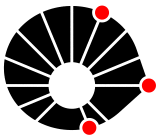

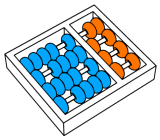

SwAV-HAR: Adaptação do modelo SwAV para reconhecimento de atividades humanas

B. P. Moura *E. Borin*

Relatório Técnico - IC-PFG-26-02
Projeto Final de Graduação
2026 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

The contents of this report are the sole responsibility of the authors.
O conteúdo deste relatório é de única responsabilidade dos autores.

Agradecimentos

Primeiramente, gostaria de agradecer à minha família. Aos meus pais, Maria Estela e João, que acreditaram na realização desta jornada e me ensinaram o valor dos estudos, do comprometimento e do esforço por meio do exemplo de dedicação em suas carreiras no serviço público do Estado de São Paulo. Além disso, agradeço pelo apoio incondicional e pela força, que me permitiram chegar até aqui.

Agradeço ao meu irmão Ricardo e aos meus tios, especialmente Edson, Lucimar e Regina Helena *(in memoriam)*, que contribuíram na minha formação pessoal para além da graduação, sempre com companheirismo. Ao meu parceiro de vida, Rodrigo, agradeço por me acompanhar em diferentes momentos e por ser conselheiro nas fases difíceis.

Ao meu orientador de projeto e de iniciação científica, Edson Borin, e a todos os professores associados, doutorandos, mestrandos e colegas do grupo de pesquisa de Representação de Conhecimento do H.IAAC, o meu sincero agradecimento. Com todos vocês, aprofundei meus conhecimentos em inteligência artificial, conheci os desafios da carreira acadêmica e dei meus primeiros passos na pesquisa científica e tecnológica.

SwAV-HAR: Adaptação do modelo SwAV para reconhecimento de atividades humanas

Beatriz Vitória Pereira Moura* Edson Borin

Resumo

Este relatório apresenta o SwAV-HAR, uma adaptação do modelo autossupervisionado SwAV [1] para o domínio de séries temporais de sensores inerciais, desenvolvido no escopo do projeto final de graduação em Engenharia de Computação pela Universidade de Campinas (UNICAMP). O modelo é voltado ao reconhecimento de atividades humanas (HAR) e à identificação de transições posturais entre atividades nos conjuntos de dados HAPT [2] e DAGHAR [3]. A adaptação utiliza o *backbone* ResNet-SE-5 e introduz transformações de aumento baseadas em *temporal multi-crop*. O SwAV-HAR é avaliado sob os protocolos *freeze*, *full fine-tuning* e *from scratch*, com acurácia de classificação média de 88% no cenário *in-domain*, tornando-se competitivo em relação ao treinamento supervisionado e outras técnicas do estado da arte de *Self-Supervised Learning (SSL)*, destacando-se principalmente em regimes de poucas amostras rotuladas.

Além disso, o projeto propõe a detecção de janelas de transição por meio de *scores* derivados dos protótipos aprendidos, mensurados por meio da distância ao protótipo e da taxa de transição das atribuições de protótipos, avaliados no conjunto de dados HAPT. Os resultados indicam que a métrica de taxa de transição impõe a maior diferenciação entre as classes estáveis e de transição postural, que apresenta a dificuldade de separação geométrica devido à pequena quantidade de amostras disponíveis para o aprendizado do modelo. Finalmente, as direções futuras do projeto envolvem a extensão dos protocolos de coleta e de avaliação para outros conjuntos de dados, além da inclusão de métodos que preservem a temporalidade, promovendo maior capacidade de aprendizado na presença de amostras de fronteira ou janelas de transição.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Pesquisa desenvolvida com suporte financeiro parcial da FUNCAMP. H.IAAC - Representação de Conhecimento.

Sumário

1 Introdução e Objetivos

A área de reconhecimento de atividades humanas, ou HAR[1], baseada em sensores é essencial para o desenvolvimento de estudos e aplicações em ramos como saúde, manufatura, esporte e automação residencial. Dispositivos equipados com acelerômetros e giroscópios viabilizam a aquisição contínua de dados de séries temporais associadas a padrões de movimento humano, constituindo a principal fonte de dados para sistemas de HAR embarcados.

Entretanto, o desenvolvimento de modelos de HAR ainda enfrenta desafios, como a dificuldade de extrair características devido à similaridade entre diferentes atividades, a discrepância de distribuição observada diante de variações entre dispositivos e o alto custo da anotação manual de dados, que limita a escalabilidade de conjuntos rotulados, além de introduzir inconsistências decorrentes da subjetividade do processo de anotação [4]. Nesse contexto, uma lacuna pouco explorada é a detecção de atividades de transição, que são desconsideradas na maioria dos conjuntos de dados públicos e, por isso, pouco acessíveis em abordagens supervisionadas convencionais [5].

Este projeto final de graduação propõe a implementação e avaliação do modelo SwAV-HAR, uma adaptação do método autossupervisionado SwAV [1] para séries temporais de sensores inerciais, investigando as seguintes perguntas de pesquisa:

- Modelos de aprendizado autossupervisionado adaptados para HAR e baseados em agrupamento são capazes de aprender representações latentes que, quando submetidas a um classificador, alcançam acurácia comparável à obtida por modelos supervisionados?
- A adoção de pré-treinamento autossupervisionado com a técnica SwAV reduz a dependência de dados rotulados em HAR, mantendo desempenho competitivo em relação a modelos supervisionados em cenários de escassez de rótulos?
- Técnicas de agrupamento aplicadas às representações aprendidas durante o pré-treino capturam estrutura latente suficiente para detectar atividades humanas de transição?

Para responder a essas questões, este relatório apresentará análises exploratórias, protocolos de avaliação e discussões acerca da arquitetura SwAV-HAR aplicada aos conjuntos de dados do *benchmark* DAGHAR [3] e ao conjunto de dados de atividades de transição HAPT [2].

[1] Do inglês: *Human Activity Recognition*.

2 Fundamentação Teórica

2.1 Reconhecimento de atividades humanas (HAR)

Com o avanço tecnológico promovido pela consolidação da Internet e de sistemas distribuídos, as interações com as interfaces homem-máquina estimularam a disseminação de dispositivos e sensores, que por sua vez impulsionaram a era da Internet das Coisas (IoT). Nesse aspecto, esses dispositivos equipados com sensores e as plataformas de armazenamento e de distribuição de dados possibilitaram as aplicações de monitoramento de atividades cotidianas da população global por meio de dispositivos móveis e vestíveis. Assim, os trabalhos atuais de revisão sobre HAR, como os de Chen *et al.* [6] e de Plötz *et al.* [4], definem o **reconhecimento de atividades humanas (HAR)** como um processo de identificação das atividades realizadas pelo indivíduo, tais como sentar, levantar, andar e correr. Entre as tarefas de análise mais abordadas estão a classificação e a detecção de anomalias, com aplicações no auxílio às atividades de vida diária.

2.1.1 Atividades de transição

Para as coletas de dados de HAR, as **atividades de transição** são definidas por Reyes-Ortiz *et al.* [5] como eventos de curta duração, ocorridos no intervalo entre atividades mais longas. Alguns exemplos para este tipo de atividade são "sentado-para-em pé" e "sentado-para-deitado", encontrados entre as 6 classes de transição do conjunto de dados HAPT [2]. Essa subclasse de atividades é frequentemente tratada como ruído nas séries temporais. No entanto, devido à natureza instável do sinal, as atividades de transição podem causar erros de predição, segundo Reyes-Ortiz *et al.* [5]. As arquiteturas voltadas para esse tipo de atividade são pouco exploradas em trabalhos de HAR, assim essa subclasse de atividades pode influenciar o desempenho de modelos. Nesse contexto, o modelo *Transition-Aware HAR* (TAHAR) [5] está entre as poucas opções desenvolvidas para suavizar flutuações e identificar atividades de transição.

2.1.2 Desafios na aquisição de dados de HAR

Além das classes de atividades, a extração de dados de sensores inerciais, como acelerômetro e giroscópio, enfrenta os desafios de restrições de recursos de *hardware* e *software* dos dispositivos, além da divergência de condições de coleta, como o posicionamento dos dispositivos, da especificação de sensores e da influência das características dos dados, tais como variações entre usuários e o desbalanceamento de classes. Esses fatores influenciam a qualidade dos dados coletados, dificultam a compreensão da complexidade das atividades humanas e a generalização dos modelos de inteligência artificial (IA) em cenários de menor controle ambiental.

Por consequência, a curadoria de séries temporais de HAR é muito custosa devido à necessidade de cobrir ampla variabilidade de atividades, definir protocolos e, posteriormente, rotular manualmente grandes volumes de dados, como abordado por Plötz *et al.* [4]. Ademais, os dispositivos utilizados para aquisição e monitoramento são produzidos com

enfoque na portabilidade, portanto são compactos, limitados em memória, bateria e poder de processamento.

No âmbito deste projeto, alguns desses fatores, como a divergência entre protocolos de coleta e a natureza dos dados selecionados, influenciaram desde a seleção e tratamento dos dados até as análises comparativas de desempenho do modelo SwAV-HAR. Assim, os conjuntos de dados que foram coletados com especificações e protocolos diferenciados, como HAPT, foram submetidos à etapa de pré-processamento visando à uniformidade da coleção de dados. Por outro lado, a natureza sensorial das informações, por exemplo, os sinais de baixa energia, constitui um fator relevante no domínio de aprendizado de representações. Esse aspecto é explorado na seção de resultados, na qual se discute em detalhes como as características das amostras de séries temporais impactam o desempenho da arquitetura de classificação de atividades.

2.2 Aprendizado autossupervisionado para HAR

Considerando os desafios iniciais no reconhecimento de atividades humanas, a partir da segunda década dos anos 2000, o desenvolvimento das técnicas de aprendizado autossupervisionado promoveu uma transformação nas arquiteturas de aprendizado profundo destinadas ao reconhecimento de atividades humanas, buscando contornar parte das limitações encontradas e gerando um contraponto às técnicas clássicas de aprendizado de máquina, como discutido por Haresamudram *et al.* [7].

Durante essa transição da aplicação dos modelos treinados *end-to-end* em HAR para o estado da arte em *Self-Supervised Learning*, a definição de HAR passou a ser organizada em etapas de coleta, pré-processamento, segmentação, extração de características e classificação [7]. Nesse âmbito, o **aprendizado autossupervisionado** baseia-se no paradigma de pré-treino do modelo para extrair representações num espaço latente compreensível e, em seguida, transferir tais representações para o ajuste de diferentes tarefas-alvo, práticas denominadas *pretrain-then-finetune* e *transfer learning*.

2.3 *Deep Clustering*

A partir do aprendizado de representações e da transferência de aprendizado, surge um mecanismo de aprendizado em SSL, fundamentado na otimização conjunta da extração de características e do agrupamento de atividades, consolidando o campo de ***Deep Clustering***. De acordo com Chen *et al.* [6], o agrupamento profundo é empregado principalmente para lidar com a escassez de dados rotulados em HAR, unindo o aprendizado de representações com o caráter não-supervisionado dos algoritmos de agrupamento em modelos como o *DeepCluster* [8]. Sendo assim, a integração dos objetivos de agrupamento em redes neurais com funções de perda definidas no escopo de SSL permitiu a exploração de espaços de características com maior separabilidade entre as atividades e maior correspondência semântica em relação às atividades humanas reais.

3 Trabalhos Relacionados

Para o desenvolvimento deste projeto, o estado da arte foi investigado nas áreas destacadas na seção de fundamentação teórica. Assim, a revisão bibliográfica foi conduzida a partir da busca de palavras-chave em um conjunto de bases de dados diversificadas, tais como *Google Scholar*, *IEEE Xplore* e *arXiv*. Os trabalhos foram organizados por eixos temáticos e métricas como h5-index e número de citações foram considerados para a seleção. A Figura 1 apresenta o gráfico com os resultados encontrados a partir de 2020 para um conjunto de palavras-chave inseridas no *Google Scholar*.

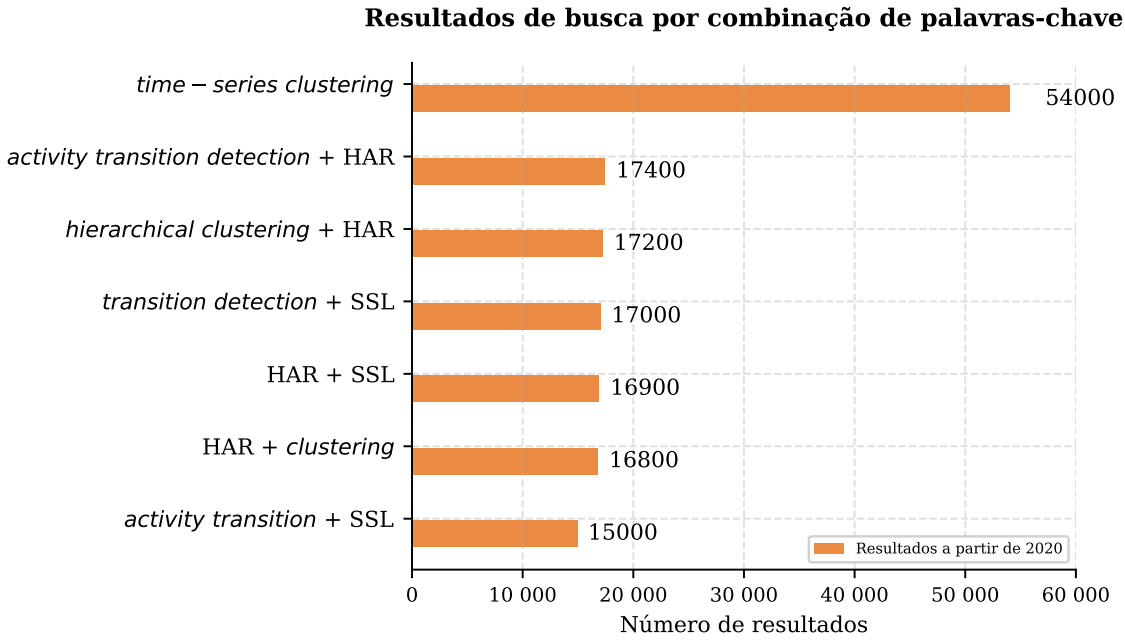

Figura 1: Resultados de busca para palavras-chave no *Google Scholar*.

Nesse aspecto, o gráfico evidencia uma tendência de maior concentração de resultados para termos mais amplos, como *time-series clustering*, ao passo que as combinações mais específicas produziram quantidades menores de correspondências.

A partir desta triagem, os trabalhos relacionados ao SwAV-HAR foram organizados em três eixos temáticos: agrupamento não-supervisionado para HAR, métodos contrastivos e baseados em agrupamento, e identificação de atividades de transição. Cada um dos eixos representa uma camada de motivação para o modelo proposto.

3.1 Agrupamento não-supervisionado para HAR

No campo das técnicas de agrupamento para HAR, houve um avanço no aspecto das representações utilizadas, visto que métodos tradicionais, aplicados diretamente ao espaço de entrada, evoluíram para arquiteturas de aprendizado de representações latentes otimizadas

para a tarefa de agrupamento. Nesse contexto, *Deep Clustering* se constituiu como estado da arte na tarefa de descoberta de padrões, principalmente para grandes volumes de dados não rotulados, uma mudança consolidada pela introdução de estruturas profundas capazes de extrair automaticamente *features* discriminativas dos sinais de HAR, a exemplo do artigo de Abedin *et al.* [9]. Entre esses métodos, a arquitetura DISC *(Deep Inertial Sensory Clustering)* [10] aprimorou a utilização de *autoencoders* recorrentes com a adição de camadas de redes neurais convolucionais, ou CNNs[2], e um tipo particular de redes neurais recorrentes, as unidades recorrentes com porta, ou GRUs[3], obtendo representações mais compactas e discriminativas dos sinais inerciais.

Contudo, o trabalho de Abedin *et al.* [9] aponta que arquiteturas como o DISC ainda podem depender de algoritmos tradicionais de agrupamento aplicados após o treinamento, ou de penalidades explícitas de separação de grupos (*clusters*) durante o processo de otimização. Essa dependência demonstra uma limitação estrutural, visto que o agrupamento e o aprendizado de representações não ocorrem de forma conjunta de fato, o que motiva a investigação de métodos que integrem ambos os objetivos no processo de treinamento.

3.2 Métodos contrastivos e métodos baseados em protótipos

A integração entre aprendizado de representações e agrupamento foi explorada por métodos contrastivos de instâncias e métodos baseados em pseudo-rótulos de agrupamento. No campo do aprendizado contrastivo, SimCLR [11] estabeleceu um marco com a discriminação de instâncias de imagens voltado ao aprendizado de representações.

Todavia, trata-se de um modelo que, ao abdicar dos bancos de memória, exige lotes massivos para manter um número suficiente de pares negativos, tornando-se custoso computacionalmente. Já no domínio de HAR, o TF-C *(Time-Frequency Consistency)* [12] adaptou o paradigma contrastivo para séries temporais com a construção de pares positivos consistentes no domínio do tempo e da frequência simultaneamente. Por isso, TF-C se destaca pelo desempenho em regime de poucos rótulos quando combinado com extratores de características no *benchmark* de da Luz *et al.* [13], que sistematiza a avaliação de técnicas de SSL para HAR em conjuntos de dados do *benchmark* DAGHAR [3], adotados neste trabalho.

Paralelamente, o modelo DeepCluster [8] foi pioneiro ao propor a substituição de rótulos reais por atribuições de *K-Means* sobre os *embeddings* da rede durante o treinamento. No entanto, o método exige múltiplas passagens sobre o conjunto de dados para atualização dos centroides, etapa corrigida no DeepCluster-v2 com a substituição da camada de classificação pela comparação explícita dos *embeddings*. Além disso, DeepCluster-v2 ainda incorpora um MLP e técnicas de treinamento de aumento dos dados e *multi-clustering* mais robustas, porém com operação ainda *offline*.

Considerando as dificuldades para obter resultados de alto desempenho com as técnicas *offline*, o modelo SwAV *(Swapping Assignments between multiple Views)* [1] resolve parte das limitações ao formular o problema da predição trocada *(swapped prediction)* entre visões aumentadas dentro do lote associado à comparação de atribuições a protótipos aprendidos.

