15: Mapa de objetos filtrado para mãos sobrepostas. a. Imagem analisada. b. Mapa de objetos da face. c. Mapa de objetos filtrado. d. Mapa de objetos das mãos sobrepostas. e. Mapa de objetos filtrado.

4.1.7 O Sistema de Detecção

O diagrama em blocos da Figura 4.16 esquematiza o funcionamento do sistema de detecção das mãos e da face. Primeiramente, utilizam-se as tabelas de probabilidades para gerar as tabelas de classes. Em seguida o sistema é inicializado no primeiro quadro do vídeo, gerando as regiões de interesse $\mathcal{R}_{f,t+1}$, $\mathcal{R}_{md,t+1}$ e $\mathcal{R}_{me,t+1}$. Então seleciona-se o próximo quadro e geram-se os mapas de objetos das regiões de interesse calculadas no quadro anterior, que são utilizados para fazer a detecção das mãos e da face. De posse dos objetos de interesse, os descritores são extraídos. Em seguida geram-se as novas regiões de interesse $\mathcal{R}_{f,t+1}$, $\mathcal{R}_{md,t+1}$ e $\mathcal{R}_{me,t+1}$ e seleciona-se o próximo quadro, repetindo-se o processo até o final do vídeo. A Figura 4.17 mostra as regiões de interesse e os objetos de interesse detectados em um quadro. A seguir são explicados em mais detalhes os blocos de Inicialização, Detecção das mãos e Detecção da face.

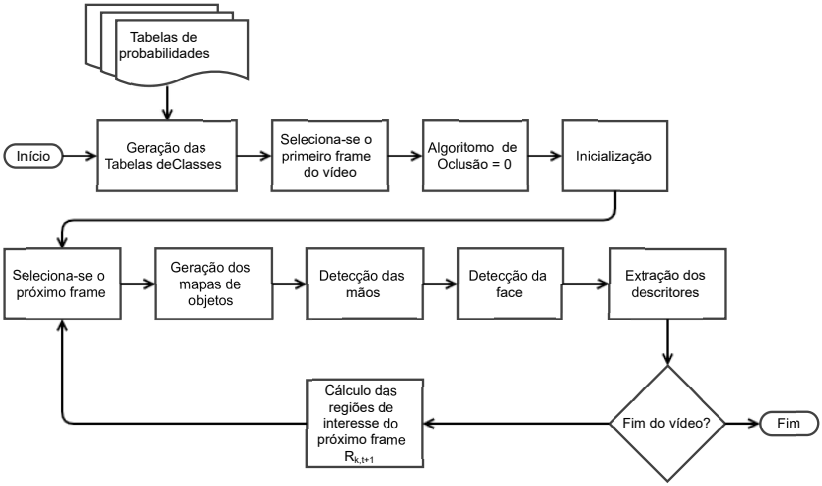


Figura 4.16: Diagrama em blocos do sistema de detecção da face e das mãos. $\mathcal{R}_{k,t+1}$ representa a região de interesse em $t+1$, onde $k = \{f, md, me, ms\}$.

Inicialização

Durante a inicialização, assume-se que a face não sofre oclusão e que as mãos estão separadas, não estão cruzadas e não sofrem oclusão. O

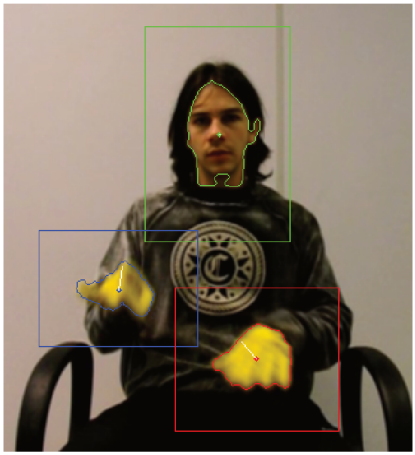


Figura 4.17: Regiões de interesse e objetos de interesse detectados em um quadro. Os contornos em verde, azul e vermelho representam os objetos {f}, {md} e {me}, respectivamente. Os retângulos são as regiões de interesse $\mathcal{R}_f$, $\mathcal{R}_{md}$ e $\mathcal{R}_{me}$. Os pontos em verde, azul e vermelho são o centro de massa C_t dos respectivos objetos. A linha branca indica o C_{t+1} calculado para cada objeto, baseando-se na sua posição, velocidade e aceleração atual.

diagrama em blocos da Figura 4.18 ilustra o processo de inicialização. Inicialmente, as regiões de interesse $\mathcal{R}_f$, $\mathcal{R}_{me}$ e $\mathcal{R}_{md}$ são iguais ao tamanho da imagem. Então são gerados os mapas de objetos para cada região, e a face é selecionada como o maior objeto no mapa de $\mathcal{R}_f$. Como os pixels das mãos possuem a mesma classe, e as regiões de interesse $\mathcal{R}_{me}$ e $\mathcal{R}_{md}$ são iguais, selecionam-se os dois maiores objetos no mapa de $\mathcal{R}_{me}$ ou $\mathcal{R}_{md}$ e classifica-se o mais à esquerda como sendo a mão esquerda e o mais à direita como sendo a mão direita. Em seguida calculam-se as regiões de interesse $\mathcal{R}_{f,t+1}$, $\mathcal{R}_{me,t+1}$ e $\mathcal{R}_{md,t+1}$.

Detecção das Mãos

A detecção das mãos consiste em selecionar qual dos objetos detectados no mapa de objetos da região de interesse de cada mão, $\mathcal{R}_{md}$ e $\mathcal{R}_{me}$, representa a respectiva mão. Como já foi mencionado, os objetos em uma região de interesse são detectados através do algoritmo de rastreamento de contornos proposto em [49]. Quando as mãos estão

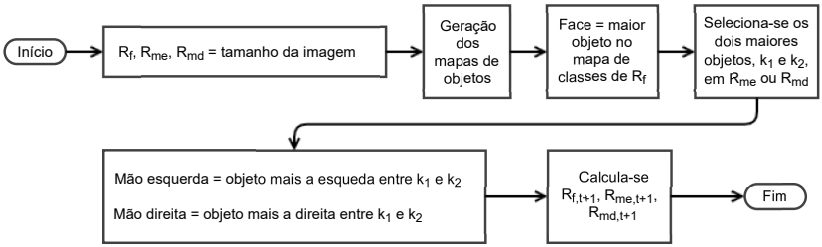


Figura 4.18: Inicialização do sistema de deteção.

separadas, temos uma região de interesse para cada mão, $\mathcal{R}_{md}$ e $\mathcal{R}_{me}$. Porém quando as mãos se tocam ou se sobrepõem detecta-se apenas um objeto, pois as mãos possuem a mesma cor, e tem-se então apenas uma região de interesse $\mathcal{R}_{ms}$. Devido ao fato das mãos se sobreporem, ocluindo uma à outra, chamou-se isso de oclusão das mãos. Porém, como quando as mãos se tocam o efeito obtido é o mesmo (geração de um único objeto), refere-se, neste trabalho, tanto as mãos sobrepostas como as mãos se tocando como oclusão.

O sistema de detecção das mãos, representado na Figura 4.19 é dividido em duas partes principais: a detecção das mãos quando não ocorre oclusão e a detecção das mãos sobrepostas quando ocorre oclusão. Inicialmente é verificado se o algoritmo de oclusão está ativado (1) ou desativado (0). Se estiver desativado o sistema não detectou oclusão e procede-se normalmente com a detecção das mãos, porém se estiver ativado significa que no quadro anterior as mãos estavam se ocluindo e o sistema muda para o modo de detecção das mãos sobrepostas. A Figura 4.20 mostra alguns quadros onde as mãos estão separadas, então se ocluem e voltam a se separar. A seguir são explicados estes dois modos de funcionamento do sistema.

Algoritmo de Oclusão Desativado

Quando o algoritmo de oclusão está desativado significa que no quadro anterior não havia oclusão entre as mãos. Portanto, é o bloco detecção das mãos (Figura 4.19) que detecta o objeto correspondente à mão esquerda e à mão direita nos seus respectivos mapas de objetos. Os dois objetos são armazenados em uma variável temporária TempMãos para utilização posterior. Em seguida é verificado se ocorre intersecção

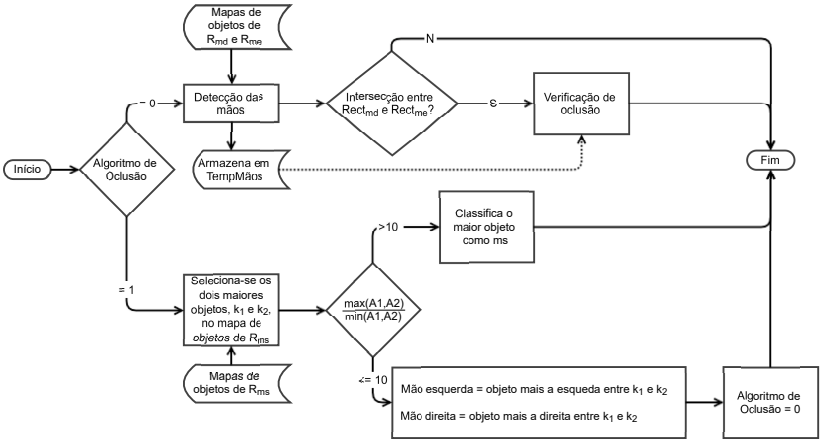


Figura 4.19: Sistema de detecção das mãos.

Figura 4.20: Oclusão das mãos. Da esquerda para a direita: as mãos se aproximam; as mãos se sobrepõem, detecta-se o objeto k_{ms} e ativa-se o Algoritmo de Oclusão; as mãos continuam sobrepostas; as mãos começam a se separar, mas continuam sobrepostas; as mãos se separam, detectam-se os objetos k_{md} e k_{me}, e desativa-se o Algoritmo de Oclusão.

entre o menor retângulo de contorno de cada mão. Como pode ser visto na Figura 4.21, se não houver intersecção as mãos estão separadas. Porém se houver intersecção pode ou não haver oclusão, e o bloco Verificação de oclusão é responsável por fazer essa verificação. A seguir são explicados os blocos Detecção das mãos e Verificação de oclusão.

Detecção das mãos Todos os objetos contidos no mapa de classe da região $\mathcal{R}_{mx}$, onde x pode ser d ou e, são detectados. Se existir apenas um este é classificado como k_{mx}. Porém se existirem mais de um objeto, selecionam-se os dois maiores e verifica-se se a área de um dos objetos é maior que um limiar λ e a do outro é menor. Caso a resposta seja sim, o menor objeto é descartado, pois é considerado ruído, e o outro

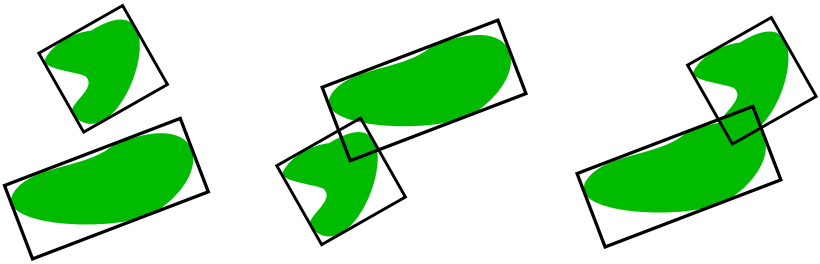


Figura 4.21: Intersecção entre os retângulos de menor contorno. À esquerda os objetos estão separados. No centro, há intersecção entre os retângulos de menor contorno, porém os objetos não se sobrepõem. À direita ocorre oclusão.

é classificado como k_{mx}. Caso a área de ambos os objetos seja maior que λ, então o objeto que possuir seu centro de massa C_t mais próximo do C_{t+1} calculado no quadro anterior é classificado como k_{mx}. Esse processo é ilustrado no diagrama em blocos da Figura 4.22.

Verificou-se empiricamente que um objeto com área menor que $\lambda = \frac{1}{5}$ da área do objeto k_{mx} no quadro anterior pode ser considerado ruído, pois é muito improvável que o mesmo sofra uma variação tão grande de tamanho entre dois quadros consecutivos.

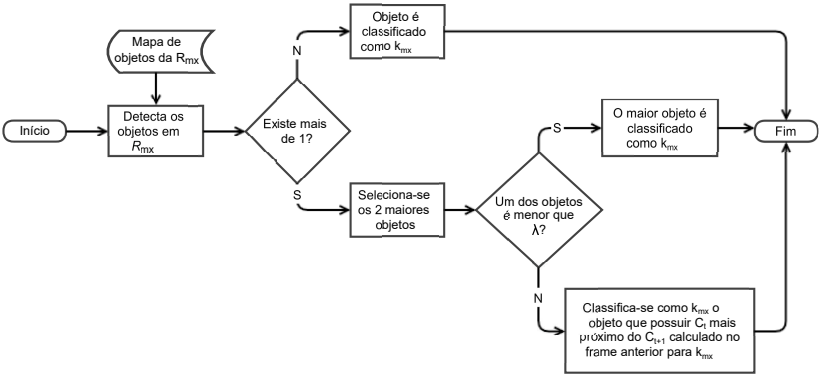


Figura 4.22: Diagrama do bloco Detecção das mãos do diagrama da Figura 4.19.

Verificação de oclusão Este bloco é responsável por verificar se as mãos estão se ocluindo ou não. Para isso, é criada uma nova região de interesse, $\mathcal{R}_{temp}$, que engloba os dois objetos k_{md} e k_{me} detectados anteriormente, e gera-se um mapa de objetos para essa região. Detectam-se os objetos no mapa de objetos, e se houver apenas um significa que as mãos estão se ocluindo, e este é classificado como k_{ms}. Se houver mais de um, selecionam-se os dois maiores e verifica-se se ambos possuem área maior que $\lambda_{\text{mãos}}$. Se as mãos estiverem se ocluindo, então um objeto terá uma área grande e o outro uma área pequena, sendo considerado ruído. O objeto menor que $\lambda_{\text{mãos}}$ é considerado ruído, e o outro é classificado como k_{ms}. Em ambos os casos de oclusão, o Algoritmo de Oclusão é ativado. Caso os dois objetos possuam área maior que $\lambda_{\text{mãos}}$, então as mãos estão separadas, e os objetos detectados anteriormente e armazenados em TempMãos são utilizados como k_{md} e k_{me}. Todo esse processo é ilustrado no diagrama em blocos da Figura 4.23.

Detectou-se empiricamente que se algum contorno possuir uma área menor que $\lambda_{\text{mãos}} = \frac{1}{5}$ da área de qualquer uma das mãos no quadro anterior, o mesmo pode ser considerado ruído, para esta aplicação.

Algoritmo de Oclusão Ativado

Quando o Algoritmo de Oclusão está ativado, significa que no quadro anterior as mãos estavam sobrepostas, e é necessário verificar se elas continuam sobrepostas ou se elas se separaram. Pode-se ver na Figura 4.19 que, primeiramente são detectados os dois maiores objetos, k_1 e k_2, no mapa de objetos de $\mathcal{R}_{ms}$. Em seguida deve-se verificar se esses dois objetos representam as mãos sobrepostas e ruído, ou a mão direita e a mão esquerda separadas. Para isso parte-se do pressuposto que, sendo A_1 e A_2 a área dos objetos k_1 e k_2, se $A_1 \approx A_2$ cada objeto representa uma mão. Porém, se uma das áreas for muito maior que a outra, interpreta-se o menor objeto como ruído e o maior como a sobreposição das mãos. Verificou-se, empiricamente, que se um objeto for cerca de dez vezes maior que o outro, o menor pode ser considerado ruído. Portando, se

$$\frac{\max(A_1, A_2)}{\min(A_1, A_2)} \leq 10 \tag{4.7}$$

os objetos estão separados, e classifica-se o mais a esquerda como k_{me} e o mais a direita como k_{md}, e desativa-se o Algoritmo de Oclusão.

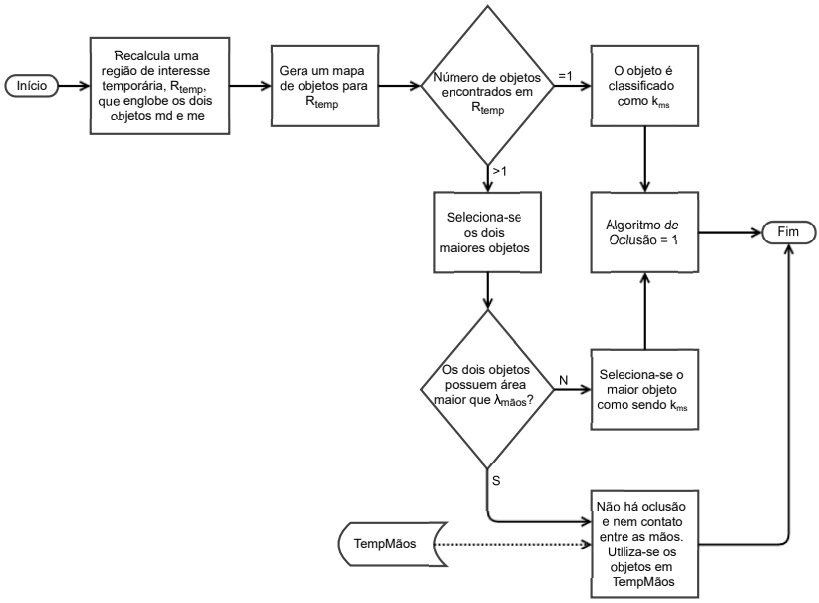


Figura 4.23: Diagrama do bloco Verificação de oclusão do diagrama da Figura 4.19.

Caso a razão entre o máximo e o mínimo das áreas seja maior que 10, o maior objeto é classificado como k_{ms}. Obviamente, se for detectado apenas um objeto no mapa de objetos de $\mathcal{R}_{ms}$, então este é classificado como k_{ms}. É importante notar que o algoritmo não é capaz de tratar os casos em que as mãos se cruzam ocluindo uma à outra, pois após a oclusão as mãos estariam cruzadas e a mão direita estaria do lado esquerdo e por isso seria classificada como mão esquerda, e vice-versa.

Detecção da Face

O último bloco do diagrama da Figura 4.16 a ser tratado neste capítulo é responsável pela detecção da face. A Figura 4.24 ilustra os passos para a detecção da face. Primeiramente detecta-se o maior objeto no mapa de objetos da $\mathcal{R}_f$ e verifica-se se o $Rect_{md}$ ou $Rect_{me}$ atuais intersectam-se com o $Rect_f$ do quadro anterior centrado no C_{t+1} calculado no quadro anterior. Caso não haja intersecção, então as mãos não estão ocluindo a face e o objeto detectado é classificado como k_f.

Caso contrário, a face pode estar sendo ocluída.

A oclusão da face ocorre quando uma mão ou as duas estão entre a face e a câmera. Esse fato faz com que na região de interesse da face, $\mathcal{R}_f$, sejam detectados vários contornos de tamanho menor que a face e com centro de massa diferente do centro de massa da face, como é mostrado na imagem central da Figura 4.25. Como os parâmetros da face utilizados pelo sistema são o seu C_t e o seu *Rect*, somente esses dois parâmetros são calculados durante a oclusão da face. Se a área do objeto detectado for maior que um limiar λ_{face}, então o centro de massa C_t da face é calculado. Caso o objeto seja menor que λ_{face}, significa que grande parte da face está sendo ocluída, e por isso o C_t da face é mantido constante. Em ambos os casos o $Rect_f$ é mantido constante, pois assume-se que em um curto intervalo de tempo (período da oclusão) a forma e a posição da face tenham uma baixa variação, o que em geral é válido para gestos da LIBRAS.

O parâmetro λ_{face} foi escolhido como sendo igual a metade da área da face no quadro anterior. Isso implica que o algoritmo só irá calcular um novo C_t para a face se a área do objeto detectado for maior que metade da área da face no momento anterior à oclusão.

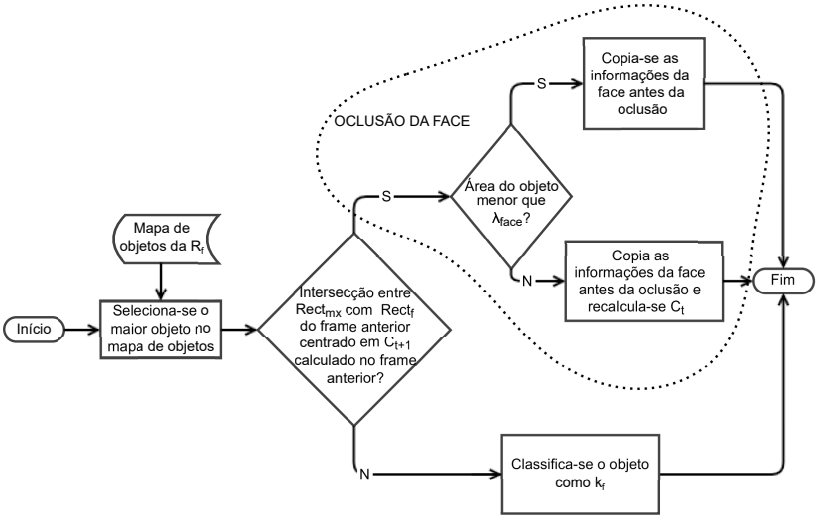


Figura 4.24: Sistema para detecção da face.

