university of groningen

faculty of science and engineering

Pooling Operators in Graph Neural Networks for Human Activity Recognition

George Bondar

university of groningen / faculty of science and engineering

University of Groningen

Pooling Operators in Graph Neural Networks for Human Activity Recognition

Bachelor's Thesis

To fulfill the requirements for the degree of
Bachelor of Science in Computing Science
at University of Groningen under the supervision of
Prof. dr. Alexander Lazovik (Computer Science, Distributed Systems)
and
M.Sc. Andrés Tello (Computer Science, Distributed Systems)

George Bondar (s4060989)

November 15, 2023

Contents

Abstract

The field of Human Activity Recognition (HAR) has wide-ranging applications in areas such as healthcare, industry, and security. Graph Neural Networks (GNNs) have shown promising results in HAR due to their robustness to missing data and transfer learning capabilities. The accuracy of graph classification models depends to a large extent on the pooling operation. As a consequence, the scope of the research will fall back upon empirically analysing different pooling strategies based on a carefully selected model. The research questions are outlined, and the methodology is presented. The paper also includes background information on both GNNs for HAR and pooling operators. Finally, the paper presents limitations of graph pooling methods and proposes new metrics beyond empirical experiments for evaluating the performance gains of graph pooling models.

1 Introduction and Motivation

1.1 Human Activity Recognition Importance

Patients with chronic conditions or elderly patients, industrial workers, athletes, and even burglars are a few subject examples for which careful monitoring can be achieved. Perhaps, healthcare providers need to detect changes in the activity of any of their patients, or companies might want to improve the safety of their workers by preventing accidents and injuries through careful monitoring of the workers in the industrial setting. Athletes might want to monitor their progress, prevent injuries or easily set measurable goals, whereas someone might want to identify unusual and suspicious behaviour, such as break-ins, for security purposes. These are some of the use cases where *Human Activity Recognition (HAR)* plays a major role. It offers valuable insights into human behaviour providing efficient solutions to the scenarios presented above.

1.2 Graph Neural Networks Importance

The data collected from a multitude of sensors for HAR applications often presents representations of complex relationships and frequently includes incomplete or noisy information. In such contexts, *Graph Neural Networks (GNNs)* prove very useful. On one hand, it provides the necessary tools to capture complex relationships, while on the other hand, it is robust to missing data [1]. Furthermore, GNNs can be trained on one HAR dataset and then applied to a different but related dataset, obtaining good performative transfer learning capabilities which, in general, consist of expensive and time-consuming operations.

1.3 Scope of Research

In the field of GNNs, *graph classification* is a pivotal task. Its objective is to predict class labels of a set of graphs rather than individual nodes or edges [2]. This task is highly relevant to HAR, where the goal often involves categorising entire activity sequences. A crucial area for advancement centres around *pooling operators*, which are essential for successful graph classifications [3]. Pooling operators serve as a fundamental component responsible for gathering information from neighbouring nodes in a graph and obtaining a comprehensive graph-level representation [4]. This process significantly influences the performance and efficiency of GNNs [5].

I propose HAR using GNNs to be captured as a graph classification problem. After thorough analysis of existing works on the topic similar to and including the following proposed models: [6], [7], [8], and [9], I chose as a baseline for the experimental procedure the model presented in [7]. The reason behind is that it is the sole proposed model that approaches the challenge of recognizing human activities as a form of graph classification. Nonetheless, there are several aspects which make the work done in [7] of high value, one of which being the capability of innovatively combining deep transfer learning and GNNs to enhance Sensor-Based HAR. This approach effectively adapts pre-trained GNN models to new datasets (i.e. it leverages the knowledge gained from one dataset to improve the model's performance on another, potentially related dataset), conserving resources and expanding the model's versatility.

1.4 Research Questions

The primary objective of the research project is to conduct an in-depth analysis of diverse pooling operators implemented within the chosen model. Accordingly, I explored various pooling operators for graph classification and presented the outcomes through comprehensive documentation and meticulously designed experimental procedures.

This leads to the following research questions:

Q1. What are the limitations of existing graph pooling methods in terms of capturing and representing complex graph structures, and how does it impact their performance gains?

Q2. What are the criteria or metrics beyond empirical experiments that can be used to evaluate the performance gains of graph pooling models?

By delving into the research questions outlined above, I envision making noteworthy contributions to the field of GNNs focusing on *Graph Pooling Operations* applied to HAR. These contributions are not limited to a specific area of expertise, as the implications of my work extend across various domains. I firmly believe that my research will establish a solid foundation, serving as a springboard for researchers to explore diverse facets beyond the scope of HAR. Framing HAR as a Graph Classification problem allows us to explore Pooling Operators and translate our results to any domain where the solution relies on vector representations of entire graphs.

1.5 Thesis Outline

The structure of the thesis document is the following: section *2* presents background information on all necessary concepts that build the foundation for understanding the scope of the research. Section *3* provides an analysis of the 4 proposed models for HAR and showcases the relevant works done regarding pooling operators. The same section incorporates key aspects that describe framing a task at hand as a graph classification task. Section *4* exposes the methodology behind the project whereas the setup of the experimental procedure is outlined in section *5*. The results obtained are showcased in section *6* and the analysis of the pooling operators is provided in section *7*. In section *8*, the research project is concluded and some of the potential future work is presented.

2 Background

To be able to understand the beneficial capabilities of GNNs for HAR and the importance of the pooling operation in a graph classification task, we first need some introductory knowledge that conveys all the necessary background information.

2.1 Graph Theory

First and foremost we need to understand what a graph is. According to [10], a graph is a representation of the *relations* between a collection of *entities*. These relations are described as *edges*, whereas the entities themselves are considered *nodes*. Both edges and nodes can convey information called *features*. The graph itself can also hold information in a so-called *global attribute* [10]. The features can be later utilised for the training procedure of the GNN model, after which they are transformed from the original features into *embeddings* in a higher latent space Z. We can further specialise graphs by including directionality of edges. This means that in a so-called *directed* graph every edge has a source node and a destination node. In *undirected* graphs there is no notion of the source and destination nodes, as the information flows both ways [10].

2.2 Neural Networks

Neural network is a type of machine learning process, which mimics the human brain through the way that computers are thought to process data [11]. Deep learning is considered to be a subset of machine learning, which is essentially a neural network that has at least 3 layers [12]. GNNs are deep learning based methods which operate on graph domain [1]. Taking into consideration all the attributes of a graph (i.e., nodes, edges, and the overall structure of the graph), [13] defines a GNN as an "optimizable transformation" across all the attributes of the graph but with a critical aspect in mind, which is that the transformation should preserve all the symmetries of the given graph (i.e., the transformation should be permutation-invariant of the graph structure).

