algorithms

MDPI

Article

On Information Granulation via Data Clustering for Granular Computing-Based Pattern Recognition: A Graph Embedding Case Study

Alessio Martino [1,*], Luca Baldini [2] and Antonello Rizzi [2]

[1] Department of Business and Management, LUISS University, Viale Romania 32, 00197 Rome, Italy
[2] Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza", Via Eudossiana 18, 00184 Rome, Italy; luca.baldini@uniroma1.it (L.B.); antonello.rizzi@uniroma1.it (A.R.)
* Correspondence: amartino@luiss.it; Tel.: +39-06-8522-5957

Abstract: Granular Computing is a powerful information processing paradigm, particularly useful for the synthesis of pattern recognition systems in structured domains (e.g., graphs or sequences). According to this paradigm, granules of information play the pivotal role of describing the underlying (possibly complex) process, starting from the available data. Under a pattern recognition viewpoint, granules of information can be exploited for the synthesis of semantically sound embedding spaces, where common supervised or unsupervised problems can be solved via standard machine learning algorithms. In this work, we show a comparison between different strategies for the automatic synthesis of information granules in the context of graph classification. These strategies mainly differ on the specific topology adopted for subgraphs considered as candidate information granules and the possibility of using or neglecting the ground-truth class labels in the granulation process. Computational results on 10 different open-access datasets show that by using a class-aware granulation, performances tend to improve (regardless of the information granules topology), counterbalanced by a possibly higher number of information granules.

Keywords: structural pattern recognition; supervised learning; graph classification; inexact graph matching; granular computing; information granulation; data mining and knowledge discovery

Citation: Martino, A.; Baldini, L.; Rizzi, A. On Information Granulation via Data Clustering for Granular Computing-Based Pattern Recognition: A Graph Embedding Case Study. *Algorithms* **2022**, *15*, 148. https://doi.org/10.3390/a15050148

Academic Editor: Xiao Huang

Received: 11 April 2022
Accepted: 24 April 2022
Published: 27 April 2022

1. Introduction

In the early 2000s, Granular Computing emerged as a novel information processing paradigm that exploits pivotal mathematical structures called *granules of information* to describe an underlying set of (likely complex) data, describing a (likely complex) process under analysis [1,2]. The concept of information granulation dates back to the mid-1990s, thanks to soft computing and fuzzy logic pioneer Lotfi Aliasker Zadeh. In their words:

> *Among the basic concepts which underlie human cognition there are three that stand out in importance. The three are: granulation, organization and causation.*
>
> L.A. Zadeh [3]

and

> *Informally, granulation of an object A results in a collection of granules of A, with a granule being a clump of objects (or points) which are drawn together by indistinguishability, similarity, proximity or functionality. In this sense, the granules of a human body are the head, neck, arms, chest, etc. In turn, the granules of a head are the forehead, cheeks, nose, ears, eyes, hair, etc. In general, granulation is hierarchical in nature. A familiar example is granulation of time into years, years in months, months into days and so on.*
>
> L.A. Zadeh [3]

This philosophical viewpoint behind the birth of Granular Computing as human-inspired information processing paradigm has been embraced by other well-known scholars, notably Ronald R. Yager. In their words:

> *Language, which is central to most human cognitive activities, is based on granularization. In addition human facilities for distinctions are limited. Being limited, at the very least by language and perhaps additionally by their ability to perceive, human beings have been developed a granular view of the world. Thus, we see that the objects with which humankind perceives, measures, conceptualizes and reasons are granular.*
>
> R.R. Yager and D. Filev [4]

As the 1990s marked the golden age of fuzzy logic and fuzzy-based pattern recognition, L.A. Zadeh further argues that information granules should be inherently fuzzy since most of human reason and concept formation are fuzzy rather than crisp [3,5].

Clearly, Granular Computing is not a set of computational pipelines and is not a set of algorithms; rather, it can be considered as a goal-driven umbrella that, according to Y.Y. Yao, in their "Granular Computing manifesto" [6] should fulfill the following points:

- Be a truthful representation of the real world;
- Be consistent with human thinking and problem solving;
- Allow a simplification of the problem;
- Provide economic and low-cost solutions.

Thus, in the realm of Granular Computing, we can easily find techniques that have not been developed for Granular Computing but, nonetheless, they satisfy the above goals. Indeed, information granulation can be performed by a plethora of different strategies, notably fuzzy sets [3,7,8], rough sets [8–10], shadowed sets [11], interval analysis [12] and data clustering [13–15].

In a general sense, the aim of a clustering algorithm is to group a finite set of data into a finite number of groups in such a way that the elements within the same cluster are more similar with each other with respect to elements belonging to other clusters [16]. The vagueness of such definition is not by chance as data clustering is an ill-posed problem. There are two ways to confirm such statement. First, according to J. Hadamard [17], a problem is said to be well-posed if a unique solution exists and the solution's behavior changes continuously with the initial conditions. Data clustering does not satisfy (at least) the first requirement. Second, data clustering is an inverse problem (i.e., when we want to calculate, starting from a finite set of data, the causal factors that produced them), which are known to be ill-posed.

The lack of notion of what is a valid cluster and, in consequence, what is an appropriate clustering algorithm to detect such clusters depends on the application at hand [18], leading to the development of several families of clustering algorithms available in the current literature [19]. The interested reader can refer to [16,20] for detailed surveys on clustering algorithms. Here, we briefly discuss two families of clustering algorithms widely employed in a Granular Computing setting: hierarchical and partitional clustering. Hierarchical clustering algorithms aim at building a dendrogram that represents the dependency among clusters; in particular, hierarchical clustering algorithms can be further divided in two main families:

- Agglomerative clustering algorithms, which work in a bottom-to-top fashion by starting with each data point forming its own cluster and then merging clusters until all data points belong to the same cluster;
- Divisive clustering algorithms, which work in a top-to-bottom fashion by starting with a single cluster embracing the entire dataset and then iteratively splitting the clusters until each data point forms its own cluster.

Conversely, partitional clustering algorithms aim at assigning each data object to one and only one cluster, where the number of clusters can be set a priori. In the latter

case, *k*-means stems as the flagship of partitional algorithms [21], alongside its extensions towards non-Euclidean dissimilarity measures [22,23] and fuzzy extensions [24].

In the former case (i.e., hierarchical clustering), the algorithm natively returns a multi-scale overview of the dataset (i.e., the dendrogram linking its clusters at different scales). In the latter case (i.e., partitional clustering), one can let fewer or more clusters emerge, leading to partitions having different sizes. The possibility of analyzing the data (hence, letting information granules to emerge) at different levels of abstraction is at the very heart of the Granular Computing paradigm, which is why both approaches are commonly used in a Granular Computing approach.

Recently, the Granular Computing paradigm has been employed for the synthesis of pattern recognition systems, as well as in structured domains such as graphs, sequences and images [25,26]. The rationale behind these pattern recognition systems is to automatically extract recurrent and/or meaningful substructures (i.e., subgraphs, subsequences, portions of images) suitable to be considered as granules of information. On the top of these pivotal elements, it is possible to build an embedding space in such a way that the pattern recognition problem is cast from the structured domain towards the Euclidean space. The latter, being a metric space, allows to comfortably use one of the many statistical classifiers currently available in the pattern recognition and machine learning literature [27].

In this work, we exploit clustering-based approaches for the automatic synthesis of information granules in the context of graph classification. The information granules reflect pivotal subgraphs extracted from the training data endowed with high discriminative power. On the top of these information granules, we perform an embedding procedure thanks to the symbolic histograms approach. In particular, we review and compare two different candidate topologies for the synthesis of granules of information (paths and cliques) and we compare two additional strategies for their synthesis: a stratified approach, where the ground-truth labels of the classification problem play an important role in the information granules synthesis, and a non-stratified approach, where the ground-truth labels are completely discarded in the process.