[2]Do inglês: *Convolutional Neural Networks*.
[3]Do inglês: *Gated Recurrent Units*.

Concretamente, dadas duas visões globais derivadas de uma imagem, a representação de uma visão é utilizada para predizer o código da outra. A formulação deste problema preserva a invariância da representação a diferentes aumentações da mesma amostra, um dos objetivos de métodos contrastivos. No entanto, SwAV não aplica o mecanismo de distinção par a par entre instâncias, mas indiretamente propõe a comparação de códigos de agrupamento. Nesse aspecto, SwAV é contrastivo quanto ao objetivo, a invariância a aumentações, mas não quanto ao mecanismo, que se apoia na consistência entre os códigos gerados. Desse modo, o processo de agrupamento e o aprendizado de características tornam-se concomitantes e *online*.

Consequentemente, a arquitetura SwAV trouxe aumento no desempenho de tarefas de classificação de imagens com resultados comparáveis ao DeepCluster-v2 e custo computacional significativamente menor. Portanto, mecanismos de atribuição de protótipos com contraste de códigos para o domínio de HAR, estendidos para a natureza de atividades de transições posturais, surgem como um caminho de pesquisa em aberto. Por essa razão, a adaptação do modelo SwAV é potencialmente competitiva com os modelos contrastivos de SSL já aplicados ao domínio de sinais temporais inerciais.

3.3 Identificação de atividades de transição

Conforme discutido na fundamentação teórica, as atividades de transição são pouco exploradas pelos modelos de aprendizado, e isso é refletido nos resultados apresentados para a última combinação de palavras-chave na Figura 1. Nesse escopo, a arquitetura TAHAR [5] utiliza as probabilidades de máquinas de vetores de suporte (SVMs) e filtro de suavização temporal para tratar as transições como atividades desconhecidas.

Embora eficaz no cenário supervisionado, a abordagem ressalta que técnicas de agrupamento não-supervisionado ainda carecem de mecanismos específicos para separação dessa classe de atividades, ponto também reforçado por Amrani *et al.* [10] na arquitetura DISC. Esses autores argumentam que modelos de *Deep Clustering* tradicionais falham no domínio sensorial, visto que desconsideram a natureza sequencial dos sinais, levando à união de atividades transitórias e convencionais num único grupo. Trata-se então de uma limitação que persiste tanto no DISC quanto no método SwAV, citado anteriormente. Por isso, a inclusão e análise de modelos de agrupamento não-supervisionado que lidam com conjuntos de dados de HAR com transições podem ampliar a exploração das atividades humanas de maneira geral.

A Tabela 1 apresenta uma síntese de cobertura temática dos trabalhos discutidos, a fim de posicionar o modelo SwAV-HAR no escopo de trabalhos que adaptam o poder de generalização dos protótipos treináveis para as atividades humanas convencionais e de transição.

Tabela 1: Síntese de cobertura temática dos trabalhos relacionados.

Trabalho	Agrup. não-supervisionado	Representação latente	SSL / Contrastivo	Pseudo-rótulos	Protótipos treináveis	Agrup. *online*	Transições
DISC	Sim	Sim	–	–	–	–	–
SimCLR	–	Sim	Sim	–	–	–	–
TF-C	–	Sim	Sim	–	–	–	–
DeepCluster	Sim	Sim	–	Sim	–	–	–
DeepCluster-v2	Sim	Sim	–	Sim	–	–	–
SwAV	Sim	Sim	–	Sim	Sim	Sim	–
TAHAR	–	–	–	–	–	–	Sim
SwAV-HAR	Sim	Sim	–	Sim	Sim	Sim	Sim

4 Metodologia

As subseções a seguir descrevem a organização do projeto desde a seleção e pré-processamento dos conjuntos de dados até a definição do *encoder* e dos métodos de agrupamento, passando pela descrição e o detalhamento das arquiteturas SwAV e SwAV-HAR.

4.1 Seleção de conjuntos de dados de HAR

A seleção dos dados utilizados para treinamento, teste e avaliação foi realizada considerando a organização dos tipos de classes de atividades existentes para HAR. Desse modo, optou-se pela utilização da coleção curada de seis conjuntos de dados públicos reportada no *benchmark* DAGHAR [3] e pelo conjunto de dados público com classes de atividades de transição HAPT [2]. A coleção de bases de dados do DAGHAR é composta pelos seguintes conjuntos: Ku-HAR, MotionSense, RealWorld Waist, RealWorld Thigh, UCI HAR e WISDM.

Esses dados de séries temporais foram coletados a partir de sensores de acelerômetro e giroscópio nos três eixos (x, y, z) presentes em *smartphones* posicionados em diferentes partes do corpo, como coxa e cintura. As classes de atividades reportadas na coleção DAGHAR são sentado, em pé, subir e descer escadas, caminhar e correr. Assim, as informações foram padronizadas no *benchmark* com unidades de aceleração $\mathrm{m/s^2}$, taxa de amostragem 20 Hz, segmentação em janelas fixas de 3 s sem sobreposição e remoção do componente de gravidade. Além disso, DAGHAR [3] adota o particionamento dos dados ao nível de usuário, utilizando a razão aproximada 70/20/10 para treino, validação e teste observada na Figura 22 do apêndice de estudos de ablação.

Já para incluir atividades de transição, a base de dados HAPT *(Smartphone-Based Recognition of Human Activities and Postural Transitions)* [2], proposta por Reyes-Ortiz *et al.*, consiste numa extensão do UCI-HAR que acrescenta as atividades de transição postural ao conjunto de rótulos. Para o HAPT, foram reportadas 12 atividades, 6 básicas e 6 de transição, para 30 participantes. As seis atividades básicas são: caminhar, subir e descer escadas, sentado, em pé e deitado, enquanto as transições incluem os 6 pares posturais entre as atividades estáticas: de em pé para sentado, de sentado para em pé, de sentado para deitado, de deitado para sentado, de em pé para deitado e de deitado para em pé.

Segundo Reyes-Ortiz *et al.*, as coletas do HAPT foram realizadas com sensores de acelerômetro e giroscópio de *smartphone* posicionados na cintura, com frequência de amostra-

gem de 50 Hz. Por isso, a base de dados foi pré-processada neste projeto para adequá-la ao formato DAGHAR, através de 4 etapas: (1) remoção da componente gravitacional dos canais de acelerômetro por meio de um filtro Butterworth passa-alta; (2) reamostragem de 50 Hz para 20 Hz; (3) segmentação por janela fixa de 60 *timesteps*, equivalente a 3 s a 20 Hz, com o rótulo de cada janela determinado pela moda das atividades presentes no intervalo; e (4) mapeamento das 6 classes de transição numa única classe (6), resultando em 7 classes no total, com a proporção 70/20/10 para os subconjuntos de treino, validação e teste, respectivamente.

A Figura 2 mostra a composição das classes no conjunto de treino. O Apêndice A.4 apresenta figuras extras com as composições de classes para os conjuntos de validação e de teste, além da análise do total de amostras por conjunto de dados.

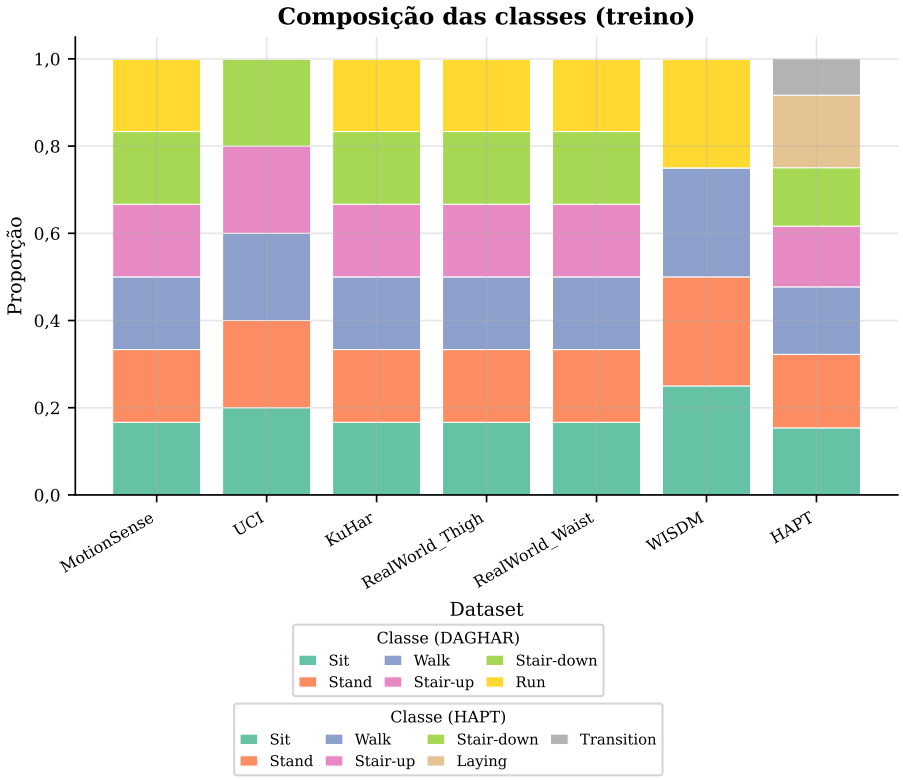

Figura 2: Composição das classes por conjunto de dados (treino).

4.2 Seleção de arquitetura *encoder* (*backbone*)

Como detalhado na fundamentação teórica, técnicas de aprendizado autossupervisionado adotam a etapa de pré-treinamento para extração de características latentes. Nesse sentido, a arquitetura escolhida para o *encoder* é essencial para capturar padrões sensoriais em HAR. Por isso, a seleção do *backbone* para SwAV-HAR foi baseada nos experimentos e resultados do *benchmark* de *encoders* reportados por da Luz *et al.* [13], que avalia múltiplas

arquiteturas nos conjuntos de dados DAGHAR. Considerou-se ainda a arquitetura utilizada no modelo SwAV [1] para imagens, o *backbone* ResNet-50.

Com base na exploração dos resultados, a arquitetura ResNet-SE-5 foi selecionada devido à eficácia em regimes de escassez de rótulos em cenários de HAR e à estabilidade em múltiplos conjuntos de dados de DAGHAR no *benchmark*. Adicionalmente, por ser uma adaptação do conceito original de redes residuais (ResNet) para o domínio de sinais temporais 1D com adição de blocos *Squeeze-and-Excitation* (SE), ResNet-SE-5 é um codificador compatível com o *pipeline* do SwAV. A Figura 3 exibe o diagrama que descreve a arquitetura ResNet-SE-5.

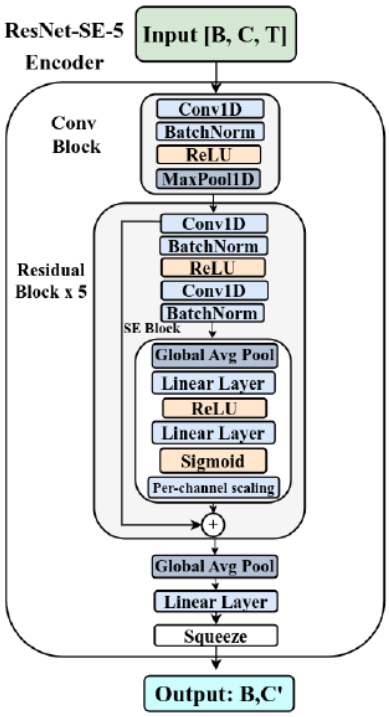

Figura 3: Diagrama para o *backbone* ResNet-SE-5. [13]

4.3 Seleção de método de agrupamento

Já no eixo temático de técnicas de agrupamento, o método *K-Means* foi selecionado com o objetivo de propor uma avaliação externa com um algoritmo simples e comparar com resultados do modelo de aprendizado. Nesse sentido, espera-se que o *K-Means* seja capaz de identificar as classes de atividades a partir da geometria do espaço latente. Então, o algoritmo particiona um conjunto de dados em $K = n_{classes}$ grupos disjuntos, minimizando a soma dos quadrados das distâncias euclidianas de cada ponto ao centroide do seu grupo. Esse processo ocorre de forma iterativa até a convergência, com inicialização dos K centroides pela heurística `K-Means ++` de `scikit-learn`, seguida pela atribuição de cada amostra ao centroide mais próximo e pelos cálculos sucessivos dos centroides com a média

de amostras atribuídas a cada grupo.

4.4 *Swapping Assignments between multiple Views of the same image* (SwAV)

O método autossupervisionado SwAV, proposto por Caron *et al.* [1], desenvolve o aprendizado de *features* visuais sem a necessidade de comparações entre pares, técnica aplicada em outros trabalhos de aprendizado contrastivo, como SimCLR [11]. Nesse sentido, SwAV promove o aprendizado de representações visuais por meio de uma estratégia de transformação de dados, codificação com redes neurais profundas e um método de agrupamento de dados via atribuições probabilísticas.

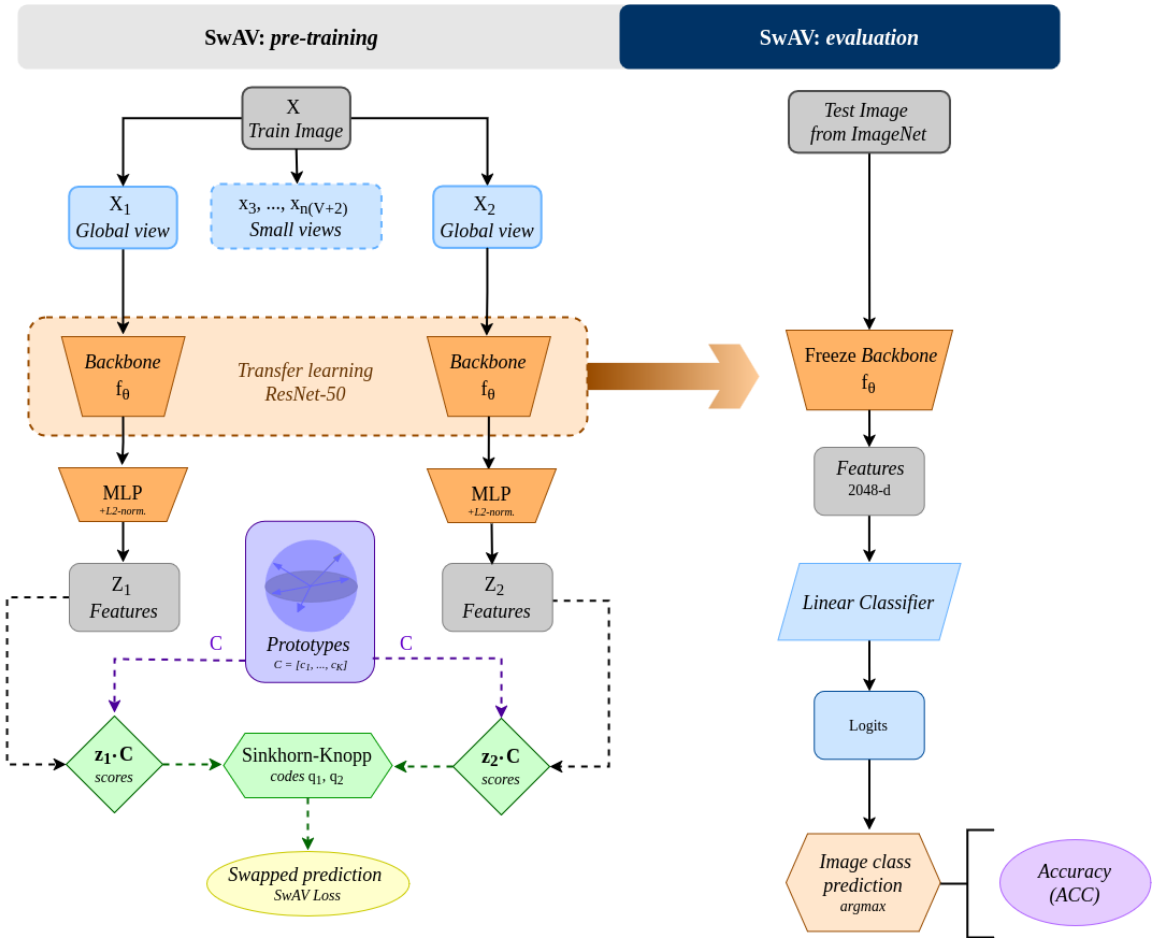

Figura 4: Diagrama da arquitetura SwAV.

A partir dessa premissa, a arquitetura do modelo é organizada em duas fases distintas ilustradas na Figura 4. Durante o pré-treino autossupervisionado, o modelo abrange quatro etapas: transformação das imagens de entrada com o intuito de aumentar o volume de dados e gerar múltiplos recortes da imagem original; extração de *features* representativas

através da codificação de cada recorte da imagem por uma rede neural ResNet-50; projeção dessas *features* num espaço de menor dimensão por meio de uma cabeça de projeção (MLP), seguida de normalização L2 na hiperesfera unitária de 128 dimensões; e agrupamento das características extraídas para um conjunto de tamanho K de protótipos treináveis C, com determinação de códigos de *views* da mesma imagem de maneira trocada, visando a predição da atribuição de grupos para as representações geradas. Após o pré-treino, na fase de avaliação, o *backbone* treinado é congelado e um classificador linear independente é treinado sobre suas representações, com o objetivo de aferir a qualidade das *features* aprendidas. Desse modo, cada etapa do algoritmo SwAV é apresentada nas subseções a seguir.

4.4.1 Transformação de imagens

Inicialmente, o modelo recebe como entrada uma das imagens de um conjunto de dados selecionado, como ImageNet [14]. Assim, SwAV introduz uma estratégia de *multi-crop* para aumentar a variabilidade de imagens e adota um conjunto diversificado de visualizações x_{nt} para cada imagem de um lote. As visualizações são definidas em duas visualizações de alta resolução 224×224 pixels *(global views)* que preservam o contexto global da imagem e V visualizações de baixa resolução 96×96 pixels *(local views)*, que abrangem pequenas partes ou detalhes da imagem original. Desse modo, Caron *et al.* [1] demonstraram que essa escolha aumentou o número de comparações e a precisão do modelo, sem elevar significativamente o custo computacional.