2.3 GNNs for HAR

HAR has started to receive a great amount of attention since the advancements made regarding deep learning in the past few years. GNNs have emerged as promising tools for improving the performance of HAR using deep learning techniques. The reason is the fast capabilities of GNNs for determining segments of missing labels in the training data based on train settings that handle different tasks when working with the structure of the graph (i.e., node-level tasks, edge-level tasks, and graph-level tasks). It is worth mentioning that GNNs can only handle small parts of missing data and in the case of a large amount, the performance significantly decreases, resulting in a low prediction probability computed by the corresponding model [1].

2.3.1 Graph Train Settings

There are 3 different train settings defined by [1] for a graph: *supervised setting* in which the data provided is fully labelled, *semi-supervised setting* in which a great amount of data is unlabeled and only a small proportion consists of labelled data and last but not least, *unsupervised setting*, in which the data is fully unlabeled.

2.3.2 Graph Tasks

When it comes to the 3 different tasks mentioned above (i.e., node, edge, and graph-level), [1] defines them as follows: *node-level tasks* are primarily based on node classification (i.e., trying to categorise nodes into several classes), node regression (i.e., predicts a continuous value for each node) and node clustering (i.e., aims to partition the nodes into several disjoint groups, where similar nodes should be in the same group). *Edge-level tasks* mainly consist of predicting links between 2 nodes or classifying the type of an edge. Finally, *graph-level tasks* refer to graph classification, regression, and matching, all of which train a model on graph representations.

2.4 Graph Convolutional Networks (GCNs)

Before we dive into understanding pooling operators, we need some knowledge that will further allow us to properly assess the choice of the GNN model used and its architecture. According to [14], *convolution* in simple terms is the repeated application of a filter to an input. *Graph Convolutional Networks (GCNs)* are a specific architecture or model type within the GNN family. They contain a so-called *convolutional layer*.

2.5 Pooling Operators

Finally, let's see what *pooling operators* are and how they work. A pooling operator is a mechanism used to downsample or reduce the number of nodes in a graph while preserving the semantic information of the graph [4]. Pooling operators play a critical role in GNNs by aggregating information from a group of nodes and producing a smaller, coarser-grained representation of the original graph [15].

2.5.1 Graph Pooling

The predominant category of pooling operators within GNNs, and the focus of our interest, is the *global* or *graph pooling*. Although these terms are interchangeably employed across various state-of-the-art literature, they essentially convey the same concept. For the sake of simplicity and improved readability, I will adopt the term *graph pooling* throughout the rest of the paper. Graph pooling refers to methods that reduce the graph to a single node, discarding all topological information [16]. They play a major role in determining the representation of a graph as their sole purpose is to map the set of nodes of a graph into a compact representation that resembles a meaningful structure of the entire graph [17].

2.5.2 Characteristics

When we refer to graph pooling there are certain key concepts that allow us to differentiate between various operators. I am going to discuss a couple of them.

2.5.2.1 Learnability

The first one refers to the *learnability* of the graph pooling operators. Here we can distinguish between 2 types: *fixed pooling* and *learnable* or *adaptive pooling*. On one hand, fixed graph pooling refers to methods which have the number of nodes of the pooled graph as a constant and completely independent from the input graph size. On the other hand, if the number of nodes of the pooled graph is actually an adaptive function of the input graph then those methods are considered to be part of learnable or adaptive graph pooling. For both types the number of nodes of the pooled graph is represented as a parameter in the pooling operation. In the context of choosing a pooling operator with regard to its learnability characteristics, it is advisable to prioritise learnable or adaptive techniques if the relative graph size plays a pivotal role in addressing a specific task [16].

2.5.2.2 Spatial Scope

The second key concept we need to address is regarding the *spatial scope* of the graph pooling operations. Within this context, we again distinguish between 2 categories: *flat pooling* and *hierarchical pooling*. The former encompasses approaches that directly yield a graph-level representation in a single step, typically involving operations such as averaging or summing all node embeddings to form the graph representation. In contrast, the latter encompasses methods that gradually reduce the size of the graph through two primary mechanisms: *Node Clustering Pooling* and *Node Drop Pooling*. Node clustering pooling involves grouping nodes into clusters to form a coarsened graph, which can be computationally intensive in terms of time and space. On the other hand, node drop pooling selects a subset of nodes from the original graph to construct a coarsened version, which proves to be more efficient, especially for larger-scale graphs, albeit with some inherent information loss. A significant point of distinction between these two categories lies in the fact that node clustering pooling generates new nodes for the coarsened graph, whereas node drop pooling preserves nodes from the original graph [4].

2.6 Hyperparameter Optimisation

Hyperparameter optimisation is an integral aspect of machine learning and deep learning model development, contributing significantly to model performance and generalisation [18]. In this section, we delve into the concept of hyperparameters, their critical role in shaping models, and methodologies for optimising them effectively.

Hyperparameters are essential configuration settings that govern a machine learning model's behaviour and learning process. Unlike model parameters, which are learned from data during training, hyperparameters are predefined by researchers or practitioners before the training process begins. These settings influence various aspects of the model, including its capacity, learning rate, regularisation strength, and architecture [18].

Efficient hyperparameter tuning can significantly impact a model's performance, leading to improved accuracy, reduced overfitting, and enhanced generalisation. Neglecting hyperparameter optimisation may result in suboptimal models that fail to capture underlying patterns in the data [18].

2.6.1 Methods

Several techniques are available for hyperparameter optimisation, ranging from manual tuning to automated approaches:

Grid Search [19]: Grid search involves manually specifying a range of values for each hyperparameter and evaluating the model's performance for all possible combinations. While straightforward, it can be computationally expensive and impractical for high-dimensional search spaces.

Random Search [19]: Random search randomly samples hyperparameters from predefined distributions. This method is more efficient than grid search and often leads to better results.

Bayesian Optimisation [20]: Bayesian optimisation leverages probabilistic models to select promising hyperparameters iteratively. It efficiently explores the hyperparameter space and is particularly useful for expensive-to-evaluate models.

Genetic Algorithms [21]: Genetic algorithms are inspired by the process of natural selection. They evolve a population of hyperparameter sets over multiple generations to discover optimal configurations.

Tools like *sklearn*, *Hyperopt*, and *Optuna* offer automated hyperparameter tuning using various optimisation algorithms, simplifying the process.

3 State of the Art

3.1 Challenges

The field of HAR with GNNs presents unique challenges that demand thoughtful consideration. Two prominent challenges are the *acquisition of high-quality data* [7, 9] and the *intrinsic temporal characteristics* [6, 8] of human activities. Addressing these challenges is pivotal to advancing the field and developing effective HAR solutions.

The first challenge revolves around the collection of high-quality data. Achieving a balanced dataset often necessitates real-time interventions to ensure precise data interpretation. The second challenge pertains to the temporal nature of human activities. Human activities follow specific chronological sequences, such as the tendency to shower after a workout, not before. Addressing these chronological properties is imperative for accurate activity recognition.

3.2 Proposed Models

Numerous models have emerged to harness the capabilities of GNNs for HAR, providing solutions to the aforementioned challenges.