The remainder of the paper is structured as follows: in Section 2, we describe the four main building blocks of the GRALG framework (extraction and synthesis of granules of information, graph embedding and classification), which we exploit in order to perform our two-fold investigation. In Section 3, we discuss in detail four granulation strategies based on different combination of subgraph topologies (paths vs. cliques) and stratification (class-aware vs. no stratification). In Section 4, we detail how these building blocks co-operate in order to synthesize an optimized model for graph classification. In Section 5, we show the computational results obtained by the four combinations of topology and granulation and, finally, Section 6 concludes the paper.

2. High-Level Framework Description

The GRALG framework is composed of the following four main modules:

- Extractor, which is in charge of extracting, from the training set, a suitable set of candidate information granules;
- Granulator, which is in charge of building an alphabet of symbols starting from the candidate information granules provided by the Extractor block;
- Embedder, which is in charge of mapping a graph dataset towards the Euclidean space;
- Classifier, which is in charge of training and testing a suitable classification system in the Euclidean space spanned by the Embedder block.

2.1. Extractor and Granulator

Let P be an unknown, oriented and possibly complex process to be modeled, where the inputs are annotated graphs and where the output domain is a finite set of class labels. Furthermore, let $\mathcal{S}$ be an input-output sampling of P and let $\mathcal{S}_{\text{tr}}$, $\mathcal{S}_{\text{val}}$ and $\mathcal{S}_{\text{ts}}$ be a split of $\mathcal{S}$ into training, validation and test sets, respectively. The split should be performed so that each subset should share the same statistics, in order to be considered as

valid representation of the same process. Moreover, this split must satisfy the following partition properties:

- The union of the three sets yields the original set: $\mathcal{S}_{\text{tr}} \cup \mathcal{S}_{\text{val}} \cup \mathcal{S}_{\text{ts}} = \mathcal{S}$;
- The intersection of any two distinct sets is empty: $\mathcal{S}_{\text{tr}} \cap \mathcal{S}_{\text{val}} = \mathcal{S}_{\text{tr}} \cap \mathcal{S}_{\text{ts}} = \mathcal{S}_{\text{ts}} \cap \mathcal{S}_{\text{val}} = \emptyset$.

The Extractor is a block that takes as input $\mathcal{S}_{\text{tr}}$ and returns a bucket $\mathcal{B}$ of subgraphs drawn from graphs in $\mathcal{S}_{\text{tr}}$.

Conversely, the Granulator block takes as input $\mathcal{B}$ and returns an alphabet of symbols $\mathcal{A} \subset \mathcal{B}$, namely suitable granules of information, by running a clustering algorithm over $\mathcal{B}$, possibly electing the representative elements of the clusters as granules of information [13–15,28]. In particular, the latter runs a Basic Sequential Algorithmic Scheme (BSAS) [29] clustering algorithm driven by the following parameters:

- Q, namely, the maximum number of allowed clusters;
- θ, namely, the maximum radius for including a pattern into one of the already-discovered clusters;
- A dissimilarity measure $d(\cdot, \cdot)$ between any two patterns.

The clustering algorithm addresses the dissimilarity between patterns (i.e., graphs) thanks to a parametric Graph Edit Distance (GED) [30,31] dissimilarity measure called node Best Match First (nBMF). The interested reader is referred to ([32], Appendix A) for a full mathematical formulation of nBMF. At the end of the clustering procedure, each cluster is evaluated by means of a suitable quality index that jointly takes into account its cardinality (i.e., number of patterns) and compactness (i.e., average pattern-to-center distance) and only the centers of well-formed (i.e., compact and populated) clusters are admitted to be part of the alphabet.

Since the Extractor and the Granulator blocks are the main focus of this paper, the four variants we aim to investigate will be thoroughly described in Section 3.

2.2. Embedder

The Embedder block takes as input the alphabet as returned by the Granulator block and runs an embedding function in order to cast each graph (belonging to an input graph set, e.g., $\mathcal{S}_{\text{tr}}$) towards the Euclidean space.

The mapping function yields the so-called symbolic histogram [33] by transforming each input graph $\mathcal{G}$ into an n-length feature vector of the form:

$$\mathbf{h}(\mathcal{A}, \mathcal{G}) = [\text{occ}(s_1, \mathcal{G}), \ldots, \text{occ}(s_n, \mathcal{G})] \tag{1}$$

where $\mathcal{A} = \{s_1, \ldots, s_n\}$ and $\text{occ} : \mathcal{A} \times \mathcal{G} \to \mathbb{N}_0^+$ is the enumeration function that counts the number of times each symbol $s \in \mathcal{A}$ appears in $\mathcal{G}$.

The counting procedure operates as follows:

1. The input graph $\mathcal{G}$ is decomposed into its constituent parts, yielding a decomposition $\mathcal{G}' = \{g_1, \ldots, g_k\}$. For the sake of consistency with the Extractor and Granulator modules, in case of path-based subgraphs, the decomposition follows a Breadth First Search strategy, whereas in case of cliques-based granulation, the decomposition follows the maximal clique decomposition.
2. For the i^{th} symbol in $\mathcal{A}$:
   (a) The pairwise dissimilarities between s_i and all subgraphs in $\mathcal{G}'$ are evaluated;
   (b) All dissimilarities below a threshold ζ_i are retained and considered as a 'hit';
   (c) The i^{th} entry in $\mathbf{h}$ is filled with the number of occurrences (i.e., the number of 'hits');
3. Repeat step 2 for $i = 1, \ldots, n$.

There are two important aspects in this procedure that need to be addressed. First, the threshold ζ_i for scoring a 'hit' is defined as:

$$\zeta_i = 1.1 \cdot \Phi(\mathcal{K}_i) \tag{2}$$

where $\mathcal{K}_i$ is the cluster whose representative element is s_i and $\Phi(\cdot)$ is a function that evaluates the compactness of the cluster (further details in Section 3). In this manner, the threshold scales in a symbol-aware fashion by accounting the size of its cluster.

Second, the pairwise dissimilarities between symbols and subgraphs is evaluated by means of the very same nBMF dissimilarity measure already used by the Granulator block.

2.3. Classifier

The Classifier block trains a classification system on the embedded version of $\mathcal{S}_{\text{tr}}$, say $\mathbf{H}_{\text{tr}}$, namely an $|\mathcal{S}_{\text{tr}}| \times n$ instance matrix with patterns (i.e., graphs from $\mathcal{S}_{\text{tr}}$) organized as rows.

In order to validate the behavior of the classifier, the Classifier block also needs the embedded version of $\mathcal{S}_{\text{val}}$, say $\mathbf{H}_{\text{val}}$. The ability of the classification system, previously trained on $\mathbf{H}_{\text{tr}}$, in predicting the ground-truth labels of $\mathbf{H}_{\text{val}}$ dictates the performance of the Classifier.

In this work, we use a K-Nearest Neighbors (K-NN) decision rule [34] with $K = 5$ as the classification system.

3. Information Granulation Strategies

3.1. Random Walk

The random walk extractor, proposed in [32], takes as input a bucket $\mathcal{B}$ of candidate information granules extracted via a plain random walk on the graphs belonging to the training set.

In a plain random walk [35,36], the next-hop $u \in \mathcal{V}$ is chosen uniformly at random among the neighbors of the current node $v \in \mathcal{V}$. Formally, the probability of moving from v to u is:

$$p_{v \to u} = \begin{cases} \frac{1}{\deg(v)}, & \text{if } u \in \mathcal{N}(v) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{N}(v)$ is the neighborhood of node v and $\deg(v)$ is its degree, i.e., $\deg(v) = |\mathcal{N}(v)|$.

