PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

JUAREZ MONTEIRO DOS SANTOS JÚNIOR

ANALISANDO A VIABILIDADE DE *DEEP LEARNING* PARA RECONHECIMENTO DE AÇÕES EM *DATASETS* PEQUENOS

Porto Alegre
2018

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANALISANDO A VIABILIDADE DE *DEEP LEARNING* PARA RECONHECIMENTO DE AÇÕES EM *DATASETS* PEQUENOS

JUAREZ MONTEIRO DOS SANTOS JÚNIOR

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Rodrigo Coelho Barros

Porto Alegre
2018

Ficha Catalográfica

S237a Santos Júnior, Juarez Monteiro dos

Analisando a Viabilidade de Deep Learning para Reconhecimento de Ações em Datasets Pequenos / Juarez Monteiro dos Santos Júnior . – 2018.

103 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rodrigo Coelho Barros.

1. Aprendizado de Máquina. 2. Redes Neurais. 3. Redes Neurais Convolucionais. 4. Reconhecimento de Ações. I. Barros, Rodrigo Coelho. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecário responsável: Marcelo Votto Texeira CRB-10/1974

Juarez Monteiro dos Santos Júnior

Analisando a Viabilidade de *Deep Learning* para Reconhecimento de Ações em *Datasets* Pequenos

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 6 de Março de 2018.

BANCA EXAMINADORA:

Prof. Dr. Ricardo Matsumura de Araújo (Centro de Desenvolvimento Tecnológico/UFPEL)

Profa. Dra. Soraia Raupp Musse (PPGCC/PUCRS)

Prof. Dr. Rodrigo Coelho Barros (PPGCC/PUCRS - Orientador)

DEDICATÓRIA

Eu dedico esta dissertação a minha mãe, Mara da Silva dos Santos, e ao meu pai, Juarez Monteiro dos Santos, por todo amor, carinho e compreensão que eles me forneceram durante toda a minha jornada.

"*The ending isn't any more important than any of the moments leading to it.*"
(To the Moon)

AGRADECIMENTOS

Eu gostaria de agradecer a Pontifícia Universidade Católica do Rio Grande do Sul, aos funcionários envolvidos no Programa de Pós Graduação em Ciência da Computação, por confiarem e terem me fornecido a oportunidade de conduzir minha pesquisa, fornecendo toda e qualquer assistência que eu precisei durante o curso.

Agradeço ao meu orientador, Rodrigo Barros, por sua paciência e ajuda. Obrigado por todo o conhecimento e confiança que você me forneceu durante o período deste trabalho. Também quero agradecer todos os professores que contribuíram de alguma maneira para a minha formação como um estudante de mestrado.

Eu quero carinhosamente agradecer meus amigos do grupo de *Machine Intelligence and Robotics*, em especial os alunos Henry Cagnini, João Aires e Renan Maidana, os quais estavam sempre disponíveis para me auxiliar, motivar e fornecer ótimas sugestões. Certamente meu trabalho não teria sido o mesmo sem vocês. Eu também gostaria de agradecer meus amigos, Arthur Wurth, Andersen Lopes, Andressa Manczak e Monique Ribas, apesar de toda a distância, vocês me entenderam, me auxiliaram, e sempre tiveram palavras de conforto para mim. Em especial eu quero reconhecer e agradecer meu colega de pesquisa, Roger Granada, por todo auxílio, conhecimento transmitido e pela amizade construída. Eu agradeço a todos, definitivamente esta parte da minha vida teria sido muito mais difícil sem vocês, meus amigos.

Eu reconheço e agradeço o apoio financeiro recebido através do projeto desenvolvido pela empresa Hewlett Packard, sem tal apoio eu não teria como concluir o curso de mestrado. Eu também gostaria de agradecer todos os membros envolvidos nesse projeto e as oportunidades que foram-me oferecidas.

Eu agradeço a todas as pessoas que me auxiliaram e me ajudaram de alguma forma durante o desenvolvimento desta dissertação, e aqueles que eu esqueci de mencionar aqui. Eu espero que vocês saibam que eu sou extremamente grato por tudo o que vocês fizeram por mim.

Por fim, mas não menos importante, eu gostaria de agradecer meus pais, Juarez e Mara, não só por todo amor e auxílio que vocês me deram, mas também pelo esforço em tentar compreender o caminho que eu escolhi, e por sempre estarem ao meu lado nas minhas decisões. Eu agradeço as minhas irmãs, Juliana e Luciana, pelo apoio e pela compreensão, apesar da distância vocês sempre acreditaram e confiaram em mim.

ANALISANDO A VIABILIDADE DE *DEEP LEARNING* PARA RECONHECIMENTO DE AÇÕES EM *DATASETS* PEQUENOS

RESUMO

Reconhecimento de ação é a tarefa de visão computacional que identifica qual ação esta ocorrendo em dada sequência de *frames*. Abordagens tradicionais dependem de características extraídas dessas imagens e algoritmos específicos de domínio, muitas vezes resultando em uma precisão limitada. Os avanços substanciais na aprendizagem profunda e a disponibilidade de conjuntos de dados maiores permitiram que técnicas produzam um desempenho sem conhecimento específico do domínio para reconhecer as ações que estão sendo realizadas, tendo como base apenas sequências de vídeo. No entanto, os algoritmos de aprendizagem profunda geralmente requerem conjuntos de dados rotulados muito grandes para o treinamento. Devido à sua maior capacidade, tais algoritmos geralmente sofrem com *overfitting* em conjunto de dados pequenos, proporcionando assim um menor poder de generalização. Este trabalho tem como objetivo explorar a aprendizagem profunda no contexto de conjuntos de dados pequenos para reconhecimento de ações. Nosso objetivo é alcançar resultados, mesmo nos casos em que os dados rotulados não sejam abundantes. Para isso, investigamos diferentes arquiteturas profundas, diferentes métodos de processamento, e diferentes métodos de fusão, fornecendo diretrizes e boas práticas para o aprendizado profundo em conjuntos de dados de tamanho pequeno.

Palavras-Chave: aprendizado de máquina, redes neurais, redes neurais convolucionais, reconhecimento de ações.

ANALYZING THE FEASIBILITY OF DEEP LEARNING TO RECOGNIZE ACTIONS IN SMALL DATASETS

ABSTRACT

Action recognition is the computer vision task of identifying which action is happening in a given sequence of *frames*. Traditional approaches rely on handcrafted features and domain-specific algorithms, often resulting in limited accuracy. The substantial advances in deep learning and the availability of larger datasets have allowed techniques that yield better performance without domain-specific knowledge to recognize actions being performed based on the raw information from video sequences. However, deep learning algorithms usually require very large labeled datasets for training, and due to their increased capacity their often overfit small data, hence providing lower generalization power. This work aims to explore deep learning in the context of small-sized action recognition datasets. Our goal is to achieve significant performance even in cases in which labeled data is not abundant. In order to do so, we investigate distinct network architectures, data pre-processing, and fusion methods, providing guidelines and good practices for using deep learning in small-sized datasets.

Keywords: machine learning, neural networks, convolutional neural networks, action recognition.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

GPU – *Graphics Processing Unit*

CNN – *Convolutional Neural Network*

SVM – *Support Vector Machine*

LSTM – *Long Short-Term Memory*

ML – *Machine Learning*

DL – *Deep Learning*

OF – *Optical Flow*

DOF – *Dense Optical Flow*

IA – Inteligência Artificial

ANN – *Artificial Neural Networks*

RELU – *Rectified Linear Unit*

MLP – *Multi-layer Perceptron*

SGD – *Stochastic Gradient Descent*

LR – *Learning Rate*

RNN – *Recurrent Neural Network*

HMM – *Hidden Markov Models*

HOG – *Histogram of Gradients*

HOF – *Histogram of Optical Flow*

MBH – *Motion Boundary History*

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

Com o avanço da tecnologia e o crescimento exponencial do conteúdo de mídia, como vídeos produzidos por dispositivos móveis e sistemas de vigilância, uma quantidade cada vez maior de dados está se tornando disponível. Esta abundância de dados impôs uma forte demanda pela análise automática do conteúdo de vídeos, atraindo considerável interesse em pesquisa das áreas de visão computacional e inteligência artificial. Entre as possíveis tarefas a serem exploradas, o reconhecimento de atividades humanas encontra aplicações em uma variedade de domínios, incluindo a área de vigilância [HTWM04, LSPZ08, KCY+10], interação humano-computador [SSK+13] e monitoramento de idosos [CNW12, Ras14, Tsu15].

Embora existam muitos esforços sendo realizados nesta área de pesquisa, o reconhecimento de ações em sequências de vídeo ainda é um objetivo de pesquisa desafiador para visão computacional. Registraram-se progressos consideráveis nos últimos anos devido aos avanços no hardware, mais especificamente no que diz respeito às Unidades de Processamento Gráfico (do inglês, *Graphics Processing Unit*, GPU) , permitindo que algoritmos processem enormes volumes de dados. Este progresso em particular na tecnologia é dito ser um dos principais motivos para o sucesso de arquiteturas neurais profundas, como as Redes Neurais Convolucionais (do inglês, *Convolutional Neural Network*, CNN) [LBBH98].

Nos últimos anos, as CNNs alcançaram resultados estado-da-arte ao lidar com reconhecimento de objetos estáticos [KSH12, GDDM14, LSD15]. Encorajados por esses resultados positivos, abordagens recentes propõem o desenvolvimento de redes neurais profundas para tarefas de classificação de vídeo, como o reconhecimento de ações [CZ17]. No entanto, o reconhecimento de ações requer raciocínio sobre dados espaciais e temporais, o que acrescenta uma camada extra de complexidade para resolver o problema de forma satisfatória.

1.2 Problema

Antes do surgimento de CNNs e de redes neurais profundas em geral, pesquisas em visão computacional utilizavam algoritmos matemáticos complexos, os quais foram feitos para extrair características relevantes de vídeos. Após a extração dessas características (do inglês, *handcrafted features*, como são conhecidas), essas, normalmente, são

direcionadas como entrada para um algoritmo classificador, para distinguir qual ação está ocorrendo dado um vídeo ou imagem de entrada. Embora essas abordagens funcionem muito bem em cenários controlados, em domínios mais complexos é muito difícil saber quais características são importantes para uma determinada tarefa, uma vez que a escolha das características é dependente do problema, *i.e.*, diferentes ações podem exigir características diferentes em termos dos seus padrões de movimento. As CNNs, por outro lado, aprendem uma hierarquia de conceitos de forma automatizada, sem intervenção humana, em um processo geralmente denominado "aprendizado de representações" [GBCB16].

Para uma CNN classificar um conteúdo de vídeo, uma abordagem simples é tratar quadros de vídeo como imagens estáticas e usar a rede para classificar cada quadro individualmente. No entanto, tal abordagem perde informações importantes, pois falha na detecção das informações de movimento codificadas em uma sequência de quadros. Para resolver esta questão, várias abordagens tentam aprender informações temporais estendendo a CNN tradicional com a adição de uma dimensão temporal extra, gerando as chamadas convoluções 3D [JXYY13, SJYS15, TBF+15]. Essa estrutura pretende tornar a rede capaz de capturar conceitos ao longo de dimensões espaciais e temporais. O problema de tal abordagem é que aumenta substancialmente a quantidade de parâmetros aprendíveis, aumentando as chances de *overfitting*, quando o modelo aprende os dados de treinamento mas não é capaz de generalizar para novos dados. A maneira óbvia de evitar o *overfitting* é ter um conjunto de dados suficientemente grande para aprender corretamente a informação temporal sem afetar a capacidade de generalização da rede. Devido aos problemas de lidar com conjuntos de dados muito amplos (por exemplo, mais de 1 milhão de vídeos = bilhões de quadros), muitos autores propõem misturar a informação espacial com dados temporais extraídos por algoritmos de visão computacional, *e.g.*, dados extraídos de *optical flow* [Far03], utilizados em uma estratégia conhecida por *Two-Stream* CNN [SZ14a, WQT15, YWZ+15, FPZ16a, PHBB16].

De fato, as abordagens no estado-da-arte para o reconhecimento de ações utilizam alguma modificação da abordagem *Two-Stream* CNN [KTS+14, STWH16, FPZ16b], porém, o desempenho dessas redes em conjuntos de dados pequenos e médios não é substancialmente superior às abordagens que extraem *handcrafted features* e utilizam um algoritmo classificador junto a essas. Isto ocorre devido à dificuldade em controlar adequadamente o *overfitting* desses modelos quando se treina com poucos dados [WXW+16].

1.3 Abordagem Proposta

Este trabalho tem como objetivo medir e analisar o desempenho das CNNs para reconhecimento de ações em pequenos conjuntos de dados. Por pequeno, entende-se conjuntos de dados que variam de alguns milhares de vídeos (clipes) a centenas de milha-

res [SKD+13, RAAS12, ITKR14, DXDC13, MPK09]. Nosso objetivo é testar as principais abordagens para o reconhecimento de ação que foram usadas em grandes conjuntos de dados e verificar sistematicamente se funcionam em conjuntos de dados pequenos. Se verificarmos empiricamente que não funcionam corretamente, como apontado na literatura [WXW+16], pretendemos mencionar onde possíveis melhorias possam ser feitas para atingir melhores resultados e futuramente desenvolver arquiteturas e abordagens capazes de funcionar bem em diferentes conjuntos de dados de pequeno porte.

Existem três principais hipóteses que adotamos dado o problema abordado nesta dissertação:

- *Deep Learning* irá atingir resultados relevantes quando utilizado com *datasets* pequenos;
- Utilizar *Deep Learning* com *handcrafted features* irá produzir resultados superiores do que utilizar qualquer uma dessas técnicas separadamente;
- Os melhores modelos de *Deep Learning* serão gerados por abordagens que envolvem o aspecto temporal.

Apesar do fato de que várias estratégias de reconhecimento de ações dependem de diferentes tipos de sensores, como acelerômetros [RDML05], giroscópios [GFH09], ou sensores de ambiente, como infravermelho e sensores de movimento instalado em portas, armários, gavetas e janelas [LHP+07], este trabalho apenas se concentra em abordagens que utilizam sequências de vídeos (fluxos de pixels) para identificar as ações que estão sendo executadas.

1.4 Contribuições

A contribuição deste trabalho é uma análise em profundidade sobre pequenos conjuntos de dados usando diferentes algoritmos de reconhecimento de ação. Nós utilizamos algoritmos de classificação tais como Máquinas de Vetores de Suporte (do inglês, *Support Vector Machine*, SVM) e CNNs para imagens estáticas (VGG16 e Inception-V3); e CNNs temporais, como *Two-Stream* CNN, 3D CNN (C3D) e uma rede neural recorrente, LSTM (do inglês, Long Short-Term Memory), para verificar o impacto desses modelos em conjuntos de dados pequenos na tarefa de reconhecimento de ação. Não testamos apenas diferentes algoritmos, mas também tentamos otimizar os hiper-parâmetros da maior parte desses modelos. Para isso, realizamos uma experimentação baseada em *grid*, contendo 4 valores diferentes para o (*dropout*) e 6 valores diferentes para a taxa de aprendizado (*learning rate*).

É importante mencionar que não existem outros trabalhos similares na literatura que fazem tal comparação, e que esta análise é muito importante para os usuários que não

possuem um enorme volume de dados disponível para realizar a tarefa de classificação de ações. Este trabalho pode ser usado como fonte para que os usuários possam previamente analisar e verificar qual seria o melhor caminho a seguir em uma *dataset* pequeno.

Outra contribuição relevante é o fato de que, durante o desenvolvimento deste trabalho, nós propusemos diferentes abordagens para atingir melhores resultados com as bases de dados utilizadas nas análises. Todas as abordagens construídas durante o desenvolvimento desse trabalho foram publicadas em conferências, nacionais e internacionais, de inteligência artificial e de redes neurais.

1.5 Estrutura do Trabalho

Este trabalho está organizado da seguinte forma. A fundamentação teórica é descrita no Capítulo 2. O Capítulo 3 apresenta os trabalhos relacionados presentes na literatura e referentes a esta pesquisa. Já o Capítulo 4 detalha a metodologia que aplicamos para desenvolver este trabalho. A configuração experimental, incluindo a otimização de parâmetros, os resultados obtidos e uma discussão em relação aos experimentos desenvolvidos nesse trabalho são apresentados no Capítulo 5. Para finalizar, o Capítulo 6 conclui o trabalho com a relação de nossas limitações, direções de possíveis trabalhos futuros e publicações que realizamos durante o desenvolvimento dessa pesquisa.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma breve introdução sobre reconhecimento de ações, aplicações e desafios. Não obstante, também fornece uma explicação sobre Aprendizado de Máquina (do inglês, *Machine Learning*, ML) e Aprendizado Profundo (do inglês, *Deep Learning*, DL), bem como uma descrição de algumas abordagens que foram utilizadas nesse trabalho.

2.1 Reconhecimento de Ações

Existem diversas definições para reconhecimento de ações e reconhecimento de atividades. Turaga *et al.* [TCSU08] descrevem ações como um simples padrão de movimento que geralmente é executado por uma única pessoa em um curto período de tempo. Eles também descrevem as atividades como uma complexa sequência de ações realizadas por vários seres humanos que podem estar interagindo um com o outro de forma restrita. Uma definição de ação não-relacionada ao tempo é dada por Weinland *et al.* [WRB11], que diz que uma ação é uma sequência de movimentos gerados por um agente humano durante a execução de uma tarefa. Moeslund *et al.* [MHK06] e Poppe [Pop10] têm uma definição mais rígida para esses conceitos. Primeiro, eles definem o conceito de ação primitiva como um movimento atômico que pode ser descrito no nível dos membros. Por exemplo, “perna esquerda para frente” é um tipo de ação primitiva. Uma ação é definida pelo uso de ações primitivas descrevendo um ciclo de movimento do corpo todo. Finalmente, a atividade é definida como uma sequência de ações que gera uma interpretação para o movimento que está sendo executado. Por exemplo, “saltar obstáculos” é uma atividade que contém ações “preparar”, “pular” e “correr”. Wang *et al.* [WFG16] sugerem que o verdadeiro significado de uma ação pode não ser codificado no próprio movimento, mas sim na “mudança ou transformação que uma ação traz para o ambiente”. Por exemplo, a noção de “chutar uma bola” reside na mudança de estado da bola (aceleração) causada pela perna do jogador. Um exemplo dessas classificações é descrito por Herath *et al.* [HHP16] e ilustrado na Figura 2.1, onde o “movimento da perna” primitivo é executado pelo jogador em sua corrida. O movimento é denominado como uma ação primitiva porque mal podemos atribuir um significado a ele. Em contrapartida, o movimento coletivo de membros, que resulta em correr, tem um significado e, portanto, pode ser considerado a “ação em execução”. Equivalentemente, o chute do jogador pode ser identificado como a “ação de chutar”.

Da perspectiva do computador, Shi *et al.* [SZHW15] define a tarefa de reconhecimento de ação como a identificação das ações ou comportamentos de uma ou mais pessoas através de uma série de observações em uma sequência de vídeo. Rashidi [Ras14]

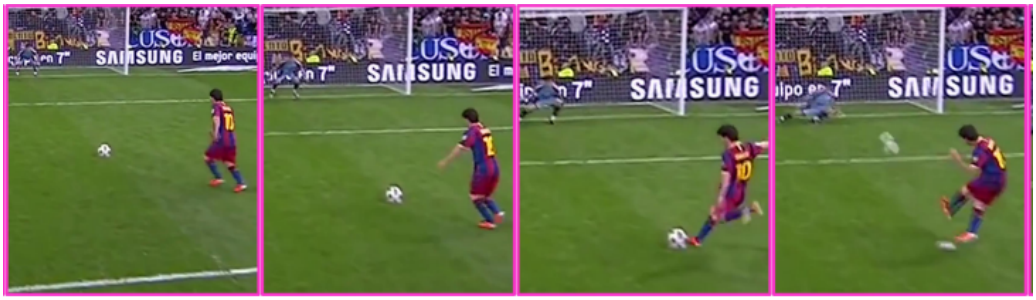


Figura 2.1 – Humano interagindo com o ambiente (adaptado de [HHP16]).

é um pouco mais amplo do que outros autores e define uma ação como uma sequência de etapas realizadas por um humano para alcançar um determinado objetivo, onde cada etapa deve ser mensurável por um estado ou por uma combinação de estados. Chaquet *et al.* [CCFC13] explicam que os sistemas de reconhecimento de ação visam identificar as ações e objetivos de um ou mais agentes a partir de uma série de observações.

Neste trabalho, aderimos à definição de ação proposta por Weinland*et al.* [WRB11], onde ele descreve que uma ação é uma sequência de movimentos gerados por um agente humano durante a execução de uma tarefa, enquanto que uma atividade pode ser descrita como uma sequência de ações que gera uma interpretação para o movimento que está sendo executado, como explicado por Moeslund *et al.* [MHK06]. Estes movimentos podem ser representados de diversas formas, sejam eles imagens, vídeos, audios, *etc*.

2.1.1 Pré-processamento

Para diferentes algoritmos de ML ou DL, é importante realizar o pré-processamento nos dados que serão utilizados. Não somente a qualidade do modelo em questão é importante, bem como a qualidade dos dados que serão inseridos nesse. Algumas técnicas de pré-processamento consistem em reduzir a redundância dos dados, eliminar os ruídos, normalizar, padronizar, extrair características mais relevantes, entre outras. Para alguns algoritmos, é desejável possuir a média dos dados centrada em zero e a variança igual entre as variáveis, fazendo com que a busca do otimizador pelo mínimo global seja menos custosa. A técnica de pré-processamento que leva os dados a tais características é conhecida como normalização. Para aplicar esta técnica em imagens contendo três canais de cores (RGB), onde os valores de pixels estão entre 0 e 255, basta subtrair os valores de cada canal da imagem por 128 e dividi-los também por 128. Esse processo não irá alterar o significado das imagens, e ainda irá auxiliar no processo de aprendizado do algoritmo utilizado.

Uma técnica de pré-processamento com o objetivo de extrair informações e que é extremamente utilizada para reconhecimento de ações é conhecida como Fluxo Óptico (do inglês, *Optical Flow*, OF). O conceito de OF foi introduzido por Gibson [Gol75] e pode

ser visto como um conjunto de campos de vetores de deslocamento entre quadros consecutivos de um vídeo o qual representa a interpretação do campo de movimento com uma intensidade de imagem variável no tempo. A versão densa do fluxo óptico (do inglês, *Dense Optical Flow*, DOF) representa como e onde cada pixel na sequência da imagem está em movimento.

2.1.2 Aplicações e Desafios em Reconhecimento de Ações

Reconhecimento de ações e reconhecimento de atividades são tópicos de pesquisa ativos em visão computacional e aprendizado de máquina. O interesse no tema é motivado por muitas aplicações, tais como o monitoramento de idosos dentro de ambientes fechados [CNW12, Ras14, Tsu15], na segurança pública [LSPZ08, KCY+10, HVL08, PW12] e no processamento de vídeo [SSK+13, KTS+14, MHK06].

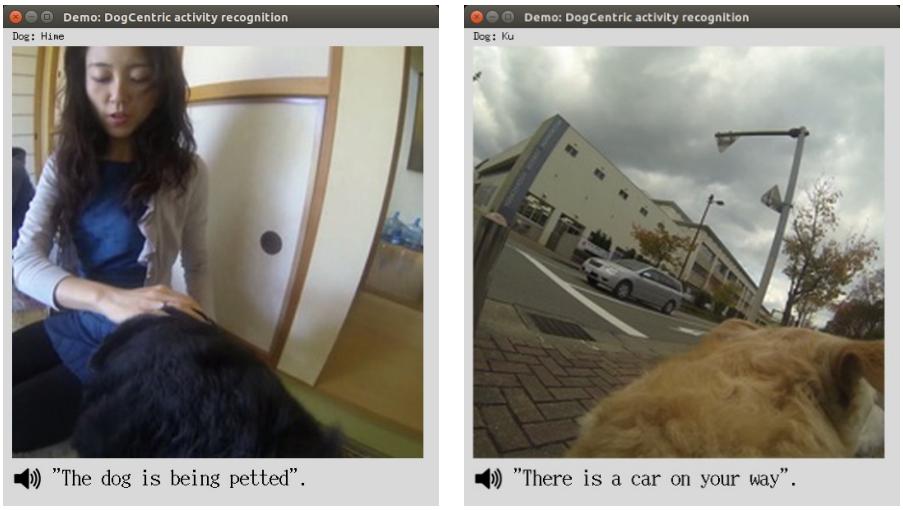
Para monitoramento de idosos, os indivíduos têm estilos de vida, hábitos ou habilidades diferentes, e, como tal, têm sua própria maneira de realizar atividades. Chen *et al.* [CNW12] apontam a importância de monitorar as mudanças ambientais e as atividades de idosos em casa. Um sistema pode processar os dados dos sensores, tomar decisões e as medidas adequadas para ajudar um habitante a realizar atividades do dia-a-dia. Para os idosos, isso estenderia o período de tempo vivendo de forma independente dentro de seu próprio ambiente doméstico.

Ao lidar com a segurança pública, o monitoramento é cansativo, caro e ineficaz, e mesmo uma pessoa dedicada não irá garantir um sistema de segurança efetivo [HVL08]. Moeslund *et al.* [MHK06] apontam para a importância de monitorar e identificar ações específicas, *i.e.*, a vigilância de um estacionamento, onde um sistema rastreia sujeitos para avaliar se eles estão prestes a cometer um crime, como roubar um carro. Assim, uma solução prática automatiza o processo de monitoramento e usa pessoas apenas para avaliar os eventos detectados.

Em relação ao processamento de vídeo, Poppe [Pop10] menciona a importância do reconhecimento de ação para uma eficiente pesquisa de vídeo baseada em conteúdo. Por exemplo, um algoritmo poderia encontrar roubos de bola em jogos de futebol ou movimentos típicos de dança em músicas. Um exemplo bem conhecido de reconhecimento de ação é o console de jogos Kinect [SSK+13], desenvolvido pela Microsoft, que permite ao usuário interagir com o jogo por meio de gestos sem qualquer dispositivo controlador. Reconhecer ações através do Kinect pode ser aplicado para fornecer funcionalidades extras de controle para o usuário, podendo ser utilizado como uma interface para jogos, ambientes virtuais, animação ou para controlar dispositivos remotamente [MHK06].

