



**SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY**

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Mathematics

**Hierarchical Pseudo-Labeling in
Hyperbolic Space for Skeleton-Based
Human Action Recognition**

Alexander Schwer





SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Mathematics

Hierarchical Pseudo-Labeling in Hyperbolic Space for Skeleton-Based Human Action Recognition

Author	Alexander Schwer
Examiner	Prof. Dr. Nassir Navab Chair for Computer Aided Medical Procedures & AR
Supervisors	Mert Kiray Chair for Computer Aided Medical Procedures & AR Prof. Dr. rer. nat. Benjamin Busam Chair for Photogrammetry & Remote Sensing
Submission Date	October 15, 2026

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, October 15, 2026

Alexander Schwer

Abstract

In this thesis, we address skeleton-based recognition of human actions while taking into account the hierarchical structure inherent to this data. We build upon HypSkeletonCLR and its variants, utilizing the established combination of exponential mapping to the Poincaré ball and a contrastive loss based on the negative hyperbolic distance. We extend this approach with an online clustering method and additional objective functions to strengthen the hierarchical structure of the skeleton sequence representation. These additional objective functions are either coupled or added as separate terms, with the cluster assignments providing additional pseudo-labels for the skeleton sequence representation. We show that cluster-based training can improve linear classification accuracy, although the baseline model without clustering achieves comparable agreement with the reference hierarchy. We also demonstrate how the choice of hyperparameters, such as the number of clusters, can influence linear classification accuracy.

Zusammenfassung

In dieser Arbeit befassen wir uns mit der skelettbasierten Erkennung menschlicher Handlungen unter Berücksichtigung der diesen Daten innewohnenden hierarchischen Struktur. Wir bauen auf HypSkeletonCLR und dessen Varianten auf und nutzen dabei die etablierte Kombination aus Exponentialabbildung in die Poincaré- Kugel und einer Kontrast-Fehlerfunktion, die auf dem negativen hyperbolischen Abstand basiert. Wir erweitern diesen Ansatz um ein Online-Clustering-Verfahren und zusätzliche Zielfunktionen, um die hierarchische Struktur der Repräsentation der Skelettsequenzen zu stärken. Diese zusätzlichen Zielfunktionen werden entweder gekoppelt oder als separate Terme hinzugefügt, wobei die Clusterzuordnungen zusätzliche Pseudo-Labels für die Repräsentation der Skelettsequenzen liefern. Wir zeigen, dass das Cluster-basierte Training die lineare Klassifikationsgenauigkeit verbessern kann, jedoch das Referenzmodell ohne Clustering vergleichbare Übereinstimmung mit der Referenzhierarchie erzielt. Wir zeigen außerdem, wie die Wahl der Hyperparameter, wie beispielsweise die Anzahl der Cluster, die lineare Klassifikationsgenauigkeit beeinflussen kann.

Contents

Abstract	IV
Zusammenfassung	V
1. Introduction	1
1.1. Motivation	1
1.2. Problem Setting	2
1.3. Research Questions	3
1.4. Contributions	4
1.5. Thesis Structure	5
2. Theoretical Background	6
2.1. Skeleton-Based Human Action Recognition	6
2.1.1. Input Representations	7
2.1.2. Spatio-Temporal Graphs	8
2.1.3. Spatio-Temporal Graph in ST-GCN	8
2.1.4. Classification Task	9
2.2. Contrastive Learning	10
2.2.1. General Principle	10
2.2.2. InfoNCE Objective	10
2.2.3. Momentum Contrast in SkeletonCLR	11
2.2.4. Extended Positive Sets and False Negatives	11
2.3. Hyperbolic Geometry and the Poincaré Ball	12
2.4. Pseudo-Labeling and Clustering	14
2.5. t-SNE and its limitations	14
3. Related Work	17
3.1. Skeleton-Based Action Recognition	17
3.2. Contrastive Learning on Skeleton Sequences	17
3.3. Hyperbolic Representation Learning	18
3.4. Hierarchy-Aware and Pseudo-Label-Based Objectives	18

4. Methodology	20
4.1. Method Overview	20
4.2. Base Representation Pipeline	21
4.3. Hierarchy-Aware Contrastive Learning	22
4.3.1. Balanced Prototype Assignment	23
4.3.2. Hyperbolic Hierarchy Objective	25
4.4. Cluster-Based Pseudo-Labeling	27
4.5. Overall Training Objective	28
5. Experimental Setup	30
5.1. Dataset	30
5.2. Implementation Details	30
5.3. Training Protocol	31
5.4. Baselines and Ablations	32
5.5. Evaluation Metrics	33
6. Results	35
6.1. Ablation Study of the Prototype Count K	35
6.2. Embedding Quality by Classification Performance	36
6.2.1. Component Ablation: Joint-Only Inputs	37
6.2.2. Component Ablation: Three-Stream Inputs	38
6.2.3. Evaluation on Unprocessed 300-Frame Inputs	39
6.3. Hierarchical Structure in Embeddings	40
6.3.1. Metrics	40
6.3.2. Projections	41
7. Discussion	43
7.1. Interpretation of Results	43
7.2. Limitations	44
8. Conclusion	45
8.1. Summary	45
8.2. Future Work	45
A. Supplementary Embedding Analysis	46
A.1. Reference Structure and Prototype Assignments	46
A.2. Hyperbolic t-SNE	48
A.3. Triplet Agreement and Permutation Control	50
A.4. t-SNE Definition	52

B. Riemannian Geometry and the Exponential Map	53
B.1. Riemannian Manifolds and Their Exponential Maps	53
B.2. Derivation for the Poincaré Ball	54
B.2.1. The Ball as a Riemannian Manifold	54
B.2.2. Geodesic Equation	55
B.2.3. Exponential Map at the Origin	55
B.2.4. Consequences of Unit Normalization	57
Abbreviations	58
List of Figures	59
List of Tables	60
Bibliography	61

1. Introduction

1.1. Motivation

Human action recognition is a central problem in computer vision because many applications require an automatic understanding of motion, interaction, and intent. In contrast to RGB video, skeleton sequences describe the body by a small collection of joint coordinates over time. This representation removes much of the appearance, background, and illumination variability of raw video while retaining the spatial and temporal structure that is directly relevant for many actions. Large skeleton benchmarks such as NTU RGB+D have therefore become standard test beds for studying action recognition from structured motion data [1].

The structure of skeleton data is not only temporal but also relational. Joints form a kinematic graph, and actions are often characterized by coordinated changes of body parts across frames. The learning problem is to map a sequence of joint positions to numerical features, called representations, from which an action, such as waving, can be recognized. Spatial-temporal graph convolutional networks exploit this structure explicitly and have shown that graph-based modeling is effective for skeleton-based recognition [2]. At the same time, strong fully supervised models depend on action labels for every training sequence. This requirement limits scalability, especially when new action categories, new camera setups, or domain-specific datasets have to be annotated.

Self-supervised and contrastive learning address this limitation by learning representations from data transformations and consistency constraints rather than relying exclusively on manual labels [3]. For skeleton sequences, methods such as CrosSCLR adapt this idea to multiple skeleton views and learn action representations before downstream evaluation [4]. However, the usual evaluation target remains flat classification accuracy: an action is either predicted correctly or incorrectly, and the geometry of the learned embedding is rarely judged by whether it reflects semantic relations among actions. Actions such as “wear jacket” and “take off jacket”, or “kicking” and “punching”, are not independent points in a label set. They belong to broader groups of motion patterns and interactions. Such organization could support grouping actions at different levels of specificity and make relations between their learned representations easier to interpret. For illustration, actions could branch into dressing actions, then jacket-related actions, with “wear jacket” and “take off jacket” as leaves.

This motivates the question of whether representation learning for skeleton-based action recognition can benefit from an embedding space and a training signal that make such latent structure more explicit. Hyperbolic spaces are attractive in this context because they can represent tree-like and hierarchical relations with low distortion compared to Euclidean spaces [5, 6]. A direct predecessor of this thesis introduced HypSkeletonCLR by embedding SkeletonCLR representations in the Poincare ball and replacing Euclidean contrastive similarities with the negatives of hyperbolic distances [7]. That work showed that hyperbolic contrastive learning can be integrated into single-view and multi-view skeleton representation learning. The present thesis investigates the training signals needed to exploit this representational capacity. Mapping embeddings to hyperbolic space does not by itself guarantee that they encode a semantically meaningful action hierarchy. HypSkeletonCLR’s instance-level contrastive objective contains no explicit constraint on hierarchical relations between groups of actions. The proposed extension groups representations around learned group representatives (prototypes), encourages hierarchical relations between these groups, and optionally uses the group assignments as additional training signals. Its central hypothesis is that explicitly encouraging relations between prototypes can improve the organization of action representations. Whether this also improves classification, and whether the resulting organization agrees with interpretable action relations, are separate empirical questions.

1.2. Problem Setting

To investigate this hypothesis, the thesis studies representation learning for skeleton-based human action recognition on the NTU RGB+D setting [1]. Each input sample is a sequence of three-dimensional human joint coordinates. The task is to learn an encoder that maps such sequences to a feature representation from which action classes can be recognized under standard evaluation splits, in particular cross-subject and cross-view evaluation. The approach follows the SkeletonCLR/CrosSCLR line of work and uses an ST-GCN backbone to encode 50-frame skeleton sequences [2, 4]. More specifically, it takes HypSkeletonCLR as the immediate baseline: the prior work adapted the SkeletonCLR training objective to the Poincare ball, considered joint-only and three-stream skeleton inputs, and evaluated both self-supervised and supervised contrastive variants [7].

The baseline training objective is contrastive: two augmented views of a sequence should have similar representations, while representations of different sequences are separated. In supervised or semi-supervised variants, label information can additionally define positive pairs, following the idea of supervised contrastive learning [8]. The downstream quality of the representation is then evaluated by training a classifier on top of the learned encoder and measuring recognition performance.

The central extension considered in this thesis is the use of cluster-based pseudo-labels on top of the hyperbolic SkeletonCLR variants. Instead of relying only on whether two views originate from the same sequence or share an available action label, the model forms clusters in the learned representation space and uses the resulting assignments as additional training signals. This is related to contrastive methods that combine representation learning with prototype or cluster assignments [9]. The three additional training signals have distinct roles: balanced assignments define intermediate groups, the prototype-level hierarchy objective regularizes relations between those groups, and pseudo-label feedback introduces additional positives into contrastive learning. A flat cluster assignment alone does not specify a hierarchy. The contribution is the integration of balanced prototype learning, an adapted hierarchy-ranking objective, and cluster-based contrastive feedback into HypSkeletonCLR, together with a component-wise analysis of their effects.

This setting introduces two coupled difficulties. First, pseudo-labels are not ground truth and can amplify errors if clusters do not correspond to meaningful action structure. Second, a better-looking embedding geometry does not automatically imply better action recognition. The study therefore examines whether clustered hyperbolic training improves downstream classification and whether the learned distances preserve interpretable relations among action classes.

1.3. Research Questions

These challenges lead to three research questions, corresponding to the three parts of Chapter 6:

- RQ1** How does the number of prototypes K affect linear classification performance, and do HypSkeletonCLR-H and HypSkeletonCLR-HPL favor the same value among the tested cluster counts?
- RQ2** How do balanced clustering, prototype-level hierarchy learning, and pseudo-label feedback affect representation quality as measured by linear classification accuracy relative to HypSkeletonCLR? How do these effects vary across cross-subject and cross-view evaluation, joint-only and three-stream inputs, and the 300-frame input setting?
- RQ3** Do the proposed training signals produce embeddings whose hierarchical organization agrees more closely with action relations encoded by a supervised reference tree than the non-clustered hyperbolic baseline? What do metrics in the original embedding space and complementary projections reveal about this organization?

The reference tree groups action classes by Ward clustering of class prototypes from a supervised HypSkeletonCLR model; it is a learned reference rather than an official NTU60 hierarchy.

These questions distinguish hyperparameter selection, class separability, and hierarchical organization. A higher linear classification accuracy supports a claim about recognition performance under the evaluation protocol; it does not establish that semantic hierarchy has been learned. Likewise, visually separated groups in a two-dimensional projection are insufficient to establish hierarchical structure. The third question therefore requires evidence beyond the classification experiments.

1.4. Contributions

The thesis adapts established prototype-learning and hierarchy-learning ideas to HypSkeletonCLR through three extension components, evaluated in several combinations:

1. **Balanced prototype assignments.** The variant **HypSkeletonCLR-C** adds a clustering objective based on balanced Sinkhorn assignments to hyperbolic prototypes. It provides a comparison for separating the effect of clustering from that of hierarchy learning.
2. **Prototype-level hierarchy learning.** The variant **HypSkeletonCLR-H** adds a hierarchy objective to the clustering component. This objective explicitly regularizes relations between prototypes while retaining the baseline representation geometry.
3. **Cluster-based contrastive feedback.** The variant **HypSkeletonCLR-PL** uses cluster assignments to define additional pseudo-positive pairs without ground-truth action labels during pre-training. **HypSkeletonCLR-HPL** combines this feedback with balanced assignments and the hierarchy objective.

The extensions apply to both joint-only inputs and the inherited three-stream setting, giving the corresponding 3s-HypSkeletonCLR variants. The empirical study first varies K , then compares classification performance across the component variants. A separate hierarchy evaluation is needed to establish whether the explicit hierarchy objective produces semantically meaningful structure. Improved semantic hierarchy is an empirical hypothesis, not an assumed consequence of the model design.

An additional analytical contribution is identifying a geometric constraint of the inherited HypSkeletonCLR pipeline: unit normalization fixes sample radii, making pairwise sample distances depend only on angular separation (Proposition B.2.1).

1.5. Thesis Structure

This thesis is structured into eight chapters. After the introduction, Chapter 2 introduces the central concepts used in this work, including skeleton-based human action recognition, contrastive learning, hyperbolic geometry, pseudo-labeling, clustering, and visualization methods. Chapter 3 reviews the research on which this thesis builds, with a focus on skeleton-based representation learning, hyperbolic embeddings, and hierarchy-aware pseudo-labeling objectives.

Chapter 4 describes the proposed method. It first summarizes the inherited HypSkeletonCLR pipeline and then introduces balanced prototype assignment, hierarchy-aware contrastive learning, cluster-based pseudo-labeling, and the resulting training objective. Chapter 5 presents the dataset, implementation details, training protocol, baselines, ablations, and evaluation metrics. Chapter 6 is organized into three parts: an ablation study of the prototype count K , an assessment of embedding quality through classification performance, and an examination of hierarchical structure. Appendix A provides supplementary visualizations and metric details.

Chapter 7 interprets the experimental findings, explains what the component comparisons show, and discusses limitations and practical implications. Finally, Chapter 8 summarizes the thesis and gives directions for future work.

2. Theoretical Background

The research questions require both a model of skeleton data and a way to assess learned representations. This chapter introduces these foundations, starting with the input structure and progressing to contrastive learning, hyperbolic geometry, and the analysis of learned structure.

2.1. Skeleton-Based Human Action Recognition

Skeleton-based human action recognition represents an action by a sequence of body joint positions over time. Each frame describes a pose through joint coordinates. Their changes across frames describe the movement. Figure 2.1 illustrates such a sequence and the spatial and temporal connections used by ST-GCN.

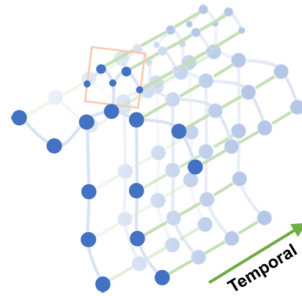


Figure 2.1.: Skeleton sequence with joints (blue dots), anatomical edges (blue), and edges between the same joint in consecutive frames (green). Source: [2, Fig. 1].

Let

$$\mathcal{T} = \{1, \dots, T\}, \quad \mathcal{J} = \{1, \dots, V\},$$

denote the temporal index set and the set of anatomical joints, respectively. If a sample contains up to M tracked persons, let

$$\mathcal{P} = \{1, \dots, M\}$$

denote the corresponding person indices. This allows the representation to include multi-person interactions, such as handshaking.

The following subsections first describe the representations derived from the joint coordinates and then introduce the spatio-temporal graph structure used to process them, as well as the action classification task.