Figura 4.25: Oclusão da face. O ponto verde representa o C_t da face, o contorno verde representa o objeto detectado como face {f}, e o retângulo verde representa o $Rect_f$. À esquerda: face não ocluída; no centro: mão oclui a face. Dentre os vários objetos detectados no mapa de classes de $\mathcal{R}_f$, o canto superior direito é selecionado, porém o $Rect_f$ e o C_t se mantém proporcional a face; á direita: mão se afastando da face.

4.2 Módulo de Extração dos Descritores

Alguns dos descritores (como a área, posição, direção do movimento, velocidade e aceleração) são calculados pelo módulo de rastreamento, pois são necessários para o funcionamento do próprio módulo. O restante dos descritores são calculados pelo módulo de extração dos descritores. Esse módulo utiliza as imagens segmentadas pelo módulo anterior para calcular todos os descritores descritos na seção 3.5.

4.3 Módulo Classificador

Este módulo foi implementado utilizando o HTK, que é um conjunto de ferramentas desenvolvidas para o reconhecimento da fala através de HMMs. O apêndice A.3 introduz o HTK e explica como construir o seu arquivo de descritores externamente, uma vez que este é gerado automaticamente pelas ferramentas do HTK responsáveis por extrair os descritores dos áudios, mas aqui devemos criá-lo utilizando os descritores extraídos dos vídeos.

O classificador utilizado neste trabalho é composto por uma rede de HMMs e o algoritmo Token Passing é utilizado para fazer a classificação, onde cada HMM possui um rótulo correspondente a um gesto

e a sequência de HMMs mais provável na rede é transcrita para fazer o reconhecimento dos gestos. Essa rede de HMMs engloba todas as combinações possíveis que o sistema é capaz de reconhecer. Como cada vídeo possui quatro gestos que começam e terminam na posição de repouso, foi criada uma rede de comunicação especializada para o reconhecimento da estrutura "repouso, gesto1, gesto2, gesto3, gesto4, repouso". As figuras 4.26 e 4.27 mostram a rede de HMMs utilizada. Nelas pode-se ver que uma rede de gestos é composta pela conexão em paralelo dos HMMs correspondentes aos gestos, e a rede de HMMs é composta pela conexão em série das redes de gestos com HMMs correspondentes ao repouso no início e final da rede.

Figura 4.26: Rede de HMMs.

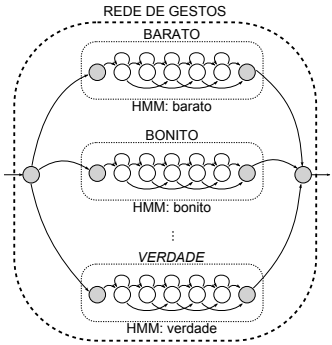


Figura 4.27: Rede de gestos. Acima dos HMMs temos a sua transcrição e abaixo o nome dado para o modelo, utilizado pelo HTK.

Cada gesto foi modelado por um HMM com estrutura esquerda direita sem transições de escape, como ilustrado na Figura 4.28. Esta estrutura foi escolhida porque os gestos se desenvolve progressivamente no tempo, sempre avançando e nunca retrocedendo para um estado anterior. A probabilidade de emissão de símbolos de cada estado foi modelada por uma soma de gaussianas de n dimensões, sendo n igual a quantidade de descritores utilizados.

Alguns gestos que possuem movimentos repetitivos (por exemplo, a mão sobe, desce, sobe, desce, ...) necessitariam de vários HMMs com número de estados diversos, ou com transições de escape, para modelar o gesto para o número de vezes que o movimento é repetido, e ainda assim o gesto ficaria limitado a um número máximo de repetições do movimento. Uma solução pode ser pensar como se esses gestos estivessem retrocedendo para um estado anterior, assim quando o movimento é repetido ele não avança para o próximo estado, mas volta para o estado do início do movimento. Esse tipo de gesto foi dividido em partes denominadas sub-gestos, que correspondem aos segmentos simples e sem repetição que compõem o gesto. Os HMMs dos sub-gestos são treinados normalmente e depois são conectados e uma transição de retorno é inserida, formando o HMM do gesto. Como exemplo, o gesto para o sinal "carro"da LIBRAS é composto por movimentos das mãos subindo e descendo alternadamente, dessa forma os sub-gestos são a mão descendo e a mão subindo. Sendo assim, os gestos com movimentos repetitivos são modelados por HMMs esquerda-direita com transição de retorno, como ilustrado na Figura 4.29.

Gestos e sub-gestos possuem uma diferença considerável em suas durações: os sub-gestos são muito rápidos(menos de dez quadros com uma aquisição de 30 quadros por segundo (fps)), enquanto que os gestos possuem uma duração maior. Como os sub-gestos duram apenas alguns quadros, poucos símbolos são extraídos de cada amostra do sub-gesto e para evitar que os HMMs que os modelam possuam mais estados do que símbolos extraídos, eles são modelados por HMMs com um número de estados reduzido. Os gestos, por possuírem uma duração maior, tendem a possuir uma variação de parâmetros maior, e por isso são modelados por HMMs com um número maior de estados. Além dos HMMs dos gestos e sub-gestos foi utilizado um HMM de um único estado emissor para modelar as mãos na posição de repouso (abaixo da linha da cintura), pois as mão tendem a ficar estacionárias.

O número de estados para os HMMs curtos e longos é determinado empiricamente na seção Testes e Resultados.

4.3.1 Treinamento dos HMMs

O treinamento dos HMMs é feito em cinco etapas: construção da estrutura dos HMMs dos gestos e sub-gestos, estimativa inicial

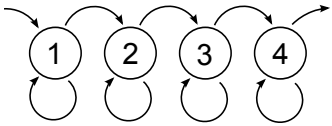


Figura 4.28: HMM com estrutura esquerda-direita sem transições de escape.

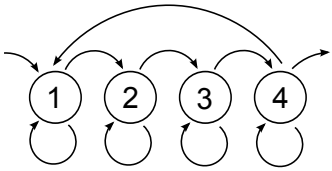


Figura 4.29: HMM com estrutura esquerda-direita com transição de retorno.

dos parâmetros dos HMMs, re-estimação dos parâmetros dos HMMs, criação dos HMMs dos gestos com movimentos repetitivos através do agrupamento dos HMMs dos sub-gestos, e a re-estimação dos parâmetros dos HMMs dos mesmos.

Para realizar o treinamento foram utilizados vários vídeos onde os gestos são realizados continuamente (em sequência e sem pausa), e para cada vídeo foi feita uma transcrição indicando os sub-gestos e os gestos, bem como a sua posição temporal no vídeo. Os descritores extraídos dos vídeos, juntamente com as transcrições são utilizados para treinar os HMMs.

Estrutura dos HMMs

Como os HMMs possuem estrutura esquerda-direita sem transições de escape, os estados apenas possuem transições para ele mesmo e para o próximo. Para essas transições, foram utilizados valores iniciais de 60% de chance de ficar no mesmo estado e 40% de ir para o próximo. Esses valores foram escolhidos por serem próximos, e foi escolhido um valor de transição para o mesmo estado maior que o de transição para o próximo devido ao fato de que a quantidade de símbolos extraídos dos gestos ser maior que o número de estados de seus HMMs. Os valores de transição são re-estimados no processo de inicialização. A estrutura criada é apresentada na Figura 4.30.

Os valores do vetor de média e da matriz de covariância das gaussianas que modelam os estados não são importantes nesta etapa, pois serão inicializados posteriormente, por isso o vetor de média $\boldsymbol{\mu} = 1$ e a matriz de covariância $\mathbf{V} = \mathbf{I}$, onde $\mathbf{I}$ é uma matriz identidade.

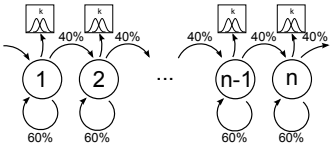


Figura 4.30: Estrutura criada para os HMMs dos sub-gestos.

Inicialização e Re-estimação dos HMMs

Como os vídeos contém vários gestos, para cada vídeo foi feita uma transcrição indicando cada gesto (ou sub-gesto) e seu tempo de origem e término. Desse modo é possível determinar qual parte de cada vídeo deve ser utilizada no treinamento dos HMMs dos gestos e sub-gestos. Para todos os HMMs é feita uma estimativa inicial de seus parâmetros e então é feita a re-estimação de Baum-Welch, como explicado no apêndice A.1.3.

Após a re-estimação dos parâmetros, os HMMs dos sub-gestos são conectados e as transições de retorno são adicionadas para formar os HMMs dos gestos com movimentos repetitivos, como mostra a figura 4.31. Para as transições de retorno foi dado um valor inicial de 40%. Com a estrutura dos HMMs desses gestos pronta e os estados dos seus sub-gestos corretamente estimados, faz-se a re-estimação dos novos HMMs.

4.4 Determinação do Número de Estados e Descritores dos HMMs

Nesta seção são realizados testes para determinar o número de estados e o número de gaussianas que modelam cada estado dos HMMs, bem como quais os descritores que serão utilizados. Utilizou-se dois conjuntos de vídeos nesta etapa: o conjunto de treinamento 1, composto por 60 vídeos; e o conjunto de validação, composto por 31 vídeos. O

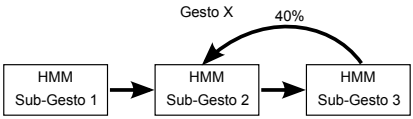


Figura 4.31: Construção do HMM de um gesto. O HMM de um determinado gesto é construído a partir da conexão dos HMMs de seus sub-gestos. Para os gestos com movimentos repetitivos é inserida uma transição de retorno com probabilidade de 40%. O novo HMM é re-estimado através do algoritmo de Baum-Welch.

conjunto de treinamento 1 foi utilizado para o treinamento dos HMMs e o conjunto de validação para os testes. Informações sobre os vídeos são dados na seção 6.1.

Como já mencionado, uma amostra de um gesto é a execução de um gesto, e observação é o conjunto de descritores extraído de cada quadro do vídeo. Como cada estado de um HMM com estrutura esquerda-direita deve emitir ao menos uma observação, o número de observações obtidas em determinada amostra de um gesto deve ser maior que o número de estados do HMM que modela o gesto. Foi verificado experimentalmente que, com uma taxa de aquisição de 30 fps, os HMMs que modelam os sub-gestos emitem cerca de 10 observações enquanto que os HMMs dos gestos emitem um número de observações um pouco maior. Logo, nos testes para verificar o número adequado de estados emissores dos HMMs, variou-se de 3 a 6 estados nos HMMs dos sub-gestos e de 6 a 13 nos HMMs dos gestos.

O número de gaussianas necessárias para modelar cada estado depende do número de estados, dos descritores utilizados e do número de amostras disponíveis para o treinamento dos HMMs. Devido a limitação de amostras de treinamento variou-se o número de gaussianas entre uma e duas, pois para um número maior de gaussianas verificou-se que a quantidade de amostras era insuficiente para realizar o treinamento.

Dentre os descritores apresentados, somente alguns serão utilizados para o reconhecimento, pois quanto maior o número de descritores, maior será o tempo de processamento e mais amostras são necessárias para treinar adequadamente os HMMs, caso contrário o excesso de descritores pode piorar o desempenho do reconhecedor. Logo, faz-se

necessário determinar quais os conjuntos de descritores que geram as melhores taxas de reconhecimento.

O teste para determinar o número de estados dos HMMs, o número de gaussianas de cada estado e os descritores utilizados foi realizado da seguinte forma:

- Determinou-se alguns conjuntos de descritores para realização dos testes, como descrito em 4.4.1;
- Para cada conjunto de descritores variou-se o número de gaussianas dos estados de 1 a 2;
- Para cada combinação acima variou-se o número de estados dos HMMs dos sub-gestos de 3 a 6;
- Para cada combinação acima variou-se o número de estados dos HMMs gestos de 6 a 13.
- Para cada estrutura gerada pela combinação dos parâmetros acima foi realizado o treinamento dos HMMs e feito o reconhecimento dos vídeos de validação.
- Foi adotada a estrutura que gerou a melhor taxa de reconhecimento de gestos.

4.4.1 Determinação dos Conjuntos de Descritores

Cada realização do teste para um conjunto de parâmetros leva várias horas devido ao treinamento dos HMMs, logo realizar os testes para muitas das possíveis combinações de descritores torna-se inviável. Então foi necessário determinar alguns conjuntos de descritores para serem testados.

Os descritores de movimento e alguns descritores da configuração da mão geram apenas um escalar, enquanto outros descritores da configuração da mão geram um vetor de valores, fazendo com que seja necessário um número maior de amostras de treinamento para estimar os parâmetros dos HMMs. Devido a esse motivo, os primeiros utilizaram a informação das duas mãos, enquanto os últimos utilizaram somente a informação da mão principal. O conjunto de descritores que utilizam a informação das duas mãos é chamado de **conjunto simples**,

enquanto o restante é chamado de **conjunto complexo**. A tabela 4.1 mostra os descritores utilizados.

Os conjuntos de descritores foram escolhidos em três etapas:

(*i*) Primeiramente foi gerado um conjunto de descritores composto por todos os descritores implementados (descritores do conjunto simples e descritores do conjunto complexo). Posteriormente foram gerados novos conjuntos compostos por somente um dos descritores complexos e pelos descritores do conjunto simples (foi gerado um conjunto para cada descritor do conjunto complexo). Também foi gerado um conjunto composto somente pelos descritores do conjunto simples ($r_{\text{mão,face}}, \alpha_{\text{mão,face}}, \theta_t, D_t, A_t, Alg_t, Ret_t, Cpt_t, \phi_t$).

(*ii*) Novos conjuntos de descritores foram gerados a partir do conjunto simples da seguinte forma: excluiu-se alternadamente um descritor do conjunto simples, gerando um novo conjunto para cada exclusão.

(*iii*) Depois de realizado o teste com os conjuntos de descritores da etapa anterior, detectou-se qual o conjunto que gerou a melhor taxa reconhecimento correto de gestos, e gerou-se novos conjuntos a partir do mesmo. Do conjunto que obteve a melhor taxa de reconhecimento, novamente, excluiu-se alternadamente um dos descritores, gerando um novo conjunto para cada descritor excluído.

4.4.2 Resultados dos Testes

As tabelas 4.2, 4.3 e 4.4 mostram os resultados do teste para os conjuntos de descritores gerados nas três etapas descritas anteriormente. Nota-se que na maioria dos casos o melhor resultado foi obtido usando-se apenas uma gaussiana para modelar cada estado, pois a quantidade de dados de treinamento é insuficiente para treinar adequadamente os HMMs com a probabilidade de emissão de símbolos dos estados modelada por mais de uma gaussiana.

Dentre os conjuntos de descritores utilizados na primeira etapa pode-se ver na tabela 4.2 que o conjunto que gerou a melhor taxa de

reconhecimento foi o conjunto composto apenas pelos descritores simples. Nos conjuntos utilizados na segunda etapa, compostos apenas pelos descritores do conjunto simples, verifica-se que houve um acréscimo na taxa de reconhecimento com a exclusão do descritor Ret_t. Os resultados mostrados na tabela 4.4, gerados a partir do conjunto que gerou a melhor taxa de reconhecimento na etapa anterior, são semelhantes aos obtidos na etapa anterior.

A estrutura dos HMMs que gerou a melhor taxa de reconhecimento foi a estrutura com 4 estados emissores para os HMMs dos sub-gestos e 11 para os HMMs dos gestos, com estados modelados por apenas uma gaussiana e utilizando o conjunto de descritores composto pelo conjunto simples excluindo-se o descritor Ret_t, reconhecendo corretamente 95.16% dos gestos e 80.65% das sequências de gestos nos vídeos.

Descritores	
Conjunto Complexo (uma mão)	Conjunto simples (duas mãos)
• 7 momentos centrais não escalados ϑ – $\vartheta_{02}, \vartheta_{03}, \vartheta_{11}, \vartheta_{12}, \vartheta_{20}, \vartheta_{21}, \vartheta_{30}$ • 7 momentos de HU φ – $\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7$ • 4 momentos invariantes a transformações afins I – I_1, I_2, I_3, I_4 • 4 momentos do contorno de uma região F – F_1, F_2, F_3, F_4 • 4 descritores de Fourier ω – $\omega_2, \omega_3, \omega_4, \omega_5$	• Posição das mãos em relação ao centro da face – $r_{\text{mão,face}}$ – $\alpha_{\text{mão,face}}$ • Direção do movimento – θ_t • Deslocamento e sua primeira diferença – D_t – A_t • Alongamento – Alg_t • Retangularidade – Ret_t • Compactação – Cpt_t • Direção – ϕ_t

Tabela 4.1: Descritores utilizados.

Descritores	ND	Rec. Max. (%)		NG
		G	S	
Todos	44	91.13	74.19	2
Conjunto Simples + ϑ	25	91.94	74.19	1
Conjunto Simples + φ	25	92.74	74.19	1
Conjunto Simples + I	22	94.35	77.42	1
Conjunto Simples + F	22	91.94	70.97	2
Conjunto Simples + ω	22	92.74	74.19	1
Conjunto Simples	18	94.35	77.42	1

Tabela 4.2: Resultado para os conjuntos de descritores gerados na primeira etapa. ND: Número de descritores; Rec. Max.: Reconhecimento Máximo; G: gestos; S: sequência de gestos; NG: Número de gaussianas; NS: Número de estados emissores dos HMMs.

Descritores	ND	Rec. Max. (%)		NG
		G	S	
Conjunto Simples - $r_{\text{mão,face}}$	16	92.74	74.19	1
Conjunto Simples - $\alpha_{\text{mão,face}}$	16	92.74	70.97	2
Conjunto Simples - θ_t	16	92.74	70.97	1
Conjunto Simples - D_t	16	94.35	77.42	1
Conjunto Simples - A_t	16	94.35	77.42	1
Conjunto Simples - Alg_t	16	94.35	77.42	1
Conjunto Simples - Ret_t	16	95.16	80.65	1
Conjunto Simples - Cpt_t	16	94.35	80.65	1
Conjunto Simples - ϕ_t	16	91.94	70.97	1

Tabela 4.3: Resultado para os conjuntos de descritores gerados na segunda etapa. ND: Número de descritores; Rec. Max.: Reconhecimento Máximo; G: gestos; S: sequência de gestos; NG: Número de gaussianas; NS: Número de estados emissores dos HMMs.

Descritores	ND	Rec. Max. (%)		NG
		G	S	
Conjunto Simples - Ret_t - $r_{\text{mão,face}}$	14	93.55	77.42	1
Conjunto Simples - Ret_t - $\alpha_{\text{mão,face}}$	14	92.74	74.19	1
Conjunto Simples - Ret_t - θ_t	14	94.35	77.42	1
Conjunto Simples - Ret_t - D_t	14	94.35	77.42	1
Conjunto Simples - Ret_t - A_t	14	94.35	77.42	1
Conjunto Simples - Ret_t - Alg_t	14	90.32	64.52	1
Conjunto Simples - Ret_t - Cpt_t	14	94.35	80.65	1
Conjunto Simples - Ret_t - ϕ_t	14	93.55	74.19	1

Tabela 4.4: Resultado para os conjuntos de descritores gerados na terceira etapa. ND: Número de descritores; Rec. Max.: Reconhecimento Máximo; G: gestos; S: sequência de gestos; NG: Número de gaussianas; NS: Número de estados emissores dos HMMs.

CAPÍTULO 5

Integrando a CNN ao Sistema

Este capítulo trata sobre a utilização de uma CNN no sistema de reconhecimento. A classificação continua sendo feita através dos HMMs, sendo a CNN utilizada como um novo descritor correspondente ao formato da mão. Na seção 5.1 são apresentadas e treinadas diferentes arquiteturas de CNNs para fazer o reconhecimento de diversas formas da mão. Na seção 5.2 são utilizados os resultados obtidos na seção 4.4, sendo integrado uma CNN aos descritores dos HMMs. São feitos novos testes e a rede de HMMs também é expandida de forma a ficar mais genérica.

5.1 CNN para o Reconhecimento das Formas da Mão

CNNs já provaram serem ótimas ferramentas para o reconhecimento de padrões em imagens. Neste trabalho as CNNs não são empregadas diretamente para o reconhecimento dos gestos, pois estes se desenvolvem no tempo e são modelados por HMMs, mas sim para a geração de descritores correspondentes ao formato da mão. As formas de mãos presentes nos gestos são analisadas e é gerada uma classe de saída correspondente a cada forma de mão.