Existem aplicações que oferecem suporte a pessoas com deficiência visual usando diferentes métodos de visão por computador. Durante este projeto, criamos uma aplicação

que sustenta a importância de construir “mapas mentais” através do reconhecimento de ação e informações em áudio [MAG+17]. Os mapas mentais fazem parte do processo de orientação em seres humanos que envolvem a memória [LM05]. O mapeamento mental dos espaços é essencial para o desenvolvimento de uma orientação e mobilidade eficientes. As pessoas com deficiência visual ou com cegueira não possuem uma importante fonte de informação (visão) para construir tais mapas mentais e devem se adaptar para construir seus mapas mentais usando fontes alternativas de informação. Nós propusemos uma arquitetura que poderia alcançar bons resultados em um pequeno conjunto de dados de reconhecimento de ação. Com a ação prevista, passamos essas informações em forma de áudio para o usuário, a fim de ajudá-lo a construir seu mapa mental. A Figura 2.2 apresenta duas telas da nossa aplicação ao detectar duas possíveis ações, acariciar e carro (do inglês, *pet* e *car*). A fim de fornecer um *feedback* auditivo, a sentença produzida pela aplicação será disponibilizada em forma de áudio para o usuário. Assim, quando a aplicação detecta alguém acariciando o cão, ele produz a frase “O cachorro está sendo acariciado” (a) e quando um carro aparece no vídeo, a aplicação produz a frase “Existe um carro no seu caminho” (b).



(a) (b)

Figura 2.2 – Aplicação feita para auxiliar pessoas com deficiência visual.

Apesar de diferentes aplicações, a tarefa de reconhecimento de ações possui muitos desafios. Por exemplo, grandes variações nas aparências de atividade e objeto, escassez de dados anotados, definição de ação ambígua e derivação de conceito em ambientes dinâmicos podem levar a problemas [HRC15]. Yu e Yuan [YY15] destacam os desafios devido à diversidade e variação das ações humanas. Às vezes, é difícil diferenciar as ações humanas devido à mudança do ambiente de fundo e outros movimentos dinâmicos que normalmente ocorrem nos vídeos. Diferentemente da detecção de objetos, que depende apenas de aparência visual, uma detecção de ação precisa levar em consideração ambas as indicações de aparência e movimento. Problemas como localização de pessoa, robus-

tez à oclusão parcial, desordem de fundo, sombras e variação de iluminação precisam ser devidamente abordados [SZHW15].

Há desafios para reconhecer as atividades devido à natureza errática da atividade humana, pois podem ser realizadas com um alto grau de liberdade em relação ao caminho e à ordem sequencial que as ações são realizadas. Embora as ações geralmente sigam algum tipo de padrão, não há restrições estritas na sequência e duração das ações. Por exemplo, para realizar a atividade de preparação de um prato, primeiro pode-se fazer o molho e depois cozinhar o arroz, enquanto outros podem cozinhar primeiro o arroz e depois fazer o molho. Essa grande variabilidade e flexibilidade na maneira como as atividades podem ser realizadas requerem abordagens escaláveis para longas sequências de ações. Weinland *et al.* [WRB11] menciona os desafios ao lidar com a grande variabilidade que uma ação pode exibir quando realizada por diferentes pessoas de diferentes gêneros e tamanhos, e com diferentes velocidades e estilos. Por exemplo, a ação de chutar uma bola pode ser realizada de forma muito diferente em cenários reais.

2.2 *Machine Learning* e *Deep Learning*

Machine Learning é um campo dentro da área da Inteligência Artificial (IA) capaz de "aprender" sem ser programado explicitamente. Em outras palavras, é a área onde algoritmos são capazes de aprender através de exemplos [GBCB16]. Mitchel [Mit97] descreve que um programa de computador é dito aprender com a experiência E em relação a alguma classe de tarefas T e medição de desempenho P se o desempenho em tarefas em T, conforme medido por P, melhora com a experiência E. Os algoritmos de ML são majoritariamente divididos em aprendizado supervisionado e não supervisionado. O aprendizado supervisionado, utilizado por todos algoritmos de classificação desta dissertação, necessita dos dados com os seus respectivos rótulos. A ideia é que através de um dado de entrada X, existe uma saída Y, e a tarefa desses algoritmos é realizar o mapeamento da função ($f : X \rightarrow Y$) através de exemplos, para que dado uma entrada não vista, o algoritmo seja capaz de mapear para a sua respectiva saída [Alp14].

Os algoritmos de ML, tais como as Redes Neurais Artificiais (do inglês, *Artificial Neural Networks*, ANN) muitas vezes foram usados para tarefas de reconhecimento de ações. No entanto, as técnicas convencionais de ML são limitadas na sua capacidade de processar dados em sua forma bruta (dados não estruturados) [LBH15]. Durante décadas, a construção de sistemas de ML exigiu conhecimentos consideráveis de domínio para criar uma representação interna a partir da qual o subsistema de aprendizado possa detectar ou classificar padrões na entrada. Algoritmos de DL resolvem esse problema criando uma representação interna que é construída em termos de características hierárquicas. Alguns algoritmos de DL, como as CNNs, possuem um conjunto de métodos que permitem que um

sistema seja alimentado com dados brutos e descubra automaticamente as representações necessárias para a detecção ou classificação desta entrada.

De acordo com Goodfellow *et al.* [GBCB16], *Deep Learning* é um tipo particular de *Machine Learning* que alcança grande poder e flexibilidade ao aprender a representar o mundo como uma hierarquia de conceitos aninhada, com cada conceito definido em relação a conceitos mais simples, e mais representações abstratas calculadas em termos menos abstratos. Para Deng e Yu [DY14], DL pode ser considerado como uma classe de técnicas de ML que exploram muitas camadas de processamento de informações não-lineares para extração e transformação de *features*, supervisionados ou não supervisionados, e para análise e classificação de padrões.

2.2.1 *Artificial Neural Networks*

Redes Neurais Artificiais é um algoritmo comumente utilizado para classificação que se inspira no cérebro humano [Alp14]. Apesar da comparação com o cérebro humano, um computador possui um poder de processamento muito menor do que o de um cérebro. Um cérebro é capaz de processar 10^{11} unidades (ou neurônios) de processamento, operando todos paralelamente. Apesar da limitação, as ANNs foram desenvolvidas para tentar realizar uma gama de tarefas com o objetivo de permitir que máquinas realizem tarefas que nós humanos somos capazes, *i.e.*, detectar um objeto e reconhecê-lo, reconhecer falas, entre outras tarefas. Geralmente a estrutura de uma ANN é constituída por camada de entrada, camadas escondidas e camada de saída.

A arquitetura mais conhecida das ANNs são conhecidas como *feedforward*, por possuírem um fluxo direto unidirecional. Uma das unidades fundamentais de uma ANN é o neurônio. O neurônio é a unidade que processa a informação que irá passar por toda a rede, e ele deverá possuir três elementos básicos: um conjunto de sinapses, uma função que irá somar os sinais de entrada ponderados pelos seus respectivos pesos e uma função de ativação para limitar a amplitude de saída do neurônio [Hay09]. A Figura 2.3, representa um neurônio artificial, onde os sinais de entrada desse neurônio (x_i), se multiplicam com os seus respectivos pesos (w_i). Após esse processo, uma junção é realizada através de uma função de soma (Σ), e o resultado dessa função (υ) é passado para a função de ativação (φ) juntamente com o *bias*. A Equação 2.1 apresenta a soma realizada dentro do neurônio antes de entrar na função de ativação. O *bias*, nesse caso, possui o efeito de aumentar ou diminuir a entrada da rede na função de ativação, permitindo que a rede se adapte melhor aos dados passados por ela [Hay09]. Por fim, o resultado da função de ativação será a saída do neurônio.

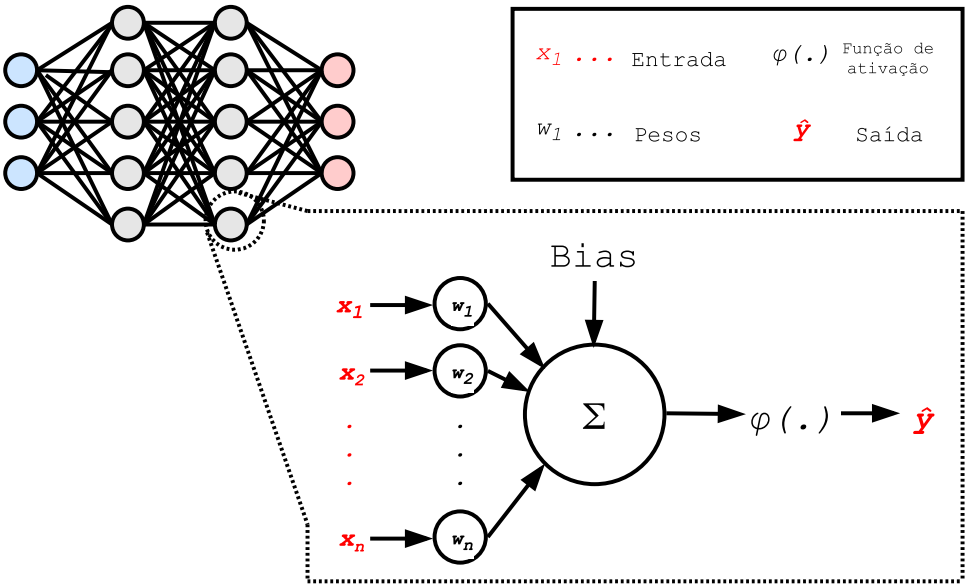


Figura 2.3 – Rede *Feedforward* com ênfase na representação de um neurônio artificial.

$$v = \sum_{i=1}^{x} x_i w_i \tag{2.1}$$

Haykin [Hay09] apresenta dois tipos de função de ativação: a função limiar e a função sigmoide. A função limiar (ou sinal), apresentada na Equação 2.2, foi utilizada no trabalho desenvolvido por McCulloch e Pitts [MP43] em 1943. Essa função irá retornar 1 caso o resultado da função de soma (v) do neurônio retorne um valor positivo, caso o contrário a função limiar irá retornar 0. A Figura 2.4 (a) apresenta o gráfico da função limiar.

$$\varphi(v) = \begin{cases} 1 \text{ se } v \geq 0 \\ 0 \text{ se } v < 0 \end{cases} \tag{2.2}$$

A função sigmoide, apresentada na Equação 2.3, possui formato de “S”, e é amplamente utilizada na construção de ANNs [Hay09]. Definida como uma função crescente ela exibe um equilíbrio entre o comportamento linear e não-linear. Apesar da sua derivada ser relativamente simples, a função sigmóide acaba saturando para valores acima de 5 e abaixo de -5, e isso acaba causando problemas na hora do treinamento da ANN, como por exemplo, zerando os gradientes, e impossibilitando a atualização correta dos pesos. Outro problema da função sigmóide é que a sua saída não é centrada em zero. Essa característica fará com que os gradientes sejam somente positivos (ou somente negativos), tendenciando a atualização dos pesos na fase de treinamento, causando um efeito “zig-zag”, e impossibilitando que a rede se aproxime do ótimo global. A Figura 2.4 (b) apresenta o gráfico da função sigmóide.

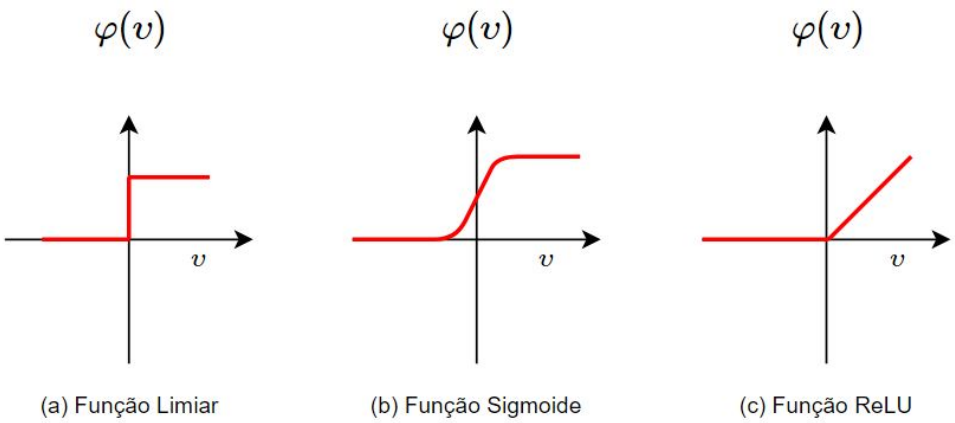


Figura 2.4 – Funções de ativação.

$$\varphi(v) = \frac{1}{1 + e^{-v}} \tag{2.3}$$

Além das duas funções de ativação mencionadas por Haykin [Hay09], a função de ativação ReLU (Unidade Linear Retificada, do inglês *Rectified Linear Unit*), proposta por Nair e Hinton [NH10], passou a ser amplamente utilizada em CNNs e outros modelos de *Deep Learning*. A função ReLU, apresentada na Equação 2.4, retorna 0 caso o valor de entrada v seja menor do que 0, e irá retornar v caso v seja maior do 0.

$$\varphi(v) = \begin{cases} 0 \text{ se } v \leq 0 \\ v \text{ se } v > 0 \end{cases} \tag{2.4}$$

Comparado com a função sigmoide, a ReLU faz com que o modelo acabe convergindo muito mais rápido, devido a sua forma linear [KSH12]. Além disso, a função ReLU comparada a função sigmóide, possui uma computação menos custosa, devido ao fato que a ReLU pode ser implementada simplesmente transformando os valores negativos de uma matriz de ativação em zero. Um ponto negativo, é que se utilizado uma taxa de aprendizado muito alta, consequentemente os gradientes podem atualizar os pesos do neurônio a ponto que esse nunca ative com uma função ReLU. Por isso ao utilizar essa função de ativação, o usuário deverá fornecer uma configuração adequada para que não ocorra a “morte” dos neurônios da rede em questão. A Figura 2.4 (c) apresenta o gráfico da função ReLU.

Perceptron e *Multi-layer Perceptron*

O Perceptron, desenvolvido por Rosenblatt [Ros58], foi a primeira arquitetura de ANN a ser descrita em forma de algoritmo e capaz de realizar aprendizado supervisionado. Apesar de ter ganhado sua devida importância, a limitação do modelo está na sua incapa-

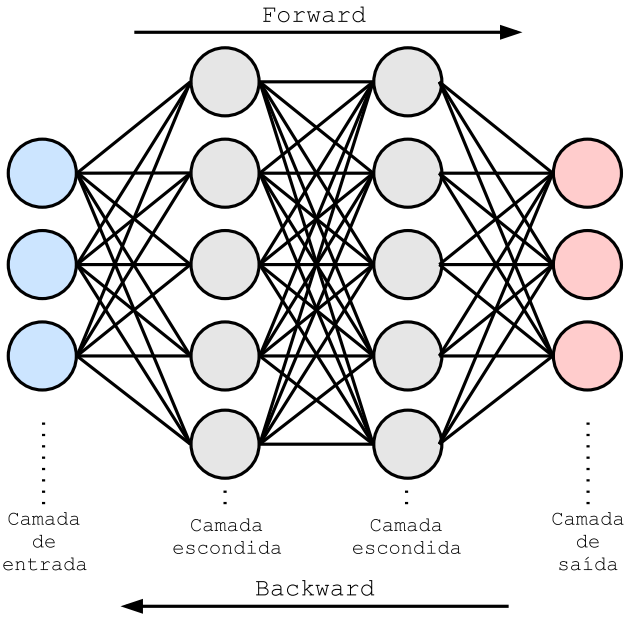


Figura 2.5 – Ilustração de uma arquitetura MLP, contendo 1 camada de entrada, 2 camadas escondidas e 1 camada de saída.

cidade de classificar dados que não são linearmente separáveis. Essa limitação, conhecida através do problema XOR, foi superada quando perceberam que ao acrescentar mais uma camada na arquitetura tornaria o modelo capaz de construir funções não lineares para a classificação dos dados. Essa nova arquitetura passou a ser conhecida como Perceptron de Múltiplas Camadas (do inglês, *Multi-layer Perceptron*, MLP) [Ros58].

Uma rede MLP, em sua estrutura mais básica, contém um camada de entrada, uma ou mais camadas escondidas e uma camada de saída. A camada inicial, não diferente de outro modelo de ANN, recebe os dados de entrada do problema, já as camadas intermediárias funcionam como extratoras de *features*, que serão responsáveis por transmitir os seus estímulos para a camada de saída, a qual irá gerar a resposta da rede. Os neurônios de cada camada (ℓ_i) deverão ser conectados com os neurônios da camada anterior (ℓ_{i-1}) e posterior (ℓ_{i+1}), com exceção as camadas ℓ_0 e ℓ_n, que não irão possuir camadas anteriores e posteriores, respectivamente. Cada camada irá produzir um vetor de saída ($y^{(\ell_i)}$), onde cada elemento desse vetor será gerado conforme apresentado na Equação 2.1. A Figura 2.5 exemplifica uma arquitetura MLP.

Treinamento

O algoritmo de treinamento da MLP é uma generalização da regra delta proposta por Widrow e Hoff [WH60] que foi utilizada para o treinamento de uma arquitetura conhecida como Adaline. O algoritmo oriundo dessa generalização foi então nomeado retropropagação do erro, ou do inglês, *backpropagation* [RHM+86]. O objetivo do algoritmo é

reduzir o erro gerado pela rede atualizando os valores dos pesos de cada neurônio em cada camada durante a fase de treinamento. O *backpropagation* consiste de duas fases: a fase de propagação da rede (*forward*) e a fase de retro-propagação do erro (*backward*).

A fase *forward* consiste na rede recebendo os dados na camada de entrada (ℓ_0) e propagando-os para as camadas posteriores (ℓ_1, ..., ℓ_n). Por fim, os valores gerados pela última camada (ℓ_n) serão comparados com os valores originais dos dados de entrada a fim de se gerar um valor escalar de custo ($J(\theta)$), onde θ representa o conjunto de pesos da rede. O algoritmo de *backpropagation* [RHM+86], permite que o valor do custo gerado volte pelas camadas calculando o gradiente (fase *backward*). Vale ressaltar que o termo *backpropagation*, muitas vezes ligado ao algoritmo, está relacionado somente ao método para computar o gradiente, enquanto o algoritmo Gradiente Descendente Estocástico (do inglês, *Stochastic Gradient Descent*, SGD) que é realmente utilizado para atualizar os pesos da rede através do gradiente previamente calculado [GBCB16]. A Equação 2.5 apresenta como a atualização dos pesos da rede (θ) é feita pelo algoritmo SGD, onde θ é o conjunto de pesos da rede, $J(\theta; x^{(i)}, y^{(i)})$ é o custo calculado através dos pesos da rede juntamente com os dados de treinamento e o seus respectivos rótulos $x^{(i)}$ e $y^{(i)}$. Por fim, ∇_θ é a derivada do custo gerado e α é a taxa de aprendizado adotada para o treinamento da rede. A taxa de aprendizado (ou do inglês, *Learning Rate*, LR) é um valor geralmente adotado entre 0 e 1, que irá determinar o impacto da atualização dos pesos. A LR é um hiper-parâmetro da rede e fica a critério do usuário determinar qual seja o seu valor. Caso seja adotado valores pequenos para a LR, a rede poderá demorar muito tempo para aprender, ou até mesmo ficar presa em um mínimo local.

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)}) \tag{2.5}$$

A atualização dos pesos da rede tem como objetivo minimizar o custo $J(\theta)$ calculado, aproximando a função que a rede gera ($\hat{f}$) da função ideal (f). Como padrão amplamente adotado pela comunidade, a ação de passar todos os dados de treino pela rede denomina-se “época”, e uma rede pode ser treinada por um determinado número de épocas, ou até que ela atinja um critério de parada previamente determinado.

Regularização

Regularização pode ser entendido como qualquer modificação feita em um algoritmo de aprendizado com a intenção de reduzir o *overfitting* [GBCB16]. O *overfitting* é um problema comum em redes *deep* ao se utilizar *datasets* pequenos. Diferentes métodos de regularização foram criados na tentativa de minimizar esse problema, por exemplo, o método de *dropout* [SHK+14]. É um método que zera os pesos de alguns neurônios com uma certa probabilidade *p*, *e.g.*, uma taxa de *dropout* de 0.5 significa que cada neurônio da rede possui 50% de chance de ter o seu peso zerado na iteração atual. Essa técnica é aplicada

somente durante o treinamento da rede neural, e ela faz com que os neurônios aprendam diferentes características, tornando-os capazes de generalizar melhor. Na prática, essa técnica cria a possibilidade de criar um *ensemble* de modelos, possibilitando otimizar o modelo atual, e encontrar a melhorar configuração possível para resolver o problema em questão.

2.2.2 *Convolutional Neural Networks*

Redes Neurais Convolucionais, ou simplesmente CNNs, são um tipo de rede neural *feedforward* que vem se destacando na área de visão computacional com tarefas como classificação de imagens, classificação de ações, reconhecimento e detecção de objetos, dentre outras [KSH12, GDDM14, LSD15]. Bengio [Ben09] explica que os métodos de *Deep Learning* visam aprender hierarquias de *features* de níveis mais altos formados pela composição de *features* de níveis mais baixos. Não obstante, as CNNs usam essas hierarquias de *features* para aprender diferentes representações de imagens, vídeos e áudio, com ou sem ruído [LKF+10]. As CNNs apresentam algumas características particulares como mapas de características locais (do inglês, *feature maps*), pesos compartilhados e, às vezes, sub-amostragem espacial ou temporal.

A Figura 2.6 mostra a estrutura de uma rede neural multicamada, onde o aprendizado profundo quebra o mapeamento complicado desejado da entrada (camada visível) para a saída (classe do objeto) em uma série de mapeamentos simples aninhados, cada um descrito por uma camada diferente do modelo. A camada de entrada contém as variáveis que podemos observar, enquanto as camadas ocultas extraem *features* cada vez mais abstratos da imagem. Assim, dada a imagem de entrada, a primeira camada pode facilmente identificar bordas. Dada a primeira descrição da camada oculta das bordas, a segunda camada oculta pode procurar com precisão cantos e contornos estendidos. Através da descrição da segunda camada oculta da imagem em termos de cantos e contornos, a terceira camada oculta pode detectar partes inteiras de objetos específicos. Finalmente, a descrição da imagem em termos dos pedaços de objeto que ela contém pode ser usada para reconhecer os objetos presentes na imagem [GBCB16].

Uma estrutura de CNN típica para classificação de imagem contém camada de entrada, camadas de convolução e *pooling*, camadas totalmente conectadas, e uma camada de saída. Um exemplo de uma arquitetura de CNN pode ser vista na Figura 2.7.

Camada Convolucional

Precisamente a operação de convolução, em sua real definição, não ocorre nas redes convolucionais, mas sim uma modificação de sua real operação é implementada [GBCB16]. A operação de convolução, no aspecto de CNNs, trata de receber um volume de entrada,

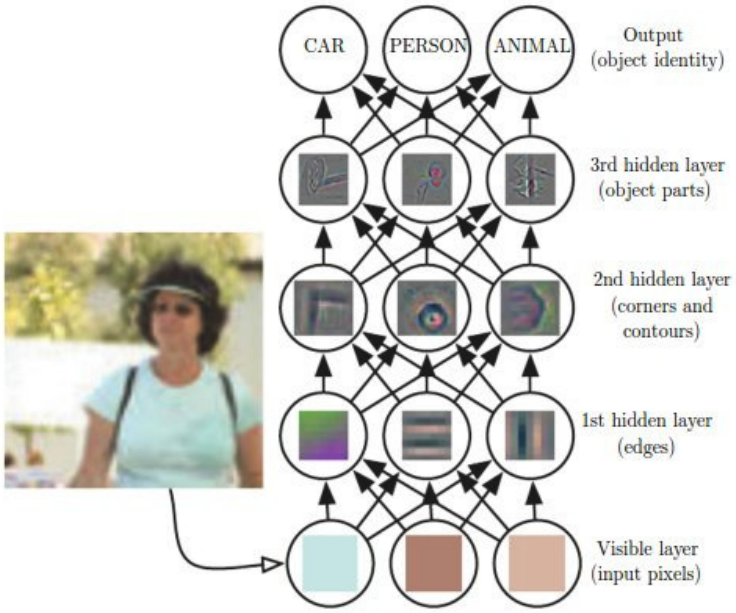


Figura 2.6 – Por uma questão de simplicidade, a figura mostra uma rede neural totalmente conectada que atua como uma rede neural convolucional. Dentro de cada nó (neurônio) é mostrado o conhecimento específico aprendido em relação ao dado de entrada. (imagem adaptada de [GBCB16])

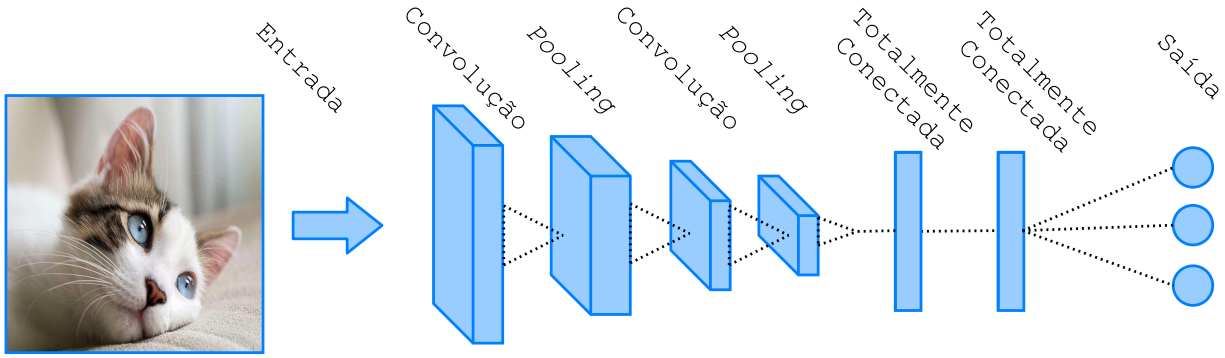


Figura 2.7 – Arquitetura de uma CNN contendo camadas de convolução, *pooling* e totalmente conectadas.

aplicar um filtro (*kernel*) composto por pesos treináveis sobre esse volume, gerando um mapa de características (do inglês, *feature map*). A ideia é que através dos ciclos de treinamento os pesos compostos em um filtro se ajustem a ponto de serem capazes de extrair informações relevantes dos dados. Com isso, o processo de convolução realiza o produto interno dos pesos do filtro com os dados de entrada passado pela função. Após o produto do interno desses pesos com o dado de entrada, deverá ser somado um valor de *bias* para gerar o resultado que irá compor o *feature map*. Uma das características relevantes da camada de convolução é a sua capacidade de compartilhar pesos. Um filtro irá utilizar dos mesmos pesos passando por todo os dados de entrada. Essa característica faz com que a rede tenha uma drástica redução de parâmetros que necessitariam ser aprendidos,

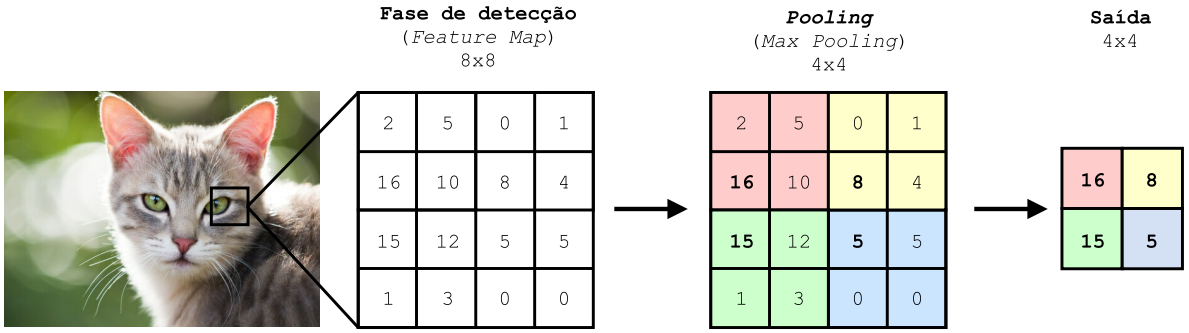


Figura 2.8 – Fluxo padrão em uma CNN, onde os dados passam pela fase de detecção e pelo processo de pooling.

consequentemente diminuindo o tempo de treinamento. Além disso, o compartilhamento de pesos tornam os filtros capazes de realizar uma melhor generalização, criando melhores padrões para os dados de entrada. Em suma, uma camada de convolução poderá ter *n* filtros, contendo cada um deles pesos aprendíveis, que irão deslizar sobre os dados passados para a camada de convolução. O tamanho do deslocamento do filtro de convolução sobre o dado de entrada denomina-se *stride*. Vale ressaltar que os pesos dos filtros de uma camada de convolução são aprendidos através do algoritmo de *backpropagation*.