2.1.1. Input Representations

Joint Positions. For each person $m \in \mathcal{P}$, time point $t \in \mathcal{T}$, and joint $j \in \mathcal{J}$, let

$$x_{t,j}^{(m)} \in \mathbb{R}^C$$

denote the coordinate vector. The joint representation of a skeleton sequence, stored as an indexed tensor, is therefore

$$X^{\text{joint}} = \left(x_{t,j}^{(m)} \right)_{(t,j,m) \in \mathcal{T} \times \mathcal{J} \times \mathcal{P}} \in \mathbb{R}^{C \times T \times V \times M}.$$

For the NTU RGB+D dataset, $C = 3$ and $V = 25$ [1].

Spatial Graph Topology. Besides the coordinate values, the skeleton representation also provides a fixed anatomical structure describing which joints are connected in the human body. For NTU RGB+D, this spatial structure is determined by the underlying 25-joint skeleton layout [1]. In graph-based models such as ST-GCN, it forms the spatial part of the graph input. In contrast, the way in which information is connected and aggregated across time is a modeling choice. The particular spatio-temporal graph construction used by ST-GCN is introduced in Section 2.1.3.

Motion and Bones. In addition to the joint coordinates, two representations are derived from X^{joint} : motion and bone [4]. The motion representation describes the temporal displacement of each joint,

$$x_{t,j}^{\text{motion},(m)} = \begin{cases} x_{t+1,j}^{(m)} - x_{t,j}^{(m)}, & t \in \{1, \dots, T-1\}, \\ 0, & t = T. \end{cases}$$

Thus, motion explicitly represents changes of the joint positions between consecutive time points.

For the bone representation, the anatomical skeleton forms a tree. Choose a root joint r , and let

$$\pi : \mathcal{J} \setminus \{r\} \rightarrow \mathcal{J}$$

assign each non-root joint j its unique parent $\pi(j)$, namely the adjacent joint on the unique path from j to r . The corresponding bone vector is

$$x_{t,j}^{\text{bone},(m)} = \begin{cases} x_{t,j}^{(m)} - x_{t,\pi(j)}^{(m)}, & j \neq r, \\ 0, & j = r. \end{cases}$$

It describes the relative position and orientation of adjacent joints. For NTU RGB+D, $r = 21$ in one-based joint numbering (array index 20).

Motion and bone are derived from joint coordinates using temporal ordering and anatomical connectivity, respectively. They expose temporal changes and local skeletal structure without additional measurements.

2.1.2. Spatio-Temporal Graphs

The input representations specify what is measured at each joint. To describe relations across joints and time, a skeleton sequence can be represented by a spatio-temporal graph

$$G_{ST} = (\mathcal{V}_{ST}, \mathcal{E}_{ST}), \quad \mathcal{V}_{ST} = \mathcal{T} \times \mathcal{J}.$$

Each vertex $(t, j) \in \mathcal{V}_{ST}$ represents joint j at time point t .

In general, the edge set

$$\mathcal{E}_{ST} \subseteq \{\{(t, i), (s, j)\} : (t, i), (s, j) \in \mathcal{V}_{ST}, (t, i) \neq (s, j)\}$$

may contain arbitrary connections between joints and time points. No particular spatial topology, temporal range, or correspondence between joints is imposed by this definition.

2.1.3. Spatio-Temporal Graph in ST-GCN

The general graph definition leaves the choice of edges open. ST-GCN makes a specific choice by combining two conceptually different sources of structure [2]. Its spatial structure is inherited from the anatomical skeleton representation, whereas its temporal structure is introduced by the model.

Let

$$G_S = (\mathcal{J}, \mathcal{E}_S)$$

denote the anatomical skeleton graph. For NTU RGB+D, the vertices correspond to the 25 skeletal joints, while $\mathcal{E}_S$ is determined by the connectivity of the underlying skeleton layout [1]. Thus, the spatial adjacency is fixed by the input representation rather than learned or defined from the joint coordinates by ST-GCN.

Spatial edges reproduce this anatomical adjacency at every time point,

$$\mathcal{E}_{\text{space}} = \{\{(t, i), (t, j)\} : t \in \mathcal{T}, \{i, j\} \in \mathcal{E}_S\}.$$

ST-GCN additionally introduces temporal edges by connecting the same joint in consecutive time points,

$$\mathcal{E}_{\text{time}} = \{\{(t, j), (t+1, j)\} : t \in \{1, \dots, T-1\}, j \in \mathcal{J}\}.$$

The graph used by the model is therefore

$$G_{ST} = (\mathcal{T} \times \mathcal{J}, \mathcal{E}_{\text{space}} \cup \mathcal{E}_{\text{time}}).$$

In particular, it contains no direct edges between different joints at different time points.

2.1.4. Classification Task

With the input representation and its graph structure specified, we can formulate the action classification task.

Let

$$\mathcal{Y} = \{1, \dots, N_{\text{cls}}\}$$

denote the set of action classes (e.g., 1 for drinking water and 2 for eating a meal) and let

$$\mathcal{D} = \{(X_i, y_i)\}_{i=1}^N, \quad y_i \in \mathcal{Y},$$

be a labeled dataset of N skeleton sequences, where

$$X_i = X_i^{\text{joint}} \in \mathbb{R}^{C \times T \times V \times M}$$

denotes the joint-position representation of sample i , as defined above, and y_i is its corresponding action label.

The probability simplex over the N_{cls} action classes is

$$\Delta^{N_{\text{cls}}-1} = \left\{ p \in [0, 1]^{N_{\text{cls}}} : \sum_{k=1}^{N_{\text{cls}}} p_k = 1 \right\}.$$

Skeleton-based action recognition aims to learn a classifier

$$F : \mathbb{R}^{C \times T \times V \times M} \longrightarrow \Delta^{N_{\text{cls}}-1},$$

which assigns a distribution over action classes to a skeleton sequence. We decompose F into an encoder f_θ and a classification head c_ψ ,

$$F = c_\psi \circ f_\theta,$$

where

$$f_\theta : \mathbb{R}^{C \times T \times V \times M} \longrightarrow \mathbb{R}^d$$

maps a skeleton sequence to a d -dimensional representation. Here, f_θ denotes the Euclidean backbone encoder used for linear evaluation; the hyperbolic contrastive representation is obtained after projection, normalization, and exponential mapping. The classification head

$$c_\psi : \mathbb{R}^d \longrightarrow \Delta^{N_{\text{cls}}-1}$$

maps this representation to a distribution over the action classes.

The encoder and classification head are trained in separate stages. First, the encoder parameters θ are learned through representation learning. For evaluation, θ is then fixed and only the parameters ψ of the classification head are trained on the labeled training set $\mathcal{D}_{\text{train}}$. While the classification head can in general be any map into $\Delta^{N_{\text{cls}}-1}$, in this thesis linear evaluation is used.

After training on $\mathcal{D}_{\text{train}}$, the resulting classifier F is evaluated on a separate evaluation set $\mathcal{D}_{\text{eval}}$, with

$$\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{eval}} = \emptyset.$$

Representation Learning. Representation learning trains the encoder to retain information useful for the task while reducing redundant or irrelevant information. This thesis uses contrastive learning for that purpose, as introduced in the following section.

2.2. Contrastive Learning

2.2.1. General Principle

Contrastive learning aims to learn a representation space in which similar samples are represented close to each other, while dissimilar samples are separated. Let z_i denote the representation of sample or augmented view i used in the contrastive objective. Conceptually, the scalar loss $\mathcal{L}_i$ for anchor i has attractive (att) and repulsive (rep) contributions,

$$\mathcal{L}_i = \mathcal{L}_i^{\text{att}} + \mathcal{L}_i^{\text{rep}}.$$

The positive and negative index sets $P(i)$ and $N(i)$ identify representations to be pulled towards, or pushed away from, the anchor z_i , respectively. The exact definition of these sets depends on the supervision available and on the contrastive learning method. In self-supervised instance discrimination, for example, two augmented views of the same input form a positive pair, while representations of other samples are typically treated as negatives [3].

2.2.2. InfoNCE Objective

A widely used realization of this principle is the InfoNCE objective. Let q_i denote an anchor or query representation, k_i its corresponding positive key, and $\{k_j\}_j$ a set of comparison keys containing the positive key k_i and the negative keys. The query is the representation being compared; its positive key is a matching representation, for example

another augmented view of the same sequence. The loss is

$$\mathcal{L}_i = -\log \frac{\exp(s(q_i, k_i)/\tau)}{\sum_j \exp(s(q_i, k_j)/\tau)},$$

where $s : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{R}$ is a similarity function on the representation space $\mathcal{Z}$, and $\tau > 0$ is the temperature. Equivalently,

$$\mathcal{L}_i = \underbrace{-\frac{s(q_i, k_i)}{\tau}}_{\mathcal{L}_i^{\text{att}}} + \underbrace{\log \sum_j \exp\left(\frac{s(q_i, k_j)}{\tau}\right)}_{\mathcal{L}_i^{\text{rep}}}.$$

The first term favors increasing the similarity between the anchor and its positive key. The second term exerts repulsive pressure across all comparison keys, including the positive key. For the positive pair, the attractive contribution dominates, so the combined objective still favors increasing its similarity. Thus, InfoNCE implements the general attractive–repulsive principle through a softmax classification objective rather than through two independently optimized pairwise losses. The temperature controls the concentration of this softmax distribution: smaller values place more emphasis on the most similar comparison samples.

2.2.3. Momentum Contrast in SkeletonCLR

SkeletonCLR adopts this InfoNCE objective together with a momentum-contrast mechanism [10, 4]. A query encoder is updated by backpropagation, whereas a key encoder is updated as an exponential moving average,

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q,$$

with momentum coefficient m . The key representations are stored in a first-in-first-out queue and serve as comparison samples for subsequent minibatches. This allows the contrastive objective to use a large set of negative representations without requiring correspondingly large minibatches, while the slowly updated key encoder provides comparatively stable target representations.

2.2.4. Extended Positive Sets and False Negatives

The definition of the positive set can be extended beyond a single augmented view. In supervised contrastive learning, all samples with the same class label may be treated as positives [8]. For an anchor i , the corresponding loss can be written as

$$\mathcal{L}_i = -\frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(s(z_i, z_p)/\tau)}{\sum_{a \neq i} \exp(s(z_i, z_a)/\tau)},$$

where $P(i)$ denotes the positive set of i . Equivalently, positive relations can be described by binary indicators or nonnegative weights over the comparison samples. This formulation is particularly useful in the present work, since the same contrastive objective can be used with ground-truth class relations as well as relations inferred from pseudo-labels.

A fundamental limitation of instance-level contrastive learning is the false-negative problem. Samples are considered negative merely because they originate from different instances, although they may depict the same action or semantically related actions. This issue is particularly relevant for skeleton-based action recognition, where different sequences can represent the same action class and several action classes may share similar motion patterns. Consequently, pushing every different instance apart can conflict with the semantic structure that the representation should ultimately capture. Pseudo-labeling provides a mechanism for modifying the positive set by identifying additional related samples. In the proposed approach, different sequences with sufficiently confident, overlapping cluster assignments can receive positive-pair weights; this may reduce false-negative repulsion when the learned groups reflect related actions. Hierarchy-aware pseudo-labels further generalize this idea by allowing the strength of such relations to depend on their estimated semantic or hierarchical similarity.

2.3. Hyperbolic Geometry and the Poincaré Ball

The contrastive objective compares representations through a similarity measure, whose meaning depends on the underlying geometry. Euclidean space is the standard embedding space in representation learning. Hyperbolic geometry provides an alternative with constant negative curvature.

Definition 2.3.1 (Hyperbolic space). A d -dimensional hyperbolic space of curvature $-c$, with $c > 0$, is a complete and simply connected d -dimensional Riemannian manifold with constant sectional curvature $-c$.

In this thesis, hyperbolic space is represented using the Poincaré ball model.

Definition 2.3.2 (Poincaré ball model). The d -dimensional Poincaré ball of curvature $-c$ is the Riemannian manifold

$$\mathbb{D}_c^d = \left\{ x \in \mathbb{R}^d : c\|x\|^2 < 1 \right\}, \quad g_x^c(u, v) = \left(\frac{2}{1 - c\|x\|^2} \right)^2 \langle u, v \rangle,$$

for $u, v \in T_x \mathbb{D}_c^d \cong \mathbb{R}^d$, where $\langle \cdot, \cdot \rangle$ is the Euclidean inner product and $\|\cdot\|$ is the Euclidean norm.

2. Theoretical Background

The underlying set is the open Euclidean ball of radius $1/\sqrt{c}$, equipped here with the hyperbolic metric g^c . Its boundary is not part of the space.

The Riemannian metric induces the hyperbolic distance

$$d_c(x, y) = \frac{1}{\sqrt{c}} \operatorname{arcosh} \left(1 + \frac{2c\|x - y\|^2}{(1 - c\|x\|^2)(1 - c\|y\|^2)} \right)$$

for $x, y \in \mathbb{D}_c^d$. For fixed $y \in \mathbb{D}_c^d$, $d_c(x, y)$ tends to infinity as x approaches the boundary, whereas the Euclidean distance $\|x - y\|$ remains bounded.

Mapping Euclidean representations to the Poincaré ball. The tangent space at the origin can be identified with Euclidean space,

$$T_0\mathbb{D}_c^d \cong \mathbb{R}^d.$$

This provides a natural connection between Euclidean neural network outputs and hyperbolic representations.

Definition 2.3.3 (Exponential map at the origin). The exponential map at the origin, $\operatorname{Exp}_0^c : T_0\mathbb{D}_c^d \rightarrow \mathbb{D}_c^d$, is given by

$$\operatorname{Exp}_0^c(v) = \begin{cases} \tanh(\sqrt{c}\|v\|) \frac{v}{\sqrt{c}\|v\|}, & v \neq 0, \\ 0, & v = 0. \end{cases}$$

The map preserves the radial direction of v while transforming its norm such that the resulting point lies inside $\mathbb{D}_c^d$.

Here, Exp_0^c is the *Riemannian exponential map* of $(\mathbb{D}_c^d, g^c)$, defined by following a geodesic with the given initial tangent vector. Appendix B gives the general definitions and derives the explicit formula from the Poincaré metric.

Hierarchical representations. Hyperbolic space is particularly well suited for representing hierarchical structures. The volume of a hyperbolic ball grows exponentially with its radius, whereas volume grows only polynomially in Euclidean space. Similarly, the number of nodes in a tree with a fixed branching factor grows exponentially with its depth. This correspondence allows tree-like structures to be embedded in low-dimensional hyperbolic spaces with comparatively low distortion [5]. This is a representational capacity; meaningful hierarchical organization still has to be encouraged by the training signal.

Hyperbolic contrastive representations. The preceding constructions can be incorporated into contrastive learning by mapping Euclidean representations into the Poincaré ball and replacing the Euclidean similarity measure by one defined through the hyperbolic distance [11, 12]. Given hyperbolic representations $q, k \in \mathbb{D}_c^d$, define

$$s_{\mathbb{D}}(q, k) = -d_c(q, k).$$

Since smaller distances correspond to more similar representations, the negative distance has the same ordering convention as an ordinary similarity score: more similar pairs receive larger values.

HypSkeletonCLR applies this construction to SkeletonCLR [7].

2.4. Pseudo-Labeling and Clustering

The next question is how to derive training signals that encourage this organization. Pseudo-labeling assigns automatically generated labels to otherwise unlabeled samples and uses these labels as additional supervision during training. Pseudo-labels may be obtained from model predictions, clustering procedures, or other data-driven assignments. In contrast to ground-truth labels, they are inferred from the available data and may therefore be noisy.

For a sample X_i , let

$$\hat{y}_i \in \{1, \dots, K\}$$

denote its pseudo-label. In a clustering-based setting, $\hat{y}_i$ can, for example, indicate the cluster or prototype to which the representation of X_i is assigned. Similar assignment patterns can provide supervision for consistent grouping across augmentations and additional contrastive positives.

Pseudo-labels can also be used in contrastive learning. While ordinary self-supervised contrastive learning defines positives through different augmentations of the same sample, pseudo-labels allow additional samples to be treated as positives when they receive the same or sufficiently similar pseudo-label. The resulting positive set is discussed in Section 2.2, while the specific pseudo-labeling strategy used in this thesis is introduced in Chapter 4.

