DANIEL JIN HUN KIM
IGOR SEIJI NAKAMURA

RECONHECIMENTO DE ATIVIDADES HUMANAS POR APRENDIZADO DE MÁQUINA

Texto apresentado ao Departamento de Engenharia Mecatrônica e de Sistemas mecânicos da Escola Politécnica da Universidade de São Paulo (PMR).

São Paulo
2020

DANIEL JIN HUN KIM
IGOR SEIJI NAKAMURA

RECONHECIMENTO DE ATIVIDADES HUMANAS POR APRENDIZADO DE MÁQUINA

Texto apresentado ao Departamento de Engenharia Mecatrônica e de Sistemas mecânicos da Escola Politécnica da Universidade de São Paulo (PMR).

Orientador:
Jun Okamoto Junior

São Paulo
2020

Conteúdo

1 Introdução

O estudo de reconhecimento de ações utilizando dados visuais (imagens e vídeos) sempre esteve ligado a avanços em áreas complementares, como reconhecimento de objetos e dinâmicas de movimento do corpo humano. Na última década, a análise de ações humanas evoluiu de situações limitadas a ambientes controlados para soluções avançadas que podem aprender e analisar milhões de vídeos que representam quase todas as atividades cotidianas [1].

O problema escolhido envolve a detecção de atividades humanas através do uso de aprendizado de máquina, capaz de identificar padrões em vídeos (sequências de frames) [2]. Desta maneira, espera-se que um modelo que tenha sido treinado por uma base de dados com muitos trechos de vídeos distintos seja capaz de analisar novos vídeos e classificá-los em uma das categorias em que fora treinada, com um elevado índice de acertos.

Para iniciar o estudo do reconhecimento de ações e atividades, é necessário definir exatamente do que se tratam estes termos e suas diferenças. Wang Et al. [3] afirma que ação é toda interação com o ambiente que cause uma mudança no mesmo. De maneira a tornar esta definição mais abrangente, Chaquet et al.[4] afirma que ações podem ser consideradas uma sequência de movimentos que cumprem uma determinada função simples, como correr, pular, chutar uma bola. Já atividades são compostas por uma sequência de ações ao longo do espaço e tempo, como pessoas fazendo exercícios físicos ou praticando um esporte específico. Pode-se dizer que a realização de uma atividade está relacionada ao conceito de interação: entre pessoas, ou entre pessoas e objetos do ambiente.

De maneira a se aprofundar nesta área, pode-se dividir este campo de estudo em 3 categorias de problemas distintos [5]: reconhecimento de ações de uma pessoa; reconhecimento de interações entre duas pessoas ou mais pessoas e reconhecimento de interação humano-objeto, conforme o diagrama na figura 1. Cada uma destas situações envolvem contextos distintos e requerem o uso de diferentes abordagens e técnicas a serem utilizadas para reconhecer o acontecimento do vídeo analisado. Nas seções a seguir, os métodos utilizados para reconhecimento de ações e atividades humanas serão brevemente descritos, bem como as categorias de problemas, suas respectivas sub-seções, e a aplicabilidade dos métodos para

cada um destes problemas serão brevemente apontados.

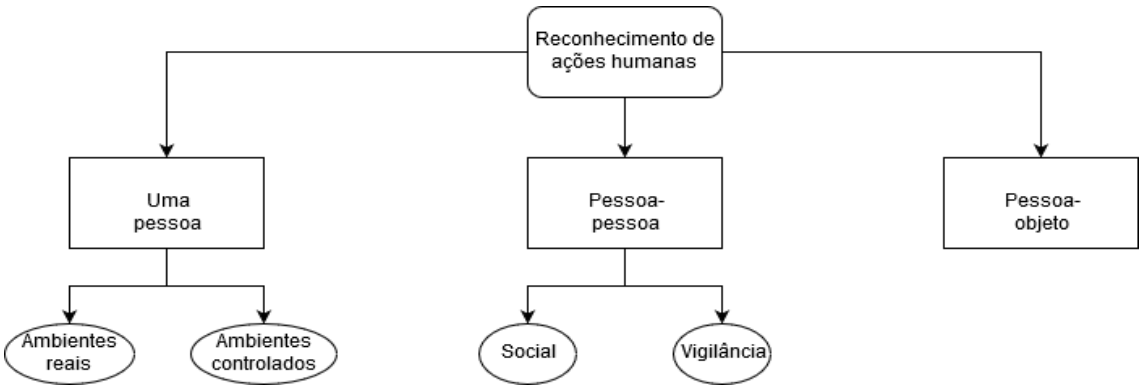


Figura 1: Tipos de problemas abordados.

Do ponto de vista mecatrônico e científico, o projeto proposto envolve conceitos de programação e engenharia elétrica, como visão computacional e aprendizado de máquina que atualmente estão atraindo cada vez mais atenção por parte do mundo acadêmico e do mercado de trabalho [6]. Além disso, a detecção de atividades humanas possui diversas aplicações práticas de grande relevância social. Na área de assistência a idosos, por exemplo, ela pode identificar e prevenir acidentes ao identificar situações perigosas em que estes possam se encontrar. Na área de assistência a bebês, por outro lado, ela pode monitorar o sono da criança, prever necessidades básicas, como comida e água e, recentemente, existem até pesquisas para identificação de autismo através de algoritmos de aprendizado de máquina [7]. Por fim, a detecção de atividades humanas também pode ser aplicada na área de segurança pública ao auxiliar o monitoramento de atividades suspeitas ou criminais [8].

2 Métodos de Reconhecimento de Ações Humanas

Há uma vasta gama de técnicas utilizadas para resolver o problema do reconhecimento de atividades humanas. De modo geral, pode-se classificar estes métodos entre:

- Soluções baseadas em extração de características
- Soluções baseadas em redes neurais profundas

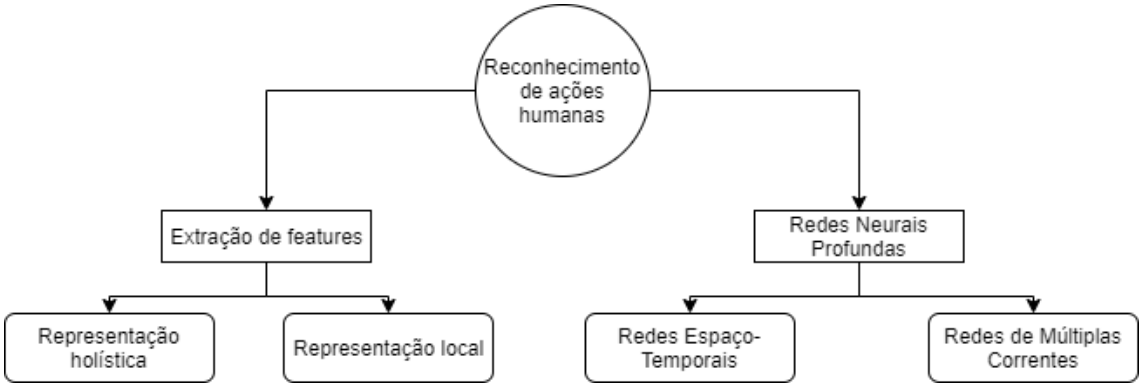


Figura 2: Taxonomia para métodos de reconhecimento de ação humana. Adaptado de [9].

2.1 Métodos baseados em extração de características locais

No domínio espacial, pontos com significativa variação local são frequentemente chamados de "pontos de interesse" por conterem uma grande quantidade de informação a respeito da imagem ou objeto em questão. Assim, ao expandir o conceito de pontos de interesse para o domínio espaço-temporal, temos como resultado, pontos de interesse espaço temporais (STIPs), que contém informações temporais (evolução do ponto de interesse ao longo de uma sequência de frames) adicionadas ao espaço, permitindo assim o reconhecimento de atividades humanas a partir de extração de características locais espaço-temporais.

Analogamente a técnicas de reconhecimento aplicadas para imagens, o reconhecimento de atividades humanas a partir de vídeos é feita na seguinte ordem: detecção de ponto de interesse $\rightarrow$ extração de descritor local $\rightarrow$ agregação de descritores locais [10]. Ou seja, uma vez que os STIPs são detectados, descritores locais extraem a forma e o movimento dos STIPs selecionados para que possa ser extraído um padrão a partir destes conjuntos e a classificação de dados novos possa ser feita.

2.2 Métodos baseados em redes neurais profundas

Recentemente, modelos de aprendizado profundo têm sido bastante empregados em tarefas de classificação de imagens [11]. Eles possuem a vantagem de realizarem simultaneamente o treinamento do modelo e a extração de características, ao contrário dos métodos tradicionais de reconhecimento de padrões [12]. Tal aprendizado de características é feito de forma hierárquica, de tal forma que

características nos níveis mais altos da hierarquia sejam formadas pela combinação de características de mais baixo nível.

Seguindo a taxonomia da figura 2, duas categorias de redes neurais profundas amplamente empregadas no reconhecimento de ações humanas são as redes espaço temporais, o que inclui a rede CNN 3D [13, 14, 2] e as redes de múltiplas correntes, cujas arquiteturas, basicamente, consistem de duas redes profundas paralelas: uma para processar informações espaciais relacionadas à aparência (cor, formato e identidade) e outra para processar informações temporais relacionadas ao movimento de objetos e pessoas [15].

2.3 Reconhecimento de ações de uma pessoa

O reconhecimento de ações de uma única pessoa a partir de vídeos pode ser considerado como a categoria mais simples de ser resolvida, uma vez que geralmente há um único objeto que necessita ser detectado, acompanhado e ter seu movimento analisado [16] para realizar o reconhecimento da ação envolvida, que é o corpo da pessoa em questão. Nesta categoria de problema, tem-se uma situação de reconhecimento de ações simples, uma vez que as bases de dados dedicadas a este tipo de situação consistem em sequências de vídeos classificadas em ações simples, como caminhando, correndo, parado em pé e caindo ([5]). Tais ações simples são classificadas como ações de baixo-nível [16], uma vez que estas são utilizadas como base para detecção de ações mais complexas, envolvendo interações humanas e com objetos.

Para realizar o treinamento de um algoritmo de aprendizado de máquina para que ele seja capaz de resolver um determinado problema (por exemplo, reconhecimento de atividades de uma pessoa), é necessário ter em mãos uma base com uma quantidade significativa de dados rotulados de acordo com sua categoria. Para ações simples realizadas por uma única pessoa, há uma grande disponibilidade de bases de dados com milhares de videoclipes rotulados de acordo com a ação que é praticada nele.

Inicialmente, o primeiro desafio na resolução do problema de reconhecimento de ações para uma única pessoa envolvida consistiam em bases de dados contendo videoclipes gravados em ambientes controlados, e portanto, não são representativos de uma situação no mundo real. Estas foram as primeiras bases de dados

constituídas por vídeos rotulados com ações: Weizmann (2001 e 2005) e KTH (2004).

As ações simples retratadas em Weizmann são filmadas de um ponto de vista único e fixado, com um fundo estático e simples. A base KTH adiciona complexidade, variando a roupagem dos indivíduos filmados e a iluminação, porém permanece pouco verossímil.

Eventualmente, o desafio evoluiu para a necessidade de identificar estas ações em situações mais complexas, que envolvam locais com iluminação não controlada (por exemplo o ambiente externo) e com fundos complexos e não estáticos. Para tal, foram consolidadas bases de dados de vídeo clipes filmados em ambientes mais verossímeis: CAVIAR (2004), ETISEO (2005), CASIA Action (2007), MSR Action (2009) e UT-Tower (2010). Adicionalmente, também foram elaboradas bases de dados com clipes extraídos diretamente da internet: HOLLYWOOD (2008), UCF Sports (2008), UCF YouTube (2009), UCF50 (2010), Olympic Sports (2010), HMDB51 (2011), cuja maioria tiveram seus clipes extraídos do YouTube [4].

2.3.1 Ambientes Controlados

Como foi explicado na seção anterior, os primeiros desafios na área de reconhecimento de ações humanas se davam em ambientes controlados (geralmente locais internos, com iluminação constante e fundo estático). Algumas das ações presentes nas bases de dados correspondentes a esta categoria são: correr, andar, acenar, pular, polichinelos, pular corda, bater palmas.

Ambas as abordagens ilustradas nas seções anteriores possuem bom desempenho nos datasets Weizmann e KTH. Para situações com uma única pessoa, métodos de extração de características baseados em representações locais possuem um desempenho excelente, com exemplos de trabalhos que atingiram a acurácia de 100% para Weizmann e 96.35% para KTH [17]. Isso se deve ao fato de que estes métodos são capazes de identificar pontos de interesse nos vídeo clipes com muita precisão, contanto que não haja movimentação significativa por parte da câmera ou um fundo complexo não-estático. Representações holísticas utilizando Energia de movimento (Motion Energy Images) e Histórico de movimento (Motion His-

tory Images) também atingiram 100% de acurácia na base de dados Weizmann, pelo mesmo motivo dos métodos de representações locais [18].

2.3.2 Ambientes Reais

Com o avanço e evolução das técnicas existentes para reconhecimento de atividades humanas, o estudo de identificação dessas atividades em ambientes mais próximos da realidade (e consequentemente mais complexos) tornou-se alvo de interesse de muitos pesquisadores da área de aprendizado de máquinas e visão computacional.

As ações que correspondem a este tipo de problema são semelhantes às da seção anterior, com o desafio adicional da ocorrência de mudanças no fundo e na iluminação, além da presença de outras pessoas no mesmo clipe. Como a maioria das filmagens ocorrem em locais públicos e/ou abertos, há a presença de algumas ações que envolvem o comportamento humano em sociedade, e que consequentemente são mais complexas: desmaio, movimentos agressivos, acenar, apontar, arremessar,

A base de dados CAVIAR, por exemplo, é constituída por clipes filmados em 2 locais distintos [4]: o primeiro conjunto de vídeos corresponde a filmagens feitas no lobby de entrada do laboratório INRIA, em Grenoble, França; o segundo conjunto se encontra no corredor de um shopping em Lisboa, Portugal.

Adicionalmente, alguns métodos também são avaliados em cima de bases de dados constituídas de vídeos retirados da internet, como mencionado na seção Bases de Dados. Estas bases adicionam mais complexidade ao problema, uma vez que são clipes registrados por cinegrafistas amadores e contém movimento de câmera significativo, fundo confuso, mudanças de ponto de vista e escala de objetos [4].

Pelos motivos citados acima, pode-se esperar que soluções mais complexas devem ser utilizadas para se obter boas acurácias para esse tipo de situação realística, em comparação com ambientes controlados. Métodos de extração de características baseados em representações locais são comumente utilizados para estes problemas, possuindo desempenho equivalente ou até superior ao de algumas arquiteturas profundas. Para as bases UCF Sports e UCF50, o estado da

arte é composto por estes métodos, atingindo acurácias de 88.2% e 94.4% respectivamente ([19], [20]). Uma das principais hipóteses para este desempenho superior de métodos mais antigos em comparação com redes neurais profundas para estas bases de dados se baseia na "insuficiência de dados disponíveis" [1].

Outra abordagem bastante utilizada para resolver problemas mais complexos de reconhecimento de ações humanas envolve a fusão entre métodos de extração de características locais com arquiteturas de redes neurais profundas. Utilizando-se esta abordagem, foram obtidos resultados de estado da arte para as bases HOLLYWOOD e Olympic Sports, com acurácias de 73.7% e 96.6% respectivamente ([21], [22]). Estes resultados oriundos de arquiteturas mistas demonstram empiricamente que as estruturas aprendidas pelas redes neurais são complementares às características extraídas por algoritmos de representação local [1].

2.4 Reconhecimento de interações entre múltiplas pessoas

A seção anterior 2.3 analisou pesquisas focadas, principalmente, em atividades de baixo nível, como pular, correr e acenar a mão. Essencialmente, essas atividades envolvem um único elemento, sem nenhuma interação entre duas ou mais pessoas. Porém, desde atividades como um simples aperto de mão até um jogo de futebol envolvem movimentos de diferentes pessoas. A grande dificuldade, no entanto, é entender que tais interações abrangem muito mais do que simplesmente analisar as ações de cada pessoa separadamente. Por exemplo, em um jogo de futebol é uma tarefa muito mais difícil localizar e acompanhar múltiplos elementos sincronizadamente para assim classificar a atividade como "jogando futebol", ao invés de simplesmente, "correndo" [23].

Há ainda outros desafios no reconhecimento de interações a partir de vídeos. O mais notável é a mudança no ângulo da câmera [23], que afeta como a interação é observada. Enquanto aplicações como vigilância utiliza câmeras estáticas, outras como esportes e vídeos gravados a partir de smartphones são gravações dinâmicas. Idealmente, o reconhecimento da interação deveria ser invariante à variação do ângulo da câmera. Porém, dependendo do ponto de vista, é possível que, durante uma interação, partes do corpo de uma pessoa estejam bloqueadas pelo corpo da outra pessoa. Assim, movimentos fundamentais podem não estar visíveis, prejudicando o reconhecimento da interação. Em segundo lugar, exis-

tem as chamadas variações intra-classe [24]. Ou seja, ações dentro da mesma classe podem ser expressas por diferentes pessoas através do movimento de diferentes partes do corpo. Por exemplo, a interação de cumprimentar uma outra pessoa pode ser feita através de um aperto de mão ou através do contato entre as bochechas de duas pessoas. O estudo de interações entre pessoas é ainda mais desafiador devido à falta de grandes e variadas bases de dados [25]. A maioria deles têm um escopo bastante limitado, como esportes ou vigilância e não há um padrão comum entre as classes de interações. Por exemplo, um aperto de mão pode ser uma classe própria ou parte da classe cumprimento. Além disso, uma mesma base de dados pode conter tanto atividades de uma única pessoa e, ao mesmo tempo, interações entre múltiplas pessoas, como é o caso de UCF101 [26], UCF50 [27], Hollywood2 [28], o que dificulta o teste e a validação de métodos de reconhecimento de interações.

Analogamente ao problema de reconhecimento de ações com uma única pessoa, existem dois métodos principais para a classificação de interações a partir de vídeos:

- Métodos baseados em extração de características locais

- Métodos baseados em modelos profundos

Tais abordagens são usadas para resolver uma ampla gama de problemas e duas grandes aplicações são o reconhecimento de interações sociais e o reconhecimento de interações para um sistema de vigilância automático.