Pooling

Após a camada de convolução, os valores gerados irão passar por uma função de ativação, como por exemplo a função ReLU [NH10]. Essas duas etapas, convolução e ativação, fazem parte da **fase de detecção** [GBCB16]. Após a fase de detecção, os dados são passados para uma camada de *pooling*. A camada de *pooling* é responsável por simplificar estatisticamente a saída dos neurônios anteriores a ela. Por simplificar, entende-se que a operação de *pooling* tem a capacidade de reduzir a dimensionalidade dos dados que irão passar ela, bem como evitar ruídos. O *Max Pooling* [ZCVJ88] é um exemplo de operação de *pooling*, onde busca-se selecionar somente os valores mais altos de cada filtro que será aplicado. A Figura 2.8 ilustra o processo desde a fase de detecção até a operação de *pooling* utilizando o *Max Pooling*, onde 4 filtros de tamanho 4×4 são aplicados sobre uma entrada, *feature map*, de tamanho 8×8. O resultado dessa operação é uma saída de tamanho 4×4 com os valores mais elevados de cada *feature map*.

Em suma, a operação de *pooling* irá simplificar os valores passados por ela encontrando uma representação estatística. Essa simplificação diminui o tempo de treinamento da rede, bem como torna a rede capaz de selecionar as *features* mais relevantes de cada amostragem. Existem diferentes tipos de operações de *pooling* na literatura e cada um deles aplica-se a uma determinada ocasião.

2.2.3 *Long-Term Recurrent Convolutional Networks*

A rede *Long Short-Term Memory* (LSTM), proposta por Hochreiter e Schmidhuber [HS97], é um tipo de rede recorrente (do inglês, *Recurrent Neural Network*, RNN) que é capaz de aprender dependências de longo prazo. A principal diferença entre uma rede *feedforward* clássica e uma RNN, é que as RNNs possuem camadas de *feedback* para as camadas anteriores. Essas camadas de *feedback* permitem que as RNNs guardem informações dos dados anteriores e do estado atual que ela computa. Donahue *et al.* [DHR+17] propõem uma arquitetura que funde a ideia de uma rede CNN com uma LSTM conhecida como *Long-Term Recurrent Convolutional Networks*(LRCNs). Com o objetivo de classificar ações, a ideia principal é utilizar uma CNN para extrair *features* de imagens e passar essas *features* para uma LSTM, a qual possui a capacidade de guardar informações através do tempo, para que consiga-se classificar qual é a devida ação relacionada as imagens passadas no início do processo. Ao guardar informações de um conjunto de imagens, estamos levando em conta o aspecto temporal do problema e com isso acrescentando *features* que antes, nos modelos estáticos, não éramos capazes de computar. Por tanto, após a CNN ser capaz de extrair *features* relevantes para a LSTM, a classificação final da ação é feita computando a moda das classes gerada pela LSTM para cada imagem. Além disso, devido a flexibilidade de modelos recorrentes e as determinadas tarefas que podem ser abordadas por esses, o estilo da arquitetura proposta por Donahue *et al.* [DHR+17] não só é capaz de realizar reconhecimento de ações, bem como é capaz de descrever imagens e até mesmo descrever vídeos. A Figura 2.9 exemplifica a arquitetura de uma LRCN.

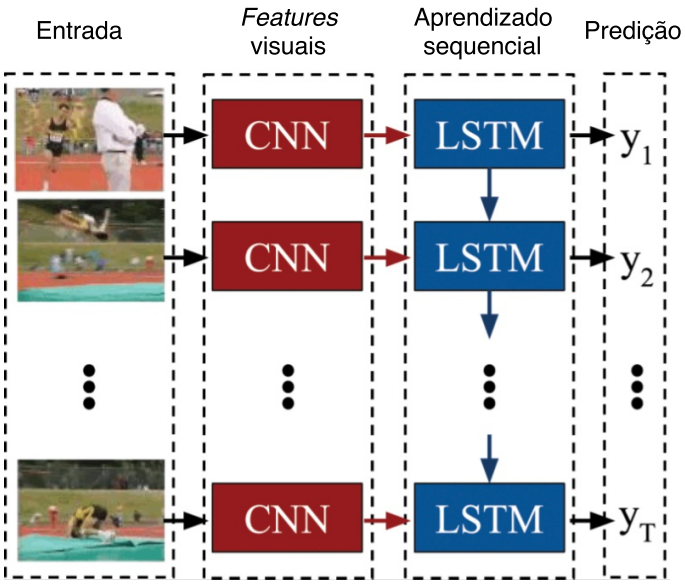


Figura 2.9 – Arquitetura LRCN para classificação de ações. A predição do modelo será computada através da moda das predições y_1 a y_T (imagem adaptada de [DHR+17]).

2.3 Resumo

Neste capítulo apresentamos um resumo dos conhecimentos necessários para entender melhor as técnicas e abordagens que foram utilizados nesse trabalho. Destacamos o assunto de *Deep Learning* bem como seus modelos que estão presentes na maior parte dessa dissertação. As CNNs foram aplicadas em ampla escala para todos os *datasets* escolhidos neste trabalho, por serem um técnica relativamente fácil de implementar e que possui ótimos resultados já publicados na literatura. No próximo capítulo iremos apresentar os trabalhos relacionados realizando uma síntese dos assuntos mais relevantes a esta pesquisa.

3. TRABALHOS RELACIONADOS

O reconhecimento de ações baseado em vídeos vem ganhando importância pois existe uma necessidade de mais assistência e vigilância para algumas tarefas humanas. De acordo com Poppe [Pop10], o reconhecimento de ação humana baseado em imagens de vídeos pode ser considerado como uma combinação de extração de características (*features*) e subsequente classificação das *features* dessas imagens. Em 1973, Johansson [Joh73] propôs um experimento chamado Exibição de Luz em Movimento (do inglês, *Moving Light Display*, MLD). O experimento MLD demonstrou que os seres humanos podem perceber os padrões de fontes de luz pontuais colocados em algumas articulações de membros, sem quaisquer outras informações adicionais, *i.e.*, outros fatores não relacionados, como o tipo de roupa, iluminação e fundo não interfeririam na tarefa de reconhecimento. Desde então, muita atenção foi dada sobre como modelar a ação humana com *features* simples para que os sistemas possam reconhecer automaticamente a atividade que está sendo realizada. Neste trabalho classificamos as contribuições para o reconhecimento de ações em dois grupos: (a) trabalhos baseados em *Handcrated Features* (Seção 3.1), que descrevem como os pontos de interesses mudam ao longo do tempo e (b) trabalhos com base no uso de *Deep Learning* (Seção 3.2), que usa um conjunto de métodos que permitem que o algoritmo descubra automaticamente as *features* a partir de dados brutos (dados os quais não foram processados previamente).

3.1 Abordagens baseadas em *Handcrafted Features*

De acordo com Turaga *et al.* [TCSU08], um sistema de reconhecimento de ações possui várias etapas com o objetivo de processar uma sequência de imagens até obter uma interpretação de alto nível desta sequência. As etapas incluem: a extração de *features* de uma sequência de imagens passadas como entradas, *features* de baixo nível e interpretação semântica de alto nível das ações primitivas. O aspecto temporal para o reconhecimento da ação também é importante para uma melhor compreensão da ação que está sendo realizada. Normalmente, a representação do tempo nas imagens leva em consideração as *features* extraídas dos quadros em uma sequência de imagens e se enquadra em duas categorias: Representações Globais e Representações Locais [Pop10, HHP16].

3.1.1 Representações Globais

Poppe [Pop10] explica que as representações globais codificam a região de interesse de uma pessoa como um todo e geralmente são obtidas através de subtração do fundo da imagem ou pelo rastreamento da pessoa em uma sequência de imagens. As representações globais comuns são derivadas de silhuetas, bordas ou fluxo óptico, que normalmente é um pouco sensível a diferentes pontos de vista e do fundo da imagem. Essas representações geralmente funcionam bem quando o domínio permite um bom controle de ruído, oclusões, subtração de fundo e rastreamento. O trabalho de Yamato *et al.* [YOI92] usa silhuetas em uma abordagem com Modelos Ocultos de Markov (do inglês, *Hidden Markov Models*, HMM) [HAJ90] para identificar uma das seis ações em uma partida de tênis: *forehand stroke*, *backhand stroke*, *forehand volley*, *backhand volley*, *smash* e *service*. A abordagem extrai recursos de baixo nível de imagens de silhueta quantificadas em *superpixels* [RM03], *i.e.*, segmentos contendo grupos de pixeis vizinhos, cada pixel contando a proporção de pixeis em preto e branco dentro de sua região subjacente.

Um dos primeiros esforços que empregam a ideia de usar a silhueta para extrair os recursos baseados em movimento foi apresentado por Bobick e Davis [BD01]. Eles primeiro constroem uma imagem de energia de movimento binário (do inglês, *Motion-Energy Image*, MEI) que representa onde o movimento ocorreu em uma sequência de imagens e em seguida eles geram uma Imagem de Histórico de Movimento (do inglês, *Motion-History Image*, MHI), que é uma imagem de valor escalar em que a intensidade é uma função em relação ao movimento. O primeiro método atribui peso igual a todas as imagens na sequência, gerando uma imagem onde o movimento ocorreu. O segundo método especifica pesos às imagens na sequência, produzindo maior peso para quadros novos e baixo peso para quadros mais antigos. Juntos, MEI e MHI podem ser entendidos como uma versão de um modelo temporal, *i.e.*, uma imagem em que cada componente de cada pixel é alguma função do movimento naquele local de pixel. A Figura 3.1 mostra a composição da imagem MHI (direita) com base no movimento realizado até a última imagem (esquerda) e a imagem MEI (centro). Observe que a parte mais branca da imagem MHI corresponde ao movimento do último *frame* (esquerda) e a mais escura corresponde aos quadros anteriores. O reconhecimento da ação é realizado usando a técnica de extração de *features* proposta por Hu [Hu62]. Bobick e Davis [BD01] mostram que MEI e MHI funcionam para identificar classes de ação simples, como sentar, curvar e agachar. No entanto, quando as ações se tornam mais complexas, MEI e MHI tendem a substituir o histórico de movimentos e, portanto, se torna difícil identificar a ação correta.

Weinland *et al.* [WRB06] expande a ideia de MHI e propõe o Volume do Histórico de Movimentos (do inglês, *Motion History Volume*, MHV) como a versão 3D do MHI. O MHV combina silhuetas de várias câmeras em um modelo 3D. A Figura 3.2 mostra as

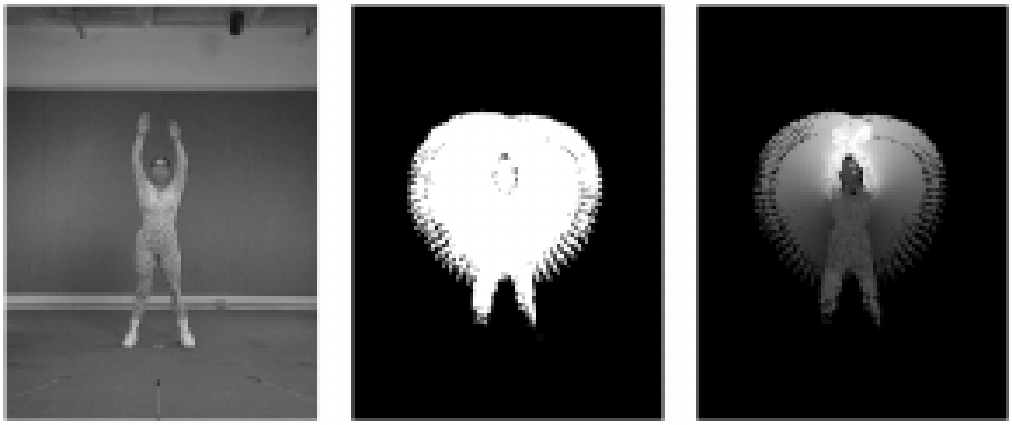


Figura 3.1 – Exemplos da imagem original seguida por sua representação no formato MEI e MHI (imagem adaptada de [BD01]).

ações sentar, caminhar, chutar e socar respectivamente, todas geradas pelo método MHV. A desvantagem deste método é que requer uma calibração precisa da câmera, uma vez que a combinação é realizada alinhando os volumes usando Transformadas de Fourier no sistema de coordenadas cilíndricas em torno do eixo medial.

Wang e Suter [WS06] usam uma sequência associada de silhuetas humanas e descritores de contorno derivados de vídeos em Média de Energia Motora (do inglês, *Average Motion Energy*, AME) e em Média de Forma de Movimento (do inglês, *Mean Motion Shape*, MMS), para caracterizar ações. A AME representa uma sequência de movimento humano em uma imagem de nível de cinza, preservando algumas informações temporais, gerando uma silhueta média. Esta silhueta é gerada calculando a intensidade média em todos os quadros centrados. De forma semelhante a AME, a MMS gera a forma média pela média dos contornos centrados de todos os quadros. O limite ou a forma podem ser facilmente obtidos a partir da silhueta binária de conectividade única usando um algoritmo de seguimento de borda. A Figura 3.3 apresenta as *features* AME e MMS para dez ações diferentes do *dataset* de ações humanas Weizmann [BGS+05]. As ações nunca vistas são

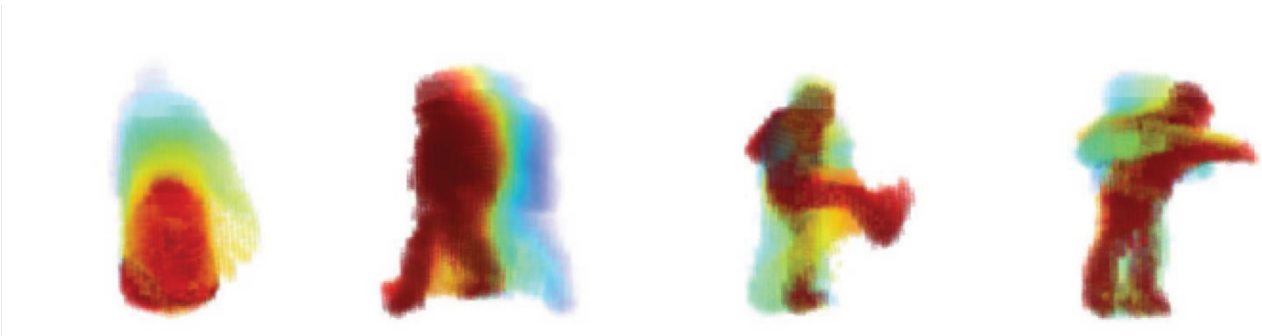


Figura 3.2 – Exemplos de imagens geradas por MHV. Da esquerda para direita: sentar, caminhar, chutar e socar. Os valores de cor codificam o tempo da última imagem do volume (imagem adaptada de [WRB06]).

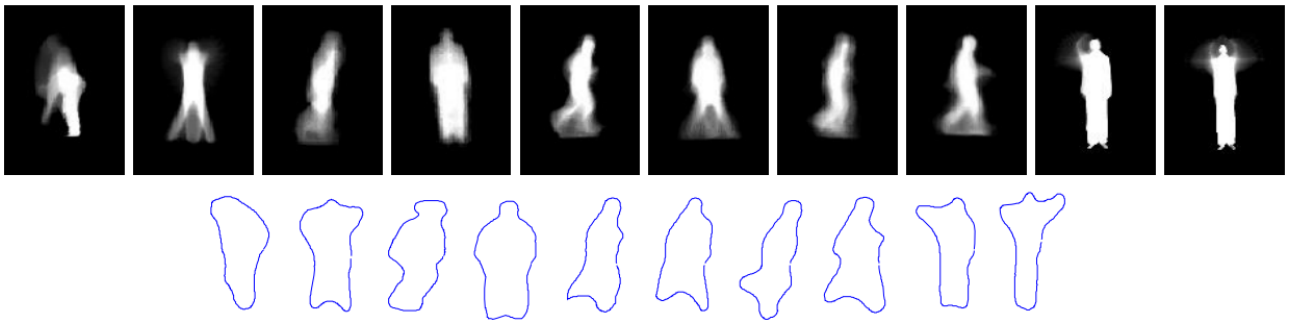


Figura 3.3 – *Features* AME e MMS para dez diferentes ações (imagem adaptada de [WS06]).

classificadas de acordo com a proximidade com a média AME e a MMS derivadas de múltiplas sequências com a mesma ação.

Em vez de usar silhuetas para a identificação de movimentos, os dados bidimensionais (*e.g.*, formas e aparências) de imagens podem ser usados para projetar uma cena 3D formando volumes com uma modificação temporal. Como um vídeo é uma sequência de imagens 2D colocadas em ordem cronológica, uma atividade pode ser representada como um volume espaço-temporal 3D específico construído pela concatenação de imagens 2D ao longo do tempo. Uma abordagem típica espaço-temporal para reconhecimento de atividade humana constrói um modelo 3D (volume espaço-temporal) representando cada atividade. Quando um vídeo não rotulado é fornecido, o sistema constrói um volume espacial 3D correspondente ao novo vídeo. Este novo volume 3D é comparado com cada modelo de atividade para medir a semelhança na forma e aparência entre os dois volumes. O sistema, então, finalmente deduz que o novo vídeo corresponde à atividade que tem a maior semelhança [AR11].

A ideia dos volumes do espaço-tempo é aplicada por Gorelick *et al.* [GBS+07] que usa partes subtraídas do fundo da imagem em vez de contornos e as empilham para um volume binário de espaço-tempo, como ilustrado na Figura 3.4. A principal ideia é identificar uma ação da representação da forma 3D induzida pelo movimento de suas silhuetas no espaço-tempo. Essa representação pode ser interpretada como uma extensão dos modelos MEI criados por Bobick e Davis [BD01].

Ke *et al.* [KSH07] usam uma coleção de sub-volumes de forma arbitrária onde cada sub-volume é uma região espacialmente coerente. Sub-volumes são obtidos por uma técnica de *cluster* não-supervisionada para segmentar o vídeo em volumes tridimensionais que são internamente consistentes na aparência. Esses subvolumes são chamados de *supervoxels* devido à sua semelhança com os *superpixels* [RM03]. Enquanto os *superpixels* são segmentos contendo grupos de pixels vizinhos, o *supervoxel* pode ser entendido como a versão volumétrica de *superpixels*. Um padrão de ação é gerado entre as regiões volumétricas geradas por *supervoxels* e encontrando o conjunto mínimo de regiões que maximizam a sobreposição entre sua união e o padrão. Embora esta abordagem seja um

Figura 3.4 – Volumes espaço-temporais criados pelo empilhamento de imagens em primeiro plano (adaptado de [GBS+07]).

pouco sensível ao fundo da imagem, ela é mais robusta ao ruído e às oclusões. A ideia é encontrar os segmentos de volume do ator automaticamente e medir sua semelhança com o padrão das ações. O reconhecimento é realizado pela busca de um subconjunto de volumes espaço-temporais segmentados que melhor corresponda à forma do padrão de uma das possíveis ações.

3.1.2 Representações locais

Poppe [Pop10] explica que as representações locais descrevem a observação como uma coleção de pacotes independentes, sendo menos sensíveis ao ruído e a oclusão parcial, e não exigem rigorosamente subtração ou rastreamento de fundo da imagem. No entanto, como elas dependem da extração de uma quantidade suficiente de pontos de interesse relevantes, o pré-processamento às vezes é necessário. Em outras palavras, os pontos de interesse são locais no espaço-tempo em que mudanças bruscas de movimento ocorrem em uma sequência de quadros. Geralmente, os pontos que sofrem um movimento de translação no tempo não resultarão na geração de pontos de interesse. Os pontos de interesse são importantes, uma vez que são mais informativos para a identificação de uma ação humana. Eles foram particularmente populares devido à sua confiabilidade em relação a ruídos, *i.e.*, mudanças de iluminação, movimentos de fundo.

Laptev e Lindeberg [LL03] representam padrões de movimento usando Pontos de Interesse Espaço-Temporais (do inglês, *Spatio-Temporal Interest Points*, STIP) como uma extensão 3D do conhecido detector de pontos de interesse 2D [HS88]. O detector de pontos de interesse 2D depende da ideia de que a intensidade da imagem mudará em grande parte em múltiplas direções, permitindo identificar pontos característicos (pontos de interesse) para fazer a ligação entre as imagens. O STIP é considerado como eventos primitivos correspondentes às estruturas de imagem bidimensionais em movimento não constante. A Figura 3.5 apresenta a correspondência entre o ponto usando STIP no mo-

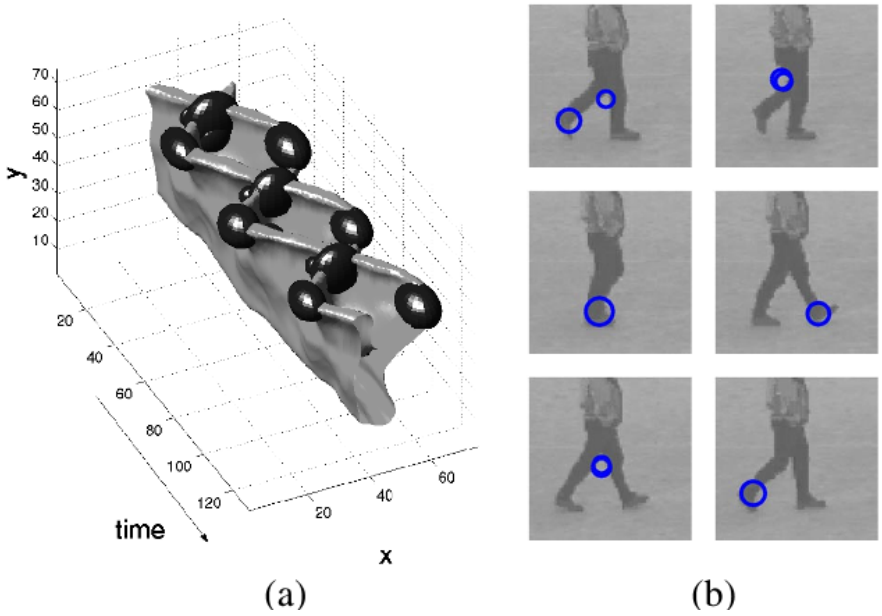


Figura 3.5 – Pontos de interesse espaço-temporais do movimento das pernas de uma pessoa caminhando: (a) representação espaço-temporal 3D do movimento de uma perna em movimento; (b) Pontos de interesse representados em uma sequência de *frames* (imagem adaptada de [Lap05]).

vimento das pernas de uma pessoa a pé. Enquanto o detector de pontos de interesse 2D Harris encontra locais espaciais em uma imagem com mudanças significativas em duas direções ortogonais, o STIP identifica pontos com grandes variações espaciais e movimentos não constantes. Uma desvantagem deste método é que ele produz um pequeno número de pontos de interesse estáveis que muitas vezes não são suficientes para caracterizar sequências complexas. Schuldt *et al.* [SLC04] aplica um classificador SVM em *features* geradas pelo método STIP no *dataset* KTH[1] [SLC04], mostrando sua aplicabilidade ao reconhecer ações. Essa ideia de extrair pontos interesses locais esparsos de um volume espaço-temporal 3D foi adotada por vários pesquisadores, mostrando que, como sugeriram Laptev e Lindeberg [LL03], características locais escassas que caracterizam o movimento local são suficientes para representar ações.

Usando a Relação Espaço-Temporal (do inglês, *Spatio-Temporal Relationship*, STR) Ryoo e Aggarwal [RA09] apresentam a STR, que considera as relações espaciais e temporais entre as *features* detectadas, medindo a semelhança estrutural entre conjuntos de *features* extraídas de vídeos. As atividades complexas estruturadas são detectadas ao computar as relações espaço-temporais em pares entre as características locais. O método pode classificar ações simples, *i.e.*, ações executadas por um único indivíduo com fundo estático, bem como reconhecer atividades de nível de interação, *i.e.*, atividades realizadas por duas ou mais pessoas, como aperto de mãos ou chutar.

[1] http://www.nada.kth.se/cvap/actions/

Fiaz e Ijaz [FI10] propõem um método que usa o padrão de silhueta de regiões de imagens do humano extraído da segmentação de uma cena. Por fim, as *features* geradas são passadas para uma rede neural artificial para identificar a ação humana. Este método transforma as imagens bidimensionais binárias contendo regiões de imagens do humano em vetores de uma única dimensão que representam o padrão de silhueta na imagem. Outros vetores são gerados representando o padrão de movimento humano. Um perceptron de três camadas é treinado usando estes vetores para a finalidade da classificação de diferentes ações. O conjunto de treinamento é composto por 2900 imagens, extraídas dos vídeos de amostra, com uma média de 480 amostras para cada classe. As amostras de vídeo contêm atividades humanas simples que incluem: permanecer em pé, sentar, curvar, caminhar e correr. Essas atividades representam as classes da rede neural mais a classe "nenhuma ação" onde nenhuma atividade está sendo executada.

Bansal *et al.* [BKGG13] realiza o reconhecimento da ações da vida cotidiana com base nos movimentos das mãos e no uso do objeto que o ator do vídeo utiliza. O método proposto primeiro detecta as regiões da mão através da segmentação de cores e da identificação da pele. Como alguns objetos podem ter a mesma cor da pele, a subtração de fundo é realizada eliminando objetos imóveis com cor de pele. Uma vez que os objetos podem dar sugestões da atividade (*e.g.*., o uso da faca pode indicar a atividade de corte), os objetos são identificados como "não está em uso" e "em uso". Um modelo híbrido de SVM e HMM dinâmico combina a informação estrutural e temporal para inferir a atividade em conjunto. Todos os experimentos foram realizados utilizando o *dataset* de ação na cozinha KSCGR [SKD+13].