2.5. t-SNE and its limitations

Assessing how these training signals organize the representation space requires tools for examining the learned embeddings. t-SNE is used later in this thesis to visualize learned representations and to inspect whether they exhibit apparent hierarchical structure. Since

such visualizations are necessarily low-dimensional, it is important to clarify what t-SNE preserves and, in particular, what conclusions about the presence or absence of hierarchy can be drawn from them.

t-distributed Stochastic Neighbor Embedding (t-SNE) maps high-dimensional representations to two dimensions by matching local neighbourhood probabilities [13]. The full mathematical definition is given in Appendix A.4.

The ordinary t-SNE plots use Euclidean distances after mapping samples and prototypes to the origin's tangent space (Section 6.3.2). The hyperbolic variant uses hyperbolic input affinities and a Poincaré-disk output (Appendix A.2).

Thus, t-SNE attempts to reproduce local neighbourhood probabilities rather than distances or global geometry. In particular, the objective does not require preservation of Euclidean or hyperbolic distances, angles, geodesics, or the global arrangement of different groups. Consequently, distances between well-separated clusters in a t-SNE visualization should not be interpreted as distances in the original representation space.

Dimensional limitation. There is an additional fundamental limitation when the intrinsic dimension of the learned representations exceeds the dimension of the visualization. Suppose that the representations sample an underlying d -dimensional manifold $\mathcal{M}$. If $d > 2$, then its complete local neighbourhood structure cannot be represented faithfully in two dimensions.

More generally, the following statement holds.

Proposition 2.5.1 (Dimensional obstruction). *Let $\mathcal{M}$ be a d -dimensional manifold and let $m < d$. Then there exists no topological embedding*

$$f : \mathcal{M} \longrightarrow \mathbb{R}^m.$$

Proof. Assume, for contradiction, that such an embedding f exists. Let

$$\varphi : U \longrightarrow V$$

be a coordinate chart of $\mathcal{M}$, where $V \subseteq \mathbb{R}^d$ is open. Since f is an embedding,

$$f \circ \varphi^{-1} : V \longrightarrow \mathbb{R}^m$$

is continuous and injective.

Consider the canonical inclusion

$$\iota : \mathbb{R}^m \longrightarrow \mathbb{R}^d, \quad \iota(x_1, \dots, x_m) = (x_1, \dots, x_m, 0, \dots, 0).$$

Then

$$g = \iota \circ f \circ \varphi^{-1} : V \longrightarrow \mathbb{R}^d$$

2. Theoretical Background

is continuous and injective. By the invariance of domain theorem, $g(V)$ must be open in $\mathbb{R}^d$. However,

$$g(V) \subseteq \mathbb{R}^m \times \{0\}^{d-m},$$

and this subset has empty interior in $\mathbb{R}^d$. Hence $g(V)$ cannot be open, which is a contradiction. Therefore no such embedding exists. $\square$

This result concerns an underlying manifold, not merely a finite plotted sample; t-SNE optimizes neighbourhood probabilities rather than a topological embedding.

For this reason, t-SNE visualizations in this thesis are treated as qualitative diagnostics rather than evidence for or against hierarchical structure. They can reveal apparent clusters or local groupings, but the absence of a visually recognizable hierarchy in two dimensions does not imply that no hierarchy exists in the original embedding space. Hierarchical structure may depend on radial organization, distances, or branching relations that cannot be represented faithfully in the two-dimensional visualization. Claims about hierarchy should therefore primarily rely on quantities computed directly in the learned embedding space.

The next chapter uses these concepts to compare existing skeleton-learning methods and identify how hierarchy-aware supervision can extend them.

3. Related Work

This chapter reviews skeleton-based representation learning, hyperbolic embeddings, and hierarchy-aware pseudo-labeling.

3.1. Skeleton-Based Action Recognition

Graph-based approaches are well established for skeleton-based human action recognition. ST-GCN introduced a spatio-temporal graph representation that directly exploits the anatomical structure of skeleton sequences [2]. It subsequently became a widely used backbone for skeleton representation learning.

The present thesis follows the line of work built around this architecture rather than investigating new action-recognition backbones. In particular, SkeletonCLR and CrosSCLR use ST-GCN for self-supervised representation learning and additionally exploit joint, motion, and bone representations as complementary views of a skeleton sequence [4]. This architecture also forms the basis of the hyperbolic extension HypSkeletonCLR [7], on which the present work builds.

3.2. Contrastive Learning on Skeleton Sequences

On this shared ST-GCN backbone, the training objective determines which relations between sequences shape the learned features. SkeletonCLR adapts momentum-based contrastive learning to skeleton sequences, using augmented versions of the same sequence as positive pairs and a memory queue to provide negatives [4]. CrosSCLR extends this idea by transferring information between joint, motion, and bone views. Similar samples identified in one representation can thereby provide additional positive relations in another representation [4].

Two further methods included in the HypSkeletonCLR comparison are AimCLR and ActCLR [7, Table 5.7]. AimCLR uses extreme skeleton augmentations and an attention-guided feature-drop module to expose the encoder to more diverse movement patterns, extending the range of positive examples used for self-supervised representation learning [14]. ActCLR identifies discriminative motion regions, termed actionlets, without labels. It uses these regions to guide data augmentation and feature pooling, treating informative

motion and static regions differently [15]. Whereas AimCLR and ActCLR primarily change which motion information is exposed to the encoder, the present extension retains the inherited input pipeline and adds supervision through learned prototypes and their relations.

The comparison includes both single-stream models and the three-stream variants 3s-SkeletonCLR, 3s-CrosSCLR [4], 3s-AimCLR [14], and 3s-ActCLR [15], which combine joint, motion, and bone representations. The same work also reproduces SkeletonCLR and 3s-SkeletonCLR as direct Euclidean baselines for HypSkeletonCLR and 3s-HypSkeletonCLR, respectively [7, Tables 5.6 and 5.7]. These methods provide the comparison context for the hyperbolic baseline extended in the present thesis.

CrosSCLR is particularly relevant to the present work because its cross-view consistency mechanism moves beyond supervision defined purely by instance identity. Relations inferred from the learned representation space can themselves influence subsequent training.

Gong et al. [16] combine masked joint reconstruction with contrastive learning. More recent self-supervised advances include AMR, which uses a decoupled decoder and motion-guided reconstruction [17].

3.3. Hyperbolic Representation Learning

Hyperbolic representation learning was established as an effective approach for embedding hierarchical data by Poincaré embeddings [5]. Hyperbolic neural networks subsequently extended such representations to learned neural-network models [6]. The geometric motivation and the Poincaré ball model used in this thesis were introduced in Section 2.3 and are not repeated here.

Exponential mapping into the Poincaré ball and the use of negative hyperbolic distance in contrastive objectives are established in prior work: Ermolov et al. [11] use this combination for supervised metric learning, and Ge et al. [12] for scene–object relations.

HypSkeletonCLR applies this construction to SkeletonCLR [7]. This thesis adopts that baseline and studies additional clustering and hierarchy objectives to encourage hierarchical structure.

3.4. Hierarchy-Aware and Pseudo-Label-Based Objectives

Pseudo-labeling provides supervisory targets that are inferred during training rather than supplied as manual annotations. A common approach is to derive these targets from the current representation or prediction space. DeepCluster, for example, repeatedly clusters learned representations and uses the resulting assignments as pseudo-labels for subsequent

network training [18]. Prototypical Contrastive Learning similarly introduces cluster prototypes into contrastive representation learning [9]. These methods demonstrate how intermediate group structure can provide supervision beyond instance-level contrastive pairs.

Pseudo-labeling has also been applied directly to action recognition. Cross-Model Pseudo-Labeling (CMPL) trains two models with different architectural biases and uses confident predictions from each model as pseudo-labels for the other [19]. Although CMPL does not model a hierarchy, it is relevant as an alternative pseudo-labeling strategy: the quality of the supervisory signal is improved by obtaining it from a complementary representation rather than exclusively from the model being trained.

The methods above primarily produce flat pseudo-labels. Hierarchical clustering additionally represents relations between the inferred groups. This can express that distinct groups share a broader parent, preserving relatedness without assigning all their samples to the same flat cluster. Hyperbolic Hierarchical Clustering (HypHC) formulates hierarchical clustering through continuous optimization of embeddings in hyperbolic space [20]. This establishes a connection between the geometry of the learned representation and a discrete hierarchical organization.

Most directly related to the present work is the scalable Hyperbolic Hierarchical Clustering (sHHC) method of Long and van Noord [21]. Their approach first obtains balanced pre-clusters and then constructs a hierarchy between the corresponding cluster representatives in hyperbolic space. The resulting hierarchy is used to derive continuous hierarchical pseudo-labels for self-supervised learning. Compared with applying hierarchical clustering directly to all samples, operating on pre-clusters also makes the hierarchical optimization substantially more scalable.

The present thesis transfers this idea to skeleton-based action representations. Starting from HypSkeletonCLR, balanced cluster assignments and relations between cluster prototypes are introduced as additional supervisory signals. In contrast to the cross-modal setting of Long and van Noord, the focus is on whether explicitly inducing hierarchical organization within an existing hyperbolic skeleton representation improves the learned features and downstream action recognition. We retain balanced pre-clustering and prototype-level hierarchy learning, but use a sampled ranking surrogate and flat assignment-based contrastive feedback instead of sHHC’s affinity-weighted hierarchy loss and cross-modal hierarchical pseudo-labels (Section 4.3.2). The following chapter develops this extension by specifying prototype assignments, a hierarchy objective, and pseudo-label feedback within a common training framework.

4. Methodology

4.1. Method Overview

The proposed method builds on the hyperbolic contrastive learning pipeline of HypSkeletonCLR [7]. The skeleton encoder, projection head, momentum encoder, memory bank, and mapping of learned representations to the Poincaré ball are retained. The contribution of this thesis lies instead in modifying the objectives acting on the learned representations.

The central extension is inspired by sHHC [21]. Rather than applying a hierarchy objective directly to all individual samples, the learned representations are assigned to K learned prototypes with balanced soft assignment targets. A hyperbolic hierarchy objective regularizes relations between these prototypes. Both objectives are optimized jointly, with hierarchy learning acting at the prototype level. Using a small set of shared representatives makes the hierarchy objective cheaper than operating on all samples and separates group membership from relations between groups.

Figure 4.1 summarizes the pipeline; the orange branch is detailed in Figure 4.3. Given a skeleton sequence, HypSkeletonCLR produces hyperbolic representations

$$z_i \in \mathbb{D}_c^d.$$

These representations participate in the hyperbolic contrastive objective, while an additional branch computes balanced cluster assignments to learned prototypes. The assignments contribute a clustering loss, whereas relations between the prototypes contribute the hierarchy loss.

A second extension uses cluster assignments as pseudo-labels. In the reported soft variant, confident pairs receive additional positive weights according to the overlap of their assignment distributions. A distance-floor penalty discourages these pairs from becoming too close. This feedback replaces the queue-based contrastive objective with a weighted objective on both views of the current minibatch.

The main modeling choices concern the level of hierarchy learning, the form of the hierarchy loss, and the source of additional positives. Relative to HypSkeletonCLR, the extension adds prototype-level supervision; relative to sHHC, it uses a ranking surrogate and flat cluster assignments for contrastive feedback. These choices retain the skeleton backbone while making group relations part of its training objective.

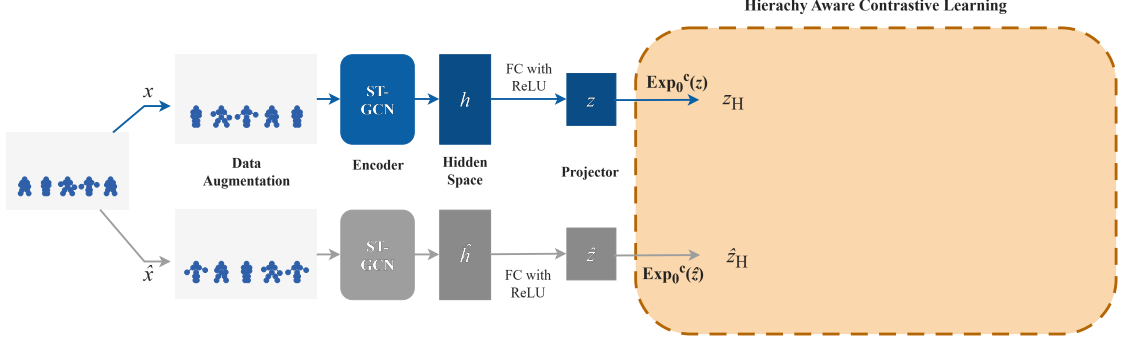


Figure 4.1.: Overview of the hierarchy-aware extension of HypSkeletonCLR. Hyperbolic representations receive balanced soft assignments to learned prototypes, whose relations are regularized by the hierarchy objective. See Section 4.3.

4.2. Base Representation Pipeline

We summarize the representation pipeline inherited from HypSkeletonCLR [7], which extends SkeletonCLR [4], and identify where the additional objectives act.

Given a skeleton sequence x , two augmented views are processed by an ST-GCN encoder and its momentum-updated counterpart. The resulting hidden representations pass through projection heads. Each output h_i is L_2 -normalized before the exponential map:

$$\tilde{h}_i = \frac{h_i}{\|h_i\|_2}, \quad z_i = \text{Exp}_0^c(\tilde{h}_i).$$

The inherited unit normalization has an important geometric consequence: every sample embedding lies at hyperbolic distance 2 from the origin, independently of curvature (Proposition B.2.1). For fixed normalized projector outputs, larger cosine similarity always corresponds to smaller hyperbolic distance. Curvature changes this relationship nonlinearly and can therefore affect contrastive probabilities and gradients, despite preserving pairwise distance rankings. Samples consequently have no radial variation, although hierarchical grouping through angular relationships remains possible. The trainable prototypes introduced in Section 4.3.1 retain radial freedom, providing additional flexibility for representing group relations.

Similarity in the contrastive objective is defined by negative hyperbolic distance,

$$s_{\mathbb{D}}(z_i, z_j) = -d_c(z_i, z_j),$$

as introduced in Section 2.3.

Figure 4.2 shows the inherited single-view architecture. The MoCo-style queue stores normalized tangent-space key representations, which are mapped to the ball for distance

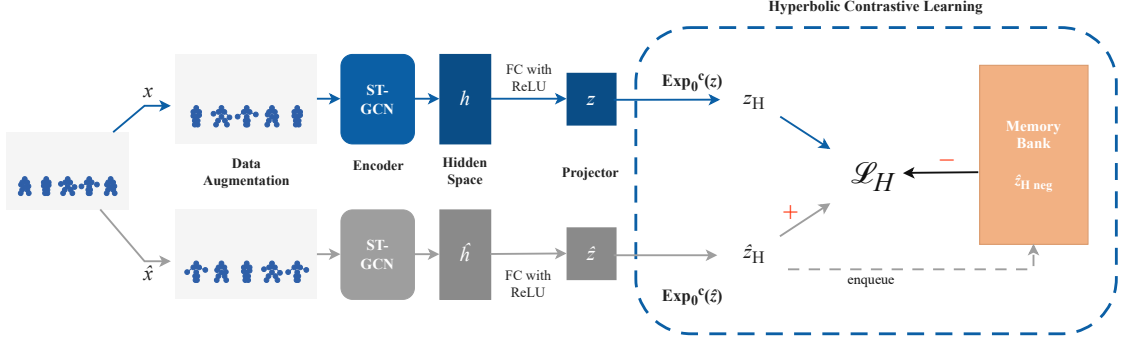


Figure 4.2.: Single-view HypSkeletonCLR architecture used as the representation backbone in this thesis. Reproduced from [7].

comparisons. The base objective compares each query with its paired key and the queued keys. The momentum encoder and queued representations receive no gradients.

Three-stream representation. Joint, motion, and bone inputs use separate encoders. Their normalized projection outputs are averaged and normalized again to unit Euclidean norm before mapping:

$$\tilde{h}_i^{\text{all}} = \text{normalize} \left(\frac{\tilde{h}_i^{\text{joint}} + \tilde{h}_i^{\text{motion}} + \tilde{h}_i^{\text{bone}}}{3} \right), \quad z_i^{\text{all}} = \text{Exp}_0^c(\tilde{h}_i^{\text{all}}).$$

This fusion is applied to both query and key views. Clustering uses the fused representations and one shared prototype set. Pseudo-label contrastive learning and the distance floor also use the fused representations. The base queue-based objective instead sums separate joint, motion, and bone losses.