2.4.1 Interações sociais

As interações sociais envolvem uma gama muito ampla de atividades, como, por exemplo, beijar, abraçar e apertar a mão de uma outra pessoa [5]. Analogamente aos métodos da seção 2.3, em geral, as abordagens baseadas em extração de características locais começam identificando características de baixo nível para depois reconhecer as atividades de alto nível [25]. Assim, é possível reconhecer tais interações da mesma forma que o reconhecimento de ações de uma única pessoa [29, 30]. Um aperto de mão, por exemplo, pode ser reconhecido pelas ações simultâneas de duas pessoas como "esticar o braço" e "recuar o braço" [29]. Porém, o desempenho de tais métodos não é tão bom pelo fato do reconhecimento de tais

interações envolverem mais do que simplesmente analisar as ações de cada pessoa separadamente. É necessário, também, analisar as relações espaço-temporais entre os diferentes indivíduos envolvidos. Informações como movimentação e posicionamento de cada pessoa e aos seus arredores são essenciais para um bom resultado e foram largamente empregadas [31, 32, 33, 34, 35]. As pessoas realizando atividades como beijos, abraços e aperto de mão, por exemplo, possuem alta proximidade física e a orientação das suas cabeças pode fornecer informações relevantes, visto que durante uma interação, supõe-se que duas pessoas estão frente a frente [31].

Como mencionado na seção 2.3, a seleção manual das características espaço-temporais relevantes para o reconhecimento de interações não é uma tarefa trivial. Portanto, foram desenvolvidas muitas soluções baseadas em modelos profundos. Para o caso de reconhecimento de interações, também foram implementadas redes de duas correntes [15, 36, 37], redes espaço-temporais [14, 38] e redes recorrentes profundas [39, 40]. Porém, os modelos profundos necessitam de uma grande quantidade de dados para serem treinados. Apesar das bases de dados focadas em interações humanas estarem crescendo, a quantidade de dados ainda é relativamente baixa, dada a complexidade do problema [25]. Uma solução parcial para isso seria a transferência de aprendizagem [41], visto que usar um modelo pré treinado reduz o tempo de treinamento e a quantidade de dados necessária [25].

Considerando que atividades podem ser descritas por diversas modalidades de características, grande atenção tem sido dada aos métodos multimodais [24], especialmente aqueles que focam no uso de emoções para o reconhecimento de atividades. Os estudos na área de computação afetiva, por exemplo, partem do pressuposto de que somente a informação visual não é suficiente para entender atividades humanas e, portanto, informação adicional é necessária [42]. Entender o contexto social e emocional em que duas pessoas se encontram pode ser crucial para uma interpretação correta da interação. Por exemplo: colocar a mão no ombro de alguém pode ser facilmente identificado, mas dependendo do contexto, pode significar consolo ou simplesmente desejo de chamar a atenção da outra pessoa [25]. O ato de empurrar uma outra pessoa pode ser um ato de violência ou simplesmente uma brincadeira amigável entre dois amigos. Assim, diversos autores modelaram a habilidade de uma pessoa de expressar, reconhecer e controlar

os seus estados afetivos em termos de gestos de mãos, expressões faciais, fala, atividades cardíacas e respiratórias e gestos dos ombros [43, 44, 45]. Porém, apesar do futuro aparentar ser promissor, ainda há grandes limitações relacionadas aos métodos multimodais. A falta de bases de dados adequadas para testar e validar a eficiência de sistemas multimodais, a dificuldade em relacionar características de baixo nível extraídas de vídeos com conceitos de alto nível, como emoções, e a dificuldade em selecionar características relevantes, que muitas vezes são grandemente dependentes da aplicação são alguns dos desafios que devem ser superados no futuro próximo [24].

2.4.2 Interações na área de vigilância

Essencialmente, há três tipos de análise de vídeos de sistemas de vigilância [46]. A análise manual é aquela feita por uma pessoa que deve observar as imagens das câmeras uma a uma até identificar uma atividade suspeita. Já a análise semi automática é aquela feita parte por uma pessoa, parte por um sistema automatizado. Por fim, a análise automática é aquela feita por um sistema autônomo, onde a análise, o processamento e a classificação do evento é independente de qualquer intervenção humana. Dessas, a análise automática é de grande interesse, pois o monitoramento manual contínuo e simultâneo de vídeos provenientes de diversas câmeras é cansativo e propenso a erros [47]. Além disso, atualmente, áreas públicas e privadas possuem uma crescente taxa de vigilância, através de câmeras instaladas nas mesmas. Ou seja, a alta quantidade de dados em formato de vídeos faz o aprendizado de máquina uma técnica ainda mais interessante para automatizar sistemas de vigilância [48].

Uma grande área de pesquisa dentro de sistemas de vigilância é a identificação automática de atividades suspeitas [47]. Uma atividade suspeita é definida como qualquer atividade fora do comum que expõe uma pessoa ou um grupo de pessoas ao perigo em um contexto particular [49]. Considerando tal definição, a vigilância automática de atividades suspeitas a partir de imagens de vídeo possui diversas aplicações. Dentro da área de interações, especificamente, algumas delas seriam a identificação de atos de violência ou crimes e o monitoramento automático de estudantes que estão fazendo provas [47]. No caso de atividades violentas ou criminosas como vandalismo, roubos e brigas, elas não podem ser interrompidas no momento de seu início. Porém, um sistema de vigilância inteligente pode reconhecer tal atividade e acionar um alarme para ajudar a entidade de segurança local

a tomar as medidas necessárias. Já um sistema de monitoramento de estudantes fazendo provas tem como objetivo identificar atividades suspeitas como movimentos incomuns da cabeça para proibir a cópia, um estudante trocar de lugar com o outro, o contato entre diferentes estudantes e a troca ilegal de material durante a prova. As grandes vantagens de tal sistema são detectar as atividades suspeitas, encorajar a redução delas e reduzir a necessidade de monitoramento manual, que muitas vezes pode ser cansativo e propenso a erros [50]. Novamente, para tais problemas, existem soluções baseadas em extração de características locais [51, 52, 53, 50] e baseadas em modelos profundos [54, 55, 56, 57].

2.5 Reconhecimento de interações humano-objeto

Desde atividades básicas como atender o celular até atividades em grupo como futebol envolvem diferentes objetos que são cruciais para o reconhecimento das mesmas. Assim, o reconhecimento de objetos e as suas interações com humanos em uma cena pode auxiliar grandemente na tarefa de reconhecimento de atividades. Ele envolve entender a cena e o evento, reconhecer os objetos que podem ser manipulados, analisar os movimentos humanos, e observar o efeito de tais movimentos nos objetos presentes na cena [58]. Em geral, o reconhecimento de interações humano-objeto é feito através da detecção do objeto e da pessoa, além da criação das suas respectivas caixas delimitadoras. Após isso, é feito o reconhecimento da ação humana que é a chave para identificar a interação humano-objeto em questão [59]. Os resultados de tais etapas são, em geral, agrupadas nas triplas <humano, ação, objeto> de forma bastante sucinta [58, 59, 60]. Um exemplo disso é a figura 3 em que a interação humano-objeto pode ser expressa pela tripla <humano, segurar, copo>. Ou a figura 4 que mostra um exemplo de interação <pessoa, atender celular, celular>.

Figura 3: Exemplo de interação humano-objeto <pessoa, segurar, copo>. Extraído de [60].

Figura 4: Exemplo de interação humano-objeto <pessoa, atender celular, celular>. Extraído de [61].

Porém, a interpretação de imagens e vídeos contendo pessoas interagindo com diferentes objetos pode se tornar uma tarefa bastante complexa. Um desafio na área, por exemplo, é identificar a interação relacionada a dois objetos que são aparentemente semelhantes, como mostra a figura 5, que contém um spray na esquerda e uma garrafa de água na direita. Um outro desafio é identificar interações cujos objetos possuem trajetória semelhante, como ilustra a figura 5. Por fim, um desafio ainda maior é distinguir entre interações que envolvem objetos semelhantes cujas trajetórias também são semelhantes, como por exemplo, atender celular e fazer uma ligação, conforme ilustra a figura 6.

As soluções para tais desafios são, analogamente aos mencionados nas seções 2.3 e 2.4, baseados em extração de características locais [58, 62, 63] ou em redes neurais profundas [64, 59, 65] que são capazes de modelar não somente as relações espaciais entre humano e objeto, mas também as suas relações temporais.

3 Requisitos

Para definir os requisitos que nortearão o desenvolvimento do sistema a ser entregue no final deste trabalho de formatura, é necessário especificar o escopo do problema a ser resolvido. Desta maneira, foi definido que o presente trabalho irá focar no problema mais simples de reconhecimento de ações de uma pessoa em ambientes controlados, detalhado na seção 2.3.1.

Portanto, a resolução de situações que envolvam interações entre múltiplas pessoas - como explicado na seção 2.4 - e interação humano-objeto - visto na seção 2.5 - não estarão dentro do estudo realizado neste trabalho.

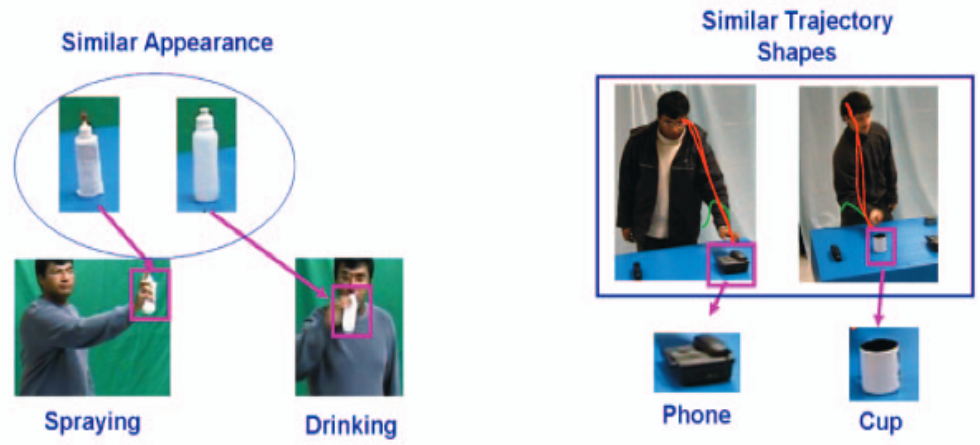


Figura 5: a) Objetos aparentemente semelhantes (spray na esquerda e garrafa de água na direita). b) Trajetórias de interações que são semelhantes (atender celular na esquerda e beber água na direita). Extraído de [58].

Figura 6: Objetos e trajetórias de interações que são semelhantes (atender celular em cima e fazer ligação embaixo). Extraído de [61].

Para atacar o primeiro problema, o algoritmo a ser utilizado precisaria ser capaz de detectar todos as pessoas presentes na cena, ser capaz de identificar quais de fato estão envolvidas em alguma ação relevante e entender a interação entre elas.

Quanto a detecção humano-objeto, a interpretação de sequências de vídeos contendo pessoas interagindo com diferentes objetos também é uma tarefa complexa. Ela envolve o reconhecimento dos objetos que podem ser manipulados, as

pessoas presentes na cena, identificação de quais participam ativamente da ação relevante, análise de seus movimentos e observação do efeito de tais movimentos nos objetos presentes na cena [58].

Portanto a resolução desta categoria de problemas demandaria conhecimentos que vão muito além da detecção de ações para uma pessoa. Consideramos que o entendimento do problema e aplicação de um método que atenda aos requisitos demandados por estes 2 problemas vão além do escopo deste trabalho de formatura.

Adicionalmente, delimitar a extensão da complexidade do sistema a ser desenvolvido para atuar apenas em ambientes controlados elimina 3 fatores que dificultam o reconhecimento de ações e o desempenho de algoritmos de aprendizado de máquina em imagens e vídeos como um todo ([5], [1]). São eles:

- Variações de pontos de vista, ou seja, situações em que a câmera não é estática e muda seu ângulo de filmagem;
- Mudanças de iluminação e oclusões de objetos de interesse;
- Presença de ruídos no fundo da cena: podem ser pessoas, animais ou objetos não-estáticos que não sejam objeto de interesse para a detecção da atividade.

3.1 Bases de dados

Se tratando de um problema a ser resolvido por meio de técnicas de aprendizado de máquina, o primeiro passo é definir qual base de dados será utilizada para treinar e testar o algoritmo. Esta deve conter uma quantidade suficiente de vídeos categorizados, contendo apenas ações de uma única pessoa, obtidas em um ambiente controlado. Na literatura, as bases de dados Weizmann [66] e KTH [67] foram extensamente utilizadas com o propósito de se estudar o mesmo problema proposto para este trabalho.

A base Weizmann é composta por 90 sequências de vídeos de baixa-resolução,com taxa de 50 frames por segundo e cada frame com dimensão de 180x144 pixels. Estas sequências demonstram 9 pessoas distintas, cada uma realizando 10 ações. São elas: andar, correr, trotar, polichinelos, pular para frente, pular e permanecer no mesmo ponto, pular de lado, acenar com ambas as mãos, acenar com uma mão

e se curvar. Os vídeos se encontram em formato AVI, e o tamanho total da base é de 340 MB. Este conjunto de dados é pioneiro no estudo de reconhecimento de ações, e considerado o mais simples para esta tarefa [4].

A base KTH é composta por 2391 sequências de vídeos, contendo 6 classes distintas de ações: andar, trotar, correr, boxear, acenar e bater palmas. Tais ações são realizadas diversas vezes por 25 pessoas em 4 cenários diferentes: ambientes externos (s1), ambientes externos com variação de escala (s2), ambientes externos com variação de vestimentas dos indivíduos filmados (s3) e ambientes internos (s4). Para todos estes ambientes, o fundo é homogêneo ao longo de uma sequência, como uma câmera estática que captura vídeos a uma taxa de 25 frames por segundo. Estas sequências já foram previamente reescalonadas para 160x120 pixels, e na média, duram 4 segundos. Estas sequências são gravadas em um arquivo AVI e são disponibilizadas na internet em formato comprimido, totalizando 1,2 GB. Este conjunto de dados, em relação à base Weizmann, adiciona um certo grau de complexidade, com o acréscimo de novas classes em conjunto com uma variação de ambientes e vestimentas.

Estas bases, portanto, são compostas por ações humanas simples em ambientes relativamente controlados. Desta maneira, entendemos que a implementação de um algoritmo de aprendizado de máquina (a ser definido e discutido na seção seguinte) para ser treinado e testado em cima do escopo de ações definidos por essas bases é um começo natural e adequado no estudo de atividades reconhecimento humanas. A relação de ações a serem classificadas pelo algoritmo pode ser vista na tabela 1.

Em seguida, após obtermos os resultados de treino e teste do algoritmo nas bases KTH e Weizmann, o objetivo será montar uma base de dados própria, contendo filmagens dos membros do grupo e conhecidos destes, performando as ações descritas na tabela 1, buscando analisar e entender a performance de um algoritmo de aprendizado de máquina, variando as condições dos dados de teste em relação as encontradas nos dados de treino.

Com o centro deste trabalho de formatura na resolução do problema de detecção de atividades humanas inicialmente a partir das bases KTH e Weizmann, e em seguida testando em uma base própria, o objetivo será o estudo do re-

Tabela 1: Tabela com relação de ações presentes nas bases de dados escolhidas

Ações	KTH	Weizmann
Curvar	✗	✓
Polichinelo	✗	✓
Pular para frente	✗	✓
Pular no lugar	✗	✓
Correr	✓	✓
Andar de lado	✗	✓
Pular com 1 perna	✗	✓
Andar	✓	✓
Acenar com 1 mão	✗	✓
Acenar com 2 mãos	✓	✓
Correr lentamente (trotar)	✓	✗
Lutar boxe	✓	✗
Bater palmas	✓	✗

Figura 7: Exemplos de frames da base de dados KTH. Extraído de [67].

conhecimento dos movimentos humanos, e como estes movimentos podem ser classificados em diferentes ações (para este trabalho, limitadas às ações da tabela 1). Desta maneira, iremos deixar de lado informações que envolvam o ambiente, objetos, outras pessoas e interações entre estas partes, uma vez que o escopo de classificação envolve apenas ações simples. Portanto, iremos analisar a capaci-

Figura 8: Exemplos de frames da base de dados Weizmann. Extraído de [66].

dade do algoritmo a ser implementado em aprender a classificar ações apenas por padrões de movimentação do corpo humano.

3.2 Rede Convolucional Neural 3D

Primeiramente proposta por [13], o método selecionado para resolver o problema proposto foi a rede neural convolucional 3D (CNN 3D). A CNN 3D é uma rede espaço temporal que tem como diferencial a habilidade de capturar as dependências espaciais intrínsecas às imagens (dois pixels adjacentes possuem alto grau de correlação) e as dependências temporais intrínsecas a sequências de frames (dois frames consecutivos também possuem alto grau de correlação). Ou seja, ela combina a convolução 2D com informações temporais, visto que redes convolucionais 2D conseguem obter informações espaciais de imagens com bastante eficácia [68].

A diferença entre as operações de convolução 2D e 3D estão ilustradas nas figuras 9 e 10. Uma convolução 2D aplicada em uma imagem ou em um volume de vídeo vai resultar em uma outra imagem 2D e assim, redes convolucionais tradicionais perdem a informação temporal a cada operação de convolução. Apenas a convolução 3D preserva a informação temporal da sequência de frames, pois a convolução 3D aplicada em um volume de vídeo tem como saída um outro volume de vídeo. Mais especificamente, a convolução 3D é feita através de um filtro 3D (ou kernel) que percorre as 3 dimensões espaciais e temporal de um cubo que é

formado através do empilhamento de diversos frames sequenciais do vídeo [13]. Tal empilhamento de frames sequenciais para a formação do cubo estão melhor ilustrados nas figuras 10, 11.

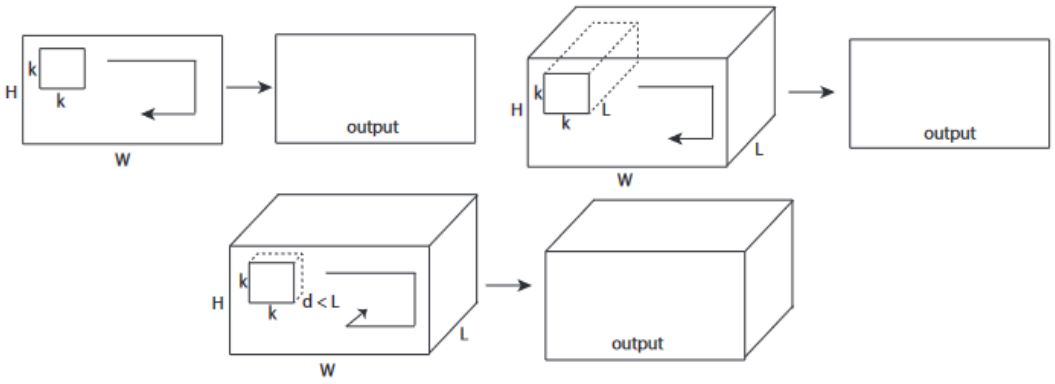


Figura 9: a) Convolução 2D aplicada em uma imagem resulta em outra imagem. b) Convolução aplicada em um volume de vídeo resulta em um outro volume de vídeo; c) Convolução em um volume de vídeo tem como saída um outro volume de vídeo. Extraído de [14].