The model proposed in [6] leverages the temporal properties of human activities to predict unlabeled data. This approach falls into the realm of node classification, since human activities were represented as nodes within a fully connected graph. By comparing this proposed model against similar standard models they were able to showcase its proficient performance. The basis behind its good performance is entailed from the capabilities of the model to predict unlabeled activities based on the known ones.

A deep transfer learning model presented in [7], nicely handles modalities variation (i.e., different types of sensors with data missing). The structure of the model investigated follows the one of a Residual Graph Convolutional Neural Network(ResGCNN) which is further modified to be suitable for HAR. The experiments conducted showcase the performance of the model.

The model proposed in [8] is conducted under a semi-supervised setting based on Graph Convolutional Networks(GCNs). This model introduces an innovative approach to sensor data modelling, involving the creation of fully connected subgraphs through sliding windows. The usage of sliding windows allowed the authors to target the second challenge of HAR presented above, namely, the intrinsic temporal characteristics of human activities. Furthermore, an integral component of the proposed model incorporated a spectral graph convolution method, enabling the model to make accurate predictions regarding the relationships between subgraphs. Notably, subgraph representations of the data were the leading factor for showcasing a superior performance of the proposed model.

Finally, the work done in [9] investigates thoroughly the information exchange loss between wireless communication channels and proposes a model which is carefully adapted to combat this downside. The adaptation consists of grouping all channels into the same layer offering a better interaction and ultimately a better representation of the network. The cross-channel communication is implemented by a message passing GNN over a fully connected graph where each channel represents a node.

3.3 Graph Classification

There are various paths to explore when it comes to using GNNs for HAR. Ultimately, the direction that I am going to follow boils down to a *graph classification task*. Morris, in [22], defines GNNs for graph classification as: "GNNs that learn a graph level output". The main difference between node-level representations which is a common objective among GNNs and graph-level representation which is incorporated in the graph classification problem is the pooling layer. A well-established overview of pooling layers is also portrayed in [22]. Additionally, [22] captures limitations of graph classifications and possible solutions to overcome them.

The work done in [23] has provided more insights into graph classification, conducting over 47.000 experiments in which an analysis of 5 popular models across 9 benchmarks was offered. The authors also provided strongly reassuring evidence which suggests that structural information across some datasets has not been properly utilised until now. They managed to build up the evidence by comparing GNNs to structure-agnostic baselines, which were also provided by them.

3.4 Pooling Operators

In the context of this research, I have carefully selected based on their relevance, quality, and applicability to the research questions, [4], [24], [25], and [26] as primary references for the investigation of pooling operators.

Beginning with [4], the work by Liu et al., encapsulates a robust theoretical framework that underpins the description and understanding of pooling operators. It lays the groundwork for comprehending the mathematical and conceptual foundations that govern these operators, fostering a deeper appreciation of their functionality and implications. Additionally, [4] sheds light on the prevailing challenges and promising opportunities associated with diverse pooling operators, providing a holistic view of the current landscape and potential directions for further research.

Moving to [24], this reference introduces an innovative paradigm shift in GNNs by focusing on the adaptability of readout functions to effectively learn representations of graph structured data. These adaptive readout functions dynamically adjust how the network combines information from different parts of a graph. Through iterative refinement, they selectively emphasise or de-emphasize features, optimising GNNs to effectively learn and represent diverse graph-structured data for various tasks. This adaptability is particularly noteworthy as it enables GNNs to flexibly tailor their processing to diverse contexts, making them highly versatile for a wide range of applications. The research drive emerges from the complexities faced during model training using standard neighbourhood aggregation techniques, which inherently operate within permutation-invariant hypothesis spaces. The examined adaptive readout functions hold potential in the context of neural networks that do not inherently lead to permutation-invariant spaces. This is exemplified through empirical examinations of potential extensions and adaptations, particularly in situations where graphs may be presented in a canonical manner. Finally, within [24], various classes of adaptive and differentiable readout functions are introduced, and their effectiveness is showcased through the experimental procedure conducted.

Reference [25] makes a significant contribution to the discourse by introducing an end-to-end deep learning architecture explicitly tailored for graph classification tasks. It encapsulates the essence of utilising GNNs in real-world scenarios, emphasising the practical applications of pooling operators.

The proposed pooling operators are designed to handle graphs with different levels of complexity. They extract important features from the graph, ensuring a clear understanding of the graph's information for tasks like classification. Notably, the proposed architecture exhibits a remarkable feature by accepting graphs regardless of their structural complexities. The pivotal focus of [25] revolves around addressing two primary challenges: firstly, the challenge of extracting valuable features that characterise the rich information embedded within a graph for classification purposes, and secondly, the challenge of sequentially interpreting a graph in a coherent and consistent manner. Additionally, [25] serves as a bridge between theoretical concepts and real-world implementations, highlighting the practical considerations and implications of pooling operators in GNNs.

Lastly, [26] introduces the concept of differentiable pooling mechanisms. They enable the network to capture information at different scales within graphs, especially in situations where representing the hierarchy of information is crucial for the task at hand. These mechanisms hold immense promise for enhancing the adaptability and performance of GNNs. Furthermore, [26] delves deep into the intricate details of differentiable pooling, offering insights into differentiation mechanisms and their pivotal role in capturing multi-scale information within graphs. This reference is pivotal for researchers seeking to leverage hierarchical pooling operators effectively.

3.4.1 Standard vs. Advanced

The examination of pooling operators conducted in this research encompasses various types, including *standard operators* (i.e., mean, max, and sum) and *advanced operators* (i.e., top-k pooling and graph multiset transformer (GMT)).

To provide a concise explanation of these standard operators and their functionality, we can refer to [27]. *Mean pooling* involves calculating the average of the feature vectors within a specified neighbourhood or region of the graph. It is a straightforward method that computes the mean value of the features, effectively summarising the information within the neighbourhood. *Max pooling*, on the other hand, selects the maximum value from the feature vectors within the defined region. This operator focuses on capturing the most salient or dominant features present in the neighbourhood. *Sum pooling* aggregates the feature vectors by simply summing their values within the specified region. It provides a comprehensive view of the cumulative information contained within the neighbourhood. These standard operators are fundamental in the context of pooling in GNNs. They play a crucial role in aggregating information from neighbouring nodes or subgraphs, aiding in the creation of informative graph-level representations.

Within the realm of the advanced pooling operators, we can first refer to *hierarchical pooling* operators, which are renowned for their complexity and effectiveness. These operators encompass diverse techniques, such as *top-k pooling*, which is designed to select the most informative elements within a set. For a deeper understanding of these hierarchical pooling methods, we turn to [28] and [29]. These sources delve into the mathematical foundations, algorithms, and practical applications of hierarchical pooling, providing valuable insights into their use cases, performance evaluations, and comparisons with other pooling techniques. Together, they offer a comprehensive perspective on the landscape of hierarchical pooling techniques. Furthermore, our investigation extends to *flat pooling* operators. Among these techniques, we delve into *graph multiset transformer pooling (GMT)*, which leverages the power of set transformers to capture multi-scale information within graphs. A valuable resource for understanding flat pooling, including graph multiset pooling, can be found in [30].