4.3. Hierarchy-Aware Contrastive Learning

HypSkeletonCLR changes the geometry in which contrastive representations are learned, but it does not explicitly require the resulting representations to form a hierarchy. The main methodological extension of this thesis is therefore to add an objective that directly encourages hierarchical organization.

Applying a hierarchy objective to every individual training sample would be computationally expensive and would make the hierarchy sensitive to instance-level variation. Following the main idea of sHHC [21], the problem is therefore decomposed into two stages. First, learned representations are assigned to a smaller number of balanced

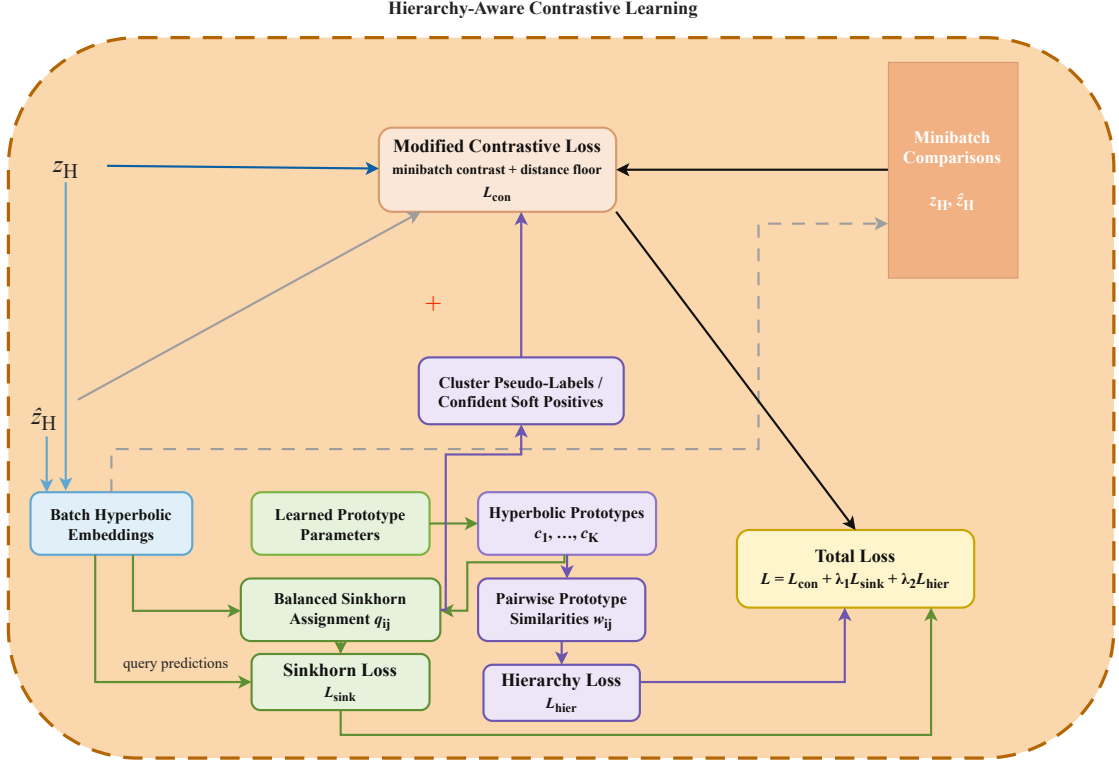


Figure 4.3.: Hierarchy-Aware Contrastive Learning overview. Sinkhorn uses balanced key-view targets and query predictions. The term L_{con} includes the minibatch pseudo-label loss and the distance-floor penalty.

pre-clusters. Second, the resulting cluster prototypes are organized hierarchically in hyperbolic space.

This decomposition has an additional interpretation in the present setting. The balanced clustering determines which intermediate groups are represented, whereas the hierarchy objective determines how these groups are related to one another.

4.3.1. Balanced Prototype Assignment

Let

$$z_1, \dots, z_N \in \mathbb{D}_c^d$$

denote a collection of learned representations and let K be the number of pre-clusters. For each representation z_i , the clustering component computes hyperbolic distances to the learned prototypes $\mu_1, \dots, \mu_K \in \mathbb{D}_c^d$. A softmax over the negative distances gives the

predicted probabilities

$$p_{ik} = p(y_i = k \mid z_i) = \frac{\exp[-d_c(z_i, \mu_k)/\tau_{\text{clust}}]}{\sum_{\ell=1}^K \exp[-d_c(z_i, \mu_\ell)/\tau_{\text{clust}}]}.$$

Here $k \in \{1, \dots, K\}$, and the prototype softmax temperature $\tau_{\text{clust}} > 0$ controls the sharpness of the distribution. Closer prototypes receive higher probabilities. Each augmented view yields a row-stochastic prediction matrix $P = (p_{ik}) \in \mathbb{R}^{N \times K}$.

Following the pre-clustering stage of Long and van Noord [21], Sinkhorn iterations produce a row-stochastic assignment matrix $Q = (q_{ik}) \in \mathbb{R}^{N \times K}$ while targeting balanced column masses:

$$\sum_{i=1}^N q_{ik} = \frac{N}{K} \quad \text{for every } k.$$

Uniform prototype marginals discourage concentrating assignment mass on a few prototypes, following balanced online clustering [22]. This regularizes minibatch assignment mass without assuming equal action frequencies. Prototypes don't need to correspond one-to-one to action classes. Finite Sinkhorn iterations approximate the column constraint. Equal hard cluster sizes and non-collapsed representations are not guaranteed.

During training, Sinkhorn balancing uses the entrywise kernel $(P^k)^{\tau_{\text{clust}}/\epsilon}$, where P^k contains the momentum key encoder's probabilities and $\epsilon > 0$. Uniform sample and prototype marginals define the balance target. The assignments supervise the query encoder's predictions on the other augmented view of the same samples, encouraging consistent cluster membership across augmentations. With q_{ik} denoting these balanced key-view targets and p_{ik} the query-view probabilities, the cross-entropy objective is

$$\mathcal{L}_{\text{Sinkhorn}} = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K q_{ik} \log p_{ik}.$$

The loss is averaged over the minibatch. Balanced assignments are held fixed during differentiation; gradients update the query encoder and prototypes through the query-view probabilities.

The cluster prototypes are trainable variables, rather than averages extracted from the current assignments. Each prototype is parameterized by a tangent-space vector $v_k \in \mathbb{R}^d$ through

$$u_k = \frac{\tanh \|v_k\|_2}{\|v_k\|_2} v_k, \quad \mu_k = \exp_0^c(u_k) \in \mathbb{D}_c^d.$$

At zero, the ratio is defined by its continuous limit. At $c = 1$, the squashing step has the same coordinate formula as the origin exponential map. The composition bounds prototype radii by $d_1(0, \mu_k) = 2 \tanh \|v_k\|_2 \in [0, 2)$, placing them inside the sample sphere

of radius 2 established in Proposition B.2.1. This is consistent with their intended role as group representatives.

Using trainable prototypes, as in SwAV [22], avoids recomputing centroids between optimization steps. Here, the same persistent representatives are updated by clustering and hierarchy losses, so both objectives shape a shared group structure.

4.3.2. Hyperbolic Hierarchy Objective

The reported H and HPL experiments use a sampled softmax ranking loss (`ranking_ce`), inspired by sHHC [21] and hyperbolic hierarchical clustering [20]. This smooth *ranking surrogate* uses softmax and cross-entropy to encourage greater estimated ancestor depth for the sampled anchor-positive pair than for either competing pair. The practical rationale for this surrogate is to express the preferred pair directly as a three-way classification target.

Affinity and triplet sampling. Let $A = (A_{ij}) \in \mathbb{R}^{K \times K}$ be the prototype affinity matrix. For distinct prototype indices $i, j \in \{1, \dots, K\}$,

$$A_{ij} = \frac{\exp[-d_c(\mu_i, \mu_j)/\tau_A]}{\sum_{u \neq v} \exp[-d_c(\mu_u, \mu_v)/\tau_A]}, \quad A_{ii} = 0,$$

where $\tau_A > 0$ controls how strongly distance affects affinity. An exponential moving average smooths these values: $\bar{A} \leftarrow \rho \bar{A} + (1 - \rho)A$, initialized with the first affinity matrix. The matrix is kept symmetric, with zero diagonal and unit total mass. Affinity values and sampled triplets are held fixed when differentiating the hierarchy loss.

An anchor a is sampled uniformly. A positive $p \neq a$ is sampled with weight $\bar{A}_{ap}$, while a negative $n \notin \{a, p\}$ is sampled with weight $\max_{i,j} \bar{A}_{ij} - \bar{A}_{an}$. If the available weights sum to zero, sampling is uniform over valid indices. This favors nearer positives and farther negatives, but does not guarantee that (a, p) is the highest-affinity pair within each sampled triplet. Here $a, p, n \in \{1, \dots, K\}$ are prototype indices: A_{ap} , A_{an} , and A_{pn} are the entries of A for the anchor–positive, anchor–negative, and positive–negative pairs, respectively. Their barred versions are entries of the smoothed matrix $\bar{A}$.

Ancestor-depth estimate and ranking loss. The depth of the shared ancestor is estimated from distances using the Gromov product at the origin [20, Appendix A.2, Definition A.2]:

$$\hat{r}_{ij} = \max \left\{ 0, \frac{1}{2} [d_c(0, \mu_i) + d_c(0, \mu_j) - d_c(\mu_i, \mu_j)] \right\}.$$

On a rooted tree, the Gromov product equals the depth of the lowest common ancestor; in hyperbolic space, it approximates the hyperbolic LCA depth [20, Appendix A.3,

Lemma A.1]. For a sampled triplet (a, p, n) , define the score vector and its softmax probabilities by

$$\mathbf{s}_{apn} = (\hat{r}_{ap}, \hat{r}_{an} + m, \hat{r}_{pn} + m)^\top, \quad \boldsymbol{\pi}_{apn} = \text{softmax}(\mathbf{s}_{apn}),$$

with margin $m \geq 0$. The one-hot target $\mathbf{t} = (1, 0, 0)^\top$ selects the anchor-positive pair. Its cross-entropy is $-\mathbf{t}^\top \log \boldsymbol{\pi}_{apn} = -\log \pi_{apn,1}$, where the logarithm is applied componentwise. Averaging over the sampled triplets $\mathcal{T}$ gives

$$\mathcal{L}_{\text{hierarchy}} = \frac{1}{|\mathcal{T}|} \sum_{(a,p,n) \in \mathcal{T}} \log \left[1 + e^{\hat{r}_{an} + m - \hat{r}_{ap}} + e^{\hat{r}_{pn} + m - \hat{r}_{ap}} \right].$$

Thus, the sampled positive pair is encouraged to have a deeper shared ancestor than either competing pair. The target describes a relation between prototypes, not a ground-truth action label.

This log-sum-exp form follows the multi-negative softmax ranking construction in Sohn’s N-pair work [23, Equations (3) and (6)]. Here, ancestor-depth estimates replace embedding inner products, the candidates are three pairs within one prototype triplet, and a margin is added. The citation supports the ranking construction; this particular hierarchical adaptation is not the N-pair method itself.

Affinity magnitudes influence sampling rather than directly weighting the loss. The depth estimate also uses only origin and pairwise distances, avoiding the construction of explicit merge points. These choices simplify the objective, but no advantage over the original sHHC objective has been demonstrated. The hierarchy loss directly updates only the prototypes. Their changed positions influence subsequent encoder updates through clustering and, when enabled, recomputed pseudo-label weights. But assignment targets receive no gradients.

Difference from sHHC. For each triplet, the original sHHC objective has the form

$$\ell_{\text{sHHC}}(\mathbf{r}, \mathbf{w}) = \mathbf{1}^\top \mathbf{w} - \text{softmax}(\mathbf{r})^\top \mathbf{w},$$

where, in the pair order $(a, p), (a, n), (p, n)$,

$$\mathbf{r} = (r_{ap}, r_{an}, r_{pn})^\top, \quad r_{ij} = d_c(0, z_{i \vee j}), \quad \mathbf{w} = (w_{ap}, w_{an}, w_{pn})^\top.$$

Here $z_{i \vee j}$ is the hyperbolic representation of the lowest common ancestor (LCA) of pre-clusters i and j , w_{ij} is the similarity between their centroids, and $\mathbf{1} = (1, 1, 1)^\top$ [21, Equation (10)]. Thus, sHHC directly weights all three pairs by their similarities. Here, affinities only guide sampling; cross-entropy selects (a, p) with target $\mathbf{t} = (1, 0, 0)^\top$, using the depth estimates $\hat{r}_{ij}$ and margin m . Furthermore, sHHC uses hierarchy nodes as cross-modal pseudo-labels, whereas the extension below uses flat cluster assignments for contrastive learning.

4.4. Cluster-Based Pseudo-Labeling

Balanced key-view assignments also provide pseudo-labels for contrastive learning. For a minibatch of N clips, let $\mathbf{q}_i = (q_{i1}, \dots, q_{iK})^\top$ and define the hard assignment

$$\hat{y}_i = \arg \max_{k \in \{1, \dots, K\}} q_{ik}.$$

These labels identify learned prototypes, not action categories. They depend on the current minibatch and can change during training.

To exclude uncertain assignments, both clips in a pseudo-positive pair must pass the confidence threshold η , following confidence-based pseudo-label selection [19]. With $b_i = \mathbf{1}[\max_k q_{ik} \geq \eta]$, the reported soft variant uses

$$U_{ij} = \mathbf{1}[i \neq j] b_i b_j \langle \mathbf{q}_i, \mathbf{q}_j \rangle.$$

We use the dot product to retain graded assignment overlap that a hard-label match would discard. Thus, pairs with different hard assignments can receive positive weight. The optional hard variant replaces the dot product by $\mathbf{1}[\hat{y}_i = \hat{y}_j]$. Assignments and the resulting weights are held fixed during differentiation.

To limit the influence of large pseudo-clusters, weights are normalized over the other clips:

$$\bar{U}_{ij} = \begin{cases} \frac{U_{ij}}{\sum_{\ell=1}^N U_{i\ell}}, & \sum_{\ell=1}^N U_{i\ell} > 0, \\ 0, & \text{otherwise.} \end{cases} \quad B_{ij} = \lambda_{\text{aug}} \mathbf{1}[i = j] + \lambda_{\text{P}} \bar{U}_{ij}.$$

The diagonal of B represents the augmentation relation for the same clip; its off-diagonal entries represent pseudo-positive relations.

Comparing both minibatch views uses current prototype assignments, avoiding storage or recomputation for queued keys. Define the view-indexed set

$$\mathcal{I} = \{(i, v) : i = 1, \dots, N, v \in \{q, k\}\}, \quad A(a) = \mathcal{I} \setminus \{a\}.$$

Both query and momentum-key views act as anchors, each with $2N - 1$ comparisons. Key features remain detached from encoder gradients. The queue is still updated but does not enter this loss.

The clip-level mask is repeated across both views, excluding self-comparisons:

$$W_{(i,v),(j,w)} = \mathbf{1}[(i, v) \neq (j, w)] B_{ij}.$$

An anchor with eligible pseudo-positives therefore has pseudo-positive mass $2\lambda_{\text{P}}(t)$, while its paired augmentation has weight λ_{aug} . Thus, λ_{P} specifies mass in the clip-level mask, before expansion to views.

The loss extends the multiple-positive formulation of supervised contrastive learning [8] to weighted pseudo-positives and negative hyperbolic distance:

$$\mathcal{L}_{\text{pseudo}} = -\frac{1}{2N} \sum_{a \in \mathcal{I}} \frac{1}{\sum_{b \in A(a)} W_{ab}} \sum_{b \in A(a)} W_{ab} \log \frac{\exp[-d_c(z_a, z_b)/\tau]}{\sum_{r \in A(a)} \exp[-d_c(z_a, z_r)/\tau]}.$$

Since $\lambda_{\text{aug}} > 0$, every anchor retains its paired augmentation. The loss is averaged over all $2N$ view anchors.