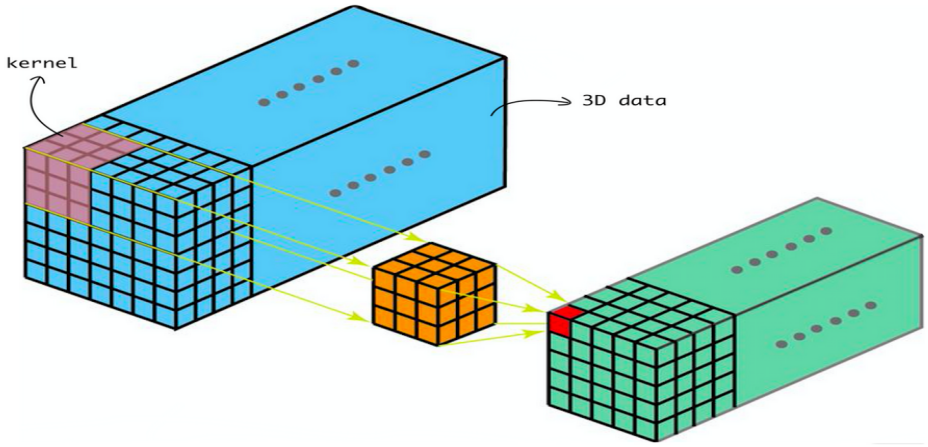


Figura 10: Ilustração de uma convolução 3D. Extraído de [69].

O racional por trás da escolha desta arquitetura se baseia em nosso objetivo de detectar atividades humanas simples. Desta maneira, buscamos arquiteturas que obtiveram resultados estado-da-arte em bases de dados contendo este tipo de ação (as bases KTH e Weizmann que serão usadas neste trabalho são as principais expoentes). Neste caso, a arquitetura CNN3D apresentada por [13] obteve resultados expressivos na base KTH, sendo que foi precursor para a elaboração de redes neurais mais complexas, capazes de analisar ações envolvendo interações

Figura 11: Ilustração do empilhamento de uma sequência de frames. Extraído de [70].

entre pessoas e objetos. Estas arquiteturas, porém, necessitam de uma quantidade muito maior de dados para serem treinadas, que não é o caso das bases contendo ações simples - a base Weizmann possui apenas 90 vídeos, enquanto a base KTH é composta por 600 vídeos de apenas 15 segundos em média cada. Concluindo, decidimos que inicialmente, a arquitetura a ser implementada será conforme à descrita por [13] e, uma vez que esta seja devidamente testada, iremos montar uma base de dados com vídeos próprios contendo as mesmas ações das bases usadas para treinar e validar a arquitetura, visando testar a performance da rede neural com variações nos ângulos de filmagem e nos movimentos realizados.

3.2.1 Pré-processamento

A etapa de pré-processamento dos dados é realizada antes da entrada da rede, com o intuito de acelerar o treino e teste da arquitetura, reduzindo a quantidade de informações presentes nos frames que não sejam úteis para o reconhecimento e classificação das atividades nele performadas. Esta fase consiste em 2 passos: primeiramente, é utilizado o algoritmo de Modelos Mistos Gaussianos (Gaussian Mixed Models ou GMM) para subtração do fundo da imagem. Este algoritmo está presente na biblioteca open-source OpenCV, e seu funcionamento consiste em modelar cada pixel de um vídeo por meio de conjunto de distribuições Gaussianas, com o intuito de identificar se um dado pixel com uma intensidade RGB determinada pode ser classificado como pertencente ao fundo ou ao plano frontal. Desta maneira, é possível isolar o objeto de interesse (em nosso caso, o corpo da pessoa em movimento) do resto da cena (fundo estático). Para cada frame do vídeo, este algoritmo retorna uma máscara correspondente ao plano frontal, e a

partir desta, é extraída uma caixa delimitadora que engloba o objeto de interesse mantendo apenas a porção do frame necessária para reconhecer a atividade realizada. Em seguida, buscando reduzir os requisitos de memória do computador, a resolução dos frames é reduzida. No caso do paper selecionado, para a base KTH, sua resolução foi de 160x120 para 80x60.

Após a etapa de pré-processamento dos vídeos, os frames são empilhados em grupos de 9, centrados em torno de cada frame presente no vídeo. A idéia é que cada pilha de 9 frames seja usada como entrada na arquitetura, onde a rede irá gerar uma classificação para cada um desses cubos, e a classe final é a mais frequente (método conhecido por majority voting - ou votação por maioria). Desta maneira, como pode ser visto na figura 13, a entrada possui dimensão 80x60x9. Em seguida, são aplicados 5 tratamentos distintos para os frames, gerando 5 canais de informação distintos que serão usados para alimentar a rede neural, sendo estes: gradientes e fluxo óptico nas direções X e Y, e escala de cinza. Os canais dos gradientes X, Y e da escala de cinza contêm 9 frames cada um, enquanto que os fluxos ópticos em X e Y contêm 8 frames na saída para cada um. Desta maneira, temos um total de 43 frames que irão alimentar a arquitetura para cada pilha de frames. Uma visão geral e resumida dessa etapa de tratamento de dados pode ser vista no diagrama da figura 12.

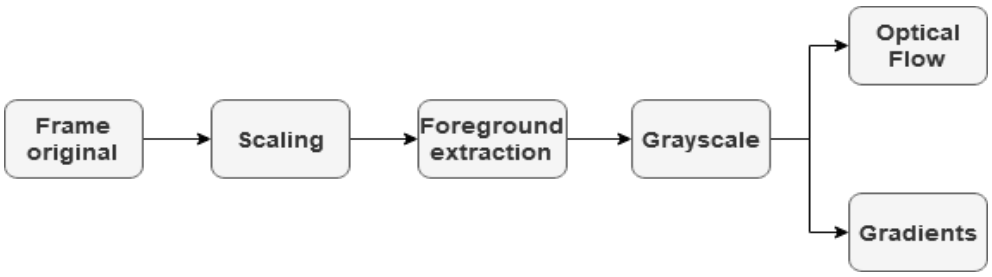


Figura 12: Diagrama resumindo a etapa de tratamento de dados.

3.2.2 Descrição da arquitetura

Como dito na seção anterior, a arquitetura a ser implementada neste trabalho foi descrita por [13] e consiste em uma rede neural convolucional, composta por um total de 8 camadas (incluindo a entrada), e seu resumo pode ser visto na tabela 2.

Tratando-se da arquitetura, são realizadas 3 convoluções. A primeira (C2) é feita por 2 filtros distintos com dimensão espacial 9x7 e dimensão temporal 3. A segunda (C4) é feita por 3 filtros de dimensão espacial 7x7 e dimensão temporal 3. A última convolução (C6) é feita por filtros 2D de dimensão espacial 6x4, para que o tamanho dos mapas de características resultante seja 1x1. Cada um dos 23*6 = 138 mapas de características na camada S5 estão conectados aos 128 mapas da camada C6 por meio destes filtros 2D. Além disso, após cada convolução 3D, há uma camada de subamostragem composta por filtros de dimensão 3X3, que realizam um processo chamado max-pooling. Esta etapa é importante para obter maior invariância e robustez contra ruídos e variações nas imagens, além de gerar mapas de características com dimensões reduzidas [71]. No caso da arquitetura proposta, usando um filtro de dimensão 3x3 irá dividir a dimensão horizontal e vertical dos mapas por 3.

Ao final da arquitetura, os 9 frames de dimensão 80x60 foram reduzidos a um vetor de características com dimensão 128x1, condensando as informações de movimentação contidas nos 9 frames. Para realizar a classificação, utiliza-se uma rede neural totalmente conectada, de maneira que os 128 nós na entrada são conectados às 6 unidades de saída (correspondendo às 6 classes de ações presentes na base KTH), operando como um classificador.

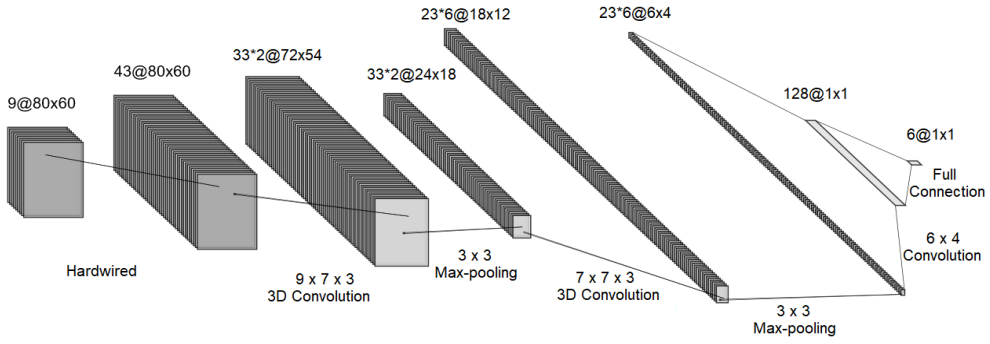


Figura 13: Ilustração da arquitetura da rede 3D CNN sugerida por [13].

3.3 Configuração do setup

Para realizar o estudo definido nesta seção, é necessário ter um ambiente de desenvolvimento, onde o programa contendo o algoritmo de implementação da

Tabela 2: Arquitetura da CNN 3D proposta por [13]

Camada	Forma da Ativação	Tamanho da Ativação	Número de Parâmetros
Entrada	9 @ 80x60	43.200	0
H1	45 @ 80x60	206.400	0
C2	2x33 @ 72x54	256.608	(9*7*3+1)*2 = 380
S3	2x33 @ 24x18	30.888	0
C4	6x23 @ 18x12	33.120	(7*7*3+1)*3 = 444
S5	6x23 @ 6x4	3.312	0
C6	128 @ 1x1	128	(128*138+1) = 17.665
Saída	6 @ 1x1	10	(128*10+1) = 1281

rede neural convolucional 3D possa ser compilado e executado. Assim, os modelos de aprendizado profundo serão desenvolvidos utilizando o TensorFlow que é uma biblioteca open-source e gratuita desenvolvida pela Google em Python [72]. Adicionalmente, é necessário possuir um espaço para armazenar as bases de dados KTH e Weizmann e que possa ser acessado pelo código de implementação, que irá utilizar estas bases para realizar o treinamento e o teste da rede neural. Para tal, será utilizado o Google Colab, que se trata de um ambiente interativo que permite escrever códigos em Python e roda inteiramente na nuvem. Desta maneira, é possível armazenar as bases de dados neste mesmo ambiente, que possui 320 GB disponíveis para serem utilizados para cada sessão.

Este ambiente de desenvolvimento também permite o uso de CPUs e GPUs hospedadas na nuvem por até 12 horas. A GPU disponibiliza até 12 GB de memória RAM, e a CPU possui um processador de 2.3GHz. Resultados estado da arte para as bases KTH e Weizmann, usando arquiteturas de redes neurais convolucionais 3D usaram um setup consistindo em um PC com processador Intel Core I7 e 8GB de memória RAM [73]. Desta maneira, é possível garantir que as configurações do Google Colab serão suficientes para a realização deste trabalho.

4 Resultados

Para gerar os resultados desta seção, foram utilizados todos os requisitos listados na seção 3. Assim, primeiramente será feito o processamento das imagens de vídeo que é o mesmo para todos experimentos. Após isso, o modelo convolucional implementado será treinado sobre as diferentes bases de dados (Weizmann, KTH

e base de dados própria) e para melhorar o desempenho do mesmo, diferentes hiperparâmetros e técnicas de regularização serão explorados. Por fim, será feita uma comparação entre o modelo do presente trabalho e do artigo [13], que foi um dos precursores da rede convolucional 3D.

4.1 Pré-processamento

A etapa de pré processamento foi implementada utilizando a biblioteca OpenCV e a sua documentação em alto nível está inserida no apêndice dentro do módulo image_processing.py. Ela segue a estrutura descrita na figura 12 e um exemplo da etapa de pré processamento aplicada na ação “bend"do dataset Weizmann contendo os cinco canais de informação é apresentado nas figuras 14, 15, 16, 17, 18 e 19.

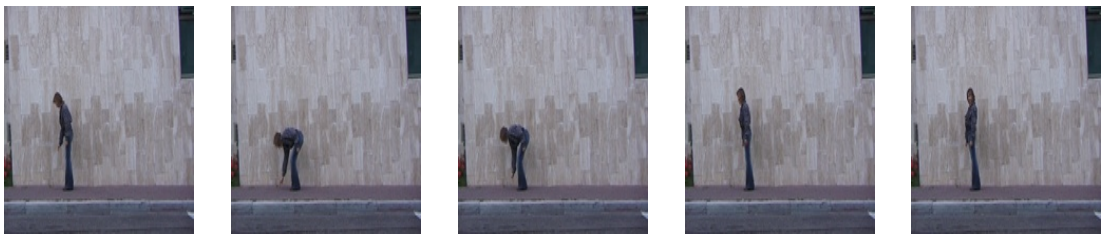


Figura 14: Frames originais da ação "bend"do dataset Weizmann. Extraído e adaptado de [66]

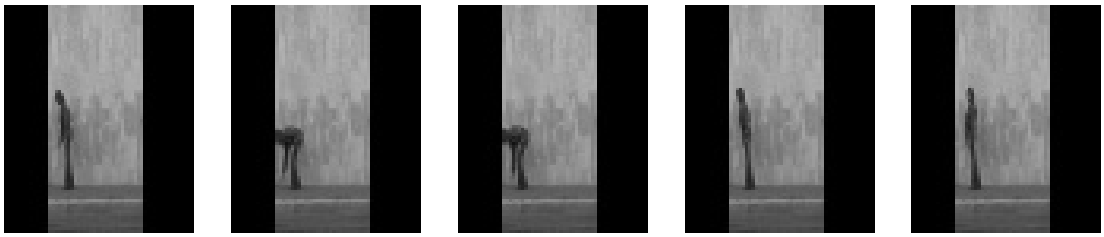


Figura 15: Frames em tons de cinza após scaling e foreground extraction

4.2 Base de dados Weizmann

Inicialmente, foi realizado o treino da arquitetura utilizando os vídeos da base Weizmann. Para realizar o treinamento, utilizamos a função de otimização de gradiente descente estocástico (Stochastic Gradient Descent - SGD), a função de perda definida foi a entropia cruzada categórica e a métrica usada para definir a

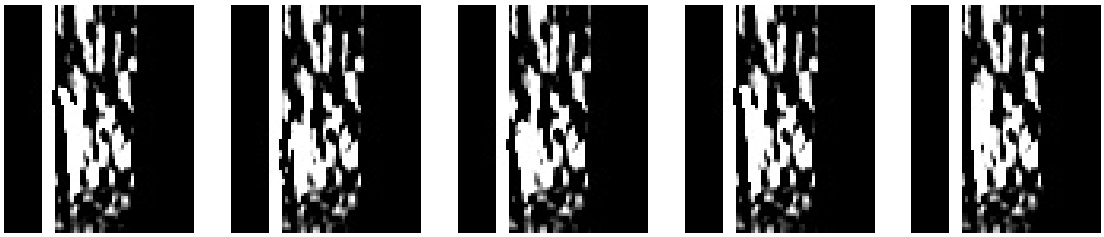


Figura 16: Gradiente na direção X a partir dos frames em tons de cinza

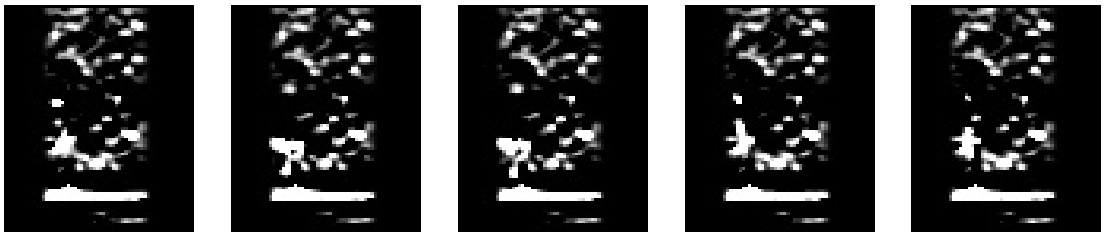


Figura 17: Gradiente na direção Y a partir dos frames em tons de cinza

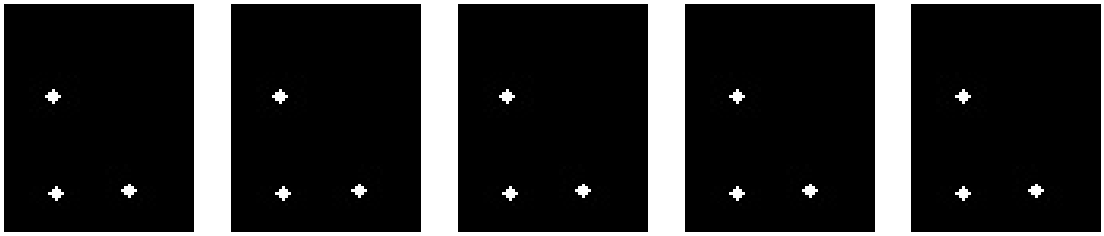


Figura 18: Fluxo óptico na direção X a partir dos frames em tons de cinza

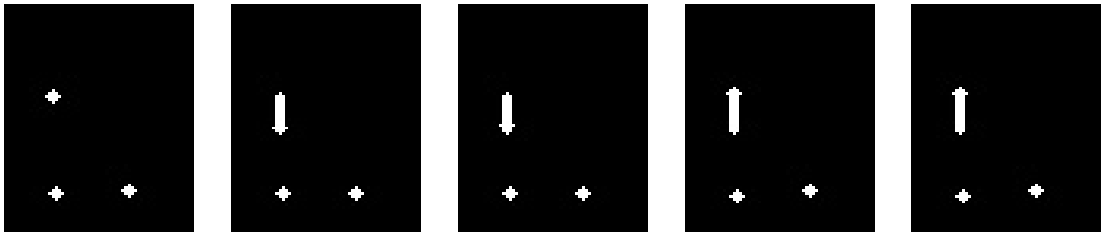


Figura 19: Fluxo óptico na direção Y a partir dos frames em tons de cinza

precisão de classificação da rede foi a acurácia (proporção entre acertos e número de exemplos total).