4.4.2 Extração de *features* representativas

Após a geração das *views*, cada uma delas é fornecida como entrada à rede ResNet-50, que promove a conversão para *features* representativas de 2048 dimensões por meio de quatro blocos residuais, seguidos de uma camada de *Average Pooling* e *flatten*. Cada bloco residual aprende padrões de complexidade crescente, tais como detecção de bordas e texturas nas primeiras camadas e representações semânticas de alto nível nos blocos mais profundos. As conexões residuais *(skip connections)* de cada bloco permitem a adição da entrada à saída transformada, o que preserva o fluxo do gradiente. Por último, a camada de *Average Pooling* adaptativa impulsiona o colapso da dimensão espacial, de forma que recortes de diferentes tamanhos produzam vetores de mesma dimensão (2048).

Assim, para cada amostra, o vetor resultante de 2048 dimensões contém as características profundas que a rede foi capaz de identificar na imagem original. Esse espaço latente, denominado espaço do *backbone*, captura representações detalhadas e gerais da entrada.

4.4.3 Cabeça de projeção (MLP) e espaço dos protótipos

Em seguida, as *features* aprendidas são projetadas para o espaço de menor dimensão através da cabeça de projeção (MLP), cuja função é mapear as características para um espaço de 128 dimensões, estrutura observada em outros modelos de aprendizado autossupervisionado, ou SSL[4]. Essa cabeça é formada por duas camadas lineares, sendo a camada oculta de mesma

[4]Do inglês: *Self-Supervised Learning*.

dimensão que a entrada. A primeira camada linear é seguida de uma normalização por lote e de uma ativação ReLU, enquanto a segunda (*Linear* $2048 \rightarrow 128$) é responsável pela compressão final para o espaço de 128 dimensões.

Depois da passagem do MLP, o vetor é ajustado para comprimento unitário via normalização L2, o que ocasiona a projeção na hiperesfera unitária de 128 dimensões, espaço em que também residem os K protótipos $C = \{c_1, ..., c_K\}$. Essa padronização é fundamental para medir a similaridade de cosseno entre os vetores de amostras e os protótipos (centroides de agrupamento), mantendo uma escala comparável e estável.

4.4.4 Agrupamento "online" e geração de códigos q

Com as representações de cada lote normalizadas, o agrupamento é implementado utilizando o algoritmo Sinkhorn-Knopp (Apêndice A.5), desenvolvido por Knopp e Sinkhorn [15]. Este algoritmo foi escolhido pelos autores do SwAV pois é eficiente para solucionar um problema de transporte que visa maximizar a similaridade entre as *features* e os protótipos sob uma restrição de equipartição, ou seja, as imagens do *batch* devem ser distribuídas de forma equilibrada entre os protótipos (centroides), evitando o colapso do modelo em soluções triviais, como a adoção de um mesmo código para todas as imagens.

Por isso, o algoritmo recebe a função de mapear B vetores de *features* $Z = [z_1, ..., z_B]$ aos protótipos $C = [c_1, ..., c_K]$ através de códigos $Q = [q_1, ..., q_B]$. Calcula-se o produto escalar entre os *embeddings* L2-normalizados das *views* e os protótipos disponíveis, resultando em logits de dimensão $[B * n_{views}, K]$. Em seguida, a esses logits são aplicadas a suavização exponencial $\exp(\cdot/\varepsilon)$ e a normalização alternada das linhas e colunas da matriz durante 3 iterações, suficientes para obter convergência rapidamente. Assim, as iterações promovem a distribuição uniforme entre *clusters*, visto que a normalização das linhas garante que cada protótipo receba peso total equivalente entre as amostras, enquanto a normalização de colunas promove a distribuição do peso total da amostra entre os protótipos. Então, o procedimento retorna uma distribuição de probabilidade suavizada q (código), utilizada no problema da predição trocada.

4.4.5 Problema da predição "trocada"

Com base no mapeamento de *features* e protótipos, a diferença principal do SwAV em relação a métodos contrastivos é a utilização do problema da predição trocada. Em vez de comparar vetores de características z de duas visões, SwAV formula a tarefa de predizer o código q de uma visão a partir da representação z de outra visão da mesma imagem, em que os códigos são sempre computados a partir das visões globais. Assim, é levantada a hipótese central do método:

- Se duas visões, geradas a partir do aumento da imagem original, aprendem as mesmas informações através da rede, então deve ser possível predizer o código ou a atribuição de uma visão a partir da representação da outra.

Sendo assim, a rede é treinada para minimizar a discrepância entre as predições e os códigos trocados baseando-se na função de perda para cada par de visões (t, s):

$$\mathcal{L}(\mathbf{z}_t, \mathbf{z}_s) = \ell(\mathbf{z}_t, \mathbf{q}_s) + \ell(\mathbf{z}_s, \mathbf{q}_t), \tag{1}$$

onde o termo $\ell(\mathbf{z}, \mathbf{q})$ se trata da entropia cruzada entre o código e a probabilidade prevista pela rede, ou seja:

$$\ell(\mathbf{z}_t, \mathbf{q}_s) = -\sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)} \tag{2}$$

$\mathbf{q}_s^{(k)}$ é a probabilidade da visão s ser mapeada para o protótipo k e $\mathbf{p}_t^{(k)}$ é a previsão da visão t ser mapeada ao protótipo k, calculada através da *softmax*:

$$\mathbf{p}_t^{(k)} = \frac{\exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_{k'}\right)} \tag{3}$$

$\mathbf{z}_t^\top \mathbf{c}_k$ corresponde à similaridade entre a representação da visão e o protótipo e τ é parâmetro de temperatura da distribuição de probabilidade.

4.4.6 Avaliação das representações

Com a conclusão do pré-treino, a cabeça de projeção e os protótipos são descartados no modelo SwAV e apenas o *backbone*, com pesos ajustados ao longo do processo autossupervisionado, é aproveitado para as tarefas *downstream*. Nesse sentido, a cabeça de projeção explorou um espaço de aprendizado especializado para o objetivo de agrupamento e o *backbone* aprendeu representações gerais e transferíveis da entrada. O trabalho de Caron *et al.* aderiu ao procedimento padrão de aferição da qualidade das representações com o protocolo *linear probe*, amplamente adotado em SSL como referência de avaliação, pois isola as contribuições do pré-treino da capacidade do classificador. Com o *backbone* congelado, um classificador linear, de apenas uma camada *Linear*, é treinado sobre as *features* de 2048 dimensões extraídas. Assim, a acurácia obtida reflete a qualidade do aprendizado durante o pré-treino.

4.5 Arquitetura do modelo SwAV-HAR

Considerando a arquitetura do SwAV para *features* visuais comentada na seção anterior, este projeto final de graduação propõe a implementação do modelo SwAV-HAR para resolver a tarefa de classificação de atividades humanas (HAR) e a identificação de atividades de transição. A Figura 5 mostra uma visão geral da arquitetura do SwAV-HAR.

Nesta seção é discutida a arquitetura do modelo SwAV-HAR com ênfase nas etapas que se diferenciam do SwAV e outras etapas adicionais. Assim, a descrição da arquitetura do modelo SwAV-HAR abrange as fases de pré-treino e de avaliação. Na fase de pré-treino de SwAV-HAR são abordados os tópicos: a descrição dos dados de entrada; as transformações de aumento de séries temporais; a adaptação do *encoder* ResNet-SE-5; cabeça de projeção (MLP) e espaço dos protótipos; função de perda e atribuição de códigos; o agrupamento de características; e a fase de avaliação, por sua vez, descreve a metodologia de avaliação

das representações aprendidas, descrevendo os protocolos de *linear probe*, *full fine-tuning* e análise de agrupamento utilizados para aferir a qualidade do pré-treino.

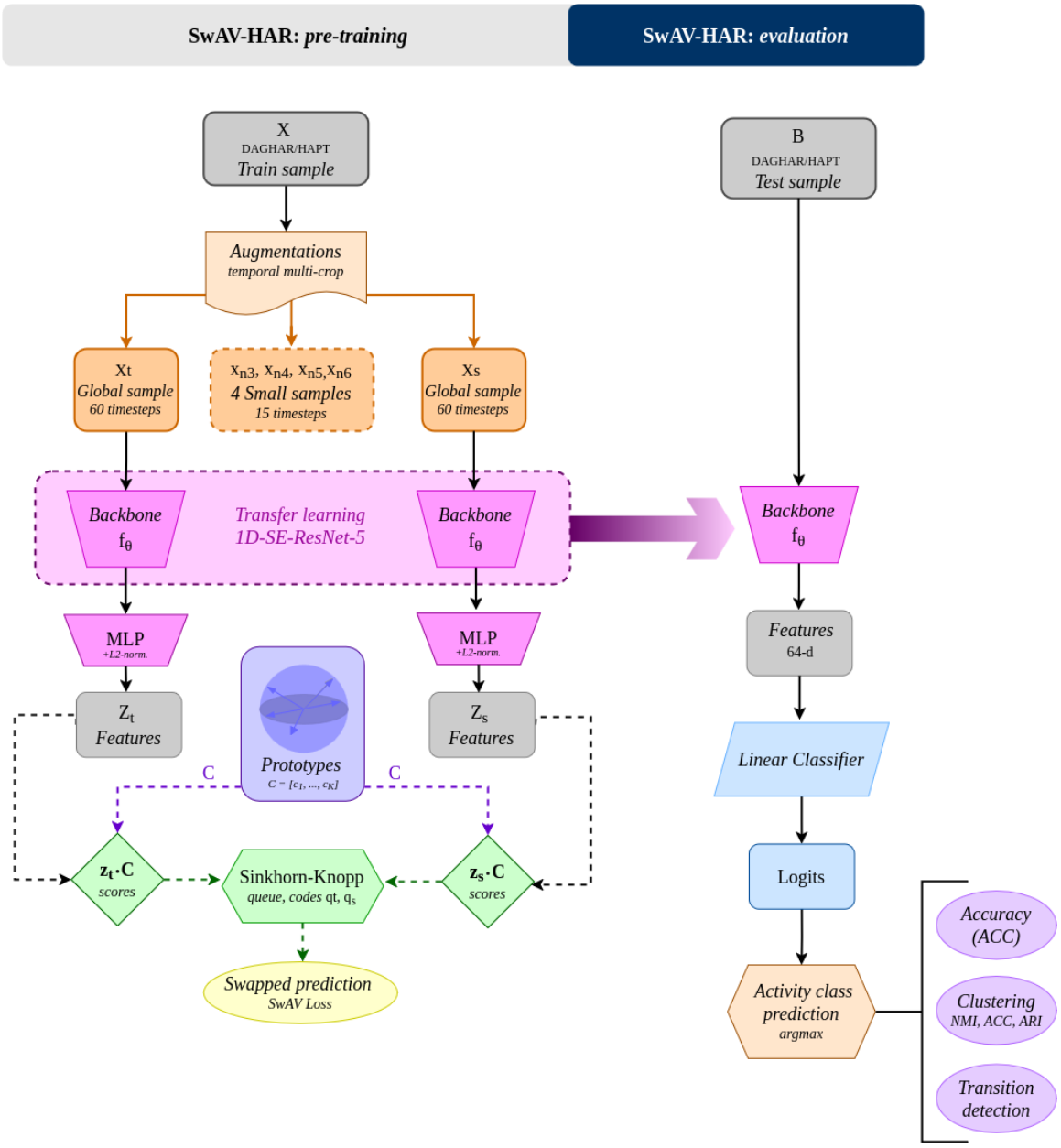

Figura 5: Diagrama da arquitetura SwAV-HAR.

4.5.1 Descrição dos dados de entrada: séries temporais de sensores inerciais

Originalmente, o modelo SwAV adota como entrada uma imagem RGB, com $H \times W$ *pixels* organizados em um tensor $[3, H, W]$. Durante a etapa inicial de adaptação da arquitetura, promoveu-se a mudança de domínio de imagem 2D para série temporal 1D, reduzindo a dimensionalidade. SwAV-HAR emprega uma janela de série temporal de sensores inerciais cujas amostras seguem o formato especificado no *benchmark* DAGHAR [3], detalhado na subseção de seleção de conjuntos de dados. Assim, cada amostra é um tensor $[6, 60]$, com 6 canais (acelerômetro-x, acelerômetro-y, acelerômetro-z, giroscópio-x, giroscópio-y e giroscópio-z) e 60 *timesteps* amostrados a $20\,\text{Hz} \times 3\,\text{s}$.

O código implementado consome os dados do repositório oficial do DAGHAR no formato *standardized view* e cada conjunto de dados selecionado é armazenado num arquivo .csv, com uma janela por linha. Assim, a classe `DAGHARDataset` realiza a leitura e a extração das colunas de sensores, além da reorganização dos tensores e da aplicação das transformações de aumento configuradas, que serão descritas a seguir.

4.5.2 Transformações de aumento: *temporal multi-crop*

No modelo SwAV-HAR, a estratégia central de *multi-crop* é modificada para o domínio temporal com a criação de 2 *views* globais e 4 *views* pequenas por amostra. Para as *views* globais, os 60 *timesteps* foram mantidos, preservando a janela completa. Assim, aplicaram-se 3 transformações, que constituem a configuração *baseline*:

- *Gaussian Noise*: $x + N(0,\, 0{,}05)$, simula a ocorrência de ruídos de sensor.
- *Scaling*: amplitude por canal $\times(1 + N(0,\, 0{,}1))$, simula uma variação de ganho.
- *Permutation*: divide a janela em $n_{seg} = 5$ segmentos e embaralha a ordem temporal.

Já para as *views* menores, o conjunto de transformações é descrito abaixo:

- *Random Crop*: 15 *timesteps* - subjanela aleatória de $\approx 0{,}75\,\text{s}$.
- *Gaussian Noise* e *Scaling* com os parâmetros idênticos aos configurados na *view* global.

A transformação *permutation* não é aplicada às *views* pequenas pois a subjanela de 15 *timesteps* já representa um recorte temporal restrito. Durante os estudos de ablação de transformações de aumento, as seguintes transformações foram ativadas:

- *Global Scaling*: fator de escala fixo para os 6 canais.
- *Axis Permutation*: permutação dos 3 eixos de acelerômetro e de giroscópio de forma independente.

A partir da definição das transformações, cada uma delas é aplicada à mesma amostra e a classe de transformações de aumento retorna um tensor das *views* concatenadas, em que as 2 primeiras são globais, $[6, 60]$, e as 4 últimas são pequenas, $[6, 15]$.

4.5.3 *Backbone*: ResNet-SE-5

No trabalho do modelo SwAV [1], foram divulgados resultados com o *backbone* ResNet-50, que mapeia imagens para vetores. Já neste projeto, o *backbone* padrão adotado é ResNet-SE-5, uma rede ResNet 1D com blocos residuais *Squeeze-and-Excitation* [13], importada diretamente da biblioteca Minerva [16].

O *backbone* possui 5 blocos residuais SE, em que cada bloco promove convoluções 1D seguidas de uma *branch* de *Squeeze-and-Excitation* para redefinir a importância de cada canal com *global average pooling*, camada *fully-connected*, *ReLU*, camada adicional *fully-connected* e sigmoide, respectivamente.

O método *forward* recebe a lista de 6 *views*, 2 *views* globais e 4 *views* pequenas, e as processa de forma agrupada por resolução temporal. Assim, o processamento conjunto por resolução gera ganhos de eficiência, necessitando de apenas 2 passagens pelo *backbone*, uma para cada categoria de *views*. Sendo assim, o resultado é um vetor de 64 dimensões por *view*, $[B * 6, 64]$, onde B representa o tamanho do *batch*. Diferentemente do SwAV original, cuja saída do *backbone* possui 2048 dimensões, a representação compacta de 64 dimensões impõe dificuldades à cabeça de projeção no que tange ao aprendizado de transformações expressivas a partir de um espaço de entrada menor. Por essa razão, a cabeça de projeção do SwAV-HAR utiliza uma camada oculta de 256 dimensões (4×64) para ampliar a capacidade de aprendizado de transformações não-lineares.

4.5.4 Cabeça de projeção (MLP) e espaço dos protótipos

Com as *features* resultantes da passagem pela rede ResNet-SE-5, a cabeça de projeção de SwAV-HAR é implementada de duas formas: quando `hidden_mlp=0`, utiliza-se apenas a camada linear `Linear(64,128)`. No entanto, na segunda forma, emprega-se o MLP com camada oculta de 256 dimensões para superar as limitações da dimensão do *backbone*. Nesse caso, aplica-se uma camada linear, seguida de *batch normalization*, *ReLU* e outra camada linear, respectivamente.

Assim como no SwAV, a versão adaptada aplica a normalização L2 para projetar os *embeddings* no espaço da hiperesfera unitária de 128 dimensões, o espaço dos protótipos. Os protótipos C são implementados com os pesos da camada linear sem viés $(128, K, bias = False)$, que também são normalizados na esfera antes de cada passagem *forward*. Desse modo, o produto escalar entre um *embedding* e um protótipo é equivalente à similaridade de cosseno, resultando nos *logits* utilizados para geração de códigos $[B \times 6, K]$. Desta forma, durante o desenvolvimento do modelo SwAV-HAR, com 6 classes de atividades e 1 classe de transição, a quantidade de protótipos (K) é um hiperparâmetro configurável, cujo valor adequado para o domínio de HAR foi discutido nos estudos de ablação.

4.5.5 Função de perda SwAV-HAR e atribuição de códigos

Conceitualmente, manteve-se a função de perda do SwAV analisada na equação (1), com o problema da predição trocada para as 6 *views*, mas adotou-se a precisão dupla *(float64)* internamente durante a execução do algoritmo de Sinkhorn-Knopp para garantir a estabilidade numérica para valores pequenos de *epsilon*, já que nesses casos os *scores* divididos

por ϵ podem causar *overflow*. Deve-se ressaltar ainda que os códigos q, gerados pelo algoritmo de Sinkhorn-Knopp, são computados somente a partir das 2 *views* globais, enquanto as demais apenas predizem esses códigos. Essa assimetria viabiliza o *multi-crop*, já que as *views* pequenas aumentam o número de comparações na perda, sem aumentar o custo do algoritmo.