5.1.1 Definição das Formas da Mão

A LIBRAS define um conjunto de formas que as mãos podem assumir na realização dos gestos. Para determinado número de gestos seria necessário então escolher somente o subconjunto de formas necessárias para esses gestos. Porém, aqui, as mãos estão contidas em uma imagem bidimensional, fazendo com que a mudança do ângulo de visão entre a câmera e a mão, a rotação e a translação de mão, gerem uma forma de mão, na imagem, totalmente diferente da forma correta. Devido a esse motivo foram escolhidas algumas formas que não fazem parte da libras, mas que se apresentam presentes nas imagens. As formas de mão escolhidas, onde cada uma representa uma classe são ilustradas na figura 5.1.

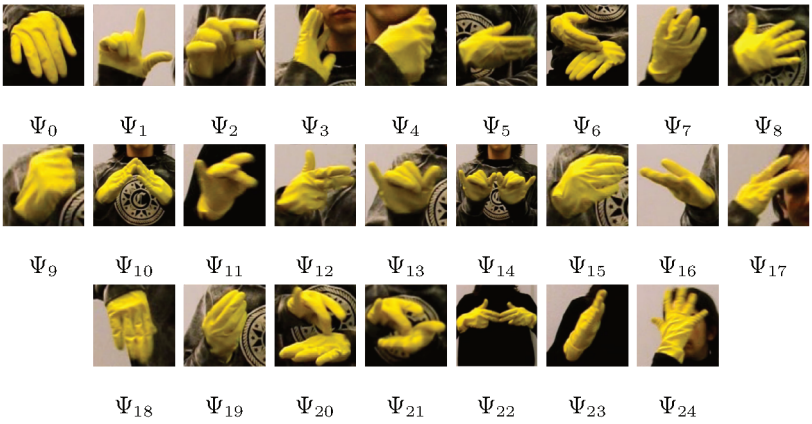


Figura 5.1: Tipos de configurações da mão principal.

São utilizadas somente imagens da mão principal, pois a mão secundária tem apenas uma função auxiliar no gesto. Porém quando as mãos se tocam, o algoritmo de detecção das mãos utiliza a mescla das duas mãos como um formato da mão principal, e devido a isso foram utilizadas algumas classes representando as mãos juntas, como Ψ_6, Ψ_{10}, etc.

5.1.2 Escolha da Arquitetura da CNN

A escolha da estrutura utilizada foi baseada em estruturas bem populares e que já demostraram um bom desempenho em trabalhos passados, como a LeNet-5 [50], AlexNet [42], VGG-16 [51] e GoogLeNet [52]. Aqui será apresentado uma breve descrição dessas redes, e em seguida serão apresentadas as redes utilizadas neste trabalho.

LeNet-5 é uma rede convolucional apresentada em 1998 para a classificação de imagens alfanuméricas e serviu de padrão para a maioria das estruturas que existem atualmente. Consiste de duas camadas convolucionais com uma camada de subamostragem (hoje também chamadas de pooling) após cada uma delas, seguidas de duas camadas totalmente conectadas mais a camada de saída, totalizando 5 camadas com parâmetros que devem ser treinados. A LeNet-5 original utiliza uma imagem de entrada de 32x32 pixels em tons de cinza e função de ativação sigmoide para os seus neurônios. Nas suas camadas de subamostragem utiliza average-pooling e funções sigmoides. Suas convoluções possuem *stride length* igual a 1 e não possuem *padding*.

AlexNet possui uma estrutura parecida com a LeNet-5, porém é uma rede mais profunda e com muito mais parâmetros. Desenvolvida para reconhecer imagens de três canais e 224x224 pixels. Esta rede ficou em primeiro lugar no desafio ILSVRC de 2012, no qual se deve classificar 100 mil imagens entre as mil classes possíveis, obtendo uma taxa de erro de 16,4% no estilo de classificação top-5. Esta rede utiliza filtros maiores (como 11x11), *stride length* maiores que 1x1, convoluções com *padding*, função de ativação RELU e max-pooling.

VGG-16 é uma rede com 16 camadas com parâmetros que devem ser treinados, e ficou em segundo lugar no ILSVRC de 2014 com uma taxa de erro igual a 7,32% no estilo de classificação top-5. Todas as suas camadas convolucionais possuem filtros 3x3, *stride length* 1x1 e *padding*, bem como as de max-pooling possuem filtro 2x2 e *stride length* 2x2. No lugar de utilizar campos locais receptivos de tamanho grande, como 5x5, 7x7, 11x11, como utilizado em outras redes, essa rede implementa várias camadas convolucionais de filtro 3x3 em sequência (mantendo as dimensões espaciais das camadas) com a finalidade de compensar o tamanho do filtro ao mesmo tempo em que diminui a complexidade computacional. A cada camada de subamostragem, a quantidade de canais dobra e o tamanho da camada é dividido pela metade. A rede é

composta por camadas convolucionais, algumas de max-pooling, e em seu final camadas totalmente conectadas.

Finalmente, a GoogLeNet, que ficou em primeiro lugar no ILSVRC de 2014 com uma taxa de erro igual a 6,67% no estilo de classificação top-5, utiliza uma abordagem um pouco diferente. Ela utiliza a arquitetura Inception, que utiliza campos locais receptivos de tamanhos diferentes. A GoogLeNet é composta de várias camadas Inception, juntamente com camadas convolucionais e max-pooling, totalizando 22 camadas com parâmetros.

Uma camada ou módulo Inception pode ser vista como uma subrede. O módulo Inception utilizado na GoogLeNet é ilustrado na figura 5.2. Como pode ser visto, a entrada do módulo passa por um banco de filtros que gera uma saída a partir da concatenação da saída dos filtros individuais. Então, enquanto uma camada convolucional comum gera uma saída a partir de apenas um tamanho de filtro, a saída do modulo Inception é a concatenação de camadas convolucionais com filtros de tamanho 1x1, 3x3 e 5x5. Baseando-se na ideia de que a operação de pooling se mostrou essencial nas CNNs, também utilizou-se um caminho de pooling, cuja saída também é concatenada na saída do módulo. Obviamente, para que as saídas possam ser concatenadas, todas as convoluções possuem *stride length* = 1x1 e *padding* de modo a preservar o tamanho da entrada. O operação de pooling também deve preservar o tamanho da entrada, possuindo assim um filtro 3x3, *stride length* = 1x1 e *padding*.

Como a saída da camada de pooling preserva a mesma quantidade de canais da entrada, adiciona-se uma convolução 1x1 (bloco P1X1) de modo a controlar a quantidade de canais correspondente ao pooling. Tudo isso gera uma saída que possui o mesmo tamanho da entrada, e uma quantidade de canais igual a soma da quantidade de canais dos blocos IC1X1, IC3X3, IC5X5 e P1X1.

Os blocos R3X3 e R5X5 existem para diminuir a complexidade computacional envolvida. Sem eles teríamos convoluções 3x3 e 5x5 aplicadas diretamente na camada de entrada, o que gera um alto custo computacional. Para amenizar este problema as convoluções 1x1 são utilizadas para reduzir a quantidade de canais da entrada. A ideia é que estas convoluções funcionem como o pescoço de uma garrafa, compactando as informações nas áreas de alto custo computacional, para

depois expandir novamente. O único parâmetro que devemos controlar nos módulos é a quantidade de canais de cada bloco.

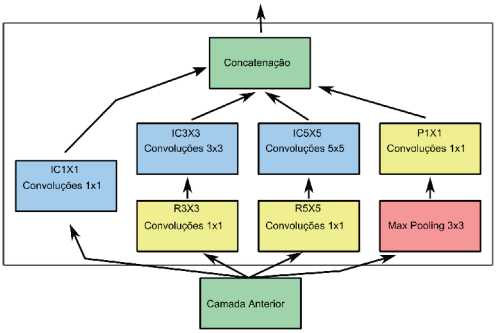


Figura 5.2: Módulo Inception.

5.1.3 CNNs Implementadas

Atualmente a camada de subamostragem mais utilizada é a max-pooling e a função de ativação é a RELU, pois esta não apresenta os problemas de lentidão na aprendizagem gerados pela função sigmoide ou Tanh. Por isso, todas a redes implementadas neste trabalho utilizaram max-pooling, RELU e softmax na saída. As imagens utilizadas possuem somente um canal em tons de cinza com 64x64 pixels. Essas possuem uma borda de 3 pixels com a imagem da mão em seu centro.

A LeNet-5 utilizada neste trabalho é apresentada na figura 5.3. Aqui utilizou-se uma imagem de entrada com 64x64 pixels, pois o resultado obtido foi melhor que com uma imagem de 32x32. O número abaixo de cada camada representa o número de canais ou filtros e a quantidade de neurônios em cada canal, por exemplo: 6@60x60 significa que esta camada possuí 6 canais com 60x60 neurônios em cada um. C1, S1 e F3 estão para camada de convolução 1, camada de max-pooling 1 e camada totalmente conectada 3. A tabela 5.1 contém o número de canais, o tamanho da camada, o tamanho do filtro, o *stride length* e o tamanho do *padding* para cada camada da rede.

Também foi implementado uma variação da AlexNet, devido ao tamanho da imagem de entrada utilizada neste trabalho ser menor, excluindo-se uma camada de pooling e diminuindo o tamanho dos filtros

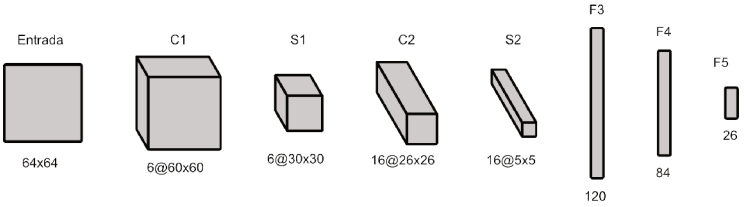


Figura 5.3: LeNet-5 utilizada no trabalho.

	Canais	Tamanho	Filtro	Stride	Padding
Entrada	1	64x64	-	-	-
C1	6	60x60	5x5	1x1	-
S1	6	30x30	2x2	2x2	-
C2	16	26x26	5x5	1x1	-
S2	16	5x5	2x2	2x2	-
F3	-	120	-	-	-
F4	-	84	-	-	-
F5	-	25	-	-	-

Tabela 5.1: Parâmetros da LeNet-5 utilizada no trabalho.

dos canais nas camadas. Essa rede é representada na figura 5.4 e seus parâmetros são dados na tabela 5.2.

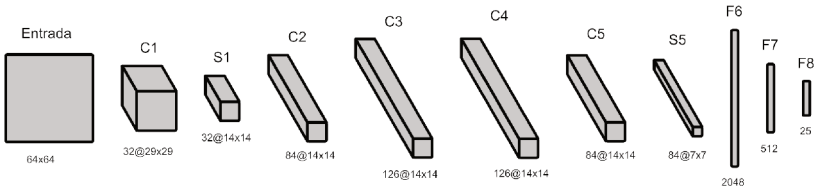


Figura 5.4: Variação da AlexNet utilizada no trabalho.

A próxima rede implementada foi uma variação da VGG-16, mas composta por 13 camadas, que aqui foi chamada de VGG-13. Ela é semelhante a VGG-16, porém menos profunda devido a imagem de entrada ter um tamanho menor. A tabela 5.3 mostra os seus parâmetros.

Para finalizar, a última estrutura implementada foi a GoogLeNet,

	Canais	Tamanho	Filtro	Stride	Padding
Entrada	1	64x64	-	-	-
C1	32	29x29	8x8	2x2	-
S1	32	14x14	3x3	2x2	-
C2	84	14x14	5x5	1x1	2
C3	126	14x14	1x1	1x1	-
C4	126	14x14	3x3	1x1	1
C5	84	14x14	3x3	1x1	1
S5	84	7x7	2x2	2x2	-
F6	-	2048	-	-	-
F7	-	512	-	-	-
F8	-	25	-	-	-

Tabela 5.2: Parâmetros da AlexNet utilizada no trabalho.

representada na figura 5.5 e na tabela 5.4. Nota-se uma nova nomenclatura, sendo I2 igual a módulo Inception da segunda camada e IC1x1, R3x3, IC3x3, R5x5, IC5x5 e P1x1 as camadas do respectivo módulo. No início da rede há uma camada convolucional seguida de max-pooling, isso não é necessário mas é útil para diminuir o tamanho da camada de 64x64 para 32x32, o que por consequência diminui a quantidade e parâmetros que devem ser calculados.

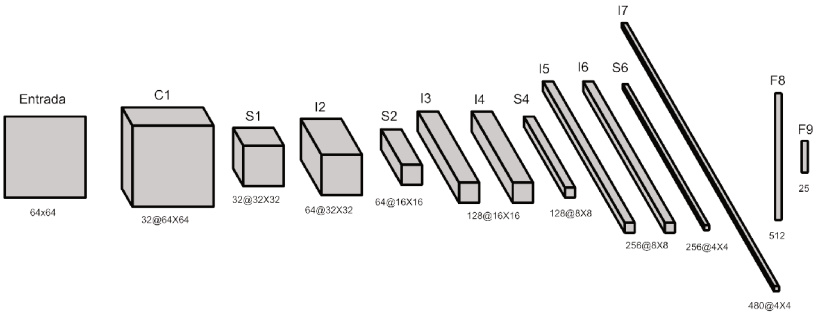


Figura 5.5: Variação da GoogLeNet utilizada no trabalho.

	Canais	Tamanho	Filtro	Stride	Padding
Entrada	1	64x64	-	-	-
C1	32	64x64	3x3	1x1	1
C2	32	64x64	3x3	1x1	1
S2	32	32x32	2x2	2x2	-
C3	64	32x32	3x3	1x1	1
C4	64	32x32	3x3	1x1	1
S4	64	16x16	2x2	2x2	-
C5	128	16x16	3x3	1x1	1
C6	128	16x16	3x3	1x1	1
C7	128	16x16	3x3	1x1	1
S7	128	8x8	2x2	2x2	-
C8	256	8x8	3x3	1x1	1
C9	256	8x8	3x3	1x1	1
C10	256	8x8	3x3	1x1	1
S10	256	4x4	2x2	2x2	-
F11	-	2048	-	-	-
F12	-	2048	-	-	-
F13	-	25	-	-	-

Tabela 5.3: Parâmetros da VGG-13 utilizada no trabalho.

5.1.4 Treinamento e Testes

Após a rotulação e extração das imagens dos vídeos do conjunto de treinamento 1 e do conjunto de validação (detalhes sobre os vídeos na seção 6.1), foram obtidas 9038 imagens, sendo separadas 6175 para treinamento e 2863 para testes. As redes foram treinadas utilizando-se os algoritmos de *deep learning* da Dlib através de uma GeForce 1050 Ti.

Para o treinamento adotou-se regularização com *weight dacay* = $5 * 10^{-4}$, mini-batch = 256 e uma taxa de aprendizagem inicial igual a 10^{-2}. Quando a função de custo da rede apresenta pouca variação ou oscilação durante várias épocas, a taxa de aprendizagem é multiplicada

	Canais	Tamanho	Filtro	Stride	Padding	IC1x1	R3x3	IC3x3	R5x5	IC5x5	P1x1
Entrada	1	64x64	-	-	-	-	-	-	-	-	-
C1	32	64x64	7x7	1x1	3	-	-	-	-	-	-
S1	32	32x32	2x2	2x2	-	-	-	-	-	-	-
I2	64	32x32	-	-	-	16	16	24	6	8	16
S2	64	16x16	2x2	2x2	-	-	-	-	-	-	-
I3	128	16x16	-	-	-	32	32	56	8	16	24
I4	128	16x16	-	-	-	16	38	64	12	24	24
S4	128	8x8	2x2	2x2	-	-	-	-	-	-	-
I5	256	8x8	-	-	-	88	44	88	16	32	48
I6	256	8x8	-	-	-	64	56	112	24	32	32
S6	256	4x4	2x2	2x2	-	-	-	-	-	-	-
I7	480	4x4	-	-	-	140	106	212	26	64	64
F8	-	512	-	-	-	-	-	-	-	-	-
F9	-	25	-	-	-	-	-	-	-	-	-

Tabela 5.4: Parâmetros da GoogLeNet utilizada no trabalho.

por 0.1. Esse processo é repetido até que a taxa de aprendizagem atinja o valor de 10^{-6}, e o treinamento é encerrado.

A tabela 5.5 mostra o resultado das redes no conjunto de treinamento e no de testes. A rede que obteve a melhor acurácia foi a AlexNet, provavelmente devido a baixa quantidade de amostras de treinamento utilizadas para treinar as redes mais profundas. A maioria dos erros foi devido a dois fatores:

- Certas imagens de classes diferentes se tornam muito parecidas sob certo ângulo de visão;

- Erros do algoritmo de extração das mãos, causando muita distorção devido ao borrão do movimento ou outros fatores, como a iluminação.

Os resultados parecem promissores. Talvez variando-se as estruturas das CNNs e seus parâmetros de treinamento os mesmos pudessem ser melhorados um pouco, porém como foi dito acima, a maioria dos erros são devidos a erros do algoritmo de extração das mãos ou oclusão da mão, tornado difícil a tarefa de classificação até mesmo para uma pessoa.

	Precisão Treinamento	Precisão Teste	Erros no Teste
LeNet-5	1	0,9795	60
AlexNet	1	0,9839	47
VGG-13	1	0,9771	67
GoogLeNet	1	0,9822	52

Tabela 5.5: Resultados Das Redes

5.2 Utilizando a CNN nos HMMs

Nos testes realizados na seção 4.4.2, o melhor desempenho foi de 95.16% na classificação dos gestos e 80.65% na classificação das sequências de gestos, utilizando HMMs de estrutura esquerda-direita com 11 estados emissores para os gestos, 4 para os sub-gestos e 1 estado para a posição de repouso. Esse desempenho é dado para uma rede de HMMs bem limitada, sendo definida como uma sequência de quatro gestos entre a posição de repouso no início e final do vídeo. Para redes menos limitadas a performance cai drasticamente. Notou-se que a maioria dos erros ocorria entre gestos razoavelmente similares, como os gestos preto e verdade, que possuem formas de mão parecidas e são realizados em posições similares. Foram utilizados 16 descritores compostos pelo seguinte conjunto para cada uma das mãos:

- Distância da mão em relação ao centro da face $r_{\text{mão,face}}$;
- Ângulo da mão em relação ao centro da face $\alpha_{\text{mão,face}}$;
- Direção do movimento θ_t;
- Deslocamento (velocidade) D_t;
- Primeira diferença do deslocamento (aceleração) A_t;
- Alongamento Alg_t;
- Compactação Cpt_t;
- Direção ϕ_t;

Ao contrário da seção 5.1, onde interpreta-se cada saída da CNN como correspondente a uma classe discreta da forma da mão, aqui

se utiliza os valores das 25 saídas softmax como descritores contínuos para os HMMs. As saídas foram adicionadas ao conjunto de descritores acima, totalizando 41 descritores. O resultado obtido foi uma pequena queda na taxa de reconhecimento em relação ao conjunto de 16 descritores. Enquanto antes os erros eram entre gestos similares, agora apareceram erros entre gestos completamente diferentes, como verdade e pessoa. Com essa informação chegou-se a conclusão de que provavelmente os movimentos de transição entre os gestos estão interferindo no reconhecimento, uma vez que eles não foram modelados e os HMMs dos gestos devem englobá-los no reconhecimento. Então decidiu-se modelar os movimentos de transição por um único HMM de 3 estados emissores, e utilizá-lo como um modelo de ligação entre os gestos. A ideia é que como esse único HMM é modelado para todos os movimentos de transição, seus estados possuam uma variância muito grande, aproximando-se de uma distribuição uniforme.

Com a implementação do movimento de transição na rede de HMMs, foram realizados novos testes com novas combinações de descritores. A tabela 5.6 mostra o resultado dos testes para três diferentes estruturas de redes de HMMs (rede1, rede2 e rede3), onde os conjuntos de descritores utilizados na tabela estão enunciados a seguir:

- Conjunto A: $r_{\text{mão,face}}$, $\alpha_{\text{mão,face}}$, θ_t, D_t, A_t, Alg_t, Cpt_t, ϕ_t;
- Conjunto B: $r_{\text{mão,face}}$, $\alpha_{\text{mão,face}}$, θ_t, Alg_t, Cpt_t, ϕ_t;
- Conjunto C: $r_{\text{mão,face}}$, $\alpha_{\text{mão,face}}$, θ_t;
- Conjunto D: $r_{\text{mão,face}}$, $\alpha_{\text{mão,face}}$, θ_t, ϕ_t;

Todas as estruturas de redes presumem que em cada vídeo as mãos comecem e terminem na posição de repouso, e são definidas a seguir:

- A rede 1 é representada na figura 5.6. Esta estrutura de rede é semelhante a utilizada nos testes anteriores, porém contendo modelos para os movimentos de transição entre os gestos. Essa estrutura é fixa, devendo os vídeos conter sempre uma sequência de quatro gestos.
- A rede 2 (figura 5.7) possui uma estrutura semelhante a anterior, mas contém transições de escape entre os gestos e o movimento

para a posição de repouso. Dessa forma, essa estrutura permite que os vídeos possuam de um a quatro gestos.