This source provides in-depth insights into the principles, methodologies, and practical applications of flat pooling. It likely discusses the theoretical underpinnings, implementation strategies, and experimental outcomes, highlighting its advantages in various contexts. By delving into these diverse pooling operators, including top-k pooling and flat pooling techniques like graph multiset pooling, we equip ourselves with a comprehensive understanding of their intricacies, benefits, and practical implications. This allows us to properly assess the analysis of the pooling operators for determining their strengths and limitations, and to understand the new metrics proposed for evaluating their performative capabilities.

4 Methodology

The methodology behind the research project consists of evaluating existing pooling operators while simultaneously exploring ways to introduce my original ideas to enhance their performance. Although this might not be the case, an analysis of existing pooling operators is guaranteed to be conducted.

4.1 Preliminaries

To set the stage for our exploration, let's establish some preliminary definitions. The ones presented in subsections *4.1.1.* and *4.1.2* are adjusted from the work of Baek et al., [17].

4.1.1 Graph Neural Networks (GNNs)

Consider a graph G described by an adjacency matrix $A \in \{0,1\}^{n\times n}$ and a node set V with $|V| = n$ nodes, coupled with c-dimensional node features $X \in \mathbf{R}^{n\times c}$. GNNs acquire feature representations for diverse nodes using neighborhood aggregation techniques, which are encapsulated in the subsequent message-passing function:

$$H_u^{(l+1)} = \text{UPDATE}^{(l)}(H_u^{(l)}, \text{AGGREGATE}^{(l)}(\{H_v^{(l)}, \forall v \in \mathcal{N}(u)\}))$$

Here, $H^{(l+1)}$ signifies the node features computed after l-steps of the GNN, $\mathcal{N}(u)$ denotes the set of neighboring nodes of u, and $H_u^{(1)}$ is initialized as the input node features X_u. UPDATE and AGGREGATE are differentiable functions chosen arbitrary.

This is a simplified version of the formula above:

$$H^{(l+1)} = \text{GNN}^{(l)}(H^{(l)}, A^{(l)})$$

4.1.2 Graph Pooling

While the message-passing functions excel in producing node representations, a complementary POOLING operator becomes imperative to derive an entire graph representation $h_G \in \mathbf{R}^d$. This operator is defined as:

$$h_G = \text{POOL}(\{H_v \,|\, v \in V\})$$

For the POOLING operator, conventional choices involve employing the average or sum over all node features $H_v, \forall v \in V$ within the given graph.

4.2 Analysis Strategy

A raw interpretation of the pooling operations' analysis consists of the following steps:

1. I investigated more thoroughly the challenges and opportunities of graph pooling operators.
2. For determining the optimal pooling operator there are a couple of potential strategies that I can choose from:
   (a) On one hand, since this is a graph classification problem, I can conduct an experimental procedure in which I compare the accuracy of the different pooling operators and choose the one that produces the highest.

(b) On the other hand, since I can obtain access to different HAR datasets, another approach for determining the optimal pooling operator would be to look for an operator which does not necessarily provide the highest accuracy but one that is offering a good performance over multiple datasets entailing consistent results.

Considering the methodology employed to address the aforementioned research questions, a significant aspect comes to light, namely, what is the most suitable pooling operator for the model that was decided upon. This exploration is motivated by answering the first research question which will prompt an understanding of why the worst performing pooling operators failed, based on the limitations found. On the contrary, by answering the second research question, the best pooling operator can be showcased through the metrics proposed.

4.3 Development

4.3.1 Tools

Python is the primary programming language used for developing the project and carrying out experimental procedures. The main framework used is *PyTorch*, which is a deep learning framework built on top of Python. *PyTorch Geometric* is an extension library for PyTorch that offers a wide variety of tools for working with graph-structured data (e.g. common data loaders and transforms that can be used to load and process the data, benchmark datasets that can be used for testing purposes of models, different graph neural network operators). Ultimately, this is the main choice for the library used.

GitHub will be the main version control and collaboration tool used for the development of the project. A private repository will be created which will host all the necessary source code. Once the research is done, the complete version of the code will be made available on a platform that needs to be decided upon.

4.3.2 Limitations

All the needed software is accessible, however, there is a need for computational power. Without a good *Graphics Processing Unit (GPU)* running and training a model can become very costly. This would be the case when you have to wait for a model to finish its tasks approximately 24 hours only then to realise that the model does not have a good prediction probability. Then, you would have to make the necessary adjustments and wait another round of 24 hours. Part of the solutions would be hardware with a good GPU or perhaps a paid subscription for hosting servers that would provide access to a suitable GPU. This can be for example but not limited to: *Google Colab* or *Amazon hosting servers*. Another option would be the *Habrok University Cluster* for which access would need to be granted. There are human activities data sets available on which the research can be conducted and there is no need for ethical approval.

4.3.3 Codebase

The development of the codebase for this project followed a systematic approach. Initially, I cloned the repository that housed the code for the selected model, as mentioned in [7]. Subsequently, I meticulously reviewed the repository contents, retaining only those elements directly relevant to the research's objectives. This selective process resulted in the preservation of *3 critical files* (i.e., data_tnda.py, mmhealth_dataset.py, and pamap2_chevclassify.py) from the original codebase, each of which underwent substantial refactoring and updating to align with the project's specific requirements. These retained files encompass all the essential functionalities necessary for effectively processing a single dataset. Given that the experimental procedure involved the use of 3 distinct datasets, I replicated this process 3 times, creating a dedicated set of files for each dataset (i.e., data_{DS-NAME}.py, graph_{DS-NAME}.py, model_{DS-NAME}.py).

To provide further clarity, the first file *data_{DS-NAME}.py* specialises in data preprocessing, focusing on the initial preparation and data manipulation. This initial step forms the basis for subsequent analysis and model training, ensuring that the data is appropriately structured for the project's goals. The implementation of the file itself consists of 4 main parts:

Data Loading: In this first step, the raw data from the dataset files are loaded. Additionally, a new column is appended to each data entry, which contains unique subject identifiers. This step ensures that the data remains distinguishable for different subjects throughout the analysis.

Data Splitting: To facilitate model training and evaluation, the second step revolves around splitting the dataset into three distinct subsets: training, testing, and validation. It was performed based on *subject id*, which guarantees the independence between training and evaluation data. The split ratios employed here are approximately 60% for training, 20% for testing, and 20% for validation. This separation allows for rigorous training, robust testing, and fine-tuning of models.

Sliding Window Application: This section of the file implements a sliding window technique taking into consideration the unique subject identifiers incorporated in the first part. It involves dividing the time-series data into overlapping windows of specific durations. By doing so, the file creates temporal segments of the data, which are essential for capturing dynamic patterns and dependencies in human activity recognition.