3.2 Abordagens Baseadas em *Deep Learning*

Recentemente técnicas de *Deep Learning* permitiram grandes avanços na área de visão computacional [KSH12]. *Deep Learning* para a classificação de imagens começou em 1990 com o trabalho de LeCun *et al.* [LCDH+90], onde foi apresentado a primeira ideia de redes neurais convolucionais. Desde a evolução das placas gráficas as CNNs têm sido foco de cientistas da visão computacional, os quais conseguiram aperfeiçoar os resultados em diversas tarefas como classificação, detecção, segmentação, entre outras. As técnicas atuais podem classificar ações em esportes [KTS+14], ações do dia-a-dia (atender o telefone, comer, escrever, *etc.*) [JXYY13] e até mesmo possuir um resultado de classificação elevado para *datasets* grandes, *i.e.*, ImageNet o qual possui mais de 15 milhões de imagens com mais de 1000 diferentes classes [RDS+15]. Mesmo com os avanços na área, o reconhecimento de ações ainda é desafiador usando CNNs devido ao fato de que as ações não têm apenas a dimensão espacial, mas também a temporal. Para preencher essa la-

cuna, cientistas da área buscam desenvolver novas abordagens que utilizam *hand-crafted features* no aspecto temporal em uma única rede neural convolucional.

Ji *et al.* [JXYY13] criam uma nova estrutura para CNNs, chamada 3D-CNN (Seção 2.2.2), que visa reconhecer ações humanas utilizando o aspecto temporal. A arquitetura foi testada com o *dataset* TRECVID 2008 [RFO+09] e os resultados foram comparados com uma 2D-CNN. Os autores também comparam a 3D-CNN com outras abordagens do autor, no entanto, usando imagens do *dataset* KTH [SLC04]. Os autores obtiverem resultados superiores quando comparados com outras abordagens de classificação de ações para o mesmo *dataset*. A Figura 3.6 ilustra um exemplo da arquitetura 3D proposta pelos autores.

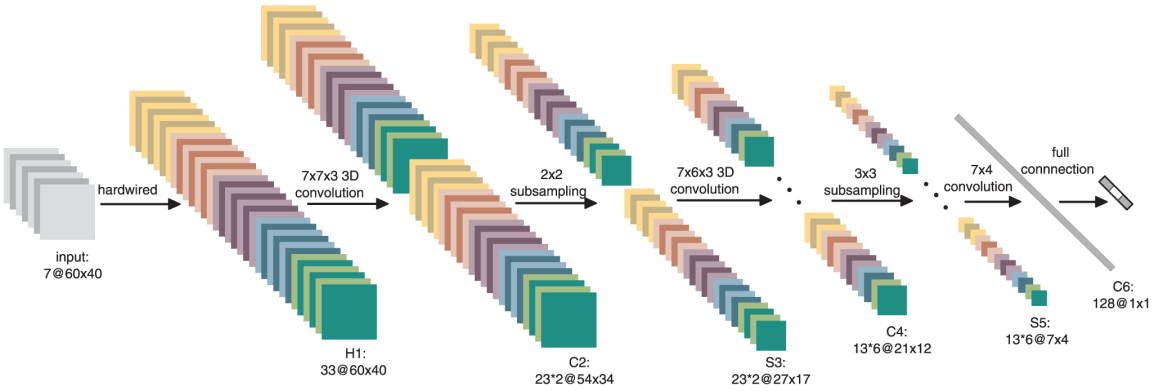


Figura 3.6 – Arquitetura 3D-CNN contendo camadas de convolução, *pooling* e camadas totalmente conectadas (Imagem adaptada de Ji *et al.* [JXYY13]).

Simonyan e Zisserman [SZ14a] propõem uma arquitetura com dois canais, conhecida como Two-Stream, a qual incorpora redes espaciais e temporais. Os autores afirmam que, à medida que os vídeos podem ser decompostos em componentes espaciais e temporais, a abordagem Two-Stream pode ser aplicada [GM92]. Esta hipótese afirma que o córtex visual humano contém o fluxo ventral (responsável pelo reconhecimento de objetos) e o fluxo dorsal (responsável pelo reconhecimento do movimento) como duas vias separadas. Assim, a arquitetura apresentada contém duas CNNs: a espacial, que transporta informações sobre cenas e objetos, e a temporal, que transmite o movimento através dos quadros, indicando o movimento do observador e dos objetos. A Figura 3.7 ilustra a Two-Stream CNN, contendo o fluxo espacial e temporal.

Embora a aparência estática usada na CNN espacial seja uma pista útil uma vez que algumas ações estão fortemente associadas a objetos particulares, as imagens individuais podem ser ambíguas, e aspectos de movimento tornam-se necessários. A ideia principal de usar a CNN temporal é que ela irá fornecer *features* adicionais para o reconhecimento de ação, uma vez que várias atividades são baseadas na informação de movimento. O fluxo espacial é treinado usando quadros de vídeo estável, enquanto o fluxo temporal é treinado usando o *dense optical flow* [Gol75] gerado pela sequência de imagens.

O *optical flow* pode ser visto como um conjunto de campos de vetor de deslocamento entre quadros consecutivos e representa a interpretação do campo de movimento

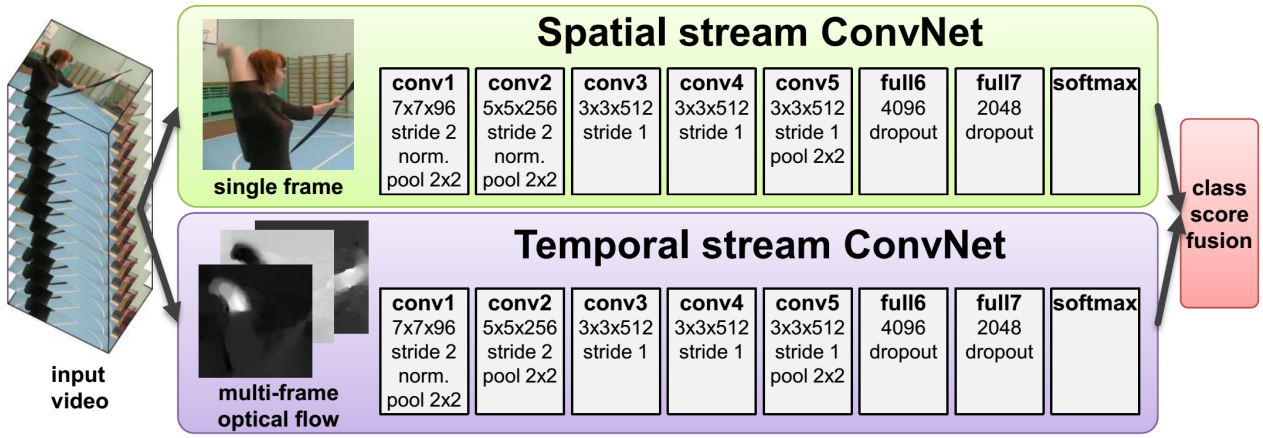


Figura 3.7 – Two-stream CNN contendo o fluxo espacial (cor verde) e temporal (cor roxa).

com uma intensidade de imagem variável no tempo. A versão densa (DOF) representa como e onde cada pixel na sequência da imagem está em movimento. Cada fluxo executa reconhecimento de vídeo por conta própria, onde por fim suas caracteristiscas serão concatenadas e então passadas a uma SVM linear para a classificação final. Os experimentos realizados utilizaram os *datasets* UCF-101 [SZS12] e HMDB51 [KJG+11].

Karpathy *et al.* [KTS+14] fornecem uma avaliação empírica extensiva de abordagens múltiplas para ampliar CNNs na classificação de vídeo. Os autores fornecem uma arquitetura que processa a entrada da rede (imagens) em duas resoluções espaciais (*Fovea* e *Context Stream*). A rede foi testada com os *datasets* Sports-1M [KTS+14] e UCF-101 [SZS12]. A ideia é tratar todos os vídeos com clipes de tamanho fixo, onde cada clipe contém vários quadros contíguos no tempo, o que torna possível ampliar a conectividade da rede na dimensão do tempo para aprender *features* espaço-temporais. Existem várias opções de como estender a conectividade, os autores apresentam três categorias de padrões de conectividade (fusão inicial, fusão tardia e fusão lenta (respectivamente, do inglês, *early fusion*, *late fusion*, *slow fusion*)), como pode ser visto na Figura 3.8.

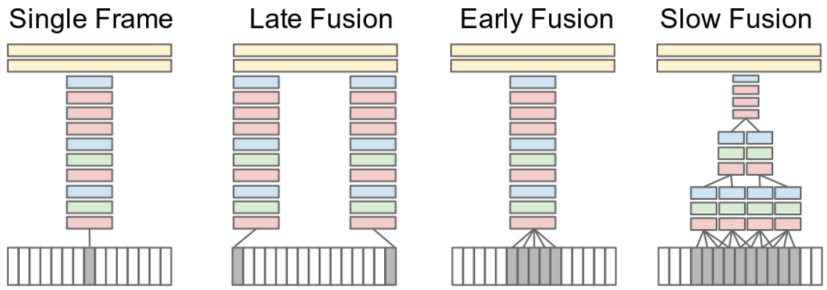


Figura 3.8 – Quatro diferentes métodos de fusão sobre a dimensão temporal através da rede proposta por Karpathy *et al.* [KTS+14]. As caixas em vermelho, verde e azul indicam respectivamente etapas de convolução, normalização e *pooling* (imagem adaptada de [KTS+14]).

Karpathy *et al.* [KTS+14] propuseram uma arquitetura CNN de multi-resolução que tem como objetivo proporcionar tempos de execução mais rápidos sem sacrificar o desem-

penho. A Figura 3.9 mostra ambos os fluxos, onde os quadros de entrada contém duas resoluções diferentes. Enquanto o fluxo *context* recebe os quadros amostrados em metade da resolução original, o fluxo *fovea* recebe a região central da imagem na resolução original. Os experimentos foram realizados utilizando os *datasets* Sports-1M [KTS+14] e UCF-101 [SZS12]. A arquitetura apenas utilizando um único quadro por vez e utilizando de fusão lenta atingiu 0,61 de precisão no *dataset* Sports-1M. Para o *dataset* UCF-101, o melhor resultado, 0,65 de precisão, foi alcançado pré-treinando a arquitetura com o *dataset* Sports-1M.

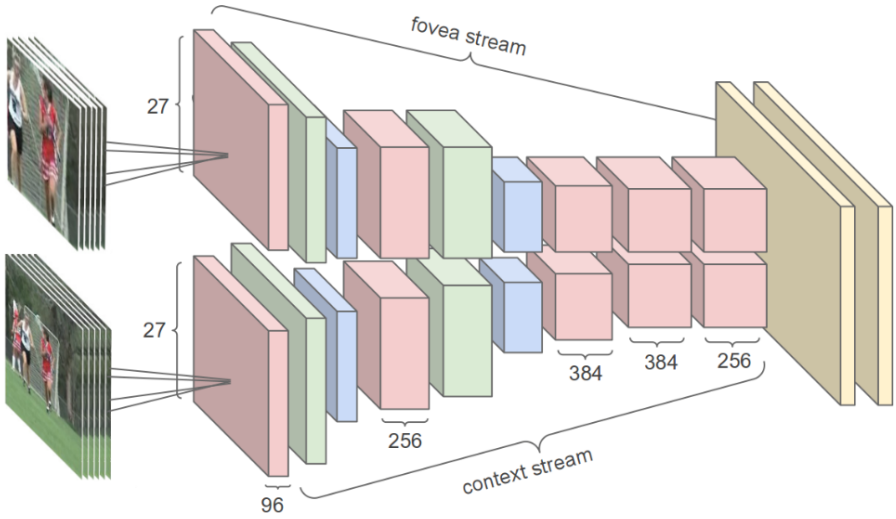


Figura 3.9 – CNN de multi-resolução apresentado por Karpathy *et al.* [KTS+14]. Imagens são passadas para a rede em dois fluxos separados, ambos os fluxos consistem de etapas de convolução (vermelho), normalização (verde) e *pooling* (azul), e por fim concatenam-se em camadas totalmente conectadas (amarelo).

Wang *et al.* [WQT15] propõem o descritor convolucional profundo de trajetória (do inglês, *Trajectory-pooled Deep-convolutional Descriptor*, TDD) que combina os benefícios de ambas as *hand-crafted features* (*i.e.*, trajetórias melhoradas [WS13]) e *features* profundamente aprendidas (*i.e.*, Two-Stream CNN [SZ14a]). Eles utilizam uma Two-Stream CNN [SZ14a] para extrair *features* dos vídeos, e após a penúltima camada totalmente conectada da CNN, ocorrem etapas de fusão e normalização com as *features* aprendidas e com as extraídas pelo algoritmo de trajetórias. Os *datasets* UCF101 [SZS12] e HMDB51 [KJG+11] foram utilizados para medir o desempenho da abordagem proposta.

Ye *et al.* [YWZ+15] propõem uma avaliação de arquiteturas do tipo Two-stream para classificação de vídeo. Os experimentos foram realizados utilizando a Two-Stream CNN [SZ14a] (chamada CNN_M) e a rede VGG_19 [SZ14b] nos *datasets* UCF-101 [SZS12] e *Columbia Consumer Videos* (CCV) [JYC+11]. Arquiteturas diferentes (CNN-M e VGG-19) e diferentes parâmetros (*dropout* e *learning rate*), bem como diferentes pontos de fusão de redes espaciais e temporais são testadas para verificar qual é a melhor configuração

para classificação de ações. Os experimentos mostraram que para o aspecto espacial, utilizando a arquitetura VGG_19, foram produzidos melhores resultados em ambos os *datasets*, indicando que as redes mais profundas funcionam melhor para problema. Para o fluxo temporal, a CNN-M obteve melhores resultados, demonstrando que as pequenas redes são melhores para serem treinadas a partir do zero com pequenos *datasets*, uma vez que o fluxo espacial pode usar algum *dataset* maior como pré-treino para ajustar a rede previamente.

Gkioxari e Malik [GM15] apresentam uma abordagem para produzir detecção consistente de ações em vídeos através do tempo, chamados de tubos de ação (do inglês, *action tubes*). Inspirado no trabalho feito por Girshick *et al.* [GDDM14], inicialmente algumas regiões são selecionadas e classificadas usando CNNs estáticas. As partes de movimento são utilizadas de duas maneiras: a amplitude de movimento, usada para eliminar regiões que não são susceptíveis de conter a ação, e para incorporar partes de movimento que irão construir modelos para detecção da ação. Por fim, inserem-se os vetores de *features* em uma SVM, a fim de gerar às classificações das ações. A Figura 3.10 ilustra a arquitetura do modelo proposto. Os *datasets* UCF Sports [RAS08] e J-HMDB [JGZ+13] foram utilizados para realizar os experimentos.

Tran *et al.* [TBF+15] propõem o aprendizado de *features* espaço-temporais utilizando uma rede convolucional 3D (C3D). Essa arquitetura é diferente da proposta por Ji *et al.* [JXYY13], uma vez que cria volumes ao realizar convoluções e *poolings*. Assim, ao invés de calcular uma convolução 2D de múltiplas imagens, que resulta em uma imagem 2D, a arquitetura proposta por Tran *et al.*executa uma convolução 3D resultando em um volume após o processo. Comparado com a 3D-CNN [JXYY13], a C3D tem a capacidade de modelar a informação temporal melhor devido às operações de convolução 3D e de *pooling* 3D.

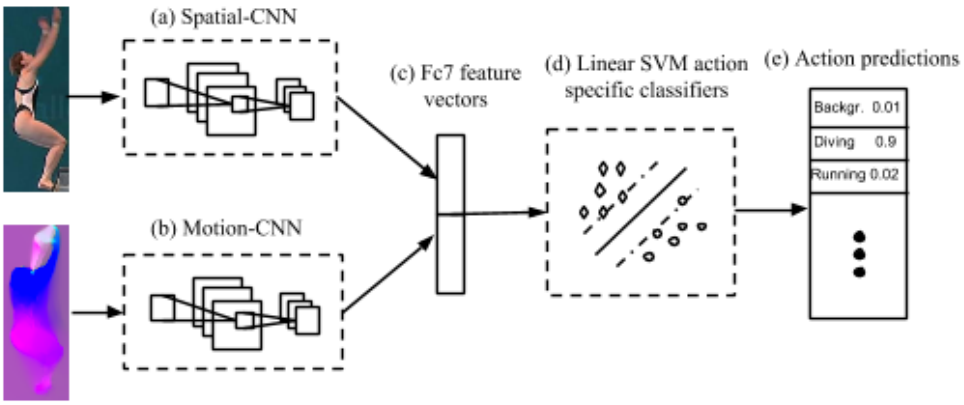


Figura 3.10 – Modelo proposto por Gkioxari and Malik [GM15] o qual utiliza um classificador SVM (d) em *features* espaço temporais. As *features* são extraídas da camada Fc7 (c) oriundas de duas CNNs, CNN espacial (a) e de uma CNN temporal (b), as quais foram treinadas para identificar as mesmas ações respectivamente (e).

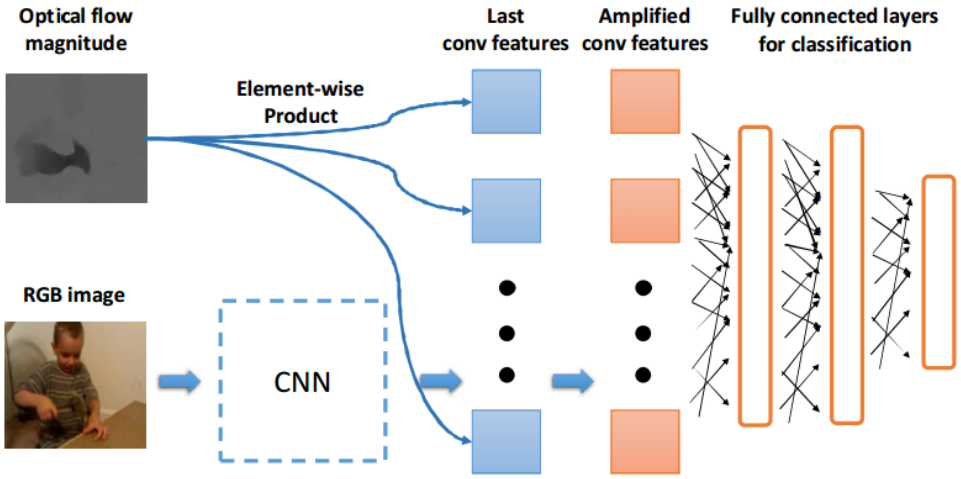


Figura 3.11 – Amplificações de *features* em uma CNN espacial utilizando *Optical Flow* (imagem adaptada de [PHBB16]).

Os experimentos foram realizados utilizando o *dataset* UCF-101 [SZS12] e a utilização de *hand-crafted features* aprimorou os resultados obtidos pela abordagem·

Park *et al.* [PHBB16] propõem uma amplificação de *features* com *hand-crafted features* (*i.e.*, OF) para variar a entrada dos mapas de *features* intermediários nas CNNs. A ideia principal é que é útil para uma rede que considera somente o aspecto espacial saber que existe algo em movimento, da mesma forma que uma rede temporal "sabe" o que esta sendo representado nas partes móveis de um vídeo. Para fazer isso, eles propõem amplificar as ativações da última camada convolucional de uma CNN pela magnitude do OF, permitindo que uma rede apenas espacial passe a entender quais partes de uma imagem estão se movendo antes das camadas totalmente conectadas. A Figura 3.11 apresenta uma amplificação das *features* utilizando OF (imagem em cinza) aprimorando as ativações da última camada convolucional (camada em azul). Existem também métodos de fusão para combinar mapas de *features* espaciais de CNNs treinadas em diferentes *datasets*. A principal ideia é combinar uma rede de dois canais usando uma fusão multiplicativa, que pretende amplificar ou suprimir ativações com base nas *features* de cada imagem. Os experimentos realizados utilizando os *datasets* UCF101 [SZS12] e HMDB51 [KJG+11] mostram que ocorreram melhorias nos resultados em relação as abordagens anteriores.

Feichtenhofer *et al.* [FPZ16a] afirmam que a arquitetura Two-Stream desenvolvida de Simonyan e Zisserman [SZ14a] não é capaz de reconhecer o que está em movimento, uma vez que os aspectos espacial e temporal são mesclados somente no final da arquitetura. Por isso, os autores focaram em desenvolver uma arquitetura que seja capaz de fundir as características espaciais e temporais em diversos níveis de granularidade, colocando as respostas do canal na mesma posição de pixel em correspondência, *i.e.*, concatenar as redes em uma camada convolucional em vez de concatená-las somente no fim da arquitetura. Por exemplo, algumas ações como escovar os dentes e escovar o cabelo possuem a mão se movendo em uma determinada localização espacial e, portanto, a rede temporal pode

reconhecer esse movimento enquanto a rede espacial reconhece a localização (dentes ou cabelos), e a sua combinação então discrimina a ação. Experimentos foram realizados utilizando os*datasets* UCF-101 [SZS12] e HMDB51 [KJG+11].

3.3 Resumo

Neste capítulo apresentamos os principais trabalhos relacionados a esta dissertação, bem como suas características, os *datasets* utilizados por esses trabalhos. Todavia, a literatura carece de trabalhos com caráter de análise que visam entender a relação do domínio do problema com as abordagens utilizadas. Destacamos os trabalhos que utilizam CNNs, bem como as arquiteturas Two-stream e C3D, as quais foram utilizadas na fase de experimentos desta dissertação. Outra relevante ideia, a qual foi utilizada nesse trabalho, foi o método de extração de *features* gerando o *Optical Flow* das imagens. No próximo capítulo iremos apresentar a metodologia empregada afim de realizar a análise que propomos nesta pesquisa.

4. METODOLOGIA

Esta dissertação possui cunho experimental a fim de analisar a viabilidade de técnicas de *Deep Learning* para reconhecimento de ações em *datasets* pequenos. Este capítulo apresenta a metodologia empregada para a execução dos experimentos necessários que visam comprovar as hipóteses apontadas na Seção 1.3 desta dissertação. Com isso, iremos detalhar os *datasets* utilizados, as técnicas de pré-processamento, as métricas necessárias para avaliar os experimentos e as arquiteturas aplicadas durante a fase de experimentação.

4.1 *Datasets*

Neste trabalho, nos concentramos em *datasets* pequenos que contenham comportamento individual, *i.e.*, onde uma única pessoa é o alvo da ação, e dois tipos diferentes de fundo (estático ou dinâmico). Devido a grande quantidade de experimentos e poucas GPUs para uso, nos encontramos em um limite computacional onde tivemos que escolher um número reduzido de *datasets* para realizar a nossa análise. Para tal, foi escolhido uma margem de 3 *datasets*, cada um possuindo características próprias a fim de enriquecer a análise proposta nesta dissertação.

O ***Kitchen Scene Context based Gesture Recognition***[1] [SKD+13], ou KSCGR, é um *dataset* de ações de cozinha lançado como um desafio na *International Conference on Pattern Recognition*[2] (ICPR 2012). O conjunto de dados contém cenas capturadas por um sensor de kinect fixado na parte superior da cozinha (*background* estático), fornecendo sequências de imagens de cor e profundidade sincronizadas. Cada vídeo tem uma duração de 5 a 10 minutos, contendo entre 9.000 e 18.000 *frames*. O *dataset* contém videos de atores cozinhando 5 diferentes pratos utilizando ovos: “presunto e ovos” (*ham and eggs*), “omelete” (*omelet*), “ovos mexidos” (*scrambled egg*), “ovo cozido” (*boiled egg*), e “crepe de ovo em tiras” (*kinshi-tamago*). Os organizadores do *dataset* atribuíram rótulos a cada quadro, indicando o tipo de ação realizada pelos atores durante o processo de cozinhar os pratos. O conjunto de dados rotula as ações em 9 tipos: “quebrar” (*breaking*), “misturar” (*mixing*), “assar” (*baking*), “misturar” (*turning*), “cortar” (*cutting*), “ferver” (*boiling*), “temperar” (*seasoning*), “descascar” (*peeling*) e “nenhuma” (*none*), onde (*none*) significa que não existe uma ação no quadro atual. A Figura 4.1 mostra a distribuição de *frames* do *dataset*.

O ***DogCentric Activity dataset***[3] [ITKR14], ou simplesmente DogCentric, consiste em vídeos de primeira pessoa com cenas ao ar livre de câmeras portáteis montadas nas

[1] http://www.murase.m.is.nagoya-u.ac.jp/KSCGR/
[2] http://www.icpr2012.org/
[3] http://robotics.ait.kyushu-u.ac.jp/~yumi/db/first_dog.html

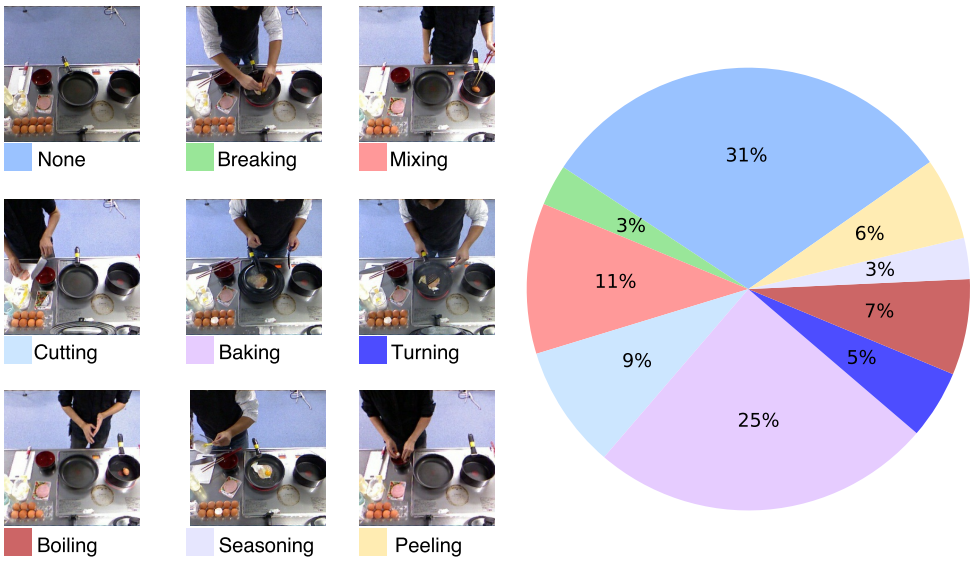


Figura 4.1 – Imagens de cada ação e a sua respectiva quantidade em percentual disponíveis no *dataset* KSCGR.

costas de um cão. Estes tipos de vídeos (vídeos em primeira pessoa - também conhecidos como vídeos egocêntricos) são muito desafiadores devido ao intenso movimento de câmera. O *dataset* de ações DogCentric contém vídeos de ações de cães com resolução de 320 $\times$ 240 (gravada a 48 *frames* por segundo), realizada de um ponto de vista egocêntrico para animais. O conjunto de dados é dividido em 209 vídeos contendo 10 ações diferentes realizadas por 4 cães. As 10 ações inclusas no *dataset* são: “esperando por um carro passar” (simplificado no *dataset* como “carro”, do inglês, *Car*), “bebendo água” (*Drink*), “comendo” (*Feed*), “olhando para a esquerda” (*Left*), “olhando para a direita” (*Right*), “sendo acariciado” (*Pet*), “brincando com bola” (*Play*), “sacudindo o corpo” (*Shake*), “farejando” (*Sniff*), “caminhando” (*Walk*). Esse *dataset* é desbalanceado contendo 4.920 *frames* de *Car*, 3.300 de *Drink*, 3.795 de *Feed*, 1.950 de *Left*, 1.380 de *Right*, 3.740 de *Pet*, 3.545 de *Play*, 1.880 de *Shake*, 4.960 de *Sniff* e 4.175 de *Walk*, somando-se 33.645 *frames* ao total. A Figura 4.2 ilustra uma imagem de cada ação do *dataset*.