A shared prototype need not imply identical motion. We therefore add a distance-floor penalty to discourage excessive attraction between distinct clips; paired augmentations are exempt. For $a = (i, v)$ and $b = (j, w)$, set

$$V_{ab} = \mathbf{1}[i \neq j] \lambda_P(t) \bar{U}_{ij}, \quad \mathcal{L}_{\text{floor}} = \frac{\sum_{a,b \in \mathcal{I}} V_{ab} [\delta_{\min} - d_c(z_a, z_b)]_+^2}{\sum_{a,b \in \mathcal{I}} V_{ab}}.$$

Here $[x]_+ = \max(x, 0)$, and the loss is zero when the total weight is zero. This global weighted mean excludes all pairs from the same clip, including its two augmented views. The threshold $\delta_{\min}$ defines a soft penalty, not a guaranteed minimum separation.

As in supervised HypSkeletonCLR, additional positive relations modify the contrastive objective [7]. Here they are inferred from assignments rather than supplied by class labels. This is also related to CrossCLR’s use of learned relations to introduce additional positives [4]. Assignments therefore serve two roles: they supervise clustering and determine instance-level pseudo-positive weights. Soft balancing does not guarantee semantically correct pairs, so the effect of this feedback must be assessed empirically.

4.5. Overall Training Objective

To optimize the instance-level and prototype-level relations jointly, the hierarchy-aware framework combines the inherited hyperbolic contrastive objective with the two prototype-level objectives. Without cluster-based pseudo-label feedback, the total objective is

$$\mathcal{L} = \mathcal{L}_{\text{hypNCE}} + \lambda_S \mathcal{L}_{\text{Sinkhorn}} + \lambda_H \mathcal{L}_{\text{hierarchy}},$$

where the weights λ_S and λ_H are scheduled functions of the global optimizer step, with the argument omitted for readability. The hierarchy term is zero on steps without a hierarchy update. Warm-up, ramp, and update intervals are specified in Section 5.3.

With hierarchy loss disabled, this variant is denoted **HypSkeletonCLR-C**. With the hierarchy objective, it is denoted **HypSkeletonCLR-H**. The corresponding three-stream variants add the prefix **3s-**.

The three terms have distinct roles. The hyperbolic contrastive objective operates at the level of individual representations and maintains the instance-discrimination signal

of HypSkeletonCLR. The Sinkhorn objective introduces balanced intermediate groups, while the hierarchy objective organizes the corresponding group prototypes relative to one another.

When cluster-based pseudo-labeling is enabled, the contrastive component is replaced by the pseudo-label-guided formulation together with the distance-floor term,

$$\mathcal{L} = \mathcal{L}_{\text{pseudo}} + \lambda_{\text{F}}\mathcal{L}_{\text{floor}} + \lambda_{\text{S}}\mathcal{L}_{\text{Sinkhorn}} + \lambda_{\text{H}}\mathcal{L}_{\text{hierarchy}}.$$

The scheduled weight λ_{F} controls the distance-floor penalty. During an epoch-level transition, the queue-based contrastive term is blended with $\mathcal{L}_{\text{pseudo}} + \lambda_{\text{F}}\mathcal{L}_{\text{floor}}$.

The **HypSkeletonCLR-PL** variant uses pseudo-label feedback without hierarchy loss. Adding the hierarchy loss gives **HypSkeletonCLR-HPL**. The same convention is used for the three-stream models.

5. Experimental Setup

5.1. Dataset

All experiments use the NTU RGB+D 60 skeleton benchmark [1]. It contains 60 action classes recorded from multiple subjects and camera views. Each sample stores 3D coordinates for 25 body joints and up to two persons. We evaluate on the standard cross-subject and cross-view protocols: cross-subject separates by actor identity, while cross-view trains on cameras 2 and 3 and validates on camera 1.

The implementation uses the processed NTU60 frame-50 files inherited from the SkeletonCLR setting. The inherited preprocessing (i) estimates the valid length L by counting frames with nonzero coordinates for the first joint of the first person, assuming trailing zero padding, (ii) centers each skeleton on joint 21, (iii) resamples the initial L frames to 50 frames using linear interpolation, and (iv) applies a per-frame rotation about the vertical y -axis using the shoulder direction (joints 5 and 9). The resulting 50-frame tensors are stored as `train_position.npy`, `val_position.npy`, `train_label.pkl`, and `val_label.pkl` for each split. The main setting uses joint coordinates. Three-stream experiments additionally use motion and bone streams, following SkeletonCLR and CrosSCLR [4].

Labels are not used as supervision during self-supervised pre-training, only for book-keeping, linear evaluation, and the post-training hierarchy analysis.

5.2. Implementation Details

The implementation and experiment configurations are available at <https://github.com/AlexanderSchwer/HypSkeletonCLR-HPL>. The code extends HypSkeletonCLR [7], which builds on SkeletonCLR and an ST-GCN encoder [2, 4]. Large language models (LLMs) assisted in generating experiment code and configuration files.¹

Unless stated otherwise, the encoder uses hidden channel width 16, hidden dimension 256, dropout 0.5, edge-importance weighting, and the NTU spatial graph. The projection

¹LLMs were also used to refine the wording of the thesis.

dimension is fixed to 32 and embeddings are mapped to a Poincaré ball with curvature $-c = -1$, matching the strongest reported HypSkeletonCLR settings.

Contrastive pre-training uses a MoCo-style queue of size 32768, momentum 0.999, and temperature 0.07. Optimization uses SGD with Nesterov momentum, weight decay 10^{-4} , learning rate 0.01, and batch size 128. The learning rate is warmed up for 15 epochs from factor 0.001 and then cosine-annealed until epoch 300. Checkpoints are saved every five epochs.

Pre-training uses a dual-view feeder with two independently augmented views: temporal cropping with padding ratio 6 and shear transformation with amplitude 0.5. Linear evaluation disables these augmentations.

Hierarchy-objective variants. The sampled ranking objective in Section 4.3.2 corresponds to `ranking_ce`, the option used for the reported H and HPL experiments.

Three-stream projection heads. With `mlp=True`, the inherited three-stream implementation reused the joint stream’s already constructed projection head inside the motion and bone heads, for both query and key encoders. This introduced shared parameters and an additional projection layer in the motion and bone streams. Commit 26c00df (10 September 2026) in `net/skeletonclr_3views.py` corrected this by constructing each head from its own encoder’s final layer. All eight component models in Table 6.3 use the corrected structure. The non-clustered three-stream baseline was also evaluated with corrected heads on cross-view. Its results are reported in Section 6.2.2.

5.3. Training Protocol

These model settings are used within a two-stage protocol of representation learning and linear evaluation. Each model is pre-trained for 300 epochs and then evaluated with a frozen encoder and newly trained linear classifier. The non-clustered reference uses the original HypSkeletonCLR contrastive objective throughout. Clustered training uses prototype softmax temperature $\tau_{\text{clust}} = 0.1$ and 20 Sinkhorn iterations. The regularization parameter $\epsilon = 0.05$ defines the Sinkhorn kernel $P^{\tau_{\text{clust}}/\epsilon}$, with the power applied entrywise to the key-view probabilities. Sinkhorn and hierarchy weights have maxima $\lambda_S = 1.0$ and $\lambda_H = 0.2$. Both use 1000 warm-up steps followed by a 2000-step linear ramp. The hierarchy term is evaluated every 200 global steps, using 512 prototype triplets and margin 0.05.

In the cluster-count ablation, HPL uses the queue-based augmentation objective for 50 epochs, then transitions to the soft pseudo-label objective over 20 epochs. Follow-up HPL runs start in pseudo mode at epoch 1. Loss jumps observed during the transition in

the cluster-count ablation were considered undesirable and motivated this change. The step-based warm-up of the additional losses was retained. Both schedules optimize the same final HPL objective for most of training. Only its introduction changes. All HPL runs within the K sweep share the original schedule, which is also retained by the reused joint-only cross-view HPL component result. The effect of the schedule change was not separately ablated.

The augmentation weight is $\lambda_{\text{aug}} = 1.0$, and the total clip-level pseudo-positive weight $\lambda_{\text{P}}(t)$ reaches a maximum of 0.2. For an anchor with eligible pseudo-positives, counting both views of each other clip gives total pseudo-positive weight $2\lambda_{\text{P}}(t)$, with maximum 0.4.

The pseudo weight and distance-floor weight use a 1000-step warm-up and a 2000-step linear ramp measured from the start of training. The floor threshold is 0.1 and its maximum weight is 1.0. These step ramps have already elapsed when pseudo mode starts at epoch 51; its introduction is controlled by the epoch-level transition. In runs starting in pseudo mode, zero pseudo weight during warm-up leaves an augmentation-only minibatch loss, not the queue-based objective.

For linear evaluation, only the classifier heads are trainable. They are trained for 100 epochs with SGD, Nesterov momentum, batch size 128, learning rate 8.0, and no weight decay. The learning rate is reduced by factor 0.1 at epoch 80; the best validation Top-1 accuracy is reported.

5.4. Baselines and Ablations

The protocol provides a basis for testing which added training signals affect representation quality. Large sweeps already reported for HypSkeletonCLR are not repeated. The joint-only baseline was reproduced with the preprocessing, backbone, optimization settings, learning-rate schedule, and linear-evaluation protocol used for the main runs, as an implementation check. The tables retain the published 50-frame joint-only reference values and label them as reported.

We use the reported curvature-study results as the reference configuration. Higher joint-only accuracies were obtained through additional, model-specific pre-training parameter tuning whose configuration was not disclosed [7]. Those results are therefore excluded from the component comparison.

The HPL loss-introduction schedule differs in the cluster-count ablation, as described in Section 5.3. Feature dimension, curvature, optimizer, and linear-evaluation settings remain fixed.

The first ablation varies the prototype count K for joint-only **HypSkeletonCLR-H** and **HypSkeletonCLR-HPL** on the cross-view split, using $K \in \{3, 4, 5, 10, 20, 40, 60\}$.

The reported selection of K^* is based on linear Top-1 accuracy, with Top-5 as a complementary measure. Assignment balance, entropy, and stability are diagnostics; no numerical results for these quantities are used to justify the current selection.

The component comparison at $K^* = 40$ includes non-clustered HypSkeletonCLR and the C, H, PL, and HPL variants on both splits, with separate tables for joint-only and three-stream inputs. The cross-view HPL entry retains the $K = 40$ sweep result and therefore its 50-epoch augmentation-only phase. A separate comparison of the reproduced baseline and HypSkeletonCLR-H uses unprocessed 300-frame joint inputs at $K = 5$ on cross-view. Numerical results are stated in Chapter 6.

5.5. Evaluation Metrics

To assess recognition in these comparisons, the main metrics are Top-1 and Top-5 validation accuracy from linear evaluation. Top-1 is primary; Top-5 measures whether the correct class appears among the five highest-scoring predictions. Neither metric directly measures semantic relations between action classes.

Post-training Hierarchy Evaluation

To examine how the learned relations align with an external reference, we additionally analyze class hierarchies after training. The hierarchy evaluation reuses joint-only, 50-frame cross-view checkpoints at epoch 300 for the non-clustered baseline, C, and H, with $K = 40$ for the clustered variants. It extracts 32-dimensional embeddings using the frozen query encoder including its trained projection head, followed by the training normalization and Poincare mapping. Data augmentation is disabled. Labels are used to select the same 20 validation samples per class for every model (seed 42) and to form class prototypes for evaluation.

The quantitative comparison uses all 1,200 selected samples. Class prototypes are tangent-space means mapped back to the ball, and average linkage over hyperbolic prototype distances yields the predicted class trees. Rooted triplet agreement compares these trees with the stored supervised reference (`hypskeletonclr_ward`). The stored reference was obtained by Ward clustering of class prototypes from a supervised HypSkeletonCLR model. It represents model-derived action relations rather than an official NTU60 taxonomy. The metric is defined in Section 6.3.1, with the permutation control in Appendix A.3.

The projection comparison uses C and H with all 20 cached samples from each of eight predefined dressing classes, giving 160 samples per model. Each fit jointly projects these samples and all 40 learned checkpoint prototypes. Class-by-prototype heatmaps and projection links use nearest-prototype predictions in the original hyperbolic space,

without Sinkhorn rebalancing. The reference subtree preserves the stored hierarchy’s branching after removing omitted leaves and unary nodes. Ordinary t-SNE is presented in Section 6.3.2. The reference-tree and assignment diagnostics and the complementary hyperbolic t-SNE view are presented in Appendix A. All reuse cached embeddings and checkpoint parameters without additional training or encoder inference.

6. Results

Using the model names defined in Section 4.5 and the protocol from Chapter 5, we evaluate the proposed training signals in three parts. First, the ablation study of the prototype count K addresses RQ1. Second, classification experiments address RQ2 by comparing the linear separability of the learned representations. Third, the analysis of hierarchical structure addresses RQ3, separating metrics in the original embedding space from two-dimensional projections. The hierarchy evaluation compares the baseline, C, and H models on a shared subset of cross-view validation samples.

Following Chapter 5, classification performance is measured by linear evaluation on NTU60. Top-1 accuracy is the primary metric, and Top-5 accuracy is complementary. Differences in accuracy are reported in percentage points. Classification gains and semantic hierarchy are evaluated as distinct claims. Reported baseline accuracies are taken from Almeida Jordan [7]. The corrected-head three-stream cross-view baseline and the 300-frame baseline are reproduced locally and identified separately in the tables.

6.1. Ablation Study of the Prototype Count K

The study varies $K \in \{3, 4, 5, 10, 20, 40, 60\}$ for joint-only HypSkeletonCLR-H and HypSkeletonCLR-HPL on the cross-view split. Its purpose is to select K^* for the subsequent comparison and to determine whether the two variants favor the same tested prototype count.

Table 6.1 shows that both variants attain their highest Top-1 and Top-5 accuracies at $K = 40$. HypSkeletonCLR-H reaches 74.20% Top-1 and 95.35% Top-5, while HypSkeletonCLR-HPL reaches 72.41% and 94.68%, respectively. This agreement motivates $K^* = 40$ for the main component comparison. Performance does not increase monotonically with K , and the selected count should not be interpreted as the number of semantic action groups discovered by the model. The neighbouring tested counts, $K = 20$ and $K = 60$, are relatively far from $K = 40$, so the sweep does not establish a narrow optimum. One hypothesis is that $K = 40$ balances prototype capacity and assignment stability: $K = 20$ may combine distinct motion patterns, whereas $K = 60$ spreads the fixed minibatch’s assignment mass over more prototypes, potentially making their updates less stable. Changing K can also reorganize the jointly learned representations and assignments, rather than merely refining existing groups, so performance need not

Table 6.1.: Cluster-count ablation for joint-only H and HPL. All rows use the cross-view split. Accuracies are in percent; bold values identify the best result within each variant.

Model	K	Top-1	Top-5
HypSkeletonCLR, reported	-	70.29	93.98
HypSkeletonCLR-H	3	72.97	94.89
HypSkeletonCLR-H	4	72.97	94.82
HypSkeletonCLR-H	5	71.12	94.41
HypSkeletonCLR-H	10	71.66	94.72
HypSkeletonCLR-H	20	71.78	94.44
HypSkeletonCLR-H	40	74.20	95.35
HypSkeletonCLR-H	60	72.18	94.85
HypSkeletonCLR-HPL	3	71.84	94.28
HypSkeletonCLR-HPL	4	71.76	94.38
HypSkeletonCLR-HPL	5	69.78	94.00
HypSkeletonCLR-HPL	10	70.16	94.00
HypSkeletonCLR-HPL	20	69.24	93.74
HypSkeletonCLR-HPL	40	72.41	94.68
HypSkeletonCLR-HPL	60	70.76	94.25

vary smoothly. These explanations remain untested, and variability across training seeds may also contribute.

HypSkeletonCLR-H has higher Top-1 accuracy than HypSkeletonCLR-HPL at every tested count; the difference is 1.79 percentage points at $K = 40$. Pseudo-label feedback may pull different action classes together when they share a prototype, weakening class separation. This is a hypothesis about the gap, not a measured assignment-error rate. The HPL runs in this ablation use augmentation-only contrastive learning for the first 50 epochs before introducing the pseudo-label objective with a 20-epoch transition. Follow-up HPL experiments remove this epoch-level phase, as described in Section 5.3. The present comparison does not isolate the effect of that schedule, so the performance difference cannot be attributed to the warm-up without a matched schedule ablation. The cross-view HPL result retained in the component table below is the $K = 40$ result from this sweep.

