Comparison between Propagation Kernels and GNNs for Human Gesture Recognition

Nicolas Fleury-Rousseau (1955755) , Siyuan Jiang (2023661)

Abstract

Skeleton-based human gesture recognition has attracted significant attention in recent years due to its applications in human-computer interaction, surveillance, and health monitoring. In this work, we compare two different paradigms for graph-based gesture recognition: the Propagation Kernel (PK), a non-parametric method based on iterative feature diffusion, and a Graph Neural Network (GNN) architecture that learns node and graph embeddings end-to-end.

We evaluate both methods on the MSRC-12 dataset, where each gesture is represented as a spatio-temporal skeleton graph. Our experiments show that the Propagation Kernel achieves a validation accuracy of 96.56%, slightly outperforming the best GNN model which reaches 94.23%. While the PK method offers robustness and simplicity, the GNN approach provides greater flexibility and modeling capacity at the cost of increased computational demands.

We discuss the strengths and limitations of each approach and highlight future directions, including the use of attention mechanisms, adaptive propagation strategies, and hybrid models combining kernel and learning-based methods to further improve performance in gesture recognition tasks.

1 Introduction

Human gesture recognition is a critical task in various domains such as human-computer interaction, surveillance, and health monitoring. Recent advances in sensor technology and skeleton-based datasets have made it possible to represent human movements as temporal graphs, where nodes correspond to joints and edges to anatomical or functional connections.

Two main families of approaches have emerged to process graph data: Graph Neural Networks (GNNs) and Graph Kernels. GNNs leverage neural message-passing mechanisms to learn node and graph-level representations in an end-to-end fashion. Despite their strong performance, GNNs typically require large amounts of training data and are often computationally intensive. On the other hand, graph kernels offer a non-parametric alternative by comparing graphs based on their structural and label similarities, often with less need for hyperparameter tuning or large datasets.

In this project, we focus on the comparison between *Propagation Kernel* and a GNN-based approach on human gesture recognition.

2 Related Work

2.1 Graph Kernels for Structured Data

Graph kernels are a family of methods that define similarity functions between graphs based on their structural and label properties. Traditional kernels include the Weisfeiler-Lehman kernel [Shervashidze *et al.*, 2011], which leverages node label refinements through neighborhood aggregation, and the shortest-path kernel [Borgwardt and Kriegel, 2005].

The Propagation Kernel (PK), proposed by Neumann et al. [Neumann *et al.*, 2016], extends this idea by iteratively propagating node features across the graph and comparing feature distributions using histograms. PK is particularly effective for attributed graphs and is efficient due to its ability to reuse computation between propagation steps. Several recent works have applied PKs in chemical compound analysis, bioinformatics, and scene understanding, but their use in gesture recognition is still limited.

2.2 Graph Neural Networks for Human Action Recognition

Graph Neural Networks (GNNs) have become a dominant paradigm for learning from structured data. Kipf and Welling's Graph Convolutional Network (GCN) [Kipf and Welling, 2017] introduced a message-passing framework where node embeddings are updated via local neighborhood aggregation. Variants like Graph Attention Networks (GAT) [Veličković *et al.*, 2018] and GraphSAGE [Hamilton *et al.*, 2017] further improve flexibility and generalization.

In the context of human motion analysis, skeleton-based GNNs such as ST-GCN [Yan *et al.*, 2018] have achieved strong performance by modeling spatial and temporal dependencies across joints. These models treat joints as graph nodes and define edges based on anatomical connections or learned attention patterns.

While GNNs offer end-to-end learning and expressive modeling capacity, they typically require substantial labeled

data and careful tuning, and may be computationally intensive during training.

2.3 Datasets for Skeleton-based Gesture Recognition

Public datasets like MSRC-12 [Fothergill *et al.*, 2012], NTU RGB+D [Shahroudy *et al.*, 2016], and Kinetics-Skeleton [Yan *et al.*, 2018] provide skeleton data extracted from depth cameras or pose estimation systems. MSRC-12 is especially suited for lightweight experiments and kernel-based methods due to its manageable size and clearly segmented gestures.