O ***UCF YouTube Action Dataset***[4] [LLS09], ou UCF-11, consiste em 1.600 vídeos extraídos do YouTube capturados em condições descontroladas, como vídeos gravados por um amador usando uma câmera de mão. O *dataset* contem 11 diferentes ações: “arremesso no basquete” (*basketball shooting*), “andar de bicicleta” (*bike/cycling*), “mergulhar” (*diving*), “tacada no golfe” (*golf swinging*), “andar a cavalo” (*horse back riding*), “malabarismo com bola” (*soccer juggling*), “andar de balanço” (*swinging*), “rebatida no tênis” (*tennis swinging*), “pulo de trampolim” (*trampoline jumping*), “cortada no voleibol” (*volleyball spiking*), e “caminhar com o cachorro” (*walking with a dog*). Todos os vídeos no conjunto de dados foram coletados manualmente do YouTube e as imagens possuem tamanho de 240$\times$320 pixeis. Cada vídeo foi convertido em uma taxa de *frames* de 29,97 fps e as ano-

[4] http://crcv.ucf.edu/data/UCF_YouTube_Action.php

Figura 4.2 – Exemplo das imagens disponíveis para cada ação contidas no *dataset* Dog-Centric.

tações foram feitas em conformidade, contendo uma única ação associada ao vídeo inteiro. Este conjunto de dados é muito desafiador devido a grandes variações no movimento da câmera, aparência do objeto e pose, escala de objeto, ponto de vista, fundo desordenado, condições de iluminação, *etc*. É importante notar que algumas ações também são realizadas com animais como um cavalo ou cachorro, ou objetos como a bicicleta. A Figura 4.3 mostra dois exemplos das ações *basketball shooting* e *horseback riding*.

Figura 4.3 – Exemplo de duas classes do UCF-11, “*basketball shooting*” (à esquerda) e “*horseback riding*” (à direita).

A Tabela 4.1 sintetiza as características mais relevantes dos três *datasets* escolhidos para a realização desse trabalho, onde “Ações” é o número de ações executadas; “Clipes” é o número de clipes que o *dataset* possui; “*Background*” é mudança na posição da câmera durante a gravação (o fundo pode ser estático ou dinâmico); “Lugares” indica se o *dataset* é gravado em lugares diferentes; “Resolução” é a resolução na qual os vídeos do *dataset* foram disponibilizados; “Lançamento” é o ano em que o *dataset* foi lançado; e “Atores"que indica se as ações são realizadas individualmente ou por grupos de pessoas.

Tabela 4.1 – Informações referente aos 3 *datasets* selecionados para reconhecimento de ações.

Dataset	**Ações**	**Clipes**	***Background***	**Lugares**	**Resolução**	**Lançamento**	**Atores**
KSCGR	9	35	estático	não	640x480	2012	individual
DogCentric	10	209	dinâmico	sim	320x240	2014	individual
UCF-11	11	1.168	dinâmico	sim	320x240	2009	híbrido

4.2 Pré-Processamento

Para os 3 *datasets* selecionados neste trabalho, iremos utilizar duas etapas de pré-processamento. São elas: redimensionamento de imagem e geração do *dense optical flow* [Far03]. O redimensionamento é importante, uma vez que reduz o espaço multidimensional requerido pelas CNNs para aprender *features* adequadas para a classificação da imagem, bem como o tempo do processamento total do método. Nesta etapa, todas as imagens dos conjuntos de dados são redimensionadas para uma resolução fixa de 256×256. O segundo passo gera a representação do DOF de *frames* adjacentes. Em poucas palavras, o DOF representa o deslocamento 2D de pixeis entre *frames* gerando vetores correspondentes ao movimento de pontos do primeiro *frame* para a segundo. O DOF gera esses vetores de deslocamento, tanto para deslocamentos horizontais como verticais, em relação a todos os pontos dentro dos *frames*. Para gerar a imagem final para cada sequência de *frames*, combinamos os vetores de OF horizontais e verticais, associando uma cor à sua magnitude e a intensidade da cor à direção dos mesmos. A saída do passo de pré-processamento de dados consiste em dois conjuntos de dados contendo os dados originais com canais RGB e tamanho redimensionado, e os dados de DOF que encapsulam o movimento em *frames*. A Figura 4.4 ilustra a imagem em RGB (à esquerda), e uma representação em DOF de uma sequência de 2 *frames* (à direita) para o *dataset* KSCGR.

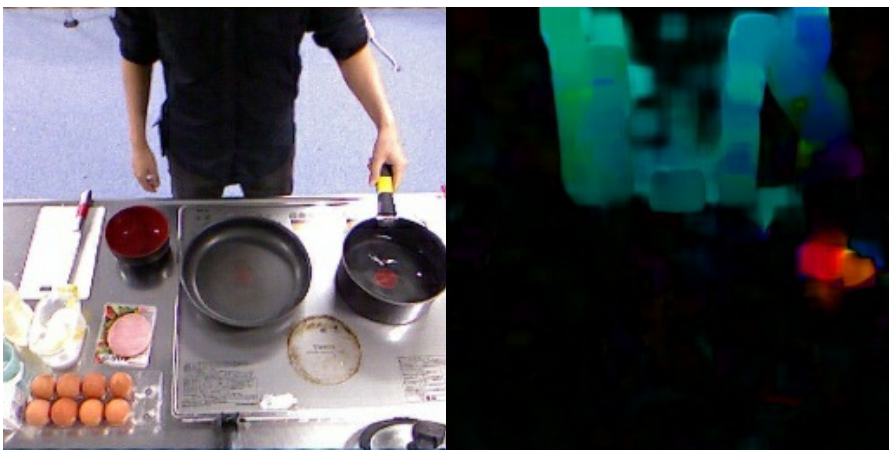


Figura 4.4 – Imagem em RGB (à esquerda) e em *Dense Optical Flow* (à direita).

Por fim, todas as imagens são normalizadas, subtraindo-se os valores de cada canal da imagem por 128 e dividindo-os também por 128. Também irá subtrair-se das imagens o valor médio de todo o conjunto de treino. Como já explicado na Seção 2.1.1, o processo de normalização irá centralizar os dados e resultará com que a busca do otimizador pelo mínimo global seja menos custosa.

4.3 Métricas

Para medir a qualidade dos experimentos, nós utilizamos métricas como a acurácia (do inglês, *accuracy*), precisão (do inglês, *precision*), revocação (do inglês, *recall*) e a medida F1 (do inglês,*F1-score*). Foram escolhidas essas métricas porque são amplamente utilizadas para avaliação de tarefas de reconhecimento de ações. Essas métricas, bem como o uso do conceito de verdadeiro positivo/negativo e falso positivo/negativo, tiveram seu amplo uso inicial na medicina, e com o tempo essas métricas de avaliação foram adotadas pelas áreas de Aprendizado de Máquina e Mineração de Dados [Pow11].

A acurácia é uma medida projetada para relatar a porcentagem de dados classificados corretamente. Equação 4.1 mostra como a acurácia é calculada, onde *vp* (verdadeiro positivo) representa classes positivas corretamente classificadas, *vn* (verdadeiro negativo) representa classes negativas corretamente classificadas, *fp* (falso positivo) indica classes negativas incorretamente classificadas, e *fn* (falso negativo) indica classes positivas incorretamente classificadas. Vale ressaltar que essas medidas não somente são aplicáveis em contextos de classificação binária, mas também em classificações onde há mais de duas classes.

$$Acurácia = \frac{vp + vn}{vp + vn + fp + fn} \tag{4.1}$$

A precisão indica a proporção de exemplos positivos previstos que estão corretamente classificados. Esta medida é comumente usada pela área de aprendizado de máquina, mineração de dados e recuperação de informação [Pow11], é dada por:

$$Precisão = \frac{vp}{vp + fp} \tag{4.2}$$

A medida de revocação mede a proporção de exemplos positivos que estão corretamente classificados. A Equação 4.3 define como a medida de revocação é calculada.

$$Revocação = \frac{vp}{vp + fn} \tag{4.3}$$

A medida F1 (também conhecido como *F1-score*) é a média harmônica da precisão e da revocação, sintetizando o *trade-off* entre as duas medidas. A medida F1 é dada por:

$$F1 = 2 \times \frac{Precisão \times Revocação}{Precisão + Revocação} \tag{4.4}$$

4.4 Modelos

Neste trabalho realizamos a análise do desempenho de modelos de *Deep Learning* em *datasets* pequenos. Para tal análise, seguimos a mesma separação adotada por Poppe *et al.* [Pop10] e implementamos modelos de Classificação Direta (do inglês, *Direct Classification*, Seção 4.4.1) e Modelos de Estado-Espaço Temporais (do inglês, *Temporal State-Space Models*, Seção 4.4.2).

4.4.1 *Classificação Direta*

Os modelos de Classificação Direta (modelos estáticos), dispostos nesta seção não levam em conta o aspecto temporal do problema, classificando *frame* a *frame* as ações ao tratar o problema de maneira estática. Optamos por utilizar duas arquiteturas de rede convolucional que se classificam como Classificação Direta. Elas são as redes VGG-16 [SZ14b] e a Inception-V3 [SVI+16]. Estas arquiteturas foram escolhidas devido a fatores como: são modelos tradicionais na área, são amplamente disponíveis em diversos *frameworks* de DL, e possuem um número reduzido de parâmetros em comparação a outras arquiteturas. A seguir serão apresentados maiores detalhes a respeito das arquiteturas escolhidas.

VGG16: As redes VGG [SZ14b], desenvolvidas pelo Grupo de Geometria Visual (VGG) de Oxford, apresentam uma estratégia simples e efetiva de construção de CNNs profundas, apenas empilhando blocos de convoluções+*pooling* de mesmo tamanho. A VGG16 contém 16 camadas divididas em grupos de convoluções usando filtros 3×3 em cada camada convolucional para representar *features* complexas. A VGG vem com a ideia de que a convolução em sequência de 3×3 pode emular o efeito de campos receptivos maiores, como 5×5 e 7×7 usados em redes anteriores. Em todo o *pipeline*, a rede recebe uma sequência de imagens na entrada, passando-as através de várias camadas convolucionais, agrupando camadas e camadas totalmente conectadas, terminando em uma função *softmax* que gera a probabilidade de cada imagem para cada classe. A robustez das redes VGG foi comprovada por várias tarefas de reconhecimento visual [Gir15, CLS15, RHGS15, FPZ16a, SLD17]. Embora as redes VGG tenham o recurso

convincente da simplicidade arquitetônica, elas exigem muita computação. Quando comparado com redes anteriores como o AlexNet [KSH12], a arquitetura VGG emprega 3 vezes mais parâmetros. Para reduzir o número de parâmetros, outras redes profundas foram propostas, como a Inception-V3 [SVI+16].

Inception-V3: A rede Inception-V3 do Google [SVI+16] (também chamada de V3), é uma Rede Neural Convolucional de 48 camadas baseada em módulos *inception*. Esses módulos contêm filtros convolucionais com várias dimensões diferentes em paralelo, cobrindo diferentes grupos de informações. A ideia principal dos módulos *inception* é a estratégia dividir-transformar-juntar (do inglês, *split-transform-merge*), onde a entrada é dividida em grupos dimensionais pequenos, transformadas por um conjunto de filtros de tamanhos diferentes e por fim mescladas por uma camada de concatenação. Esses módulos tentam se aproximar do poder representacional de camadas grandes e densas, com menor complexidade computacional. A V3 é uma evolução da arquitetura GoogLeNet [SLJ+15], que foi a vencedora do desafio *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [RDS+15] em 2014. Neste mesmo desafio, a V3 obteve um erro de 4,2% na tarefa de classificação considerando as 5 top classificaçõs por imagem. As principais melhorias na arquitetura V3 quando comparadas com sua versão anterior estão ligadas ao aumento de profundidade, de 22 camadas para 48 camadas, e uma redução no tamanho dos filtros para um máximo de 3×3. A arquitetura *Inception* da GoogLeNet foi projetada para funcionar bem com um número reduzido de parâmetros quando comparado com as arquiteturas anteriores. Por exemplo, a GoogLeNet contém cerca de 7 milhões de parâmetros, o que representa uma redução de 9 vezes em relação a AlexNet [KSH12] que contém 60 milhões de parâmetros.

4.4.2 *Modelos de Estado-Espaço Temporais*

A seguir são apresentadas abordagens que levam em conta o aspecto temporal, seja através de *features* oriundas de algum algoritmo para representar a temporalidade que ocorre em uma ação ou um modelo que é capaz de capturar essa temporalidade por si só. Os modelos escolhidos e apresentados a seguir são modelos tradicionais na literatura que atingiram estado-da-arte em *datasets* específicos.

Two-Stream Network: proposta por Simonyan e Zisserman [SZ14a], a rede calcula dois fluxos de reconhecimento separados. O primeiro é espacial, um espacial contendo imagens RGB estáticas, enquanto que o segundo é temporal, contendo o movimento realizado em uma sequência de imagens. Ambas as redes são finalmente fundidas por uma fusão tardia (do inglês, *late fusion*), onde os resultados de predição são combinados pela concatenação ou pela média. A fusão dos resultados pode ser obtida pela média do *softmax* obtidos pela arquitetura, ou utilizar a concatenação seguida de um algoritmo classificador

como a SVM [GM15]. A rede Two-Stream tem como objetivo fundir as *features* extraídas de imagens estáticas por uma CNN com as *features* extraídas do *optical flow* de imagens através de outra CNN. Em nosso trabalho, utilizamos tanto a VGG-16 quanto a V3, separadamente, em ambos os fluxos, recebendo *frames* de vídeos no fluxo estático e imagens geradas pelo OF no fluxo temporal. Após, nós utilizamos uma SVM para separar tanto os resultados concatenados quanto as médias desses. A geração de imagens de OF é explicada neste capítulo na Seção 4.2. Mais detalhes sobre a rede Two-Stream podem ser vistos na Seção 3.2.

C3D: Proposta por Tran *et al.* [TBF+15], a rede convolucional 3D (C3D) utiliza camadas convolucionais 3D para extrair *features* temporais de uma sequência de *frames*. Diferente de uma convolução 2D, a C3D faz uso de camadas de convolução e *pooling* 3D que recebem um bloco de *frames* capazes de identificar *features* temporais apropriadas para classificação em vídeos. Embora os modelos de redes neurais 3D sejam adequados para lidar com dados temporais, como sequências de vídeo, eles acabam possuindo muitos parâmetros por envolver uma dimensão a mais na resolução do problema. Com isso, existe uma certa facilidade em possuir problemas como o *overfitting*, onde o modelo aprende rápido demais os dados de treino e não se torna capaz de generalizar para dados não vistos anteriormente.

Em nosso trabalho, iremos utilizar uma C3D para classificar ações em vídeos com pequenas modificações na arquitetura, as quais serão melhor explicadas no Capítulo 5, devido a limitações de hardware.

LRCN: Os modelos de RNN são projetados para lidar com informações sequenciais,reconhecendo padrões em dados usando não apenas a entrada atual, mas também a informação que percebem anteriormente no tempo. Tais redes são chamadas recorrentes porque utilizam os mesmos pesos para cada elemento em uma sequência com a saída sendo dependente da informação anterior. A unidade recorrente (do inglês, *recurrent unity*) pode ser entendida como uma memória sobre o que a rede já processou até agora. Usar informações anteriores é satisfatório para muitas tarefas, tais como para o reconhecimento de fala [GJ14] e geração de texto [SMH11]. Embora os modelos RNN sejam adequados para lidar com dados temporais, como sequências de vídeo, eles têm uma limitação significativa conhecida como “desaparecimento dos gradientes”, o que dificulta a retro-propagação de um sinal de erro através de um intervalo temporal de longo alcance e, consequentemente, sendo difícil treiná-los para aprender padrões muito longos no tempo. Para lidar com esse problema, Hochreiter e Schmidhuber [HS97] propõem a abordagem conhecida como LSTM, que permite o aprendizado de longo alcance, incorporando unidades de memória que permitem que a rede aprenda, esqueça e atualize estados ocultos com novas informações.

Neste trabalho, nós utilizamos uma adaptação da Rede Convolucional Recorrente de Longo Prazo (LRCN) [DHR+17], que combina um extrator de *features* visuais (CNN) com um modelo que pode aprender a reconhecer e sintetizar a dinâmica temporal para tarefas

envolvendo dados sequenciais (LSTM). Nossa LRCN difere-se da original, uma vez que não a treinamos de ponta a ponta, usando a CNN apenas como extrator de *features* para a rede recorrente, tendo em vista o custo computacional de atualizar os pesos da CNN.

4.5 *Handcrafted Baselines*

O trabalho proposto por Wang *et al.* [WKSL11] apresenta uma abordagem que agrupa diferentes descritores, chamada por ele de Trajetórias Densas (do inglês, *Dense Trajectories*). As Dense Trajectories são *handcrafted features* geradas por outros 5 descritores: Trajetórias (do inglês, *Trajectories*), Histograma de Gradientes (do inglês, *Histogram of Gradients*, HOG), Histograma de Fluxo Óptico (do inglês, *Histogram of Optical Flow*, HOF), Histórico de Movimento de Contorno (do inglês, *Motion Boundary History*, MBH), este último podendo ser dividido em MBHx e MBHy. A arquitetura do modelo proposto por Wang é representada na Figura 4.5.

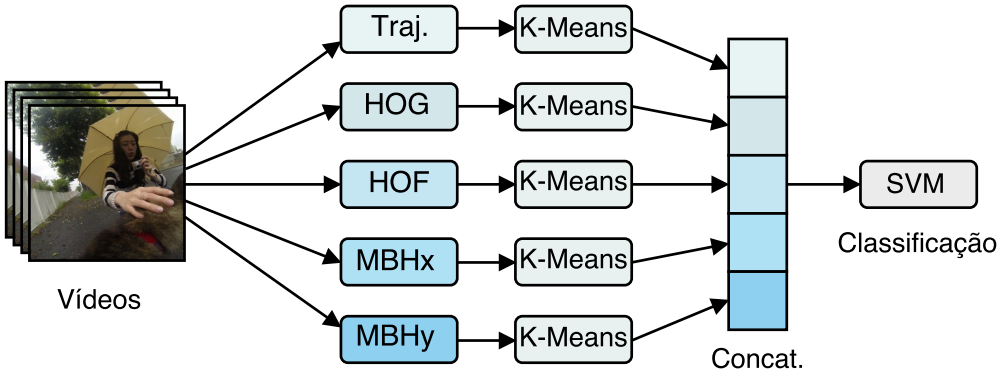


Figura 4.5 – Exemplificação da arquitetura da abordagem *Dense Trajectories* proposta por Wang *et al.* [WKSL11].

A Dense Trajectories é uma abordagem de alta relevância na literatura, sendo um dos trabalhos mais citados que utilizam *handcrafted features* para classificação de ações, e foi um dos trabalhos que serviram de inspiração para criação de modelos que capturam o aspecto temporal em redes convolucionais [SZ14a]. Além de propor uma abordagem nova, os autores obtiveram resultados superiores em quatro datasets diferentes. Por ser um trabalho com tais características e por utilizar diferentes descritores, optamos por implementar a abordagem de Dense Trajectories como *baseline* para esta pesquisa. Por tanto, deverão ser extraídas as *features* de todos os 5 descritores citados anteriormente e com isso treinar 5 diferentes modelos de agrupamento através do algoritmo K-Means a fim de obter um modelo capaz de agrupar as *features* providas pelos descritores. O processo de treinar o modelo de agrupamento com os descritores pode ser visualizado no Algoritmo 1. Após isso, treina-se um modelo de SVM com as *features* concatenadas de cada um dos 5 descritores

Algoritmo 1: Gerando agrupamento por descritor

Entrada: Vídeos (V)
Saída: Modelo de agrupamento para cada descritor

```
1  início
2      D ← [Trajectory, HOG, HOF, MBHx, MBHy]
3      para v em V faça
4          para d em D faça
5              feature_d ← extrair_features(d, v)
6              feature_normalizada_d ← normalizar_features(feature_d)
7              agrupamento_d ← gerar_agrupamento(feature_normalizada_d)
8          fim
9      fim
10     retorna cluster
11 fim
```

(Algoritmo 2). Por fim, serão extraídas as *features* dos vídeos de teste, agrupadas nos mo-

Algoritmo 2: Gerando classificador SVM

Entrada: *Clusters* (C), Vídeos (V)
Saída: Classificador SVM

```
1  início
2      D ← [Trajectory, HOG, HOF, MBHx, MBHy]
3      para v em V faça
4          para d em D faça
5              feature_d ← extrair_features(d, v)
6              feature_normalizada_d ← normalizar_features(feature_d)
7              para c_d em C faça
8                  histograma_d ← gerar_hist(c_d, feature_normalizada_d)
9                  hist_norm_d ← normalizar_histograma(histograma_d)
10                 hist_concat ← concatenar_histograma(hist_norm_d)
11             fim
12         fim
13     fim
14     modelo_svm ← treinar_svm(hist_concat)
15     retorna modelo_svm
16 fim
```

delos treinados, e classificadas pelo SVM. Esta etapa pode ser melhor compreendida no Algoritmo 3.

Nós utilizamos os mesmos valores de parâmetros bem como o número de inicializações do algoritmo K-Means sugeridos por Wang *et al.* [WKSL11]. Uma limitação importante a se mencionar desta abordagem é que ela apenas funciona para vídeos que possuem uma única ação. Portanto não foi possível realizar os experimentos de *baseline* com o *dataset* KSCGR.

Algoritmo 3: Gerando classificação

Entrada: *Clusters* (C), vídeo (v), modelo SVM treinado (SVM)
Saída: Classificação do vídeo

```
início
    D ← [Trajectory, HOG, HOF, MBHx, MBHy]
    para d em D faça
        feature_d ← extrair_features(d, v)
        feature_normalizada_d ← normalizar_features(feature_d)
        para c_d em C faça
            histograma_d ← gerar_hist(c_d, feature_normalizada_d)
            hist_norm_d ← normalizar_histograma(histograma_d)
            hist_concat ← concatenar_histograma(hist_norm_d)
        fim
    fim
    classificacao ← classificar_video(hist_concat, SVM) retorna classificacao
fim
```

Com a finalidade de obter uma gama maior de resultados de *baseline*, também são treinadas SVMs com essas *features* separadamente. Por tanto, serão gerados 6 resultados finais de *baseline* para cada *dataset* apto a ser utilizado. Para otimizar os parâmetros do classificador SVM que irá utilizar as *features* geradas pelos *basselines* nós seguimos as orientações do trabalho proposto por Hsu *et al.* [HCL03]. Onde iremos realizar uma busca em *grid*, com um modelo utilizando kernel RBF e outro linear. Para os modelos lineares e com o kernel RBF os valores para o hiperparâmetro C variam de 2^{-5}, 2^{-3}, ..., 2^{15}, e especialmente para o modelo com o kernel RBF também é aconselhado variar o hiperparâmetro γ de 2^{-15}, 2^{-13}, ..., 2^{3}. Dado essa combinação, para cada descritor deverá ser gerado 11 modelos para o SVM linear e 110 modelos com o kernel RBF, totalizando 726 modelos treinados.

4.6 Resumo

Neste capítulo apresentamos a metodologia empregada para realizar a análise proposta neste trabalho. Apresentamos os *datasets* KSCGR, DogCentric e UCF-11 e as suas principais características. O KSCGR apresenta vídeos gravados através de uma câmera estática de atores realizando ações na cozinha. O desafio de tal *dataset* está na dificuldade de reconhecer as ações, uma vez que cada ator pode realizar a mesma ação de uma maneira completamente diferente um do outro. O *dataset* DogCentric apresenta vídeos egocêntricos de ações realizadas por cachorros. Existem vários desafios em realizar classificação de ações com esse *dataset*, uma delas é o fato do vídeo ser egocêntrico e a câmera estar fixa nas costas de um cachorro, existe um forte movimento de câmera, o que

acaba dificultando a classificação final do vídeo. Por fim, o *dataset* UCF-11 consiste em vídeos amadores de pessoas praticando diferentes modalidades de esportes. Os desafios estão na qualidade dos vídeos, bem como as mesmas ações serem realizadas com fundos completamente diferentes.

Neste capítulo também apresentamos os modelos de DL e as *baselines* que iremos utilizar para realizar os experimentos desta pesquisa. A Tabela 4.2 sumariza os modelos mencionados anteriormente e as suas variações que fazem parte dos nossos experimentos.

No próximo capítulo iremos abordar os experimentos realizados bem como uma análise sobre os resultados gerados.

Tabela 4.2 – Modelos de *Deep Learning* utilizados nos experimentos deste trabalho.

Descrição	Modelo	Tipo
Modelo Inception-V3 treinado com imagens RGB	V3-RGB	Estático
Modelo Inception-V3 treinada com imagens em *Optical Flow*	V3-OF	Estático
Modelo SVM treinado com *features* extraídas da última camada totalmente conectada de uma arquitetura Inception-V3	V3-FC-SVM	Estático
Modelo SVM treinado com *features* extraídas da última camada de convolução de uma arquitetura Inception-V3	V3-CONV-SVM	Estático
Modelo LRCN treinado com *features* extraídas da última camada totalmente conectada de uma arquitetura Inception-V3	V3-FC-LRCN	Temporal
Modelo LRCN treinado com *features* extraídas da última camada de convolução de uma arquitetura Inception-V3	V3-CONV-LRCN	Temporal
Modelo Two-Stream treinado com a concatenação do softmax da arquitetura Inception-V3 treinada com imagens RGB e OF	V3-CONCAT-TWO-STREAM	Temporal
Modelo Two-Stream treinado com a média do softmax da arquitetura Inception-V3 treinada com imagens RGB e *Optical Flow*	V3-MEAN-TWO-STREAM	Temporal
Modelo VGG16 treinado com imagens RGB	VGG16-RGB	Estático
Modelo VGG16 treinado com imagens em *Optical Flow*	VGG16-OF	Estático
Modelo SVM treinado com *features* extraídas da última camada totalmente conectada de uma arquitetura VGG-16	VGG16-FC-SVM	Estático
Modelo SVM treinado com *features* extraídas da última camada de convolução de uma arquitetura VGG-16	VGG16-CONV-SVM	Estático
Modelo LRCN treinado com *features* extraídas da última camada totalmente conectada de uma arquitetura VGG-16	VGG16-FC-LRCN	Temporal
Modelo LRCN treinado com *features* extraídas da última camada de convolução de uma arquitetura VGG-16	VGG16-CONV-LRCN	Temporal
Modelo Two-Stream treinado com a concatenação do softmax da arquitetura VGG-16 treinada com imagens RGB e *Optical Flow*	VGG16-CONCAT-TWO-STREAM	Temporal
Modelo Two-Stream treinado com a média do softmax da arquitetura VGG-16 treinada com imagens RGB e *Optical Flow*	VGG16-MEAN-TWO-STREAM	Temporal
Modelo C3D treinado com imagens RGB	C3D	Temporal

5. EXPERIMENTOS

Este capítulo visa apresentar todos os passos da fase de experimentação. São apresentados detalhes sobre o ambiente utilizado (Seção 5.1), otimização de parâmetros empregada (Seção 5.2), e resultados obtidos (Seção 5.3) na tentativa de validação das hipóteses levantadas.