Uma vez que a base possui 90 vídeos (sendo 9 vídeos para cada uma das 10 classes), decidimos dividir a base entre 80 vídeos para treino e 10 vídeos para teste (1 vídeo para cada classe). Como pode ser visto na figura 20, obtivemos uma perda de 0.23 e uma acurácia de treino de 92.6% usando essa configuração, após 3 épocas (a base de dados foi alimentada para a rede neural 3 vezes) e com um lote de 32 pilhas de frames (o treino em cada época é feito em partes, dividido em grupos de 32 pilhas de frames cada). Essas pilhas de frames são os grupos de 9 frames conforme descrito na seção 3.2.1. Além disso, pode-se considerar que as acurácias de treino mencionadas neste presente trabalho são referentes aos empilhamentos de 9 frames classificados corretamente. Ou seja, para a base de dados Weizmann, 92,6% das pilhas de frames foram classificadas corretamente.

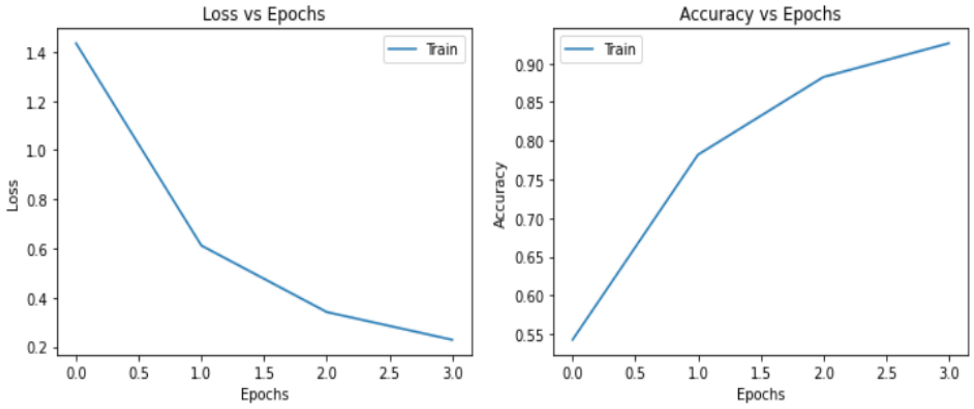


Figura 20: Gráficos das curvas de perda e de acurácia em função do número de épocas na base Weizmann

Após obter este resultado no treino, fizemos o teste com os 10 vídeos, obtendo uma acurácia de apenas 60%, ou seja, 6 classes de 10 foram preditas corretamente. Concluímos que a base possui um tamanho pequeno para treinar uma arquitetura de rede neural, o que leva a um sobreajuste nos parâmetros da arquitetura (elevada acurácia de treino em comparação com uma acurácia de teste reduzida). Adicionalmente, apenas 1 vídeo para cada classe é uma quantidade muito pequena para se obter uma métrica de acurácia que realmente indique a capacidade preditiva do algoritmo. Desta maneira, existem algumas alternativas que podemos utilizar para mitigar a questão da escassez de dados desta base.

Entre elas, iremos adotar o uso de técnicas de aumento de dados (data augmentation) para gerar novos vídeos a partir dos já existentes; o treinamento usando validação cruzada também será feito, uma vez que ele permite obter a acurácia para diferentes combinações de dados de treino e teste, nos fornecendo uma percepção melhor da capacidade de generalização verdadeira da rede.

4.2.1 Data Augmentation para o treino na base Weizmann

Conforme visto na seção 4.2 foi aplicada a técnica de data augmentation para resolver o problema de overfitting [74]. Inicialmente, foram aplicadas as seguintes transformações geométricas: inversão horizontal dos frames, inversão vertical dos frames e rotação em 45 graus dos frames. As figuras 21, 22 e 23 mostram exemplos dessas três transformações na ação "bend" do conjunto de dados Weizmann. Assim, considerando que todos os vídeos sofrerão três transformações, a quantidade de vídeos e de frames para treino irá aumentar em quatro vezes e, portanto, a técnica tem um grande potencial para solucionar o problema de overfitting. Essa relação entre o número de frames antes e depois da aplicação da técnica de data augmentation está expressa na tabela 3.

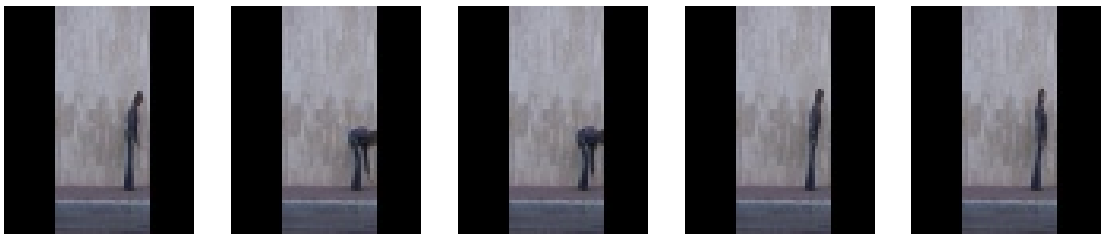


Figura 21: Frames invertidos horizontalmente da ação "bend" da base de dados "Weizmann". Extraído e adaptado de [66]

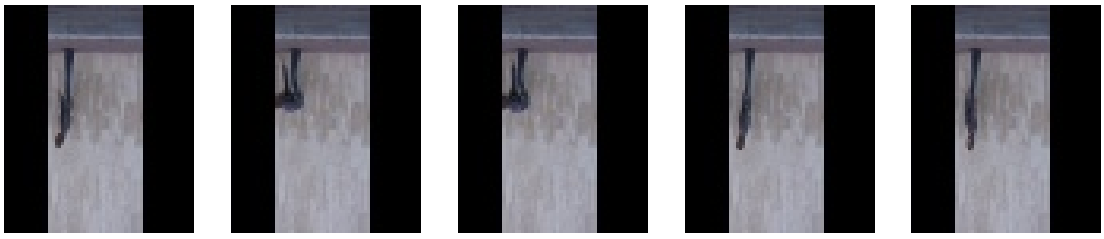


Figura 22: Frames invertidos verticalmente da ação "bend" da base de dados "Weizmann". Extraído e adaptado de [66]

Figura 23: Frames rotacionados em 45 graus da ação “bend” da base de dados “Weizmann”. Extraído e adaptado de [66]

Tabela 3: Tabela com número de frames no dataset Weizmann antes e depois da técnica de data augmentation

Ações	Quantidade de frames	
	Original	Data Augmentation
Andar	399	1596
Curvar	378	1512
Correr	164	656
Andar de lado	216	864
Acenar com 1 mão	417	1668
Acenar com 2 mãos	397	1588
Polichinelo	498	1992
Pular no lugar	308	1232
Pular com 1 perna	213	852
Pular para frente	215	860

Na etapa de validação do modelo, os mesmos conjuntos de treino e de teste da seção 4.2 foram utilizados, ou seja, 80 vídeos para treino e 10 vídeos para teste. Entretanto, mesmo com o aumento do número de frames, os resultados obtidos não foram satisfatórios. Após um treino de 15 épocas, a acurácia de treino foi de 72.8%, enquanto que a acurácia de teste foi de apenas 30%. Ou seja, o modelo conseguiu classificar corretamente apenas 3 vídeos dos 10 de teste. As curvas de perda e acurácia de teste por época podem ser vistas na figura 24.

Uma possível razão para o desempenho baixo da rede é a discrepância entre os números de frames por classe, como expressa na tabela 3. A classe "Polichinelo", por exemplo, possui 1992 frames, três vezes mais do que a classe "Correr"que possui apenas 656 frames. Portanto, para mitigar tanto a questão da escasse de dados desta base e o desbalanço entre o número de frames por classe, serão adicionados vídeos de uma base de dados própria, conforme descrito na seção 4.4.

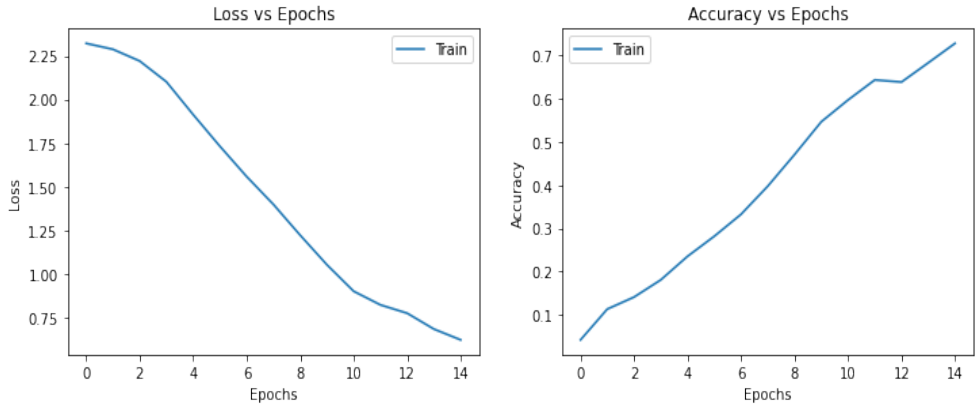


Figura 24: Gráficos das curvas de perda e de acurácia em função do número de épocas na base Weizmann com data augmentation.

4.3 Base de dados KTH

A base KTH é composta por 6 ações distintas, sendo que há um total de 25 pessoas diferentes realizando cada uma dessas atividades em 4 cenários diferentes. Portanto, ao todo há 600 vídeos nesta base. Seguindo a divisão destes dados em treino e teste feita por [13], são usados todos os vídeos de 16 pessoas para treinar a arquitetura (384 vídeos), e os vídeos das 9 pessoas restantes para testar (216 vídeos).

Um grande problema enfrentado durante o treinamento usando essa rede se deu por conta do preenchimento da memória RAM. O Google Colab possui até 12 gb disponíveis, sendo que o espaço ocupado por todos os frames, adicionados aos frames pré-processados que compõem os 5 canais de informação acabam necessitando de um armazenamento maior. Para contornar este problema, utilizamos uma função geradora em Python, no qual as pilhas de frames são alimentadas como entrada na rede neural em grupos separados, de modo que a memória RAM não seja sobrecarregada com todos os dados de treino de uma única vez.

Para garantir que a rede neural não tenha problemas de sobreajuste, implementamos o treinamento de modo que as pilhas de frames oriundas de vídeos de classes distintas são alimentadas para a arquitetura de maneira aleatória.

Realizando o treino após 40 épocas e usando os mesmos parâmetros utilizados para o treino da base Weizmann(otimizador SGD, função de perda é a entropia

cruzada categórica e acurácia como métrica de classificação), obtivemos uma acurácia de 99% de treino, com uma perda de 0,015. Esses dados podem ser vistos na imagem 25.

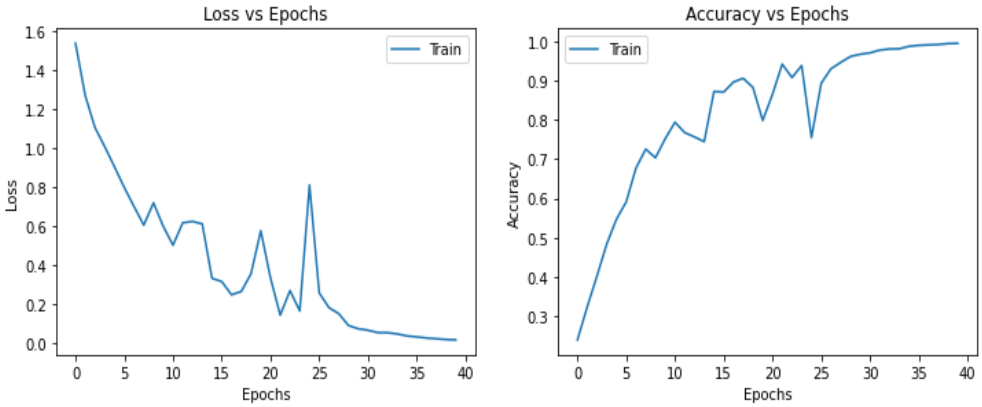


Figura 25: Gráficos das curvas de perda e de acurácia em função do número de épocas na base KTH.

A acurácia de teste obtida com essa rede treinada foi de apenas 42,59%, indicando que ainda há uma melhora considerável para ser feita neste treino. Em particular, notamos que há uma grande quantidade de frames nos vídeos dessa base que não possuem qualquer informação de movimento realizada pelos indivíduos filmados, ou seja, a câmera capturou apenas o fundo estático. Isto ocorre nas classes andar, correr e correr lentamente (trotar), onde por vários segundos, o indivíduo acaba saindo do campo de filmagem da câmera, principalmente para as 2 últimas classes. Acreditamos que estes frames possam atrapalhar o treinamento da rede, uma vez que não estão de fato transmitindo qualquer informação a respeito do movimento que se deseja classificar. Para demonstrar isso empiricamente, analisamos as predições feitas pela rede treinada, e notamos que a acurácia para essas 3 classes era menor - em torno de 30%, enquanto que as classes de bater palmas e acenar possuíam melhor desempenho - em torno de 60%.

4.3.1 Alterações na base KTH

Tendo em vista os problemas que enfrentamos com a base KTH descritos na seção 4.3, realizamos o tratamento de todos os vídeos das classes correr e andar, cortando cada vídeo e deixando apenas os frames que continham o corpo da pessoa completamente enquadrado e realizando o movimento descrito pela classe.

A edição foi feita utilizando o aplicativo Fotos do Windows. Desta maneira, a quantidade de vídeos das classes *correr* e *andar* aumentaram em 4 vezes, na medida que cada vídeo original foi cortado e dividido em outros 4 vídeos.

Além disso, após uma análise feita a respeito das 6 classes presentes na base, concluímos que as classes boxear e correr lentamente (trotar) seriam retiradas de nossos testes. Para a classe boxear, nossa decisão partiu da conclusão de que esta ação não possui relevância em ser detectada para fins práticos (vigilância, monitoramento de multidões). Quanto a classe trotar, concordamos que não há valor prático em diferenciá-la da classe correr, e decidimos manter apenas a 2ª.

Desta forma, realizamos os testes com esta base usando 4 classes: correr, andar, acenar e bater palmas. Inicialmente, buscamos equilibrar o número de frames por classe, de maneira a manter a base de treino balanceada. A tabela 4 mostra a quantidade média de frames por cada vídeo de cada classe. Portanto, utilizamos os 400 vídeos da classe *correr* (4 vezes mais vídeos devido às edições feitas nesta classe), que possui a menor quantidade de frames disponíveis (12970 frames), e utlizamos os vídeos das outras classes de maneira que estes tivessem a quantidade de frames mais próxima possível deste patamar. Ao final, a classe *Andar* possuía 44 vídeos, *Acenar* possuía 30 vídeos, e *bater palmas* possuía 24 vídeos.

Tabela 4: Quantidade média de frames por vídeo para cada classe

Classe	**Frames por vídeo**
Andar	292 frames
Correr	130 frames
Acenar	537 frames
Bater Palmas	431 frames

Usando esta configuração para nossas bases de treino, obtivemos uma acurácia de 95% de treino e uma perda de 0.14. O treino foi realizado utilizando o SGD como algoritmo de otimização, por um total de 50 épocas. O resultado dele pode ser visto na figura 26.

Destrinchando a acurácia na base de teste, notamos que esta varia em grande escala para cada classe: *correr* possui uma acurácia muito elevada de 94.53%,

andar obteve uma boa acurácia de 76.56%, enquanto que *acenar* possui um índice de acertos de 45.71%, e *bater palmas* possui apenas 11.11% de seus vídeos classificados corretamente.

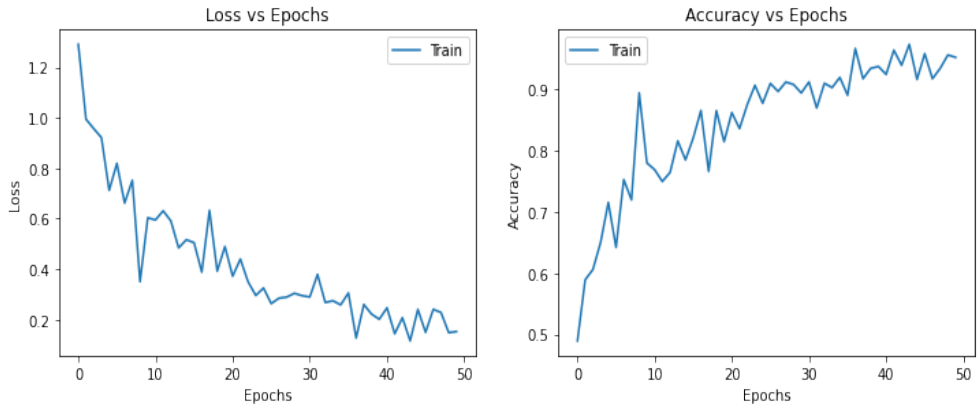


Figura 26: Gráficos das curvas de perda e de acurácia em função do número de épocas na base KTH com frames balanceados e editados.

Tabela 5: Acurácia na base de teste

Classe	**Acurácia de teste**
Andar	76.56%
Correr	94.53%
Acenar	45.71%
Bater Palmas	11.11%
Base completa	84.82%

Analisando-se a discrepância nos resultados obtidos para cada classe com esta configuração, concluímos que o balanceamento deveria ser feito por quantidade de vídeos, e não de frames. Isso se deve a característica dos vídeos das classes de acenar e bater palmas: apesar de possuírem uma quantidade de frames consideravelmente maior que as classes de correr e andar, a informação contida nesses vídeos é repetida - ou seja, os movimentos realizados ao longo do vídeo são muito repetitivos, ao contrário dos vídeos de *correr* e *andar*, que possuem padrões de movimentos variados ao longo de uma quantidade reduzida de frames. Logo, consideramos que o correto de fato seria balancear nossas bases de treino e teste por número de vídeos originais. Desta maneira, usamos os 100 vídeos disponíveis para as classes de acenar e palmas, e os 400 vídeos para as classes de correr e andar (que contém a mesma quantidade de informação dos 100 vídeos originais).

Após alguns testes, observamos que o melhor resultado foi obtido com o treino por 50 épocas. Acima disso, a rede ficava sobreajustada e pecava na performance na base de testes. Desta forma, obtivemos uma acurácia de 95.6% e uma perda de 0.19. O gráfico do treino pode ser observado na figura 27.