Populating the bucket $\mathcal{B}$ relies on two important parameters:

- W, the user-defined number of subgraphs in $\mathcal{B}$, i.e., $W = |\mathcal{B}|$;
- o, the user-defined maximum number of nodes for subgraphs in $\mathcal{B}$.

and it works as follows:

1. Start with $\mathcal{B} = \emptyset$;
2. Let $W' = \frac{W}{o}$ be the number of subgraphs to be extracted for each of the candidate subgraph orders;
3. For $l = 1, \ldots, o$:
   (a) Let $\mathcal{B}^{(l)} = \emptyset$ be a temporary bucket containing only subgraphs of order l;
   (b) Until $|\mathcal{B}^{(l)}|$ equals W':
      i. Extract uniformly at random a graph $\mathcal{G}$ from the training set;
      ii. Extract uniformly at random a node v from $\mathcal{G}$;
      iii. Start a simple random walk of length l from node v;
      iv. The subgraph emerged from the random walk is added to $\mathcal{B}^{(l)}$;
   (c) $\mathcal{B} = \mathcal{B} \cup \mathcal{B}^{(l)}$;

The so-collected bucket $\mathcal{B}$ is the main input to the Granulator module.

As anticipated in Section 2, the Granulator block runs the BSAS clustering algorithm on $\mathcal{B}$ in order to extract suitable information granules, properly collected in the alphabet $\mathcal{A}$. Given a set of m candidate values for θ, i.e., $\{\theta_1, \ldots, \theta_m\}$, the Granulator block runs m instances of BSAS, each with a different value for θ. Recalling that θ identifies the radius of inclusion, exploring different θ values allows us to explore $\mathcal{B}$ at different levels of resolutions, which, in turn, will yield granules of information at different levels of granularity [37–40]. Recalling that each instance of BSAS is bounded by a maximum number of clusters Q, the Granulation block yields at most $\mathcal{O}(Q \cdot m)$ clusters in the worst case. Each cluster returned by the clustering ensemble, regardless of the θ value that generated it, undergoes a cluster quality evaluation in order to address whether that cluster is suitable for being elected as granule of information. Specifically, for any cluster $\mathcal{K}$, we define a cluster quality index $F(\mathcal{K})$ that reads as:

$$F(\mathcal{K}) = \eta \cdot \Phi(\mathcal{K}) + (1 - \eta) \cdot \Psi(\mathcal{K}) \tag{4}$$

namely, as a linear convex combination between the compactness of the cluster $\Phi(\mathcal{K})$ and its cardinality $\Psi(\mathcal{K})$ weighted by a trade-off parameter $\eta \in [0, 1]$. In turn, the compactness of the cluster is defined as the average pattern-to-center distance:

$$\Phi(\mathcal{K}) = \frac{1}{|\mathcal{K}| - 1} \sum_i d(g^\star, g_i) \tag{5}$$

where $g_i \in \mathcal{K}$ is the ith pattern of cluster $\mathcal{K}$ and $g^\star$ is the representative element of the cluster. Since the Granulator deals with non-geometric entities (i.e., graphs), we consider the medoid of the cluster as its representative [41]. Finally, the cardinality of the cluster is defined as the relative size of the cluster $\mathcal{K}$ with respect to the overall number of candidate information granules:

$$\Psi(\mathcal{K}) = 1 - \frac{|\mathcal{K}|}{|\mathcal{B}|} \tag{6}$$

Since both $\Phi(\mathcal{K})$ and $\Psi(\mathcal{K})$ are negative-oriented, a cluster is considered for being part of the alphabet $\mathcal{A}$ if and only if its quality index $F(\mathcal{K})$ is below a threshold $\tau_F \in [0, 1]$. The medoids whose clusters satisfying such condition compose the alphabet of symbols $\mathcal{A}$.

3.2. Clique

The clique extractor aims at investigating a particular subgraph topology: the clique, namely an induced subgraph that is complete [42,43].

In this scenario, the bucket $\mathcal{B}$ will be filled with a subset of the cliques extracted from the graphs belonging to the training set. The end-user is still required to specify W, as for Section 3.1, yet the maximal order parameter o is meaningless as cliques are concerned since the formation (and the size) of a clique is strictly topology-dependent rather than user-defined.

The clique extractor works as follows:

1. Start with $\mathcal{B} = \emptyset$;
2. For each graph $\mathcal{G}$ from the training set:
   (a) Evaluate $\mathcal{C}$ as the maximal clique decomposition of $\mathcal{G}$. The maximal clique decomposition of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be evaluated thanks to the Bron–Kerbosh algorithm [44] with a worst-case complexity of $\mathcal{O}(3^{|\mathcal{V}|/3})$ [45];
   (b) $\mathcal{B} = \mathcal{B} \cup \mathcal{C}$;
3. Let $\mathcal{B}'$ be a set of W subgraphs selected uniformly at random from $\mathcal{B}$;
4. $\mathcal{B} \leftarrow \mathcal{B}'$.

The so-collected bucket $\mathcal{B}$ is the main input to the granulator module.

The Granulator block works exactly as the one described in Section 3.1, yet on a bucket $\mathcal{B}$ composed of cliques only.

3.3. Stratified Clique

The two extractors and granulator strategies in Sections 3.1 and 3.2 populate the bucket $\mathcal{B}$ uniformly at random. Such procedures present the following two potential drawbacks:

- The information about the ground-truth labels (freely available in classification problems) is not exploited in the extraction and granulation stages;
- A uniformly at random selection can bias the contents of $\mathcal{B}$, especially in case of unbalanced datasets; indeed, training graphs pertaining to the majority class have a higher change of being selected.

In order to overcome both problems, we further proposed a stratified clique-based extractor and granulator [46]. The main objective of the stratified extractor is to build the bucket $\mathcal{B}$ as a set-of-sets $\mathcal{B} = \{\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(p)}\}$, with p being the number of classes for the classification problem at hand, with the constraint that $\mathcal{B}^{(i)}$ contains subgraphs drawn from the subset of training graphs belonging to the ith class only.

The stratified clique-based extractor works as follows:

1. For each ground-truth class $i = 1, \ldots, p$:
   (a) Let $\mathcal{S}_{\text{tr}}^{(i)}$ be the subset of the training set containing only patterns belonging to class i;
   (b) Calculate the (relative) frequency of the ith class as $f_i = \left\lfloor \frac{|\mathcal{S}_{\text{tr}}^{(i)}|}{|\mathcal{S}_{\text{tr}}|} + \frac{1}{2} \right\rfloor$, where the operator $\left\lfloor x + \frac{1}{2} \right\rfloor$ rounds x to the nearest integer;
   (c) Evaluate $W_i = \left\lfloor W \cdot f_i + \frac{1}{2} \right\rfloor$, namely the size of $\mathcal{B}^{(i)}$;
   (d) Set $\mathcal{B}^{(i)} = \emptyset$;
   (e) For each graph $\mathcal{G} \in \mathcal{S}_{\text{tr}}^{(i)}$:
      i. Evaluate $\mathcal{C}$ as the maximal clique decomposition of $\mathcal{G}$;
      ii. Update $\mathcal{B}^{(i)} = \mathcal{B}^{(i)} \cup \mathcal{C}$.
   (f) If $|\mathcal{B}^{(i)}| > W_i$, then replace $\mathcal{B}^{(i)}$ with a uniform random selection of W_i of its own subgraphs.