These datasets facilitate benchmarking graph-based learning methods, both in terms of accuracy and runtime efficiency.

3 Dataset and Graph Representation

3.1 MSRC-12 Gesture Dataset

The MSRC-12 dataset is a public benchmark provided by Microsoft for evaluating gesture recognition models using skeleton data [Fothergill *et al.*, 2012]. It contains motion sequences captured from 30 individuals performing 12 different gestures in natural settings. Each skeleton frame consists of 20 joints, recorded at approximately 30 Hz using a depth camera with 3D positions.

Each sample in the data set is made up of two files: a `.csv` file containing the raw skeleton over time, and a `.tagstream` file annotating when specific gestures occur. The `XQPCTick` values provide high-resolution timestamps for aligning gesture labels with the corresponding frames in the skeleton sequence.

3.2 Graph Construction

To apply graph-based learning techniques, we represent each skeleton frame as an undirected graph. The nodes correspond to the 20 body joints, and the edges encode the anatomical connections between them (e.g. spine, arms, and legs). This fixed skeleton topology is shared between all frames and subjects.

3.3 Labels and Temporal Segmentation

Gesture labels are assigned based on the `.tagstream` annotations. Each gesture is also assigned a unique class label from a fixed tagset of 12 predefined actions (e.g., *Throw*, *Shoot*, *Kick*, *Crouch*).

A custom loader is used to parse the raw `.csv` and `.tagstream` files, align timestamps, filter valid frames, and convert them into graph objects compatible with kernel-based and neural graph models. Since only the end of each gesture is annotated, we consider the 30 frames preceding the annotation as the complete gesture, given that the movements are short in duration.

4 Methodology

For the experimentation, we will measure the accuracy with which each method can classify a gesture among the following:

- 0 : Start Music/Raise Volume
- 1 : Change weapon
- 2 : Move up the tempo of the song
- 3 : Kick
- 4 : Crouch or hide
- 5 : Navigate to next menu
- 6 : Put on night vision goggles
- 7 : Wind up the music
- 8 : Shoot a pistol
- 9 : Take a Bow to end music session
- 10 : Throw an object
- 11 : Protest the music

Accuracy will simply be computed as the number of correctly identified samples divided by the total number of samples.

4.1 Propagation Kernel

The Propagation Kernel (PK), introduced by Neumann et al. [Neumann *et al.*, 2016], is a fast, flexible graph kernel designed to capture both structural and node-level feature similarities between graphs.

The Propagation Kernel works by iteratively propagating node features across the graph and summarizing the feature distributions using histograms. Similarity between two graphs is computed by comparing their feature histograms across all propagation steps.

Mathematical Formulation Let $G = (V, E, X)$ be a graph with $|V| = n$ nodes, edges E, and a node feature matrix $X \in R^{n \times d}$. The propagation kernel operates as follows:

1. **Initialization:** At time $t = 0$, the node features are $X^{(0)} = X$.
2. **Propagation:** At each time step $t = 1, \ldots, T$, features are updated using a normalized adjacency matrix $\hat{A}$:

   $$X^{(t)} = \hat{A} X^{(t-1)}$$

   where $\hat{A} = D^{-1} A$ is the row-normalized adjacency matrix of G.
3. **Histogram Construction:** After each propagation step t, the distribution of node features $X^{(t)}$ is binned into a histogram $H_G^{(t)}$ using quantization.
4. **Kernel Computation:** The final kernel value $K(G, G')$ between two graphs G and G' is computed as:

   $$K(G, G') = \sum_{t=0}^{T} \langle H_G^{(t)}, H_{G'}^{(t)} \rangle$$

   where $\langle \cdot, \cdot \rangle$ denotes the dot product.