Dessa maneira, a perda final é a média das contribuições sobre as duas *views* globais utilizadas como âncora. Adicionalmente, foi implementada uma fila de *embeddings* de iterações recentes, cuja finalidade é a estabilização do algoritmo para um lote nas primeiras etapas do treinamento. Os conjuntos de dados selecionados possuem volumes de amostras menores do que o ImageNet, utilizado originalmente, e o tamanho do lote é ajustado entre os valores $B \in \{64, 128, 256\}$ conforme a disponibilidade de amostras de treinamento. Assim, para lotes menores é difícil satisfazer a condição de equipartição do algoritmo de Sinkhorn-Knopp, justificando o uso da fila. Por último, os *embeddings* da fila não geram gradiente, mas diversificam o conjunto de pontos antes da distribuição uniforme dos protótipos.

4.5.6 O agrupamento de características

Após o mapeamento entre as *features* de séries temporais e os protótipos através da atribuição dos códigos, neste projeto são apresentadas três perspectivas complementares de análise da estrutura do espaço latente aprendido. A primeira utiliza diretamente os protótipos aprendidos pelo SwAV-HAR, em que cada amostra é atribuída ao protótipo de maior *score* via *argmax*. Contudo, como K pode ser maior que o número de classes $\in \{6, 7\}$, o mapeamento ótimo é resolvido pelo algoritmo Húngaro retangular sobre a matriz de custo de dimensão $[n_{classes} \times K]$, onde cada entrada representa a sobreposição entre amostras atribuídas a um protótipo e as amostras de cada classe verdadeira. Por hipótese, isso permite identificar protótipos especializados em regiões de fronteira entre atividades, que indicariam a presença de janelas de transição.

Já a segunda perspectiva é a aplicação do algoritmo *K-Means* com $K = n_{classes}$ diretamente sobre as *features* extraídas do *backbone*, permitindo avaliar a separabilidade global das atividades. A terceira e última perspectiva mede a separabilidade das classes verdadeiras no espaço das *features* do *backbone* sem aplicação de nenhum algoritmo de agrupamento, apenas com métricas calculadas diretamente sobre os rótulos reais, a fim de avaliar a contribuição do pré-treino separadamente.

4.6 Treinamento e avaliação das representações

Concluído o pré-treino, a cabeça de projeção e os protótipos não são utilizados nos protocolos de classificação, como no SwAV, mas os protótipos são preservados para a análise de agrupamento e a detecção de transições. Assim, apenas o *backbone* é aproveitado nas tarefas de avaliação. Esta seção descreve os protocolos de treinamento para classificação de atividades, as métricas quantitativas para avaliar a qualidade das representações e as métricas qualitativas empregadas na análise dos resultados.

4.6.1 Protocolos de treinamento

O modelo SwAV-HAR adota os protocolos de treinamento amplamente utilizados em aprendizado autossupervisionado para avaliar a qualidade das representações aprendidas: *freeze*, *full fine-tuning* e *from scratch*. No modo *freeze*, o *backbone* é carregado a partir do *checkpoint* pré-treinado e todos os parâmetros são congelados. Assim, as *features* são extraídas uma única vez em modo de inferência e o classificador é treinado sobre essas representações.

Já no modo *full fine-tuning*, o *backbone* é inicializado com os pesos do *checkpoint* SwAV-HAR e descongelado durante o treinamento com uma cabeça linear adicional. Assim, *backbone* e cabeça de classificação são treinados de ponta a ponta (*end-to-end*) com taxas de aprendizado configuradas com base na busca de hiperparâmetros nos estudos de ablação.

Por último, o modo *from scratch* instancia o *backbone* com pesos aleatórios e o treina junto à cabeça de classificação com uma taxa de aprendizado configurada, servindo como *baseline* supervisionado sem pré-treino. De maneira complementar, o protocolo de "amostras por classe" (SPC) executa os três modos descritos, variando o total e o percentual de amostras do conjunto de dados durante o treinamento do classificador, o que permite avaliar a eficiência do pré-treino em regimes de poucos rótulos.

4.6.2 Métricas quantitativas

Considerando as perspectivas de avaliação da qualidade das representações, foram extraídas 5 métricas para agrupamento, sendo elas: *silhouette score*, *Davies-Bouldin* (DB), *Normalized Mutual Information* (NMI), *Adjusted Rand Index (ARI)* e acurácia (ACC *Clustering*) dos agrupamentos. Além disso, com o classificador treinado sobre as *features* do *backbone* congelado, também se calculou a acurácia linear (ACC. linear) entre as classes preditas pelo classificador e as classes reais com quantidade de amostras reduzida no treinamento com protocolo SPC.

4.6.3 Métricas de detecção de transições

Para o conjunto de dados HAPT, a detecção de transições é investigada a partir do problema de classificação binária. Dado o modelo SwAV-HAR pré-treinado no modo *freeze*, avalia-se se as janelas rotuladas como transição (classe 6) são detectáveis no espaço latente por meio dos protótipos aprendidos. Para isso, são computados 2 valores contínuos *(scores)* a partir da similaridade de cosseno entre os *embeddings* e os K protótipos. O primeiro é a distância ao protótipo, definida como o complemento da similaridade máxima de uma janela com qualquer protótipo:

$$s_{\text{dist}}(i) = 1 - \max_{k} \ \text{sim}(z_i, c_k) \tag{4}$$

onde $\text{sim}(z_i, c_k)$ é a similaridade de cosseno entre o *embedding* z_i e o protótipo c_k. Valores elevados de $s_{\text{dist}}(i)$ correspondem a baixa similaridade máxima com os protótipos, indicando que a janela não pertence com clareza a nenhum protótipo aprendido, comportamento esperado para janelas de transição.

O segundo valor é a taxa de transição de protótipos, que mede a diversidade de atribuições rígidas numa janela deslizante de tamanho W centrada na janela i. Para cada janela j, define-se o protótipo dominante como aquele com maior similaridade de cosseno com o *embedding* de j, $\hat{k}_j = \arg\max_k \mathrm{sim}(z_j, c_k)$. O *score* é então dado pela fração de protótipos dominantes distintos adotados nas W_i janelas vizinhas de i:

$$s_{\mathrm{tr}}(i) = \frac{\left|\{\, \hat{k}_j : j \in \mathcal{W}(i) \,\}\right|}{W_i} \tag{5}$$

onde $\mathcal{W}(i)$ é a vizinhança de tamanho $W = 5$ centrada em i e $W_i = |\mathcal{W}(i)| \leq W$ é o número efetivo de janelas, sendo menor nas bordas da sequência. $|\{\cdot\}|$ denota a cardinalidade do conjunto de protótipos dominantes distintos. Valores próximos de 1 indicam que janelas vizinhas adotam protótipos distintos, evidenciando uma mudança de atividade.

As Tabelas 2 e 3 resumem todas as métricas quantitativas avaliativas.

Tabela 2: Métricas quantitativas de agrupamento não-supervisionado (pré-treino).

Perspectiva	**Métrica**	**Intervalo**	**Finalidade**
Protótipos SwAV-HAR	NMI	$[0, 1]$	Informação mútua normalizada entre atribuições de protótipos e rótulos verdadeiros.
	ARI	$[-1, 1]$	Concordância par a par entre atribuição de protótipos e classes verdadeiras, corrigida pelo acaso.
	ACC	$[0, 1]$	Acurácia após mapeamento entre os K protótipos e as $n_{classes}$ via algoritmo Húngaro retangular.
K-Means ($K = n_{classes}$)	NMI	$[0, 1]$	Informação mútua normalizada entre agrupamentos *K-Means* e rótulos verdadeiros.
	ARI	$[-1, 1]$	Concordância entre partição *K-Means* e classes verdadeiras, corrigida pelo acaso.
	ACC	$[0, 1]$	Acurácia após mapeamento entre os agrupamentos e $n_{classes}$ via algoritmo Húngaro.
	Silhouette	$[-1, 1]$	Coesão intra-cluster relativa à separação inter-cluster dos grupos *K-Means*.
	DB	$[0, \infty)$	Razão entre dispersão intra-cluster e separação inter-cluster.
Separabilidade das classes	Silhouette	$[-1, 1]$	Coesão e separação das $n_{classes}$ verdadeiras no espaço de *features*, sem aplicação de algoritmo de agrupamento.
	DB	$[0, \infty)$	Razão entre dispersão e separação das classes verdadeiras no espaço de *features*.

Tabela 3: Métricas quantitativas de avaliação de classificação e detecção de transições.

Perspectiva	**Métrica**	**Intervalo**	**Finalidade**
Classificador	ACC *freeze*	$[0, 1]$	Acurácia de classificação com *backbone* congelado; medição da separabilidade das representações SSL puras.
	ACC *full fine-tuning*	$[0, 1]$	Acurácia com *backbone* descongelado e inicializado pelos pesos pré-treinados; medição do ganho da inicialização SSL.
	ACC *from scratch*	$[0, 1]$	Acurácia com *backbone* de pesos aleatórios; serve como *baseline* sem pré-treino.
SPC	ACC (1–100%)	$[0, 1]$	Acurácia em função do percentual de amostras rotuladas por classe, executada nos três modos descritos.
Detecção de transições (HAPT)	Dist. protótipo	$[0, 1]$	*Score.* Complemento da maior similaridade de cosseno entre uma janela e os protótipos; sob a hipótese de teste, valores altos indicam regiões de fronteira entre atividades.
	Taxa de transição	$[0, 1]$	*Score.* Fração de protótipos dominantes distintos em uma vizinhança deslizante de W janelas; valores altos (picos) sinalizam transições entre atividades.
	AUROC	$[0, 1]$	*Métrica.* Área sob a curva ROC de cada *score*; mede a separação entre janelas estáveis e de transição.
	AP	$[0, 1]$	*Métrica.* Precisão média (área sob a curva de precisão-revocação); enfatiza a classe minoritária.

4.6.4 Métricas qualitativas

Além dos indicadores quantitativos, técnicas de inspeção visual, tais como gráficos com o algoritmo t-SNE e a visualização da matriz de confusão, foram empregadas para fomentar as discussões a respeito da separabilidade das classes nos espaços latentes aprendidos. A técnica t-SNE *(t-distributed Stochastic Neighbor Embedding)* [17] consiste na redução de dimensionalidade não-linear e na projeção dos *embeddings* em duas dimensões, preservando a estrutura local de vizinhança. Já a visualização da matriz de confusão permite identificar quais atividades são confundidas pelos agrupamentos não-supervisionados com base nos resultados de revocação para cada uma das classes.

4.7 Ambiente de desenvolvimento

4.7.1 Linguagem, bibliotecas e *frameworks*

O modelo SwAV-HAR foi implementado com a linguagem Python nas versões v3.10+ e v3.13, quando executado em Dev-Container ou localmente em ambiente virtual (venv), respectivamente. O projeto admite *scripts shell* para orquestração da *pipeline end-to-end*, descritos no repositório reprodutível. As dependências principais do projeto foram pré-instaladas via imagem Docker, com destaque para o *framework* de *deep learning*, PyTorch, as bibliotecas scikit-learn, NumPy, Pandas, SciPy e Matplotlib. Além disso, utilizou-se a biblioteca externa Minerva [16], que fornece a implementação base para o *backbone* ResNet-SE-5. Outras dependências são listadas no repositório de SwAV-HAR.

4.7.2 Recursos computacionais

O desenvolvimento deste projeto também contou com a disponibilidade do *cluster* Recod, com *drivers* NVIDIA e suporte a CUDA para execução dos experimentos e validação do modelo com a utilização de unidades de processamento gráfico (GPUs). Abaixo, as Tabelas 4 e 5 exibem as especificações das 3 GPUs utilizadas nos experimentos e a duração média de execução.

Tabela 4: Especificações do ambiente de GPU utilizado nos experimentos.

Propriedade	**Valor**
Modelo de GPU	NVIDIA Tesla P100-PCIE-12GB
Quantidade de GPUs	3
Memória por GPU	12 288 MiB (12 GB HBM2)
Arquitetura	Pascal (GP100)
Precisão FP32	~9,3 TFLOPS
Precisão FP16	~18,7 TFLOPS
CUDA suportada	até 12.2

A Tabela 5 exibe a duração média de execução dos experimentos no *cluster* Recod, com pré-treino em 3 GPUs (Tesla P100-PCIE-12GB) e a avaliação em uma GPU de mesma especificação.

Tabela 5: Duração média de execução de experimentos no *cluster*.

Fase	**Configuração**	**GPUs**	**Duração**
Pré-treino SwAV-HAR (30k *steps*)	Média entre conjuntos de dados	3	≈40 min
Avaliação – modo *freeze*	Extração de *features*	1	≈10 s
	Treino do classificador	1	≈2 min
	Agrupamento + gráfico t-SNE	1	≈10 s
Modo *full fine-tuning* (≈ 100 épocas)	Treinamento do *backbone* + classificador	1	≈5–8 min
Modo *from scratch* (≈ 100 épocas)	Inicialização do treino com pesos aleatórios	1	≈5–8 min
Inferência	400 a 2 400 amostras	1	<10 s

5 Resultados

5.1 Resultados quantitativos

As avaliações das métricas descritas na seção de metodologia foram realizadas em dois cenários: *in-domain* e *cross-domain* com *leave-one-dataset-out* (LODO). No cenário *in-domain*, SwAV-HAR é pré-treinado e avaliado no mesmo conjunto de dados, enquanto no cenário LODO o modelo é pré-treinado nos outros 6 conjuntos de dados com os subconjuntos de treino concatenados e a avaliação ocorre apenas no conjunto de dados alvo. As análises apresentadas referem-se aos experimentos com o número de protótipos $K = 12$ e variante *baseline* com 30 000 *steps* de pré-treino. A escolha desta configuração e a busca de hiperparâmetros foram exploradas no apêndice de estudos de ablação.

As Tabelas 6, 7 e as Figuras 6, 7 e 8 apresentam os resultados encontrados no cenário *in-domain* para as métricas avaliativas quantitativas e protocolo SPC[5]. Considerando os três modos de avaliação, *full fine-tuning* evidencia a maior acurácia (ver Figura 6), com média de 88%, atingindo valor máximo de $acc = 97\%$ no UCI e $acc = 93{,}5\%$ no HAPT. O desempenho do modo *from scratch*, com a média de 85%, comparada ao modo *full fine-tuning* evidencia que o pré-treino resulta em representações latentes mais separáveis em relação à utilização de pesos aleatórios, com ganhos em conjuntos de dados menores. No modo *freeze*, por sua vez, a acurácia é sistematicamente inferior, com média de 75%, mas o comportamento dos conjuntos de dados indica que as coletas de UCI, RealWorld Waist e WISDM atingem resultados melhores, enquanto RealWorld Thigh e Ku-HAR permanecem com a acurácia entre 55% e 60%.

[5]Do inglês: *Samples per class*.

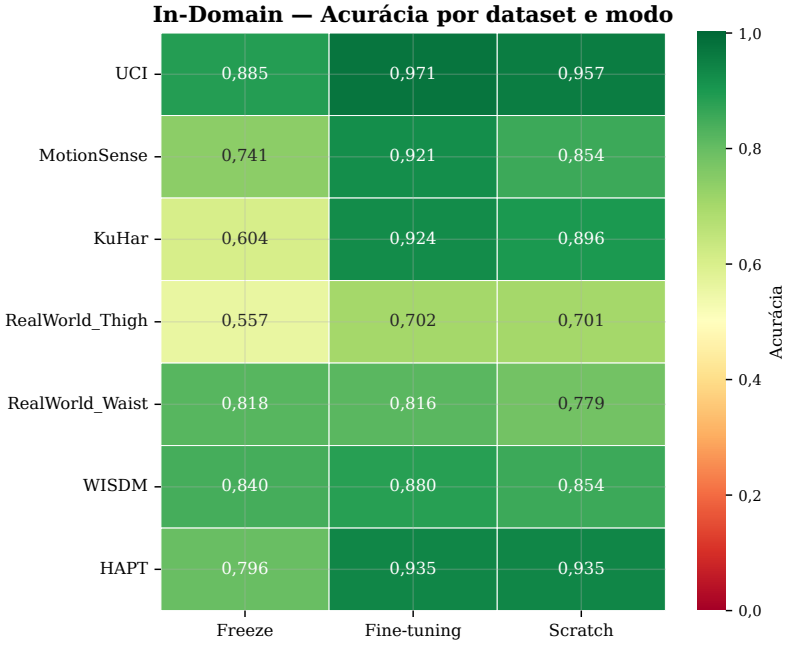

Figura 6: *Heatmap* para avaliação de acurácia no cenário *in-domain*.

A disparidade do modo *freeze* reflete-se no grau de alinhamento entre os agrupamentos aprendidos e as classes reais, considerando as métricas de agrupamento com variações de NMI_{km} entre 0,34 (HAPT), 0,36 (WISDM) e 0,67 (UCI), verificadas na Tabela 6. Assim, a acurácia de agrupamento é superior nos modos *full fine-tuning* e *from scratch*, com melhores resultados de alinhamento para o treinamento *full fine-tuning* na maioria dos conjuntos de dados, exceto UCI e WISDM. Já para a métrica *Adjusted Rand Index* (ARI), a concordância entre as partições e classes verdadeiras de atividades é mais elevada no modo *full fine-tuning* na maioria dos conjuntos, exceto para UCI, WISDM e RealWorld Waist, demonstrando os ganhos de agrupamento com treinamento conjunto do *backbone* e do classificador.

Tabela 6: *K-Means–* Resultados *in-domain* com total de amostras.