Due to the set-of-sets nature of the stratified bucket, the Granulator described in Section 3.1 and later employed in Section 3.2 loses its effectiveness. To overcome this problem, we let p different Granulators to run independently on each of the class-stratified buckets, yielding a class-related alphabet, say $\mathcal{A}^{(i)}$ for the ith class, to be merged together later in order to form the overall alphabet $\mathcal{A}$. Specifically:

1. Start with $\mathcal{A} = \emptyset$;
2. For each ground-truth class $i = 1, \ldots, p$:
   (a) Consider the sub-bucket $\mathcal{B}^{(i)}$ containing subgraphs drawn from $\mathcal{S}_{\text{tr}}^{(i)}$;
   (b) Run the Granulator (see Section 3.1) on $\mathcal{B}^{(i)}$, yielding $\mathcal{A}^{(i)}$;
   (c) $\mathcal{A} = \mathcal{A} \cup \mathcal{A}^{(i)}$.

The above procedure allows to have, in the final alphabet $\mathcal{A}$, information granules extracted from all of the problem-related classes, which should yield not only a better characterization of the data by means of a thorough extraction of pivotal elements, but also a better separation of the classes themselves. Indeed, we advance the hypothesis that if a particular subgraph is pivotal to characterize a given problem-related class, then the number of occurrences in the symbolic histogram (cf. Equation (1)) will be higher if the graph to be embedded belongs to the very same class. This comes with the drawback of a potentially higher-dimensional alphabet: indeed, since the Granulator from Section 3.1 has to be repeated p times, the worst-case size of $\mathcal{A}$ grows as $\mathcal{O}(p \cdot m \cdot Q)$.

3.4. Stratified Random Walk

The stratified path-based extractor, originally proposed in [41], works as follows:

1. For each ground-truth class $i = 1, \ldots, p$:
   (a) Let $\mathcal{S}_{\text{tr}}^{(i)}$ be the subset of the training set containing only patterns belonging to class i;
   (b) Calculate the (relative) frequency of the ith class as $f_i = \left\lfloor \frac{|\mathcal{S}_{\text{tr}}^{(i)}|}{|\mathcal{S}_{\text{tr}}|} + \frac{1}{2} \right\rfloor$;
   (c) Evaluate $W_i = \left\lfloor W \cdot f_i + \frac{1}{2} \right\rfloor$, namely the size of $\mathcal{B}^{(i)}$;
   (d) Evaluate $W_i' = \left\lfloor W_i \cdot o + \frac{1}{2} \right\rfloor$, namely the number of subgraphs to be extracted for each of the candidate subgraphs order, yet considering only graphs belonging to class i;
   (e) Set $\mathcal{B}^{(i)} = \emptyset$;
   (f) For $l = 1, \ldots, o$:
      i. Set $\mathcal{B}^{(i,l)} = \emptyset$, namely a temporary bucket that will hold subgraphs of order l extracted from graphs of class i;
      ii. Until $|\mathcal{B}^{(i,l)}|$ is equal to W_i':
         A. Extract uniformly at random a graph $\mathcal{G}$ from $\mathcal{S}_{\text{tr}}^{(i)}$;
         B. Extract uniformly at random a node v from $\mathcal{G}$;
         C. Start a simple random walk of length l from node v;
         D. The subgraph emerged from the random walk is added to $\mathcal{B}^{(i,l)}$;
      iii. $\mathcal{B}^{(i)} = \mathcal{B}^{(i)} \cup \mathcal{B}^{(i,l)}$

The Granulator block works exactly as the one described in Section 3.3, yet on a bucket-of-buckets $\mathcal{B}$ composed of random walks (cf. Section 3.1).

4. Model Synthesis and Testing

In this section, we explain in detail how the above-described four blocks (Extractor, Granulator, Embedder and Classifier) co-operate in order to synthesize the classification model and subsequent testing of its final performance on the test set. Since there are several hyper-parameters involved in the model synthesis, notably:

- The clustering algorithm parameters for the Granulator block;
- The weights for the nBMF dissimilarity measure (used in both Granulator and Embedder).

We employ a differential evolution algorithm [47] for an automatic tuning of these parameters, hence driving the overall model synthesis.

The model synthesis starts by triggering the Extractor block, which yields the set of candidate information granules $\mathcal{B}$ by properly processing the training graphs. As $\mathcal{B}$ is returned, the differential evolution optimization scheme can take place.

The search space for the optimization procedure is defined as:

$$\left[\begin{array}{cccccccccc} Q & \eta & \tau_F & w_{\text{sub}}^{\text{node}} & w_{\text{ins}}^{\text{node}} & w_{\text{del}}^{\text{node}} & w_{\text{sub}}^{\text{edge}} & w_{\text{ins}}^{\text{edge}} & w_{\text{del}}^{\text{edge}} & \Pi_{\text{edge}} \quad \Pi_{\text{node}} \end{array} \right] \tag{7}$$

Each individual from the evolving population forwards $\mathcal{B}$ to the Granulator block. As anticipated in Section 2, the Granulator runs a BSAS-based ensemble driven by a suitable dissimilarity measure $d(\cdot, \cdot)$, a maximum number of allowed clusters Q and a threshold value θ. Recall that the dissimilarity measure between any two patterns is evaluated by a parametric nBMF dissimilarity measure that, in turn, depends on:

- Six real-valued weights that account for the importance of each atomic transformation (insertion, deletion and substitution) on nodes and edges: $w_{\text{sub}}^{\text{node}}$, $w_{\text{ins}}^{\text{node}}$, $w_{\text{del}}^{\text{node}}$, $w_{\text{sub}}^{\text{edge}}$, $w_{\text{ins}}^{\text{edge}}$, $w_{\text{del}}^{\text{edge}}$;

- A set Π_{edge} of parameters, if needed, in order to drive the dissimilarity measure between edges;
- A set Π_{node} of parameters, if needed, in order to drive the dissimilarity measure between nodes.

At the end of the Granulation stage, the alphabet $\mathcal{A}$ is available for feeding the Embedder block, which yields the embedded version of $\mathcal{S}_{\text{tr}}$ and $\mathcal{S}_{\text{val}}$, namely $\mathbf{H}_{\text{tr}}$ and $\mathbf{H}_{\text{val}}$.

These two instance matrices are finally fed to the Classifier block, which trains a K-NN decision rule on $\mathbf{H}_{\text{tr}}$ and predicts the ground-truth labels on $\mathbf{H}_{\text{val}}$.

Each individual is evaluated by means of a fitness function (to be minimized) formalized as follows:

$$f = \alpha \cdot \pi + (1 - \alpha) \cdot \sigma \tag{8}$$

where π is an error term and σ is a penalty term. Specifically:

$$\pi = 1 - \frac{1}{p} \sum_{i=1}^{p} \text{rec}(i) \tag{9}$$

where $\text{rec}(i)$ is the recall of class i. Hence, Equation (9) reads as the complement of the balanced accuracy [48]. On the other hand, σ is a penalty term defined as:

$$\sigma = \frac{|\mathcal{A}|}{|\mathcal{B}|} \tag{10}$$

in order to penalize large alphabets and foster the optimization towards smaller alphabets.

At the end of the optimization stage, the best individual is retained, along with the best alphabet $\mathcal{A}^\star$ synthesized with its genetic code and the two instance matrices $\mathbf{H}^\star_{\text{tr}}$ and $\mathbf{H}^\star_{\text{val}}$ embedded against $\mathcal{A}^\star$.

The choice of the trade-off parameter $\alpha \in [0, 1]$ in Equation (8) that weights performance against dimensionality can hardly be set a priori. In order to overcome this problem, a second lightweight optimization stage can be employed in order to further reduce the size of the alphabet.

In this second optimization stage, the genetic code is simply a binary mask:

$$\mathbf{m} = \{0, 1\}^{|\mathcal{A}^\star|} \tag{11}$$

and each individual from the evolving population:

1. Projects $\mathbf{H}^\star_{\text{tr}}$ and $\mathbf{H}^\star_{\text{val}}$ on the subset of columns spanned by the indices $\{i : \mathbf{m}_i = 1\}$, say $\mathbf{H}^{\star\prime}_{\text{tr}}$ and $\mathbf{H}^{\star\prime}_{\text{val}}$;
2. Trains the classifier on $\mathbf{H}^{\star\prime}_{\text{tr}}$ and validates its performance on $\mathbf{H}^{\star\prime}_{\text{val}}$.