5.1 Ambiente

Como foi dito no Capítulo 1, a grande ascensão das abordagens de DL foram conquistadas devido a evolução do hardware, mais precisamente das GPUs. Desta forma, os experimentos realizados nesse trabalho utilizaram uma GPU Tesla K40, a qual possui 12 GB de memória. Como linguagem para o desenvolvimento das CNNs, bem como para os outros algoritmos de ML implementados, utilizamos a linguagem Python. Já para a implementação das CNNs, bem como das LSTMs, utilizamos o *framework* de desenvolvimento Keras[1] juntamente com o *framework* TensorFlow[2] [ABC+16] como *backend*. Para a implementação dos algoritmos de ML utilizamos a biblioteca Scikit-Learn[3] [PVG+11]. A hierarquia de desenvolvimento utilizada nesse trabalho listada acima pode ser vista na Figura 5.1, onde a linguagem Python é utilizada para a implementação do trabalho através do *framework* Keras, que por sua vez, abstrai o TensorFlow e conecta-se com a GPU.

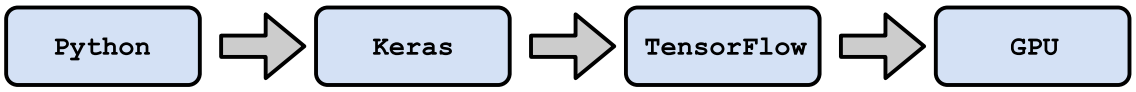


Figura 5.1 – Hierarquia de desenvolvimento utilizada para a implementação do trabalho proposto.

5.2 Configurações e Otimização de Parâmetros

Para todas as CNNs utilizadas neste trabalho utilizamos a técnica de pré-treino, devido ser uma técnica altamente recomendada ao lidar pequenos conjuntos de dados [GFF08, CMS12, AYX+08, WTS+13, PY10]. Tal técnica diminui o tempo de treinamento, evitando

[1] https://keras.io/
[2] https://www.tensorflow.org
[3] http://scikit-learn.org.

que a rede precise aprender características mais elementares e ainda assim combatendo o *overfitting*, possibilitando que a rede aprenda somente *features* mais específicas do problema em questão. Cada CNN foi pré-treinada utilizando o *dataset* ImageNet[4] [DDS+09] e os seus respectivos pesos foram carregados diretamente da biblioteca central Keras. Para modelos que utilizam o SVM como algoritmo classificador, nós o treinamos com *features* de treino + validação, oriundos das camadas de convolução (*conv*) e totalmente conectadas (*fc*). Por fim, testamos o modelo com as *features* de teste. É importante mencionar que, para o SVM, não alteramos qualquer hiperparâmetro, uma vez que decidimos testar diferentes valores para os modelos de DL. Pois focamos em analisar as melhores hiperparâmetros indiferentes do classificador.

5.2.1 Configurações dos modelos

Para todas as CNNs, a fase de treinamento utiliza gradiente descendente estocástico com o *momentum* de $0,9$. A ativação de cada convolução (incluindo as dentro dos módulos "*inceptions*" para a rede V3) se da pela função ReLU. Na fase de treino utilizamos 128 imagens para cada iteração de treinamento nos modelos de DL. Para reduzir as chances de *overfitting*, aplicamos o *dropout* nas camadas totalmente conectadas. Uma vez que estamos trabalhando com pequenos conjuntos de dados, não precisamos de muitas épocas para convergir o modelo. Com isso em mente, estabelecemos o limite máximo de épocas para 30. Além disso, nós implementamos a técnica de *early stopping*, a qual finaliza o treinamento caso verifique que a função caiu em um mínimo local. De fato, a maioria de nossos experimentos não levam mais de 15 épocas para convergir. Para modelos que utilizam uma SVM como classificador final adotamos a configuração do classificador como a padrão fornecida pelo Scikit-Learn.

5.2.2 Otimização de Parâmetros

Neste trabalho, a maior parte dos modelos implementados utilizam a otimização de parâmetros conhecida como *grid search* com o objetivo de encontrar os melhores hiperparâmetros para nossos modelos. O *grid search* é um processo relativamente simples e altamente recomendado quando os modelos parametrizados não levam tanto tempo para treinar. Basicamente, o *grid search* realiza todas as combinações possíveis entre os hiperparâmetros previamente estipulados, tentando encontrar o melhor modelo para o seu problema. Neste trabalho, utilizamos o *grid search* para encontrar os melhores hiperparâmetros para todos os nossos modelos de DL. Os hiperparâmetros otimizados nessa pes-

[4] http://www.image-net.org/

quisa são: *dropout* e o *learning rate*. Para o *dropout* utilizamos o *grid search* contendo os seguintes valores: 0.5, 0.7, 0.9 e 0.95. Já para o LR utilizamos: $5e^{-3}$, $1e^{-3}$, $5e^{-4}$, $1e^{-4}$, $5e^{-5}$ e $1e^{-5}$. Conforme mencionado anteriormente, o método de *grid search* testará todas as combinações possíveis, nesse caso, utilizando 4 valores diferentes para o *dropout* e 6 diferentes valores para o LR. Com isso, obtemos no total de 24 combinações possíveis para validar. Neste trabalho possuímos 9 arquiteturas de DL as quais passaram pelo método de otimização de parâmetros, resultando em 216 modelos treinados. Além disso, possuímos 8 modelos os quais não passam por otimização de parâmetro, sendo esses os modelos que utilizam SVM ou a abordagem Two-Stream. Em suma, possuímos no total 224 modelos treinados para cada *dataset*, somando-se 672 modelos treinados no total.

5.3 Resultados

5.3.1 Visão Geral

Conforme mencionado na Seção 5.2.2, treinamos 224 modelos para cada *dataset*, totalizando 672 modelos que envolvem técnicas de DL para 3 diferentes *datasets*. Os melhores modelos concentraram-se em abordagens que fazem o uso das arquiteturas LRCN e Two-Stream. Para o *dataset* KSCGR o melhor modelo atingiu 73,3% de acurácia utilizando *features* da camada FC de uma V3 em uma arquitetura LRCN. Não obstante, o melhor resultado obtido no *dataset* DogCentric também foi utilizando a arquitetura LRCN, porém com *features* da camada de convolução de uma VGG16, atingindo 60% de acurácia. Já para o *dataset* UCF-11, o melhor resultado atingiu 78,6% de acurácia, sendo obtido através da média da saída de uma rede V3 treinada com imagens em RGB e OF em uma arquitetura Two-Stream.

Além dos modelos temporais terem obtidos os melhores resultados para os 3 *datasets*, vale ressaltar a relevância dos modelos estáticos. No *dataset* DogCentric o modelo que utiliza a arquitetura V3 e imagens RGB obteve acurácia e precisão 3% inferior ao melhor modelo. Já para o *dataset* KSCGR, também utilizando uma arquitetura V3 e imagens em RGB, o modelo obteve a melhor precisão de todos os testes, resultando em 77,3% de precisão, sendo superior em $\approx$7% quando comparado com o modelo que obteve a segunda melhor acurácia. Para o *dataset* UCF-11, o modelo que utilizou *features* extraídas de uma camada de convolução da rede V3 para treinar um modelo de SVM obteve uma acurácia $\approx$3% inferior e uma precisão $\approx$1% inferior quando comparado com o melhor modelo para este *dataset*. Levando-se em conta que as diferenças obtidas pelos modelos estáticos obtiveram no máximo 7% de diferença na acurácia em relação aos modelos temporais, o *trade-off* entre custo computacional e acurácia deve ser levado em conta, visto que o treina-

mento e a classificação de modelos estáticos são extremamente mais rápidos que modelos temporais, devido a quantidade de parâmetros treináveis que esses modelos exigem.

Existem algumas correlações relevantes apresentadas na fase de validação dos modelos propostos em relação aos parâmetros de *dropout* e *learning rate*. Mais especificamente é que ao passo que aumentamos o valor de *dropout*, os melhores resultados são obtidos com LRs menores. É interessante também analisarmos os resultados obtidos por classe nos melhores modelos. Para tal, a Figura 5.2 ilustra as matrizes de confusão geradas através dos resultados obtidos dos melhores modelos para cada *dataset*, onde as linhas representam as classes verdadeiras e as colunas representam as classes preditas. Os tons de azul representam o valor em cada célula, indo cromaticamente a partir de um azul escuro representando valores mais altos para um azul mais claro representando valores mais baixos. A matriz de confusão mostra valores normalizados, isto é, os valores previstos são divididos pelo número total de valores reais para cada célula. Com isso, conseguimos perceber que os modelos selecionados possuem uma significativa capacidade em classificar corretamente algumas classes, aumentando conforme o tamanho do *dataset* e de suas características.

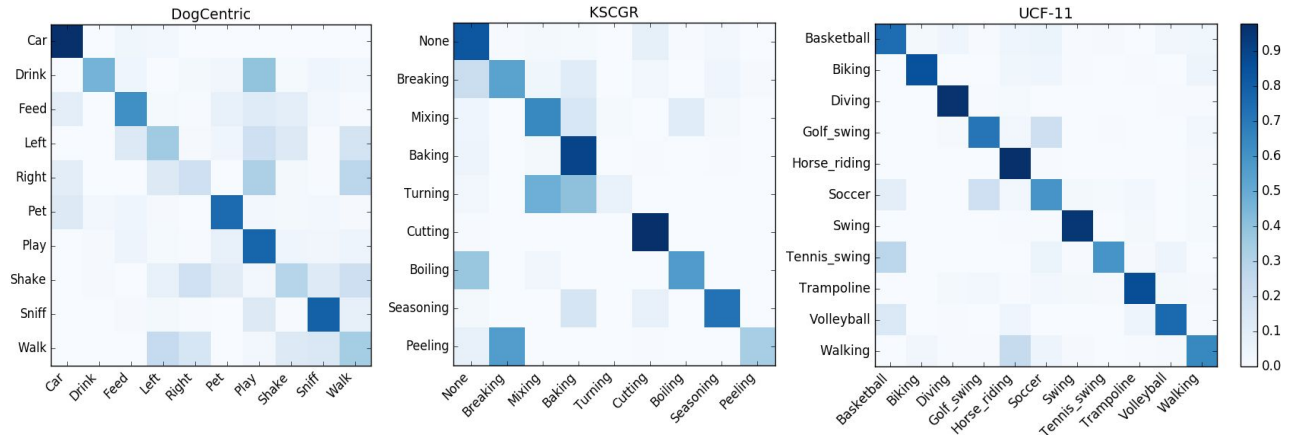


Figura 5.2 – Matriz de confusão normalizada dos melhores modelos para os *datasets* DogCentric, KSCGR e UCF-11 respectivamente.

Analisando os *datasets* observamos que os mesmos possuem imagens extremamente parecidas entre diferentes classes, sendo uma difícil tarefa até mesmo para os humanos realizarem a classificação. Um exemplo extraído do *dataset* KSCGR é apresentado na Figura 5.3, onde a ação de *Peeling* é erroneamente classificada por *Breaking*.

Conforme mencionado no Capítulo 4.5, nós propormos executar a abordagem *baseline* proposta por Wang *et al.* [WKSL11]. Com isso, realizamos experimentos utilizando os *datasets* DogCentric e UCF-11, uma vez que, já mencionado anteriormente, essa abordagem não pode ser executada em vídeos multi-classe. Foram treinados 121 modelos para cada descritor (HOG, HOF, MBHx, MBHy, *Trajectories* e *Dense*), totalizando 726 modelos treinados. Os resultados para os *datasets* foram competitivos aos obtidos com abordagens utilizando *Deep Learning*. Mais detalhes sobre os resultados dos melhores modelos para

Figura 5.3 – Semelhança entre as ações do *dataset* KSCGR, à esquerda a ação de *Breaking* e à direita a ação *Peeling*.

as abordagens *baselines* implementadas estão presentes nas seções dos *datasets* DogCentric (Seção 5.3.3) e UCF-11 (Seção 5.3.4).

Nós também enfrentamos uma série de problemas com o modelo de redes neurais 3D devido à nossa restrição de hardware e aos nossos *datasets* pequenos. A memória necessária para treinar a rede ultrapassa a disponível para nós. Uma vez que o uso de uma C3D implica na criação de uma sequência de imagens como entrada e a arquitetura original proposta por Tran *et al.* [TBF+15] consiste em oito camadas convolucionais, decidimos fazer duas modificações. Primeiro, decidimos reduzir o tamanho da imagem de 240×240 a 64×64. Isso nos dá uma contribuição de 64×64×16, o que reduz a entrada em 92%. Em segundo lugar, removemos 4 camadas convolucionais e 2 camadas de *pooling*, o que representa um total de 3.136.779 parâmetros, em comparação com a arquitetura original a qual possui 52.870.922 parâmetros.

Apesar das nossas modificações na arquitetura C3D, não obtivemos bons resultados em nossos experimentos. Todas as execuções resultaram em *overfitting*, nas quais as primeiras épocas obtiveram uma acurácia de 98% e uma acurácia de validação em torno de 10% e mantiveram o mesmo padrão durante todo o treinamento. O *overfitting* possui dois motivos principais para ocorrer: primeiro, estamos usando imagens menores que as usuais, o que reduz a quantidade de informações de cada imagem. Isso dificulta o processo de aprendizado, pois com menos informações a rede se torna incapaz de separar elementos de diferentes classes com sucesso. O segundo motivo é o tamanho de nossos *datasets*. Uma rede neural como a C3D requer uma grande quantidade de dados para poder treinar adequadamente um modelo. Os *datasets* utilizados neste trabalho são pequenos e não possuem dados suficientes para que a C3D evite o *overfitting*. Devido a esses resultados, decidimos não relatar nossos experimentos utilizando a C3D, uma vez que a rede não conseguiu aprender informações suficientes do *dataset* a fim de obter bons resultados.

5.3.2 KSCGR

Apesar do *dataset* KSCGR apresentar algumas classes muito parecidas entre si, conforme ilustrado na Figura 5.3, ainda assim conseguimos atingir resultados relevantes. Mesmo sendo um *dataset* pequeno, os melhores modelos conseguiram atingir em torno de 73% de acurácia, 75% de precisão, 73% de revocação e 71% de F1 na fase de teste. A Tabela 5.3 apresenta o melhor resultado de teste para cada uma das arquiteturas treinadas utilizando o *dataset* KSCGR, bem como para os *baselines* implementados e para os melhores resultados encontrados na literatura. Podemos verificar que ao utilizar a arquitetura V3, sendo para extração de *features* ou diretamente, obtemos os melhores resultados quando comparados com os resultados que utilizaram a arquitetura VGG16. A diferença desses resultados chegam a atingir uma diferença de $\approx$16% de acurácia, quando comparados os modelos V3-FC-LRCN e VGG16-FC-LRCN. Essa diferença segue para os outros modelos apresentados e também para outras métricas, tais como a precisão, a revocação, e a F1.

Ao utilizarmos o algoritmo SVM com as *features* da arquitetura V3, resultaram-se em um dos melhores resultados gerados pelos experimentos, ficando atrás somente do modelo que utiliza redes recorrentes em sua arquitetura, com uma diferença apenas de $\approx$1% em relação ao melhor modelo (V3-FC-LRCN).

O melhor resultado com modelos estáticos durante a fase de validação foi obtido utilizando uma arquitetura V3 com o *dropout* de 0,90 e o LR de 5e-3. O fato do valor de *dropout* ser alto justifica a ideia de evitar o *overfitting*, auxiliando a generalização dos neurônios no modelo e permitindo que o mesmo seja capaz de obter bons resultados na fase de teste. Todavia, o valor de *dropout* que obteve melhores resultados independente do LR, foi o de 0,70. Da mesma forma que o melhor valor para LR foi o de 5e-3 independente do valor de *dropout*. Já para os modelos temporais o melhor resultado de validação obteve-se utilizando as *features* da última camada totalmente conectada de uma rede V3 em uma arquitetura LRCN (V3-FC-LRCN) com o valor de 0,95 de *dropout* e o LR de 1e-4. Os resultados de validação dos modelos treinados com a arquitetura LRCN podem ser vistos na Tabela 5.2 e os resultados para os modelos treinados com as arquiteturas estáticas podem ser vistos na Tabela 5.1.

Como foi mencionado anteriormente a arquitetura V3 se destaca quando utilizada com o *dataset* KSCGR. Observamos que a utilização da camada de convolução (conv) ou a camada totalmente conectada (fc) resultaram em valores próximos de acurácia. Com isso, é mais prático utilizarmos os valores da camada de conv, uma vez que a quantidade de *features* é muito menor do que a da camada fc. Em questões de praticidade e capacidade computacional, deverá levar-se em conta o modelo V3-FC-SVM, uma vez que esse obteve o segundo melhor resultado obtido entre os experimentos e necessita muito menos tempo de treinamento em relação ao melhor modelo (V3-FC-LRCN).

Em relação aos resultados estado-da-arte apresentados na Tabela 5.3, ambos os autores atingem resultados relevantes para o *dataset* KSCGR, onde um deles utiliza abordagens de DL. Bansal *et al.* [BKGG13] apresentam uma abordagem a qual utiliza SVM e HMM com *handcrafted features* para a classificação das ações, chegando a atingir 64% de acurácia, e após uma abordagem de pós-processamento passam a obter 68% de acurácia. Monteiro *et al.* [MGBM17] utilizam fusões dos resultados de 3 redes convolucionais para gerar a classificação das ações do *dataset* KSCGR. Após a fusão dos resultados das CNNs escolhidas pelo autor, os resultados então são passados para um modelo recorrente (LSTM) o qual irá gerar a classificação final. O autor também optou por utilizar uma técnica de pós-processamento nos seus resultados, que variaram de 78% a 79% de acurácia, respectivamente com e sem pós-processamento. A diferença entre o melhor resultado do estado-da-arte e o melhor resultado obtido por nós neste trabalho é de ≈6% de acurácia e ≈5% de precisão. Consideramos este resultado positivo, pois com apenas um modelo conseguimos atingir um resultado tão bom quanto a fusão de 3 modelos em uma rede recorrente. Este fato pode ter ocorrido devido a otimização de parâmetros que realizamos no processo de busca do melhor modelo para resolver o problema.

Tabela 5.1 – Acurácia de validação para as arquiteturas V3 e VGG16 utilizando o *dataset* KSCGR

Dropout	Learning Rate	V3-RGB	V3-OFL	VGG16-RGB	VGG16-OFL
0,50	5e-3	0,6978	**0,6842**	0,2916	**0,4234**
	1e-3	0,6338	0,6019	**0,6216**	0,4229
	5e-4	0,6076	0,6249	0,6056	0,4190
	1e-4	0,5961	0,5985	0,5946	0,3621
	5e-5	0,6294	0,5710	0,5823	0,3794
	1e-5	0,5903	0,4342	0,6115	0,3426
0,70	5e-3	0,7532	0,6569	0,2961	0,3866
	1e-3	0,6377	0,6273	0,6193	0,3929
	5e-4	0,6299	0,6049	0,6041	0,4076
	1e-4	0,6087	0,5978	0,6063	0,3853
	5e-5	0,6117	0,5297	0,5790	0,3509
	1e-5	0,6050	0,3506	0,5950	0,3387
0,90	5e-3	**0,8320**	0,6418	0,2991	0,2976
	1e-3	0,7024	0,6171	0,3823	0,3777
	5e-4	0,6383	0,6580	0,5859	0,3671
	1e-4	0,5896	0,6011	0,5665	0,3485
	5e-5	0,5768	0,5221	0,5656	0,3405
	1e-5	0,5518	0,4093	0,5459	0,3251
0,95	5e-3	0,7156	0,6392	0,3017	0,2948
	1e-3	0,6413	0,6162	0,2974	0,3180
	5e-4	0,6976	0,6615	0,3005	0,3065
	1e-4	0,5825	0,5779	0,5595	0,3097
	5e-5	0,5734	0,5039	0,5541	0,2967
	1e-5	0,5608	0,3162	0,5203	0,2985

Tabela 5.2 – Acurácia de validação para a arquitetura LRCN utilizando o *dataset* KSCGR

Dropout	Learning Rate	VGG16-CONV-LRCN	VGG16-FC-LRCN	V3-CONV-LRCN	V3-FC-LRCN
0,50	5e-3	0,5498	0,5729	0,7476	0,7621
	1e-3	0,6801	0,6390	0,7633	0,7691
	5e-4	0,6626	0,6430	0,7866	0,7771
	1e-4	0,5995	0,6489	0,7969	0,8132
	5e-5	0,6259	0,5960	0,7837	0,8146
	1e-5	0,6166	0,5931	0,7906	0,7879
0,70	5e-3	0,6079	0,5509	0,7537	0,7619
	1e-3	**0,6859**	0,6578	0,7796	0,7834
	5e-4	0,6174	0,6589	0,7897	0,7859
	1e-4	0,6101	**0,7009**	0,8064	0,8155
	5e-5	0,6175	0,6414	0,8055	0,8166
	1e-5	0,6235	0,6581	0,7897	0,7903
0,90	5e-3	0,5474	0,5495	0,7608	0,7669
	1e-3	0,6499	0,6093	0,7937	0,7958
	5e-4	0,6400	0,6558	0,7985	0,8042
	1e-4	0,6582	0,6944	0,8125	0,8189
	5e-5	0,6690	0,6570	0,8066	0,8183
	1e-5	0,6532	0,6571	0,8130	0,8145
0,95	5e-3	0,5264	0,5444	0,7581	0,7691
	1e-3	0,6171	0,6325	0,7933	0,8010
	5e-4	0,6270	0,6605	0,8058	0,8110
	1e-4	0,5739	0,6595	**0,8150**	**0,8203**
	5e-5	0,6154	0,6150	0,8126	0,8107
	1e-5	0,5623	0,6236	0,8109	0,8032

Tabela 5.3 – Resultados de teste dos melhores modelos utilizando o *dataset* KSCGR.

Tipo	Arquitetura	Acurácia	Precisão	Revocação	*F-Score*
Deep	V3-RGB	0,615	**0,773**	0,615	0,660
	V3-OF	0,630	0,650	0,630	0,600
	V3-FC-LRCN	**0,733**	0,750	**0,730**	**0,710**
	V3-CONV-LRCN	0,729	0,740	**0,730**	0,700
	V3-FC-SVM	0,723	0,740	0,720	0,680
	V3-CONV-SVM	0,713	0,730	0,710	0,670
	V3-CONCAT-TWO-STREAM	0,722	0,750	0,720	0,680
	V3-MEAN-TWO-STREAM	0,721	0,740	0,720	0,690
	VGG16-RGB	0,592	0,590	0,590	0,550
	VGG16-OF	0,415	0,460	0,420	0,370
	VGG16-FC-LRCN	0,571	0,560	0,570	0,550
	VGG16-CONV-LRCN	0,560	0,600	0,560	0,510
	VGG16-FC-SVM	0,563	0,600	0,560	0,480
	VGG16-CONV-SVM	0,499	0,290	0,500	0,360
	VGG16-CONCAT-TWO-STREAM	0,604	0,620	0,600	0,570
	VGG16-MEAN-TWO-STREAM	0,598	0,650	0,600	0,560
Estado-da-arte	HCF [BKGG13]	0.640	0.620	0.630	0.610
	HCF-PP [BKGG13]	0.680	0.680	0.680	0.720
	3 CNNs-LSTM [MGBM17]	0.780	0.780	0.780	0.780
	3 CNNs-LSTM-PP [MGBM17]	**0.790**	**0.800**	**0.780**	**0.790**

5.3.3 DogCentric

Os modelos que se destacaram individualmente para o *dataset* DogCentric foram os modelos temporais, tais como as arquiteturas LRCN e Two-Stream. Apesar das dificuldades relacionadas ao *dataset*, o melhor resultado utilizando *features* da camada de convolução da VGG16 em uma arquitetura de LRCN obteve 60% de acurácia, 64% de precisão, 60% de revocação e 60% de F1. Percebemos que para o *dataset* DogCentric a

arquitetura VGG16 e a V3 mantiveram uma diferença de ≈2% para algumas métricas, o que não torna uma significativamente mais relevante que a outra. Em questão de performance pode-se dar prioridade a arquitetura VGG16, uma vez que esta tem um tempo de treinamento relativamente mais curto que o tempo de treinamento de uma arquitetura V3.

Como pode ser visto nos resultados, as imagens em RGB se destacam em relação as imagens em OF, uma vez que não houveram resultados relevantes obtidos durante a fase de teste utilizando tais imagens. Esses resultados se mantiveram mesmo durante a fase de validação, como pode ser visto na Tabela 5.8 onde são apresentados os resultados para os modelos estáticos, e na Tabela 5.4 que apresenta os resultados obtidos utilizando a arquitetura LRCN. Outro fato que evidencia a relevância das imagens em RGB é o resultado do modelo que utiliza a arquitetura V3 com imagens RGB, o qual obteve 57,1% de acurácia, 68% de precisão, 57% de revocação e 59% de F1, contra 12,8% de acurácia, 3% de precisão, 13% de revocação e 4% de F1 para o modelo que utilizou da mesma arquitetura com imagens em OF. Além disso, o resultado da V3-RGB superou o resultado do modelo que utilizou a arquitetura Two-Stream (V3-CONCAT-TWO-STREAM e V3-MEAN-TWO-STREAM) A Tabela 5.6 apresenta os resultados obtidos durante a fase de teste utilizando o *dataset* DogCentric.

A mesma análise realizada acima se encaixa para os modelos que utilizam a arquitetura VGG16, uma vez que o modelo VGG16-RGB obteve 54% de acurácia, 58,8% de precisão, 54,1% de revocação e 53,8% de F1, enquanto o modelo VGG16-OF atingiu resultados inferiores a 12% em cada métrica.

Para o *dataset* DogCentric os modelos recorrentes, os quais se utilizam da arquitetura LRCN, obtiveram os melhores resultados comparados com os outros modelos. O segundo melhor modelo (Two-Stream), para os modelos que utilizaram a rede V3, permaneceram com uma diferença de ≈2,5% de acurácia, enquanto a diferença para os modelos que utilizaram a arquitetura VGG16 permaneceram com uma diferença de ≈6,5% de acurácia.