6.2. Embedding Quality by Classification Performance

Having selected the prototype count, we next compare which combinations of training objectives support action recognition. Linear evaluation measures how well action

Table 6.2.: Ablation of clustering, hierarchy loss, and pseudo-label feedback at $K = 40$ for joint-only HypSkeletonCLR. Accuracies are in percent; bold values identify the best result within each split.

Variant	Split	Top-1	Top-5
HypSkeletonCLR baseline (reported)	cross-view	70.29	93.98
HypSkeletonCLR-C	cross-view	74.67	95.48
HypSkeletonCLR-H	cross-view	74.20	95.35
HypSkeletonCLR-PL	cross-view	71.04	94.16
HypSkeletonCLR-HPL	cross-view	72.41	94.68
HypSkeletonCLR baseline (reported)	cross-subject	64.40	90.27
HypSkeletonCLR-C	cross-subject	66.68	91.54
HypSkeletonCLR-H	cross-subject	67.80	91.76
HypSkeletonCLR-PL	cross-subject	68.20	91.78
HypSkeletonCLR-HPL	cross-subject	67.65	91.76

classes can be separated using a frozen encoder. The comparisons below therefore assess representation quality for recognition. They do not, on their own, measure the quality of a learned hierarchy. The component comparison uses $K = 40$; the separate 300-frame experiment uses $K = 5$.

6.2.1. Component Ablation: Joint-Only Inputs

Table 6.2 compares the non-clustered baseline with C, H, PL, and HPL on both evaluation splits.

All four clustered variants exceed the non-clustered reference in both Top-1 and Top-5 accuracy on each split. On cross-view, HypSkeletonCLR-C performs best with 74.67% Top-1, exceeding the 70.29% reference by 4.38 percentage points. On cross-subject, HypSkeletonCLR-PL performs best with 68.20%, exceeding the 64.40% reference by 3.80 percentage points.

The ranking changes across splits. In the available runs, H differs from C by -0.47 percentage points in Top-1 on cross-view and $+1.12$ percentage points on cross-subject. PL differs from C by -3.63 and $+1.52$ percentage points, respectively. The full HPL variant exceeds the baseline but does not achieve the highest accuracy on either split. These observations support the usefulness of clustered training for classification in the reported joint-only experiments, while providing no evidence of a uniform classification gain from adding each further component.

A possible explanation for the split-dependent ranking is that group-level constraints reduce sensitivity to individual execution styles, benefiting cross-subject transfer. On

Table 6.3.: Ablation of clustering, hierarchy loss, and pseudo-label feedback at $K = 40$ for 3s-HypSkeletonCLR. Accuracies are in percent; bold values identify the best result within each split.

Variant	Split	Top-1	Top-5
3s-HypSkeletonCLR baseline (reported, original projection heads)	cross-view	80.39	97.01
3s-HypSkeletonCLR baseline (reproduced, corrected heads)	cross-view	79.17	96.75
3s-HypSkeletonCLR-C	cross-view	78.69	96.68
3s-HypSkeletonCLR-H	cross-view	78.21	96.60
3s-HypSkeletonCLR-PL	cross-view	72.25	95.03
3s-HypSkeletonCLR-HPL	cross-view	70.93	94.43
3s-HypSkeletonCLR baseline (reported, original projection heads)	cross-subject	73.40	94.13
3s-HypSkeletonCLR-C	cross-subject	72.16	93.95
3s-HypSkeletonCLR-H	cross-subject	72.32	94.03
3s-HypSkeletonCLR-PL	cross-subject	70.75	92.71
3s-HypSkeletonCLR-HPL	cross-subject	70.15	92.33

cross-view, the same constraints may instead merge subtle action differences that C preserves. This interpretation is consistent with the ranking but would require an error analysis to test.

The tables do not report variability across independent training seeds. Consequently, the numerical differences describe the available runs and do not establish statistical significance.

6.2.2. Component Ablation: Three-Stream Inputs

The joint-only analysis leaves open how the objectives behave when multiple skeleton representations are combined. Table 6.3 reports the corresponding comparison for joint, motion, and bone inputs. The reported baseline has the highest Top-1 and Top-5 accuracies on both splits. Among the component variants, 3s-HypSkeletonCLR-C performs best on cross-view, while 3s-HypSkeletonCLR-H performs best on cross-subject.

One hypothesis for the three-stream results is that a shared prototype assignment cannot fully preserve the complementary distinctions in joint, motion, and bone features. Pseudo-label attraction could amplify such grouping errors, explaining why PL and HPL fall further behind C and H. The joint-only objective settings may also require retuning for three streams.

Reported and corrected baselines. The published baseline uses the original projection heads, whereas the component variants use the corrected architecture (Section 5.2). The reproduced cross-view baseline with corrected heads reaches 79.17% Top-1 and 96.75% Top-5, exceeding C’s 78.69% Top-1 by 0.48 percentage points. This preserves the baseline’s lead over the component variants on cross-view, but the difference from the published result cannot be attributed solely to correcting the heads. A corrected cross-subject rerun was omitted because of time constraints; the cross-view rerun had not changed the qualitative conclusion that the baseline exceeded the clustered variants.

6.2.3. Evaluation on Unprocessed 300-Frame Inputs

To test whether the joint-only gains extend beyond standard 50-frame preprocessing, we compare the reproduced baseline and H on unprocessed 300-frame NTU60 cross-view inputs. These may retain both useful temporal detail and irrelevant variation. Both runs use a pre-training learning rate of 0.1 and 300 epochs of linear evaluation with cosine decay from learning rate 8.0; H uses $K = 5$.

Table 6.4 reports 72.39% Top-1 accuracy for H versus 66.42% for the baseline, a gain of 5.97 percentage points. Top-5 rises from 92.69% to 95.14%, a gain of 2.45 points. Prototype objectives encourage features shared across samples beyond instance discrimination [9]. Clustering and the hierarchy constraint may therefore regularize learning towards shared action patterns, reducing sensitivity to irrelevant variation. This could explain H’s advantage, but the experiment does not isolate this mechanism or either loss’s contribution.

H exceeds its 50-frame $K = 5$ result of 71.12% by 1.27 percentage points, while the baseline falls 3.87 points below the reported 50-frame reference of 70.29%. Input processing and optimization settings both differ, so these comparisons do not isolate sequence length. The result supports a benefit in this setting without establishing general robustness.

Table 6.4.: Robustness comparison on unprocessed 300-frame sequences for the joint-only cross-view setting. HypSkeletonCLR-H uses $K = 5$ and is trained without pseudo-label positives. Accuracies are in percent.

Setting	Top-1	Top-5
Reproduced baseline	66.42	92.69
HypSkeletonCLR-H	72.39	95.14

6.3. Hierarchical Structure in Embeddings

The preceding comparisons assess class separability, leaving the relations between classes untested. We therefore examine whether the hierarchy objective helps the embeddings follow a reference tree of action classes. We compare the non-clustered baseline, HypSkeletonCLR-C, and HypSkeletonCLR-H after 300 training epochs. All use joint-only, 50-frame inputs on the cross-view split; C and H use $K = 40$. The analysis requires no further training.

6.3.1. Metrics

Each model is evaluated on the same 20 validation samples from each of the 60 classes, giving 1,200 samples. Seed 42 fixes this selection. We compute an average embedding for each class and group the classes using their hyperbolic distances. This gives one class tree per model, following the procedure in Section 5.5.

The reference tree was built by Ward clustering of class averages from a supervised HypSkeletonCLR model. It is a learned reference, not an official hierarchy of NTU60 actions. The score therefore measures agreement with this particular tree.

Triplet agreement considers three action classes at a time and asks whether the model tree and the reference group the same pair together. The score is the percentage of matching choices. Triples with no clear reference pair are excluded. The exact definition and tie rules are given in Appendix A.3.

Table 6.5.: Triplet agreement with the supervised reference. Values are percentages.

Model	Agreement
HypSkeletonCLR baseline	73.61
HypSkeletonCLR-C	72.29
HypSkeletonCLR-H	72.95

All three models exceed the approximately 33.3% agreement obtained by shuffling the reference labels (Appendix A.3). In this evaluation, the explicit prototype hierarchy loss did not improve reference-tree agreement over the non-clustered baseline. H exceeds C by 0.665 percentage points (from unrounded values), but this small gap does not establish a reliable improvement.

The classification ranking is reversed: C exceeds H in cross-view Top-1 (74.67% versus 74.20%). H regularizes learned prototypes, whereas this metric compares class averages with a supervised tree. Neither class labels nor that tree guide the loss, so improved reference agreement is not guaranteed. Each score uses one checkpoint and one sample selection; Chapter 7 discusses these limits.

6.3.2. Projections

The agreement scores summarize each class tree, but do not show how individual samples and learned prototypes are arranged. To inspect these relationships, Figure 6.1 shows ordinary t-SNE for C and H. To keep the plots readable, we use eight dressing actions: putting on and taking off a jacket, a shoe, glasses, and a hat/cap (NTU60 classes 14–21). These classes were selected before inspecting the plots. They provide familiar action pairs whose positions and prototype assignments can be compared.

Each panel uses the same 20 selected samples per class, giving 160 samples, together with all 40 learned prototypes. These prototypes are parameters learned during training. They differ from the class averages used to build the trees in the metric analysis. Class labels determine the selected samples and their colors; they are not used to fit the projection.

The samples and prototypes are mapped to the tangent space at the origin before fitting t-SNE. The fit uses seed 42, perplexity 30, PCA initialization, an automatic learning rate, and 1,000 iterations. Each model is fitted separately, so a position in one panel cannot be compared directly with the same position in the other.

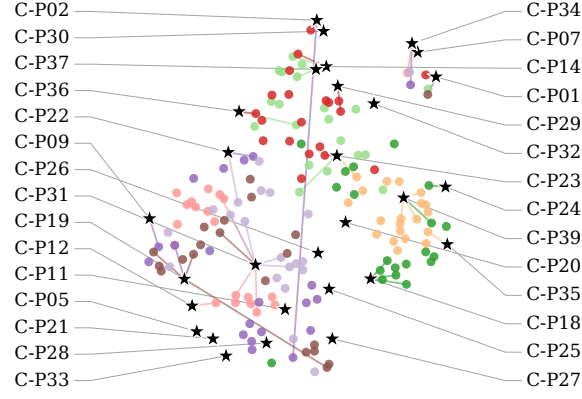
Circles show samples, and black stars show prototypes assigned to at least one selected sample. All 40 prototypes take part in the fit, but unused ones are hidden in the figure. Thin colored lines connect the first five fixed samples of each class to their nearest prototype, measured in the original hyperbolic space. No Sinkhorn rebalancing is applied to these predictions. Gray lines connect the prototype names to their stars. The names identify prototypes within each model; the same number in C and H does not mean that the two prototypes represent the same group.

C and H show similar qualitative class mixing: jacket actions are comparatively compact in both, while headwear and shoe actions overlap. These independently fitted views show no clear qualitative improvement from H. A possible reason for this mixing is that putting on and taking off the same item involve similar poses, with differences in temporal order that may not be well separated in the embeddings. Some colored lines cross visible groups. This happens because a sample’s nearest prototype in the original space need not be nearest in the two-dimensional plot.

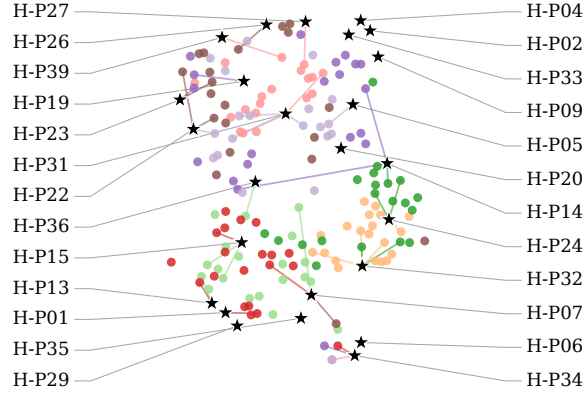
These views help show where classes mix and which prototypes their samples use. They do not establish a nested action hierarchy. As explained in Section 2.5, the distance between visible groups in t-SNE is not sufficient evidence for such a hierarchy. The reference tree and full assignment distributions are provided in Appendix A.1; the hyperbolic t-SNE view is in Appendix A.2.

Dressing actions | t-SNE in the origin tangent space

(a) C: clustering



(b) H: clustering + hierarchy



Circles: samples. Stars: learned prototypes. Gray lines: ID callouts.
Colored lines: original-space assignments (5 samples/class); checkpoint-local IDs.

- 14 put on jacket
- 15 take off jacket
- 16 put on a shoe
- 17 take off a shoe
- 18 put on glasses
- 19 take off glasses
- 20 put on a hat/cap
- 21 take off a hat/cap

Figure 6.1.: Dressing actions in ordinary t-SNE: C (top) and H (bottom). Each fit uses 160 samples and all 40 learned prototypes. Black stars show occupied prototypes; colored lines show original-space assignments for five samples per class. The panels are fitted separately.

7. Discussion

7.1. Interpretation of Results

The experiments show that clustered training can improve action recognition in the evaluated settings, while the hierarchy analysis separately evaluates class relationships.

For RQ1, both H and HPL favor $K = 40$ among the tested counts (Table 6.1). Their agreement provides a useful basis for selecting the count in the later comparison. Performance does not increase steadily with K , which makes testing the count more informative than simply adding more prototypes. H exceeds HPL throughout this sweep.

For RQ2, clustering can improve recognition in this pipeline, but adding hierarchy learning and pseudo-label feedback gives mixed benefits. Their full combination is not consistently best (Table 6.2). All joint-only variants exceed the reported reference; this does not establish that every variant improves on a matched local baseline.

One possibility is that pseudo-positive feedback reinforces imperfect cluster assignments, while constraints on prototype relations do not always support class discrimination. Combining these signals may therefore be less beneficial than using a subset. These mechanisms remain untested. Pseudo-label feedback also replaces queued keys with current-minibatch comparisons, so the comparison pool changes alongside the positive-pair weights.

H also exceeds the reproduced baseline in the separate 300-frame experiment (Table 6.4). This extends the evidence for a classification benefit to that input setting. It does not isolate sequence length as the cause of the difference from the 50-frame results.

For RQ3, all three evaluated models agree with the supervised reference tree well above the permutation baseline. H slightly exceeds C, but the explicit hierarchy objective did not improve agreement over the non-clustered baseline.

Sample assignments use one prototype level, with no explicit parent-child assignments or objectives across levels. Prototype geometry can still encode deeper relations implicitly. Missing explicit nesting is therefore a possible limitation, not a restriction to one-level trees. The hierarchy loss uses self-derived prototype affinities and affects the encoder indirectly, whereas triplet agreement evaluates class averages against a supervised model’s reference. Better prototype organization need not improve this metric. These explanations remain untested.

The normalization is inherited from HypSkeletonCLR. Proposition B.2.1 establishes that it fixes the distance of sample embeddings from the origin, restricting their radial variation. Hierarchical grouping through angular relationships remains possible. Whether this constraint contributed to the observed negative hierarchy results remains untested.

7.2. Limitations

The strength of these interpretations is limited by the available runs. Variation across training seeds was not measured, and the hierarchy comparison uses one trained model and one fixed selection of 20 samples per class for each model. The small H-C gap may change with another run or selection. The permutation spread describes shuffled class identities, not uncertainty in the model scores.

The cross-view validation split guided the choice of K and was reused for hierarchy evaluation, making that analysis exploratory. The supervised reference is not a ground-truth action taxonomy. It may reflect its source model as well as action meaning. Class averaging and average linkage also influence the extracted trees.

Limited compute restricted hyperparameter search and repeated training runs. Beyond the prototype-count ablation, loss weights, temperatures, and training schedules were not exhaustively tuned for each model variant and input stream. The settings may therefore be suboptimal, particularly when combining objectives or using three-stream inputs. Broader tuning and independent training seeds could improve the reliability of these comparisons.

The published three-stream reference uses the original projection heads, whereas the reproduced cross-view baseline uses the corrected architecture shared by the component variants (Table 6.3; Section 5.2).

The compact ST-GCN backbone may also limit representation quality, although its stacked temporal convolutions capture dependencies beyond neighbouring frames. Linear evaluation uses pooled Euclidean backbone features, bypassing the projection head and hyperbolic mapping, so it does not directly assess classification in the learned hyperbolic space. The effects of these choices on performance were not isolated.