	Freeze			*Fine-tuning*			*Scratch*		
	ACC_{km}	NMI_{km}	ARI_{km}	ACC_{km}	NMI_{km}	ARI_{km}	ACC_{km}	NMI_{km}	ARI_{km}
Dataset									
UCI	0,562	0,673	0,544	0,881	0,896	0,878	0,916	0,891	0,882
MotionSense	0,274	0,348	0,106	0,912	0,822	0,804	0,883	0,793	0,751
Ku-HAR	0,370	0,513	0,248	0,958	0,916	0,903	0,910	0,850	0,802
RealWorld Thigh	0,430	0,506	0,223	0,767	0,637	0,570	0,664	0,675	0,534
RealWorld Waist	0,373	0,393	0,174	0,806	0,681	0,612	0,807	0,674	0,614
WISDM	0,410	0,356	0,236	0,583	0,677	0,503	0,660	0,664	0,587
HAPT	0,289	0,338	0,158	0,939	0,871	0,862	0,935	0,870	0,851

Ainda a respeito da métrica de acurácia de classificação, a Figura 7 exibe as curvas

de acurácia em função do percentual de amostras utilizadas durante o treinamento do classificador, evidenciando a disparidade de acurácia no modo *freeze*. Particularmente, observa-se um pequeno ganho de acurácia no conjunto de dados RealWorld Waist, com $acc = 74\%$ para 1% de amostras no treinamento *freeze* em relação aos modos *full fine-tuning* e *from scratch*. Na maior parte dos regimes, o modo *full fine-tuning* apresenta os melhores resultados de acurácia entre os conjuntos de dados, com destaque para Ku-HAR, RealWorld Waist, WISDM e HAPT.

Tabela 7: Resultados *in-domain* - avaliação de protótipos.

Dataset	ACC_p	NMI_p	ARI_p
UCI	0,512	0,650	0,460
MotionSense	0,574	0,508	0,372
Ku-HAR	0,553	0,604	0,467
RealWorld Thigh	0,628	0,596	0,514
RealWorld Waist	0,554	0,597	0,457
WISDM	0,440	0,471	0,337
HAPT	0,493	0,523	0,342

Já no cenário de agrupamento promovido pelos protótipos durante o pré-treinamento (Tabela 7), a acurácia de agrupamento, obtida atribuindo cada amostra ao protótipo de maior pontuação e mapeando os K protótipos às classes, mantém-se sistematicamente abaixo de 63%, refletindo a natureza autossupervisionada do pré-treino, que distribui as amostras entre protótipos sem acesso aos rótulos. Sob essa ótica, os protótipos são otimizados para consistência de agrupamento, não para o alinhamento com as classes. No entanto, a métrica de NMI é mais robusta a permutações de rótulos, o que indica a captura de informação sobre as classes no cenário não-supervisionado. Nota-se ainda que as métricas de agrupamento avaliadas sobre os protótipos apresentam uniformidade entre os conjuntos de dados, sugerindo que a qualidade dos protótipos é decorrente principalmente do regime de pré-treino, enquanto a transferência do *backbone* para avaliação da acurácia de classificação depende das características intrínsecas de cada conjunto de dados, bem como dos protótipos aprendidos durante o pré-treino.

Com base na Figura 8, SwAV-HAR torna-se vantajoso tanto em conjuntos de dados pequenos, como no UCI, quanto em conjuntos de dados grandes, como RealWorld Waist, nos regimes de poucas amostras por classe. Nesse sentido, o treinamento *full fine-tuning* com *backbone* ResNet-SE-5 em todos os cenários de variação de amostras demonstra que SwAV-HAR é superior a outras técnicas de SSL no UCI, RealWorld Waist, MotionSense e KuHAR, enquanto se mantém competitivo com o estado da arte nos conjuntos de dados RealWorld Thigh e WISDM.

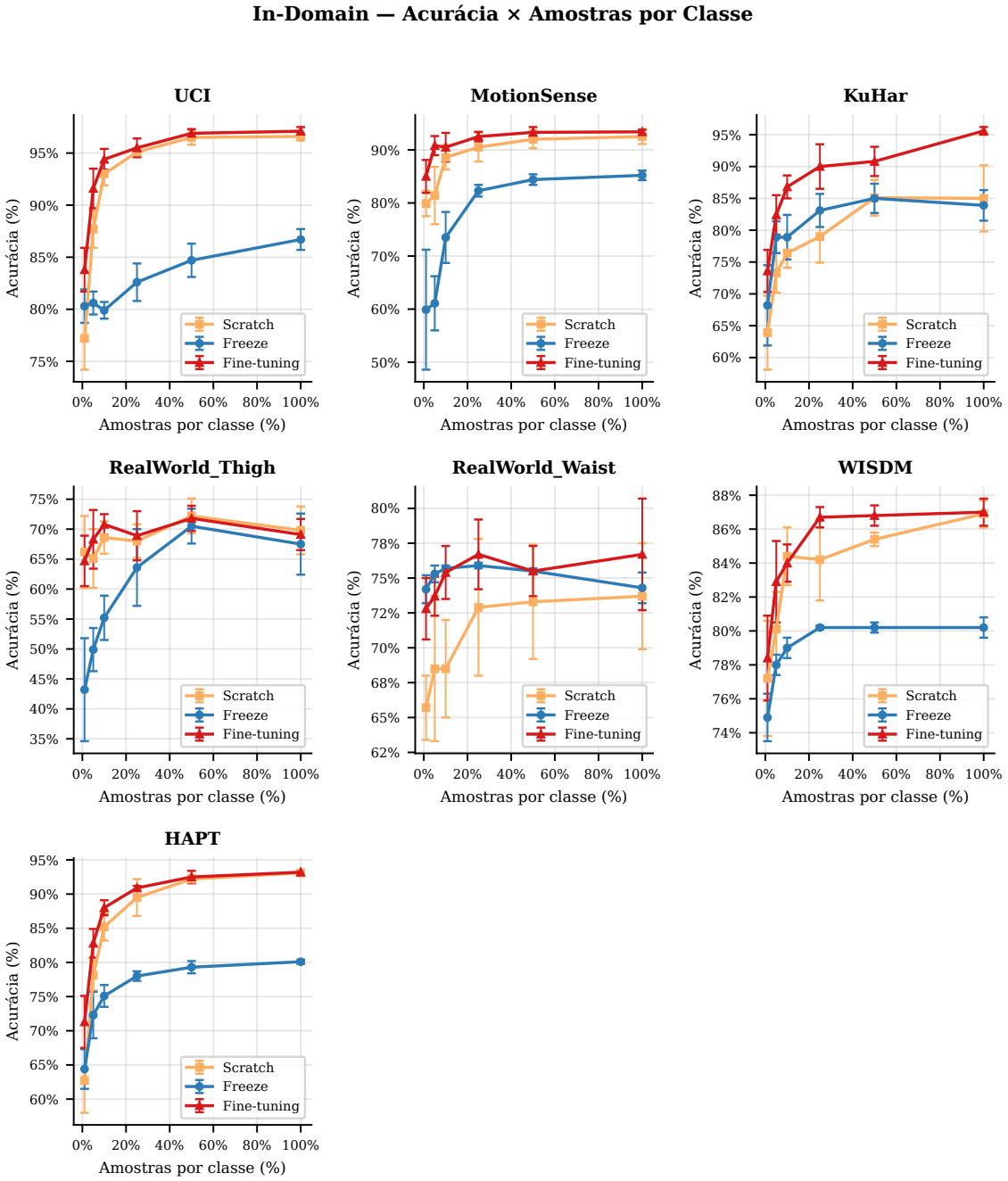

Figura 7: Acurácia de classificação - resultados *in-domain* para o protocolo SPC.

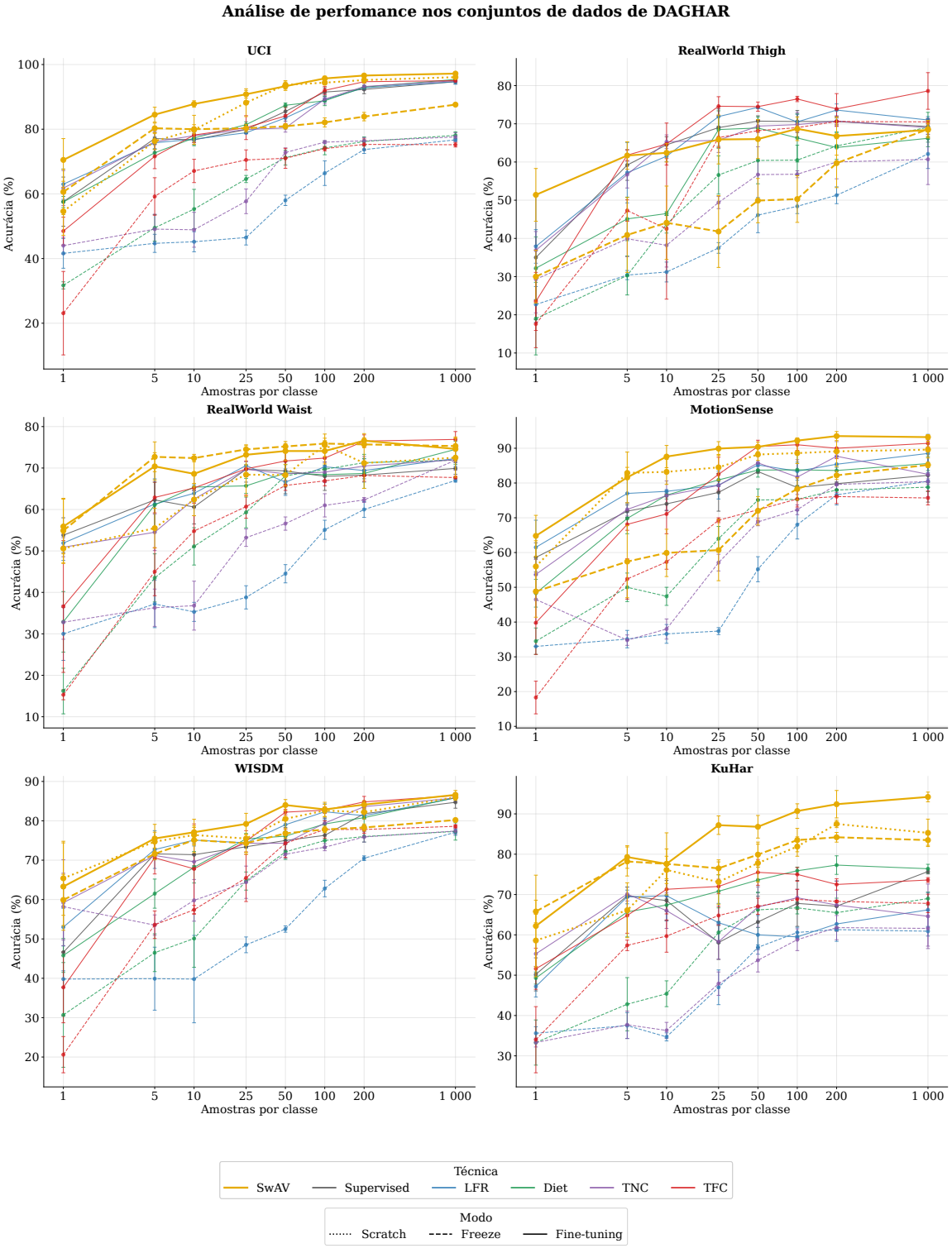

Figura 8: Comparação de desempenho de SwAV-HAR com os resultados reportados por da Luz *et al.* [13] para as técnicas LFR, Diet, TNC, TFC e classificador supervisionado.

As Tabelas 8 e 9 e a Figura 9 exibem os resultados das métricas quantitativas para o cenário *leave-one-dataset-out* (LODO). Nota-se que o modo de treinamento *full fine-tuning* fornece os melhores resultados para a acurácia de classificação, com média de $acc = 0{,}872$, superando os modos *from scratch* e *freeze* (ver Figura 9). Nesse aspecto, o ganho de *full fine-tuning* em relação ao modo *freeze* varia entre conjuntos de dados alvo, sendo mais significativo no Ku-HAR e no HAPT. Nesses conjuntos, o *backbone* pré-treinado nos demais conjuntos de dados carrega uma representação útil, mas pouco alinhada ao domínio-alvo com treinamento *freeze*, que apresenta os menores valores de acurácia observados nos experimentos LODO. Além disso, os conjuntos de dados RealWorld Thigh e RealWorld Waist ainda aparecem com desempenho inferior no regime de *full fine-tuning* em relação aos outros conjuntos de dados, evidenciando o desafio de generalização devido ao posicionamento diferenciado dos dispositivos de coleta.

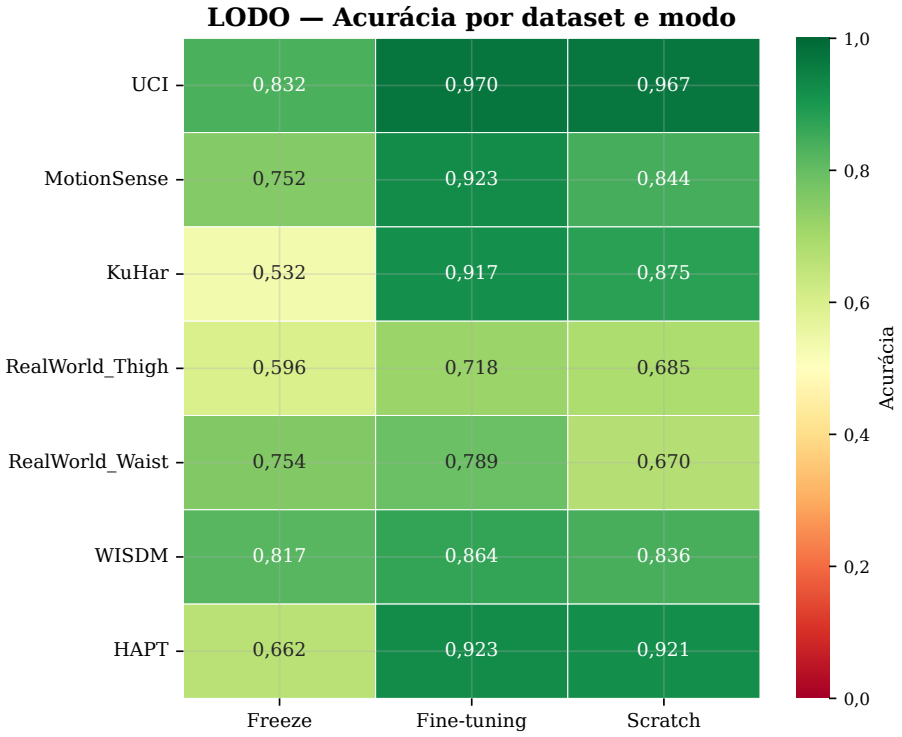

Figura 9: *Heatmap* para avaliação de acurácia no cenário LODO.

Os resultados de agrupamento para o cenário LODO na Tabela 8 indicam uma discrepância na qualidade estrutural das representações aprendidas entre os modos *freeze* e *full fine-tuning*. Entre os conjuntos de dados, mesmo com acurácia de classificação aceitável no modo *freeze* para RealWorld Waist, $acc = 0{,}75$, a estrutura do agrupamento *K-Means* é fortemente degradada, o que demonstra que as representações aprendidas têm bom poder discriminativo, mas os *embeddings* não formam grupos geometricamente naturais para esse conjunto de dados.

Tabela 8: *K-Means*– Resultados no cenário *LODO* com total de amostras.

	Freeze			*Fine-tuning*			*Scratch*		
	ACC_{km}	NMI_{km}	ARI_{km}	ACC_{km}	NMI_{km}	ARI_{km}	ACC_{km}	NMI_{km}	ARI_{km}
Dataset									
UCI	0,441	0,519	0,315	0,901	0,914	0,896	0,897	0,887	0,874
MotionSense	0,367	0,435	0,160	0,920	0,843	0,824	0,883	0,793	0,746
Ku-HAR	0,428	0,519	0,270	0,944	0,895	0,873	0,896	0,843	0,786
RealWorld Thigh	0,459	0,458	0,257	0,803	0,665	0,594	0,760	0,628	0,544
RealWorld Waist	0,267	0,328	0,154	0,813	0,682	0,629	0,642	0,645	0,518
WISDM	0,692	0,527	0,455	0,672	0,650	0,520	0,732	0,695	0,620
HAPT	0,337	0,433	0,259	0,925	0,858	0,836	0,909	0,820	0,796

Já no espaço de avaliação dos protótipos nativos de SwAV-HAR (Tabela 9), ainda que distante dos resultados adequados, a acurácia de agrupamento supera os resultados de *K-Means* no modo *freeze*, exceto para WISDM, com destaque para RealWorld Waist e MotionSense. Isso sugere que os protótipos aprendidos durante o pré-treino nos seis conjuntos da rodada tendem a apresentar boas estruturas, mas nem sempre se realinham com as classes do conjunto de dados alvo em LODO, como nos casos de WISDM e RealWorld Thigh.

Tabela 9: Resultados no cenário LODO - avaliação de protótipos.

Dataset	ACC_p	NMI_p	ARI_p
UCI	0,577	0,587	0,450
MotionSense	0,505	0,545	0,368
Ku-HAR	0,528	0,595	0,449
RealWorld Thigh	0,427	0,440	0,250
RealWorld Waist	0,583	0,525	0,423
WISDM	0,535	0,586	0,453
HAPT	0,348	0,486	0,295

Por fim, para as análises de classes de transição, os *scores* de distância ao protótipo mais próximo e taxa de transição foram avaliados no conjunto de dados HAPT no cenário *in-domain*. Os resultados são apresentados nas Figuras 10, 11 e 12. A Figura 10 apresenta as distribuições dos *scores* para janelas estáveis e de transição. Já a Figura 11 sintetiza a capacidade discriminativa de cada *score* por meio das curvas ROC e de precisão-revocação, resumidas pela área sob a curva ROC e pela precisão média (AP). Nota-se que a taxa de transição apresenta o maior poder discriminativo, com AUROC= 0,745 e AP= 0,127, enquanto a distância ao protótipo obtém AUROC= 0,243 e AP= 0,045. No caso da distância ao protótipo, o valor de AUROC abaixo de 0,5 contraria a hipótese de que regiões de fronteira estariam mais afastadas dos centroides aprendidos, visto que as janelas de transição tendem a apresentar menor distância ao protótipo. Por sua vez, os valores baixos de AP para ambos os *scores* refletem a raridade das janelas de transição no conjunto, cenário em que a precisão

é penalizada.

Os gráficos t-SNE apresentados na Figura 12 evidenciam que os agrupamentos não estão totalmente separados e a região de transição não recebe um protótipo dedicado, sendo repartida entre grupos associados às atividades estáveis vizinhas. Nesse aspecto, a distância ao protótipo falha devido à falta de isolamento das transições no espaço latente.