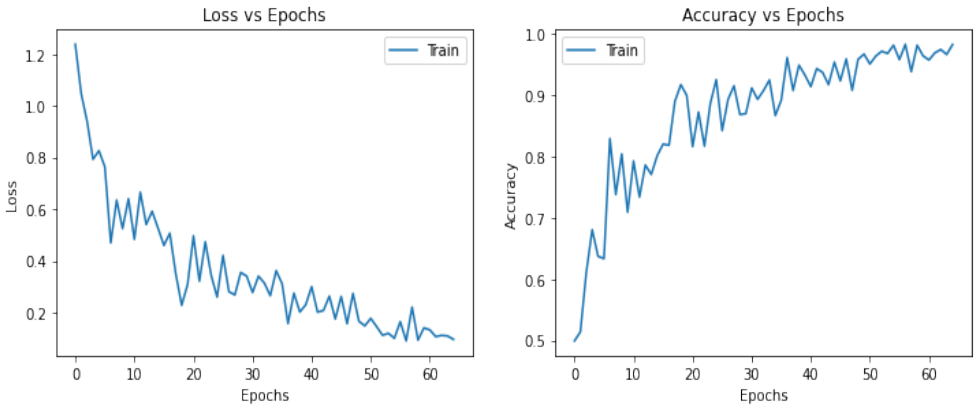


Figura 27: Gráficos das curvas de perda e de acurácia em função do número de épocas na base KTH com vídeos balanceados e editados.

Com esse modelo treinado, ele obteve uma performance consideravelmente superior na base de treino em comparação com o treino realizado com frames balanceados. Quanto aos resultados obtidos na base de teste, a acurácia ficou mais equilibrada entre as 4 classes. Enquanto que o modelo treinado com frames balanceados possuía acurácia elevada para *correr*, mediano para *andar* e pecava para *bater palmas* e *acenar*, o modelo com vídeos balanceados performa muito bem para *andar*, *correr* e *bater palmas*, e continua com uma performance mediana porém superior para *acenar*.

É importante destacar que para calcular as acurácias de cada classe, dividimos a quantidade de acertos pela quantidade de vídeos. Em contrapartida, para calcular a acurácia da base, fizemos uma média simples, somando todas as acurácias e dividindo pelo número de classes, uma vez que se fizéssemos a divisão pela quantidade de vídeos total, as classes *andar* e *correr* teriam um peso 4 vezes maior que as classes *acenar* e *bater palmas*.

Tabela 6: Acurácia na base de teste

Classe	**Acurácia de teste**
Andar	92.19%
Correr	92.19%
Acenar	60.00%
Bater Palmas	94.29%
Base completa	84.67%

4.4 Base de dados própria

Os vídeos da base de dados própria seguem a mesma estrutura dos vídeos da base Weizmann e KTH: eles contêm apenas ações de uma única pessoa executadas em um ambiente controlado. No momento há um total de 560 vídeos com um tamanho total de 615,8 MB. Eles representam 80 pessoas distintas com cada uma executando 7 ações diferentes: pular no lugar, acenar com as duas mãos, bater palmas, andar da direita para a esquerda e da esquerda para a direita e correr da esquerda para a direita e da direita para a esquerda. Além disso, essas 7 ações foram classificadas em apenas 5 classes: pular, acenar, bater palmas, andar e correr. Essa relação entre as ações e suas respectivas classes podem ser vistas na tabela 7 e alguns exemplos de uma pessoa executando tais ações podem ser vistos na figura 28.

Tabela 7: Tabela com relação de ações presentes nas bases de dados escolhidas

Ações	Classes
Pular no lugar	Pular
Acenar com as duas mãos	Acenar
Bater palmas	Bater palmas
Andar da direita para a esquerda	Andar
Andar da esquerda para a direita	Andar
Correr da direita para a esquerda	Correr
Correr da esquerda para a direita	Correr

As ações foram escolhidas de modo a se obter uma quantidade de vídeos o mais balanceada possivel entre as classes, utilizando-se também as bases Weizmann e KTH.

Figura 28: Exemplos das ações acenar, pular, bater palmas, andar e correr da base de dados própria

4.4.1 Pré processamento

A etapa de pré processamento é a mesma que foi aplicada aos conjuntos de dados Weizmann e KTH e segue a mesma estrutura descrita na figura 12. Um exemplo da etapa de pré processamento aplicada na ação “handwaving"do dataset contendo os cinco canais de informação é apresentado nas figuras 29, 30, 31, 32, 33 e 34.

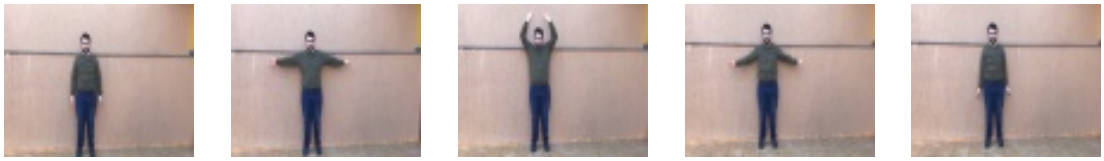


Figura 29: Frames originais da ação "handwaving"da base de dados própria.

Figura 30: Frames em tons de cinza após scaling e foreground extraction

Figura 31: Gradiente na direção X a partir dos frames em tons de cinza

Figura 32: Gradiente na direção Y a partir dos frames em tons de cinza

Figura 33: Fluxo óptico na direção X a partir dos frames em tons de cinza

Figura 34: Fluxo óptico na direção Y a partir dos frames em tons de cinza

4.4.2 Conjuntos de treino e de teste

De início, foi feita a seguinte divisão: 70 pessoas para o conjunto de treino e 10 pessoas para o conjunto de teste. Conforme explicado na seção 4.4, cada pessoa executa duas ações para as classes *andar e correr* (da direita para a esquerda e da esquerda para a direita). Isso implica que as classes *pular, acenar e bater palmas* terão 70 vídeos cada uma no conjunto de treino, enquanto que as classes *andar e correr* terão 140 vídeos cada uma. Adicionalmente, conforme explicado no final da seção 4.2.1, o desbalanço no número de frames por classe pode ser um problema no treinamento da rede. Por isso, também foi feita a contagem de frames por classe para cada um dos conjuntos. Essas relações entre a quantidade de vídeos e de frames de cada uma das classes para cada um dos subconjuntos de dados está expressa na tabela 8.

Tabela 8: Tabela com número de frames no dataset Weizmann antes e depois da técnica de data augmentation

Classes	Quantidade de vídeos - Quantidade de frames		
	Treino	Teste	Total
Pular	70 - 9049	10 - 1196	80 - 10245
Acenar	70 - 9872	10 - 1320	80 - 11192
Bater palmas	70 - 10021	10 - 1149	80 - 11170
Andar	140 - 13745	20 - 1713	160 - 15458
Correr	140 - 7430	20 - 860	160 - 8290

4.4.3 Treinamento da rede

A rede foi treinada por um total de 40 épocas atingindo uma acurácia e perda de treino de 99,78% e 0,0057, respectivamente (vide figura 35). Porém, apesar dos resultados satisfatórios no treino, observamos que ocorreu overfitting após a oitava época, pois, segundo a tabela 9, após a oitava época a acurácia no conjunto de teste se estabilizou. Portanto, será necessário implementar técnicas de regularização na arquitetura da rede neural para melhorar a capacidade de generalização da mesma.

4.5 Base de dados final

A base de dados final foi composta combinando os vídeos da base de dados própria com os vídeos da base de dados KTH e, portanto, ela também contêm

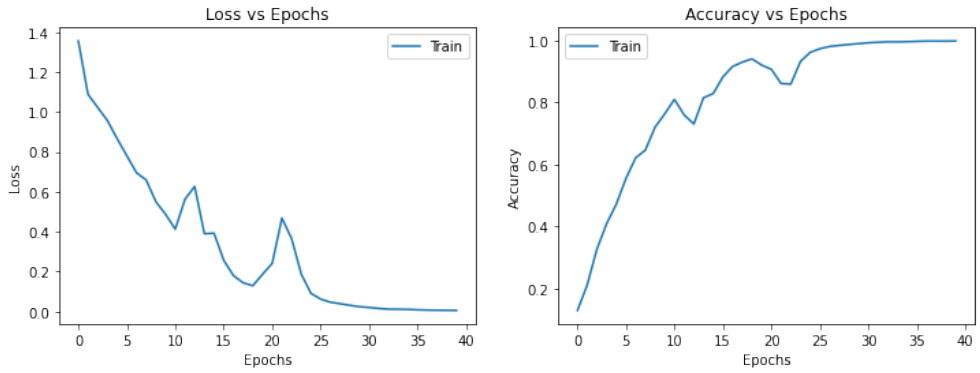


Figura 35: Curvas de acurácia e de perda após 40 épocas para o treinamento da rede na base de dados própria

Tabela 9: Acurácia e perda no conjunto de dados de teste utilizando a técnica de majority voting

Classes	8 épocas	9 épocas	10 épocas	11 épocas	40 épocas
Acenar	20%	20%	20%	40%	40%
Bater palmas	50%	30%	30%	50%	40%
Pular	70%	90%	80%	80%	50%
Andar	65%	75%	75%	75%	65%
Correr	70%	65%	70%	60%	75%
Acurácia geral	59%	60%	60%	63%	58,57%

apenas ações de uma única pessoa executadas em um ambiente controlado. A tabela 10 mostra a quantidade de vídeos de cada classe para cada uma das bases de dados.

Tabela 10: Quantidade de vídeos de cada classe para cada dataset

Base de dados		**KTH**	**Própria**	**Total**
Quantidade de vídeos por classe	**Andar**	400	160	560
	Correr	400	160	560
	Pular	0	80	80
	Bater Palmas	100	80	180
	Acenar	100	80	180
	Total	1000	560	1560

Segundo a tabela 10, há uma discrepância significativa entre a quantidade de vídeos de cada classe. A base KTH, por exemplo, conforme a seção 4.3.1, não possui nenhum vídeo da classe pular. Além disso, cada vídeo das classes andar e correr foram processados de forma a gerar 4 vídeos e, por isso, estas possuem

quatro vezes mais vídeos do que as classes bater palmas e acenar. Já no caso da base de dados própria, foi explicado na seção 4.4, que as classes correr e andar possuem duas vezes mais vídeos do que as restantes.

4.5.1 Resultados sem regularização

Inicialmente, a rede foi treinada sem o uso de regularização e utilizamos 64% dos vídeos para treino e 36% dos vídeos para teste, conforme apresentado na tabela 11.

Tabela 11: Quantidade de vídeos para treino e teste para cada classe de cada base de dados

Base de Dados		**KTH**	**Próprio**	**Total**
Quantidade de vídeos de treino/teste	**Andar**	256/144	102/58	358/202
	Correr	256/144	102/58	358/202
	Pular	0	51/29	51/29
	Bater Palmas	64/36	51/29	115/65
	Acenar	64/36	51/29	115/65
	Total	640/360	357/203	997/563

Treinando a rede com os mesmos parâmetros utilizados para o treino da base Weizmann e KTH (otimizador SGD, função de ativação ReLU, entropia cruzada categórica como função de perda, acurácia como métrica de classificação e *learning rate* de 0,001) por 40 épocas, obtivemos uma acurácia de treino de 98,73% e as acurácias de teste mostradas na tabela 12.

Tomando a classe correr como exemplo, uma acurácia de 62,4% significa que $\frac{64,2}{100} * 202 = 126$ dos vídeos de teste foram classificados corretamente. Agora, a média por quantidade de classes de 62,02% consiste apenas em uma média simples entre as acurácias, ou seja, $\frac{0,99+62,4+64,6+60+24,1}{5} = 62,02\%$. Já a média por quantidade de vídeos de 73,5% consiste em em uma média ponderada que dá mais peso às classes andar e correr por terem mais vídeos do que as outras e reflete de forma acurada a quantidade de vídeos totais que foram classificados corretamente, que são $73,5\% * 563 = 414$ vídeos. Em expressões matemáticas:

$$\frac{0,99 * 202 + 0,624 * 202 + 0,646 * 65 + 0,6 * 65 + 0,241 * 29}{202 + 202 + 65 + 65 + 29} = 73,5\%$$

Tabela 12: Quantidade de vídeos para treino e teste para cada classe de cada base de dados

Classe	**Acurácia de teste**	**Quantidade de vídeos**	**Vídeos classificados corretamente**
Andar	99%	202	99% * 202 = 200
Correr	62,4%	202	62,4% * 202 = 126
Acenar	64,6%	65	64,6% * 65 = 42
Bater Palmas	60%	65	60% * 65 = 39
Pular	24,1%	29	24,1% * 29 = 7
Média por quantidade de classes	62,02%	-	-
Média por quantidade de vídeos	73,5%	563	73,53% * 563 = 414

4.6 Regularização e otimização de hiperparâmetros

Observando os resultados listados na tabela 12, podemos observar que a arquitetura treinada está com um problema de sobreajuste, ou seja, ela está aplicando um peso muito grande aos ruídos presentes nos dados de treino durante seu aprendizado o que reduz a capacidade da rede de generalizar bem para dados nunca vistos (o que justifica uma grande discrepância entre a elevada acurácia nos dados de treino e a baixa acurácia nos dados de teste).

Para contornar o problema do sobreajuste, as seções irão focar em otimizar os hiperparâmetros do modelo e em algumas técnicas de regularização. Para todas as execuções desta seção, pode-se considerar que o modelo foi treinado por 40 épocas, com exceção da configuração da seção 4.6.6.

4.6.1 Escolha do otimizador

A escolha do otimizador correto para minimizar a função de perda de uma rede neural é essencial para o bom desempenho da mesma. Portanto, foram testados os reconhecidos otimizadores Adam (*Adaptive Moment Estimation*), *Adagrad* e *Adamax* [75]. A configuração da rede e os parâmetros utilizados são os mesmos da seção anterior 4.5.1 e a tabela 13 mostra os resultados obtidos para cada um dos otimizadores. Segundo a tabela, pode-se verificar que o otimizador *Adamax* obteve o melhor desempenho, mas que houve um *overfitting* significativo,

considerando as acurácias de treino e de teste. Portanto, nas próximas seções serão explorados diferentes valores de hiperparâmetros e diferentes métodos de regularização sempre utilizando o otimizador *Adamax*.

Tabela 13: Comparação de diferentes otimizadores para treino de 40 épocas

Otimizador	**Acurácia - Treino**	**Acurácia - Teste**
SGD	98,73%	Andar: 99% Correr: 62,4% Acenar: 64,6% Palmas: 60% Pular: 24,1% Média por quantidade de classes: 62,02% Média por quantidade de vídeos: 73,53%
Adam	98,86%	Andar: 100% Correr: 77,72% Acenar: 80% Palmas: 67,69% Pular: 37,93% Média por quantidade de classes: 73,07% Média por quantidade de vídeos: 82,77%
Adagrad	92,85%	Andar: 96,95% Correr: 73,17% Acenar: 80,43% Palmas: 58,7% Pular: 50% Média por quantidade de classes: 71,85% Média por quantidade de vídeos: 80,93%
Adamax	**99,47%**	Andar: 96,43% Correr: 92,86% Acenar: 91,67% Palmas: 47,22% Pular: 56,25% Média por quantidade de classes: **76,88%** Média por quantidade de vídeos: **86,86%**

4.6.2 Escolha do *Learning Rate* inicial

Um dos hiperparâmetros de suma importância para o bom desempenho de uma rede neural é o *learning rate* inicial. Segundo [76], valores típicos deste parâmetro se encontram no intervalo $[10^{-6}, 1]$. Portanto, foram testados os valores $\{0,00025; 0,0005; 0,00075; 0,000875; 0,001\}$ novamente para 40 épocas, conforme mostra a tabela 14, e foi concluído que o *learning rate* de 0,001 seria o mais adequado para o treinamento da rede neural.

4.6.3 Escolha da função de ativação

Para atingir resultados do estado da arte, é essencial a escolha de uma boa função de ativação. Portanto esta seção é dedicada à exploração de diferentes funções de ativação. Até o presente momento, foi usada apenas a função de ativação ReLU (*Rectified Linear Unit*), mas outras alternativas como om $PReLU$, ELU, *Leaky* $ReLU$ e $SELU$ serão exploradas nesta seção [77]. Conforme a tabela 15, os resultados obtidos pelas diferentes funções de ativação são bem parecidos, porém não melhores que os resultados obtidos na seção anterior com o uso do $ReLU$. Portanto, a função de ativação $ReLU$ continuará a ser adotada para os próximos testes.

4.6.4 Escolha da taxa de *Dropout*

Uma técnica muito comum de regularização é a técnica de *dropout*. Ela consiste consiste em zerar aleatoriamente alguns dos valores de entrada nas camadas da rede neural, o que contribui para tornar a rede mais robusta a variações nos dados de entrada e previne a ocorrência do sobreajuste [78].

Segundo [78], valores ótimos da taxa de *dropout* variam entre 0 e 50%. Para o caso da nossa arquitetura, as camadas de *dropout* foram adicionadas após as camadas convolucionais e, portanto, foram testados diferentes combinações de taxas de *dropout* para as três camadas convolucionais presente no modelo, conforme a tabela 16. A partir desta, pode-se concluir que as taxas de dropout de 15% nas três camadas convolucionais trouxe o melhor desempenho e, portanto, serão usadas para os próximos testes.

Tabela 14: Comparação do desempenho de modelos com otimizador Adamax treinados por 40 épocas testando diferentes *learning rates*

Learning Rate	**Acurácia - Treino**	**Acurácia - Teste**
0,00025	98,4%	Andar: 100% Correr: 67,3% Acenar: 70,8% Palmas: 81,5% Pular: 6,9% Média por quant. de classes: 65,3% Média por quant. de vídeos: 78%
0,0005	99,53%	Andar: 99,5% Correr: 73,3% Acenar: 78,5% Palmas: 75,4% Pular: 24,1% Média por quant. de classes: 70,2% Média por quant. de vídeos: 81%
0,00075	86,33%	Andar: 0% Correr: 0% Acenar: 0% Palmas: 100% Pular: 0% Média por quant. de classes: 20% Média por quant. de vídeos: 11,6%
0,000875	99,77%	Andar: 99,5% Correr: 71,78% Acenar: 76,92% Palmas: 81,54% Pular: 44,83% Média por quant. de classes: 74,91% Média por quant. de vídeos: 82,06%
0,001	99,47%	Andar: 96,4% Correr: 92,9% Acenar: 91,7% Palmas: 47,2% Pular: 56,3% Média por quant. de classes: 76,9% Média por quant. de vídeos: 86,9%

4.6.5 Base de treino, teste e de validação

A próxima subseção focará no uso das técnicas de regularização de *Learning Rate Decay* e de *Early Stopping*. Para implementá-las de forma eficiente, é ne-

Tabela 15: Comparação do desempenho de modelos com otimizador Adamax e *Learning Rate* inicial de 0,001 treinados por 40 épocas testando diferentes funções de ativação

Função de ativação	**Acurácia - Treino**	**Acurácia - Teste**
ELU	94,32%	Andar: 98,21% Correr: 74,11% Acenar: 63,89% Palmas: 80,56% Pular: 12,5% Média por quant. de classes: 65,9% Média por quant. de vídeos: 79,2%
SELU	99,57%	Andar: 99% Correr: 72,8% Acenar: 73,9% Palmas: 60% Pular: 31% Média por quant. de classes: 67,3% Média por quant. de vídeos: 78,7%
PReLU	99,61%	Andar: 99,5% Correr: 73,27% Acenar: 70,77% Palmas: 70,77% Pular: 27,59% Média por quant. de classes: 68,4% Média por quant. de vídeos: 79,8%
Leaky ReLU	99,4%	Andar: 100% Correr: 71,3% Acenar: 73,9% Palmas: 83,1% Pular: 24,1% Média por quant. de classes: 70,5% Média por quant. de vídeos: 80,8%

cessário dividir a base de dados de outra forma, que inclui uma base de dados de validação, além da base de treino e teste. A proporção escolhida foi de 60%, 20% e 20% dos vídeos totais, respectivamente, para as bases de dados de treino, teste e validação. Tal divisão está expressa na tabela 17.