The fitness function (to be minimized) reads as:

$$f' = \beta \cdot \pi + (1 - \beta) \cdot \frac{|\{i : \mathbf{m}_i = 1\}|}{|\mathcal{A}^\star|} \tag{12}$$

where the leftmost term reads as in Equation (9) and the rightmost term aims at fostering the evolution towards smaller alphabets by preferring sparse binary masks.

At the end of this second optimization stage, the (possibly) reduced alphabet $\mathcal{A}^{\star\prime} \subseteq \mathcal{A}^\star$ is retained, along with the projected training instance matrix $\mathbf{H}^{\star\prime}_{\text{tr}}$. The test set is itself embedded against $\mathcal{A}^{\star\prime}$, yielding $\mathbf{H}^{\star\prime}_{\text{ts}}$. The classifier is finally trained on $\mathbf{H}^{\star\prime}_{\text{tr}}$ and its final performance is evaluated on $\mathbf{H}^{\star\prime}_{\text{ts}}$.

5. Tests and Results

5.1. Datasets Description

The proposed comparison among different data granulation strategies involves 11 different datasets, with the first 6 datasets being taken from the IAM Repository [49] and

the remaining ones being taken from the TUDataset Repository [50]. A brief description of the datasets, along with the formal definition of the dissimilarity measures between nodes and edges, follows:

AIDS: The AIDS dataset consists of 2000 graphs representing molecules showing activity or not against HIV (two classes). Molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled by a three-element tuple that collects the 2D $\langle x, y \rangle$ coordinates of the atom, the chemical symbol (categorical) and its charge (integer). Although edges are originally labeled with the valence of the linkage, such a value has been discarded since it is not useful for the classification task.

Letter-L: The Letter-L dataset involves graphs that represent distorted letter drawings with a low level of distortion. The recognition task involves the 15 capital letters of the Roman alphabet that can be represented by straight lines only. Each handwritten letter is transformed into a graph by representing lines as edges and endpoints of lines as nodes. Each node is labeled by a two-dimensional real-valued vector giving its position within a reference coordinate system. Conversely, edges are unlabeled.

Letter-M: Same as Letter-L, but with medium level of distortion in handwritten digits.

Letter-H: Same as Letter-L, but with high level of distortion in handwritten digits.

GREC: The GREC dataset consists of symbols from electronic and architectural drawings and, after suitable pre-processing, graphs are extracted from such images. Ending points, corners, intersections and circles are represented by nodes and labeled with a two-dimensional attribute giving their position. The nodes are connected by undirected edges, which are labeled as "line" or "arc". An additional attribute specifies the "angle" with respect to the horizontal direction or the diameter in case of arcs.

Mutagenicity: The Mutagenicity dataset consists of 4337 graphs corresponding to chemical compounds divided into two classes, depending on their respective mutagenic propensity for being a marketable drug or not. Both nodes and edges are equipped with categorical labels: node labels identify the atom type and edge labels identify the valence of the linkage.

MUTAG: The MUTAG dataset consists of 188 graphs corresponding to chemical compounds divided into two classes according to their respective mutagenic effect on a bacterium. Both nodes and edges are equipped with categorical labels: node labels identify the atom type and edge labels identify the bond type (single, double, triple or aromatic).

DD: The DD dataset (also known as D&D) contains 1178 protein structures. Each protein is represented by a graph, in which the nodes are amino-acids and two nodes are connected by an edge if they are less than 6Å apart. Nodes encode a categorical label (i.e., the amino-acid type), whereas edges are unlabeled. The prediction task is to classify the protein structures into enzymes and non-enzymes.

NCI1: Each graph is used as representation of a chemical compound: each vertex stands for an atom of the molecule, and edges between vertices represent bonds between atoms. This dataset is relative to anti-cancer screens, where the chemicals are assessed as positive or negative to cell lung cancer. Each vertex has a categorical label representing the corresponding atom type, whereas edges are unlabeled.

ENZYMES: Each graph represents a simplified protein tertiary structure, where nodes correspond to secondary structure elements and edges connect nodes if those are neighbors along the primary structure (i.e., the amino-acid sequence) with the constraint that every node is connected to its three nearest spatial neighbors. Nodes contain a categorical label stating the type of secondary structure (α-helix, β-sheet or turn) and a real-valued vector containing 18 physical and chemical measurements. Conversely, edges are unlabeled. The prediction task is used to classify the protein structures into one of the six Enzyme Commission classes [51].

Brief statistics about the datasets can be found in Table 1, along with the reference paper in which each dataset has been originally presented, to which we refer the interested reader for more information.

Table 1. Dataset statistics.

Dataset Name	# Graphs	Avg. # Nodes	Avg. # Edges	# Classes	Balanced	Domain	Reference
AIDS	2000	15.69	16.20	2	No	Chemoinformatics	[49]
Letter-L	2250	4.7	3.1	15	Yes	Computer Vision	[49]
Letter-M	2250	4.7	3.2	15	Yes	Computer Vision	[49]
Letter-H	2250	4.7	4.5	15	Yes	Computer Vision	[49]
GREC	1100	11.5	12.2	22	Yes	Electronics	[49,52]
Mutagenicity	4337	30.3	30.8	2	No	Chemoinformatics	[49,53]
MUTAG	188	17.93	19.79	2	No	Chemoinformatics	[54,55]
DD	1178	284.32	715.66	2	No	Bioinformatics	[56,57]
NCI1	4110	29.87	32.30	2	Yes	Chemoinformatics	[57,58]
ENZYMES	600	32.63	62.14	6	Yes	Bioinformatics	[59,60]

Each dataset has been split into three disjoint sets: training, validation and test. For the six IAM datasets, we used the very same training/validation/test splits provided in the repository, whereas for the four TUDatasets, we had to perform our own splits with the following ratio: training set (50%), validation set (25%) and test set (25%). The splitting procedure has been performed in a stratified manner in order to preserve labels' distribution across the three sets.

For Letter-L, Letter-M, Letter-H, GREC and AIDS, details about the dissimilarity measures on nodes and edges can be found in ([32] Appendix B). We anticipate that GREC is the only dataset with parametric dissimilarity measures on nodes and edges, i.e., for which $\Pi_{\text{edge}} \neq \emptyset$ and $\Pi_{\text{node}} \neq \emptyset$.

For MUTAG and Mutagenicity, since in both datasets nodes and edges are equipped with categorical labels, the dissimilarity between nodes and the dissimilarity between edges are set as the plain discrete distance between labels (i.e., their distance is 1 if they are not equal and 0 otherwise) ([61] Chapter 1).

For DD and NCI1, since in both datasets nodes are equipped with categorical labels and edges are unlabeled:

- The dissimilarity between nodes reads as the plain discrete distance between labels;
- The dissimilarity between edges reads as a constant value.

Finally, for ENZYMES:

- The dissimilarity between nodes reads as the (normalized) Euclidean distance between their respective 18-length attribute vectors if and only if the two nodes refer to the same secondary structure (i.e., both are α-helices, β-sheets or turns); otherwise, their dissimilarity is 1;
- The dissimilarity between edges (that are unlabeled) reads as a constant value.

5.2. Algorithmic Setup

The software has been fully implemented in Python with the support of the following third-party libraries: NetworkX [62] and LittleBallOfFur [63] for handling and processing graph data structures, Scikit-Learn [64] for machine learning routines and SciPy [65] for optimization routines.