Os modelos que utilizam SVM seguem o mesmo padrão de resultados do parágrafo anterior, onde os experimentos que utilizam da arquitetura V3 se sobressaem em relação a rede VGG16 para o *dataset* DogCentric. Com isso, a arquitetura V3 é a melhor opção para o *dataset* DogCentric, onde essa aprende a extrair *features* mais relevantes para outras combinações de modelos.

Conforme mencionado no Capítulo 4.5, nós implementamos e executamos experimentos utilizando *handcrafted features* e tomamos como *baseline* a abordagem proposta por Wang *et al.* [WKSL11]. Os resultados para o *dataset* DogCentric foram competitivos aos resultados obtidos com abordagens utilizando *Deep Learning*, e tais resultados podem ser vistos na Tabela 5.6. Houveram dois melhores modelos para o *dataset* DogCentric, os quais foram utilizando os descritores HOG e MBHy. Ambos obtiveram resultados semelhantes, mas o modelo com o descritor MBHy se destacou obtendo ≈54% de acurácia, ≈56% de

Tabela 5.4 – Acurácia de validação para a arquitetura LRCN utilizando o *dataset* DogCentric

Dropout	Learning Rate	VGG16-CONV-LRCN	VGG16-FC-LRCN	V3-CONV-LRCN	V3-FC-LRCN
0,50	5e-3	0,7303	0,7209	0,7550	0,7764
	1e-3	0,7033	0,7356	0,7365	0,7271
	5e-4	0,6893	0,7161	0,7297	0,7641
	1e-4	0,6387	0,6947	0,6886	0,7529
	5e-5	0,6726	0,7406	0,6652	0,7298
	1e-5	0,5851	0,6756	0,6496	0,7461
0,70	5e-3	0,6856	**0,7647**	0,7538	0,7730
	1e-3	0,6962	0,7292	0,7242	0,7308
	5e-4	0,7029	0,7441	0,7258	0,7376
	1e-4	0,6703	0,7408	0,7248	0,7212
	5e-5	0,6692	0,7171	0,7066	0,7261
	1e-5	0,6376	0,7148	0,6481	0,7117
0,90	5e-3	0,6757	0,7114	0,7528	0,7637
	1e-3	**0,7496**	0,7239	0,7342	0,7377
	5e-4	0,6826	0,7276	0,7329	0,7482
	1e-4	0,6274	0,7029	0,7597	0,7120
	5e-5	0,6525	0,6908	0,7061	0,7518
	1e-5	0,5934	0,7221	0,7003	0,7260
0,95	5e-3	0,6599	0,7101	**0,7603**	0,7679
	1e-3	0,7100	0,7541	0,7590	0,7599
	5e-4	0,7285	0,7438	0,7209	**0,7766**
	1e-4	0,6492	0,7170	0,7506	0,6955
	5e-5	0,6648	0,6869	0,6892	0,7257
	1e-5	0,6379	0,6115	0,7273	0,6955

precisão, ≈54% de revocação e ≈52% de medida F-Score. A abordagem *baseline*, *Dense Trajectories*, foi inferior em relação ao melhor modelo utilizando DL, onde a diferença foi de ≈10% de acurácia, ≈8% de precisão, ≈10% de revocação e ≈14% de F-Score. Embora o

Tabela 5.5 – Resultados de validação para as arquiteturas V3 e VGG16 utilizando o *dataset* DogCentric

Dropout	Learning Rate	V3-RGB	V3-OFL	VGG16-RGB	VGG16-OFL
0,50	5e-3	0,6332	0,4531	0,7250	0,3385
	1e-3	0,6262	0,4427	**0,7833**	0,3438
	5e-4	0,6388	0,3854	0,6722	0,3594
	1e-4	0,6338	0,2552	0,6000	0,2760
	5e-5	0,6236	0,2195	0,5889	0,1875
	1e-5	0,5650	0,1295	0,4722	0,2708
0,70	5e-3	0,6395	**0,4635**	0,7389	0,3698
	1e-3	**0,6475**	0,3906	0,7722	**0,3802**
	5e-4	0,6275	0,4010	0,7250	0,3594
	1e-4	0,6364	0,2969	0,7222	0,1927
	5e-5	0,6198	0,2083	0,5750	0,2292
	1e-5	0,5463	0,1927	0,5194	0,1562
0,90	5e-3	0,6362	0,3476	0,6194	0,2812
	1e-3	0,6434	0,3281	0,7222	0,2917
	5e-4	0,6350	0,3438	0,7111	0,2760
	1e-4	0,5494	0,1890	0,6954	0,2760
	5e-5	0,5575	0,2396	0,6389	0,2135
	1e-5	0,5219	0,1250	0,5056	0,1250
0,95	5e-3	0,6069	0,3594	0,2111	0,2604
	1e-3	0,6381	0,3229	0,6862	0,2500
	5e-4	0,6357	0,2083	0,6417	0,2083
	1e-4	0,6131	0,2083	0,6333	0,2292
	5e-5	0,5787	0,2292	0,6111	0,1875
	1e-5	0,5088	0,1562	0,4972	0,1302

Tabela 5.6 – Resultados de teste dos melhores modelos utilizando o *dataset* DogCentric.

Tipo	Arquitetura	Acurácia	Precisão	Revocação	*F-Score*
Deep	V3-RGB	0,571	0,680	0,570	0,590
	V3-OF	0,128	0,030	0,130	0,040
	V3-FC-LRCN	0,594	0,670	0,590	0,600
	V3-CONV-LRCN	0,581	0,680	0,580	0,590
	V3-FC-SVM	0,558	0,660	0,560	0,560
	V3-CONV-SVM	0,550	0,650	0,550	0,550
	V3-CONCAT-TWO-STREAM	0,556	0,690	0,560	0,570
	V3-MEAN-TWO-STREAM	0,541	0,670	0,540	0,550
	VGG16-RGB	0,540	0,588	0,541	0,538
	VGG16-OF	0,118	0,110	0,120	0,090
	VGG16-FC-LRCN	0,556	0,590	0,560	0,550
	VGG16-CONV-LRCN	**0,600**	0,640	**0,600**	**0,600**
	VGG16-FC-SVM	0,508	0,620	0,510	0,500
	VGG16-CONV-SVM	0,301	**0,760**	0,300	0,340
	VGG16-CONCAT-TWO-STREAM	0,535	0,620	0,540	0,540
	VGG16-MEAN-TWO-STREAM	0,529	0,610	0,540	0,540
Baseline	Trajectories	0,286	0,330	0,290	0,270
	HOG	0,514	0,540	0,510	0,500
	HOF	0,486	0,510	0,490	0,460
	MBHx	0,514	0,590	0,510	0,520
	MBHy	**0,543**	**0,560**	**0,540**	**0,520**
	Dense	0,495	0,510	**0,500**	0,460
Estado-da-arte	Linear kernel [ITKR14]	0.530	-	-	-
	RBF kernel [ITKR14]	0.540	-	-	-
	Histogram intersection [ITKR14]	0.570	-	-	-
	Multi-channel [ITKR14]	0.610	-	-	-
	PoT+STIP+Cuboid [RRM15]	0.730	-	-	-
	PoT+ITF+STIP+Cuboid [RRM15]	0.740	-	-	-
	PoT+ITF [RRM15]	0.750	-	-	-
	2 CNNs-SVM [MAG$^+$17]	0.680	0.690	0.680	0.690
	2 CNNs-SVM-PP [MAG$^+$17]	**0.760**	**0.740**	**0.760**	**0.750**

modelo utilizando o descritor MBHy tenha obtido resultados relevantes, o melhor modelo de *baseline* gerado não conseguiu atingir resultados competitivos com os melhores modelos de DL.

Em relação aos trabalhados estado-da-arte para o *dataset* DogCentric, Iwashita *et al.* [ITKR14] apresentam uma abordagem semelhante a que utilizamos como *baseline*. As *features* extraídas de diferentes descritores são agrupadas com o conceito de *bag of visual words* e então passadas para um classificador SVM. Na fase de teste os autores obtem resultados que variam de 53% a 61% de acurácia. Enquanto Ryoo *et al.* [RRM15] utilizam uma maneira de representar as imagens através do tempo chamada de séries temporais agrupadas, e após terem gerado essas *features* eles treinam um SVM como classificador. Os autores também combinaram as *features* obtidas com abordagens de outros trabalhos e obtiveram resultados que variam de 73% a 75% de acurácia. Por fim, Monteiro *et al.* [MAG$^+$17] utilizam *features* extraídas de duas redes convolucionais para treinar um classificador SVM. Além disso, os autores propõem um método de pós-processamento na classificação final do problema. Os resultados obtidos com e sem pós-processamento são respectivamente 68% e 76% de acurácia. A maior parte dos resultados obtidos por esses trabalhos possuem resultados superiores em relação aos obtidos por nós, com uma diferença de até 16% de acurácia. Apesar do trabalho que possui o melhor resultado tenha proposto uma

abordagem de *Deep Learning*, os autores utilizaram de fusões de CNNs e um método de pós-processamento para atingir tais resultados, ao contrário dos nossos experimentos que utilizam *features* apenas de uma única arquitetura.

5.3.4 UCF-11

De todos os 3 *datasets* utilizados neste trabalho, os modelos para o *dataset* UCF-11 foram os que atingiram os melhores resultados. Este fato pode ter ocorrido por diversos fatores, tais como tamanho do *dataset*. Apesar de ele ser considerado pequeno, o UCF-11 é relativamente maior que os outros *datasets* utilizados nos experimentos. Outro motivo pode ser pelo fato do UCF-11 se passar em diversos tipos de cenários, possuindo diferentes fundos para os seus vídeos, o que auxilia no processo de separação das classes nos algoritmos de classificação.

Os melhores resultados obtidos durante a fase de validação concentram-se em ≈88% utilizando modelos recorrentes e ≈86% para modelos estáticos, tal como o modelo V3-RGB. Os resultados da fase de validação para os modelos estáticos podem ser vistos na Tabela 5.8, já os resultados mencionados para os modelos recorrentes podem ser vistos na Tabela 5.7. Já na fase de teste o melhor modelo obteve 78,6% de acurácia, 80% de precisão, 79% de revocação e 78% de F1, utilizando uma rede V3 com a média das saídas de imagens em RGB e OF em uma SVM (V3-MEAN-TWO-STREAM). A maior parte dos modelos atingiram bons resultados, onde a menor diferença encontra-se em ≈3%. Os resultados mencionados da fase de teste estão apresentados na Tabela 5.9.

Apesar do melhor resultado ter sido obtido através de uma arquitetura Two-Stream, pode-se observar que esses resultados foram obtidos utilizando como *features* a média das saídas das redes que utilizaram imagens em RGB e OF. A maior diferença entre os resultados ocorreram utilizando a rede V3, pois a diferença entre os resultados gerados pelos modelos com a concatenação das saídas (V3-CONCAT-TWO-STREAM) e com a média das saídas (V3-CONCAT-TWO-STREAM) são de aproximadamente 3%. O resultado inferior, gerado pela rede que utilizou a concatenação das saídas, se equivale a uma rede estática utilizando uma SVM como algoritmo classificador (V3-CONV-SVM e V3-FC-SVM). Por isso, destaca-se o uso da média das saídas das redes como *feature* para a arquitetura Two-Stream, ao invés da utilização da concatenação.

Os modelos obtidos utilizando as representações em *optical flow* atingiram resultados significativos (63% para o modelo V3-OF) quando comparados com os mesmos testes utilizando o *dataset* DogCentric, mas não superior aos resultados obtidos ao utilizar o *dataset* KSCGR. Esse fato pode ter ocorrido devido a forte movimentação de câmera que o *dataset* DogCentric possui, dificultando a captura do OF. O fato dos resultados obtidos terem sido inferiores ao resultado obtido pelo *dataset* KSCGR pode ser explicado devido aos

vídeos terem sido gravados através de uma câmera estática, fazendo com que as *features* obtidas pelo OF sejam mais relevantes.

Tabela 5.7 – Acurácia de validação para a arquitetura LRCN utilizando o *dataset* UCF-11

Dropout	Learning Rate	VGG16-CONV-LRCN	VGG16-FC-LRCN	V3-CONV-LRCN	V3-FC-LRCN
0,50	5e-3	0,7561	0,7916	0,8592	**0,8728**
	1e-3	0,7688	0,7799	0,8707	0,8625
	5e-4	0,7408	0,7792	**0,8821**	0,8616
	1e-4	0,7277	**0,7942**	0,8676	0,8695
	5e-5	0,7303	0,7431	0,8545	0,8590
	1e-5	0,7337	0,7647	0,8516	0,8345
0,70	5e-3	0,8107	0,7828	0,8622	0,8671
	1e-3	0,7914	0,7703	0,8590	0,8658
	5e-4	0,7856	0,7755	0,8507	0,8658
	1e-4	0,7819	0,7709	0,8640	0,8585
	5e-5	0,7564	0,7699	0,8533	0,8571
	1e-5	0,6795	0,7666	0,8512	0,8534
0,90	5e-3	**0,8218**	0,7925	0,8695	0,8642
	1e-3	0,8056	0,7809	0,8576	0,8585
	5e-4	0,7760	0,7816	0,8608	0,8622
	1e-4	0,7778	0,7834	0,8590	0,8538
	5e-5	0,7515	0,7654	0,8518	0,8585
	1e-5	0,7269	0,7493	0,8503	0,8478
0,95	5e-3	0,8104	0,7903	0,8642	0,8612
	1e-3	0,7741	0,7726	0,8658	0,8673
	5e-4	0,7907	0,7647	0,8653	0,8623
	1e-4	0,7779	0,7636	0,8632	0,8605
	5e-5	0,7475	0,7607	0,8647	0,8670
	1e-5	0,6532	0,7488	0,8565	0,8475

Tabela 5.8 – Resultados de validação para as arquiteturas V3 e VGG16 utilizando o *dataset* UCF-11

Dropout	Learning Rate	V3-RGB	V3-OFL	VGG16-RGB	VGG16-OFL
0,50	5e-3	0,8568	0,6664	0,7785	0,4890
	1e-3	0,8412	0,6483	0,7709	0,4912
	5e-4	0,8266	0,6746	0,7737	0,4853
	1e-4	0,8272	0,5963	0,7441	0,4379
	5e-5	0,8432	0,5881	0,7424	0,4342
	1e-5	0,6873	0,3924	0,6825	0,3877
0,70	5e-3	0,8088	**0,6771**	0,7497	0,4802
	1e-3	0,8540	0,6638	0,7726	**0,5017**
	5e-4	0,8528	0,6421	0,7582	0,4900
	1e-4	0,8331	0,6197	0,7444	0,4452
	5e-5	0,7819	0,5938	0,7573	0,3949
	1e-5	0,7184	0,3701	0,6652	0,3370
0,90	5e-3	0,7873	0,5729	**0,7893**	0,3718
	1e-3	0,8362	0,6562	0,7718	0,4362
	5e-4	0,8568	0,6650	0,7320	0,4571
	1e-4	0,8472	0,6141	0,7232	0,4219
	5e-5	0,8356	0,5137	0,6904	0,3895
	1e-5	0,6890	0,2669	0,6960	0,2989
0,95	5e-3	0,6915	0,5887	0,1782	0,2079
	1e-3	**0,8579**	0,6333	0,7525	0,4110
	5e-4	0,8243	0,6477	0,7638	0,4160
	1e-4	0,8291	0,5836	0,6828	0,2986
	5e-5	0,8328	0,4567	0,7314	0,3011
	1e-5	0,6424	0,2368	0,6260	0,2681

Tabela 5.9 – Resultados de teste dos melhores modelos utilizando o *dataset* UCF-11.

Tipo	Arquitetura	Acurácia	Precisão	Revocação	*F-Score*
Deep	V3-RGB	0,744	0,760	0,740	0,740
	V3-OF	0,668	0,680	0,670	0,660
	V3-FC-LRCN	0,752	0,770	0,750	0,750
	V3-CONV-LRCN	0,752	0,750	0,750	0,740
	V3-FC-SVM	0,750	0,780	0,750	0,750
	V3-CONV-SVM	0,757	0,790	0,760	0,760
	V3-CONCAT-TWO-STREAM	0,755	0,770	0,760	0,760
	V3-MEAN-TWO-STREAM	**0,786**	**0,800**	**0,790**	**0,780**
	VGG16-RGB	0,708	0,700	0,710	0,700
	VGG16-OF	0,480	0,470	0,480	0,470
	VGG16-FC-LRCN	0,711	0,710	0,710	0,700
	VGG16-CONV-LRCN	0,752	0,750	0,750	0,740
	VGG16-FC-SVM	0,696	0,730	0,700	0,700
	VGG16-CONV-SVM	0,397	0,770	0,400	0,410
	VGG16-CONCAT-TWO-STREAM	0,710	0,730	0,710	0,710
	VGG16-MEAN-TWO-STREAM	0,720	0,720	0,720	0,720
Baseline	Trajectories	0,391	0,440	0,390	0,400
	HOG	0,743	0,790	0,740	0,760
	HOF	0,676	0,720	0,680	0,680
	MBHx	0,695	0,780	0,700	0,700
	MBHy	**0,762**	**0,830**	**0,760**	**0,780**
	Dense	0,760	0,780	**0,760**	0,750
Estado-da-arte	DTAM [WCDW17]	0.900	-	-	-
	Visual-DTAM [WCDW17]	0.910	-	-	-

Ao gerar a experimentação com as *baselines*, nós atingimos resultados significativos com o *dataset* UCF-11. Dois modelos se destacaram, obtendo aproximadamente o mesmo resultado de acurácia. Um dos modelos obteve-se utilizando o descritor Dense (combinação de todos os outros descritores), abordagem proposta por Wang *et al.* [WKSL11], o qual atingiu $\approx$ 76% de acurácia. O outro modelo utilizou apenas o descritor MBHy e obteve também $\approx$ 76% de acurácia. Destaca-se o modelo treinado com as *features* do descritor MBHx, pois foi treinado com um $\approx$ 5 vezes menos *features* em relação ao *Dense*. Ao comparar com os modelos de DL, os modelos de *baseline* são inferiores apenas por $\approx$ 2% de acurácia, todavia, ambos obtiveram 78% de F-Score.

Um dos trabalhos dispostos na literatura utilizando o *dataset* UCF-11 destaca-se por ter obtido resultados superiores a 90% de acurácia utilizando técnicas de *Deep Learning* com *handcrafted features*. A ideia proposta por Wang *et al.* [WCDW17] utiliza uma abordagem chamada de Modelo de Atenção de Rastreamento Dinâmico (do inglês, *Dynamic Tracking Attention Model, DTAM*). A abordagem é composta por uma CNN e uma LSTM para reconhecer a ação humana em sequências de vídeos. Enquanto a CNN realiza a extração das *features*, a LSTM é aplicada para lidar com informações sequenciais sobre ações que são extraídas de vídeos. O DTAM usa rastreamento dinâmico local para identificar objetos em movimento e rastreamento dinâmico global para estimar o movimento da câmera e corrigir os pesos do modelo de atenção de movimento. Utilizando uma fusão com a abordagem de Atenção Visual [MHGk14] e seu DTAM proposto, eles atingem 91% de acurácia. Valores de precisão, revocação e F-score não são relatados em seu trabalho.

5.4 Discussão

Neste capítulo apresentamos os experimentos realizados utilizando os *datasets* KSCGR, DogCentric e UCF-11. Foram realizados um total de 672 experimentos envolvendo técnicas de *deep learning* e 726 experimentos com *handcrafted features* como *baseline* para classificar ações. Os melhores resultados em geral foram atingidos utilizando abordagens temporais através da rede V3 com as arquiteturas LRCN e Two-Stream.

Verificamos que as abordagens de *deep learning* utilizadas neste trabalho atingiram bons resultados para os *datasets* pequenos testados quando comparados com as abordagens *baselines* que implementamos e com os resultados estado-da-arte disponíveis na literatura. Com isso, comprovamos a nossa primeira hipótese, onde levantamos o questionamento se abordagens de DL iriam atingir resultados relevantes em *datasets* pequenos para reconhecimento de ações. Também observamos que as abordagens de DL foram sempre superiores as abordagens de *handcrafted features* utilizadas como *baselines* nesta dissertação. Em alguns casos apresentados anteriormente as abordagens de *baseline* se aproximaram dos resultados das que utilizam técnicas de *deep learning*, todavia, nunca atingindo resultados superiores aos melhores modelos. Para maior parte dos experimentos executados, quando utilizamos *handcrafted features* com *features* extraídas de uma rede convolucional, tal como o uso da arquitetura Two-Stream, nós observamos que os resultados são superiores a usar qualquer uma das duas *features* separadamente em uma rede convolucional. Dado este fato, também observamos nos trabalhos estado-da-arte que a maior parte deles utilizam tais fusões de *features handcrafted* com *features deep* para atingir melhores resultados [RRM15, WCDW17]. Desse modo, admitimos a nossa segunda hipótese levantada, a qual se diz respeito que na utilização de *features* de DL com *features handcrafted* os resultados serão superior que qualquer uma dessas isoladamente. Por último, conseguimos validar a hipótese que afirma que modelos que levam em conta o aspecto temporal atingem resultados superiores quando comparados com modelos estáticos. Para os 3 *datasets* utilizados nesta dissertação os melhores modelos utilizaram do aspecto temporal para atingirem tais resultados. Neste caso as arquiteturas utilizadas que levaram em conta a temporalidade do problema foram a Two-Stream e a LRCN.

Não podemos ignorar os desafios que cada *dataset* possui, porém, conseguimos acompanhar certos padrões que nos levam a sugerir algumas práticas que o leitor possa aderir quando utilizar *datasets* pequenos para reconhecimento de ações. Apesar deste trabalho não cobrir o número de *datasets* que gostaríamos, e não conseguirmos fornecer detalhes de implementação mais genéricos para o leitor, nós fornecemos a seguir algumas práticas que podem ser adotadas pelos usuários quando os mesmos utilizarem *datasets* com características semelhantes aos utilizados por este trabalho.

Das arquiteturas testadas nesta dissertação a Inception-V3 [SVI+16] foi a que forneceu os melhores resultados para a maior parte dos *datasets* testados. Somente ao utilizar o *dataset* DogCentric que a arquitetura VGG-16 atingiu resultados melhores em apenas 2% de acurácia. Como já explicado, o *dataset* DogCentric é o menor *dataset* que utilizamos nesta dissertação, e além disso ele possui algumas características que dificultam a classificação de suas ações, tais como uma forte movimentação de câmera e imagens egocêntricas. Todavia, observou-se um padrão, onde sugerimos que para os *datasets* com estas características se utilize os valores de *dropout* entre $0,50$ e $0,70$. Já para os *datasets* com as características mais próximas ao KSCGR e UCF-11 onde as classes são mais distintas umas das outras, possua imagens estáticas ou com uma movimentação de câmera mais leve, recomenda-se a utilização de valores mais altos para *dropout*, tais como $0,90$ e $0,95$, os quais resultaram nos melhores modelos durante nossos testes. Para o hiperparâmetro *learning rate* recomenda-se utilizar 5e-3 ou 1e-3, pois foram os melhores valores encontrados durante nossa fase de otimização, levando em conta os três *datasets* testados neste trabalho.

Em relação a qual abordagem utilizar, relatamos que para os três *datasets* utilizados os melhores resultados foram obtidos através de modelos temporais, sendo a utilização de uma arquitetura LRCN para os *datasets* DogCentric e KSCGR, e uma arquitetura Two-Stream para o *dataset* UCF-11. Por isso, mesmo lidando com *datasets* pequenos, recomendamos a utilização de abordagens que levam em conta o aspecto temporal. Para os *datasets* aos quais o movimento de câmera não seja tão forte e as ações não sejam tão semelhantes, recomenda-se a utilização da abordagem Two-Stream. Uma vez que esta abordagem irá conseguir extrair de maneira mais eficaz as *features* de *optical flow* das imagens, e fundi-las com as *features* extraídas pela rede convolucional, resultando em uma possível classificação mais eficaz ao final do processo.

Verificamos que a utilização de *handcrafted features* juntamente com abordagens de *Deep Learning* podem agregar aos resultados finais em uma classificação de ações, todavia, o processo de extração de *features* e fusão de abordagens não é um processo simples, muito menos rápido. Caso o usuário busque resolver o problema em um curto espaço de tempo utilizando *datasets* pequenos, nós recomendamos que o usuário priorize a utilização de abordagens de DL uma vez que estas abordagens já possuem os mecanismos de aprender as *features* automaticamente e classificá-las, se tornando uma prática muito menos custosa e mais simples em vias de implementação. Caso tempo e capacidade computacional não seja um problema, as abordagens apresentadas pelos trabalhos estado-da-arte e os nossos experimentos mostram que ao utilizar de fusões de abordagens de DL juntamente com *handcrafted features* os modelos serão capazes de obter resultados superiores a abordagens isoladas.

6. CONCLUSÃO

Este trabalho apresentou uma análise profunda utilizando algoritmos de *deep learning* em *datasets* pequenos de ações. Foram executados mais de 672 experimentos utilizando diferentes abordagens de DL. Também foram testadas diferentes arquiteturas de redes convolucionais, diferentes parâmetros, métricas e abordagens de *handcrafted features*, a fim de possuir uma análise mais completa sobre os *datasets* utilizados.

Utilizamos duas principais arquiteturas de redes convolucionais, as redes V3 e VGG16, as quais se mostraram promissoras utilizando tanto a classificação direta, quanto a extração de *features* utilizadas por outros modelos. Utilizamos ambas as arquiteturas com pré-treinos de *datasets* maiores, como boa prática apresentada já estabelecida na literatura. Com isso, também testamos modelos temporais com a intenção verificar a viabilidade desses modelos para *datasets* pequenos.

Os melhores modelos para os *datasets* escolhidos concentraram-se em modelos temporais, onde a arquitetura LRCN obteve os melhores resultados para os *datasets* Dog-Centric e KSCGR, enquanto a arquitetura Two-Stream, a qual faz uso de imagens em RGB e em OF, obteve os melhores resultados para o *dataset* UCF-11. Porém, como explicado no Capítulo 5, para alguns *datasets* a diferença entre os resultados dos melhores modelos de diferentes arquiteturas foi infímo, podendo nos levar a escolha de modelos mais simples levando em conta a performance desses modelos.

Também analisamos o impacto do tipo de entrada nos diferentes modelos. Para a maior parte das arquiteturas e dos *datasets* notou-se que o uso das imagens em RGB resultaram em um ganho em acurácia, precisão, revocação e F1, muito maior do que o uso de imagens em OF. Também percebeu-se relações entre o uso dos parâmetros de *dropout* e *learning rate*, uma vez que esses impactam diretamente nos resultados dos modelos gerados.

Apesar dos desafios que cada *dataset* possui, acreditamos que os modelos de *deep learning* utilizados neste trabalho apresentaram bons resultados. Além disso, demonstramos o impacto desde o uso de diferentes entradas para esses modelos, tanto como o uso de diferentes valores de parâmetros. Discutimos a respeito de utilizar arquiteturas que compreendem os aspectos temporais e estáticos, bem como se dado o resultado obtido pelo melhor modelo se ainda sim seria válido utilizá-lo. Também analisamos as *features* geradas pela abordagem *baseline* escolhida e observamos que essas não seriam capazes de gerar bons resultado se enviadas para algum algoritmo classificador.