Implementation. We used the GraKeL library [Siglidis *et al.*, 2020] to implement the propagation kernel via the `PropagationAttr()` function, as our graphs are fully attributed. To capture temporal dynamics, node features were divided into n segments representing different stages of the gesture, and concatenated to form enriched state vectors used

Algorithm 1 The general propagation kernel computation

Require: Graph database $\{G^{(i)}\}_i$, number of iterations t_{MAX}, propagation scheme(s), base kernel $\langle \cdot, \cdot \rangle$
  $K \leftarrow 0$, initialize distributions $P_0^{(i)}$
  for $t = 0$ to t_{MAX} **do**
    for all graphs $G^{(i)}$ **do**
      for all nodes $u \in G^{(i)}$ **do**
        Quantize $p_{t,u}$ — Bin node information
        where $p_{t,u}$ is the row in $P_t^{(i)}$ corresponding to node u
      end for
      Compute $\Phi_i = \phi(G_t^{(i)})$ — Count bin strengths
    end for
    $K \leftarrow K + \langle \Phi, \Phi \rangle$ — Compute and add kernel contribution
    for all graphs $G^{(i)}$ **do**
      $P_{t+1}^{(i)} \leftarrow \text{propagate}(P_t^{(i)})$ — Propagate node information
    end for
  end for

in the propagation kernel. For each node (joint), the characteristic vector consists of 3D relative coordinates and 3D relative velocities, that is, (x, y, z, v_x, v_y, v_z), calculated with respect to a parent joint located closer to the center of the hip. Thus, each graph is defined as a tuple (V, E, X), where V is the set of 20 joints, E the fixed set of anatomical connections, and X the matrix of node features 20×6. The number of segments n , as well as the hyperparameters w (which balances label and structural information) and t_{max} (the number of propagation steps), were tuned to optimize performance.

4.2 Graph Neural Network

Graph Neural Networks (GNNs) offer a learnable alternative to graph kernels by encoding the structure and attributes of graphs into trainable embeddings. Among various architectures, we adopted the Graph Convolutional Network (GCN) [Kipf and Welling, 2017] due to its simplicity and effectiveness in processing fixed-topology skeleton graphs.

Each graph represents a 30-frame gesture snippet, where nodes correspond to individual joints across time (600 nodes per graph: 20 joints × 30 frames). Intra-frame anatomical edges and inter-frame temporal edges are defined to capture both spatial and temporal relationships.

Mathematical Formulation Let $G = (V, E, X)$ be a graph with $|V| = n$ nodes and node feature matrix $X \in R^{n \times d}$. The GCN layer updates node embeddings via:

$$H^{(l+1)} = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

where $H^{(0)} = X$, $\hat{A} = A + I$ is the adjacency matrix with self-loops, $\hat{D}$ is its degree matrix, $W^{(l)}$ are learnable weights, and σ is the ReLU activation function.

After L convolutional layers, we apply global mean pooling to obtain a graph-level embedding:

$$h_G = \frac{1}{n}\sum_{i=1}^{n} H_i^{(L)}$$

which is passed to a fully connected layer to produce output logits for classification.

Implementation. We implemented a configurable GCN using PyTorch Geometric. The node features include 3D coordinates (x, y, z), normalized time index t, and normalized joint ID j, resulting in 5-dimensional input vectors. We experimented with different values of depth (number of GCN layers) and width (hidden dimension), training each configuration for 5 epochs using the Adam optimizer and a learning rate of 0.01.

To ensure fairness, all models are evaluated using the same train/validation/test splits. The model with the best validation accuracy is selected and evaluated on the test set.

Advantages. GNNs allow end-to-end learning of hierarchical representations and can capture non-linear spatio-temporal dependencies. Unlike propagation kernels, they support gradient-based optimization and can be extended with advanced techniques like attention, residual connections, or temporal modeling.