- A rede 3 (figura 5.8) não possuí nenhum limite de gestos, devendo o vídeo conter no mínimo um gesto.

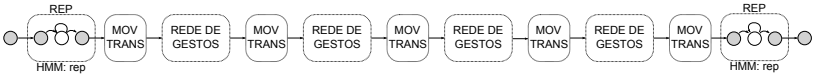


Figura 5.6: Estrutura de quatro gestos.

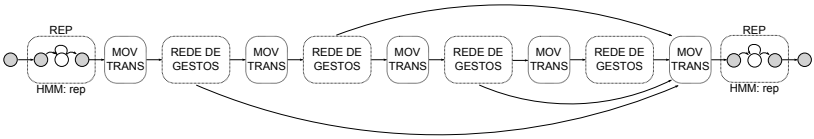


Figura 5.7: Estrutura de um a quatro gestos.

Figura 5.8: Estrutura livre.

Descritores	ND	Rede 1		Rede 2		Rede 3	
		G	S	G	S	G	S
Conjunto A	16	98.39	93.55	97.58	90.32	97.58	87.10
CNN	25	98.39	93.55	98.39	93.55	98.39	87.10
Conjunto A + CNN	41	100	100	99.19	96.77	99.19	87.10
Conjunto B	12	97.58	90.32	97.58	90.32	97.58	87.10
Conjunto B + CNN	37	100	100	100	100	100	90.32
Conjunto C + CNN	31	99.19	96.77	99.19	96.77	99.19	83.87
Conjunto D + CNN	33	100	100	100	100	100	90.32

Tabela 5.6: Resultado para diversos descritores em redes com modelo do movimento de transição. ND: Número de descritores; G: Taxa de classificação dos gestos (%); S: Taxa de classificação da sequência de gestos (%).

Analisando a tabela 5.6, primeiramente utilizou-se o conjunto A (os mesmos 16 descritores utilizados anteriormente) e obteve-se uma taxa superior à taxa obtida sem o modelo dos gestos de transição na rede 1, principalmente no reconhecimento das sequências de gestos. Já nas rede 2 e 3 esse resultado fui muito superior ao modelo sem movimentos de transição. A tabela de confusão 5.7 mostra o resultado do classificador utilizando o conjunto A e a rede 3. Podemos ver que não há nenhum erro de substituição dos gestos, apenas 3 eliminações e uma inserção na sequência de gestos. As eliminações indicam que esses descritores não são suficientes para discriminar uma sequência de gestos. A inserção ocorre devido ao movimento repetitivo do gesto carro, e pode ser corrigida com um pós-processamento, como será mencionado mais adiante.

		Sinal Predito															
		barato	bonito	branco	casa	carro	coisa	fazer	filho	feliz	hoje	muito	pessoa	praia	preto	verdade	Eliminações
Sinal Real	barato	9															
	bonito		5														
	branco			8													1
	casa				12												
	carro					13											
	coisa						6										
	fazer							3									1
	filho								14								1
	feliz									4							
	hoje										9						
	muito											7					
	pessoa												10				
	praia													8			
	preto														6		
	verdade															7	
	Inserções					1											

Tabela 5.7: Tabela de confusão para os descritores do conjunto A - Rede 3.

Utilizando somente os descritores da CNN obteve-se um resultado levemente superior nas redes 2 e 3. Pode se ver na tabela de confusão 6.4 que os erros de eliminações foram zerados, porém foram gerados novos erros de substituição e inserção. Dessa forma, utilizar somente a informação do formato da mão não parece ser suficiente para discriminar certos sinais. Finalmente, a combinação dos 16 descritores com os 25 da CNN gerou resultados superiores ao seu uso isolado, chegando a 100% de classificação correta das sequências de gestos na rede 1, e 96.77% na rede 2.

Dentre os descritores do conjunto A, o deslocamento e a sua primeira diferença são descritores altamente ruidosos. Por isso pensou-se em retirá-los e verificar o desempenho das redes. Verificou-se que a

		Sinal Predito															
		barato	bonito	branco	casa	carro	coisa	fazer	filho	feliz	hoje	muito	pessoa	praia	preto	verdade	Eliminações
Sinal Real	barato	9															
	bonito		5														
	branco			9													
	casa				12												
	carro					13											
	coisa						6										
	fazer							4									
	filho								14					1			
	feliz									4							
	hoje										9						
	muito											6			1		
	pessoa												10				
	praia													8			
	preto														6		
	verdade															7	
	Inserções					2				1	1						

Tabela 5.8: Tabela de confusão para os descritores da CNN - Rede 3.

retirada desses descritores fez a taxa de reconhecimento cair na rede 1, implicando que mesmo ruidosos eles carregam alguma informação. Porém quando o conjunto B é combinado com as saídas da CNN, a taxa de reconhecimento atinge seu pico, chegando a 100% em quase todos os campos. A tabela de confusão 5.9 apresenta esses resultados, mostrando que não houveram erros de substituição e nem de eliminação.

		Sinal Predito															
		barato	bonito	branco	casa	carro	coisa	fazer	filho	feliz	hoje	muito	pessoa	praia	preto	verdade	Eliminações
Sinal Real	barato	9															
	bonito		5														
	branco			9													
	casa				12												
	carro					13											
	coisa						6										
	fazer							4									
	filho								15								
	feliz									4							
	hoje										9						
	muito											7					
	pessoa												10				
	praia													8			
	preto														6		
	verdade															7	
	Inserções					1					1			1			

Tabela 5.9: Tabela de confusão para os descritores do conjunto B + CNN - Rede 3.

Seguindo o mesmo pensamento, no conjunto C retira-se os descritores que carregam informação da forma da mão, ficando apenas aqueles correspondentes a trajetória da mão e a saída da CNN (os quais são os principais descritores da forma da mão neste trabalho). Isso causou uma queda na taxa de reconhecimento, indicando que os descritores retirados podem ser considerados como um complemento às

saídas da CNN. Porém dentre estes descritores, talvez o que carregue mais informações seja o referente ao ângulo da mão, devido a ser menos suscetível a interferências. Então o conjunto D é semelhante ao C, porém mantém-se o descritor relativo ao ângulo da mão. O resultado foi idêntico ao obtido utilizando o conjunto B + CNN, com a diferença de utilizar 4 descritores a menos.

Como a rede 3 possui uma estrutura sem restrições quanto a quantidade de gestos, muitos dos erros são devidos aos gestos com movimentos repetitivos. Mesmo sendo modelados por HMMs com transição de retorno, algumas vezes o classificador reconhece esses gestos com uma sequência dos mesmos. Se admitirmos que um gesto não pode ser repetido em sequência, podemos aplicar um pós-processamento eliminando essas repetições. A tabela 5.10 mostra o resultado desse pós-processamento ao reconhecimento das sequências de gestos da rede 3.

Descritores	ND	Rede 3 - S
Conjunto A	16	90.32
CNN	25	93.55
Conjunto A + CNN	41	96.77
Conjunto B	12	90.32
Conjunto B + CNN	37	100
Conjunto C + CNN	31	96.77
Conjunto D + CNN	33	100

Tabela 5.10: Resultado do pós-processamento na rede 3. ND: Número de descritores; S: Taxa de classificação da sequência de gestos (%).

Com base nesses testes pode-se concluir que dentre os 8 descritores utilizados, os relativos a posição, direção do movimento e o ângulo da mão, carregam informações suficientes para um bom desempenho do classificador quando adicionados aos descritores da CNN. Além disso, a utilização de um modelo para os movimentos de transição entre os gestos se mostrou fundamental para um melhor desempenho.

CAPÍTULO 6

Testes e Resultados

Este capítulo traz informações sobre os vídeos utilizados em todo o trabalho, bem como os resultados finais obtidos. O sistema é testado em um novo conjunto de vídeos, o qual não contém nenhum vídeo utilizado durante o treinamento e validação. A seção 6.1 apresenta os gestos utilizados correspondentes aos sinais da LIBRAS, bem como detalhes sobre os vídeos e a quantidade de sinais presentes nos mesmos. Na seção 6.2 são apresentados os resultados dos novos testes, enquanto que a seção 6.3 apresenta o tempo de processamento dos quadros e as especificações da máquina utilizada.

6.1 Informações Sobre os Vídeos

Todos os vídeos deste trabalho são compostos por quatro gestos selecionados aleatoriamente detre os gestos do vocabulário utilizado. Os gestos em um vídeo são realizados continuamente, ou seja, sem pausas entre eles. Para facilitar o projeto, as mãos iniciam e terminam o vídeo na posição de repouso. As características dos vídeos são dadas na tabela 6.1.

O vocabulário utilizado é composto de quinze sinais (gestos) se-

Largura	640 pixels
Altura	480 pixels
Taxa de aquisição	30 fps
Número de gestos por vídeo	4

Tabela 6.1: Características dos vídeos.

lecionados aleatórios dentre o conjunto de sinais da LIBRAS. Esses representam um conjunto de gestos complexos devido ao alto grau de liberdade das mãos e movimentos complicados realizados três dimensões, além de existirem gestos que utilizam tanto uma quanto as duas mãos. O dicionário utilizado (apresentado na tabela 6.2) contém cinco gestos realizados com apenas a mão principal e dez com as duas mãos.

Dicionário de sinais	
Uma mão	Duas mãos
barato, bonito, feliz, filho, pessoa	branco, carro, casa, coisa, fazer, hoje, muito, praia, preto, verdade

Tabela 6.2: Sinais da LIBRAS utilizados.

Os vídeos utilizados neste trabalho foram divididos em quatro conjuntos sem intersecção: conjunto de treinamento 1, conjunto de treinamento 2, conjunto de validação e conjunto de teste. Os testes realizados nos capítulos 4 e 5 utilizaram somente o conjunto de treinamento 1 e o conjunto de validação. Já o conjunto de treinamento 2 e o conjunto de teste são utilizados neste capítulo, adicionalmente aos outros dois. A tabela 6.3 mostra a quantidade de vídeos e o número de incidência dos gestos em cada conjunto.

Os conjuntos de treinamento 1 e de validação foram utilizados durante o treinamento e testes das redes nos capítulos 4 e 5, a fim de detectar uma estrutura adequada de HMMs e CNN, bem como o desempenho dos diversos descritores. Os vídeos desses conjuntos possuem uma iluminação controlada e a distância do gesticulador em relação à câmera varia, como mostra o figura 6.1.

Já os vídeos dos conjunto de treinamento 2 e de teste são utilizados

	Conj. Treinamento 1	Conj. Treinamento 2	Conj. Validação	Conj. Teste
Nº vídeos	60	16	31	24
Nº gestos	244	64	124	96
barato	11	2	9	5
bonito	23	7	5	5
branco	16	2	9	5
carro	18	9	13	10
casa	11	2	12	10
coisa	24	6	6	4
fazer	17	3	4	3
feliz	18	5	4	4
filho	14	4	15	3
hoje	11	3	9	8
muito	12	2	7	4
pessoa	14	6	9	5
praia	21	4	9	8
preto	13	5	6	5
verdade	17	4	7	7

Tabela 6.3: Conjuntos de Vídeos

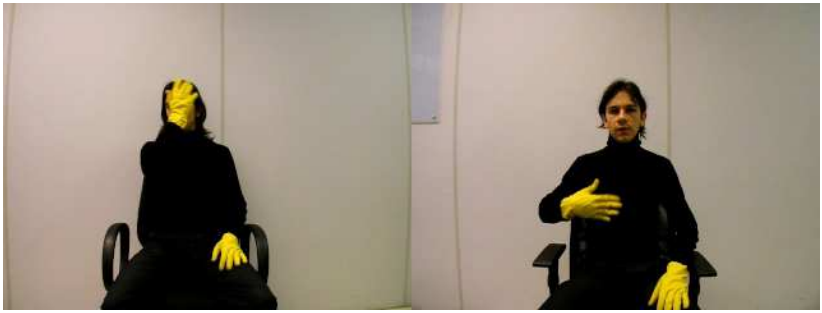


Figura 6.1: Distância do gesticular à câmera.

neste capítulo para realizar os testes das redes construídas anteriormente. Esses vídeos foram gravados muito tempo depois, e devido a isso a gesticulação varia bastante em relação aos anteriores. Além disso a câmera utilizada e a qualidade da iluminação são muito inferiores em relação aos vídeos do conjunto de treinamento 1 e de validação. A figura 6.2 mostra a diferença entre a qualidade da imagem dos vídeos antigos e dos vídeos do conjunto de treinamento 2 e de teste.

Figura 6.2: Diferença de qualidade dos vídeos. Comparação entre a qualidade da imagem dos vídeos relativos aos conjuntos de treinamento 1 e de validação (esquerda) e dos vídeos relativos aos conjuntos de treinamento 2 e de teste.

Devido a essas grandes diferenças entre o conjunto de teste e o de treinamento 1, foi gerado o conjunto de treinamento 2, o qual é adicionado no treinamento para que esse possa se generalizar melhor as diferentes condições.

6.2 Testes

No capítulo 5.2 foram realizados testes em três diferentes redes de HMMs e com diferentes conjuntos de descritores. A rede 1 exige que os vídeos contenham quatro gestos, a rede 2 permite que os vídeos possuam de um a quatro gestos, já a rede 3 permite um ou mais gestos por vídeo. Dentre os conjuntos de descritores testados, o conjunto B + CNN e o conjunto D + CNN geraram os melhores resultados.

Para testar o sistema, a CNN e os HMMs foram treinados novamente, agora utilizando todos os vídeos não pertencentes ao conjunto de teste (conjuntos de treinamento 1, treinamento 2 e validação). Em seguida a CNN foi testada com as poses dos vídeos de teste e os HMMs com os vídeos de teste.

6.2.1 Resultados dos Testes da CNN

Anteriormente, no conjunto de validação, a CNN com estrutura baseada na AlexNet (rede selecionada para o trabalho) obteve uma taxa de classificação de 98.4% (tabela 5.5). Após o novo treinamento, a mesma rede obteve uma taxa de classificação de 96.59% no conjunto de teste, errando 83 das 2434 poses testadas.

A tabela de confusão 6.4 mostra o número de cada pose utilizada e o valor predito. Dentre os 83 erros, 31 são devido a classificação incorreta entre as poses Ψ_{20} e Ψ_{21}, ilustradas na figura 5.1. Essas duas poses são muito similares, e como as amostras para o treinamento da CNN foram extraídas dos vídeos, a pose Ψ_{20} possui uma duração muito maior no sinal "preto"que a pose Ψ_{21} no sinal "verdade". Como consequência existem poucas amostras de treinamento da pose Ψ_{21} em relação a pose Ψ_{20}. Um aumento na quantidade de amostras de treinamento diminuiria esse erro.

		Pose Predita																							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Todas
Pose Real	0	378																									378
	1		42										3														45
	2			45													2										47
	3				47																						47
	4					46																					46
	5						21				1		1				1								1	1	26
	6							57																			57
	7								43		1									2						1	47
	8									273																	273
	9										185									4							189
	10											107															107
	11												92														92
	12													53													53
	13										1				36						3				1		41
	14									7						22											29
	15										1						105										106
	16																	186									186
	17																		104								104
	18										12									151					1		164
	19										1						2			3	39						45
	20																					131	7				138
	21																					24	44				68
	22																							17			17
	23												2												101		103
	24																									26	26
	Todas	378	42	45	47	46	21	57	43	280	202	107	98	53	36	22	110	186	104	160	42	155	51	17	104	28	2434

Tabela 6.4: Tabela de confusão da CNN.

6.2.2 Resultados dos Testes dos HMMs

As três redes de HMMs foram testadas no conjunto de teste, para os descritores que geraram os melhores resultados no capítulo 5.2 (conjunto B + CNN e o conjunto D + CNN). Além disso também foram

utilizados os descritores do conjunto A e os descritores da CNN isolados, para fins de comparação. Os resultados obtidos são dados na tabela 6.5.

Descritores	ND	Rede 1		Rede 2		Rede 3	
		G	S	G	S	G	S
Conjunto A	16	88.54	66.67	85.42	58.33	85.42	58.33
CNN	25	98.96	95.83	98.96	95.83	98.96	95.83
Conjunto B + CNN	37	100	100	100	100	100	100
Conjunto D + CNN	33	100	100	100	100	100	95.83

Tabela 6.5: Resultado das Redes de HMMs no Conjunto de Teste. ND: Número de descritores; G: Taxa de classificação dos gestos (%); S: Taxa de classificação da sequência de gestos (%).

Ao comparar-se os resultados obtidos nos vídeos de teste, utilizando o conjunto A, com os resultados obtidos nos vídeos de validação (tabela 5.6), pode-se notar a grande queda na taxa de reconhecimento. Isso nos diz que os descritores correspondentes ao movimento e os descritores mais simples da forma da mão não são suficientes para representar esse conjunto de sinais complexos, havendo diversos erros de substituição, inserção e exclusão de sinais.

Ainda observando a tabela 6.5, os descritores gerados pela CNN se mostram muito importantes, pois sozinhos possibilitam a discriminação de quase todos os gestos apenas utilizando a informação da forma da mão. O único erro obtido foi o sinal "preto"ser reconhecido como "verdade"em um dos vídeos. Isso aconteceu porque, como foi explicado na seção 6.2.1, a CNN possui um erro considerável no reconhecimento das poses correspondentes a esses sinais. É claro que em um vocabulário maior onde fossem utilizados mais sinais que repetissem a mesma forma da mão essa performance tenderia a cair, sendo imprescindível o uso de outros descritores.

Adicionando-se os descritores do conjunto B aos gerados pela CNN, introduz-se informações sobre o movimento das mãos e mais informações sobre a sua forma. Esse ganho de informação permitiu um reconhecimento de 100% dos vídeos. Finalmente o conjunto D retira os descritores Alg_t e Cpt_t, fazendo com que em um dos vídeos sejam reconhecidos dois sinais "preto"em sequência. Admitindo-se que o sistema não per-

mite dois sinais iguais em sequência, um pós-processamento eliminaria esse erro.

6.3 Tempo de Processamento

O trabalho se propunha a criar um sistema capaz de reconhecer gestos em tempo real. O funcionamento do sistema consiste em adquirir um vídeo com uma sequência de gestos e logo em seguida gerar a descrição. Para isso é feita a extração dos descritores quadro a quadro, sendo necessário que o tempo de processamento dos quadros seja menor que o tempo de aquisição dos mesmos. Ao final do vídeo, todos os descritores extraídos estão a disposição da rede de HMM para que seja realizado o reconhecimento. Seguem os dados referentes a máquina utilizada e aos tempos obtidos:

- Processador Intel Core i3-7100;
- Windows 10 x64;
- 8Gb RAM;
- Placa de vídeo NVIDIA GeForce GTX 1050 Ti;
- Tempo de aquisição dos vídeos: 30fps = 33.33ms por quadro;
- Tempo de processamento e extração dos descritores (conjunto A + CNN) de cada quadro: média de 5.59ms por quadro.
- Tempo para o reconhecimento: não foi encontrado nenhum comando no HTK para medir o tempo da decodificação, porém foi medido o tempo total de execução da ferramenta de decodificação, o que envolve todo um processo de inicialização. Esse tempo foi de aproximadamente 531ms para cada sequência de gestos.

Como pode ser visto, o tempo de processamento é consideravelmente menor que o tempo de aquisição, o que permite a execução em tempo real. Já o reconhecimento ocorre após o processamento do vídeo e em um intervalo de tempo muito pequeno.

CAPÍTULO 7

Conclusão

Este trabalho apresentou um sistema capaz de reconhecer uma sequência de gestos dinâmicos complexos, como os sinais da LIBRAS, utilizando apenas uma câmera comum como sensor. Os gestos são realizados continuamente, sem uma pausa intencional entre eles. Primeiramente é feita a detecção e o rastreamento, quadro à quadro, das mãos e da face utilizando técnicas baseadas na cor dos pixels e em algumas regras. Diversos descritores são extraídos a cada quadro dos vídeos.

Cada um dos quinze sinais (gestos) do vocabulário utilizado neste trabalho foi modelado por um HMM do tipo esquerda-direita ou esquerda-direita com transição de retorno, sendo que todos os HMMs são conectados para formar uma rede de HMMs que compõem o classificador. Além dos gestos, também se mostrou fundamental um modelo para os movimentos de transição entre os gestos quando não se sabe o número de gestos no vídeo.

Para verificar tanto a acurácia quanto a capacidade de generalização do sistema, os vídeos foram divididos em três conjuntos: de treinamento, validação e teste. Todo o treinamento dos HMMs e da CNN foi feito utilizando o conjunto de treinamento e de validação. O conjunto de teste foi utilizado somente no final, após a estrutura dos HMMs e

da CNN já ter sido escolhida e treinada.