4.6.6 *Learning Rate Decay* e *Early Stopping*

Empiricamente, a técnica de *Learning Rate Decay* traz grandes benefícios à otimização e à generalização de uma rede neural. Ela técnica consiste em redu-

Tabela 16: Comparação do desempenho de modelos com otimizador Adamax, *Learning Rate* inicial de 0,001 e função de ativação ReLU treinados por 40 épocas testando diferentes taxas de *dropout*

Taxa de Dropout em cada camada convolucional	**Acurácia - Treino**	**Acurácia - Teste**
15% - 15% - 15%	**99,66%**	Andar: 99,5% Correr: 75,7% Acenar: 81,5% Palmas: 80% Pular: 75,9% Média por quant. de classes: **82,5%** Média por quant. de vídeos: **85,4%**
25% - 25% - 25%	99,59%	Andar: 100% Correr: 72,3% Acenar: 86,2% Palmas: 78,5% Pular: 44,8% Média por quant. de classes: 76,3% Média por quant. de vídeos: 83,1%
25% - 50% - 25%	98,4%	Andar: 100% Correr: 73,3% Acenar: 72,3% Palmas: 78,5% Pular: 31% Média por quant. de classes: 71% Média por quant. de vídeos: 81,2%
37,5%-37,5%-37,5%	98,09%	Andar: 100% Correr: 71,8% Acenar: 66,2% Palmas: 84,6% Pular: 37,9% Média por quant. de classes: 72,1% Média por quant. de vídeos: 81%
37,5%-37,5%-50%	98,23%	Andar: 100% Correr: 69,8% Acenar: 70,8% Palmas: 89,2% Pular: 31% Média por quant. de classes: 72,2% Média por quant. de vídeos: 81%

Tabela 17: Quantidade de vídeos de cada classe para base de treino, teste e validação

Dataset		**KTH**	**Próprio**	**Total**
Quantidade de vídeos de treino/teste/ validação	**Andar**	240/80/80	96/32/32	336/112/112
	Correr	240/80/80	96/32/32	336/112/112
	Pular	0	48/16/16	48/16/16
	Bater Palmas	60/20/20	48/16/16	108/36/36
	Acenar	60/20/20	48/16/16	108/36/36
	Total	600/200/200	336/112/112	936/312/312

zir o *learning rate* quando certas condições são satisfeitas [79]. Ainda segundo [79], acredita-se que tal técnica previne o modelo de memorizar ruídos e ajuda o mesmo a aprender padrões complexos. Portanto, para a arquitetura proposta, foi implementado um *learning rate decay* que reduz o *learning rate* a 20% do valor anterior sempre que a perda de validação não diminuir após cinco épocas por um valor mínimo de 0,01.

Adicionalmente ao *Learning Rate Decay*, pode-se usar a técnica de *Early Stopping*. O funcionamento dessa técnica é bem descrito na figura 36 e ela previne o sobreajuste parando o treinamento da rede quando a acurácia da base de dados de validação deixa de aumentar [80]. Portanto, foi implementado um *Early Stopping* que para o treinamento da rede caso a perda de validação não diminuir após oito épocas por um valor mínimo de 0,01.

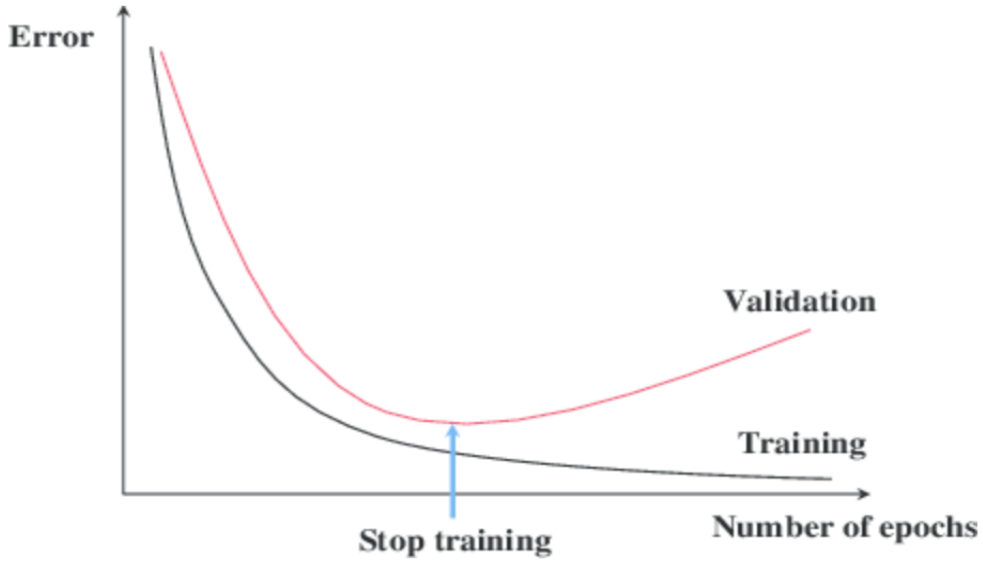


Figura 36: Funcionamento da técnica *early stopping*. Extraído de [81].

Os resultados obtidos estão mostrados na tabela 18. Segundo esta, há redução no *learning rate* após a época 15, pois não houve redução da perda de validação das épocas 11 a 15. Além disso, pode-se observar também que houve a ocorrência do *Early Stopping* após a época 18, visto que não houve redução na perda de validação por 8 épocas (da época 11 à época 18). Comparação do desempenho de modelos com otimizador Adamax, *Learning Rate* inicial de 0,001 e função de ativação ReLU treinados por 40 épocas testando diferentes taxas de *dropout*

Tabela 18: Acurácia e perda do modelo com otimizador Adamax, *Learning Rate* inicial de 0,001, função de ativação ReLU e camadas com Dropout de 15% para as bases de dados de treino e validação utilizando as técnicas de *Learning Rate Decay* e *Early Stopping*.

Época	**Treino Acurácia - Perda**	**Validação Acurácia - Perda**	**Learning Rate**
1	38,6% - 1,17	53,8% - 0,865	0,001
2	58,7% - 0,81	63,1% - 0,725	0,001
3	68,3% - 0,66	69,3% - 0,65	0,001
4	76,7% - 0,53	76,8% - 0,5	0,001
5	80% - 0,45	80,1% - 0,44	0,001
6	84% - 0,39	79,9% - 0,489	0,001
7	86% - 0,33	84% - 0,37	0,001
8	89,2% - 0,27	85,4% - 0,4	0,001
9	89,3% - 0,27	84,4% - 0,374	0,001
10	92,1% - 0,2	87,2% - 0,341	0,001
11	93,3% - 0,16	84,3% - 0,39	0,001
12	94,1% - 0,15	88,5% - 0,337	0,001
13	95,2% - 0,126	85,8% - 0,468	0,001
14	96,1% - 0,104	88,3% - 0,346	0,001
15	96,4% - 0,097	88,9% - 0,385	0,001
16	98,3% - 0,05	88,8% - 0,413	0,0002
17	98,5% - 0,044	88,9% - 0,407	0,0002
18	98,6% - 0,04	89,2% - 0,417	0,0002

Por fim, utilizando a técnica de *majority voting*, o modelo conseguiu atingir as acurácias de teste descritas na tabela 19.

4.6.7 Tempo de execução

Utilizando a divisão de treino e de teste especificada na seção 4.5.1, foi observado que, utilizando a CPU do Google Colab, cada época de treino levava em

Tabela 19: Acurácias de teste do modelo com as técnicas de *Learning Rate Decay* e *Early Stopping*.

Classe	**Acurácia de teste**
Andar	98,21%
Correr	93,75%
Acenar	88,89%
Bater Palmas	77,78%
Pular	93,75%
Acurácia média por quant. de classes	90,48%
Acurácia média por quant. de vídeos	92,95%

torno de 2 horas e 45 minutos. Ou seja, para rodar as 40 épocas, o programa levaria aproximadamente 111 horas e 50 minutos para terminar a execução. Mas utilizando a GPU, a duração de cada época de treino reduziu drasticamente para aproximadamente 13 minutos. Ou seja, o treino completo de 40 épocas levava em torno de 8 horas e 50 minutos.

4.7 Análise dos resultados

Conforme a seção 4.5, a arquitetura proposta partiu de uma acurácia de teste por quantidade de classes e por quantidade de vídeos relativamente baixas de, respectivamente, 62,02% e 73,5%. Porém, após a otimização dos hiperparâmetros e a implementação de técnicas de regularização tais acurácias saltaram para, respectivamente, 90,48% e 92,95%. A tabela 20 resume os ganhos trazidos por cada etapa de melhoria da seção 4.5.

Uma observação importante a se fazer é o aumento drástico na acurácia de teste da classe pular. Isso se deve ao fato de que, conforme a tabela 10, há poucos vídeos desta classe. A classe pular tem, exatamente, 7 vezes menos observações que as classes andar e correr e 2,25 vezes menos observações que as classes bater palmas e acenar. Portanto, é razoável pensar que nas primeiras duas configurações (inicial e com a adição do otimizador Adamax), ocorre *overfitting* do modelo nas classes com mais observações e, por isso, as acurácias da classe pular são bem baixas (24,1% e 56,25%). Tal argumento é ainda mais reforçado tendo em vista as melhorias causadas pelo uso das técnicas de *dropout*, *learning rate decay* e *early stopping*, visto que, através delas, a acurácia salta para um patamar de 93,75%.

Tabela 20: Acurácias de testes após cada etapa de otimização de hiperparâmetros e regularização

Configuração	**Acurácia de teste**
Inicial	Andar: 99% Correr: 62,4% Acenar: 64,6% Palmas: 60% Pular: 24,1% Média por quant. de classes: 62,02% Média por quant. de vídeos: 73,53%
Otimizador: Adamax	Andar: 96,43% Correr: 92,86% Acenar: 91,67% Palmas: 47,22% Pular: 56,25% Média por quant. de classes: 76,88% Média por quant. de vídeos: 86,86%
Dropout 15% - 15% - 15%	Andar: 99,5% Correr: 75,7% Acenar: 81,5% Palmas: 80% Pular: 75,9% Média por quant. de classes: 82,5% Média por quant. de vídeos: 85,4%
Learning Rate Decay/ Early Stopping	Andar: 98,21% Correr: 93,75% Acenar: 88,89% Palmas: 77,78% Pular: 93,75% Média por quant. de classes: 90,48% Média por quant. de vídeos: 92,95%

4.8 Comparativo com o artigo original

Esta seção fará uma comparação entre os resultados obtidos por este trabalho, e o artigo no qual nos baseamos para implementar a arquitetura da rede neural convolucional 3D [13].

Observando a tabela 21, o modelo desenvolvido na seção 4.3, sem o uso de qualquer técnica de regularização, apresentou um resultado superior ao do artigo original nas classes de *bater palmas* e *correr* e inferior nas classes *andar* e *acenar*. A melhora no desempenho da classe *correr* (92.19% contra 79%) se deve ao

Tabela 21: Comparativo de acurácia entre o artigo original e o presente trabalho, utilizando a técnica de majority voting sobre os dados da base KTH.

Classes	**Sem Regularização**	**Com Regularização**	**Artigo original**
Acenar	60%	95%	94%
Bater palmas	94,29%	85%	90%
Andar	92,19%	97,5%	97%
Correr	92,19%	96,25%	79%
Acurácia geral	**84,67%**	**93,44%**	**90%**

trabalho feito para editar os vídeos, removendo os frames que não adicionavam qualquer informação sobre a ação (procedimento descrito na seção 4.3.1) - isto auxilia no treinamento da rede neural. Os vídeos da classe *andar* também tiveram seus frames "vazios"removidos, porém a acurácia de teste em nosso trabalho acabou ficando um pouco pior do que o artigo original (92.19% contra 97%), uma vez que seus vídeos possuíam uma menor quantidade de frames a serem removidos em comparação com a classe *correr*, e desta maneira, a remoção destes frames não foi relevante para melhorar a acurácia de teste desta classe. Para a classe *bater palmas*, a acurácia no modelo sem regularização foi levemente superior ao do artigo original (94.29% contra 90%). Na classe *acenar*, o desempenho foi consideravelmente pior. Analisando o comportamento da arquitetura mais detalhadamente, notamos que as predições feitas em cima de uma grande quantidade dos frames desta classe foram atribuídas a ação de *bater palmas*, o que indica que há um sobreajuste presente na rede neural, uma vez que o desempenho da arquitetura na base de treino foi excelente para todas as ações (95,6% de acurácia).

De maneira a entender o impacto que o uso da regularização possui na performance da rede neural nos dados de teste, foi realizado o treinamento desta rede neural convolucional 3D usando as mesmas técnicas de regularização descritas nas seções 4.6.4 e 4.6.6. Desta forma, foram inseridas as camadas de dropout antes de cada camada convolucional da arquitetura com uma taxa de 0.15, e a base de dados utilizada foi a KTH com as mesmas 4 classes: andar, correr, bater palmas e acenar (com 400, 400, 100 e 100 vídeos para cada classe, respectivamente). Os pesos aplicados para a divisão dos vídeos nas bases de treino, validação e teste foram de 60%, 20% e 20% respectivamente e aplicados igualmente para cada uma das 4 classes.

Conforme a tabela 21, houve uma melhora drástica no desempenho da rede após a adição de regularização. Comparando com o modelo sem regularização, as classes *acenar* (60% para 95%), *andar* (92,19% para 97,5%) e *correr* (92,19% para 96,25%) tiveram um aumento nas acurácias. Porém a inclusão das técnicas de regularização acabaram deteriorando a acurácia na classe *bater palmas* (de 94,29% para 85%), por mais que a acurácia ainda seja alta.

Comparando com o artigo original[13], o modelo com regularização desenvolvido por este trabalho foi superior em todas as classes, com exceção da classe *bater palmas* e também obteve uma acurácia geral superior. Acreditamos que isso se deve ao fato de que os autores do artigo original não exploraram de forma suficiente diferentes técnicas de regularização e otimização de hiperparâmetros, como foi feito na seção 4.5.

5 Conclusão

O projeto proposto se baseia na construção de um modelo baseado em aprendizado de máquina para ser treinado sobre um conjunto de dados contendo vídeos de diversas pessoas executando ações cotidianas.

Após uma extensa pesquisa bibliográfica no campo de reconhecimento de atividades humanas, pudemos conhecer e entender quais são as categorias de problemas a serem resolvidas dentro deste escopo, e as técnicas adequadas para cada tipo de ação. Embasados por esta pesquisa, optamos pelo problema de detecção de atividades realizadas por uma única pessoa, e o modelo que escolhemos é a rede neural convolucional 3D, capaz de realizar a operação de convolução em 3 dimensões (2 dimensões espaciais e a temporal).

Após realizar a implementação de todo o pipeline de pré-processamento de dados e da arquitetura da rede neural, realizamos o treinamento da rede em cima de bases de dados abertas obtidas na internet. Enfrentamos alguns problemas com respeito a esses dados, descritos na seção 4. Após realizar alguns tratamentos nos vídeos, especialmente na base KTH, conseguimos chegar a acurácias satisfatórias, atingindo um índice de acertos de 85% com os dados de teste. Além disso, pudemos confirmar a importância da regularização em um modelo de aprendizado profundo, visto que o resultado final foi de 93% após a implementação de técnicas

como o *dropout* e o *early stopping*, sendo até mesmo superior à acurácia obtida pelo artigo [13], no qual foi baseado o modelo do presente trabalho.

De modo a validar o modelo, adquirimos 560 vídeos de cunho próprio e montamos uma outra base de dados misturando estes vídeos com os vídeos da base KTH. Inicialmente, o modelo não mostrou um desempenho muito bom, obtendo acurácias de teste por quantidade de classes e por quantidade de vídeos de, respectivamente, 62,02% e 73,53%. Porém após uma extensa exploração de hiperparâmetros e técnicas de regularização, tais acurácias saltaram para valores bastante satisfatórios de, respectivamente, 90,48% e 92,95%. Esses resultados mostram que o modelo final é bastante robusto ao problema de desbalanceamento de classes, visto que algumas classes tinham muito menos vídeos do que outras, e demonstram a eficácia do modelo convolucional em 3 dimensões no reconhecimento de dados em formato de vídeo.

Por fim, realizamos a comparação entre os resultados obtidos pelo presente trabalho e no artigo original. Concluímos que a edição dos vídeos da classe *correr*, retirando os frames "vazios" teve um impacto muito positivo na performance da arquitetura para esta categoria, o que reforça a importância de se usar dados de qualidade para realizar o treino de uma rede neural. Adicionalmente, o uso de técnicas de regularização mostrou ter um impacto muito positivo no desempenho da rede nas acurácias de teste. Estes 2 fatores explicam a melhora obtida nas acurácias por este presente trabalho, em comparação com o artigo original [13].

Referências

[1] Samitha Herath, Mehrtash Harandi e Fatih Porikli. «Going deeper into action recognition: A survey». Em: *Image and Vision Computing* 60 (abr. de 2017), pp. 4–21. ISSN: 02628856. DOI: 10.1016/j.imavis.2017.01.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0262885617300343.

[2] Andrej Karpathy et al. «Large-scale video classification with convolutional neural networks». Em: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2014), pp. 1725–1732. ISSN: 10636919. DOI: 10.1109/CVPR.2014.223.

[3] X. Wang, A. Farhadi e A. Gupta. «Actions Transformations». Em: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2016, pp. 2658–2667. DOI: 10.1109/CVPR.2016.291.

[4] Jose M. Chaquet, Enrique J. Carmona e Antonio Fernández-Caballero. «A survey of video datasets for human action and activity recognition». Em: *Computer Vision and Image Understanding* 117.6 (2013), pp. 633–659. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2013.01.013. URL: http://www.sciencedirect.com/science/article/pii/S1077314213000295.