Algorithm 2 Graph Neural Network training procedure

Require: Training dataset $\{G^{(i)}\}_i$ with graphs $G^{(i)} = (V^{(i)}, E^{(i)}, X^{(i)})$, number of epochs E, learning rate η, number of GNN layers L
  Initialize model parameters $\theta = \{W^{(1)}, \ldots, W^{(L)}\}$
  for epoch $= 1$ to E **do**
    for each mini-batch $\mathcal{B} \subset \{G^{(i)}\}$ **do**
      for each graph $G = (V, E, X) \in \mathcal{B}$ **do**
        $H^{(0)} \leftarrow X$
        for $l = 1$ to L **do**
          $H^{(l)} \leftarrow \text{ReLU}(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l-1)}W^{(l-1)})$
        end for
        $h_G \leftarrow \frac{1}{|V|}\sum_{v \in V} H_v^{(L)}$ — Graph pooling
        $\hat{y} \leftarrow \text{Softmax}(W_{\text{clf}} h_G)$
      end for
      Compute cross-entropy loss $\mathcal{L}$
      Update parameters: $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}$
    end for
  end for

5 Results

5.1 Propagation Kernel

Hyperparameter Tuning

To optimize the performance of the propagation kernel, we explored several combinations of hyperparameters:

- Number of temporal segments $n \in \{4, 8, 16\}$ used to build the concatenated node features.
- Maximum number of propagation steps $t_{\max} \in \{1, 2, 4, 8\}$.

- Number of neighboring nodes used during propagation $w \in \{1, 2, 4, 8\}$.

The figures 1, 2 and 3 illustrate the accuracy of each configuration on a validation set. Each result corresponds to a different setting of the $(n, w, t_{\max})$ triplet.

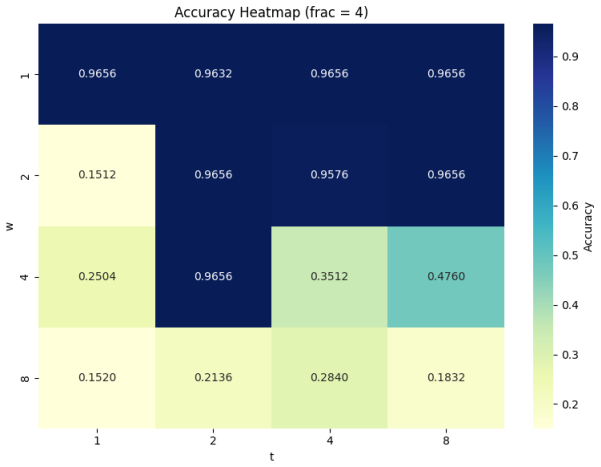

Figure 1: Heatmap for $n = 4$ illustrating the hyperparameter tuning results.

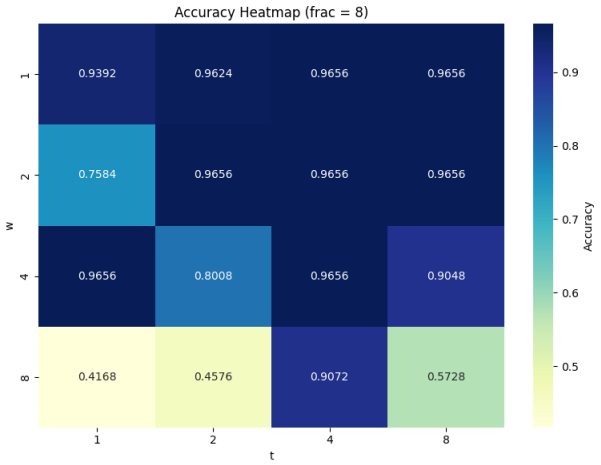

Figure 2: Heatmap for $n = 8$ illustrating the hyperparameter tuning results.

Best model result

The confusion matrix of the best model ($n = 16$, $w = 1$,$t_{max} = 1$) is represented by the figure 4.

5.2 Graph Neural Network

Hyperparameter Tuning

To optimize the performance of the Graph Neural Network (GNN) model, we explored several combinations of hyperparameters:

- Number of GNN layers (depth) $d \in \{1, 3, 5\}$.