Saving Preprocessed Data: After completing the necessary data transformations, this final part of the file saves the preprocessed data to a suitable format along with the correlation matrices used for building the graphs, in the second file. This final step ensures that the prepared data is readily accessible and can be seamlessly integrated into subsequent stages of the research.

Moving onto the second file *graph_{DS-NAME}.py*, its primary objective is to generate the graph representations on which the GCN model will be trained, validated, and tested. As previously mentioned, it plays a crucial role in the data preprocessing pipeline. Specifically, it utilises the correlation matrices computed in the first file to construct the corresponding graphs for each dataset. The correlations are leveraged to establish pairwise relationships between signal channels within the dataset, thereby informing the creation of edges in the resulting graphs. The graph is formed by representing data signal channels as nodes and connecting them with edges based on the strength of pairwise correlations. Strong correlations, with a predefined threshold of *0.2*, lead to the presence of edges connecting the corresponding nodes in the graph. These graphs serve as the foundational data structures upon which the GCN model learns and makes predictions. In this process, a *supervised training setting* is employed in correspondence with all 3 datasets utilised for conducting the experimental procedure. This configuration enables the model to learn from fully labelled data and subsequently make predictions on unseen data.

The third file *model_{DS-NAME}.py* encompasses several critical components of this project. Primarily, it defines the architecture of the GCN model and orchestrates the entire training, validation, and testing processes based on the selected pooling operators. Contained within this file are essential computations for metrics that serve as performance indicators for the model, respectively, the pooling operator. These metrics include the F1 score, test and validation losses, as well as accuracy percentages computed across distinct data subsets, encompassing the training, testing, and validation datasets. Moreover, this file provides visual insights into the model's learning process. Through generated plots, it allows for tracking the model's progress and ascertain whether it has reached an optimal state or if further training is required. Additionally, it plays a crucial role in preserving results. It stores and saves key outcomes from training and validation, including the best state of the model with all its configurations. This feature proves invaluable, as it allows for the swift loading of the best-trained model without the need to repeat the time-consuming training and validation processes. Instead, someone can focus on experimenting with unseen test data and accurately report the model's overall performance.

5 Datasets and Experimental Setup

5.1 Datasets

As previously mentioned, there are various HAR datasets made available which can be utilised for conducting the experimental procedure. The choices entailed for this research project consists of the *TNDA* dataset [31], *MHEALTH* dataset [32] and *PAMAP2* dataset [33].

In terms of the data utilised in this research project from the TNDA dataset [31], it consists of sensor signals that capture *acceleration* data (providing speed and direction), *gyroscope* measurements (indicating angular velocity), and *magnetometer* readings (offering information about magnetic field strength). These signals are recorded across the X, Y, and Z dimensions and collected from sensors placed at the *wrist*, *ankle*, and *back*. A total of 50 subjects participated in the experiments. The sets of subject ids used for *validation* and *testing* are $\{2, 9, 11, 14, 28, 33, 36, 37\}$ and $\{4, 10, 15, 22, 25, 32, 39, 41, 42, 49\}$, respectively. The rest are solely used for *training*. Regarding the *instructions* for the TNDA dataset [31], they involved the paritcipation of subjects in 8 distinct activities. These activities encompassed 3 stationary ones and 5 periodic activities, such as walking, running, cycling, and ascending/ descending stairs. The duration of each activity averaged around 2 minutes.

When it comes to the MHEALTH dataset [32], a variety of sensor signals, capturing essential information about human activities, are contained. These sensor signals include *acceleration* data gathered along the X, Y, and Z axes at specific body locations, such as the *chest* and the *left ankle*, as well as the *right lower arm*. Additionally, the dataset comprises *gyroscope* measurements, recorded specifically at the *left ankle* and the *right lower arm*. Furthermore, *magnetometer* readings are available at the *left ankle* and the *right lower arm* as well, covering the X, Y, and Z dimensions. These sensor signals collectively provide a comprehensive view of various physical activities and human motion patterns. In terms of the *instructions* accompanying the MHEALTH dataset [32], they involve the participation of 10 subjects in a diverse set of 12 activities. These activities include an array of movements such as walking, running, and cycling, thereby offering a rich representation of human physical activity. Each activity's duration typically averages around 2 minutes, ensuring a robust dataset for analysis and experimentation. The sets of subject ids used for *validation* and *testing* are $\{4\}$ and $\{3, 7\}$, respectively. The rest are again solely used for *training*.

Lastly, the PAMAP2 dataset [33] also comprises a diverse array of sensor signals, providing valuable insights into human activities within the scope of the project. These sensor measurements include *acceleration* and *gyroscopic* data, accompanied by *magnetic field readings*. They have been meticulously recorded along the X, Y, and Z axes, covering various body positions, including the *hand*, *chest*, and *ankle*. In terms of the *guidelines* accompanying the PAMAP2 dataset [33], it involves 9 subjects performing an array of 12 activities. These activities encompass various movements, including walking, running, and cycling, showcasing the diverse nature of human physical activity. The sets of subject ids used for *validation* and *testing* are $\{2\}$ and $\{5\}$, respectively. Similar to the other 2 datasets, the rest of ids are assigned for *training*.

5.2 Experimental Setup

The experimental setup for this research encompasses several crucial factors that have been consistently applied across all selected datasets, including TNDA, MHEALTH, and PAMAP2.

The initial stage of the experimental setup involved hyperparameter optimisation. This process resembled a grid search technique, an exhaustive search over a specified set of hyperparameter values. The hyperparameters considered consist of the output channel configurations, the number of Chebyshev polynomial basis functions (referred to as 'k') used in the ChebNet convolutional layer, the learning rate for the GCN model, and batch sizes for training, testing, and validation data. The choice of the tool used for conducting hyperparameter optimisation is *ParameterGrid* from the *sklearn* library. The number of training epochs and a metric known as the *patience* which is used to determine the maximum permissible consecutive rises in validation loss before interrupting the model's training, were manually verified.

The corresponding sets of values used for the optimisation of the parameters previously described, are reported in *Table 1* below. The best performing combination of values for each dataset is highlighted.

Outside of the automated hyperparameter optimisation, a manual investigation was conducted to determine whether the inclusion or exclusion of normalisation layers had any impact on the prediction accuracy of the model for each dataset. The analysis resulted in no normalisation layers needed for all 3 datasets. Moreover, the parameters for configuring input channels and defining the number of activities, essential elements of the model's architecture, were manually adjusted to suit each dataset's requirements. The most suitable value for the input channels' configuration of all 3 datasets is *128*. In regards to the number of activities, they were set accordingly to datasets specifications. For TNDA, there were *8* acitvities, whereas for MHEALTH and PAMAP2 there were *12* in total.