Foram feitos testes para determinar quais os descritores e qual a quantidade de estados mais adequada para modelar os HMMs. Um tipo especial de descritor utilizado foi as saídas de uma CNN, treinada para reconhecer as diferentes formas da mão que compõem os gestos. A utilização das CNNs gerou um aprimoramento muito importante na classificação final, chegando a obter uma acurácia de 100% no conjunto de teste. Além disso, o sistema de reconhecimento tem capacidade para ser executado em tempo real, possuindo um tempo de processamento de aproximadamente 5.6ms por quadro na máquina em que foi testado.

Apesar da alta acurácia obtida e do baixo tempo de processamento, o sistema ainda possui várias limitações: foram utilizadas luvas amarelas para facilitar o reconhecimento nos casos em que as mãos ocluem a face, e os vídeos foram gravados com um fundo estático e uma iluminação controlada; devido ao algoritmo utilizado para rastrear as mãos, não são permitidos gestos em que as mãos se ocluam e em seguida se cruzem (esse problema não foi tratado, pois gestos desse tipo não são comuns); foi utilizado somente um gesticulador e uma cor de pele; o sistema exige que o gesticulador esteja sentado em frente a câmera e que, no início e fim dos vídeos, as mãos estejam em uma posição de repouso.

Trabalhos futuros

Como futuras linhas de pesquisa, sugere-se o seguinte.

(*i*) Utilizar um algoritmo melhor para fazer a detecção e o rastreamento das mãos, de forma que não seja necessário o uso de luvas coloridas e que se adapte melhor a situações menos controladas. Além disso, que não exija que o gesticulador esteja sentado e nem que as mãos estejam em uma posição determinada no inicio e fim dos vídeos. Isso permitiria que o sistema fosse utilizado em diversas aplicações práticas, e não somente teóricas.

(*ii*) Utilizar outros sensores, como o Kinect, capazes de obter descritores mais precisos para usar em conjunto com a CNN.

(*iii*) Utilizar o sistema criado para uma situação prática, como o controle de periféricos através de gestos.

(*iv*) Testar o sistema com uma quantidade maior de gestos e com mais gestos por vídeos.

(*v*) Tornar o sistema multi-usuário, adaptando o algoritmo para detectar e rastrear as mãos e face de pessoas com diferentes tons de pele, e que o classificador seja capaz de se generalizar ou se adaptar aos diferentes usuários.

Referências bibliográficas

[1] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (Pt.1)*. Wiley-Interscience, 2 edition, 11 2000.

[2] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, 4 edition, 11 2008.

[3] Joceli Mayer and Vinicius Breda. Continuous gesture recognition using hidden markov models. In *The 2014 International Conference on Image Processing, Computer Vision, and Pattern Recognition*, pages 442–443, 2014.

[4] Praveen Kakumanu, Sokratis Makrogiannis, and Nikolaos Bourbakis. A survey of skin-color modeling and detection methods. *Pattern recognition*, 40(3):1106–1122, 2007.

[5] Xiaojin Zhu, Lei Yang, and Alex Waibel. Segmenting hands of arbitrary color. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 446–453. IEEE, 2000.

[6] Hedieh Sajedi, Mehran Najafi, and Shohreh Kasaei. A boosted skin detection method based on pixel and block information. In *Image and Signal Processing and Analysis, 2007. ISPA 2007. 5th International Symposium on*, pages 146–151. IEEE, 2007.

[7] Mehran Fotouhi, Mohammad H Rohban, and Shohreh Kasaei. Skin detection using contourlet-based texture analysis. In *Digital Telecommunications, 2009. ICDT'09. Fourth International Conference on*, pages 59–64. IEEE, 2009.

[8] Xiaohua Wang, Xi Zhang, and Jingliang Yao. Skin color detection under complex background. In *Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on*, pages 1985–1988. IEEE, 2011.

[9] A Nadian Ghomsheh, A Talebpour, and M Basseri. Regional skin detection based on eliminating skin-like lambertian surfaces. In *Computers & Informatics (ISCI), 2011 IEEE Symposium on*, pages 307–312. IEEE, 2011.

[10] Sajad Shirali-Shahreza, Hamid Beigy, and Mohammad Hassan Shirali-Shahreza. Knapsack model for pixel based skin detection. In *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on*, pages 581–584. IEEE, 2008.

[11] Hao-kui Tang and Zhi-quan Feng. Hand's skin detection based on ellipse clustering. In *Computer Science and Computational Technology, 2008. ISCSCT'08. International Symposium on*, volume 2, pages 758–761. IEEE, 2008.

[12] Y. C. Lu, S. Y. Yang, and D. C. Cherng. Background subtraction based segmentation using object motion feedback. In *2011 First International Conference on Robot, Vision and Signal Processing*, pages 224–227, Nov 2011.

[13] Q. Zhang, F. Chen, and X. Liu. Hand gesture detection and segmentation based on difference background image with complex background. In *2008 International Conference on Embedded Software and Systems*, pages 338–343, July 2008.

[14] Jae-Ho Shin, Jong-Shill Lee, Se-Kee Kil, Dong-Fan Shen, Je-Goon Ryu, Eung-Hyuk Lee, Hong ki Min, and Seung-Hong Hong. Hand region extraction and gesture recognition using entropy analysis. 2006.

[15] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. 21:745–758, 08 2003.

[16] Nguyen Dang Binh, Enokida Shuichi, and Toshiaki Ejima. Real-time hand tracking and gesture recognition system. In *Proceedings of International Conference on Graphics, Vision and Image Processing*, pages 362–368, 2005.

[17] S. C. Agrawal, A. S. Jalal, and C. Bhatnagar. Recognition of indian sign language using feature fusion. In *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, pages 1–5, Dec 2012.

[18] Mahmoud M. Zaki and Samir I. Shaheen. Sign language recognition using a combination of new vision based features. *Pattern Recognition Letters*, 32(4):572 – 577, 2011.

[19] G. R. S. Murthy and R. S. Jadon. Hand gesture recognition using neural networks. In *2010 IEEE 2nd International Advance Computing Conference (IACC)*, pages 134–138, February 2010.

[20] V. Bobić, P. Tadić, and G. Kvaščev. Hand gesture recognition using neural network based techniques. In *2016 13th Symposium on Neural Networks and Applications (NEUREL)*, pages 1–4, November 2016.

[21] Oyebade K. Oyedotun and Adnan Khashman. Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications*, April 2016.

[22] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347, November 2011.

[23] Ao Tang, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li. A Real-Time Hand Posture Recognition System Using Deep Neural Networks. *ACM Transactions on Intelligent Systems and Technology*, 6(2):1–23, March 2015.

[24] Gjorgji Strezoski, Dario Stojanovski, Ivica Dimitrovski, and Gjorgji Madjarov. Hand Gesture Recognition using Deep Convolutional Neural Networks.

[25] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. Sign Language Recognition Using Convolutional Neural Networks. In Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, editors, *Computer Vision - ECCV 2014 Workshops*, volume 8925, pages 572–578. Springer International Publishing, Cham, 2015.

[26] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *International Symposium on Computer Vision*, pages 265–270, 1995.

[27] K. Grobel and M. Assan. Isolated sign language recognition using hidden Markov models. In *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 162–167 vol.1, October 1997.

[28] Rung-Huei Liang and Ming Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 558–567, April 1998.

[29] Nianjun Liu, B. C. Lovell, P. J. Kootsookos, and R. I. A. Davis. Model structure selection training algorithms for an HMM gesture recognition system. In *Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 100–105, October 2004.

[30] Jörg Zieren and Karl-Friedrich Kraiss. Robust Person-Independent Visual Sign Language Recognition. In *Pattern Recognition and Image Analysis*, volume 3522, pages 520–528. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[31] Son Lam Phung, Abdesselam Bouzerdoum, and Douglas Chai. A novel skin color model in ycbcr color space and its application to human face detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–289. IEEE, 2002.

[32] Bernd Menser and Mathias Wien. Segmentation and tracking of facial regions in color image sequences. In *Visual Communications and Image Processing 2000*, pages 731–740. International Society for Optics and Photonics, 2000.

[33] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K Jain. Face detection in color images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):696–706, 2002.

[34] Jörgen Ahlberg. A system for face localization and facial feature extraction. 1999.

[35] Douglas Chai and Abdesselam Bouzerdoum. A bayesian approach to skin color classification in ycbcr color space. In *TENCON 2000. Proceedings*, volume 2, pages 421–424. IEEE, 2000.

[36] OpenCV 2.4.8.0 Documentation miscellaneous image transformations. `http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html?highlight=ycrcb`.

[37] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE*, pages 257–286, 1989.

[38] Przemyslaw Dymarski, editor. *Hidden Markov Models, Theory And Applications*. InTech, 2011.

[39] A. Michael Nielsen. Neural networks and deep learning. Disponível em `http://neuralnetworksanddeeplearning.com/index.html`, 2015. Acessado em 05/06/2017.

[40] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, and others. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[41] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[43] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*, chapter 8. Thomson, 3 edition, 2008.

[44] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[45] Jan Flusser and Tomas Suk. Pattern recognition by affine moment invariants. *Pattern recognition*, 26(1):167–174, 1993.

[46] L Gupta and Mandyam D Srinath. Contour sequence moments for the classification of closed planar shapes. *Pattern Recognition*, 20(3):267–272, 1987.

[47] Malayappan Shridhar and A Badreldin. High accuracy character recognition algorithm using fourier and topological descriptors. *Pattern Recognition*, 17(5):515–524, 1984.

[48] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, chapter 11. Prentice Hall, 2 edition, 2008.

[49] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *CVGIP*, 30(1):32–46, 1985.

[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), November 1998.

[51] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, September 2014.

[52] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[53] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, April 1967.

[54] G. D. Forney. The viterbi algorithm. *Proc. of the IEEE*, 61:268 – 278, March 1973.

[55] S.J. Young, N.H. Russell, and J.H.S Thornton. Token passing: a simple conceptual model for connected speech recognition systems. Technical report, 1989.

[56] Leonard E Baum and GR Sell. Growth functions for transformations on manifolds. *Pacific J. Math*, 27(2):211–227, 1968.

[57] James Baker. The dragon system–an overview. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1):24–29, 1975.

[58] L Liporace. Maximum likelihood estimation for multivariate observations of markov sources. *Information Theory, IEEE Transactions on*, 28(5):729–734, 1982.

[59] B-H Juang. Maximum-likelihood estimation for mixture multivariate stochastic observations of markov chains. *AT&T technical journal*, 64(6):1235–1249, 1985.

[60] Bing-Hwang Juang, Stephen E Levinson, and M Mohan Sondhi. Maximum likelihood estimation for multivariate mixture observations of markov chains (corresp.). *Information Theory, IEEE Transactions on*, 32(2):307–309, 1986.

[61] Stephen E Levinson, Lawrence R Rabiner, and Man Mohan Sondhi. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *Bell System Technical Journal, The*, 62(4):1035–1074, 1983.

[62] Lawrence R Rabiner, B-H Juang, SE Levinson, and MM Sondhi. Some properties of continuous hidden markov model representations. *AT&T technical journal*, 64(6):1251–1270, 1985.

[63] S. Young. *The HTK Book*. Entropic Cambridge Research Laboratory, 1997.

APÊNDICE A

HMMs

Este apêndice trata sobre os três problemas dos HMMs, apresentados anteriormente na seção 3.2. A estimação, decodificação e treinamento são apresentados na seção A.1. A seção A.2 apresenta alguns detalhes sobre a implementação dos HMMs. Na seção A.3 é apresentada uma introdução sobre o HTK, que foi utilizado neste trabalho para a implementação dos HMMs.

A.1 Estimação, Decodificação e Treinamento

A.1.1 Problema 1: Estimação

Deseja-se calcular $p\{\mathbf{X}|\lambda\}$, a probabilidade de que uma sequência de observações $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_T}\}$ tenha sido gerada dado o modelo λ. Seja $p\{\mathbf{X}, \mathbf{y}|\lambda\}$ a probabilidade de que o modelo λ tenha gerado a sequência de estados $\mathbf{y} = \{y_1, \ldots, y_T\}$ e a sequência de observações $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_T}\}$, então $p\{\mathbf{X}|\lambda\}$ pode ser calculada como o somatório de $p\{\mathbf{X}, \mathbf{y}|\lambda\}$ para todas as sequências de estados $\mathbf{y}$ possíveis, ou seja:

$$p\{\mathbf{X}|\lambda\} = \sum_{\text{para todo } \mathbf{y}} p\{\mathbf{X}, \mathbf{y}|\lambda\} \tag{A.1}$$

E

$$p\{\mathbf{X}, \mathbf{y}|\lambda\} = p\{\mathbf{X}|\mathbf{y}, \lambda\}p\{\mathbf{y}|\lambda\} \tag{A.2}$$

Onde $p\{\mathbf{X}|\mathbf{y}, \lambda\}$ é a probabilidade da sequência de estados $\mathbf{X}$ ter sido gerada pelo modelo λ dado que a sequência de estados $\mathbf{y}$ foi observada, então:

$$\begin{aligned} p\{\mathbf{X}|\mathbf{y}, \lambda\} &= p\{\mathbf{x_1}|y_1, \lambda\}p\{\mathbf{x_2}|y_2, \lambda\} \dots p\{\mathbf{x_T}|y_T, \lambda\} \\ &= b_{y_1}(\mathbf{x_1})b_{y_2}(\mathbf{x_2}) \dots b_{y_T}(\mathbf{x_T}) \\ &= \prod_{t=1}^{T} b_{y_t}(\mathbf{x_t}) \end{aligned} \tag{A.3}$$

E $p\{\mathbf{y}|\lambda\}$ é a probabilidade da sequência de estados $\mathbf{y}$ ter sido gerada pelo modelo λ, ou seja:

$$\begin{aligned} p\{\mathbf{y}|\lambda\} &= p\{y_1|\lambda\}p\{y_2|y_1, \lambda\} \dots p\{y_T|y_{T-1}, \lambda\} \\ &= \pi_{y_1} a_{y_1 y_2} \dots a_{y_{T-1} y_T} \end{aligned} \tag{A.4}$$

Sendo assim:

$$\begin{aligned} p\{\mathbf{X}, \mathbf{y}|\lambda\} &= p\{\mathbf{X}|\mathbf{y}, \lambda\}p\{\mathbf{y}|\lambda\} \\ &= \prod_{t=1}^{T} b_{y_t}(\mathbf{x_t})\pi_{y_1} a_{y_1 y_2} \dots a_{y_{T-1} y_T} \end{aligned} \tag{A.5}$$

E

$$\begin{aligned} p\{\mathbf{X}|\lambda\} &= \sum_{\text{para todo } \mathbf{y}} p\{\mathbf{X}, \mathbf{y}|\lambda\} \\ &= \sum_{\text{para todo } \mathbf{y}} \left(\prod_{t=1}^{T} b_{y_t}(\mathbf{x_t})\pi_{y_1} a_{y_1 y_2} \dots a_{y_{T-1} y_T}\right) \end{aligned} \tag{A.6}$$

O cálculo de $p\{\mathbf{X}|\lambda\}$ a partir da equação A.6 é possível, porém não é computacionalmente viável. Felizmente existem outros algoritmos capazes de calcular $p\{\mathbf{X}|\lambda\}$ de uma maneira muito mais eficaz: os algoritmos *forward* e *backward*.

O algoritmo *forward*

Este algoritmo consiste em utilizar uma variável auxiliar $\alpha_t(i)$, chamada de variável *forward*, para calcular recursivamente $p\{\mathbf{X}|\lambda\}$. Sendo:

$$\alpha_t(i) = p\{\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_t}, y_t = e_i|\lambda\} \tag{A.7}$$

Onde $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}$ é uma sequência parcial de observações (até o tempo t) e y_t é o estado atual no tempo t.

Como $\alpha_t(i)$ é a probabilidade conjunta de que os símbolos $\mathbf{X} = \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}$ foram observados e que o estado no instante t seja e_i, o produto $\alpha_t(i)a_{ij}$ é a probabilidade de que os símbolos $\mathbf{X}$ foram observados e que o estado e_j foi alcançado no instante $t+1$ a partir do estado e_i no instante t. Somando-se esse produto sobre todos os N possíveis estados e_i no instante t resulta na probabilidade de alcançar o estado e_j no instante $t+1$ a partir de todos os estados anteriores. $\alpha_{t+1}(j)$ é facilmente obtido a partir da multiplicação da probabilidade obtida no somatório pela probabilidade do estado e_j gerar o símbolo $\mathbf{x_{t+1}}$. Então:

$$p\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}, \mathbf{x_{t+1}}, y_{t+1} = e_j|\lambda\} = p\{\mathbf{x_{t+1}}|y_{t+1} = e_j, \lambda\} \sum_{i=1}^{N} p\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}, y_t = e_i|\lambda\}a_{ij} \tag{A.8}$$

Se $\alpha_t(i) = p\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}, y_t = e_i|\lambda\}$ e $\alpha_{t+1}(j) = p\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}, \mathbf{x_{t+1}}, y_{t+1} = e_j|\lambda\}$, então substituindo-se na equação A.8:

$$\begin{aligned} \alpha_{t+1}(j) &= p\{\mathbf{x_{t+1}}|y_{t+1} = e_j, \lambda\} \sum_{i=1}^{N} \alpha_t(i)a_{ij} \\ \alpha_{t+1}(j) &= b_j(\mathbf{x_{t+1}}) \sum_{i=1}^{N} \alpha_t(i)a_{ij}, \quad 1 \le j \le N, \quad 1 \le t \le T-1 \end{aligned} \tag{A.9}$$

$\alpha_1(i)$ é dado por:

$$\begin{aligned} \alpha_1(i) &= p\{\mathbf{x_1}, y_1 = e_i|\lambda\} \\ &= p\{y_1 = e_i|\lambda\}p\{\mathbf{x_1}|y_1 = e_i, \lambda\} \\ &= \pi_i b_i(\mathbf{x_1}), \quad 1 \le i \le N \end{aligned} \tag{A.10}$$

O cálculo de $\alpha_{t+1}(j)$ é computado para todos os estados e_j, $1 \le j \le N$, para um dado instante t. Então, os cálculos são iterados para para $t = 1, 2, \ldots, T-1$.

Com $\alpha_T(i)$, $1 \le i \le N$, calculados, finalmente $p\{\mathbf{X}|\lambda\}$ pode ser calculada através de

$$p\{\mathbf{X}|\lambda\} = \sum_{i=1}^{N} \alpha_T(i) \tag{A.11}$$

O algoritmo *backward*

O algoritmo *backward* também pode ser utilizado para calcular $p\{\mathbf{X}|\lambda\}$, e assim como no algoritmo *forward*, definimos uma variável auxiliar $\beta_t(i)$, que representa a probabilidade da sequência parcial de observações do instante $t+1$ até o final, dado o estado $y_t = e_i$ no instante t e o modelo λ.

$$\beta_t(i) = p\{\mathbf{x_{t+1}}, \mathbf{x_{t+2}}, \ldots, \mathbf{x_T}|y_t = e_i, \lambda\} \tag{A.12}$$

Defini-se arbitrariamente $\beta_T(i) = 1$, e calcula-se $\beta_t(i)$ recursivamente:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{x_{t+1}}) \beta_{t+1}(j), \quad t = T-1, T-2, \ldots, 1, \quad 1 \leq i \leq N \tag{A.13}$$

E

$$p\{\mathbf{X}|\lambda\} = \sum_{i=1}^{N} \beta_1(i) \tag{A.14}$$

A.1.2 Problema 2: Decodificação

A decodificação consiste em encontrar a sequência de estados $\mathbf{y}$ ótima associada à um dado modelo λ e uma sequência de observações $\mathbf{X}$. Devido a dificuldade de definição do que é a sequência de estados ótima, existem várias soluções para este problema. Aqui serão apresentados dois critérios de otimização.

Determinar os estados y_t mais prováveis em cada instante t

Este critério determina os estados mais prováveis para cada instante t. Para implementar essa solução defini-se a variável

$$\gamma_t(i) = p(y_t = e_i|\mathbf{X}, \lambda) \tag{A.15}$$

A qual representa a probabilidade de se estar no estado e_i no instante t, dado uma sequência de observações X e um modelo λ. A equação A.15 pode ser expressa em termos das variáveis *forward-backward*

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{p(X|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{A.16}$$

Uma vez que $\alpha_t(i)$ representa a sequência parcial de observações $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}$ e o estado $y_t = e_i$, enquanto que $\beta_t(i)$ representa o restante da sequência de observações $\mathbf{x_{t+1}}, \mathbf{x_{t+2}}, \ldots, \mathbf{x_T}$ dado que $y_t = e_i$. $p(X|\lambda)$ é apenas um fator de normalização de modo que

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \tag{A.17}$$

Pode-se determinar o estado mais provável y_t no instante t através de

$$y_t = \underset{1 \leq i \leq N}{\operatorname{argmax}}[\gamma_t(i)], \quad 1 \leq t \leq T \tag{A.18}$$

Apesar da equação A.18 determinar os estados mais prováveis para cada instante t, podem existir alguns problemas com a sequência de estados resultante. Por exemplo, quando o HMM possui transições de estados com probabilidade zero ($a_{ij} = 0$ para determinado i e j), a sequência de estados ótima resultante pode nem ser uma sequência válida. Isso se deve ao fato de que a solução da equação A.18 determina o estado mais provável para cada instante t, sem levar em consideração a probabilidade de ocorrência das sequências de estados.