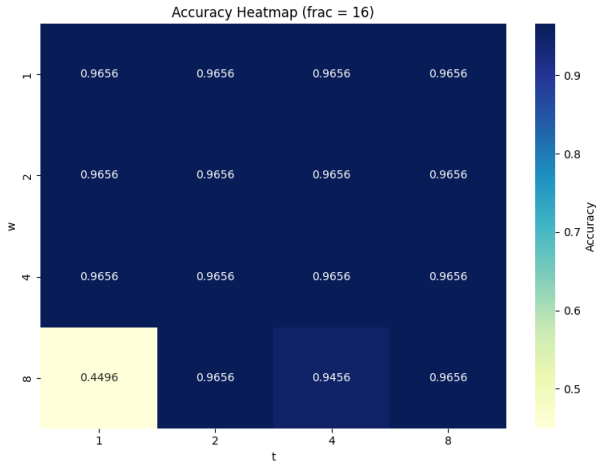

Figure 3: Heatmap for $n = 16$ illustrating the hyperparameter tuning results.

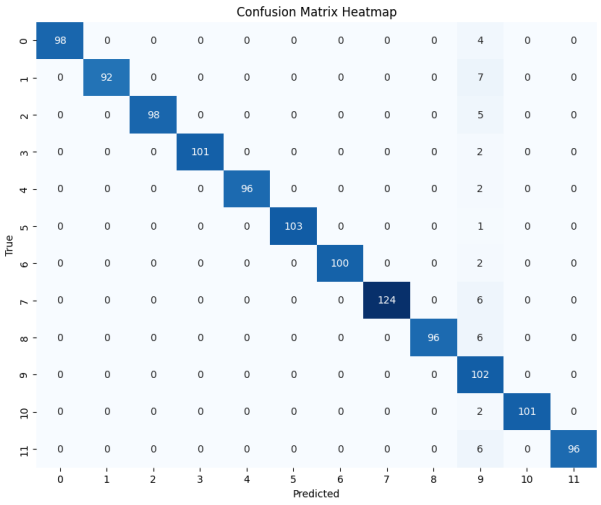

Figure 4: Confusion matrix of the best model of propagation kernel ($n = 16$, $w = 1$,$t_{max} = 1$)

- Number of hidden units per layer (width) $w \in \{25, 100, 300, 500\}$.

The heatmap in Figure 6 illustrates the validation accuracy achieved for each (d, w) configuration. The validation accuracy generally increases with both greater depth and greater width, but not monotonically. In particular, depth 5 with width 300 achieves the highest performance, suggesting that deeper and wider networks capture richer spatio-temporal patterns but may also introduce some redundancy if excessively large.

Best Model Result

The best GNN configuration is obtained with depth $d = 5$ and width $w = 300$, reaching a validation accuracy of 94.23%. This result confirms the importance of a sufficient receptive field depth to capture temporal dependencies in the skeleton sequence while maintaining a large enough hidden dimension for expressive feature learning.

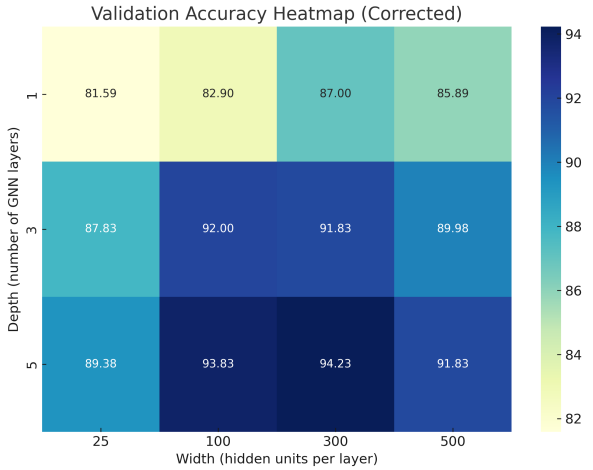

Figure 5: Validation accuracy heatmap for GNN models with different depth and width configurations.