With the optimal parameter combinations determined across all datasets, the next stage for conducting the experiments entailed the analysis of the various pooling operators selected within the scope of the project. The standard operators mean, max, and sum retained a consistent model architecture, which included 3 convolutional layers and 1 linear layer. However, for advanced pooling methods like top-k pooling and the graph multiset transformer, architectural modifications were introduced.

In the case of top-k pooling, the modifications involved the incorporation of 3 pooling layers after each convolutional layer, as opposed to the single pooling layer in the conventional approach. Additionally, in the forward part of the model's architecture, global max pooling and global average pooling are used to compute the maximum and average values for each feature over all nodes in each graph in the batch. The results are then concatenated. After all convolution and pooling operations are completed, the feature representations obtained from the 3 different levels of pooling are combined. This is a common practice in GNNs to capture features at different scales. Furthermore, top-k pooling introduced a new parameter according to [34] called *ratio = 0.8*, which signifies the fraction of nodes to keep after applying the pooling operation. This means that the top 80% of nodes with the highest importance scores will be retained, and the remaining 20% will be discarded.

In the case of GMT pooling, the node features of the convolutional layers were concatenated in the forward pass before the pooling layer was applied. Additionally, GMT pooling also introduced 2 new parameters according to [35]. These parameters are *k = 10* and *heads = 4*, where k specifies the number of representative nodes after the pooling operation (i.e., after applying the pooling operation, the graph will be downsampled to include only k representative nodes) and heads determines the number of multi-head attentions (i.e., allows the model to attend to 4 different parts of the input simultaneously, capturing diverse relationships and patterns).

An important distinction that should be noted between the standard operators and the advanced ones is the number of output channels. For the advanced operators, the number of output channels was fixed at *128*, whereas for the standard operators, it was *multiplied by 2* after each convolutional layer. This design choice was made to improve accuracy and enhance the overall performance of the operators.

The results for all these operators were reported accordingly in the next section.

Dataset	**Parameter**	**Set of Values**
TNDA	Output Channels	{32, 64, **128**}
	Coefficient k	{**2**, 3, 4}
	Learning Rate	{0.0001, **0.0002**, 0.0003, 0.0004, 0.0005, 0.001, 0.005}
	Batch Sizes	{32, **64**, 128}
	Training Epochs	{50, **100**, 125, 200}
	Patience Factor	{25, 30, **35**, 40, 50}
MHEALTH	Output Channels	{32, 64, **128**}
	Coefficient k	{**2**, 3, 4}
	Learning Rate	{0.0001, 0.0002, 0.0003, 0.0004, **0.0005**, 0.001, 0.005}
	Batch Sizes	{**32**, 64, 128}
	Training Epochs	{50, **100**, 125, 200}
	Patience Factor	{25, 30, **35**, 40, 50}
PAMAP2	Output Channels	{32, 64, **128**}
	Coefficient k	{**2**, 3, 4}
	Learning Rate	{0.0001, 0.0002, 0.0003, **0.0004**, 0.0005, 0.001, 0.005}
	Batch Sizes	{**32**, 64, 128}
	Training Epochs	{50, **100**, 125, 200}
	Patience Factor	{25, **30**, 35, 40, 50}

Table 1: Hyperparameter Optimisation Details

6 Results

The results of the conducted experiments are presented in *Table 2* below, offering a comprehensive overview of each pooling operator's performance across the previously mentioned 3 distinct datasets. These findings hold significant importance in guiding the design of GNN-based models for diverse real-world applications related to human activity recognition, as well as in addressing the proposed research questions. As these experiments primarily sought to assess the impact of various pooling operators, the reported metrics, including *accuracy* and *F1 score*, offer valuable insights into the overall classification performance of each pooling operator. The best performing operator for each dataset is highlighted.

Dataset	Pooling Operator	Accuracy (%)	F1 Score
TNDA	**Sum**	**0.911**	**0.913**
	Max	0.903	0.904
	Mean	0.875	0.875
	Top-k	0.793	0.793
	GMT	0.845	0.846
MHEALTH	Sum	0.803	0.778
	Max	0.716	0.683
	Mean	0.760	0.749
	Top-k	0.708	0.708
	GMT	**0.875**	**0.862**
PAMAP2	Sum	0.822	0.814
	Max	**0.830**	**0.826**
	Mean	0.699	0.687
	Top-k	0.757	0.738
	GMT	0.697	0.695

Table 2: Performance Metrics for Pooling Operators on Different Datasets

7 Discussion

7.1 Limitations

To address the first research question, an exploration into the limitations of the selected graph pooling methods and their impact on the performance across different datasets resembles the initial step of the analysis.

Sum pooling method tends to underrepresent complex graph structures due to its inherent simplicity of summing neighbouring nodes' information. Such a simplistic approach may not capture intricate relationships and dependencies in the graph effectively, leading to performance issues. This is a clear observation in the case of the MHEALTH dataset, where the performance of the operator significantly dropped in comparison to the TNDA dataset.

Max pooling primarily focuses on the most dominant feature in a neighbourhood, possibly overlooking valuable information from less prominent features. In datasets reliant on subtler relationships between nodes, max pooling might underperform due to this selective approach. This limitation can be best observed when applying the operator on the MHEALTH dataset in comparison to the other 2 datasets where the accuracies are significantly higher.

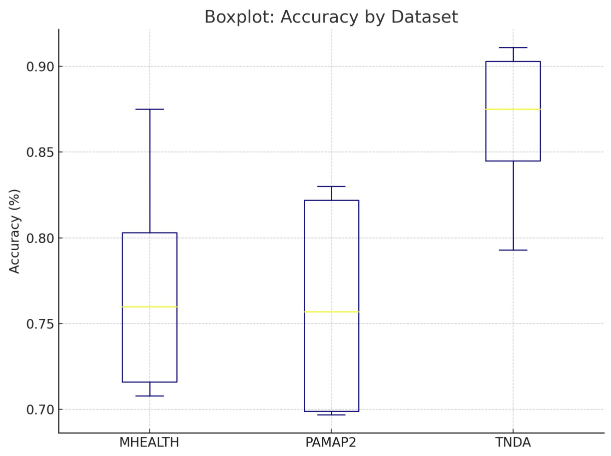
Mean pooling computes an average, which smooths out finer details in the graph. While this can be advantageous in some contexts, it can be detrimental when intricate relationships need preservation. This is clearly noticeable for both MHEALTH and PAMAP2 datasets where the accuracy using this operator significantly dropped by 11.5%, and 17.6%, respectively, in comparison with the TNDA dataset.

When it comes to the advanced operators, *top-k pooling* method can struggle with capturing the full complexity of graph structures as it drops the less influential features. This selective approach might lead to the loss of essential information, affecting performance in scenarios requiring a comprehensive understanding of the data. This is slightly noticeable in the comparison of the results obtained for TNDA against MHEALTH in terms of the accuracy and f1 score produced using this operator.