[5] C. J. Dhamsania e T. V. Ratanpara. «A survey on Human action recognition from videos». Em: *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*. Nov. de 2016, pp. 1–5. DOI: 10.1109/GET.2016.7916717.

[6] *10 Charts That Will Change Your Perspective On Artificial Intelligence's Growth*. https://www.forbes.com/sites/louiscolumbus/2018/01/12/10-charts-that-will-change-your-perspective-on-artificial-intelligences-growth/. Accessed: 2019-09-30.

[7] H. Abbas et al. «Machine learning for early detection of autism (and other conditions) using a parental questionnaire and home video screening». Em: *2017 IEEE International Conference on Big Data (Big Data)*. Dez. de 2017, pp. 3558–3561. DOI: 10.1109/BigData.2017.8258346.

[8] Dan Xu et al. *Learning Deep Representations of Appearance and Motion for Anomalous Event Detection*. 2015. arXiv: 1510.01553 [cs.CV].

[9] Samitha Herath, Mehrtash Tafazzoli Harandi e Fatih Porikli. «Going Deeper into Action Recognition: A Survey». Em: *CoRR* abs/1605.04988 (2016). arXiv: 1605.04988. URL: http://arxiv.org/abs/1605.04988.

[10] Samitha Herath, Mehrtash Harandi e Fatih Porikli. *Going Deeper into Action Recognition: A Survey*. 2016. arXiv: 1605.04988 [cs.CV].

[11] A. Kamel et al. «Deep Convolutional Neural Networks for Human Action Recognition Using Depth Maps and Postures». Em: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.9 (set. de 2019), pp. 1806–1819. DOI: 10.1109/TSMC.2018.2850149.

[12] Jindong Wang et al. «Deep learning for sensor-based activity recognition: A survey». Em: *Pattern Recognition Letters* 119 (2019). Deep Learning for Pattern Recognition, pp. 3–11. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.02.010. URL: http://www.sciencedirect.com/science/article/pii/S016786551830045X.

[13] «3D Convolutional neural networks for human action recognition». Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2013), pp. 221–231. ISSN: 01628828.

[14] Du Tran et al. «Learning spatiotemporal features with 3D convolutional networks». Em: *Proceedings of the IEEE International Conference on Computer Vision* 2015 Inter (2015), pp. 4489–4497. ISSN: 15505499. DOI: 10.1109/ICCV.2015.510. arXiv: arXiv:1412.0767v4.

[15] Karen Simonyan e Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. arXiv: 1406.2199 [cs.CV].

[16] Guangchun Cheng et al. «Advances in Human Action Recognition: A Survey». Em: *CoRR* abs/1501.05964 (2015). arXiv: 1501.05964. URL: http://arxiv.org/abs/1501.05964.

[17] Bhaskar Chakraborty et al. «A selective spatio-temporal interest point detector for human action recognition in complex scenes». Em: nov. de 2011, pp. 1776–1783. ISBN: 978-1-4577-1101-5. DOI: 10.1109/ICCV.2011.6126443.

[18] A. Eweiwi et al. «Temporal key poses for human action recognition». Em: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Nov. de 2011, pp. 1310–1317. DOI: 10.1109/ICCVW.2011.6130403.

[19] H. Wang e C. Schmid. «Action Recognition with Improved Trajectories». Em: *2013 IEEE International Conference on Computer Vision*. Dez. de 2013, pp. 3551–3558. DOI: 10.1109/ICCV.2013.441.

[20] Zhenzhong Lan et al. «Beyond Gaussian Pyramid: Multi-skip Feature Stacking for action recognition». Em: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2015, pp. 204–212. DOI: 10.1109/CVPR.2015.7298616.

[21] B. Fernando et al. «Modeling video evolution for action recognition». Em: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2015, pp. 5378–5387. DOI: 10.1109/CVPR.2015.7299176.

[22] Yingwei Li et al. «VLAD3: Encoding Dynamics of Deep Features for Action Recognition». Em: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2016.

[23] Shugang Zhang et al. «A Review on Human Activity Recognition Using Vision-Based Method». Em: *Journal of Healthcare Engineering* 2017 (jul. de 2017), pp. 1–31. DOI: 10.1155/2017/3090343.

[24] Michalis Vrigkas, Christophoros Nikou e Ioannis Kakadiaris. «A Review of Human Activity Recognition Methods». Em: *Frontiers in Robotics and Artificial Intelligence* 2 (nov. de 2015). DOI: 10.3389/frobt.2015.00028.

[25] Alexandros Stergiou e Ronald Poppe. *Analyzing Human-Human Interactions: A Survey*. 2018. arXiv: 1808.00022 [cs.CV].

[26] Khurram Soomro, Amir Roshan Zamir e Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*. 2012. arXiv: 1212.0402 [cs.CV].

[27] «UCF50: Action Recognition in Realistic Videos». Em: (). URL: http://crcv.ucf.edu/data/UCF50.php.

[28] Marcin Marszałek, Ivan Laptev e Cordelia Schmid. «Actions in Context». Em: *IEEE Conference on Computer Vision & Pattern Recognition*. 2009.

[29] M. S. Ryoo e J. K. Aggarwal. «Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities». Em: *2009 IEEE 12th International Conference on Computer Vision*. Set. de 2009, pp. 1593–1600. DOI: 10.1109/ICCV.2009.5459361.

[30] Tsz-Ho Yu, Tae-Kyun Kim e Roberto Cipolla. «Real-time Action Recognition by Spatiotemporal Semantic and Structural Forest». Em: *Proceedings of the British Machine Vision Conference*. doi:10.5244/C.24.52. BMVA Press, 2010, pp. 52.1–52.12. ISBN: 1-901725-40-5.

[31] A. Patron-Perez et al. «Structured Learning of Human Interactions in TV Shows». Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.12 (dez. de 2012), pp. 2441–2453. DOI: 10.1109/TPAMI.2012.24.

[32] Wongun Choi, Khuram Shahid e Silvio Savarese. «Learning context for collective activity recognition». Em: jul. de 2011, pp. 3273–3280. DOI: 10.1109/CVPR.2011.5995707.

[33] Wongun Choi e Silvio Savarese. «A Unified Framework for Multi-target Tracking and Collective Activity Recognition». Em: out. de 2012, pp. 215–230. DOI: 10.1007/978-3-642-33765-9_16.

[34] K. Yun et al. «Two-person interaction detection using body-pose features and multiple instance learning». Em: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. Jun. de 2012, pp. 28–35. DOI: 10.1109/CVPRW.2012.6239234.

[35] Marcel De Moraes, Sankha Mukherjee e Neil Robertson. «Deep Convolutional Poses for Human Interaction Recognition in Monocular Videos». Em: (dez. de 2016).

[36] Christoph Feichtenhofer, Axel Pinz e Andrew Zisserman. *Convolutional Two-Stream Network Fusion for Video Action Recognition*. 2016. arXiv: 1604.06573 [cs.CV].

[37] An Tran e Loong-Fah Cheong. *Two-stream Flow-guided Convolutional Attention Networks for Action Recognition*. 2017. arXiv: 1708.09268 [cs.CV].

[38] Gul Varol, Ivan Laptev e Cordelia Schmid. «Long-Term Temporal Convolutions for Action Recognition». Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018), pp. 1510–1517. ISSN: 01628828. DOI: 10.1109/TPAMI.2017.2712608.

[39] Moez Baccouche e Franck Mamalet. «for Human Action Recognition». Em: (2011), pp. 29–39.

[40] Jeff Donahue et al. «Appendices for Long-term Recurrent Convolutional Networks for Visual Recognition and Description A . Activity Recognition». Em: *Cvpr* Table 8 (2015), pp. 1–14. ISSN: 9781467369640. DOI: 10.1109/CVPR.2015.7298878. arXiv: arXiv:1411.4389v3.

[41] Karl Weiss, Taghi Khoshgoftaar e DingDing Wang. «A survey of transfer learning». Em: *Journal of Big Data* 3 (dez. de 2016). DOI: 10.1186/s40537-016-0043-6.

[42] Ningning Liu et al. «Associating Textual Features with Visual Ones to Improve Affective Image Classification». Em: *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction - Volume Part I*. ACII'11. Memphis, TN: Springer-Verlag, 2011, pp. 195–204. ISBN: 978-3-642-24599-2. URL: http://dl.acm.org/citation.cfm?id=2062780.2062805.

[43] M. Sazzad Hussain, Rafael A. Calvo e Payam Aghaei Pour. «Hybrid Fusion Approach for Detecting Affects from Multichannel Physiology». Em: *ACII*. 2011.

[44] O. Alzoubi et al. «Affect Detection and Classification from the Non-stationary Physiological Data». Em: *2013 12th International Conference on Machine Learning and Applications*. Vol. 1. Dez. de 2013, pp. 240–245. DOI: 10.1109/ICMLA.2013.49.

[45] M. A. Nicolaou, H. Gunes e M. Pantic. «Continuous Prediction of Spontaneous Affect from Multiple Cues and Modalities in Valence-Arousal Space». Em: *IEEE Transactions on Affective Computing* 2.2 (abr. de 2011), pp. 92–105. DOI: 10.1109/T-AFFC.2011.9.

[46] M. Babiker et al. «Automated daily human activity recognition for video surveillance using neural network». Em: *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*. Nov. de 2017, pp. 1–5. DOI: 10.1109/ICSIMA.2017.8312024.

[47] Rajesh Tripathi, Anand Jalal e Subhash Agrawal. «Suspicious human activity recognition: a review». Em: *Artificial Intelligence Review* 50 (fev. de 2017). DOI: 10.1007/s10462-017-9545-7.

[48] G. Sreenu e M A Durai. «Intelligent video surveillance: a review through deep learning techniques for crowd analysis». Em: *Journal of Big Data* 6 (jun. de 2019), p. 48. DOI: 10.1186/s40537-019-0212-5.

[49] Nadipuram Prasad, Jason King e Thomas Lu. «Machine intelligence-based decision-making (MIND) for Automatic Anomaly Detection - art. no. 65740f». Em: *Proceedings of SPIE - The International Society for Optical Engineering* 6574 (abr. de 2007). DOI: 10.1117/12.723635.

[50] D. Gowsikhaa e S. Abirami. «Suspicious Human Activity Detection from Surveillance Videos». Em: 2012.

[51] F. Harrou et al. «Vision-based fall detection system for improving safety of elderly people». Em: *IEEE Instrumentation Measurement Magazine* 20.6 (dez. de 2017), pp. 49–55. DOI: 10.1109/MIM.2017.8121952.

[52] Thou-Ho Chen, Ping-Hsueh Wu e Yung-Chuen Chiou. «An early fire-detection method based on image processing». Em: *2004 International Conference on Image Processing, 2004. ICIP '04.* Vol. 3. Out. de 2004, 1707–1710 Vol. 3. DOI: 10.1109/ICIP.2004.1421401.

[53] Vikas Tripathi et al. «Robust Abnormal Event Recognition via Motion and Shape Analysis at ATM Installations». Em: *Journal of Electrical and Computer Engineering* 2015 (fev. de 2015), pp. 1–10. DOI: 10.1155/2015/502737.

[54] Adrián Núñez-Marcos, Gorka Azkune e Ignacio Arganda-Carreras. «Vision-Based Fall Detection with Convolutional Neural Networks». Em: *Wireless Communications and Mobile Computing* 2017 (2017).

[55] S. Frizzi et al. «Convolutional neural network for video fire and smoke detection». Em: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society.* Out. de 2016, pp. 877–882. DOI: 10.1109/IECON.2016.7793196.

[56] Jackson Samuel et al. «Real time Violence Detection Framework for Football Stadium comprising of Big Data Analysis and Deep Learning through Bidirectional LSTM». Em: *Computer Networks* 151 (mar. de 2019). DOI: 10.1016/j.comnet.2019.01.028.

[57] Ahmad Arinaldi e Mohamad Ivan Fanany. «Cheating Video Description Based on Sequences of Gestures». Em: mai. de 2017. DOI: 10.1109/ICoICT.2017.8074679.

[58] A. Gupta, A. Kembhavi e L. S. Davis. «Observing Human-Object Interactions: Using Spatial and Functional Compatibility for Recognition». Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.10 (out. de 2009), pp. 1775–1789. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2009.83.

[59] Georgia Gkioxari et al. *Detecting and Recognizing Human-Object Interactions*. 2017. arXiv: 1704.07333 [cs.CV].

[60] Bingjie Xu et al. *Interact as You Intend: Intention-Driven Human-Object Interaction Detection*. 2018. arXiv: 1808.09796 [cs.CV].

[61] Victor Escorcia e Juan Carlos Niebles. «Spatio-temporal Human-Object Interactions for Action Recognition in Videos». Em: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. Jun. de 2013.

[62] Ben Packer, K. Saenko e Daphne Koller. «A combined pose, object, and feature model for action understanding». Em: jun. de 2012, pp. 1378–1385. ISBN: 978-1-4673-1226-4. DOI: 10.1109/CVPR.2012.6247824.

[63] L. Wang et al. «Context-Associative Hierarchical Memory Model for Human Activity Recognition and Prediction». Em: *IEEE Transactions on Multimedia* 19.3 (mar. de 2017), pp. 646–659. ISSN: 1941-0077. DOI: 10.1109/TMM.2016.2617079.

[64] Yu-Wei Chao et al. *Learning to Detect Human-Object Interactions*. 2017. arXiv: 1702.05448 [cs.CV].

[65] Ashesh Jain et al. «Structural-RNN: Deep Learning on Spatio-Temporal Graphs». Em: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2016.

[66] M. Blank et al. «Actions as space-time shapes». Em: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. Out. de 2005, 1395–1402 Vol. 2. DOI: 10.1109/ICCV.2005.28.

[67] C. Schuldt, I. Laptev e B. Caputo. «Recognizing human actions: a local SVM approach». Em: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 3. Ago. de 2004, 32–36 Vol.3. DOI: 10.1109/ICPR.2004.1334462.

[68] L. Kang et al. «Convolutional Neural Networks for Document Image Classification». Em: *2014 22nd International Conference on Pattern Recognition*. Ago. de 2014, pp. 3168–3172. DOI: 10.1109/ICPR.2014.546.

[69] *3dConv*. https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610. Accessed: 2019-11-26.

[70] *Frame Rates*. https://www.mediacollege.com/video/frame-rate/. Accessed: 2019-11-26.

[71] Y-Lan Boureau, Jean Ponce e Yann LeCun. «A theoretical analysis of feature pooling in visual recognition». Em: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 111–118.

[72] Martin Abadi et al. «TensorFlow: A system for large-scale machine learning». Em: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[73] J. Arunnehru, G. Chamundeeswari e S. Prasanna Bharathi. «Human Action Recognition using 3D Convolutional Neural Networks with 3D Motion Cuboids in Surveillance Videos». Em: *Procedia Computer Science* 133 (2018). International Conference on Robotics and Smart Manufacturing (RoSMa2018), pp. 471–477. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.07.059. URL: http://www.sciencedirect.com/science/article/pii/S1877050918310044.

[74] Connor Shorten e Taghi M. Khoshgoftaar. «A survey on Image Data Augmentation for Deep Learning». Em: *Journal of Big Data* 6 (2019), pp. 1–48.

[75] Dami Choi et al. «On Empirical Comparisons of Optimizers for Deep Learning». Em: *CoRR* abs/1910.05446 (2019). arXiv: 1910.05446. URL: http://arxiv.org/abs/1910.05446.

[76] Yoshua Bengio. «Practical recommendations for gradient-based training of deep architectures». Em: *CoRR* abs/1206.5533 (2012). arXiv: 1206.5533. URL: http://arxiv.org/abs/1206.5533.

[77] Chigozie Nwankpa et al. «Activation Functions: Comparison of trends in Practice and Research for Deep Learning». Em: *CoRR* abs/1811.03378 (2018). arXiv: 1811.03378. URL: http://arxiv.org/abs/1811.03378.

[78] Nitish Srivastava et al. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». Em: *J. Mach. Learn. Res.* 15.1 (jan. de 2014), pp. 1929–1958. ISSN: 1532-4435.

[79] Kaichao You et al. «Learning Stages: Phenomenon, Root Cause, Mechanism Hypothesis, and Implications». Em: *CoRR* abs/1908.01878 (2019). arXiv: 1908.01878. URL: http://arxiv.org/abs/1908.01878.

[80] Lutz Prechelt. «Early Stopping - But When?» Em: (mar. de 2000). DOI: 10.1007/3-540-49430-8_3.

[81] *EarlyStopping*. https://towardsdatascience.com/a-practical-introduction-to-early-stopping-in-machine-learning-550ac88bc8fd. Accessed: 2020-11-16.

A Documentação de Software

RECONHECIMENTO DE ATIVIDADES HUMANAS POR APRENDIZADO DE MÁQUINA

Version-0.1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 image_processing Namespace Reference

Implementa funções de processamento de imagens.

Functions

- def count_frames (video_cap)

  Conta a quantidade de frames de um video.
- def show_frames (video_cap)

  Mostra o conteúdo de um conjunto de frames ao usuário.
- def stack_frames (frames_list, frames_per_stack)

  Agrupa os frames de um vídeo em listas que contém uma quantidade de "frames_per_stack" de frames.
- def scaling (frames_list, scale)

  Essa função realiza reduz a escala dos frames por um fator determinado.
- def redim_weizmann (weiz_frames_list)

  Essa função redimensiona os frames da base Weizmann, para que possua uma proporção altura/largura = 1.33, igual aos frames da base KTH.
- def video2list (video_cap)

  Essa função converte um objeto de vídeo para uma lista de frames.
- def foreground_extraction (frames_list, lr, thr, hist_len)

  Essa função realiza a extração do plano frontal de um vídeo, e retorna uma lista de imagens correspondendo ao frames do mesmo.
- def grayscale (frames_list)

  Esta função recebe uma lista de frames e retorna uma lista contendo os frames em tons de cinza.
- def compute_gradient (frames_list)

  Esta função recebe uma lista com frames e retorna uma lista contendo os frames com seus gradientes na direção X e Y.
- def optical_flow (frames_list)

  Esta função recebe uma lista com frames empilhados e retorna lista contendo os frames com seus fluxos ópticos na direção X e Y.

3.1.1 Detailed Description

Implementa funções de processamento de imagens.

Author

Daniel Kim & Igor Nakamura

Date

Date: 2020-04-17

3.1.2 Function Documentation

3.1.2.1 count_frames()

```
def image_processing.count_frames (
              video_cap )
```

Conta a quantidade de frames de um video.