Other algorithm parameters have been configured as follows:

- $\alpha = 0.9$ in the fitness function of the first genetic optimization (cf. Equation (8));
- $\beta = 0.9$ in the fitness function of the second genetic optimization (cf. Equation (12));
- Maximum number of 20 generations for both genetic optimizations;

- A total of 20 and 100 individuals in the population for the first and second genetic optimization, respectively;
- Maximum walk length $o = 5$;
- Candidate θ values are $\theta = \{0.5, 0.25, 0.125\}$;
- $Q \in [1, Q_{\max}]$, where $Q_{\max} = 500$ for non-stratified Granulators and $Q_{\max} = 500/p$ for stratified Granulators [41];
- The user-defined bucket size W has been chosen according to a given percentage of the maximum number of paths or cliques that can be drawn from the training graphs. The percentages are chosen according to the following rule: the larger the dataset, the higher the subsampling rate. For the sake of shorthand, we omit any sensitivity analysis on the behavior of the Granulators as a function of W. In this regard, we refer the interested reader to our previous works [41,46,66] The values for W for both clique-based and path-based Granulators are summarized in Table 2.

Table 2. Size of W for path-based and clique-based Granulators. Here, N_P and N_C refer to the number of paths of maximum order $o = 5$ and the total number of maximal cliques that can be drawn from the training set, respectively.

Dataset	Path-Based	Clique-Based
AIDS	$10\% \times N_P$ with $N_P = 27{,}589$	$40\% \times N_C$ with $N_C = 3994$
GREC	$10\% \times N_P$ with $N_P = 21{,}009$	$40\% \times N_C$ with $N_C = 3367$
Letter-L	$10\% \times N_P$ with $N_P = 7975$	$40\% \times N_C$ with $N_C = 2391$
Letter-M	$10\% \times N_P$ with $N_P = 8217$	$40\% \times N_C$ with $N_C = 2412$
Letter-H	$10\% \times N_P$ with $N_P = 12{,}063$	$40\% \times N_C$ with $N_C = 2500$
Mutagenicity	$10\% \times N_P$ with $N_P = 449{,}519$	$40\% \times N_C$ with $N_C = 49{,}294$
MUTAG	$10\% \times N_P$ with $N_P = 15{,}530$	$40\% \times N_C$ with $N_C = 1897$
DD	$1\% \times N_P$ with $N_P = 22{,}024{,}059$	$10\% \times N_C$ with $N_C = 168{,}781$
NCI1	$5\% \times N_P$ with $N_P = 521{,}560$	$20\% \times N_C$ with $N_C = 66{,}522$
ENZYMES	$10\% \times N_P$ with $N_P = 383{,}730$	$40\% \times N_C$ with $N_C = 9897$

5.3. Computational Results

In Figures 1 and 2 we show the accuracy on the test set and the number of resulting alphabet symbols, respectively, with Figures 3 and 4 showing their respective detailed breakdown. For the sake of completeness, in Figure 5, we show also the size of the alphabet before the second genetic optimization stage. Due to the stochastic nature of the model synthesis procedure, the results shown below have been obtained by averaging across 10 different runs of the algorithm. Each figure shows a heatmap whose color scale has been normalized by rows (i.e., independently for each dataset) and ranges from white (lower values) towards blue (higher values).

By considering Figure 1, it is possible to observe that (regardless of the topology) a stratified approach leads to generally better performance (i.e., higher accuracy on the test set). The breakdown of the shifts in accuracy is shown in Figure 3: for both path-based and clique-based granulators, the majority of the differences between stratified and non-stratified approaches are positive. For path-based extractors, the only two datasets for which the opposite is true are AIDS and ENZYMES. Conversely, for clique-based extractors,

ENZYMES emerges as the only dataset showing the same phenomenon. By looking at the magnitude of the accuracy boost, cliques seem to benefit more from a stratified granulation approach: indeed, the vast majority of the boosts in terms of accuracy are upper-bounded by 10%, whereas the majority of path-based granulators reach at most a 3% accuracy boost with respect to the non-stratified counterpart.

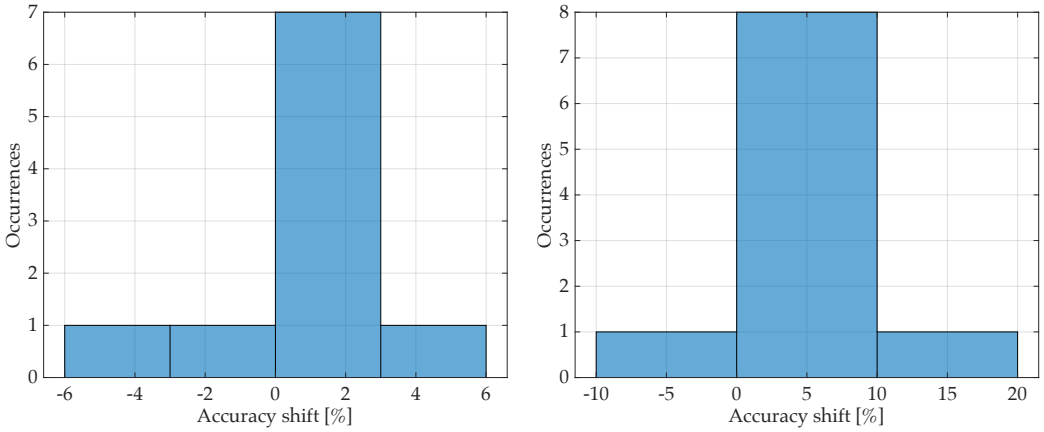
	Paths	Stratified Paths	Cliques	Stratified Cliques
Letter-L	96.8	97.2	96.31	97.51
Letter-M	91.11	92.09	58.36	77.56
Letter-H	88.31	90	74.4	77.96
AIDS	99.09	99.02	97.93	99.4
GREC	79.13	82.15	90.3	90.94
Mutagenicity	68.14	69.98	72.84	72.84
MUTAG	83.69	85.82	88.65	90.07
ENZYMES	30.44	27.33	34.89	33.78
NCI-1	73.02	73.41	66.86	67.61
DD	73.67	74.8	76.38	76.95

Figure 1. Accuracy on the test set (in percentage).

By looking at Figure 2, we can observe the number of symbols (i.e., the size of the alphabet) after the second genetic-driven feature selection stage. In principle, a clique-based approach is likely to yield a smaller alphabet: this is due to the fact that an n-vertex graph can have at most $\mathcal{O}(3^{n/3})$ cliques, whereas the number of paths can grow as $\mathcal{O}(n!)$ in the worst case, so the set of prospective information granules (even without any subsampling) is smaller. Conversely to the accuracy case, a stratified approach does not necessarily yield benefits. As can be seen in the breakdown (Figure 4), as paths are concerned, the stratification yields a smaller alphabet for 4 datasets out of 10. As cliques are concerned, the same phenomenon happens for 3 out of 10 datasets. This phenomenon is particularly evident for binary classification problems: a clear sign that building two dedicate alphabets (one for the positive class, one for the negative class) is excessive and just one alphabet would be a smarter choice.