Determinar a sequência de estados mais provável - Algoritmo de Viterbi

Este critério é o mais amplamente usado e consiste em maximizar $p(\mathbf{y}|\mathbf{X}, \lambda)$, o que é equivalente a maximizar $p(\mathbf{y}, \mathbf{X}|\lambda)$. A sequência de estados mais provável é determinada através do Algoritmo de Viterbi [53], [54]: definindo

$$\delta_t(i) = \max_{y_1, y_2, \ldots, y_t - 1} p(y_1, y_2, \ldots, y_t = e_i, \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}|\lambda) \tag{A.19}$$

Ou seja, $\delta_t(i)$ é a probabilidade da sequência de estados mais provável, no instante t, dadas as primeiras t observações e que termina no estado $y_t = e_i$. Por recursão tem-se

$$\delta_{t+1}(j) = \max_i[\delta_t(i)a_{ij}]b_j(\mathbf{x_{t+1}}) \tag{A.20}$$

Para recuperar a sequência de estados, é preciso manter um registro dos argumentos que maximizam a equação A.20, para cada t e j. Isto é

feito através do vetor $\psi_t(j)$. O procedimento completo para determinar a sequência de estados é feito em quatro passos:

(i) Inicialização:

$$\delta_1(i) = \pi_i b_i(\mathbf{x_1}), \quad 1 \leq i \leq N \tag{A.21}$$

$$\psi_1(i) = 0 \tag{A.22}$$

(ii) Recursão:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{x_t}), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{A.23}$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{A.24}$$

(iii) Finalização:

$$y_T = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_T(i)] \tag{A.25}$$

(iv) Determinação da sequência de estados:

$$y_t = \psi_{t+1}(y_{t+1}), \quad t = T-1, T-2, \ldots, 1 \tag{A.26}$$

Token Passing Model [55]

Token Passing Model é uma variação do algoritmo de Viterbi que pode ser facilmente aplicado ao caso do reconhecimento de redes de HMMs, como as da figura 3.4 por exemplo. Partindo de que cada estado i de um HMM no instante t carrega um símbolo móvel, que chamaremos de token. Cada token contém, dentre outras informações, a probabilidade parcial $\delta_t(i)$. Sendo assim, cada token representa uma correspondência parcial entre a sequência de observações $\mathbf{x_1}$ a $\mathbf{x_t}$ e o modelo sujeito a restrição de que o modelo esteja no estado i no instante t. O algoritmo *token passing* é executado a cada instante de tempo t e seus passos principais são:

(i) Propaga-se uma cópia de cada token no estado i para todos os estados j conectados, multiplicando-se $\delta_{t-1}(i)$ por $a_{ij} b_j(\mathbf{x_t})$.

(ii) Em cada estado, exclui-se todos os tokens com exceção do que possuir a maior probabilidade.

Nas redes de HMMs, quando um token é propagado do estado de saída de um HMM para o estado de entrada de outro, esse token passa a carregar a identidade do modelo que o transmitiu, bem como o instante de tempo t em que ocorreu essa transição. Uma vez que toda a sequência de observações foi processada, o token que carregar a maior probabilidade $\delta_T(i)$ corresponde a sequência de estados mais provável correspondente a sequência de observações. Como esse token também carrega informações dos HMMs pelos quais ele passou, pode-se obter a sequência de HMMs mais provável para a sequência de símbolos, bem como os instantes em que ocorreram as transições de um HMM para outro.

A.1.3 Problema 3: Treinamento

Para resolver esse problema deve-se ajustar os parâmetros λ do modelo de modo a maximizar $p(\mathbf{X}|\lambda)$ para um dado modelo. Não é conhecido nenhum método analítico para resolver esse problema, no entanto, podemos escolher λ de modo que $p(\mathbf{X}|\lambda)$ seja localmente maximizado através de um procedimento iterativo como a re-estimação de Baum-Welch. Porém para se utilizar a re-estimação de Baum-Welch, deve-se primeiro fazer uma estimativa inicial dos parâmetros do HMM. Não há nenhum método correto para isso, porém um dos métodos utilizados para HMMs do tipo esquerda-diretia é descrito a seguir.

Supondo que um HMM possua somente um estado, então a distribuição de probabilidade de emissão de símbolo desse estado poderia ser facilmente estimada como

$$b(\mathbf{s_k}) = \frac{\text{número de emissões do símbolo } \mathbf{s_k}}{\text{número total de símbolos emitidos}} = \frac{1}{T}\sum_{t=1}^{T} 1 \text{ se } \mathbf{x_t} = \mathbf{s_k} \tag{A.27}$$

Porém na prática existem múltiplos estados, e as observações não podem ser atribuídas diretamente aos estados, uma vez que a sequência de estados é desconhecida. Então, uma forma de fazer a estimação inicial dos parâmetros é dividir o sinal uniformemente pelo número de estados do HMM, e utilizar cada parte na equação A.27 para fazer a estimação dos parâmetros de cada estado, como mostrado na figura A.1. Após essa primeira estimativa é utilizado o algoritmo de Viterbi para encontrar a sequência de estados mais provável para as observações, que

são atribuídas novamente aos estados de acordo com a sequência obtida. Então a equação A.27 pode ser utilizada novamente. Esse processo é repetido até que as estimativas não mudem. Importante notar que essa estimação inicial apenas estima os valores da probabilidade de emissão de símbolos dos estados, enquanto que a matriz de transição **A** deve ser inicializada manualmente com valores aceitáveis para a aplicação, normalmente distribuídos igualmente entre as transições permitidas.

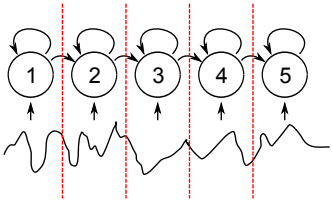


Figura A.1: Inicialização de um HMM com estrutura esquerda-direita. O sinal é dividido pelo número de estados do HMM, e cada parte é utilizada para estimar os parâmetros de um estado.

Pode-se ver que o procedimento de inicialização descrito pode não ser indicado para outras estruturas de HMMs, como mostra a figura A.2.

Feita a estimativa inicial dos parâmetros do HMM pode-se fazer a re-estimação de Baum-Welch, que em vez de atribuir as observações a estados específicos, atribui cada observação a todos os estados na proporção da probabilidade do modelo estar naquele estado no instante em que a observação foi emitida. $\gamma_t(i)$, como já foi definido anteriormente, é a probabilidade de se estar no estado e_i no instante t, dada a sequência de observações e o modelo. Então:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{valor esperado de transições a partir de } e_i \qquad \text{(A.28)}$$

Baseando-se nas equações A.27 e A.28 pode-se re-estimar os parâmetros $\boldsymbol{\pi}$ e **b**:

$$\bar{\pi}_i = \text{frequência esperada do estado } e_i \text{ no instante } (t = 1) = \gamma_1(i) \qquad \text{(A.29)}$$

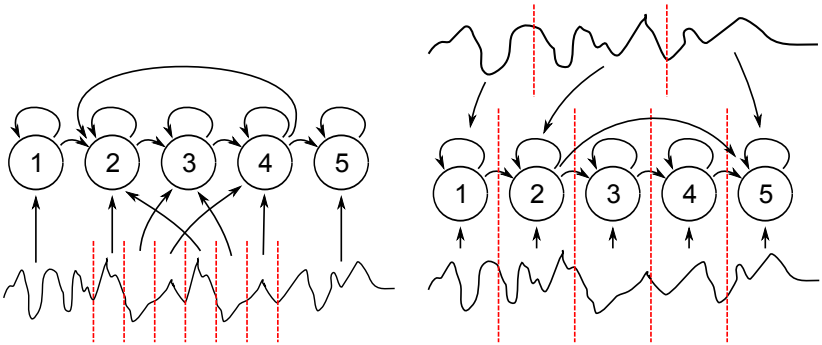


Figura A.2: Inicialização de um HMMs com estrutura diferente da esquerda-direita. À esquerda: um HMM com estrutura esquerda-direita com transição de retorno. A transição de retorno indica uma possível repetição de um trecho do sinal, e essas repetições devem ser utilizadas no treinamento de seu respectivo estado; Á direita: um HMM com estrutura esquerda-direita com transição de escape. É mostrado um sinal abaixo e outro acima do HMM. Pode-se notar que o sinal acima não pode ser dividido pelo número de estados do HMM, pois não deve ser utilizado para treinar os estados 3 e 4 do HMM.

$$\bar{b}_j(\mathbf{s_k}) = \frac{\text{valor esperado de ocorrência do estado } e_j \text{ com emissão do símbolo } \mathbf{s_k}}{\text{valor esperado de ocorrência do estado } e_j}$$
$$= \frac{\sum_{\substack{t=1 \\ \text{somente se } \mathbf{x_t}=\mathbf{s_k}}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{A.30}$$

Agora resta apenas obter uma fórmula para estimar a matriz de transição $\mathbf{A}$. Para isso definimos $\xi_t(i,j)$ como a probabilidade de se estar no estado e_i no instante t, e no estado e_j no instante $t+1$, dado o modelo e a sequência de treinamento.

$$\xi_t(i,j) = p(y_t = e_i, y_{t+1} = e_j | \mathbf{X}, \lambda) \tag{A.31}$$

Também pode-se expressar $\xi_t(i,j)$ em função das variáveis *forward* e

backward, da seguinte forma

$$\begin{aligned}\xi_t(i,j) &= \frac{\alpha_t(i)a_{ij}b_j(\mathbf{x_{t+1}})\beta_{t+1}(j)}{p(\mathbf{X}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(\mathbf{x_{t+1}})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(\mathbf{x_{t+1}})\beta_{t+1}(j)}\end{aligned} \tag{A.32}$$

Onde a divisão por $p(\mathbf{X}|\lambda)$ nos dá a medida de probabilidade. Ainda podemos relacionar $\gamma_t(i)$ com $\xi_t(i,j)$ através de

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j) \tag{A.33}$$

Desse modo pode-se interpretar

$$\sum_{t=1}^{T-1}\xi_t(i,j) = \text{valor esperado de transições de } e_i \text{ para } e_j \tag{A.34}$$

Utilizando as equações A.28 e A.34 é possível re-estimar $\mathbf{A}$ como:

$$\begin{aligned}\bar{a}_{ij} &= \frac{\text{valor esperado de transições do estado } e_i \text{ para o estado } e_j}{\text{valor esperado de transições a partir do estado } e_i} \\ &= \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}\end{aligned} \tag{A.35}$$

Dado um modelo $\lambda = (\mathbf{A}, \mathbf{b}, \boldsymbol{\pi})$, se usarmos as equações A.29, A.30 e A.35, podemos re-estimar um novo modelo $\bar{\lambda} = (\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\boldsymbol{\pi}})$. Foi provado ([56], [57]) que, ou o modelo inicial λ define um ponto crítico no qual $\bar{\lambda} = \lambda$ ou o modelo $\bar{\lambda}$ é mais provável que o modelo λ no sentido que $p(\mathbf{X}|\bar{\lambda}) > p(\mathbf{X}|\lambda)$. Em outras palavras, a sequência de treinamento $\mathbf{X}$ é mais provável de ter sido gerada pelo novo modelo $\bar{\lambda}$ do que pelo antigo modelo λ.

Se continuarmos, iterativamente, a utilizar $\bar{\lambda}$ no lugar de λ e repetir a re-estimação, podemos obter um novo modelo com uma probabilidade de gerar a sequência $\mathbf{X}$ cada vez maior. Esse processo deve ser repetido enquanto $p(\mathbf{X}|\lambda)$ para o modelo obtido na última iteração for maior que o da iteração anterior.

Observações Pertencentes a um Espaço Contínuo

Até agora considerou-se apenas o caso em que as observações são símbolos discretos pertencentes a um espaço finito, no entanto, para algumas aplicações (como este trabalho) as observações são símbolos pertencentes a um espaço contínuo. Geralmente modela-se $b_j(\mathbf{x})$ por uma soma ponderada de distribuições Gaussianas como na equação 3.4, que é reescrita aqui:

$$b_j(\mathbf{x}) = \sum_{m=1}^{\kappa} c_{jm} \mathcal{N}(\boldsymbol{\mu}_{\mathbf{jm}}, \boldsymbol{\Sigma}_{\mathbf{jm}}, \mathbf{x}) \tag{A.36}$$

As equações utilizadas para re-estimar a_{ij} e π_i são as mesmas utilizadas no caso de observações discretas (equações A.29 e A.35). A re-estimação de $b_j(\mathbf{x})$ é feita através da re-estimação dos parâmetros c_{jm}, $\boldsymbol{\mu}_{\mathbf{jm}}$ e $\boldsymbol{\Sigma}_{\mathbf{jm}}$ [58], [59], [60].

$$\bar{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)}{\sum_{t=1}^{T} \sum_{m=1}^{\kappa} \gamma_t(j,m)} \tag{A.37}$$

$$\bar{\boldsymbol{\mu}}_{\mathbf{jm}} = \frac{\sum_{t=1}^{T} \gamma_t(j,m) \mathbf{x_t}}{\sum_{t=1}^{T} \gamma_t(j,m)} \tag{A.38}$$

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{jm}} = \frac{\sum_{t=1}^{T} \gamma_t(j,m) (\mathbf{x_t} - \boldsymbol{\mu}_{\mathbf{jm}})(\mathbf{x_t} - \boldsymbol{\mu}_{\mathbf{jm}})'}{\sum_{t=1}^{T} \gamma_t(j,m)} \tag{A.39}$$

Onde $'$ denota o vetor transposto e $\gamma_t(j,m)$ é a probabilidade de se estar no estado j no instante t com a m-ésima componente da soma de Gaussianas sendo responsável pela geração de $\mathbf{x_t}$.

$$\gamma_t(j,m) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[\frac{c_{jm} \mathcal{N}(\boldsymbol{\mu}_{\mathbf{jm}}, \boldsymbol{\Sigma}_{\mathbf{jm}}, \mathbf{x_t})}{\sum_{m=1}^{\kappa} c_{jm} \mathcal{N}(\boldsymbol{\mu}_{\mathbf{jm}}, \boldsymbol{\Sigma}_{\mathbf{jm}}, \mathbf{x_t})} \right] \tag{A.40}$$

A.2 Detalhes de Implementação

Aqui são listados alguns aspectos importantes em relação a implementação dos HMMs.

- Uma ferramenta muito poderosa para implementação de redes de HMMs contínuos é o HTK (apêndice A.3).

- Problemas numéricos: considerando a variável *forward* $\alpha_t(i)$ definida na equação A.7, podemos notar que $\alpha_t(i)$ é calculada a partir de um grande número de produtos dos termos a_{ij} e b_j. Como cada um desses termos é menor que 1, para um valor de t elevado, o valor de $\alpha_t(i)$ irá exceder a faixa de precisão da máquina utilizada para o cálculo. Para contornar esse problema utiliza-se aritmética logarítmica [61].

- Múltiplas sequências de observações: na seção 3.2.3 foram apresentandos os modelos esquerda-direita, porém estes não podem ser treinados a partir de uma única sequência de observações. Isso se deve ao fato da natureza transitória dos estados do modelo, que permite que apenas uma pequena quantidade de observações sejam emitidas por cada estado antes que uma transição para o próximo estado ocorra. Então, para se ter uma quantidade de dados suficiente para treinar o modelo, é necessário utilizar múltiplas sequências de observações, como é mostrado em [61].

- Estimativa inicial dos parâmetros do HMM: na teoria, as equações de re-estimação devem resultar em valores dos parâmetros do HMM que correspondam a um máximo local da função de verosimilhança. Então a questão chave é como escolher um modelo inicial de modo que a re-estimação resulte em um máximo local que coincida com o máximo global da função de verosimilhança. Não existe uma resposta direta para essa questão, porém, a experiência dos trabalhos já realizados mostrou que tanto valores aleatórios ou uniformes para a estimativa inicial de $\boldsymbol{\pi}$ e $\mathbf{A}$ são adequados para que a re-estimação gere valores úteis, na maioria dos casos. No entanto, os parâmetros de $\mathbf{b}$ necessitam de uma boa estimativa inicial [62].

- Escolha do modelo: para implementação de um HMM, deve ser escolhido o tipo do modelo (totalmente conectado, esquerda-direita, ou alguma outra variação), o tamanho do modelo (número de estados), e os símbolos observados (pertencerão a um espaço discreto ou contínuo, escolha dos parâmetros que serão observados). Não existe nenhuma forma simples, ou teoricamente correta, de se fazer essas escolhas, e elas devem ser feitas dependendo do sinal

que será modelado. Esses parâmetros serão escolhidos e justificados ao longo do trabalho.

A.3 Hidden Markov Model ToolKit (HTK)

O HTK é um *tool kit* para a implementação de HMMs. Ele foi desenvolvido, primordialmente, para a criação de ferramentas de reconhecimento da fala baseadas em HMMs. No entanto, o HTK pode ser utilizado para outros propósitos, e, neste caso, será utilizado para fazer o reconhecimento de gestos em vídeos.

O HTK possui um conjunto de ferramentas, como as ferramentas de preparação de dados, de treinamento, de reconhecimento e de análise, sendo que as principais ferramentas são as de treinamento, utilizadas inicialmente para estimar os parâmetros de um conjunto de HMMs e as ferramentas de reconhecimento, utilizadas para fazer a transcrição dos vídeos.

Para o correto funcionamento das ferramentas utilizadas são necessários alguns arquivos como os arquivos de amostras, de estrutura do HMM, a rede de comunicação e o dicionário, que são descritos em seguida. Além destes, também são necessários arquivos de transcrições dos vídeos que serão utilizados para o treinamento, os quais devem possuir a transcrição de cada HMM no vídeo, bem como seu tempo de início e fim. Dentre as principais ferramentas que serão utilizadas neste trabalho podemos citar a ferramenta HInit, que é responsável por fazer a inicialização dos modelos, e a ferramenta HRest, a qual faz a re-estimação de *Baum-Welch*. Para fazer o reconhecimento será utilizada a ferramenta HVite, a qual utiliza um algoritmo chamado *Token Passing Model* [55] como alternativa ao algoritmo de Viterbi. O *Token Passing Model* possui a vantagem de se estender facilmete ao caso em que os vídeos contém uma sequência contínua de gestos (vários HMMs em sequência). Essas ferramentas não serão detalhadas neste trabalho, pois já estão em [63].

A.3.1 O Arquivo de Amostras

Como o HTK foi desenvolvido com foco em sistemas de fala, ele possui ferramentas para extração de descritores de arquivos de áudio. Os descritores extraídos podem ser salvos em arquivos que são utilizados

posteriormente nas ferramentas de treinamento e reconhecimento do HTK. Como os descritores utilizados neste trabalho são extraídos de vídeos e não de áudio, estes devem ser extraídos com alguma ferramenta externa ao HTK, e salvos em um arquivo com um formato reconhecido pelo HTK. Os arquivos de armazenamento de amostras reconhecidos pelo HTK consistem de uma sequência de amostras precedidas de um cabeçalho. Cada amostra é um vetor de valores (descritores) inteiros de dois *bytes* ou de valores de ponto flutuante de quatro bytes, armazenados no formato *Little-Endian*. O cabeçalho é composto dos seguintes itens:

(*i*) Número de Amostras (nA): um valor inteiro de quatro bytes correspondente ao número de amostras extraídas do vídeo. Cada amostra é um vetor de descritores.

(*ii*) Período de Amostragem (pA): um valor inteiro de quatro bytes expressando o período de amostragem, na escala de 100ns.

(*iii*) Tamanho da Amostra (tA): um valor inteiro de dois bytes correspondente ao tamanho de cada amostra, em bytes.

(*iv*) Tipo de descritores (tD): um valor inteiro de dois bytes indicando o tipo dos descritores utilizados. Como os descritores utilizados são extraídos de vídeos, não será utilizado nenhum tipo de descritor padrão do HTK, e por isso esse parâmetro deve possuir valor igual a 9, o que significa que as amostras são de um tipo definido pelo usuário.

Para ficar mais claro, supondo que deseja-se extrair uma amostra a cada quadro de um vídeo gravado a uma taxa de 20 fps. Cada amostra é composta de três descritores que serão armazenados como valores de ponto flutuante de 4 bytes. Nesse caso:

- Número de Amostras = quantidade de quadros do vídeo.
- Período de amostragem = $\frac{1}{20 * 100^{-9}}$.
- Tamanho da Amostra = 3 descritores $*$ 4 bytes por descritor = 12 bytes.
- Tipo de descritores = USER = 9.