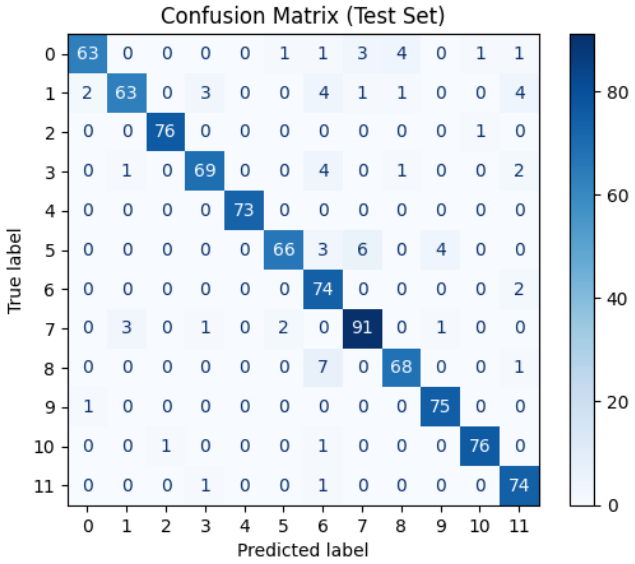

Figure 6: Confusion matrix of the best model of GNN.

The best GNN model significantly outperforms shallower and narrower architectures, highlighting the need for careful tuning of depth and width in GNN-based gesture recognition tasks.

We note that further performance gains could potentially be achieved by integrating residual connections, dropout regularization, or by exploring other architectures such as Graph Attention Networks (GAT) or Temporal GNN variants.

6 Discussion

6.1 Propagation Kernel

To optimize the hyperparameters, Figures 1, 2, and 3 reveal that the temporal fragmentation parameter n plays a crucial role in building a robust model that is less sensitive to variations in w and $t_{\max}$. A higher value of n generally leads to more stable and reliable performance, suggesting that incorporating a richer temporal resolution helps the kernel better capture motion dynamics.

As for the other hyperparameters, there is no clearly defined global trend, except that using a large window size w consistently degrades performance. This may be due to an over-smoothing effect, where relevant motion details are lost when too many neighboring features are aggregated. Some results also suggest that increasing $t_{\max}$ alongside w can counteract the degradation caused by large w values, possibly by compensating for the loss of local detail through more extensive temporal propagation.

Since the model appears to be stable with $n = 16$ and reaches the maximal accuracy of 96.56% across almost all tested combinations of w and $t_{\max}$, it seems logical to choose the most stable model with minimal computational cost—namely, $n = 16$, $w = 1$, and $t_{\max} = 1$ as the best configuration. This suggests that a fine temporal segmentation combined with minimal propagation is sufficient to capture the essential dynamics of human gestures in this dataset.

As for the confusion matrix, the results are particularly impressive, with almost perfect classification across all gesture classes. The only notable source of confusion arises with movement 9, "Take a Bow to End Music Session." This gesture likely exhibits significant intra-class variation and shares similarities with several other gestures, which may explain the misclassifications. Interestingly, removing this gesture from the dataset results in a perfect accuracy of 100%, highlighting its impact on overall performance. It would be interesting to combine this method with a K-means clustering step in order to potentially achieve better results. Since the propagation kernel seems to struggle in discriminating a specific gesture, clustering the feature representations beforehand could help group similar gestures and reduce ambiguity.

6.2 Graph Neural Network

The hyperparameter tuning results in Figure 6 show a clear trend where increasing both the depth and width of the GNN improves validation accuracy up to a point. In particular, moving from 1 to 3 layers yields a noticeable gain across all widths, suggesting that a deeper aggregation of spatial and temporal information significantly benefits gesture classification. However, further increasing depth to 5 layers does not always guarantee improvements unless coupled with sufficient width (e.g., width 300), indicating a possible trade-off between model expressiveness and overfitting risks.

The best-performing configuration (depth 5, width 300) achieved a validation accuracy of 94.23%, comparable to or even slightly better than the best result obtained using the Propagation Kernel. This demonstrates that a well-optimized GNN can effectively learn spatio-temporal patterns end-to-end without relying on handcrafted propagation schemes.

Nevertheless, the heatmap reveals some saturation effects: at very large widths (e.g., width 500), performance slightly drops for deeper models. This phenomenon could be attributed to overparameterization, making the model prone to memorizing the training data rather than generalizing to unseen gestures.

Overall, GNNs prove to be a powerful tool for skeleton-based gesture recognition, provided that architectural choices are carefully tuned based on the available data and task complexity.