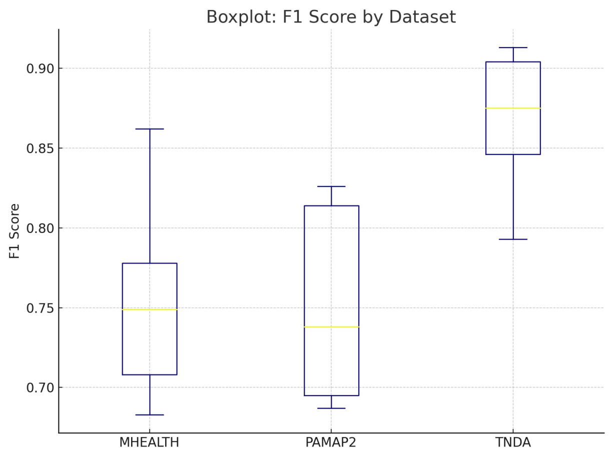
In the case of *GMT pooling*'s design which is based on attention mechanisms that preserve the most significant information might limit its accuracy in capturing nuanced structures. This limitation could hinder its performance in datasets where subtle connections between nodes are crucial. It is best observed when comparing the results obtained for MHEALTH, where the operator has the best performance, against PAMAP2, where it has the worst.

A general limitation found when discussing the analysis of pooling operators is its high dependency on the chosen model architecture. This implies that even minor alterations made to the architecture, such as the values of the parameters used, the number of pooling, convolutional or linear layers can significantly increase the performance in terms of the accuracy and F1 score. This can create a misleading perception of the operator's superiority, making definitive conclusions challenging. It also suggests that a targeted evaluation of pooling operators is almost impossible to assess unless the model's architecture is as consistent as possible for all the operators selected in the analysis.

The boxplots below show the variability of results for the same dataset just by changing the pooling operator.



(a) Boxplot - Accuracy



(b) Boxplot - F1 Score

Figure 1: Pooling Operators Performance Boxplot Charts

Another general limitation encountered is the strong dependency between the performance of the operator and the choice of the dataset. This becomes apparent when considering the performance across the 3 datasets, as different operators excel in each case. Specifically the standard operators, sum and max pooling, perform best for TNDA, and PAMAP2, respectively, while the advanced operator GMT proves most effective for the MHEALTH dataset.

7.2 Metrics

Moving onto answering the second research question, there are a couple of specific criterias discovered which are vital in assessing graph pooling methods' performance.

First of all, a counter argument for the second general limitation presented above, lies within the consistency in performance of the top-k pooling method with a margin error of less than 10% in accuracy for all the selected datasets in the experimental procedure. This implies that the pooling method does not excessively favour or disfavour a specific type of dataset. If a pooling method consistently performs well, or poorly, across various datasets, it suggests a lack of adaptability (i.e., the operator may not be effectively adjusting or responding to the unique features, patterns, or complexities present in each dataset, resulting in a limited capacity to significantly alter the achieved accuracy across diverse datasets). Thus, consistency is a valuable criterion, particularly in applications where the graph structure is diverse.

Second of all, the model's testing time is another significant factor to consider. It can be indicative of the computational efficiency of the pooling method. To prove this, a timed experimental procedure of the model's testing process for 10 iterations, on the PAMAP2 dataset, for each of the 5 operators, resulted in the following order: *top-k (4.5885 sec)<max (5.4590 sec)<mean (7.3740 sec)<sum (8.2661 sec)<GMT (14.8823 sec)*. It can be clearly observed that the top-k operator has the fastest performance, while GMT is the slowest. A shorter running time is generally preferred, especially for large-scale datasets and real-time applications. A model that can process graphs quickly without compromising accuracy is more practical.

8 Conclusion

8.1 Summary of Main Contributions

This research has elucidated the limitations of various graph pooling operators, including sum, max, mean, top-k, and GMT pooling. The analysis showcased the impact of these limitations on the overall performance of GNNs for graph classification tasks. It was demonstrated that the choice of pooling method must be carefully made, as different methods excel in specific scenarios. Notably, this study revealed that while some methods may perform exceptionally in certain contexts, it is crucial to select the appropriate pooling operator to maximise the utility of GNNs in diverse real-world applications.

Addressing the new metrics beyond empirical experiments, the research established the significance of performance consistency across diverse datasets as well as the efficiency in terms of the model's testing time. These criteria offer valuable insights into the adaptability and computational efficiency of the pooling methods, contributing to their selection and applicability in practical settings.

8.2 Future Work

While this research has shed light on various aspects of graph pooling operators and their effects on GNNs' performance in graph classification, there remain several factors for future exploration in this domain.

A notable aspect to address in future research is the development of specific metrics or evaluation methods tailored for assessing graph pooling operators independently. Currently, all the metrics used focus on the overall model performance, making it challenging to understand how each pooling operator influences the results. Introducing new metrics that can provide insights into the behaviour and efficiency of pooling methods themselves would be a valuable endeavour.

Another promising direction for future work involves creating a standardised baseline model that can serve as a common benchmark for testing various pooling operators. Such a baseline model would provide a consistent point of reference for evaluating the relative performance of different operators. This would aid in systematically comparing and contrasting the strengths and weaknesses of pooling techniques, promoting a deeper understanding of their impact.

To further enrich the understanding of graph pooling operators, future studies can expand the analysis to incorporate additional datasets and operators. Investigating a wider range of datasets with diverse characteristics would help in evaluating the generalizability of various pooling methods. By choosing other operators and applying them to different datasets, researchers can gain a more comprehensive perspective and answer additional research questions that might arise.

Exploring these avenues will contribute to the ongoing advancement of graph pooling techniques and their application in graph-based machine learning, opening doors to more efficient and effective approaches for handling complex graph structures.

Bibliography

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[2] C. Morris, *Graph Neural Networks: Graph Classification*, pp. 179–193. Singapore: Springer Nature Singapore, 2022.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[4] C. Liu, Y. Zhan, C. Li, B. Du, J. Wu, W. Hu, T. Liu, and D. Tao, "Graph pooling for graph neural networks: Progress, challenges, and opportunities," 2022.

[5] Z. Wang and S. Ji, "Second-order pooling for graph neural networks," 07 2020.

[6] A. Mohamed, F. Lejarza, S. Cahail, C. Claudel, and E. Thomaz, "HAR-GCNN: Deep Graph CNNs for Human Activity Recognition From Highly Unlabeled Mobile Sensor Data," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pp. 335–340, IEEE, 2022.

[7] Y. Yan, IEEE, S. Member, L. Wang, J. Xiong, W. Lv, L. Ma, J. Wang, J. Zhao, T. Liao, and et al., "Deep Transfer Learning with Graph Neural Network for Sensor-Based Human Activity Recognition," Mar 2022.

[8] A. Nian, X. Zhu, X. Xu, X. Huang, F. Wang, and Y. Zhao, "HGCNN: Deep graph convolutional network for sensor-based human activity recognition," in *2022 8th International Conference on Big Data and Information Analytics (BigDIA)*, pp. 422–427, 2022.