A figura A.3 mostra a estrutura do arquivo de armazenamento das amostras, iniciando com o cabeçalho e seguido das amostras.

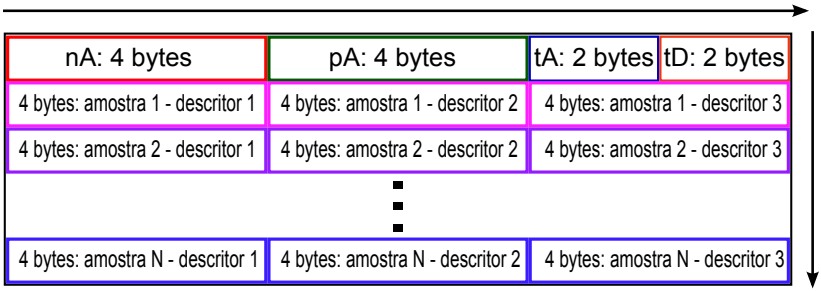


Figura A.3: Formato do arquivo de amostras.

A.3.2 O Arquivo de Estrutura do HMM

Cada HMM deve ser salvo em um arquivo, o qual define a sua estrutura e todos os seus parâmetros. Em [63] é mostrado o formato deste arquivo para diferentes tipos de estruturas, sendo que a Figura A.4 ilustra o formato do arquivo utilizado para os HMMs deste trabalho. Após o símbolo <VecSize> deve ser inserido o número de descritores utilizados Nd e o símbolo <USER>, indicando que os descritores utilizados não são os padrões do HTK. Após ~h deve ser inserido o nome do HMM, e o restante dos parâmetros devem estar inseridos entre os símbolos <BeginHMM> e <EndHMM>. Após <NumStates> deve ser inserido o número de estados Ne. Os estados 1 e Ne não são representados, pois não são estados emissores de símbolos, mas são estados utilizados pelo HTK para fins de implementação do algoritmo *Token Passing*, funcionando como nós de ligação entre os HMMs para formar uma rede de HMMs.

Os estados restantes são definidos através do símbolo <State> seguido do número do estado a ser definido. O estado 2 da Figura A.4 é modelado por uma mistura de duas pdfs gaussianas, então após o número do estado é inserido o símbolo <NumMixes> seguido do número de gaussinas. Cada pdf da mistura é definida através do símbolo <Mixture> seguido do número da pdf e do seu respectivo coeficiente de ponderação c_{jm}. Os símbolos <Mean> e <Variance> indicam os valores do vetor de média $\boldsymbol{\mu}$ e da diagonal principal da matriz de covariância

$\mathbf{\Sigma}$, com Nd valores. O terceiro estado é modelado por apenas uma pdf gaussiana, e por isso possui apenas os símbolos <Mean> e <InvCovar>, que é utilizado para especificar a matriz de covariância em sua forma inversa e triangular superior.

Após todos os estados terem sidos definidos, o símbolo <TransP> faz referência a matriz de transição, e são especificados os valores de a_{ij}.

A.3.3 A Rede de Comunicação e o Dicionário

O arquivo de rede de comunicação descreve a sequência de gestos que pode ser reconhecida e o dicionário descreve o HMM ou a sequência de HMMs que modelam cada gesto. O dicionário utilizado no HTK possui um formato simples, onde cada linha possui o seguinte formato:

GESTO [transcrição] prob P1 P2 P3 ...

Onde **GESTO** é o nome do gesto, **transcrição** é o símbolo de saída quando o gesto é reconhecido, **prob** é a probabilidade deste gesto (0.0 - 1.0). **P1, P2,** ... é a sequência de HMMs utilizada para modelar o gesto (o nome dos HMMs definidos nos arquivos de estrutura dos HMMs, como ilustrado na figura A.4). A **transcrição** e a **prob** são parâmetros opcionais. Se uma **transcrição** não for especificada o nome do gesto é utilizado como transcrição, e se os colchetes estiverem vazios a transcrição do gesto é suprimida, representando um gesto de silêncio, por exemplo. Se a **prob** não é especificada, é assumido seu valor padrão de 1.0.

Uma rede de comunicação entre gestos é definida utilizando o HTK Standard Lattice Format (SLF). Um arquivo SLF contém uma lista de nós representando os gestos e uma lista de de arcos representando as transições entre os gestos. As transições podem possuir probabilidades que podem ser utilizadas para indicar preferências em uma rede de comunicação. O arquivo de rede de comunicação entre gestos SLF não é difícil, porém exaustivo, de ser construído manualmente. O HTK disponibiliza uma ferramenta chamada HParse para a criação dos arquivos de rede comunicação de gestos SLF a partir de uma notação gramatical de alto nível baseada na extended Backus-Naur Form (EBNF), e detalhada em [63].

Como exemplo, imagine que deseja-se reconhecer gestos isolados, ou seja, em cada vídeo deve ocorrer somente um gesto e cada gesto é

representado por um único HMM. A estrutura do dicionário utilizado para este fim é mostrada na figura A.5, na qual o GESTO1 é modelado pelo HMM "gesto1" e, durante o reconhecimento, é transcrito pelo termo entre colchetes, etc.

Para gerar o arquivo de rede de comunicação entre gestos pode-se criar, como já foi dito, um arquivo de texto utilizando regras gramaticais baseadas na EBNF e utiliza-lo na ferramenta HParse. A figura A.6 mostra a estrutura do arquivo criado para gerar o arquivo de rede de comunicação de gestos. No arquivo é definido uma variável **VAR** que pode ser qualquer um dos gestos do dicionário. Abaixo, entre parenteses, está especificada a estrutura da rede, que nesse caso é apenas a ocorrência de um dos valores da variável **VAR**.

O dicionário, o arquivo de rede de comunicação entre gestos e o conjunto de HMMs são compilados pelo módulo do HTK, HNet, o qual gera uma rede de reconhecimento que consiste de um conjunto de nós conectados por arcos. Cada nó é um HMM ou um nó de conexão. Então, uma vez compilados, consiste de um conjunto de estados de HMM, emissores e não emissores de símbolos, conectados por transições. Essa compilação é feita *on-line*, como uma etapa de inicialização, no processo de reconhecimento. A figura A.7 mostra a rede de reconhecimento gerada a partir do modelo de dicionário da figura A.5 e da estrutura da rede ilustrada na figura A.6. Os estados em cinza são estados não emissores utilizados para conexão entre os HMMs da rede. Na rede de reconhecimento gerada, somente um gesto pode ocorrer, pois os HMMs dos gestos estão conectados em paralelo e todas as transições estão no mesmo sentido.

~o <VecSize> Nd <USER>
~h "nome_hmm"
<BeginHMM>
<NumStates> Ne
<State> 2 <NumMixes> 2
<Mixture> 1 0.4
<Mean> Nd
0.0 0.0 0.0 0.0 0.0 $\cdots$ 0.0 0.0 0.0 0.0
<Variance> Nd
1.0 1.0 1.0 1.0 1.0 $\cdots$ 1.0 1.0 1.0 1.0
<Mixture> 2 0.6
<Mean> Nd
0.0 0.0 0.0 0.0 0.0 $\cdots$ 0.0 0.0 0.0 0.0
<Variance> Nd
1.0 1.0 1.0 1.0 1.0 $\cdots$ 1.0 1.0 1.0 1.0
<State> 3
<Mean> Nd
0.0 0.0 0.0 0.0 0.0 $\cdots$ 0.0 0.0 0.0 0.0
<InvCovar> Nd
1.0 0.1 0.0 $\cdots$ 0.0
1.0 0.2 $\cdots$ 0.0
$\ddots$
0.0
<State> 4
$\vdots$
<State> $Ne-1$
<Mean> Nd
0.0 0.0 0.0 0.0 0.0 $\cdots$ 0.0 0.0 0.0 0.0
<Variance> Nd
1.0 1.0 1.0 1.0 1.0 $\cdots$ 1.0 1.0 1.0 1.0
<TransP> Ne
0.0 1.0 0.0 0.0 0.0 $\cdots$ 0.0 0.0 0.0 0.0 0.0
0.0 0.6 0.4 0.0 0.0 $\cdots$ 0.0 0.0 0.0 0.0 0.0
0.0 0.2 0.3 0.3 0.1 $\cdots$ 0.0 0.0 0.0 0.0 0.0
$\vdots$ $\ddots$ $\vdots$
0.0 0.0 0.0 0.6 0.4 $\cdots$ 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.6 $\cdots$ 0.0 0.0 0.0 0.0 0.0
<EndHMM>

Figura A.4: Estrutura do arquivo do HTK que armazena as informações sobre um HMM.

GESTO1 [transcrição1] gesto1
GESTO2 [transcrição2] gesto2
⋮
GESTON [transcriçãon] gesto$n$

Figura A.5: Arquivo de dicionário.

$VAR = GESTO1 | GESTO2 | ... | GESTON;

($VAR)

Figura A.6: Estrutura do reconhecedor.

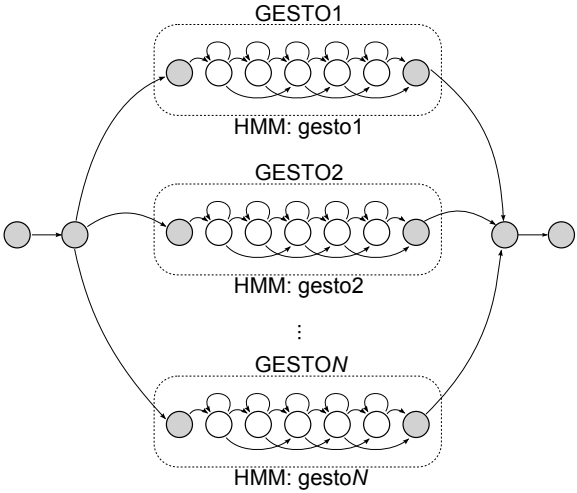


Figura A.7: Estrutura do reconhecedor para gestos isolados.

APÊNDICE B

ANNs

Na seção 3.3 foram apresentadas as ANNs. Este apêndice trata da aprendizagem dessas redes. O *gradient descent* é tratado na seção B.1. A seção B.2 apresenta o algoritmo *backpropagation*, utilizado para calcular os parâmetros necessários para o *gradient descent*. A seção B.3 apresenta a dedução das equações do algoritmo *backpropagation*. Na seção B.4 é feita uma análise das funções de custo em relação a aprendizagem da rede. Em B.5 são derivadas algumas funções de custo e funções de ativação, de forma a serem utilizadas no algoritmo *backpropagation*. Finalmente em B.6 é dado um método para inicialização dos pesos das ANNs.

B.1 Gradient Descent

Nesta seção primeiramente é analisado o algoritmo *gradient descent* em duas dimensões. Posteriormente o algoritmo é generalizado para N dimensões e aplicado a ANNs. Por fim o SGD é introduzido.

B.1.1 Gradient Descent em Duas Dimensões

Dada uma função $C(v1, v2)$, esse algoritmo tende a achar os valores $v1$ e $v2$ que minimizem C. A cada iteração ele calcula novos valores de $v1$ e $v2$ que façam C cada vez menor.

A variação de C pode ser descrita por B.1:

$$\Delta C = \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2 \tag{B.1}$$

Escrevendo B.1 em sua forma vetorial obtém-se B.2:

$$\Delta C = \nabla \mathbf{c} \cdot \mathbf{\Delta v} \tag{B.2}$$

Onde $\mathbf{\Delta v} = [\Delta v_1, \Delta v_2]^T$ e $\nabla \mathbf{c} = \left[\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right]^T$ = Gradiente de C.

Se fizermos com que ΔC seja negativo, então o novo C será menor que o antigo. Para que $\Delta C < 0$ faz-se $\Delta v = -\eta \nabla \mathbf{c}$, onde η é a taxa de aprendizagem, que deve ser um valor positivo e pequeno.

Então:

$$\Delta C = \nabla \mathbf{c} \cdot (-\eta \nabla \mathbf{c}) = -\eta \left(\frac{\partial C^2}{\partial v_1} + \frac{\partial C^2}{v_2}\right) = -\eta \|\nabla \mathbf{c}\|^2 \tag{B.3}$$

Como pode ser observado na equação B.3, ΔC é sempre negativo para $\mathbf{\Delta v} = -\eta \nabla \mathbf{c}$.

Logo, os novos valores dos parâmetros de $\mathbf{v}$ que fazem com que o erro seja menor são dados por B.4:

$$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{\Delta v} = \mathbf{v} - \eta \nabla \mathbf{c} \tag{B.4}$$

Onde a seta significa que $\mathbf{v}$ recebe o valor dele mesmo mais a sua variação.

Com esse algoritmo podemos encontrar novos valores de $\mathbf{v}$ que geram um erro C menor que o anterior. Podemos repetir o algoritmo para achar valores cada vez menores. Para calcular os novos parâmetros de $\mathbf{v}$ precisamos apenas calcular $\nabla \mathbf{c}$.

Este algoritmo nos permite encontrar um mínimo local ou, na melhor das hipóteses, o mínimo global de C.

B.1.2 Gradient Descent para N Dimensões

Podemos generalizar o algoritmo da seção B.1.1 para n dimensões apenas mudando o tamanho dos vetores:

$$\mathbf{\Delta v} = [\Delta v_1, \Delta v_2, \ldots, \Delta v_n]^T$$

$$\nabla \mathbf{c} = \left[\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}, \ldots, \frac{\partial C}{\partial v_n}\right]^T$$

B.1.3 Gradient Descent Aplicado a Redes Neurais

Apenas substitui-se os parâmetros genéricos $\mathbf{v}$ pelos pesos e biases das redes neurais. Dessa forma a equação B.4 se torna:

$$w_k \leftarrow w_k - \eta \frac{\partial C}{\partial w_k} \tag{B.5}$$

$$b_l \leftarrow b_l - \eta \frac{\partial C}{\partial b_l} \tag{B.6}$$

B.1.4 Stochastic Gradient Descent

Definindo a função de custo para apenas uma amostra x:

$$C_x = \frac{\|\mathbf{y} - \mathbf{a}\|^2}{2} \tag{B.7}$$

A função custo total é a média da função de cada amostra:

$$C = \frac{1}{n} \sum_x C_x \tag{B.8}$$

Dessa forma, para calcular $\nabla \mathbf{c}$ devemos calcular $\nabla \mathbf{c_x}$ para cada entrada x, e fazer a sua média:

$$\nabla \mathbf{c} = \frac{1}{n} \sum_x \nabla \mathbf{c_x} \tag{B.9}$$

Se a quantidade de entradas x for muito grande, isso pode fazer com que o treinamento leve um longo tempo. Para aumentar a velocidade do treinamento utiliza-se uma variação chamada de SGD, que consiste

em estimar o gradiente $\nabla \mathbf{c}$ através de um conjunto menor de amostras escolhidas aleatoriamente, chamado de *mini-batch*.

Primeiramente escolhe-se um *mini-batch* de m amostras e calcula-se $\nabla \mathbf{c_{x_j}}$, para cada amostra x_j. Se m for suficientemente grande espera-se que a média do gradiente para esse conjunto de amostras seja aproximadamente a média do gradiente para todas as amostras:

$$\sum_{j}^{m} \frac{\nabla \mathbf{c_{x_j}}}{m} \approx \sum_{x} \frac{\nabla \mathbf{c_x}}{n} = \nabla \mathbf{c} \tag{B.10}$$

Então as equação B.5 e B.6 tornam-se:

$$w_k \leftarrow w_k - \frac{\eta}{m} \sum_{j} \frac{\partial C_{x_j}}{\partial w_k} \tag{B.11}$$

$$b_l \leftarrow b_l - \frac{\eta}{m} \sum_{j} \frac{\partial C_{x_j}}{\partial b_l} \tag{B.12}$$

Após calculados os parâmetros escolhe-se um novo *mini-batch* e repete-se o processo. Quando as amostras terminarem é dito que completou-se uma *epoch* do treinamento. Depois de completar uma *epoch* inicia-se outra, repetindo-se todo o processo.

B.2 O Algoritmo Backpropagation

O algoritmo *backpropagation* é uma forma eficiente de se calcular as derivadas da equações B.5 e B.6.

Já definimos as grandezas z como sendo a soma ponderada das entradas de um neurônio adicionada do seu bias, e $\sigma(z)$ como a saída do neurônio (equação 3.13). Também definimos o vetor $\mathbf{a}$ como a saída da rede. Agora vamos definir o vetor $\mathbf{a^k}$ como sendo o vetor de saída da camada k, e $a_j^k = \sigma(z_j^k)$ como a saída do neurônio j da camada k, onde z_j^k é o z do neurônio j da camada k.

Para que esse algoritmo possa ser utilizado, a função de custo deve exibir duas propriedades:

(i) A função de custo pode ser escrita como uma média da função de custo para cada amostra de treinamento, ou seja, $C = \frac{1}{n} \sum_x C_x$.

(*ii*) A função de custo pode ser escrita como uma função das saídas da rede neural.

Como pode ser visto, o erro quadrático médio $C = \frac{1}{2n}\sum_x \|\mathbf{y} - \mathbf{a}^L\|^2$ satisfaz ambas as condições, onde L é a última camada da rede.

Se houver uma pequena variação nos parâmetros de um neurônio, haverá uma pequena variação em seu z, e essa variação se propagará até a saída da rede, gerando uma variação na função de custo. A taxa de variação da função de custo com relação a essa variação dos parâmetros de um neurônio chamaremos de erro devido ao neurônio j da camada k, dado por:

$$\delta_j^k = \frac{\partial C}{\partial z_j^k} \tag{B.13}$$

Através do algoritmo *backpropagation* pode-se calcular todos os δ_j^k e a partir deles, calcular $\frac{\partial C}{\partial w_{ji}^k}$ e $\frac{\partial C}{\partial b_j^k}$. O algoritmo nos permite calcular o erro devido aos neurônios da última camada e a partir desses calcular o erro devido as camadas anteriores. É definido pelas equações B.22, B.30, B.35 e B.40 (dedução em B.3), mostradas abaixo na sua forma elementar e vetorial:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \quad , \quad \delta^{\mathbf{L}} = \nabla \mathbf{c}^{\mathbf{L}} \odot \sigma'(\mathbf{z}^{\mathbf{L}}) \tag{B.14}$$

$$\delta_j^k = \sum_i \delta_i^{k+1} \cdot w_{ij}^{k+1} \cdot \sigma'(z_j^k) \quad , \quad \delta^{\mathbf{k}} = (\mathbf{W^{k+1}}^T \delta^{\mathbf{k+1}}) \odot \sigma'(\mathbf{z^k}) \tag{B.15}$$

$$\frac{\partial C}{\partial b_j^k} = \delta_j^k \quad , \quad \nabla \mathbf{c_b^k} = \delta^{\mathbf{k}} \tag{B.16}$$

$$\frac{\partial C}{\partial w_{ji}^k} = \delta_j^k \cdot a_i^{k-1} \quad , \quad \nabla \mathbf{C_w^k} = \delta^{\mathbf{k}} \mathbf{a^{k-1}}^T \tag{B.17}$$

Observar que o valor de δ_j^L depende da derivada da função de custo e da função de ativação escolhida. A seção B.5 mostra como calcular a derivada de algumas funções de custo.

O algoritmo *backpropagation* em conjunto com o SGD para a aprendizagem da rede pode ser implementado através dos seguintes passos:

(*i*) Para cada época:

(a) Escolher aleatoriamente um conjunto de m amostras de treinamento (mini-batch) dentre todas as amostras.

(b) Para cada exemplo de treinamento x fazer a saída da primeira camada da rede $\mathbf{a}^{x,1}$ igual à entrada da rede.

(c) Repetir os seguintes passos para cada amostra do mini-batch:

i. **Feedforward**: para cada $k = 1, 2, \ldots, L$, calcular $\mathbf{z}^{x,k} = \mathbf{W}^k\mathbf{a}^{x,k-1} + \mathbf{b}^k$ e $\mathbf{a}^{x,k} = \sigma(\mathbf{z}^{x,k})$.

ii. **Erro na saída**: calcular o vetor de erro da última camada $\delta^{x,L}$.

iii. **Propagar o erro para as camadas anteriores**: para cada $k = L, L-1, L-2, \ldots, 2$, calcular $\delta^{x,k}$.

iv. **Calcular o gradiente da função de custo para a amostra x**: $\frac{\partial C_x}{\partial w^k_{ji}} = \delta^{x,k}_j \cdot a^{x,k-1}_i$ e $\frac{\partial C_x}{\partial b^k_j} = \delta^{x,k}_j$.

(d) Atualizar os pesos e biases para todas as camadas de acordo com:

$$w^k_{ji} \leftarrow w^k_{ji} - \frac{\eta}{m}\sum_x \frac{\partial C_x}{\partial w^k_{ji}}$$

$$b^k_j \leftarrow b^k_j - \frac{\eta}{m}\sum_x \frac{\partial C_x}{\partial b^k_j}$$

(*ii*) Voltar para o passo 1 e repetir o processo até que a função de custo C retorne um valor aceitável.