6.1 Limitações

O estudo feito nesta dissertação possui algumas limitações que impedem que tenhamos uma conclusão mais ampla referente aos resultados obtidos. Umas das limitações deste trabalho é que realizamos os experimentos somente em 3 *datasets*, que por sua vez, nos limita a capacidade de poder generalizar as nossas conclusões para a maior parte das situações. Com isso as conclusões que foram apresentadas se limitam aos *datasets* que possuam características semelhantes aos utilizados em nossos experimentos.

Um empecilho também foi a indisponibilidade de recursos de hardware, o que causou uma grande demanda de GPUs para executar os nossos experimentos. Uma vez que não possuímos GPUs em abundância, levou-se demasiado tempo para que executássemos todos os testes necessários. Devido a tal limitação, também não fomos capazes de executar em sua real capacidade a arquitetura C3D, e outras arquiteturas de trabalhos mais recentes, como por exemplo a arquitetura I3D apresentada por Carreira *et al.* [CZ17], o qual faz uso de 32 GPUs simultaneamente.

Também pode ser visto como uma limitação o fato das arquiteturas Two-Stream e LRCN não serem *end-to-end*, ou seja, os modelos não foram treinados do início ao fim. No caso da Two-Stream primeiramente treinaram-se dois modelos, um com imagens em RGB e outro com imagens em OF, após isso, fez-se a fusão dos resultados como exemplifica os autores dessa arquitetura. Após tal fusão, treinou-se um modelo classificador com esses dados. O fato do treinamento de ambas as redes e a fusão dos dados não terem ocorrido paralelamente, faz com que exista um possível *gap* de perda de informações, fazendo com que ocorra um possível depreciação no resultado final do modelo. O mesmo pode ter ocorrido para a arquitetura LRCN, a qual utiliza de *features* gerados por uma CNN para alimentar a rede recorrente.

6.2 Trabalhos Futuros

Como trabalho futuro, nós pretendemos desenvolver abordagens de *deep learning* capazes de suprir as limitações encontradas nos *datasets* pequenos, como por exemplo o *overfitting*. Evitar o *overfitting* não é uma tarefa fácil, por isso iremos construir uma estrutura capaz de lidar bem com regularização e que não possua parâmetros em excesso. Pretendemos também explorar as arquiteturas existentes a fim de criarmos uma que se adapte melhor a esse tipo de dados. Uma ideia ligada a essa é construir redes convolucionais a partir de algoritmos evolutivos (*Evolving Deep Neural Networks*) [RAHL18], sendo promissor para conseguirmos construir arquiteturas que se adaptem bem a este tipo de domínio. Não obstante, temos a intenção de testar outras abordagens de *deep learning*, tais como

a proposta por Carreira *et al.* [CZ17], as quais não foram testadas neste trabalho dado uma limitação de tempo e capacidade computacional. Também gostaríamos de utilizar as abordagens testadas neste trabalho em outros *datasets* pequenos, para que seja possível expandir a análise feita.

6.3 Publicações

Esta seção apresenta os trabalhos aceitos que desenvolvemos durante o período desta pesquisa. Vale ressaltar que o artigo referente a este trabalho foi submetido para avaliação na International Joint Conference on Neural Networks (IJCNN) de 2018.

1. **Monteiro, J.**; Aires, J. P. S.; Granada, R.; Barros, R. C.; Meneguzzi, F. R. Virtual Guide Dog: An Application to Support Visually-Impaired People through Deep Convolutional Neural Networks. In: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, Anchorage. (**Best Student Paper Award**) (Qualis A1)

2. **Monteiro, J.**; Granada, R.; Barros, R. C.; Meneguzzi, F. R. Deep Neural Networks for Kitchen Activity Recognition. In: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, Anchorage (Qualis A1)

3. Granada, R.; Pereira, R. F.; **Monteiro, J.**; Barros, R. C.; Ruiz, D.; Meneguzzi, F. R. Hybrid Activity and Plan Recognition for Video Streams. In: AAAI Conference on Artificial Intelligence (AAAI-17). Plan, Activity and Intent Recognition workshop, 2017, San Francisco, CA. (Qualis A1)

4. Aires, J. P. S.; **Monteiro, J.**; Granada, R.; Meneguzzi, F. R.; Barros, R. C. Improving Activity Recognition using Temporal Regions. In: Symposium on Knowledge Discovery, Mining and Learning, 2017, Uberlândia, MG. Proceedings of the 5th Symposium on Knowledge Discovery, Mining and Learning (KDMiLe), 2017. (Qualis B3)

5. Silva, L. P.; Granada, R.; **Monteiro, J.**; Ruiz, D. Using Scene Context to Improve Object Recognition. In: Symposium on Knowledge Discovery, Mining and Learning, 2017, Uberlândia, MG. Proceedings of the 5th Symposium on Knowledge Discovery, Mining and Learning (KDMiLe), 2017. (Qualis B3)

6. Granada, R.; **Monteiro, J.**; Barros, R. C.; Meneguzzi, F. R. A Deep Neural Architecture for Kitchen Activity Recognition. In: The Florida Artificial Intelligence Research Society Conference (FLAIRS), 2017, Marco Island, Florida. (Qualis B1)

7. Maidana, R. ; **Monteiro, J.** ; Granada, R.; Amory, A.; Barros, R. C. Deep Neural Networks for Handwritten Chinese Character Recognition. In: Brazilian Conference on Intelligent Systems (BRACIS), 2017, Uberlândia, MG. (Qualis B2)

8. Aires, J. P. S.; **Monteiro, J.**; Granada, R.; Barros, R. C.; Meneguzzi, F. R. Temporal Regions for Activity Recognition. In: 26th International Conference on Artificial Neural Networks (ICANN), 2017, Alghero, Sardinia. (Qualis B1)

9. Simões, G. S.; Wehrmann, J.; Paula, T. S.; **Monteiro, J.**; Barros, R. C. DataSex: um dataset para indução de modelos de classificação para conteúdo adulto. In: Symposium on Knowledge Discovery, Mining and Learning (KDMiLe), 2016, Recife. (Qualis B2)

REFERÊNCIAS BIBLIOGRÁFICAS

[ABC+16] Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; Zheng, X. "Tensorflow: A system for large-scale machine learning". In: Proceedings of the USENIX Conference on Operating Systems Design and Implementation, 2016, pp. 265–283.

[Alp14] Alpaydin, E. "Introduction to machine learning". MIT press, Massachusetts, USA, 2014, vol. 2.

[AR11] Aggarwal, J. K.; Ryoo, M. S. "Human activity analysis: A review", *ACM Computing Surveys (CSUR)*, vol. 43–3, 2011, pp. 16.

[AYX+08] Ahmed, A.; Yu, K.; Xu, W.; Gong, Y.; Xing, E. "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks". In: Proceedings of European Conference on Computer Vision (ECCV), 2008, pp. 69–82.

[BD01] Bobick, A. F.; Davis, J. W. "The recognition of human movement using temporal templates", *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23–3, 2001, pp. 257–267.

[Ben09] Bengio, Y. "Learning deep architectures for ai", *Foundations and Trends in Machine Learning*, vol. 2–1, 2009, pp. 1–127.

[BGS+05] Blank, M.; Gorelick, L.; Shechtman, E.; Irani, M.; Basri, R. "Actions as space-time shapes". In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2005, pp. 1395–1402.

[BKGG13] Bansal, S.; Khandelwal, S.; Gupta, S.; Goyal, D. "Kitchen activity recognition based on scene context". In: Proceedings of the IEEE International Conference on Image Processing (ICIP), 2013, pp. 3461–3465.

[CCFC13] Chaquet, J. M.; Carmona, E. J.; Fernández-Caballero, A. "A survey of video datasets for human action and activity recognition", *Computer Vision and Image Understanding*, vol. 117–6, Jun 2013, pp. 633–659.

[CLS15] Ch'eron, G.; Laptev, I.; Schmid, C. "P-cnn: Pose-based cnn features for action recognition". In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 3218–3226.

[CMS12] Cireşan, D. C.; Meier, U.; Schmidhuber, J. “Transfer learning for latin and chinese characters with deep neural networks”. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–6.

[CNW12] Chen, L.; Nugent, C. D.; Wang, H. “A knowledge-driven approach to activity recognition in smart homes”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 24–6, 2012, pp. 961–974.

[CZ17] Carreira, J.; Zisserman, A. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4724–4733.

[DDS+09] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. “Imagenet: A large-scale hierarchical image database”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255.

[DHR+17] Donahue, J.; Hendricks, L. A.; Rohrbach, M.; Venugopalan, S.; Guadarrama, S.; Saenko, K.; Darrell, T. “Long-term recurrent convolutional networks for visual recognition and description”, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39–4, April 2017, pp. 677–691.

[DXDC13] Das, P.; Xu, C.; Doell, R. F.; Corso, J. J. “A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2634–2641.

[DY14] Deng, L.; Yu, D. “Deep learning: Methods and applications”, *Foundations and Trends in Signal Process*, vol. 7–3–4, Jun 2014, pp. 197–387.

[Far03] Farnebäck, G. “Two-frame motion estimation based on polynomial expansion”. In: Proceedings of the Scandinavian Conference on Image Analysis (SCIA), 2003, pp. 363–370.

[FI10] Fiaz, M. K.; Ijaz, B. “Vision based human activity tracking using artificial neural networks”. In: Proceedings of the International Conference on Intelligent and Advanced Systems (ICIAS), 2010, pp. 1–5.

[FPZ16a] Feichtenhofer, C.; Pinz, A.; Zisserman, A. “Convolutional two-stream network fusion for video action recognition”, *arXiv preprint arXiv:1604.06573*, 2016.

[FPZ16b] Feichtenhofer, C.; Pinz, A.; Zisserman, A. “Convolutional two-stream network fusion for video action recognition”, *arXiv preprint arXiv:1604.06573*, 2016.

[GBCB16] Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. “Deep learning”. MIT press, Massachusetts, USA:, 2016, vol. 1.

[GBS+07] Gorelick, L.; Blank, M.; Shechtman, E.; Irani, M.; Basri, R. “Actions as space-time shapes”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 29–12, 2007, pp. 2247–2253.

[GDDM14] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.

[GFF08] Gutstein, S.; Fuentes, O.; Freudenthal, E. “Knowledge transfer in deep convolutional neural nets”, *International Journal on Artificial Intelligence Tools*, vol. 17–03, 2008, pp. 555–567.

[GFH09] Győrbíró, N.; Fábián, Á.; Hományi, G. “An activity recognition system for mobile phones”, *Mobile Networks and Applications*, vol. 14–1, 2009, pp. 82–91.

[Gir15] Girshick, R. “Fast r-cnn”. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448.

[GJ14] Graves, A.; Jaitly, N. “Towards end-to-end speech recognition with recurrent neural networks”. In: Proceedings of the International Conference on Machine Learning (ICML), 2014, pp. II–1764–II–1772.

[GM92] Goodale, M. A.; Milner, A. D. “Separate visual pathways for perception and action”, *Trends in neurosciences*, vol. 15–1, 1992, pp. 20–25.

[GM15] Gkioxari, G.; Malik, J. “Finding action tubes”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 759–768.

[Gol75] Goldstein, E. B. “The perception of multiple images”, *AV Communication Review*, vol. 23–1, 1975, pp. 34–68.

[HAJ90] Huang, X. D.; Ariki, Y.; Jack, M. A. “Hidden Markov models for speech recognition”. Edinburgh university press, Edinburgh, SCT, 1990, vol. 2004.

[Hay09] Haykin, S. S. “Neural networks and learning machines”. Pearson Upper Saddle River, New Jersey, USA, 2009, vol. 3.

[HCL03] Hsu, C.-W.; Chang, C.-C.; Lin, C.-J. “A practical guide to support vector classification”, Relatório Técnico, Department of Computer Science, National Taiwan University, 2003.

[HHP16] Herath, S.; Harandi, M.; Porikli, F. “Going deeper into action recognition: A survey”, *arXiv preprint arXiv:1605.04988*, 2016.

[HRC15] Hasan, M.; Roy-Chowdhury, A. K. “A continuous learning framework for activity recognition using deep hybrid feature models”, *IEEE Transactions on Multimedia*, vol. 17–11, 2015, pp. 1909–1922.

[HS88] Harris, C.; Stephens, M. “A combined corner and edge detector.” In: Proceedings of the Alvey Vision Conference, 1988, pp. 50.

[HS97] Hochreiter, S.; Schmidhuber, J. “Long short-term memory”, *Neural computation*, vol. 9–8, 1997, pp. 1735–1780.

[HTWM04] Hu, W.; Tan, T.; Wang, L.; Maybank, S. “A survey on visual surveillance of object motion and behaviors”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34–3, 2004, pp. 334–352.

[Hu62] Hu, M.-K. “Visual pattern recognition by moment invariants”, *IRE transactions on information theory*, vol. 8–2, 1962, pp. 179–187.

[HVL08] Haering, N.; Venetianer, P. L.; Lipton, A. “The evolution of video surveillance: an overview”, *Machine Vision and Applications*, vol. 19–5-6, 2008, pp. 279–290.

[ITKR14] Iwashita, Y.; Takamine, A.; Kurazume, R.; Ryoo, M. S. “First-person animal activity recognition from egocentric videos”. In: Proceedings of the International Conference on Pattern Recognition (ICPR), 2014.

[JGZ+13] Jhuang, H.; Gall, J.; Zuffi, S.; Schmid, C.; Black, M. J. “Towards understanding action recognition”. In: Proceedings of the International Conference on Computer Vision (ICCV), 2013, pp. 3192–3199.

[Joh73] Johansson, G. “Visual perception of biological motion and a model for its analysis”, *Perception & psychophysics*, vol. 14–2, 1973, pp. 201–211.

[JXYY13] Ji, S.; Xu, W.; Yang, M.; Yu, K. “3D convolutional neural networks for human action recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35–1, 2013, pp. 221–231.

[JYC+11] Jiang, Y.-G.; Ye, G.; Chang, S.-F.; Ellis, D.; Loui, A. C. “Consumer video understanding: A benchmark database and an evaluation of human and machine performance”. In: Proceedings of ACM International Conference on Multimedia Retrieval (ICMR), 2011, pp. 29.

[KCY+10] Kim, I. S.; Choi, H. S.; Yi, K. M.; Choi, J. Y.; Kong, S. G. “Intelligent visual surveillance—a survey”, *International Journal of Control, Automation and Systems*, vol. 8–5, 2010, pp. 926–939.

[KJG+11] Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; Serre, T. “Hmdb: a large video database for human motion recognition”. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2011, pp. 2556–2563.

[KSH07] Ke, Y.; Sukthankar, R.; Hebert, M. “Spatio-temporal shape and flow correlation for action recognition”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8.

[KSH12] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. “Imagenet classification with deep convolutional neural networks”. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[KTS+14] Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. “Large-scale video classification with convolutional neural networks”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1725–1732.

[Lap05] Laptev, I. “On space-time interest points”, *International Journal of Computer Vision*, vol. 64–2-3, 2005, pp. 107–123.

[LBBH98] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. “Gradient-based learning applied to document recognition”, *IEEE*, vol. 86–11, 1998, pp. 2278–2324.

[LBH15] LeCun, Y.; Bengio, Y.; Hinton, G. “Deep learning”, *Nature*, vol. 521–7553, 2015, pp. 436–444.

[LCDH+90] Le Cun, B. B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. “Handwritten digit recognition with a back-propagation network”. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 1990.

[LHP+07] Logan, B.; Healey, J.; Philipose, M.; Tapia, E. M.; Intille, S. “A long-term evaluation of sensing modalities for activity recognition”. In: Proceedings of the International Conference on Ubiquitous Computing, 2007, pp. 483–500.

[LKF+10] LeCun, Y.; Kavukcuoglu, K.; Farabet, C.; et al.. “Convolutional networks and applications in vision.” In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2010, pp. 253–256.

[LL03] Laptev, I.; Lindeberg, T. “Space-time interest points”. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2003, pp. 432–439.

[LLS09] Liu, J.; Luo, J.; Shah, M. “Recognizing realistic actions from videos “in the wild””. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 1996–2003.

[LM05] Lahav, O.; Mioduser, D. “Blind persons’ acquisition of spatial cognitive mapping and orientation skills supported by virtual environment”, *International Journal on Disability and Human Development*, vol. 4–3, 2005, pp. 231–238.

[LSD15] Long, J.; Shelhamer, E.; Darrell, T. “Fully convolutional networks for semantic segmentation”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431–3440.

[LSPZ08] Lin, W.; Sun, M.-T.; Poovandran, R.; Zhang, Z. “Human activity recognition for video surveillance”. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCS), 2008, pp. 2737–2740.

[MAG+17] Monteiro, J.; Aires, J. P.; Granada, R.; Barros, R. C.; Meneguzzi, F. “Virtual guide dog: An application to support visually-impaired people through deep convolutional neural networks”. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2267–2274.

[MGBM17] Monteiro, J.; Granada, R.; Barros, R. C.; Meneguzzi, F. “Deep neural networks for kitchen activity recognition”. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2048–2055.

[MHGk14] Mnih, V.; Heess, N.; Graves, A.; kavukcuoglu, k. “Recurrent models of visual attentio”. In: Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 2014, pp. 2204–2212.

[MHK06] Moeslund, T. B.; Hilton, A.; Krüger, V. “A survey of advances in vision-based human motion capture and analysis”, *Computer vision and image understanding*, vol. 104–2, 2006, pp. 90–126.

[Mit97] Mitchell, T. M. “Machine Learning”. McGraw-Hill, New York, USA, 1997, 1 ed..

[MP43] McCulloch, W. S.; Pitts, W. “A logical calculus of the ideas immanent in nervous activity”, *The bulletin of mathematical biophysics*, vol. 5–4, 1943, pp. 115–133.

[MPK09] Messing, R.; Pal, C.; Kautz, H. “Activity recognition using the velocity histories of tracked keypoints”. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2009.

[NH10] Nair, V.; Hinton, G. E. “Rectified linear units improve restricted boltzmann machines”. In: Proceedings of the International Conference on Machine Learning (ICML), 2010, pp. 807–814.

[PHBB16] Park, E.; Han, X.; Berg, T. L.; Berg, A. C. “Combining multiple sources of knowledge in deep cnns for action recognition”. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 2016, pp. 1–8.

[Pop10] Poppe, R. “A survey on vision-based human action recognition”, *Image and Vision Computing*, vol. 28–6, Jun 2010, pp. 976–990.

[Pow11] Powers, D. M. W. “Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation”, *Journal of Machine Learning Technologies*, vol. 2–1, 2011, pp. 37–63.

[PVG+11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. “Scikit-learn: Machine learning in Python”, *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825–2830.

[PW12] Popoola, O. P.; Wang, K. “Video-based abnormal human behavior recognition—a review”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42–6, 2012, pp. 865–878.

[PY10] Pan, S. J.; Yang, Q. “A survey on transfer learning”, *IEEE Transactions on knowledge and data engineering*, vol. 22–10, 2010, pp. 1345–1359.

[RA09] Ryoo, M. S.; Aggarwal, J. K. “Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities”. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2009, pp. 1593–1600.

[RAAS12] Rohrbach, M.; Amin, S.; Andriluka, M.; Schiele, B. “A database for fine grained activity detection of cooking activities”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 1194–1201.

[RAHL18] Real, E.; Aggarwal, A.; Huang, Y.; Le, Q. V. “Regularized evolution for image classifier architecture search”, *arXiv preprint arXiv:1802.01548*, 2018.

[RAS08] Rodriguez, M. D.; Ahmed, J.; Shah, M. “Action mach a spatio-temporal maximum average correlation height filter for action recognition”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.

[Ras14] Rashidi, P. “Stream sequence mining for human activity discovery”, *Plan, Activity, and Intent Recognition*, 2014, pp. 123–148.

[RDML05] Ravi, N.; Dandekar, N.; Mysore, P.; Littman, M. L. “Activity recognition from accelerometer data”. In: Proceedings of the Conference on Artificial Intelligence (AAAI), 2005, pp. 1541–1546.

[RDS+15] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; Fei-Fei, L. “ImageNet Large Scale Visual Recognition Challenge”, *International Journal of Computer Vision (IJCV)*, vol. 115–3, 2015, pp. 211–252.

[RFO+09] Rose, T.; Fiscus, J.; Over, P.; Garofolo, J.; Michel, M. “The trecvid 2008 event detection evaluation”. In: Proceedings of the Workshop on Applications of Computer Vision (WACV), 2009, pp. 1–8.

[RHGS15] Ren, S.; He, K.; Girshick, R.; Sun, J. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 2015, pp. 91–99.

[RHM+86] Rumelhart, D. E.; Hinton, G. E.; McClelland, J. L.; et al.. “A general framework for parallel distributed processing”, *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 1, 1986, pp. 45–76.

[RM03] Ren, X.; Malik, J. “Learning a classification model for segmentation”. In: Proceedings of the IEEE on International Conference on Computer Vision (ICCV), 2003, pp. 10–17.

[Ros58] Rosenblatt, F. “The perceptron: A probabilistic model for information storage and organization in the brain.”, *Psychological review*, vol. 65–6, 1958, pp. 386.

[RRM15] Ryoo, M. S.; Rothrock, B.; Matthies, L. “Pooled motion features for first-person videos”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 896–904.

[SHK+14] Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. “Dropout: a simple way to prevent neural networks from overfitting.”, *Journal of machine learning research*, vol. 15–1, 2014, pp. 1929–1958.

[SJYS15] Sun, L.; Jia, K.; Yeung, D.-Y.; Shi, B. E. “Human action recognition using factorized spatio-temporal convolutional networks”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4597–4605.

[SKD+13] Shimada, A.; Kondo, K.; Deguchi, D.; Morin, G.; Stern, H. “Kitchen scene context based gesture recognition: A contest in icpr2012”. In: *Advances in Depth Image Analysis and Applications*, Springer, Berlin, GER, 2013, pp. 168–185.

[SLC04] Schuldt, C.; Laptev, I.; Caputo, B. “Recognizing human actions: a local svm approach”. In: Proceedings of the International Conference on Pattern Recognition (ICPR), 2004, pp. 32–36.

[SLD17] Shelhamer, E.; Long, J.; Darrell, T. “Fully convolutional networks for semantic segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39–4, April 2017, pp. 640–651.

[SLJ+15] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. “Going deeper with convolutions”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[SMH11] Sutskever, I.; Martens, J.; Hinton, G. “Generating text with recurrent neural networks”. In: Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML’11), 2011, pp. 1017–1024.

[SSK+13] Shotton, J.; Sharp, T.; Kipman, A.; Fitzgibbon, A.; Finocchio, M.; Blake, A.; Cook, M.; Moore, R. “Real-time human pose recognition in parts from single depth images”, *Communications of the ACM*, vol. 56–1, 2013, pp. 116–124.

[STWH16] Shi, Y.; Tian, Y.; Wang, Y.; Huang, T. “Sequential deep trajectory descriptor for action recognition with three-stream cnn”, *arXiv preprint arXiv:1609.03056*, 2016.

[SVI+16] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. “Rethinking the inception architecture for computer vision”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818–2826.

[SZ14a] Simonyan, K.; Zisserman, A. “Two-stream convolutional networks for action recognition in videos”. In: Proceedings of the Advances in Neural Information Processing Systems (ANIP), 2014, pp. 568–576.

[SZ14b] Simonyan, K.; Zisserman, A. “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*, 2014.

[SZHW15] Shi, Y.; Zeng, W.; Huang, T.; Wang, Y. “Learning deep trajectory descriptor for action recognition in videos using deep neural networks”. In: Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), 2015, pp. 1–6.

[SZS12] Soomro, K.; Zamir, A. R.; Shah, M. “Ucf101: A dataset of 101 human actions classes from videos in the wild”, *arXiv preprint arXiv:1212.0402*, 2012.

[TBF+15] Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. “Learning spatiotemporal features with 3d convolutional networks”. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4489–4497.

[TCSU08] Turaga, P.; Chellappa, R.; Subrahmanian, V. S.; Udrea, O. “Machine recognition of human activities: A survey”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18–11, Nov 2008, pp. 1473–1488.

[Tsu15] Tsukiyama, T. “In-home health monitoring system for solitary elderly”, *Procedia Computer Science*, vol. 63, 2015, pp. 229–235.

[WCDW17] Wang, C.-Y.; Chiang, C.-C.; Ding, J.-J.; Wang, J.-C. “Dynamic tracking attention model for action recognition”. In: Proceedings of Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 1617–1621.

[WFG16] Wang, X.; Farhadi, A.; Gupta, A. “Actions transformations”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[WH60] Widrow, B.; Hoff, M. E. “Adaptive switching circuits”, Relatório Técnico, Stanford University Eletronics Labs, 1960.

[WKSL11] Wang, H.; Kläser, A.; Schmid, C.; Liu, C.-L. “Action recognition by dense trajectories”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 3169–3176.

[WQT15] Wang, L.; Qiao, Y.; Tang, X. “Action recognition with trajectory-pooled deep-convolutional descriptors”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4305–4314.

[WRB06] Weinland, D.; Ronfard, R.; Boyer, E. “Free viewpoint action recognition using motion history volumes”, *Computer vision and image understanding*, vol. 104–2, 2006, pp. 249–257.

[WRB11] Weinland, D.; Ronfard, R.; Boyer, E. “A survey of vision-based methods for action representation, segmentation and recognition”, *Computer Vision and Image Understanding*, vol. 115–2, Fev 2011, pp. 224–241.

[WS06] Wang, L.; Suter, D. “Informative shape representations for human action recognition”. In: Proceedings of the International Conference on Pattern Recognition (ICPR), 2006, pp. 1266–1269.

[WS13] Wang, H.; Schmid, C. “Action recognition with improved trajectories”. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2013, pp. 3551–3558.

[WTS+13] Wagner, R.; Thom, M.; Schweiger, R.; Palm, G.; Rothermel, A. “Learning convolutional neural networks from few samples”. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–7.

[WXW+16] Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; Van Gool, L. “Temporal Segment Networks: Towards Good Practices for Deep Action Recognition”. Springer, Berlin, GER, 2016.

[YOI92] Yamato, J.; Ohya, J.; Ishii, K. “Recognizing human action in time-sequential images using hidden markov model”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1992, pp. 379–385.

[YWZ+15] Ye, H.; Wu, Z.; Zhao, R.-W.; Wang, X.; Jiang, Y.-G.; Xue, X. “Evaluating two-stream cnn for video classification”. In: Proceedings of the ACM on International Conference on Multimedia Retrieval (ICMR), 2015, pp. 435–442.

[YY15] Yu, G.; Yuan, J. “Fast action proposals for human action detection and search”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1302–1311.

[ZCVJ88] Zhou, Y.-T.; Chellappa, R.; Vaid, A.; Jenkins, B. K. “Image restoration using a neural network”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36–7, 1988, pp. 1141–1151.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br