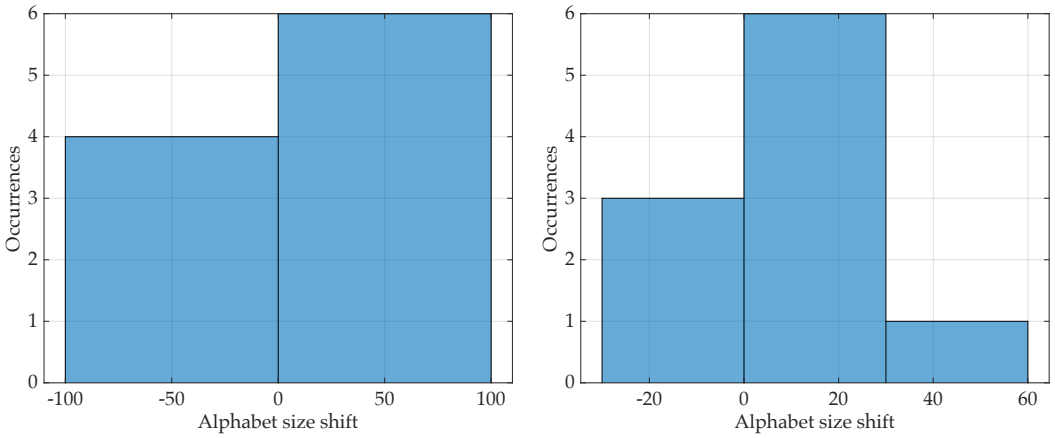
	Paths	Stratified Paths	Cliques	Stratified Cliques
Letter-L	21	16	17	21
Letter-M	45	55	103	101
Letter-H	80	81	127	98
AIDS	4	5	2	1
GREC	176	150	105	123
Mutagenicity	180	276	23	34
MUTAG	11	36	5	6
ENZYMES	116	99	122	130
NCI-1	230	134	12	21
DD	134	197	96	148

Figure 2. Size of the alphabet after feature selection.



(**a**) (**b**)

Figure 3. Distribution of the accuracy boosts for paths and cliques across the 10 datasets. A positive shift means that the stratified approach outperforms the non-stratified approach. (**a**) Paths. (**b**) Cliques.



(**a**) (**b**)

Figure 4. Distribution of the differences of the alphabet size for paths and cliques across the 10 datasets (after feature selection). A positive shift means that the stratified approach underperforms the non-stratified approach. (**a**) Paths. (**b**) Cliques.

In Table 3, we finally show a brief comparison against current approaches for graph classification. Competitors span a variety of techniques, including classifiers working on the top of GEDs [49,67], kernel methods [68–71] and several embedding techniques [68,72,73], including Granular Computing-based [32,74] and those based on neural networks and deep learning [75–79]. We can see that our method has comparable performances against current approaches in the graph classification literature. It is worth remarking that, aside from the mere numerical results, our approach has considerable higher perspectives to be deployed for analytical investigation by field experts thanks to its interpretability advantages, a common facet of Granular Computing-based systems [32,74], since the resulting set of automatically extracted granules of information can be analyzed by field experts in order to gather further insights on the problem at hand, paving the way for knowledge discovery.

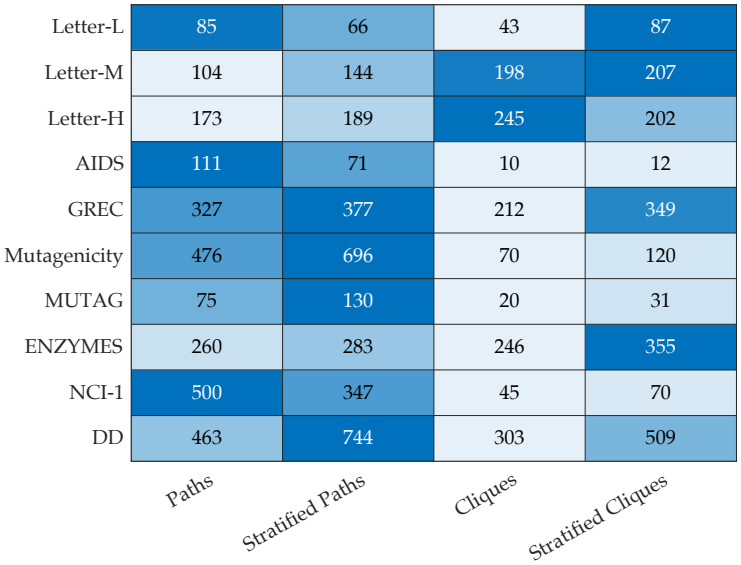


Figure 5. Size of the alphabet before feature selection.

Table 3. Comparison against state-of-the-art graph classification system in terms of classification accuracy.

Technique	AIDS	GREC	Letter-L	Letter-M	Letter-H	Mutagenicity	MUTAG	DD	NCI1	ENZYMES	Reference
Bipartite Graph Matching + *K*-NN	-	86.3	91.1	77.6	61.6	-	-	-	-	-	[67]
Lipschitz Embedding + SVM	98.3	96.8	99.3	95.9	92.5	74.9	-	-	-	-	[72]
Graph Edit Distance + *K*-NN	97.3	95.5	99.6	94	90	71.5	-	-	-	-	[49]
Graph of Words + *K*-NN	-	97.5	98.8	-	-	-	-	-	-	-	[73]
Graph of Words + kPCA + *K*-NN	-	97.1	97.6	-	-	-	-	-	-	-	[73]
Graph of Words + ICA + *K*-NN	-	58.9	82.8	-	-	-	-	-	-	-	[73]
Topological embedding	99.4	-	-	-	-	77.2	-	-	-	-	[68,80]
FMGE	99.0	-	-	-	-	76.5	-	-	-	-	[68,81]
Attribute Statistics	99.6	-	-	-	-	76.5	-	-	-	-	[68,82]
Hypergraph Embedding + SVM	99.3	-	-	-	-	77.0	84.6	-	72.7	43.1	[74]
ODD ST_+ kernel	82.06 *	-	-	-	-	-	-	-	84.97 *	-	[69]
ODD ST_+^{TANH} kernel	82.54 *	-	-	-	-	-	-	-	84.57 *	-	[69]
Laplacian kernel	92.6	-	-	-	-	70.2	-	-	-	-	[68,83]
Treelet kernel	99.1	-	-	-	-	77.1	-	-	-	-	[68,84]
Treelet kernel with MKL	99.7	-	-	-	-	77.6	-	-	-	-	[68,85]
WJK Hypergraph kernel + SVM	99.5 *	-	-	-	-	82 *	90.9 *	78.9 *	70.7 *	-	[70]
CGMM + linear SVM	84.16 *	-	-	-	-	-	91.18 *	-	-	-	[76]
G-L-Perceptron	-	70	95	64	70	-	-	-	-	-	[77]
G-M-Perceptron	-	75	98	87	81	-	-	-	-	-	[77]
C-1NN	-	-	96	93	84	-	-	-	-	-	[77]
C-M-1NN	-	-	98	81	71	-	-	-	-	-	[77]
EigenGCN-1	-	-	-	-	-	80.1	-	77.5	76.0	65.0	[75]
EigenGCN-2	-	-	-	-	-	78.9	-	77.0	76.7	64.5	[75]
EigenGCN-3	-	-	-	-	-	79.5	-	78.6	77.0	64.5	[75]
GCN with logical descriptors	-	96.93	96.64	85.27	79.91	-	-	-	-	-	[78]
MPNN	-	89.5	91.3	81.2	64.24	-	-	-	-	-	[79]
MPNN (no set2set)	-	92.98	94.8	86.1	75.7	-	-	-	-	-	[79]
Deep Graphlet Kernel	-	-	-	-	-	-	82.66 *	-	62.48 *	27.08 *	[71]
GRALG Paths	99.09	79.13	96.80	91.11	88.31	68.14	83.69	73.67	73.02	30.44	This work
GRALG Stratified Paths	99.02	82.15	97.20	92.09	90.00	69.98	85.82	74.80	73.41	27.33	This work
GRALG Cliques	97.93	90.3	96.31	58.36	74.4	72.84	88.65	76.38	66.86	34.89	This work
GRALG Stratified Cliques	99.22	92.74	97.24	80.22	77.82	72.7	90.07	76.95	67.61	33.78	This work

* Results refers to cross-validation rather than a separate test set.

6. Conclusions

In this work, we performed a two-fold investigation of clustering-based strategies for the automatic synthesis of information granules in the graph domain for solving (graph) classification problems on labeled graphs.