[9] W. Huang, L. Zhang, W. Gao, F. Min, and J. He, "Shallow convolutional neural networks for human activity recognition using wearable sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[10] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Networks*, vol. 129, pp. 203–221, 2020.

[11] AWS, "What is a neural network? ai and ml guide - aws."

[12] IBM, "What is deep learning?," 2023.

[13] B. S.-L. E. R. A. P. A. B. Wiltschko, "A gentle introduction to graph neural networks," 2021.

[14] J. Brownlee, "How do convolutional layers work in deep learning neural networks?," Apr 2020.

[15] D. Mesquita, A. Souza, and S. Kaski, *Rethinking pooling in graph neural networks.* 2020.

[16] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, "Understanding pooling in graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2022.

[17] J. Baek, M. Kang, and S. J. Hwang, "Accurate learning of graph representations with graph multiset pooling," 2021.

[18] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," 2020.

[19] J. Bergstra, J. Ca, and Y. Ca, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, p. 281–305, 2012.

[20] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[21] K. Ganapathy, "A study of genetic algorithms for hyperparameter optimization of neural networks in machine translation," 2020.

[22] C. Morris, "Graph neural networks: Graph classification," *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 179–193, 2022.

[23] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A FAIR COMPARISON OF GRAPH NEURAL NETWORKS FOR GRAPH CLASSIFICATION," Feb 2022.

[24] D. Buterez, J. P. Janet, S. J. Kiddle, D. Oglic, and P. Liò, "Graph neural networks with adaptive readouts," 2022.

[25] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[26] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[27] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2019.

[28] C. Cangea, P. Veličković, N. Jovanović, T. Kipf, and P. Liò, "Towards sparse hierarchical graph classifiers," 2018.

[29] B. Knyazev, G. W. Taylor, and M. R. Amer, "Understanding attention and generalization in graph neural networks," 2019.

[30] J. Baek, M. Kang, and S. J. Hwang, "Accurate learning of graph representations with graph multiset pooling," 2021.

[31] Y. Yan, D. Chen, Y. Liu, J. Zhao, B. Wang, X. Wu, X. Jiao, Y. Chen, H. Li, and X. Ren, "Tndahar," 2021.

[32] G. R. Banos, Oresti and A. Saez, "MHEALTH Dataset." UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5TW22.

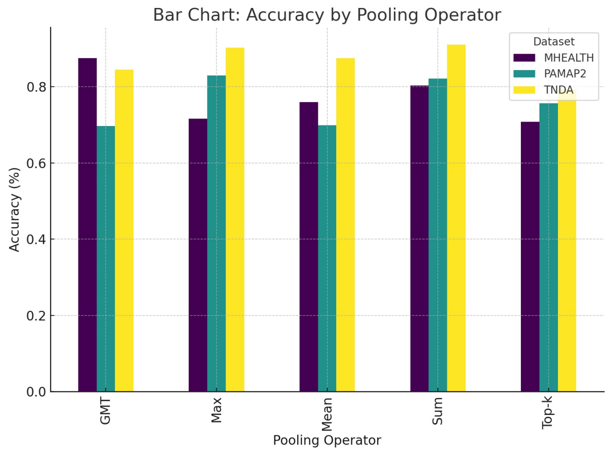
[33] A. Reiss, "PAMAP2 Physical Activity Monitoring." UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5NW2H.

[34] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," May 2019. [Online] URL: https://github.com/pyg-team/pytorch_geometric/blob/master/examples/proteins_topk_pool.py.

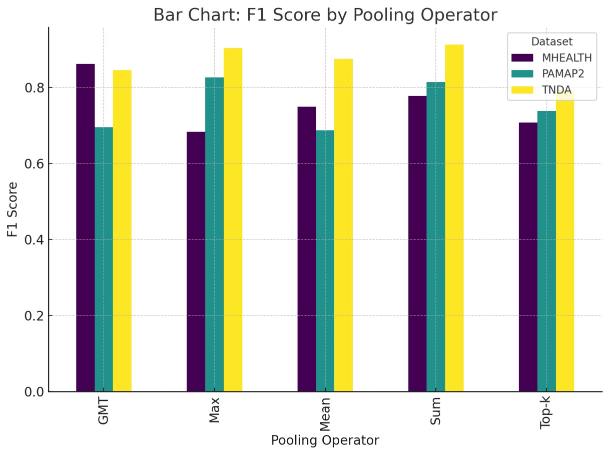
[35] M. Fey and J. E. Lenssen, “Fast graph representation learning with pytorch geometric,” May 2019. [Online] URL: https://github.com/pyg-team/pytorch_geometric/blob/master/examples/proteins_gmt.py.

Appendices

I have provided insightful charts below that facilitate a clearer examination of how the pooling operators perform across all 3 datasets.

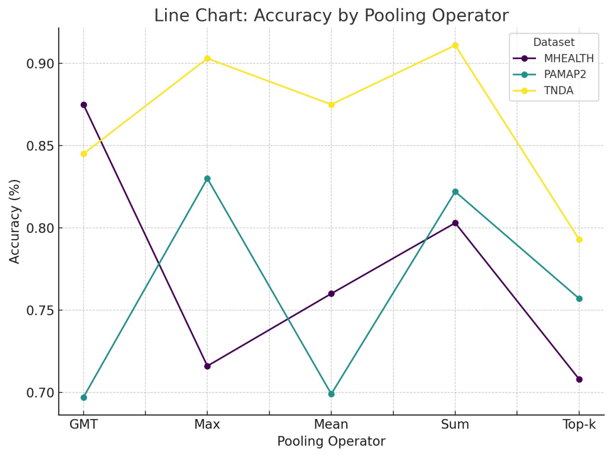


(a) Accuracy

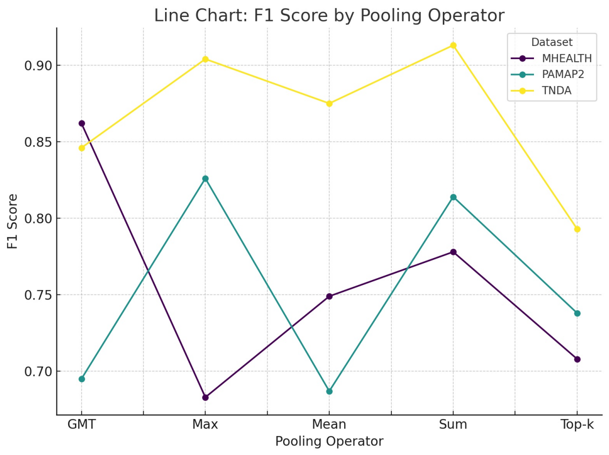


(b) F1 Score

Figure 2: Pooling Operators Performance Bar Charts



(a) Accuracy



(b) F1 Score

Figure 3: Pooling Operators Performance Line Charts