The dressing plots cover eight classes, and projection changes distances. Visible groups and assignment links are therefore inspection tools rather than proof of a complete action hierarchy.

The evaluation provides a way to examine action relationships beyond classification accuracy. Triplet agreement compares class trees, while prototype assignments and focused projections show how individual samples are grouped. These complementary analyses reuse fixed model parameters and previously extracted embeddings, requiring little additional computation.

8. Conclusion

8.1. Summary

This thesis developed and evaluated an extension of HypSkeletonCLR with balanced prototype assignments, a hierarchy objective, and cluster-based pseudo-label feedback. The experiments showed higher classification scores in the evaluated settings and provided a practical way to examine the relationships learned between action classes.

For RQ1, both H and HPL achieved their highest accuracy at $K = 40$ among the tested counts on the joint-only cross-view split. We therefore used $K = 40$ for the component comparison.

For RQ2, C leads for joint-only cross-view and PL for joint-only cross-subject, while the full combination is not consistently best. H also outperforms the reproduced baseline in the separate 300-frame experiment.

For RQ3, the explicit hierarchy objective did not improve agreement with the supervised reference tree over the non-clustered baseline in this evaluation. Thus, the observed recognition gains do not establish improved semantic hierarchy. But note, optimizing prototype relations need not improve a class-level reference-tree metric.

The analysis also identifies the fixed sample radius imposed by the inherited normalization, motivating a controlled comparison with variable sample norms.

8.2. Future Work

Repeated training seeds and sample selections could test the stability of these findings. An independent action hierarchy and evaluation data held apart from model selection would strengthen the hierarchy analysis.

A HypHC-inspired head trained on fixed features could test hierarchy learning separately from changes in recognition features [20]. HCSC-inspired nested prototype levels could supply explicit coarse-to-fine assignments and make contrastive pair selection depend on more than one grouping level [24]. Whether either improves skeleton representations remains to be tested.

Further comparisons could allow controlled variation in sample embedding norms, evaluate more expressive encoders under matched protocols, and compare Euclidean and hyperbolic classifier heads on the same frozen projected representations with comparable parameter counts and tuning budgets.

A. Supplementary Embedding Analysis

This appendix supplements the hierarchy analysis in Section 6.3. The dressing analyses use the same trained C and H models, 160 selected samples per model, and all 40 learned prototypes; the permutation control uses the full 60-class comparison.

A.1. Reference Structure and Prototype Assignments

The dressing reference is an induced subtree of the supervised reference hierarchy. Omitted leaves and unary nodes are removed while the remaining branching is retained.

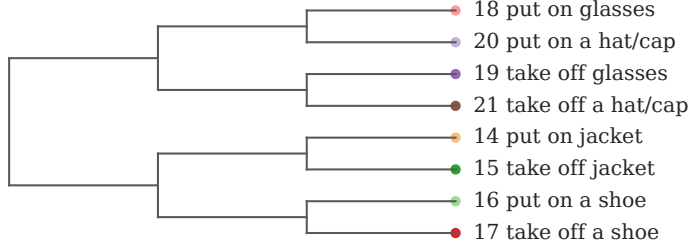
Each selected sample is assigned to its nearest learned prototype in the original hyperbolic space, before class filtering or projection. This is the maximum-probability prediction under the model’s distance-based prototype distribution; Sinkhorn rebalancing is not applied. Numerical ties use the same tolerances as the triplet evaluation and would be shown separately in the assignment tables, with their links omitted. There are no tied sample assignments in the dressing subset.

Figure A.1 pairs the dressing reference tree with class-by-prototype assignment percentages. Within the headwear branch, the reference pairs putting on glasses with putting on a hat, and pairs their removal actions. Jacket and shoe actions form two further pairs under the other major branch. The heatmaps retain all 20 samples in each row, including those whose links are omitted from the projection for readability.

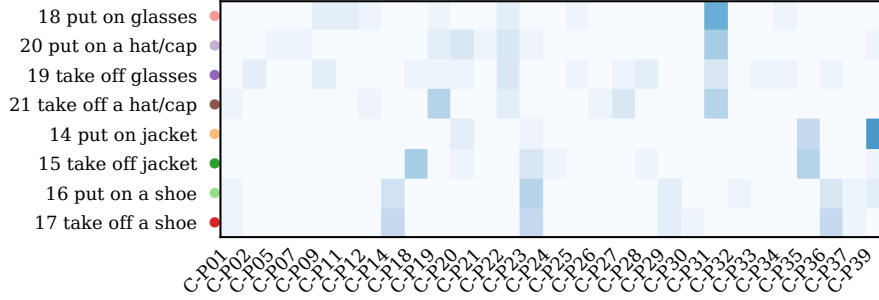
The dressing samples occupy 29 of 40 prototypes in C and 25 in H. H therefore distributes this subset over fewer distinct prototypes, but occupancy alone does not establish more concentrated per-class assignments or better hierarchy. Assignments provide partial correspondence with the reference: putting on glasses and putting on a hat share the most frequent prototype within each model. In C, 10/20 and 7/20 samples are assigned to prototype 31; in H, 13/20 and 8/20 are assigned to prototype 31. Prototype indices are specific to each model: the repeated index does not identify a matched cluster across models. Other reference pairs need not share their most frequent prototype. Putting on and taking off a jacket, for example, favor prototypes 39 and 18 in C, and prototypes 32 and 14 in H, respectively. Each dressing class occupies 4–13 prototypes in C and 3–14 in H. These flat assignment distributions describe class sharing and fragmentation; they do not by themselves establish nested hierarchy or improvement from the hierarchy objective.

Dressing actions

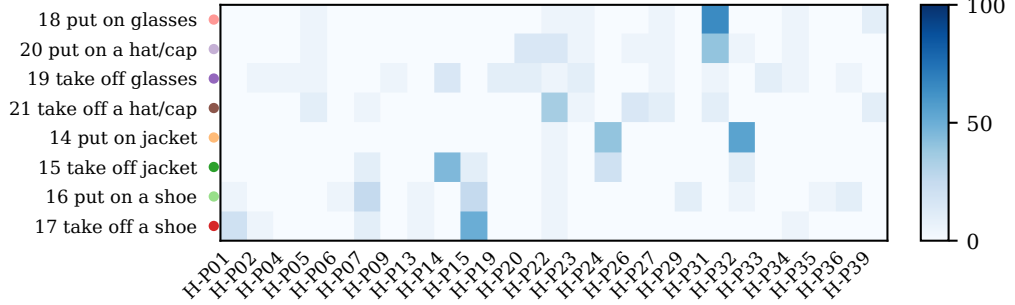
(a) Pruned supervised reference (topology only)



(b) C: predicted assignments (%)



(c) H: predicted assignments (%)



20 samples/class; each row sums to 100%. Unoccupied columns omitted.
 Prototype IDs are local to each checkpoint; ties retain their own column.

Figure A.1.: Dressing actions: pruned supervised reference and predicted prototype assignments for C and H. Each heatmap row contains 20 samples and sums to 100%; both use the same color scale. Unoccupied prototype columns are omitted. Prototype indices are specific to each model, and tree edge lengths carry no distance information.

A.2. Hyperbolic t-SNE

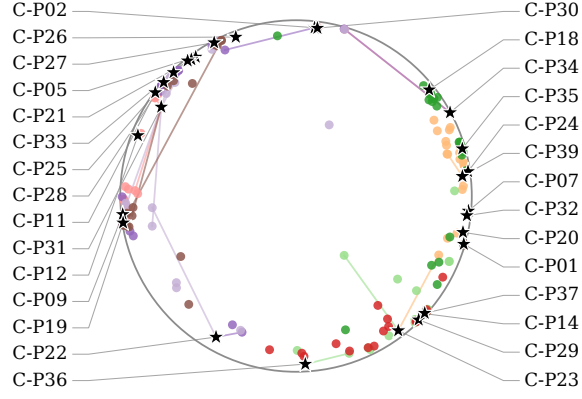
The assignments show which samples share a prototype. To inspect their arrangement alongside the ordinary t-SNE view, we also consider a projection in hyperbolic space. This complementary view uses hyperbolic neighborhood affinities and a Poincaré-disk output, allowing us to inspect class mixing without first imposing Euclidean tangent-space affinities. It remains a projection diagnostic, not a test of hierarchy depth.

Figure A.2 uses Poincare inputs, hyperbolic nearest-neighbor affinities, and Barnes–Hut optimization in a unit Poincare disk. Settings are perplexity 30, learning rate 0.1, 250 exaggeration iterations, and 750 main iterations. The sample colors, prototype IDs, and original-space assignment links match the ordinary t-SNE view in Figure 6.1. Circles show samples and stars show occupied prototypes; unused prototypes are hidden after fitting. Colored links annotate predictions for five samples per class, while gray lines identify prototype stars.

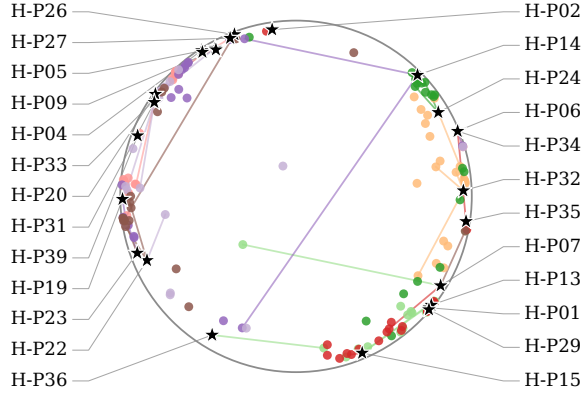
Both models place many samples and prototypes around the disk perimeter, with overlapping class colors. The sample-only proportions at Euclidean radius $r \geq 0.98$ are 32.5% for C and 24.4% for H (52 and 39 of 160 samples). H’s smaller boundary fraction describes this projection; it does not measure hierarchy depth in the original embeddings. The visible groupings and assignment links do not establish a complete recovery of the reference branching.

Dressing actions | Hyperbolic t-SNE

(a) C: clustering



(b) H: clustering + hierarchy



Circles: samples. Stars: learned prototypes. Gray lines: ID callouts.
Colored lines: original-space assignments (5 samples/class); checkpoint-local IDs.

- | | |
|--------------------|-----------------------|
| 14 put on jacket | 18 put on glasses |
| 15 take off jacket | 19 take off glasses |
| 16 put on a shoe | 20 put on a hat/cap |
| 17 take off a shoe | 21 take off a hat/cap |

Figure A.2.: Hyperbolic t-SNE for C (top) and H (bottom), each fitted with 160 dressing samples and all 40 learned prototypes. Colored links show original-space predictions for five samples per class. Only occupied prototypes are displayed. Disk radius does not measure hierarchy depth in the original space.

A.3. Triplet Agreement and Permutation Control

Construction of the trees. To complement the projections quantitatively, the evaluation in Section 6.3.1 uses the same 20 validation samples from each of the 60 action classes for the baseline, C, and H. This gives 1,200 samples per model, selected with seed 42. For each class, we average its embeddings in the tangent space at the origin and map the mean back to the Poincaré ball. These class averages are separate from the trainable cluster prototypes.

We compute hyperbolic distances between the class averages and use average linkage to construct each model tree, merging groups with the smallest average distance between their members. The reference tree is obtained by Ward clustering of class averages from a supervised HypSkeletonCLR model.

Pair selection and agreement. Consider an unordered triple $t = \{a, b, c\}$ of distinct action classes. Its three possible pairs are $\{a, b\}$, $\{a, c\}$, and $\{b, c\}$. The lowest common ancestor (LCA) of two classes is the deepest tree node containing both classes. Depth counts the levels from the root. The reference selects the pair with the deepest common ancestor:

$$p_R(t) = \arg \max_{\{i,j\} \subset t} \text{depth}(\text{LCA}_R(i, j)).$$

A reference triple is resolved when this pair is unique. Triples without a uniquely closer pair are unresolved and excluded.

For the model tree, let $h_M(i, j)$ be the merge height at which classes i and j first belong to the same group. This is their cophenetic height. The model selects

$$p_M(t) = \arg \min_{\{i,j\} \subset t} h_M(i, j).$$

Thus, the comparison asks which pair joins first in each tree. It does not simply select the smallest distance between the original class embeddings.

For a hypothetical example, let a denote putting on a jacket, b taking off a jacket, and c kicking. The relationship $((a, b), c)$ groups the jacket actions together before joining them with kicking. If the reference selects this pair, a model grouping a with b earns one point; either other pair earns zero. This illustrates the rule and does not report an observed reference triple.

Let $\mathcal{T}_R$ contain all resolved reference triples. The agreement is the fraction for which the selected pairs match, with an indicator equal to one for a match and zero otherwise:

$$\text{TA}(M, R) = \frac{1}{|\mathcal{T}_R|} \sum_{t \in \mathcal{T}_R} \mathbf{1}[p_M(t) = p_R(t)].$$

A numerically tied model prediction receives zero credit and remains in the denominator. Numerical ties are assessed using relative tolerance 10^{-6} and absolute tolerance 10^{-8} . Results are reported as percentages.

Calculation of the reported score. For 60 classes, there are $\binom{60}{3} = 34,220$ possible triples. The reference resolves 30,238 of them; the remaining 3,982 are excluded. None of the three model trees yields tied predictions at the stated tolerance. HypSkeletonCLR-H selects the reference pair for 22,060 of the eligible triples, giving

$$100\% \times \text{TA}(H, R) = 100\% \times \frac{22,060}{30,238} \approx 72.95\%.$$

The score therefore means that H and the reference choose the same pair in about 73 out of every 100 resolved class triples.

Interpretation. The score measures agreement with a particular learned class tree, rather than classification accuracy or recovery of an official NTU60 taxonomy. Hierarchical clustering constructs a tree from the embeddings, so agreement alone does not establish that their original geometry is strongly hierarchical. The result depends on the reference, class averaging, and tree construction.

For each resolved triple, uniformly guessing among the three pairs gives an expected agreement of $1/3$, or approximately 33.3%. The permutation control below provides a comparison using the actual trees.

The permutation control keeps the trees fixed and shuffles the reference leaf labels 100 times, preserving its topology. Seed 42 produces the same permutations for all models. Resolved reference triples are recomputed after each shuffle. Mean agreement is approximately 33.3%, with a standard deviation of about 1.58 percentage points. This describes agreement when class identities are disconnected from the reference tree. The spread is not a confidence interval for a model score or for the H-C difference.

Table A.1.: Agreement under 100 reference-label permutations. Means are in percent; standard deviations (SD) are in percentage points and describe variation across permutations.

Model	Permutation mean	Permutation SD
HypSkeletonCLR baseline	33.32	1.58
HypSkeletonCLR-C	33.31	1.58
HypSkeletonCLR-H	33.37	1.58

A.4. t-SNE Definition

Definition A.4.1 (t-distributed Stochastic Neighbor Embedding). t-distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction method primarily used to visualize high-dimensional data [13]. Given representations

$$z_1, \dots, z_N \in \mathbb{R}^d,$$

t-SNE first constructs a probability distribution that describes local neighbourhood relations in the original representation space. For $i \neq j$, define

$$p_{j|i} = \frac{\exp(-\|z_i - z_j\|^2 / (2\sigma_i^2))}{\sum_{k \neq i} \exp(-\|z_i - z_k\|^2 / (2\sigma_i^2))}.$$

The bandwidth σ_i determines the size of the neighbourhood around z_i and is chosen according to a prescribed perplexity. The conditional probabilities are symmetrized as

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}.$$

For visualization, t-SNE assigns each z_i a point

$$y_i \in \mathbb{R}^2$$

and defines similarities in the visualization space using a Student t -distribution,

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq \ell} (1 + \|y_k - y_\ell\|^2)^{-1}}.$$

The low-dimensional coordinates are found by minimizing

$$\mathcal{L}_{\text{tSNE}} = D_{\text{KL}}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

with respect to $y_1, \dots, y_N$.

B. Riemannian Geometry and the Exponential Map

This appendix supplies the geometric definitions underlying Section 2.3 and derives the exponential map used to embed network outputs in the Poincaré ball. Throughout, $c > 0$, the curvature is $-c$, and $\|\cdot\|$ denotes the Euclidean norm. Riemannian norms are marked explicitly by a metric subscript.