B.3 Dedução das Equações do Algoritmo Backpropagation

Este algoritmo possuí quatro equações, uma para calcular o erro devido aos neurônios da última camada δ^L_j, outra para calcular o erro dos neurônios das outras camadas δ^k_j. E mais duas equações para se calcular $\frac{\partial C}{\partial w^k_{ji}}$ e $\frac{\partial C}{\partial b^k_j}$.

B.3.1 Erro Devido aos Neurônios da Última Camada

O erro devido ao neurônio j da última camada é dado por:

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} \tag{B.18}$$

Sabendo-se que $C(\mathbf{a}^L)$ e $\mathbf{a}^L(\mathbf{z}^L)$ podemos utilizar a regra da cadeia do cálculo para reescrever B.18 da seguinte forma:

$$\delta_j^L = \frac{\partial C}{\partial \mathbf{a}^L}\frac{\partial \mathbf{a}^L}{\partial z_j^L} \tag{B.19}$$

A variação do custo C é o somatório da variação devido a todos os seus parâmetros, então B.19 pode ser escrita como:

$$\delta_j^L = \sum_i \frac{\partial C}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_j^L} \tag{B.20}$$

Como a saída a_i^L depende apenas de sua entrada z_i^L, então $\frac{\partial a_i^L}{\partial z_j^L} \neq 0$ apenas para $i = j$,logo B.20 torna-se:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{B.21}$$

Como $a_j^k = \sigma(z_j^k)$, então podemos escrever $\frac{\partial a_j^L}{\partial z_j^L}$ como $\sigma'(z_j^L)$, gerando a primeira equação:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \tag{B.22}$$

Que em sua forma vetorial fica

$$\delta^{\mathbf{L}} = \nabla \mathbf{c}^{\mathbf{L}} \odot \sigma'(\mathbf{z}^{\mathbf{L}}) \tag{B.23}$$

Onde $\odot$ representa o produto termo a termo dos vetores.

B.3.2 Erro Devido aos Neurônios das Camadas Restantes

O erro devido ao neurônio j da camada $L-1$ é dado por:

$$\delta_j^{L-1} = \frac{\partial C}{\partial z_j^k} \tag{B.24}$$

Sabendo-se que $C(\mathbf{a}^L)$ e $\mathbf{a}^L(\mathbf{z}^L)$, então $C(\mathbf{z}^L)$. Como $\mathbf{z}^L(\mathbf{a}^{L-1})$ e $\mathbf{a}^{L-1}(\mathbf{z}^{L-1})$, então $\mathbf{z}^L(\mathbf{z}^{L-1})$. Da mesma foram que anteriormente, podemos utilizar a regra da cadeia para reescrever B.24 como:

$$\delta_j^{L-1} = \frac{\partial C}{\partial \mathbf{z}^L} \frac{\partial \mathbf{z}^L}{\partial z_j^{L-1}} \tag{B.25}$$

A variação do custo C é o somatório da varição devido a todos os seus parâmetros, então B.25 torna-se:

$$\delta_j^{L-1} = \sum_i \frac{\partial C}{\partial z_i^L} \frac{\partial z_i^L}{\partial z_j^{L-1}}$$

$$\delta_j^{L-1} = \sum_i \frac{\partial z_i^L}{\partial z_j^{L-1}} \delta_i^L \tag{B.26}$$

De acordo com a expressão B.26 consegue-se expressar o erro devido a penúltima camada em função do erro devido a última camada. Agora devemos calcular o valor de $\frac{\partial z_i^L}{\partial z_j^{L-1}}$. Sendo $z_i^L = \sum_v w_{iv}^L \cdot \sigma(z_v^{L-1}) + b_i^L$, então $\frac{\partial z_i^L}{\partial z_j^{L-1}} \neq 0$ apenas para $v = j$. Então:

$$\frac{\partial z_i^L}{\partial z_j^{L-1}} = w_{ij}^L \cdot \sigma'(z_j^{L-1}) \tag{B.27}$$

Substituindo-se B.27 em B.26 obtém-se:

$$\delta_j^{L-1} = \sum_i \delta_i^L \cdot w_{ij}^L \cdot \sigma'(z_j^{L-1}) \tag{B.28}$$

Usando a mesma lógica para calcular o erro devido a camada $L-2$,

obtém-se:

$$\delta_j^{L-2} = \sum_i \delta_i^{L-1} \cdot w_{ij}^{L-1} \cdot \sigma'(z_j^{L-2}) \tag{B.29}$$

As equações para as camadas inferiores seguem o mesmo padrão, então pode-se formular a seguinte equação genérica para o erro devido a camada k:

$$\delta_j^k = \sum_i \delta_i^{k+1} \cdot w_{ij}^{k+1} \cdot \sigma'(z_j^k) \tag{B.30}$$

Sendo i o número de neurônios na camada $k+1$. Colocando B.30 em sua forma vetorial

$$\delta^{\mathbf{k}} = (\mathbf{W^{k+1}}^T \delta^{\mathbf{k+1}}) \odot \sigma'(\mathbf{z^k}) \tag{B.31}$$

B.3.3 Cálculo de $\frac{\partial C}{\partial b_j^k}$

Como feito anteriormente, reescrevemos a equação utilizando a regra da cadeia:

$$\frac{\partial C}{\partial b_j^k} = \frac{\partial C}{\partial \mathbf{z}^k} \frac{\partial \mathbf{z}^k}{\partial b_j^k}$$

$$\frac{\partial C}{\partial b_j^k} = \sum_i \frac{\partial C}{\partial z_i^k} \frac{\partial z_i^k}{\partial b_j^k} \tag{B.32}$$

Resolvendo as derivadas parciais de B.32:

$$\frac{\partial z_i^k}{\partial b_j^k} = \begin{cases} 0 \text{ se } i \neq j \\ 1 \text{ se } i = j \end{cases} \tag{B.33}$$

$$\frac{\partial C}{\partial z_i^k} = \delta_i^k \tag{B.34}$$

Substituindo B.33 e B.34 em B.32 obtemos a terceira equação:

$$\frac{\partial C}{\partial b_j^k} = \delta_j^k \tag{B.35}$$

A equação B.35 nos diz que a taxa de variação da função de custo com relação ao bias de um neurônio é igual ao erro devido ao mesmo neurônio.

Na sua forma vetorial, B.35 fica

$$\nabla \mathbf{c_b^k} = \delta^{\mathbf{k}} \tag{B.36}$$

B.3.4 Cálculo de $\frac{\partial C}{\partial w_{ji}^k}$

Procede-se da mesma forma feita na etapa anterior.

$$\frac{\partial C}{\partial w_{ji}^k} = \frac{\partial C}{\partial \mathbf{z}^k} \frac{\partial \mathbf{z}^k}{\partial w_{ji}^k}$$

$$\frac{\partial C}{\partial w_{ji}^k} = \sum_v \frac{\partial C}{\partial z_v^k} \frac{\partial z_v^k}{\partial w_{ji}^k} \tag{B.37}$$

Resolvendo as derivadas parciais de B.37:

$$\frac{\partial C}{\partial z_v^k} = \delta_v^k \tag{B.38}$$

$$\frac{\partial z_v^k}{\partial w_{ji}^k} = \frac{\partial \sum_m w_{vm}^k \cdot a_m^{k-1} + b_v^k}{\partial w_{ji}^k} = \begin{cases} a_i^{k-1}, \text{ para } v = j \\ 0, \text{ para } v \neq j \end{cases} \tag{B.39}$$

Substituindo B.38 e B.39 em B.37 obtemos a quarta e última equação:

$$\frac{\partial C}{\partial w_{ji}^k} = \delta_j^k \cdot a_i^{k-1} \tag{B.40}$$

A equação B.40 diz que a taxa de variação da função de custo em relação ao peso que um neurônio j dá a sua entrada i é igual ao erro devido a este neurônio multiplicado pelo valor da sua entrada i.

Finalmente, colocando B.40 em sua forma matricial

$$\nabla \mathbf{C_w^k} = \delta^{\mathbf{k}} \mathbf{a^{k-1}}^T \tag{B.41}$$

B.4 Análise das Funções de Custo

Aqui serão analisadas as funções de custo MSE e *cross-entropy* quanto as suas consequências na aprendizagem da rede neural.

O Erro Quadrático Médio

O erro quadrático médio ou MSE foi definido em 3.17 para um conjunto de amostras. A equação B.42 define o MSE para uma única amostra x.

$$C_x = \frac{\|\mathbf{y} - \mathbf{a}^L\|^2}{2} = \frac{1}{2}\sum_i (y_i - a_i^L)^2 \tag{B.42}$$

Para simplificar a notação, consideraremos que o custo é sobre uma única amostra de x, e que a rede possui apenas um neurônio com uma entrada. Dessa forma podemos eliminar os índices e o somatório, e B.42 torna-se:

$$C = \frac{(y-a)^2}{2} \tag{B.43}$$

Lembrando que $a = \sigma(z)$, onde $z = w \cdot x + b$. Como foi visto em B.1.3, a aprendizagem da rede é dada pela variação do peso e do bias em uma taxa determinada por $\frac{\partial C}{\partial w}$ e $\frac{\partial C}{\partial b}$. Usando a regra da cadeia para diferenciar B.43:

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x \tag{B.44}$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) \tag{B.45}$$

Nota-se em B.44 e B.45 que a aprendizagem é proporcional a derivada da saída do neurônio $\sigma'(z)$. Se analisarmos a forma de $\sigma(z)$ quando o neurônio é um sigmoide (figura 3.7), para uma saída saturada perto de 1 ou 0 $\sigma'(z)$ possui um valor muito baixo, o que faz com que a aprendizagem ocorra de forma mais lenta nessa região. Uma forma de solucionar esse problema é escolher uma função de custo que não apresente este comportamento.

Cross-Entropy

Essa função de custo não apresenta o problema de redução da velocidade de aprendizagem quando a saída do neurônio está saturada em neurônios do tipos sigmoide, como no caso do MSE.

Para ilustrar o seu comportamento utilizaremos uma rede de apenas um neurônio com múltiplas entradas. Então a função de cross-entropy

é dada por:

$$C = \frac{-1}{n} \sum_x [yln(a) + (1-y)ln(1-a)] \tag{B.46}$$

Onde n é o numero de amostras de treinamento, e o somatório é sobre todas as amostras x.

A função de cross-entropy possui as características de uma função de custo, ou seja, nunca é negativa e se aproxima de zero quando a saída da rede é próxima da saída desejada.

Derivando B.46 em relação aos pesos e ao bias do neurônio obtém-se:

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j(\sigma(z) - y) \tag{B.47}$$

$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x (\sigma(z) - y) \tag{B.48}$$

As expressões B.47 e B.48 nos dizem que a taxa com que o neurônio aprende é controlada pelo erro entre a saída desejada e a saída obtida $\sigma(z) - y$, o que elimina a desaceleração da aprendizagem nas regiões de saturação do neurônio quando este é um sigmoide.

A função B.46 é para uma rede de apenas um neurônio. Generalizando, obtemos a função de cross-entropy para uma rede com múltiplas camadas e neurônios:

$$C = \frac{-1}{n} \sum_x \sum_j [y_j ln(a_j^L) + (1-y_j)ln(1-a_j^L)] \tag{B.49}$$

O somatório em j é sobre todos os neurônios da camada de saída.

B.5 Derivando as Funções de Custo

Para a implementação do algoritmo *backpropagation* deve-se, primeiramente, calcular o resultado de B.18 ou B.22, dadas por

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

Pode-se notar que a δ_j^L depende da derivada parcial da função de custo em relação a entrada z_j^L, ou da derivada parcial da função de custo em relação a saída a_j^L multiplicada pela derivada parcial da função de

ativação do neurônio em relação a sua entrada z_j^L.

Aqui deduziremos estas derivadas para as funções de custos apresentadas. Também será omitido o índice L durante o desenvolvimento, sendo apresentado somente quando necessário.

B.5.1 Derivando $\sigma(z_j^L)$ para uma Sigmoide

A função de ativação sigmoide é dada em 3.13 como

$$\sigma(z_j) = \frac{1}{1+e^{-z_j}}$$

Então a sua derivada é dada por

$$\sigma'(z_j) = \frac{\partial \sigma(z_j)}{\partial z_j} = \frac{\partial}{\partial z_j}(1+e^{-z_j})^{-1}$$

$$\sigma'(z_j) = -(1+e^{-z_j})^{-2}\frac{\partial}{\partial z_j}(1+e^{-z_j})$$

$$\sigma'(z_j) = \frac{e^{-z_j}}{(1+e^{-z_j})^2} = \frac{1}{1+e^{-z_j}}\left(\frac{1-1+e^{-z_j}}{1+e^{-z_j}}\right)$$

$$\sigma'(z_j) = \frac{1}{1+e^{-z_j}}\left(\frac{1+e^{-z_j}}{1+e^{-z_j}} - \frac{1}{1+e^{-z_j}}+\right) = \frac{1}{1+e^{-z_j}}\left(1 - \frac{1}{1+e^{-z_j}}\right)$$

$$\sigma'(z_j) = \sigma(z_j)(1-\sigma(z_j)) \tag{B.50}$$

De acordo com a expressão B.50, $\sigma'(z_j)$ pode ser expresso em função do próprio $\sigma(z_j)$. Reescrevendo B.50 em sua forma vetorial para uma determinada camada k obtém-se

$$\sigma'(\mathbf{z_k}) = \sigma(\mathbf{z^k})(1-\sigma(\mathbf{z^k})) \tag{B.51}$$

B.5.2 Obtendo δ_j^L para o Erro Quadrático Médio e Sigmoide

Para obter o valor de δ_j^L devemos calcular $\frac{\partial C}{\partial a_j^L}$. Omitindo-se o índice L, temos que o erro quadrático médio para uma única amostra é dado por

$$C = \frac{\|\mathbf{y}-\mathbf{a}\|^2}{2} = \frac{1}{2}\sum_i (y_i - a_i)^2$$

A sua derivada parcial é, então

$$\frac{\partial C}{\partial a_j} = \frac{2}{2}(y_j - a_j)(-1)$$

$$\frac{\partial C}{\partial a_j^L} = a_j^L - y_j^L \tag{B.52}$$

Expressando B.52 em sua forma vetorial

$$\nabla \mathbf{c^L} = \mathbf{a^L} - \mathbf{y} \tag{B.53}$$

Pode-se, então, calcular $\delta^{\mathbf{L}}$ substituindo-se B.51 e B.53 em B.23:

$$\delta^{\mathbf{L}} = (\mathbf{a^L} - \mathbf{y}) \odot \sigma'(\mathbf{z^L}) \tag{B.54}$$

B.5.3 Obtendo δ_j^L para Cross-Entropy e Sigmoide

Dada a função cross-entropy (B.49) para uma única amostra, e omitindo-se o índice L:

$$C = -1 \sum_j [y_j ln(a_j) + (1 - y_j) ln(1 - a_j)]$$

Podemos derivá-la como:

$$\frac{\partial C}{\partial a_j} = \frac{-y_j}{a_j} - \left(\frac{1 - y_j}{1 - a_j}\right) \frac{\partial}{\partial a_j}(1 - a_j)$$

$$\frac{\partial C}{\partial a_j} = \frac{-y_j}{a_j} + \frac{1 - y_j}{1 - a_j}$$

$$\frac{\partial C}{\partial a_j^L} = \frac{-y_j(1 - a_j^L) + a_j^L(1 - y_j)}{a_j^L(1 - a_j^L)} \tag{B.55}$$

Agora podemos calcular $\delta^{\mathbf{L}}$ substituindo B.50 e B.55 em B.23:

$$\delta_j^L = \left(\frac{-y_j(1 - a_j^L) + a_j^L(1 - y_j)}{a_j^L(1 - a_j^L)}\right) \sigma(z_j^L)(1 - \sigma(z_j^L))$$

$$\delta_j^L = \left(\frac{-y_j(1 - a_j^L) + a_j^L(1 - y_j)}{a_j^L(1 - a_j^L)}\right) a_j^L(1 - a_j^L))$$

$$\delta_j^L = a_j^L - y_j \tag{B.56}$$

Que em sua forma vetorial fica

$$\delta^{\mathbf{L}} = \mathbf{a^L} - \mathbf{y^L} \tag{B.57}$$

B.5.4 Obtendo δ_j^L para Log-Likelihood e Softmax

Dada a função log-likelihood (3.20) para uma única amostra, e omitindo-se o índice L:

$$C = -\sum_j y_j \ln a_j$$

Substituindo a_j pela função softmax obtém-se

$$C = -\sum_j y_j \ln\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right)$$

$$C = -\sum_j y_j z_j + \sum_j y_j \ln(\sum_k e^{z_k}) \tag{B.58}$$

Para obter δ_j^L vamos utilizar a expressão B.18, pois podemos facilmente diferenciar com respeito a z_j^L. Então

$$\delta_j = \frac{\partial C}{\partial z_j} = -y_j + \sum_j y_j \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Como y representa uma probabilidade, então $\sum_j y_j = 1$. E $\frac{e^{z_j}}{\sum_k e^{z_k}} = a_j$, então

$$\delta_j^L = a_j^L - y_j \tag{B.59}$$

Que em sua forma vetorial fica

$$\delta^{\mathbf{L}} = \mathbf{a^L} - \mathbf{y^L} \tag{B.60}$$

Podemos observar que a forma de $\delta^{\mathbf{L}}$ para a função de cross-entropy e para a log-likelihood são idênticas.

B.6 Inicialização dos Pesos de uma ANN

Os pesos e biases normalmente são inicializados como variáveis aleatórias com uma distribuição Gaussiana, e espera-se que o algoritmo de treinamento seja responsável por encontrar os valores que minimizem a função de custo. A questão que fica é escolher os parâmetros da distribuição Gaussiana.

Para exemplificar, vamos supor uma rede com 1000 neurônios de entrada e que estamos inicializando os pesos como variáveis aleatórias com uma distribuição normal. Também vamos supor que metade dos neurônios de entrada possuem valor 1 e a outra metade valor zero. Então a soma ponderada de um determinado neurônio da primeira camada oculta será $z = \sum_j w_j x_j + b$, sendo w_j o peso dado a entrada x_j e b o bias do respectivo neurônio. Como metade das entradas x_j possuem o valor zero, z será a soma de 501 variáveis aleatórias com distribuição normal, ou seja, z também será uma variável aleatória com distribuição Gaussiana de média zero, porém com desvio padrão igual à $\sqrt{501} \approx 22.4$, como pode ser visto na figura B.1.

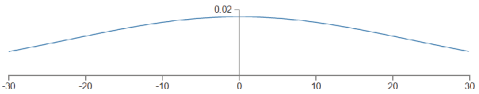


Figura B.1: Distribuição do z de um neurônio com a rede inicializada com distribuição normal.

Podemos ver pela figura B.1 que provavelmente $z >> 1$ ou $z << -1$. Dessa forma a saída do neurônio $\sigma(z)$ estará saturada. Dessa forma, se o algoritmo de treinamento fizer pequenas variações nos pesos, a saída do neurônio permanecerá praticamente inalterada (z continuará sendo alto e $\sigma(z)$ continuará na região de saturação). Isso significa que esses pesos aprenderão muito lentamente quando utilizado o algoritmo *gradient descent*. Um problema semelhante ao apresentado na seção B.4.

Uma solução para diminuir esse efeito é inicializar os pesos como variáveis aleatórias com distribuição Gaussiana de média zero e desvio padrão igual a $1/\sqrt{n_{in}}$, onde n_{in} é o número de neurônios da camada de entrada. Pensando no exemplo anterior, onde temos 1000 entradas com

500 nulas e 500 com valor 1, agora determinado neurônio da primeira camada oculta possuirá um z com distribuição Gaussiana de média zero e desvio padrão igual a $\sqrt{3/2} = 1.22$, como mostrado na figura B.2.

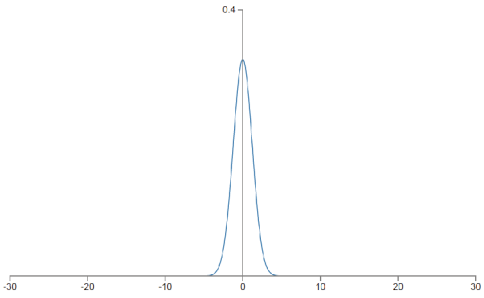


Figura B.2: Distribuição do z de um neurônio com a rede inicializada com distribuição Gaussiana de média zero e desvio padrão $1/\sqrt{n_{in}}$.

Observando a figura B.2 pode-se notar que agora z estará mais próximo de zero, e será menos provável que a saída do neurônio esteja saturada.

Importante observar que os biases podem ser inicializados com uma distribuição normal. O resultado final do treinamento após determinada quantidade de épocas tende a ser semelhante independentemente de qual das duas formas os pesos foram inicializados, com a diferença que na segunda forma serão necessários uma quantidade menor de épocas.