6.3 Comparison

The comparative analysis between the Propagation Kernel (PK) and the Graph Neural Network (GNN) approaches highlights fundamental trade-offs between non-parametric and learnable models for skeleton-based gesture recognition.

The Propagation Kernel achieves slightly higher validation accuracy (96.56%) compared to the best GNN configuration (94.23%), demonstrating its effectiveness in capturing structural and feature-based similarities through simple propagation and histogram aggregation mechanisms. Its stability across a wide range of hyperparameters, particularly when using fine temporal segmentation ($n = 16$), makes it a robust choice when computational simplicity and interpretability are desired.

On the other hand, the GNN approach, although slightly less accurate on this dataset, offers superior flexibility. By learning node embeddings end-to-end through multiple message-passing layers, GNNs can potentially model more complex and subtle spatio-temporal patterns that may not be easily captured by fixed propagation schemes. Furthermore, GNNs are naturally extensible: one can integrate attention mechanisms, dynamic edge learning, or deeper hierarchical structures to further improve performance.

Another critical difference lies in scalability and adaptability. Propagation Kernels require explicit feature propagation and histogram binning at each inference step, which may become costly for very large graphs. In contrast, once trained, a GNN model can generalize to new graphs with a simple forward pass, making it more suitable for real-time applications or continuous learning scenarios.

Overall, the Propagation Kernel stands out for its simplicity, robustness, and strong performance in small to medium-sized datasets with fixed graph structures. GNNs, meanwhile, open up richer possibilities for modeling, but require careful tuning and are more computationally intensive during training. The choice between these two paradigms ultimately depends on the specific needs of the application: efficiency and simplicity versus flexibility and long-term scalability.

7 Conclusion

In this project, we compared two fundamentally different approaches for human gesture recognition based on skeleton graph data: the Propagation Kernel (PK) and a Graph Neural Network (GNN) architecture.

The Propagation Kernel demonstrated excellent performance, achieving a validation accuracy of 96.56%. Its strength lies in its simplicity, robustness across hyperparameter variations, and low computational cost during inference. However, its main limitation is its rigidity: once the propagation and histogram construction schemes are defined, the model cannot easily adapt or learn new relational patterns beyond the handcrafted propagation mechanism.

The Graph Neural Network, although achieving a slightly lower validation accuracy of 94.23%, offers greater modeling flexibility and scalability. Through end-to-end learning, GNNs can automatically discover complex spatio-temporal relationships without manual feature design. Nonetheless, GNNs also come with challenges: they require careful hyperparameter tuning, larger computational resources during training, and are more prone to overfitting, particularly on small datasets.

Looking forward, several promising directions could enhance both approaches. For Propagation Kernels, integrating adaptive propagation mechanisms or learned feature binning strategies could improve their capacity to capture fine-grained motion variations. For GNNs, exploring more sophisticated architectures such as Graph Attention Networks (GATs), spatio-temporal GNNs, or transformer-based models on graphs could further boost performance, particularly on more complex or larger-scale datasets.

Moreover, combining the strengths of both paradigms—using propagation-based preprocessing to enrich GNN inputs, or designing hybrid models—represents an exciting avenue for future research. Expanding experiments to larger datasets and evaluating robustness to noise and occlusions would also provide valuable insights into the practical deployment of graph-based gesture recognition systems.

References

[Borgwardt and Kriegel, 2005] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM)*, pages 74–81, 2005.

[Fothergill *et al.*, 2012] Simon Fothergill, Helena Mentis, Pushmeet Kohli, and Sebastian Nowozin. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1737–1746, 2012.

[Hamilton *et al.*, 2017] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[Neumann *et al.*, 2016] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, 102:209–245, 2016.

[Shahroudy *et al.*, 2016] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang 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)*, pages 1010–1019, 2016.

[Shervashidze *et al.*, 2011] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and

Karsten M Borgwardt. Weisfeiler-lehman graph kernels. In *Journal of Machine Learning Research*, volume 12, pages 2539–2561, 2011.

[Siglidis *et al.*, 2020] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[Yan *et al.*, 2018] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.