Parameters

video_cap	objeto VideoCapture

Returns

num_frames: inteiro contendo a quantidade de frames

Examples

```
count_frames(video_cap)
```

Definition at line 38 of file image_processing.py.

3.1.2.2 show_frames()

```
def image_processing.show_frames (
              video_cap )
```

Mostra o conteúdo de um conjunto de frames ao usuário.

Parameters

video_cap	objeto VideoCapture

Returns

None

Examples

```
>>> show_frames(video_cap)
```

Definition at line 60 of file image_processing.py.

3.1.2.3 stack_frames()

```
def image_processing.stack_frames (
              frames_list,
              frames_per_stack )
```

Agrupa os frames de um vídeo em listas que contém uma quantidade de "frames_per_stack" de frames.

Parameters

frames_list	lista contendo uma sequência de frames
frames_per_stack	quantidade de frames por lista

Returns

stacked_frames_list: lista de listas de imagens, correspondendo aos frames de um vídeo

Examples

```
>>> stack_frames(frames_list, 9)
```

Definition at line 95 of file image_processing.py.

3.1.2.4 scaling()

```
def image_processing.scaling (
              frames_list,
              scale )
```

Essa função realiza reduz a escala dos frames por um fator determinado.

Parameters

frames_list	lista contendo uma sequência de frames
scale	inteiro que indica a escala a ser aplicada no redimensionamento dos frames. Ex: 2, para escalar a imagem para a metade do tamanho original

Returns

updated_list: uma lista de imagens, com os frames na escala especificada

Examples

```
>>> scaling(frames_list, 2)
```

Definition at line 125 of file image_processing.py.

3.1.2.5 redim_weizmann()

```
def image_processing.redim_weizmann (
              weiz_frames_list )
```

Essa função redimensiona os frames da base Weizmann, para que possua uma proporção altura/largura = 1.33, igual aos frames da base KTH.

Parameters

weiz_frames_list	lista de frames de um vídeo da base Weizmann

Returns

weiz_frames_list: uma lista de imagens, com os frames da base Weizmann com formato (135, 180, 3)

Examples

```
>>> redim_weizmann(weiz_frames_list)
```

Definition at line 154 of file image_processing.py.

3.1.2.6 video2list()

```
def image_processing.video2list (
              video_cap )
```

Essa função converte um objeto de vídeo para uma lista de frames.

Parameters

video_cap	sequência de frames de um vídeo

Returns

frames_list: lista contendo os frames do vídeo

Examples

```
>>> video2list(video_cap)
```

Definition at line 178 of file image_processing.py.

3.1.2.7 foreground_extraction()

```
def image_processing.foreground_extraction (
              frames_list,
              lr,
              thr,
              hist_len )
```

Essa função realiza a extração do plano frontal de um vídeo, e retorna uma lista de imagens correspondendo ao frames do mesmo.

Parameters

frames_list	lista contendo uma sequência de frames
lr	um número decimal que representa a taxa de aprendizado do algoritmo de subtração
thr	um número inteiro que representa o limiar para definir a distância máxima ao qual um pixel ainda é considerado como pertencente ao fundo
hist_len	um número inteiro que representa o histórico de frames considerados para o background model

Returns

updated_list: uma lista de imagens, contendo os frames com o plano frontal extraído

Examples

```
>>> foreground_extraction(video_cap, lr = 0.85, thr = 24, hist_len = 15)
```

Definition at line 212 of file image_processing.py.

3.1.2.8 grayscale()

```
def image_processing.grayscale (
              frames_list )
```

Esta função recebe uma lista de frames e retorna uma lista contendo os frames em tons de cinza.

Parameters

frames_list	lista contendo uma sequência de frames

Returns

frames_grayscale: uma lista de imagens, com os frames em escalas de cinza

Examples

```
>>> grayscale(frames_list)
```

Definition at line 261 of file image_processing.py.

3.1.2.9 compute_gradient()

```
def image_processing.compute_gradient (
              frames_list )
```

Esta função recebe uma lista com frames e retorna uma lista contendo os frames com seus gradientes na direção X e Y.

Parameters

frames_list	lista contendo uma sequência de frames

Multiple return:

Parameters

gradient_x_list	uma lista de imagens, contendo o gradiente na direção X dos frames contidos em frames_list
gradient_y_list	uma lista de imagens, contendo o gradiente na direção Y dos frames contidos em frames_list

Definition at line 280 of file image_processing.py.

3.1.2.10 optical_flow()

```
def image_processing.optical_flow (
              frames_list )
```

Esta função recebe uma lista com frames empilhados e retorna lista contendo os frames com seus fluxos ópticos na direção X e Y.

Parameters

frames_list	lista contendo uma sequência de frames

Multiple return:

Parameters

opt_x_frames	uma lista de imagens, contendo o fluxo óptico na direção X, para cada frame
opt_y_frames	uma lista de imagens, contendo o fluxo óptico na direção Y, para cada frame

Definition at line 297 of file image_processing.py.

3.2 utils Namespace Reference

Implementa funções de uso geral.

Functions

- def get_weizmann_filepaths ()
  Obtém um dicionário de dicionários referentes ao dataset Weizmann como descrito em get_filepaths.
- def get_kth_filepaths ()
  Obtém um dicionário de dicionários referentes ao dataset KTH como descrito em get_filepaths.
- def create_names_list (path, action)
  Obtém a lista dos nomes das pessoas que executam as ações de um dataset específico.
- def create_actions_list (path, person)
  Obtém a lista de ações referentes a um dataset específico.
- def get_filepaths (path, names_list, actions_list)
  Obtém os caminhos para os arquivos de video de um dataset específico.
- def create_actions_regex_dict (name, actions_list)
  Cria um dicionário de regex para buscar os nomes dos arquivos desejados.
- def get_person_filepaths (path, actions_regex_dict)
  Obtém os caminhos dos arquivos a partir do regex_dict recebido.
- def load_dataset (dataset)
  Lê os arquivos txt relacionados aos conjuntos de dados de treino e de teste e os retorna em uma namedtuple.
- def make_dataframe_from_filepath (txt_filepath, column_header)
  Monta um objeto pandas.DataFrame a partir de um arquivo txt.
- def make_train_test_sets (dataset, filepaths, test_people)
  Escreve arquivos txt contendo os nomes dos videos e suas respectivas classes.

3.2.1 Detailed Description

Implementa funções de uso geral.

Author

Daniel Kim & Igor Nakamura

Date

Date: 2020-04-17

3.2.2 Function Documentation

3.2.2.1 get_weizmann_filepaths()

```
def utils.get_weizmann_filepaths ( )
```

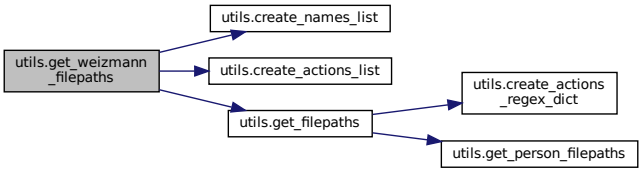
Obtém um dicionário de dicionários referentes ao dataset Weizmann como descrito em get_filepaths.

Returns

Um dicionário de dicionários

Definition at line 30 of file utils.py.

Here is the call graph for this function:



3.2.2.2 get_kth_filepaths()

```
def utils.get_kth_filepaths ( )
```

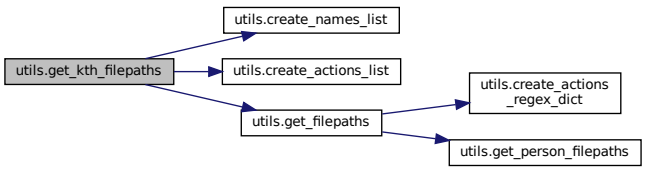
Obtém um dicionário de dicionários referentes ao dataset KTH como descrito em get_filepaths.

Returns

Um dicionário de dicionários

Definition at line 43 of file utils.py.

Here is the call graph for this function:



3.2.2.3 create_names_list()

```
def utils.create_names_list (
              path,
              action )
```

Obtém a lista dos nomes das pessoas que executam as ações de um dataset específico.

Parameters

path	caminho para para a pasta desejada. Ex: 'Weizmann/' ou 'KTH/'
action	nome da ação. Ex: 'bend' ou 'handclapping_d4_uncomp'

Returns

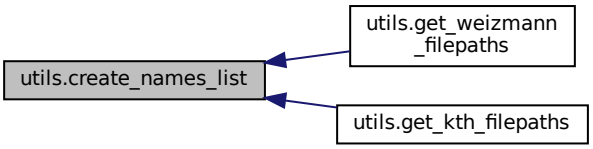
names_list: lista dos nomes das pessoas de um dataset. Como todas as pessoas executam o mesmo conjunto de ações, essa variável contêm todos os nomes das pessoas do dataset.

Examples

```
>>> names_list = ['daria', 'shahar', 'lena', 'lyova', etc]
>>> names_list = ['person01', 'person02', 'person03', etc]
```

Definition at line 68 of file utils.py.

Here is the caller graph for this function:



3.2.2.4 create_actions_list()

```
def utils.create_actions_list (
              path,
              person )
```

Obtém a lista de ações referentes a um dataset específico.

Parameters

path	caminho para para a pasta desejada. Ex: 'Weizmann/' ou 'KTH/'
person	pessoa que executa a ação. Ex: 'daria' ou 'person01'

Returns

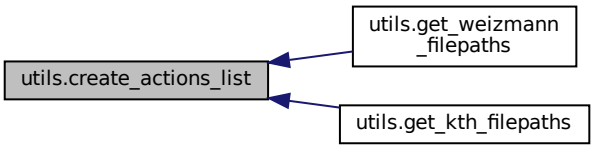
actions_list: lista de ações referentes a uma pessoa do dataset. Como uma única pessoa executa todas as ações do dataset, essa variável contêm todas as ações do dataset.

Examples

```
>>> actions_list = ['bend', 'wave1', 'wave2', 'walk', 'skip', etc]
>>> actions_list = ['handclapping_d4_uncomp', 'boxing_d3_uncomp', etc]
```

Definition at line 91 of file utils.py.

Here is the caller graph for this function:



3.2.2.5 get_filepaths()

```
def utils.get_filepaths (
              path,
              names_list,
              actions_list )
```

Obtém os caminhos para os arquivos de video de um dataset específico.

Parameters

path	caminho para para a pasta desejada. Ex: 'Weizmann/' ou 'KTH/'
names_list	lista com os nomes das pessoas que executam a ação. Ex: 'daria' para Weizmann e 'person01'para KTH
actions_list	lista com as ações. Ex: 'bend', 'walk', etc para Weizmann e 'walking_d1_uncomp', 'person23_boxing_d2_uncomp' para KTH

Returns

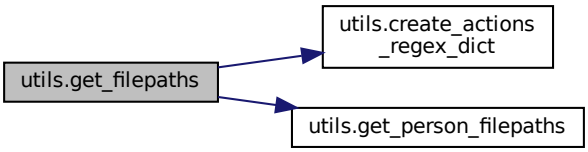
filepaths: dicionário de dicionários cujos 'keys' do dicionário mais externo são os nomes das pessoas. Os 'keys' dos dicionários mais internos são as ações associadas a uma pessoa específica. Por fim, os 'values' do dicionário são o caminho completo para o arquivo de vídeo.

Examples

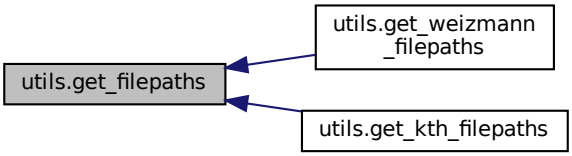
```
>>> print(filepaths['daria']['bend'])
'Weizmann/daria_bend.avi'
>>> print(filepaths['person01']['boxing_d1_uncomp'])
'KTH/person01_boxing_d1_uncomp'
```

Definition at line 120 of file utils.py.

Here is the call graph for this function:



Here is the caller graph for this function:



3.2.2.6 create_actions_regex_dict()

```
def utils.create_actions_regex_dict (
              name,
              actions_list )
```

Cria um dicionário de regex para buscar os nomes dos arquivos desejados.

Parameters

name	nome de uma pessoa. Ex: 'daria' ou 'person01'
actions_list	lista de ações. Já descrito na função get_filepaths

Returns

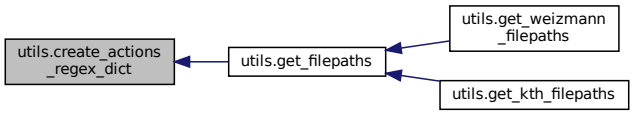
actions_regex_dict: dicionário cujos 'keys' são as ações contidas em actions_list e cujos 'values' são regex que serão usados para procurar o nome do arquivo de vídeo

Examples

```
>>> print(actions_regex_dict['bend'])
r'daria_bend*'
>>> print(actions_regex_dict['running_d3_uncomp'])
r'person25_running_d3_uncomp*'
```

Definition at line 149 of file utils.py.

Here is the caller graph for this function:



3.2.2.7 get_person_filepaths()

```
def utils.get_person_filepaths (
              path,
              actions_regex_dict )
```

Obtém os caminhos dos arquivos a partir do regex_dict recebido.

Parameters

path	caminho para para a pasta desejada. Ex: 'Weizmann/' ou 'KTH/'
actions_regex_dict	idém ao descrito em 'create_actions_regex_dict'

Returns

filepaths: dicionário cujos 'keys' são as ações e cujos 'values' são os caminhos dos arquivos de vídeo contendo as ações de uma pessoa específica

Examples

```
>>> filepaths['bend'] = 'Weizmann/daria_bend.avi'
>>> filepaths['boxing_d1_uncomp'] = 'KTH/person01_boxing_d1_uncomp'
```

Definition at line 171 of file utils.py.

Here is the caller graph for this function:

3.2.2.8 load_dataset()

```
def utils.load_dataset (
              dataset )
```

Lê os arquivos txt relacionados aos conjuntos de dados de treino e de teste e os retorna em uma namedtuple.

Parameters

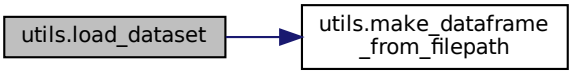
dataset	nome do dataset. Ex: 'weizmann' ou 'kth'

Returns

Uma namedtuple cujos campos são X_train, Y_train, X_test e Y_test. Os valores dos campos iniciados com X e com Y são, respectivamente, dataframes contendo os caminhos para os arquivos de vídeo e as suas respectivas classes.

Definition at line 190 of file utils.py.

Here is the call graph for this function:



3.2.2.9 make_dataframe_from_filepath()

```
def utils.make_dataframe_from_filepath (
              txt_filepath,
              column_header )
```

Monta um objeto pandas.DataFrame a partir de um arquivo txt.

Parameters

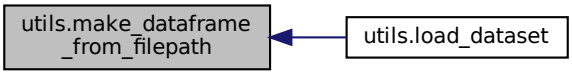
txt_file	caminho para o arquivo txt
column_header	cabeçalho da coluna do dataframe

Returns

Um objeto pandas.DataFrame com os dados do arquivo txt e cabeçalho segundo o argumento column_header

Definition at line 211 of file utils.py.

Here is the caller graph for this function:



3.2.2.10 make_train_test_sets()

```
def utils.make_train_test_sets (
              dataset,
              filepaths,
              test_people )
```

Escreve arquivos txt contendo os nomes dos videos e suas respectivas classes.

Parameters

dataset	nome do dataset. Ex: 'weizmann' ou 'kth'
filepaths	dicionário no formato daquele retornado pelo método get_filepaths
test_people	lista de strings contendo os nomes das pessoas que farão parte do conjunto de testes. Ex: ['denis'], ['person10', 'person11']

Definition at line 230 of file utils.py.

Chapter 4

File Documentation

4.1 src/image_processing/image_processing.py File Reference

Namespaces

- image_processing

  Implementa funções de processamento de imagens.

Functions

- def image_processing.count_frames (video_cap)

  Conta a quantidade de frames de um video.
- def image_processing.show_frames (video_cap)

  Mostra o conteúdo de um conjunto de frames ao usuário.
- def image_processing.stack_frames (frames_list, frames_per_stack)

  Agrupa os frames de um vídeo em listas que contém uma quantidade de "frames_per_stack" de frames.
- def image_processing.scaling (frames_list, scale)

  Essa função realiza reduz a escala dos frames por um fator determinado.
- def image_processing.redim_weizmann (weiz_frames_list)

  Essa função redimensiona os frames da base Weizmann, para que possua uma proporção altura/largura = 1.33, igual aos frames da base KTH.
- def image_processing.video2list (video_cap)

  Essa função converte um objeto de vídeo para uma lista de frames.
- def image_processing.foreground_extraction (frames_list, lr, thr, hist_len)

  Essa função realiza a extração do plano frontal de um vídeo, e retorna uma lista de imagens correspondendo ao frames do mesmo.
- def image_processing.grayscale (frames_list)

  Esta função recebe uma lista de frames e retorna uma lista contendo os frames em tons de cinza.
- def image_processing.compute_gradient (frames_list)

  Esta função recebe uma lista com frames e retorna uma lista contendo os frames com seus gradientes na direção X e Y.
- def image_processing.optical_flow (frames_list)

  Esta função recebe uma lista com frames empilhados e retorna lista contendo os frames com seus fluxos ópticos na direção X e Y.

4.2 src/utils/utils.py File Reference

Namespaces

- utils

  Implementa funções de uso geral.

Functions

- def utils.get_weizmann_filepaths ()

  Obtém um dicionário de dicionários referentes ao dataset Weizmann como descrito em get_filepaths.
- def utils.get_kth_filepaths ()

  Obtém um dicionário de dicionários referentes ao dataset KTH como descrito em get_filepaths.
- def utils.create_names_list (path, action)

  Obtém a lista dos nomes das pessoas que executam as ações de um dataset específico.
- def utils.create_actions_list (path, person)

  Obtém a lista de ações referentes a um dataset específico.
- def utils.get_filepaths (path, names_list, actions_list)

  Obtém os caminhos para os arquivos de video de um dataset específico.
- def utils.create_actions_regex_dict (name, actions_list)

  Cria um dicionário de regex para buscar os nomes dos arquivos desejados.
- def utils.get_person_filepaths (path, actions_regex_dict)

  Obtém os caminhos dos arquivos a partir do regex_dict recebido.
- def utils.load_dataset (dataset)

  Lê os arquivos txt relacionados aos conjuntos de dados de treino e de teste e os retorna em uma namedtuple.
- def utils.make_dataframe_from_filepath (txt_filepath, column_header)

  Monta um objeto pandas.DataFrame a partir de um arquivo txt.
- def utils.make_train_test_sets (dataset, filepaths, test_people)

  Escreve arquivos txt contendo os nomes dos videos e suas respectivas classes.

Index