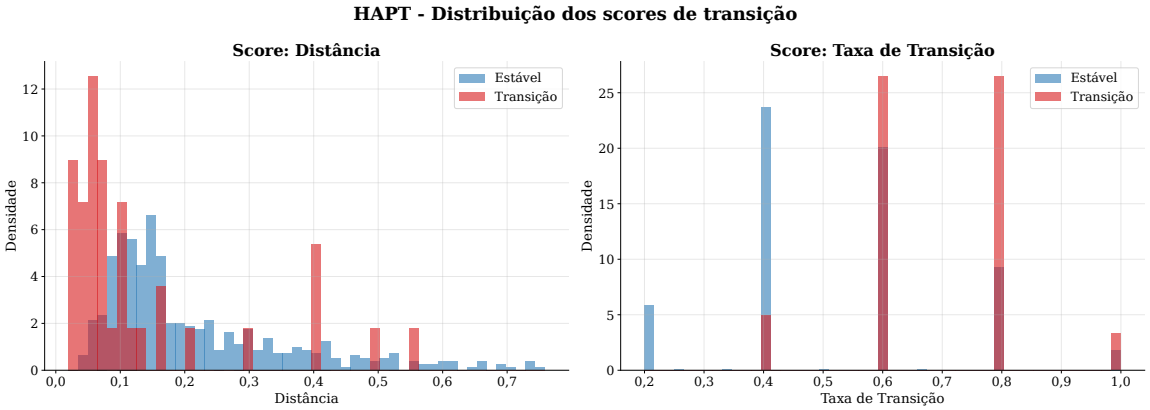

Figura 10: Distribuição das métricas de avaliação de atividades de transição.

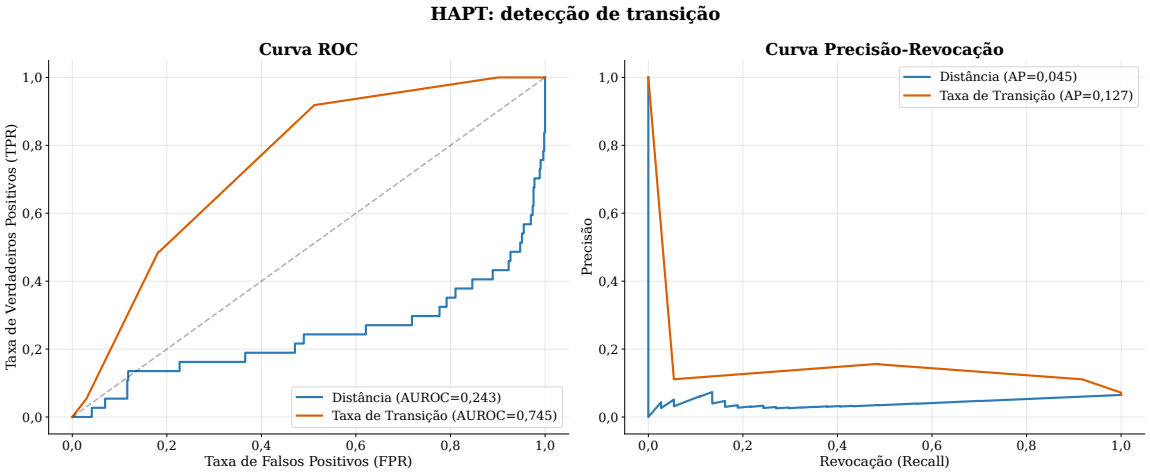

Figura 11: Comparação das métricas de transição com curva ROC e precisão-revocação.

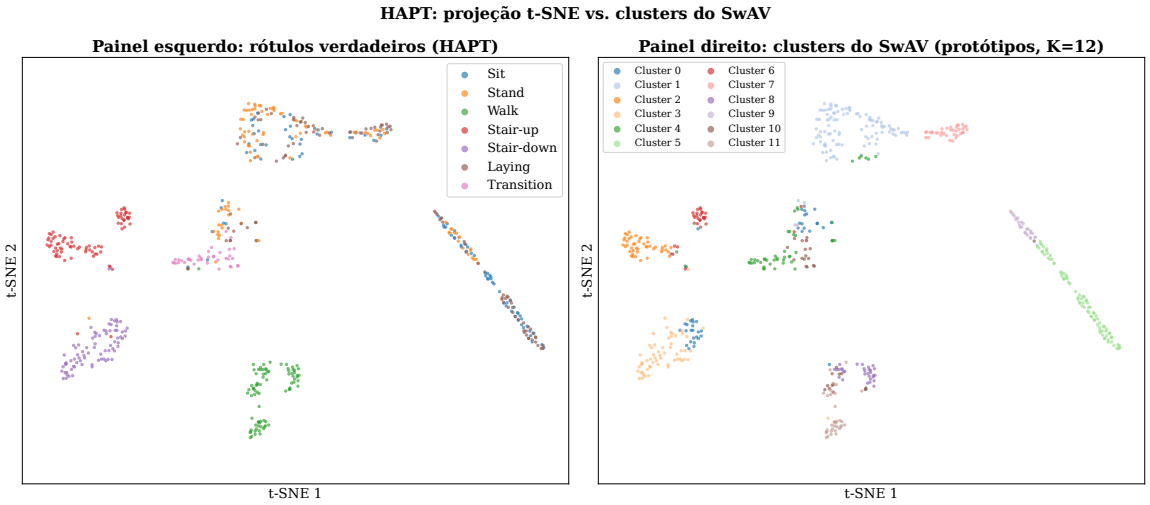

Figura 12: Gráficos t-SNE para os rótulos verdadeiros (esquerda) e agrupamentos com SwAV-HAR (direita) no conjunto de dados HAPT.

5.2 Resultados qualitativos

Esta subseção apresenta os gráficos t-SNE para os 6 conjuntos de dados nas Figuras 13 e 14, comparando através da projeção compartilhada a geometria das classes para os espaços latentes de protótipos e *K-Means*. Além disso, são apresentadas as matrizes de confusão para cada conjunto de dados.

A análise qualitativa das projeções t-SNE evidencia a diferença de nível de detalhamento entre as estratégias de agrupamento. As projeções dos protótipos SwAV-HAR ($K = 12$) fragmentam as atividades em múltiplos grupos, com mais de um protótipo por classe, conferindo capacidade para representar variações intraclasse e regiões de fronteira. Já no que diz respeito aos agrupamentos com *K-Means* ($K = n_{classes}$) sobre o *backbone* ajustado com *full fine-tuning*, os agrupamentos são mais compactos e globalmente separados para as respectivas classes de atividades. Nos dois cenários de agrupamento, as atividades de alta intensidade tendem a formar regiões mais delimitadas, enquanto as posturas estáticas ocupam regiões vizinhas. Nota-se ainda que conjuntos de dados com menor número de amostras, como Ku-HAR, têm projeções mais esparsas.

No aspecto de desempenho de acurácia de agrupamento, as matrizes de confusão reafirmam as observações dos gráficos t-SNE. Observa-se que o erro mais frequente para o agrupamento por protótipos é a confusão entre as classes sentado e em pé nos conjuntos de dados Ku-HAR, RealWorld Thigh e RealWorld Waist. Isso se deve à natureza de baixa energia das classes, em que as diferenças se restringem sobretudo à orientação do sensor. Já as classes de atividades dinâmicas são recuperadas com maior acurácia, apesar de algumas confusões entre as atividades de subir/descer escadas e caminhar no conjunto de dados RealWorld Thigh, com revocação $revoc = 0{,}46$.

Do ponto de vista de *K-Means*, a confusão é reduzida para as classes sentado e em pé, com melhorias significativas nos conjuntos de dados MotionSense e RealWorld Waist, com

revocação da classe em pé $revoc_{MotionSense} = 0{,}97$ e $revoc_{RealWorldWaist} = 0{,}72$, respectivamente. Entretanto, no conjunto de dados WISDM, ainda se observa o colapso dessas classes, evidenciando a influência das particularidades de cada conjunto de dados para as classes similares no aspecto de energia. Deve-se ressaltar ainda que as fontes de ganhos de revocação observados nas classes e conjuntos de dados são referentes tanto ao ajuste fino do *backbone* inicializado com os pesos aprendidos de SwAV-HAR, quanto ao algoritmo de *K-Means* executado com o número de centroides exatamente igual ao número de classes.

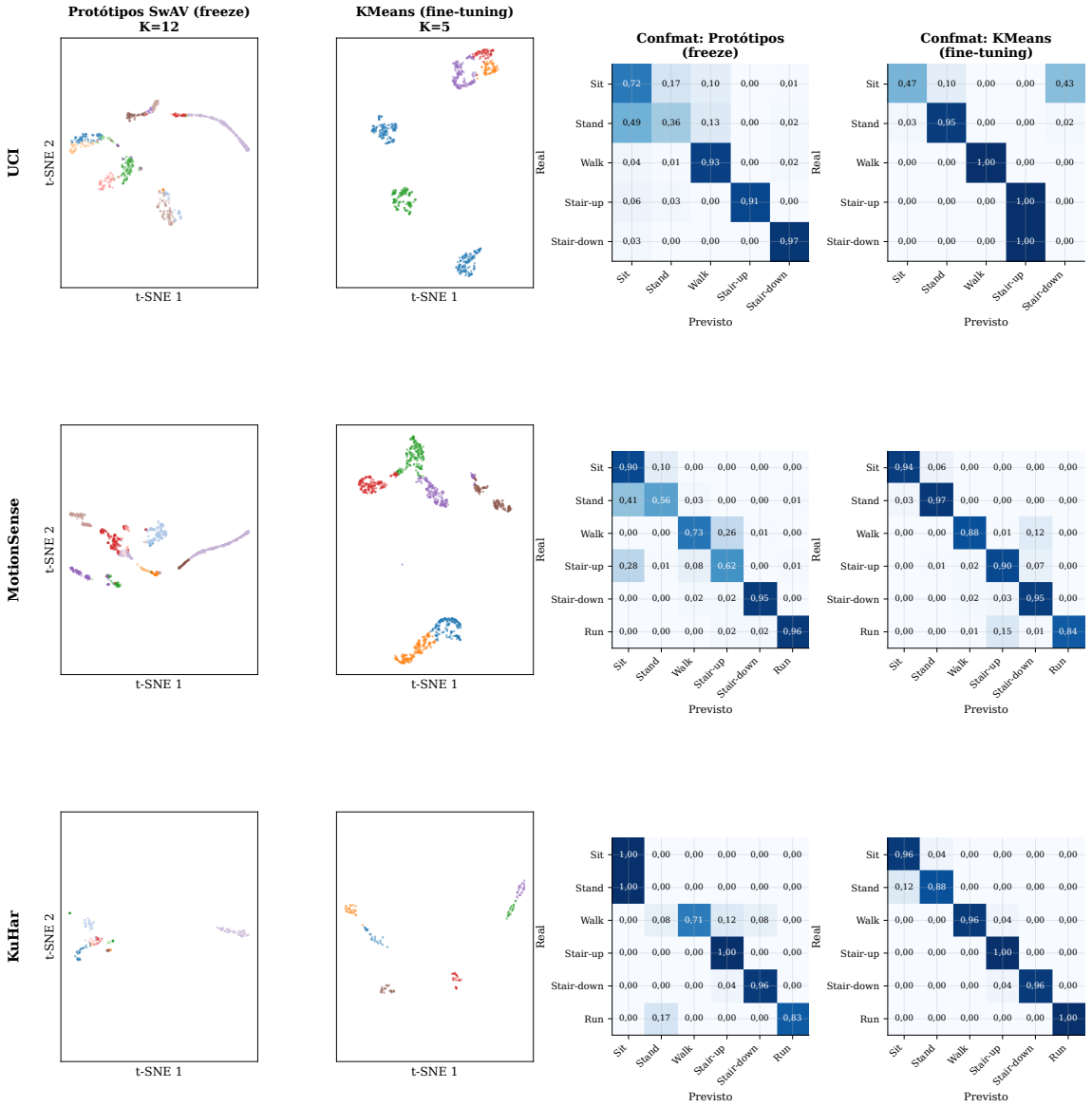

Figura 13: Resultados qualitativos para os conjuntos de dados UCI, MotionSense e KuHAR.

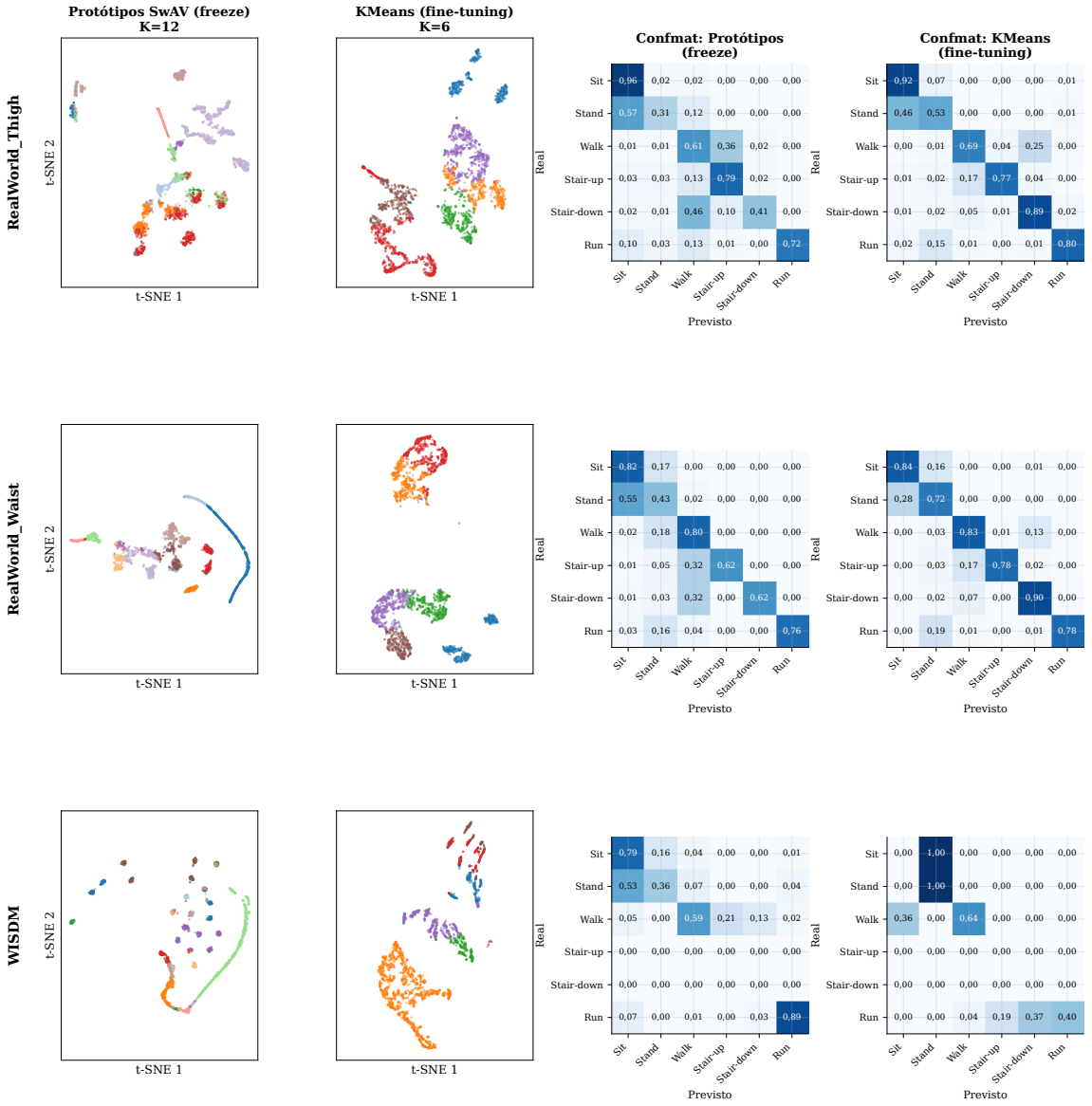

Figura 14: Resultados qualitativos para os conjuntos de dados RealWorld Thigh, RealWorld Waist e WISDM.

6 Conclusão

Este trabalho investigou a adaptação do SwAV, um modelo de aprendizado autossupervisionado baseado em agrupamento de protótipos, para o reconhecimento de atividades humanas a partir de sinais inerciais. Os experimentos conduzidos sobre sete conjuntos de dados públicos de HAR e o modelo foi submetido a três protocolos de avaliação *downstream: freeze, from scratch* e *full fine-tuning* nos cenários *in-domain* e *cross-domain* (LODO). Os resultados demonstraram que as representações aprendidas pelo SwAV-HAR atingem acurácia de classificação competitiva com modelos supervisionados e técnicas autossupervisionadas do estado da arte no modo *full fine-tuning* para parte dos conjuntos de dados. Já o desempenho com o protocolo de avaliação *freeze* é comparável ao obtido com técnicas supervisionadas em conjuntos de dados menores, com maior regularidade estrutural em regime de poucos rótulos, como UCI. O comportamento do modelo em função do percentual de rótulos evidencia a consistência da hipótese inicial de que as representações latentes simplificam a fronteira de decisão que o classificador deve aprender, tornando a métrica de acurácia competitiva com arquiteturas supervisionadas.

No aspecto da identificação das janelas de transição, conclui-se a partir das métricas e da inspeção visual que essas amostras ocupam uma região no espaço latente pouco delimitada e não se traduzem num protótipo dedicado. Considerando que o *backbone* foi exposto aos padrões sensoriais de poucas janelas de transição no conjunto de treino do HAPT, a falta de isolamento das classes indica que a técnica de agrupamento tende a favorecer os padrões dominantes das classes estáveis e mais consistentes sob os efeitos de aumento dos dados. Desse modo, a dificuldade de identificar as classes de transição evidencia a limitação do objetivo autossupervisionado diante das poucas amostras de transição postural.

Com o desenvolvimento do projeto SwAV-HAR, destacam-se três direções futuras: a adoção de técnicas de adaptação de domínio para o alinhamento de distribuições, visando superar os resultados LODO; a incorporação de modelos capazes de preservar a temporalidade no *backbone*, focando em detectar transições de forma robusta; e a extensão do protocolo de avaliação para cenários de personalização, de forma que o modelo seja adaptado a novos usuários. Finalmente, o trabalho sugere que o aprendizado autossupervisionado baseado em agrupamento é uma alternativa promissora para o paradigma supervisionado em HAR e com potencial para ser explorado em generalização de domínio e outras tarefas não-supervisionadas.