B.1. Riemannian Manifolds and Their Exponential Maps

Definition B.1.1 (Smooth manifold). A smooth d -dimensional manifold is a Hausdorff, second-countable topological space M with a smooth atlas. Each chart is a homeomorphism $\varphi : U \rightarrow V$, where $U \subseteq M$ and $V \subseteq \mathbb{R}^d$ are open; the chart domains cover M , and all transition maps between overlapping charts are smooth.

Thus, a manifold admits local Euclidean coordinates, although its global shape need not be Euclidean. The tangent space $T_p M$ consists of the velocities at p of smooth curves through p , with velocities identified when their coordinate derivatives agree. It is a d -dimensional vector space [25].

Definition B.1.2 (Riemannian manifold). A Riemannian manifold is a pair (M, g) , where M is a smooth manifold and g assigns an inner product

$$g_p : T_p M \times T_p M \longrightarrow \mathbb{R}$$

to every $p \in M$. Each g_p is symmetric, bilinear, and positive definite. The assignment is smooth: in every coordinate chart, the coefficients $g_{ij}(p) = g_p(\partial_i, \partial_j)$ are smooth functions of p .

The metric defines tangent-vector lengths and curve lengths by

$$\|v\|_{g_p} = \sqrt{g_p(v, v)}, \quad L_g(\gamma) = \int_a^b \|\dot{\gamma}(t)\|_{g_{\gamma(t)}} dt.$$

On a connected manifold, the induced distance is the infimum of these lengths over piecewise smooth curves joining the two points. In particular, the metric tensor g_p measures tangent vectors at one point, while the induced distance compares points of M [26].

Geodesics. The metric determines a unique torsion-free, metric-compatible connection, the Levi-Civita connection ∇ . A geodesic is a curve satisfying

$$\nabla_{\dot{\gamma}} \dot{\gamma} = 0.$$

Its velocity is parallel along the curve, and its Riemannian speed is constant. Geodesics are locally length-minimizing; they need not minimize length globally on an arbitrary Riemannian manifold [26].

Definition B.1.3 (Riemannian exponential map). For $p \in M$ and $v \in T_p M$, let $\gamma_{p,v}$ be the unique maximal geodesic with

$$\gamma_{p,v}(0) = p, \quad \dot{\gamma}_{p,v}(0) = v.$$

Let $\mathcal{U}_p \subseteq T_p M$ contain precisely those vectors for which this geodesic is defined throughout $[0, 1]$. The Riemannian exponential map at p is

$$\text{Exp}_p : \mathcal{U}_p \longrightarrow M, \quad \text{Exp}_p(v) = \gamma_{p,v}(1).$$

The set $\mathcal{U}_p$ contains a neighbourhood of zero. On a geodesically complete manifold it equals all of $T_p M$. The map follows the geodesic with initial velocity v for one unit of time; this segment has length $\|v\|_{g_p}$ [26].

B.2. Derivation for the Poincaré Ball

These definitions specify what the exponential map must satisfy. We now derive its form for the Poincaré ball by identifying the metric and solving the associated geodesic equation.

B.2.1. The Ball as a Riemannian Manifold

Consider

$$\mathbb{D}_c^d = \{x \in \mathbb{R}^d : c\|x\|^2 < 1\}, \quad g_x^c(u, w) = (\lambda_x^c)^2 \langle u, w \rangle, \quad \lambda_x^c = \frac{2}{1 - c\|x\|^2}. \quad (\text{B.1})$$

Since $\mathbb{D}_c^d$ is open in $\mathbb{R}^d$, it inherits the Hausdorff and second-countability properties, and the identity chart gives it a smooth manifold structure. Its tangent spaces identify naturally with $\mathbb{R}^d$. The factor λ_x^c is smooth and strictly positive on the ball, so g^c is a smooth positive-definite metric. Thus $(\mathbb{D}_c^d, g^c)$ is a Riemannian manifold.

The coordinate identification does not identify the Riemannian norm with the standard Euclidean norm:

$$\|v\|_{g_x^c} = \lambda_x^c \|v\|, \quad \|v\|_{g_0^c} = 2\|v\|. \quad (\text{B.2})$$

This factor of two determines the scaling in the exponential map. Consequently, $d_c(0, \text{Exp}_0^c(v)) = 2\|v\|$; the Euclidean unit-normalized network outputs used here therefore map to hyperbolic radial distance 2 from the origin, as stated in Proposition B.2.1. To obtain the map itself, we next determine the geodesics of this metric.

B.2.2. Geodesic Equation

In coordinates, the geodesic equation is

$$\ddot{\gamma}^k + \sum_{i,j=1}^d \Gamma_{ij}^k(\gamma) \dot{\gamma}^i \dot{\gamma}^j = 0, \quad k = 1, \dots, d,$$

where the Christoffel symbols of the Levi-Civita connection are

$$\Gamma_{ij}^k = \frac{1}{2} \sum_{\ell=1}^d g^{k\ell} (\partial_i g_{j\ell} + \partial_j g_{i\ell} - \partial_\ell g_{ij}).$$

Here, $(g^{k\ell})$ is the inverse metric matrix [26]. Substituting $g_{ij} = (\lambda_x^c)^2 \delta_{ij}$ and using

$$\partial_i \log \lambda_x^c = \frac{2cx_i}{1 - c\|x\|^2}$$

gives

$$\Gamma_{ij}^k(x) = \frac{2c}{1 - c\|x\|^2} (x_i \delta_{jk} + x_j \delta_{ik} - x_k \delta_{ij}),$$

where δ_{ij} is the Kronecker delta. Consequently, the geodesic equation in vector form is

$$\ddot{\gamma} + \frac{4c\langle \gamma, \dot{\gamma} \rangle}{1 - c\|\gamma\|^2} \dot{\gamma} - \frac{2c\|\dot{\gamma}\|^2}{1 - c\|\gamma\|^2} \gamma = 0. \quad (\text{B.3})$$

B.2.3. Exponential Map at the Origin

For the embedding construction, the required case is a geodesic starting at the origin. We therefore specialize the equation to this setting.

Let $v \neq 0$, set $a = \|v\|$, and let $e = v/a$. It suffices to construct a radial solution with initial position zero and velocity v , because the geodesic initial-value problem has a unique solution. Seek a solution of Equation (B.3) of the form $\gamma(t) = r(t)e$. Substitution reduces the full equation to

$$\ddot{r} + \frac{2cr}{1-cr^2}\dot{r}^2 = 0, \quad r(0) = 0, \quad \dot{r}(0) = a. \quad (\text{B.4})$$

A solution of this scalar equation therefore satisfies the full geodesic equation with the required initial position and velocity.

Multiplying the equation by $(1-cr^2)^{-1}$ yields

$$\frac{d}{dt} \left(\frac{\dot{r}}{1-cr^2} \right) = 0.$$

The initial conditions imply $\dot{r}/(1-cr^2) = a$. Separating variables and integrating from zero to t gives

$$\int_0^{r(t)} \frac{ds}{1-cs^2} = at, \quad \frac{1}{\sqrt{c}} \operatorname{artanh}(\sqrt{c}r(t)) = at.$$

Solving for $r(t)$, we obtain

$$r(t) = \frac{1}{\sqrt{c}} \tanh(\sqrt{c}at), \quad \gamma_{0,v}(t) = \tanh(\sqrt{c}\|v\|t) \frac{v}{\sqrt{c}\|v\|}.$$

This curve stays in the ball for every finite t , and uniqueness of the geodesic initial-value problem identifies it with $\gamma_{0,v}$. Evaluating at $t = 1$ gives

$$\operatorname{Exp}_0^c(v) = \begin{cases} \tanh(\sqrt{c}\|v\|) \frac{v}{\sqrt{c}\|v\|}, & v \neq 0, \\ 0, & v = 0. \end{cases} \quad (\text{B.5})$$

For $v = 0$, the geodesic is constant; the same value follows by continuity. Equation (B.5) is precisely the map in Definition 2.3.3 and agrees with [6, Equation (11)].

B.2.4. Consequences of Unit Normalization

Proposition B.2.1 (Fixed sample radius and angular reduction). *Let $c > 0$ and let $\tilde{h}_i, \tilde{h}_j \in \mathbb{R}^d$ satisfy $\|\tilde{h}_i\|_2 = \|\tilde{h}_j\|_2 = 1$. Set $z_\ell = \text{Exp}_0^c(\tilde{h}_\ell)$ for $\ell \in \{i, j\}$ and $\rho_{ij} = \langle \tilde{h}_i, \tilde{h}_j \rangle$. Then*

$$d_c(0, z_i) = d_c(0, z_j) = 2,$$

and

$$d_c(z_i, z_j) = \frac{1}{\sqrt{c}} \operatorname{arcosh}\left(1 + \sinh^2(2\sqrt{c})(1 - \rho_{ij})\right).$$

Thus the pairwise distance is strictly decreasing in the cosine similarity ρ_{ij} .

Proof. Write $s = \sqrt{c} > 0$ and $t = \tanh s$. Equation (B.5) gives $z_\ell = (t/s)\tilde{h}_\ell$ and $c\|z_\ell\|^2 = t^2$. The distance formula in Section 2.3 therefore yields

$$d_c(0, z_\ell) = \frac{1}{s} \operatorname{arcosh}\left(1 + \frac{2t^2}{1-t^2}\right) = \frac{1}{s} \operatorname{arcosh}(\cosh(2s)) = 2.$$

Since $\|z_i - z_j\|^2 = 2(t^2/s^2)(1 - \rho_{ij})$,

$$\frac{2c\|z_i - z_j\|^2}{(1 - c\|z_i\|^2)(1 - c\|z_j\|^2)} = \frac{4t^2}{(1 - t^2)^2}(1 - \rho_{ij}) = \sinh^2(2s)(1 - \rho_{ij}).$$

Substitution gives the stated pairwise-distance formula. Strict decrease follows because $\sinh^2(2s) > 0$ and arcosh is strictly increasing on $[1, \infty)$. $\square$

Abbreviations

HypSkeletonCLR	Hyperbolic variant of SkeletonCLR
C	Balanced clustering
H	Clustering with hierarchy learning
PL	Clustering with pseudo-label feedback
HPL	Clustering with hierarchy learning and pseudo-label feedback
3s	Three-stream model (joint, motion, and bone)

List of Figures

2.1. Skeleton sequence with spatial and temporal connections	6
4.1. Overview of the hierarchy-aware extension of HypSkeletonCLR. Hyperbolic representations receive balanced soft assignments to learned prototypes, whose relations are regularized by the hierarchy objective. See Section 4.3.	21
4.2. Single-view HypSkeletonCLR architecture used as the representation backbone in this thesis. Reproduced from [7].	22
4.3. Hierarchy-Aware Contrastive Learning overview. Sinkhorn uses balanced key-view targets and query predictions. The term L_{con} includes the minibatch pseudo-label loss and the distance-floor penalty.	23
6.1. t-SNE of dressing actions and prototypes	42
A.1. Dressing reference and prototype assignments	47
A.2. Hyperbolic t-SNE of dressing actions and prototypes	49

List of Tables

6.1. Cluster-count ablation on the cross-view split	36
6.2. Ablation of clustering, hierarchy loss, and pseudo-label feedback at $K = 40$ for joint-only HypSkeletonCLR. Accuracies are in percent; bold values identify the best result within each split.	37
6.3. Ablation of clustering, hierarchy loss, and pseudo-label feedback at $K = 40$ for 3s-HypSkeletonCLR. Accuracies are in percent; bold values identify the best result within each split.	38
6.4. Robustness comparison on unprocessed 300-frame sequences for the joint-only cross-view setting. HypSkeletonCLR-H uses $K = 5$ and is trained without pseudo-label positives. Accuracies are in percent.	39
6.5. Triplet agreement with the reference hierarchy	40
A.1. Reference-label permutation control	51

Bibliography

- [1] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. “NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1010–1019.
- [2] S. Yan, Y. Xiong, and D. Lin. “Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018, pp. 7444–7452. DOI: 10.1609/aaai.v32i1.12328.
- [3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A Simple Framework for Contrastive Learning of Visual Representations.” In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 1597–1607.
- [4] L. Li, M. Wang, B. Ni, H. Wang, J. Yang, and W. Zhang. “3D Human Action Representation Learning via Cross-View Consistency Pursuit.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 4741–4750.
- [5] M. Nickel and D. Kiela. “Poincare Embeddings for Learning Hierarchical Representations.” In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 6338–6347.
- [6] O.-E. Ganea, G. Becigneul, and T. Hofmann. “Hyperbolic Neural Networks.” In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018, pp. 5345–5355.
- [7] C. Almeida Jordan. “Skeleton-Based Human Action Recognition with Hyperbolic Contrastive Learning.” Master’s thesis. Technical University of Munich, 2025.
- [8] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. “Supervised Contrastive Learning.” In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 18661–18673.
- [9] J. Li, P. Zhou, C. Xiong, and S. C. H. Hoi. “Prototypical Contrastive Learning of Unsupervised Representations.” In: *International Conference on Learning Representations*. 2021.

- [10] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. “Momentum Contrast for Unsupervised Visual Representation Learning.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 9729–9738.
- [11] A. Ermolov, L. Mirvakhabova, V. Khrulkov, N. Sebe, and I. Oseledets. “Hyperbolic Vision Transformers: Combining Improvements in Metric Learning.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 7409–7419.
- [12] S. Ge, S. Mishra, S. Kornblith, C.-L. Li, and D. Jacobs. “Hyperbolic Contrastive Learning for Visual Representations Beyond Objects.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 6840–6849.
- [13] L. van der Maaten and G. Hinton. “Visualizing Data using t-SNE.” In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605.
- [14] T. Guo, H. Liu, Z. Chen, M. Liu, T. Wang, and R. Ding. “Contrastive Learning from Extremely Augmented Skeleton Sequences for Self-Supervised Action Recognition.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. 2022, pp. 762–770. DOI: 10.1609/aaai.v36i1.19957.
- [15] L. Lin, J. Zhang, and J. Liu. “Actionlet-Dependent Contrastive Learning for Unsupervised Skeleton-Based Action Recognition.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 2363–2372.
- [16] T. Gong, Q. Chu, B. Liu, and N. Yu. “Rethinking Masked Data Reconstruction Pretraining for Strong 3D Action Representation Learning.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 3. 2025, pp. 3149–3157. DOI: 10.1609/aaai.v39i3.32324.
- [17] S. Sun, Z. Cheng, Z. Zhang, J. Dong, Z. Li, and M. Wang. “Exploring Adaptive Masked Reconstruction for Self-Supervised Skeleton-Based Action Recognition.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2026, pp. 13974–13983.
- [18] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. “Deep Clustering for Unsupervised Learning of Visual Features.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149. arXiv: 1807.05520.

- [19] Y. Xu, F. Wei, X. Sun, C. Yang, Y. Shen, B. Dai, B. Zhou, and S. Lin. “Cross-Model Pseudo-Labeling for Semi-Supervised Action Recognition.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 2959–2968. DOI: 10.1109/CVPR52688.2022.00297. arXiv: 2112.09690.
- [20] I. Chami, A. Gu, V. Chatziafratis, and C. Re. “From Trees to Continuous Embeddings and Back: Hyperbolic Hierarchical Clustering.” In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 15065–15076. arXiv: 2010.00402.
- [21] T. Long and N. van Noord. “Cross-modal Scalable Hyperbolic Hierarchical Clustering.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 16655–16664.
- [22] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments.” In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.
- [23] K. Sohn. “Improved Deep Metric Learning with Multi-class N-pair Loss Objective.” In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016.
- [24] Y. Guo, M. Xu, J. Li, B. Ni, X. Zhu, Z. Sun, and Y. Xu. “HCSC: Hierarchical Contrastive Selective Coding.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 9706–9715. arXiv: 2202.00455.
- [25] J. M. Lee. *Introduction to Smooth Manifolds*. 2nd ed. Vol. 218. Graduate Texts in Mathematics. Springer, 2013. ISBN: 978-1-4419-9982-5. DOI: 10.1007/978-1-4419-9982-5.
- [26] J. M. Lee. *Introduction to Riemannian Manifolds*. 2nd ed. Vol. 176. Graduate Texts in Mathematics. Springer, 2018. ISBN: 978-3-319-91755-9. DOI: 10.1007/978-3-319-91755-9.