Referências

[1] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2021.

[2] J. Reyes-Ortiz, D. Anguita, L. Oneto, and X. Parra, "Smartphone-Based Recognition of Human Activities and Postural Transitions." UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C54G7M.

[3] O. Napoli, D. Duarte, P. Alves, D. H. P. Soto, H. E. de Oliveira, A. Rocha, L. Boccato, and E. Borin, "A benchmark for domain adaptation and generalization in smartphone-based human activity recognition," *Scientific Data*, vol. 11, p. 1192, Nov 2024.

[4] T. Plötz and Y. Guan, "Deep learning for human activity recognition in mobile computing," *Computer*, vol. 51, no. 5, pp. 50–59, 2018.

[5] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[6] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, "Deep learning for sensor-based human activity recognition: Overview, challenges and opportunities," *CoRR*, vol. abs/2001.07416, 2020.

[7] H. Haresamudram, I. Essa, and T. Plötz, "Assessing the state of self-supervised human activity recognition using wearables," 2022.

[8] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," 2019.

[9] A. Abedin, F. Motlagh, Q. Shi, S. H. Rezatofighi, and D. C. Ranasinghe, "Towards deep clustering of human activities from wearables," 2020.

[10] H. Amrani, D. Micucci, and P. Napoletano, "Unsupervised deep learning-based clustering for human activity recognition," in *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)*, pp. 1–6, 2022.

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[12] X. Zhang, Z. Zhao, T. Tsiligkaridis, and M. Zitnik, "Self-supervised contrastive pre-training for time series via time-frequency consistency," 2022.

[13] G. P. C. P. Da Luz, D. H. P. Soto, O. O. Napoli, A. Rocha, L. Boccato, and E. Borin, "Benchmarking encoders and self-supervised learning for smartphone-based human activity recognition," *IEEE Access*, vol. 14, pp. 37451–37475, 2026.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[15] P. Knopp and R. Sinkhorn, "Concerning nonnegative matrices and doubly stochastic matrices.," *Pacific Journal of Mathematics*, vol. 21, no. 2, pp. 343 – 348, 1967.

[16] Discovery UNICAMP, "Minerva: A PyTorch-Lightning-based framework for training machine learning models." GitHub repository, 2024.

[17] Scikit-learn Developers, "sklearn.manifold.tsne." scikit-learn documentation.

[18] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering – a decade review," *Information Systems*, vol. 53, pp. 16–38, 2015.

[19] L. Mahon and T. Lukasiewicz, "Efficient deep clustering of human activities and how to improve evaluation," in *Proceedings of The 14th Asian Conference on Machine Learning* (E. Khan and M. Gonen, eds.), vol. 189 of *Proceedings of Machine Learning Research*, pp. 722–737, PMLR, 12–14 Dec 2023.

[20] M. Woo, A. A. Harby, F. Zulkernine, and H. M. Abdulsalam, "A two-phase hybrid clustering framework exploring transitional activities in HAR," *Discover Artificial Intelligence*, vol. 5, no. 1, p. 233, 2025.

A Estudos de Ablação

A.1 Combinações de transformações de aumento

Avaliou-se o impacto das transformações de aumento para a arquitetura SwAV-HAR com os experimentos das transformações organizadas em 3 grupos de variantes: A - *add-one-in*, B - *leave-one-out* e C - novas transformações de aumento. A Tabela 10 apresenta a definição dos grupos de variantes de aumento dos dados. As transformações foram aplicadas a cada experimento nos conjuntos de dados UCI e HAPT, com $K = n_{classes}$, *backbone* ResNet-SE-5 e modo *freeze*. O *multi-crop* é fixado em todas as variantes e a configuração *baseline* consiste nas seguintes transformações: *gaussian noise* ($\sigma = 0{,}05$), permutação de segmentos (5 segmentos) e escalonamento de amplitude ($\sigma = 0{,}1$).

Tabela 10: Estudo de ablação - variantes de transformações de aumento.

Grupo	`Variante`	**Descrição (+ *multi-crop*)**
A – *add-one-in*	`only_crop`	Apenas *crop*.
	`only_noise`	Apenas *gaussian noise*.
	`only_scaling`	Apenas escalonamento de amplitude.
	`only_permut`	Apenas permutação de segmentos.
B – *leave-one-out*	`no_noise`	Baseline sem *gaussian noise*.
	`no_scaling`	Baseline sem escalonamento.
	`no_permutation`	Baseline sem permutação.
	`baseline`	*Gaussian noise* + *scaling* + permutação.
C – Novas transformações	`only_global_scaling`	Apenas escalonamento global.
	`only_axis_permut`	Apenas permutação de eixos.
	`baseline_global_scal`	Baseline + escalonamento global.
	`baseline_axis_permut`	Baseline + permutação de eixos.
	`full_aug`	Todas as transformações de aumento.

A partir do conjunto de gráficos da Figura 15, nota-se a existência de um *trade-off* entre poder discriminativo das *features* e coerência de agrupamento com padrões distintos nos dois conjuntos de dados. Em UCI, a configuração *baseline* tem a menor acurácia linear entre as variantes, $acc = 0{,}83$, mas lidera as métricas de agrupamento com *K-Means*. A análise da seção B *(leave-one-out)* mostra que remover a transformação `scaling` eleva a acurácia linear ao custo do colapso da acurácia de *K-Means*, com uma queda de $acc_{K\text{-}Means} = 0{,}74$ para $acc_{K\text{-}Means} = 0{,}56$.

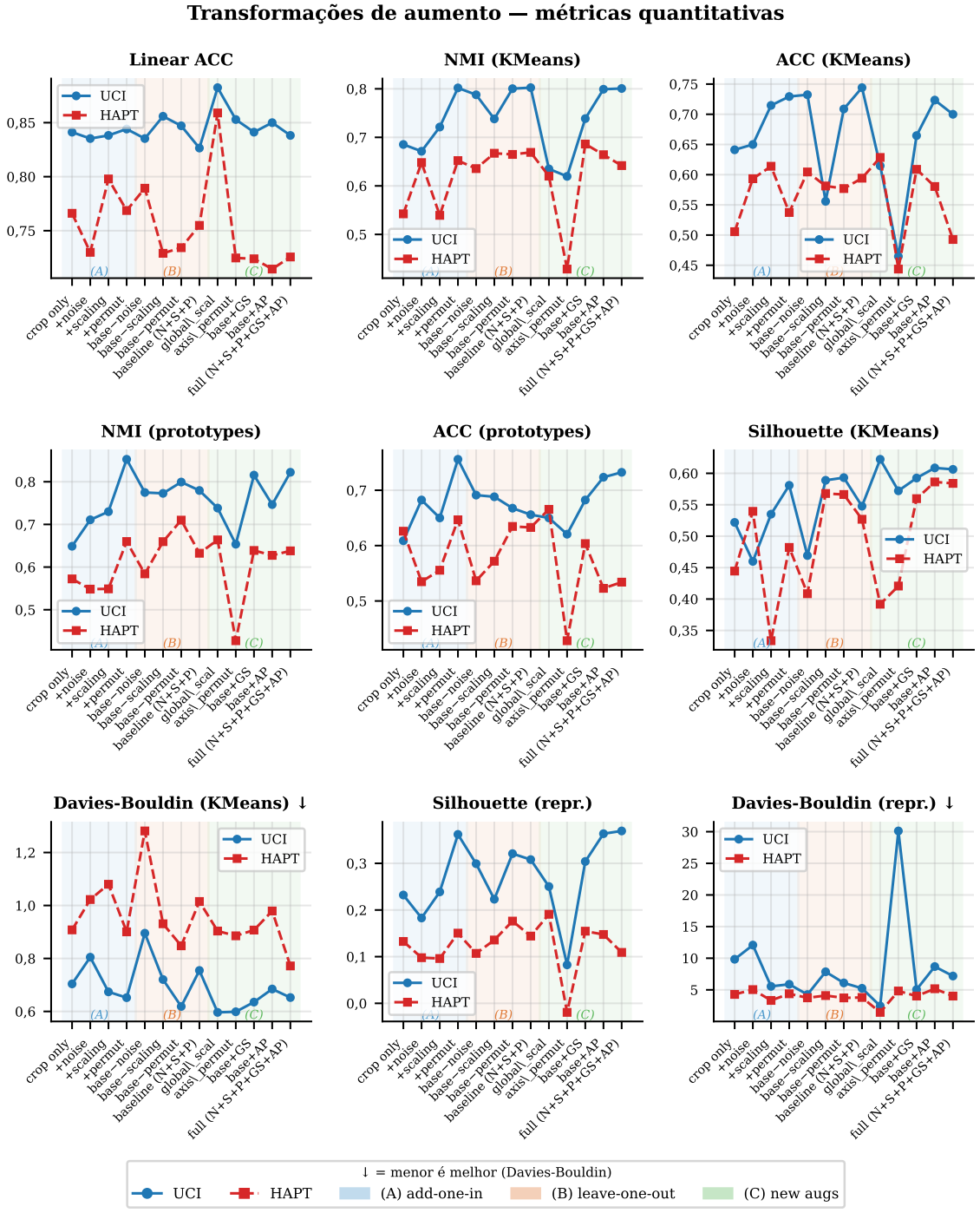

Figura 15: Ablação de transformações - comparação de métricas avaliativas.

Já para HAPT, o padrão é mais assimétrico, a remoção de escalonamento ($acc = 0{,}729$)

ou permutação ($acc = 0{,}734$) promove uma queda em relação à configuração *baseline* ($acc = 0{,}76$), enquanto remover o ruído gaussiano ($acc = 0{,}79$) provoca uma melhora de acurácia, indicando que o escalonamento e a permutação ajudam a capturar a variação estrutural das atividades. Por último, para os dois conjuntos de dados, a configuração `only global scaling` apresenta a melhor acurácia linear, com $acc = 0{,}88$ para o UCI e $acc = 0{,}86$ para o HAPT, embora apresente redução na métrica NMI(*K-Means*) em relação ao *baseline*, mais acentuada para o UCI. Esse resultado sugere que a invariância de magnitude maximiza a separabilidade e produz grupos geometricamente compactos ($silhouette_{K\text{-}Means} = 0{,}62$), mas, devido à queda de NMI (*K-Means*), os agrupamentos têm menor alinhamento com as classes de atividade.

Com base nos resultados, foram escolhidas três variantes de transformações para a ablação do número de protótipos: `baseline`, `baseline global scaling` e `full aug`. A configuração `baseline` serve como referência para as métricas de agrupamento e para avaliar a interação com representações de estruturas densas. Já `baseline global scaling` apresenta o melhor agrupamento no HAPT ($NMI_{Kmeans} = 0{,}69$, $acc_{K\text{-}Means} = 0{,}61$) e acurácia linear inferior, tornando-se mais informativo para medir como a invariância de magnitude interage com os efeitos de K. Por fim, `full aug` se justifica por ser um limite superior de intensidade de aumento dos dados, embora seus resultados revelem que a permutação dos eixos prejudica a organização dos protótipos, mesmo com representações de *backbone* geometricamente mais coesas para o UCI ($silhouette_{repr} = 0{,}37$).

A.2 Número de protótipos (K)

Como detalhado em seções anteriores, no modelo SwAV-HAR os K protótipos são aprendidos com o *backbone* através de *backpropagation*. No SwAV para imagens, os autores identificaram que o método é robusto a K quando existem protótipos suficientes. Como os conjuntos de dados selecionados têm quantidades menores de classes, 6 para os conjuntos de dados de DAGHAR e 7 classes para o HAPT, o modelo é mais sensível ao impacto de K, o que justifica o estudo de ablação com diferentes variações de K e combinações de transformações.

Desse modo, os experimentos foram conduzidos no HAPT, com os conjuntos de transformações `baseline`, `full_aug` e `baseline_global_scaling` sob o protocolo *freeze*, em duas rodadas:

- **Rodada 1**: $K \in \{7, 12, 30, 60, 150, 300\}$, 10 000 *steps* de pré-treino e 25 000 de avaliação.
- **Rodada 2**: $K \in \{12, 30, 60\}$, 30 000 *steps* de pré-treino, com busca de taxa de aprendizado (LR) automática.
- $K = n_{classes} = 7$ representa a referência para HAPT.

A Figura 16 reúne as métricas quantitativas extraídas durante as rodadas R1 (tracejado) e R2 (sólido). Como nenhuma configuração dominou consistentemente em todas as métricas individualmente, adotou-se um ranqueamento global, em que cada configuração recebeu um

rank por métrica e a média dos *ranks* constitui a pontuação final, apresentada na Figura 17.

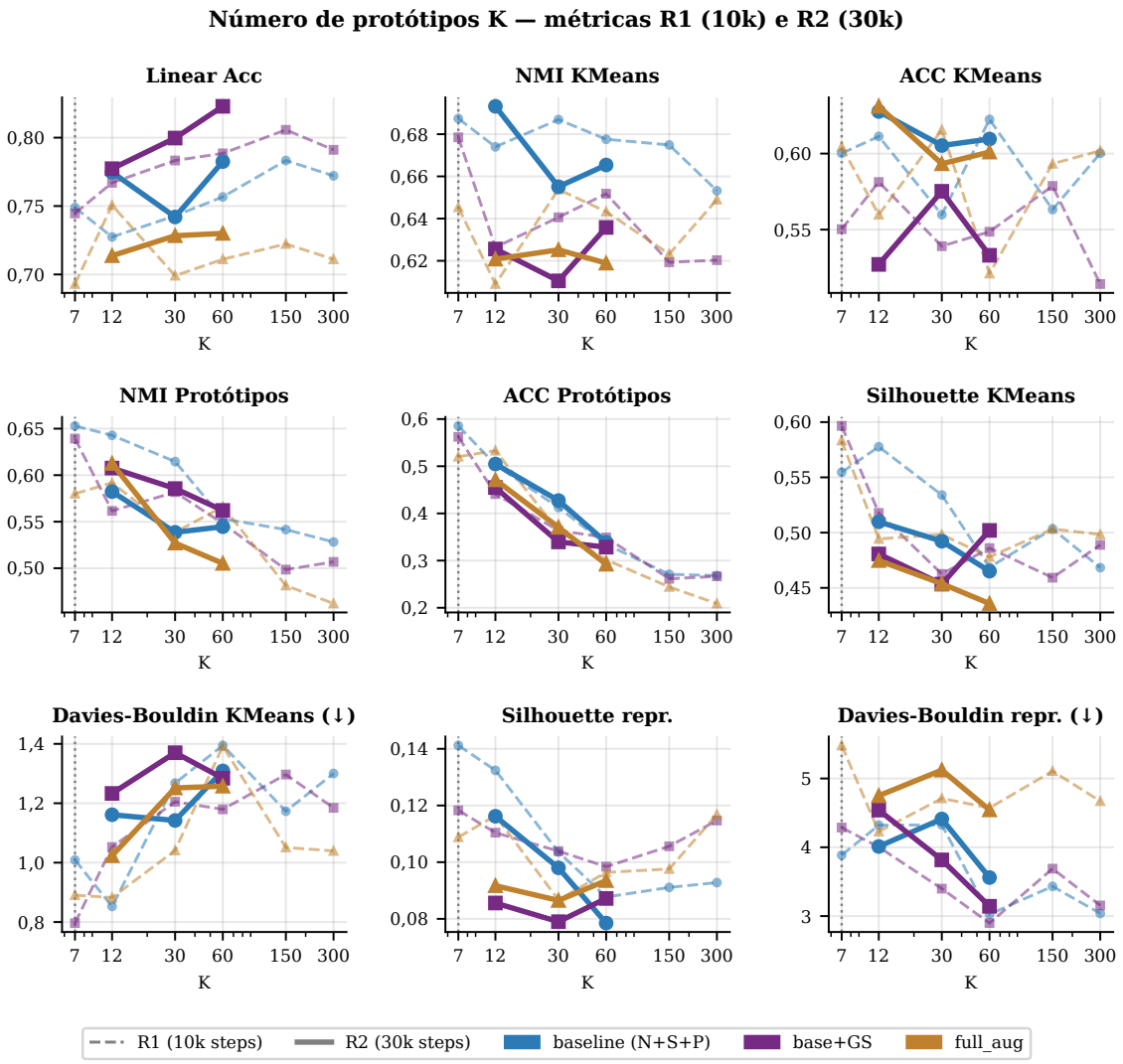

Figura 16: Ablação de K + transformações - comparação de métricas avaliativas.

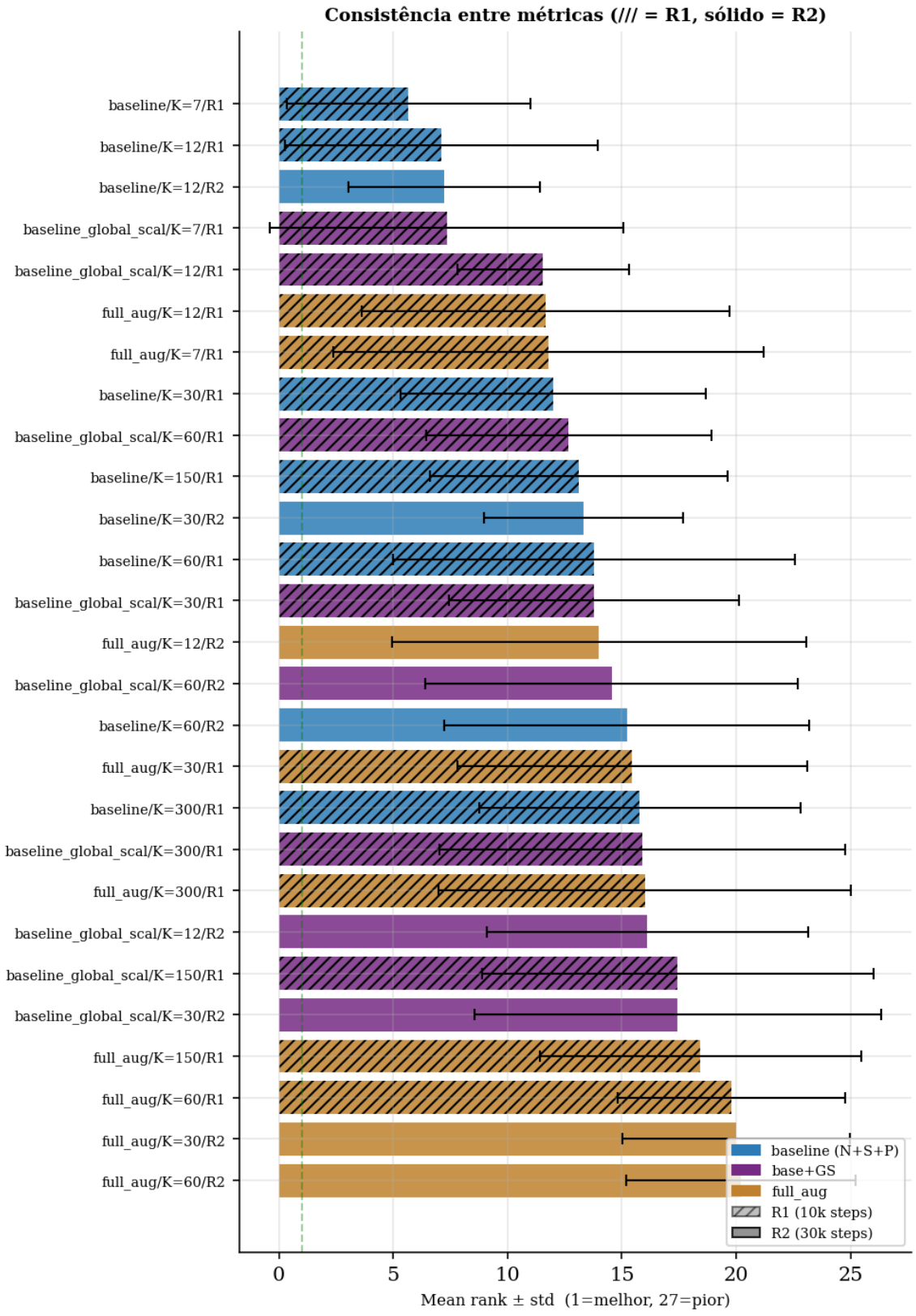

Figura 17: Ablação de K + transformações - *ranking* de experimentos.

O procedimento de ranqueamento evidenciou que `baseline`/$K = 12$ na rodada R2 ocupa posição competitiva com menor variação e pontuação média encontrada entre os experimentos, em contraste com as configurações de maior K, que maximizaram a acurácia linear, mas deterioraram as métricas de agrupamento. Portanto, a escolha de $K = 12$ para a arquitetura do modelo SwAV-HAR reflete a proporção de $2 \times n_{classes}$ para os conjuntos de dados de DAGHAR, promovendo o poder discriminativo das classes sem a fragmentação do espaço de *features* na configuração *baseline*.

A.3 Busca de hiperparâmetros

A busca de hiperparâmetros para a arquitetura SwAV-HAR foi organizada em 3 etapas, uma para cada protocolo de treinamento. Antes da primeira etapa, o *split* de treino foi utilizado para construir um novo conjunto de validação, com 20% dos usuários. No HAPT, a métrica alvo de acurácia foi substituída por acurácia balanceada devido ao desbalanceamento entre janelas de atividades. Para o modo *from scratch*, a otimização se concentrou nos hiperparâmetros de taxa de aprendizado, $lr \in [1 \times 10^{-4}, 1 \times 10^{-2}]$, *weight decay* em $[1 \times 10^{-6}, 1 \times 10^{-3}]$, épocas em $[50, 200]$ e tamanho do mini *batch* em $\{64, 128, 256\}$. Assim, a primeira abordagem foi uma busca aleatória com 30 tentativas, seguida do refinamento bayesiano via Optuna concentrado nas melhores regiões. Nesse aspecto, a busca completa foi realizada nos conjuntos UCI, HAPT e RealWorld Thigh, enquanto os hiperparâmetros dos demais conjuntos de dados foram obtidos pela média de taxas de aprendizado e de tamanhos dos lotes dos três conjuntos otimizados. Já no modo *freeze*, a busca se restringiu à taxa de aprendizado da cabeça linear com *grid search* sobre valores em escala logarítmica e 5 000 *steps*. Os resultados são exibidos na Figura 18.

O modo *full fine-tuning* apresentou o maior espaço de busca, com cinco hiperparâmetros por conjunto de dados: taxa de aprendizado da cabeça (lr), taxa de aprendizado do *backbone* (`lr_backbone`), *weight decay*, épocas e tamanho de mini-lote, otimizados por Optuna com 20 tentativas para cada um dos conjuntos de dados. Além disso, a convergência foi acelerada com a definição dos intervalos de taxa de aprendizado nos valores ótimos encontrados na etapa *from scratch*, com a taxa de aprendizado restrita a $[0{,}3 \times lr_{scratch}, 3 \times lr_{scratch}]$, enquanto `lr_backbone` permaneceu no intervalo $[0{,}01 \times lr_{scratch}, 0{,}3 \times lr_{scratch}]$. Dessa forma, o *backbone* é atualizado com taxa menor do que a taxa empregada na cabeça. A Figura 19 exibe as melhores taxas de aprendizado encontradas em cada conjunto de dados.

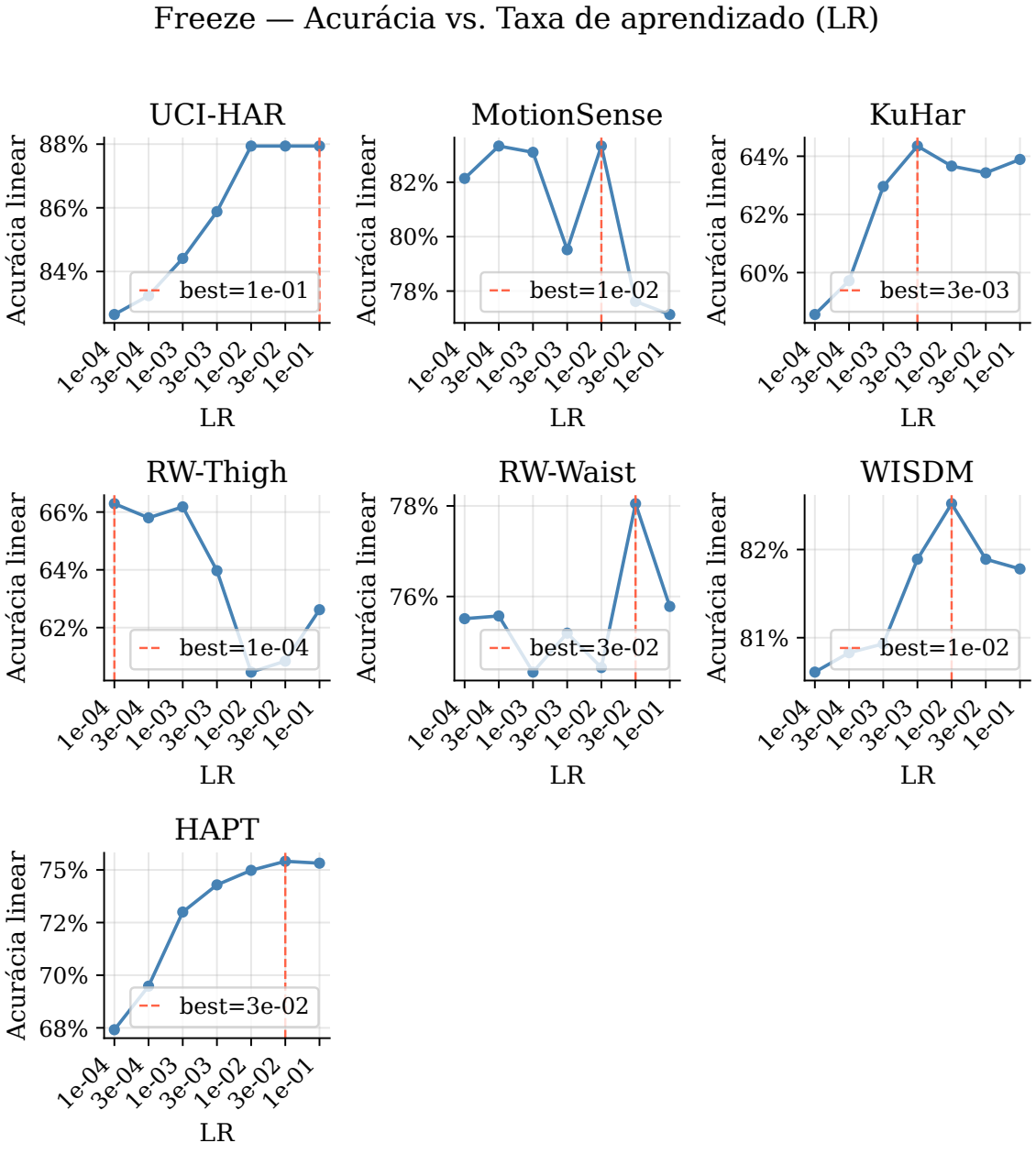

Figura 18: Busca de hiperparâmetros - acurácia em função da taxa de aprendizado (LR).

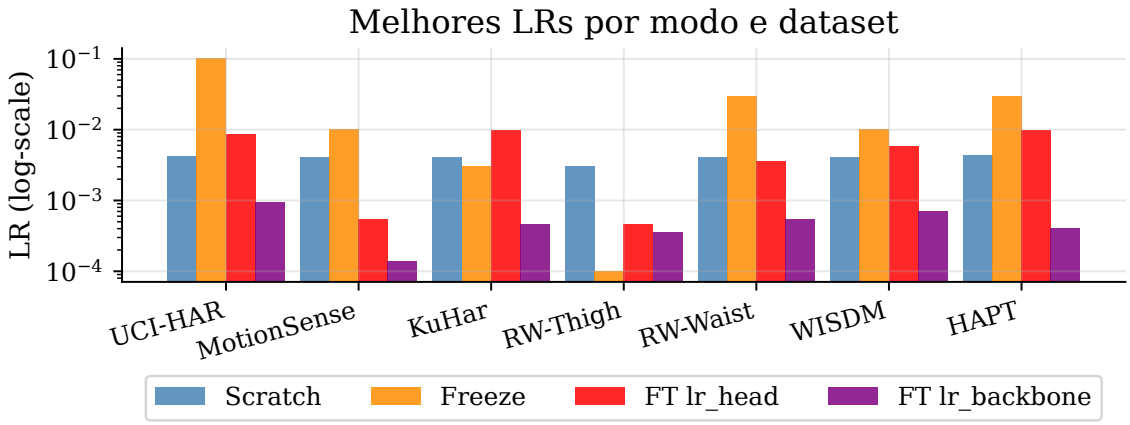

Figura 19: Busca de hiperparâmetros - Taxa de aprendizado (LR) em função do conjunto de dados.

A.4 Figuras adicionais

A.4.1 Composição de classes por conjunto de dados e número de amostras

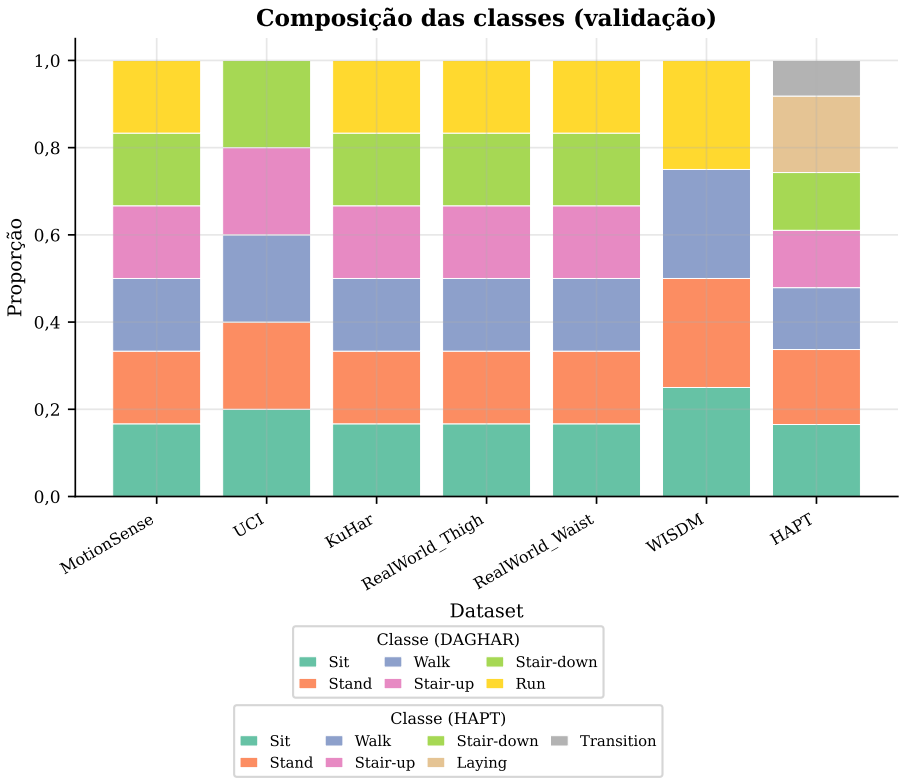

Figura 20: Composição das classes por conjunto de dados (validação).

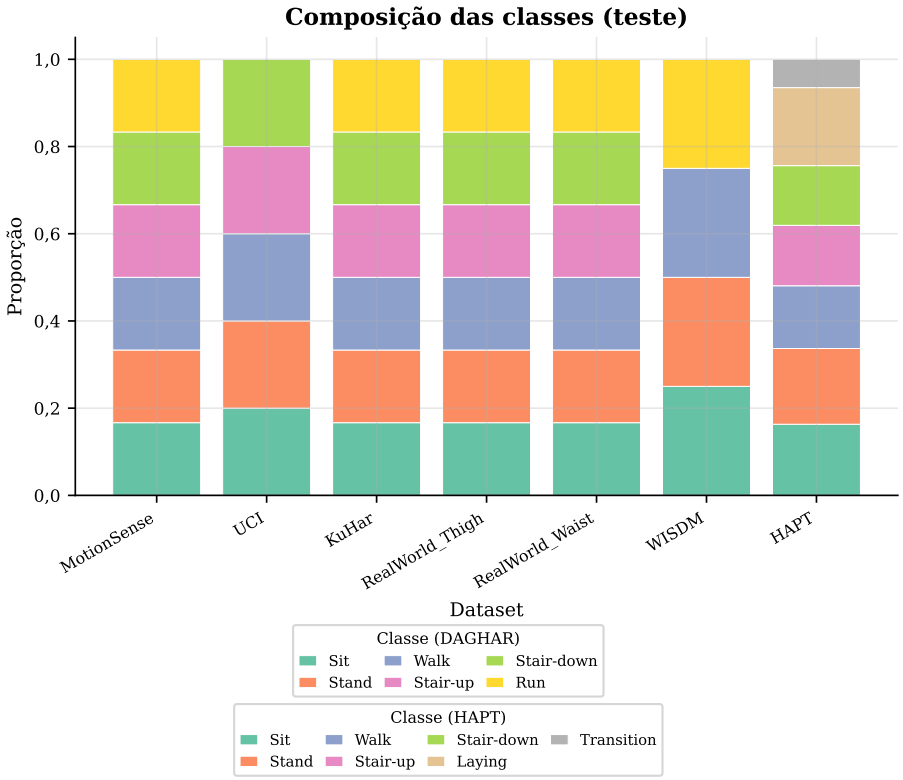

Figura 21: Composição das classes por conjunto de dados (teste).

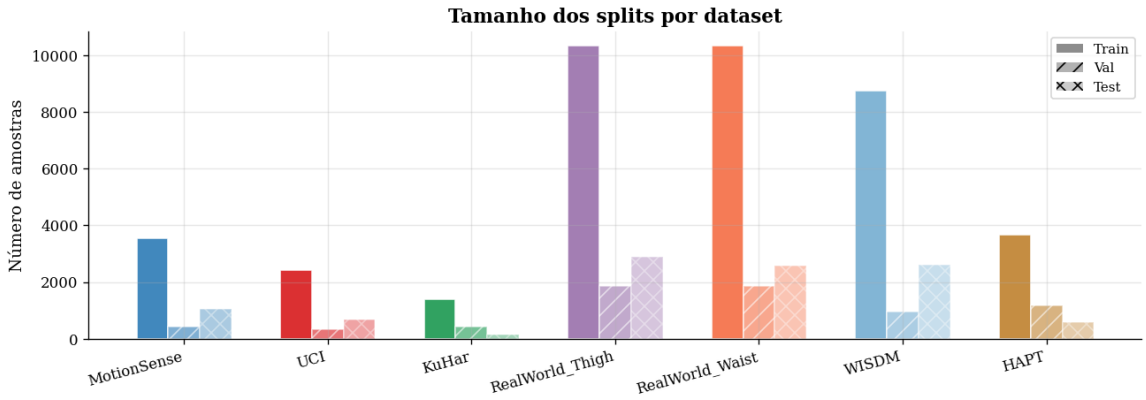

Figura 22: Tamanho dos subconjuntos de treino, validação e teste por conjunto de dados.

A.5 Algoritmo de Sinkhorn-Knopp

Código 1: Algoritmo de Sinkhorn-Knopp para SwAV-HAR.

```
def distributed_sinkhorn(out, epsilon, n_iters, world_size=1):
    # float64 evita overflow para epsilon pequeno.
    Q = torch.exp(out.double() / epsilon).t()  # [K, B]
    B = Q.shape[1] * world_size
    K = Q.shape[0]
    sum_Q = torch.sum(Q)
    if world_size > 1:
        dist.all_reduce(sum_Q)
    Q /= sum_Q
    for _ in range(n_iters):
        # linha: cada prototipo recebe peso igual no batch.
        sum_rows = torch.sum(Q, dim=1, keepdim=True)
        if world_size > 1:
            dist.all_reduce(sum_rows)
        Q /= sum_rows * K
        # coluna: cada amostra distribui peso entre prototipos.
        Q /= torch.sum(Q, dim=0, keepdim=True) * B
    Q *= B
    return Q.t().to(out.dtype)  # [B, K]
```