Our investigation jointly considers the subgraph topology in order to extract granules of information (i.e., cliques vs. paths extracted via a random walk) and the possibility of performing a class-aware, stratified clustering over the set of candidate information granules to build class-specific alphabets of symbols.

These strategies are tested within the GRALG framework, where the set of automatically extracted granules of information serve as pivotal subgraphs for building and embedding space. The soundness of the latter is addressed by a statistical classifier working on the embedding space. A two-stage optimization process takes care of tuning the hyper-parameters of the classification model and performing an additional feature selection to ensure the selection of optimal granules of information.

In order to address the behavior of the granulation strategies, we performed a two-fold comparison in terms of classification performance and number of granules of information. To this end, 10 different open-access datasets of labeled graphs pertaining to different application domains have been considered. Computational results show that, regardless of whether cliques or paths are employed, running a class-aware granulation is beneficial in terms of classification accuracy. However, it has the potential drawback of yielding a higher number of granules of information (i.e., higher dimensional embedding space).

Although in this paper we used the BSAS free-clustering algorithm as the core clustering algorithm for the Granulator module, in principle, any clustering algorithm suitable for dealing with structured data and suitable for being equipped with an ad hoc dissimilarity measure can be employed instead. In the GRALG framework, the Granulator process is repeated many times (i.e., at least as many times as the are individuals in the first optimization stage), and therefore, the computational complexity of the clustering algorithm plays an important role in order to ensure reasonable training times (i.e., the BSAS clustering algorithm is known to scale linearly with respect to the number of patterns). Future research endeavours may investigate the trade-off in terms of training times against granulation quality with different clustering algorithms.

Author Contributions: Conceptualization, A.M. and A.R.; methodology, A.M. and L.B.; software, A.M. and L.B.; validation, A.M. and L.B.; formal analysis, A.M. and L.B.; investigation, A.M.; resources, A.R.; data curation, A.M. and L.B.; writing—original draft preparation, A.M. and L.B.; writing—review and editing, A.M. and A.R.; supervision, A.M. and A.R.; project administration, A.M. and A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The six datasets taken from the IAM Repository can be downloaded from https://fki.tic.heia-fr.ch/databases/iam-graph-database (access date 26 April 2022). The four datasets taken from the TUDataset Repository can be downloaded from https://chrsmrrs.github.io/datasets (access date 26 April 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BSAS	Basic Sequential Algorithmic Scheme
GRALG	GRanular computing Approach for labeled Graphs
GED	Graph Edit Distance
K-NN	*K*-Nearest Neighbors
nBMF	node Best Match First

References

1. Bargiela, A.; Pedrycz, W. *Granular Computing: An Introduction*; Kluwer Academic Publishers: Boston, MA, USA, 2003.
2. Pedrycz, W.; Skowron, A.; Kreinovich, V. *Handbook of Granular Computing*; John Wiley & Sons: Hoboken, NJ, USA, 2008.
3. Zadeh, L.A. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets Syst.* **1997**, *90*, 111–127. [CrossRef]

4. Yager, R.; Filev, D. Operations for granular computing: Mixing words and numbers. In Proceedings of the 1998 IEEE International Conference on Fuzzy Systems Proceedings, IEEE World Congress on Computational Intelligence (Cat. No.98CH36228), Anchorage, AK, USA, 4–9 May 1998; Volume 1, pp. 123–128. [CrossRef]
5. Zadeh, L. Fuzzy logic = computing with words. *IEEE Trans. Fuzzy Syst.* **1996**, *4*, 103–111. [CrossRef]
6. Yao, Y. Perspectives of granular computing. In Proceedings of the 2005 IEEE International Conference on Granular Computing, Beijing, China, 25–27 July 2005; Volume 1, pp. 85–90.
7. Pedrycz, A.; Hirota, K.; Pedrycz, W.; Dong, F. Granular representation and granular computing with fuzzy sets. *Fuzzy Sets Syst.* **2012**, *203*, 17–32. [CrossRef]
8. Dubois, D.; Prade, H. Bridging gaps between several forms of granular computing. *Granul. Comput.* **2016**, *1*, 115–126. [CrossRef]
9. Pawlak, Z. Rough sets. *Int. J. Comput. Inf. Sci.* **1982**, *11*, 341–356. [CrossRef]
10. Zhang, Q.; Zhang, Q.; Wang, G. The Uncertainty of Probabilistic Rough Sets in Multi-Granulation Spaces. *Int. J. Approx. Reason.* **2016**, *77*, 38–54. [CrossRef]
11. Pedrycz, W. Shadowed sets: Representing and processing fuzzy sets. *IEEE Trans. Syst. Man Cybern. Part B* **1998**, *28*, 103–109. [CrossRef]
12. Kreinovich, V. Interval Computation as an Important Part of Granular Computing: An Introduction. In *Handbook of Granular Computing*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2008; pp. 1–31. [CrossRef]
13. Pedrycz, W. Proximity-based clustering: A search for structural consistency in data with semantic blocks of features. *IEEE Trans. Fuzzy Syst.* **2013**, *21*, 978–982. [CrossRef]
14. Ding, S.; Du, M.; Zhu, H. Survey on granularity clustering. *Cogn. Neurodyn.* **2015**, *9*, 561–572. [CrossRef]
15. Peters, G.; Weber, R. DCC: A framework for dynamic granular clustering. *Granul. Comput.* **2016**, *1*, 1–11. [CrossRef]
16. Jain, A.K.; Murty, M.N.; Flynn, P.J. Data Clustering: A Review. *ACM Comput. Surv.* **1999**, *31*, 264–323. [CrossRef]
17. Hadamard, J. Sur les problèmes aux dérivés partielles et leur signification physique. *Princet. Univ. Bull.* **1902**, *13*, 49–52.
18. von Luxburg, U.; Williamson, R.C.; Guyon, I. Clustering: Science or Art? In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*; Guyon, I., Dror, G., Lemaire, V., Taylor, G., Silver, D., Eds.; PMLR: Bellevue, WA, USA, 2012; Volume 27, pp. 65–79.
19. Bouveyron, C.; Hammer, B.; Villmann, T. Recent developments in clustering algorithms. In *ESANN 2012*; ESANN: Bruges, Belgium, 2012; pp. 447–458.
20. Xu, R.; Wunsch, D. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678. [CrossRef] [PubMed]
21. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, 21 June–18 July 1967; Volume 1, pp. 281–297.
22. Kaufman, L.; Rousseeuw, P. Clustering by means of Medoids. In *Statistical Data Analysis Based on the L1 Norm and Related Methods*; Dodge, Y., Ed.; Elsevier: Amsterdam, The Netherlands, 1987; pp. 405–416.
23. Huang, Z. Clustering large data sets with mixed numeric and categorical values. In Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, Singapore, 23–24 February 1997; pp. 21–34.
24. Bezdek, J.C. *Pattern Recognition with Fuzzy Objective Function Algorithms*; Kluwer Academic Publishers: Norwell, MA, USA, 1981.
25. Livi, L.; Del Vescovo, G.; Rizzi, A. Graph Recognition by Seriation and Frequent Substructures Mining. In Proceedings of the ICPRAM 2012—1st International Conference on Pattern Recognition Applications and Methods, Algarve, Portugal, 6–8 February 2012; Volume 1, pp. 186–191.
26. Rizzi, A.; Del Vescovo, G. Automatic Image Classification by a Granular Computing Approach. In Proceedings of the 2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing, Maynooth, Ireland, 6–8 September 2006; pp. 33–38. [CrossRef]
27. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2009.
28. Pedrycz, W. *Knowledge-Based Clustering: From Data to Information Granules*; John Wiley & Sons: Hoboken, NJ, USA, 2005.
29. Theodoridis, S.; Koutroumbas, K. *Pattern Recognition*, 4th ed.; Academic Press: Cambridge, MA, USA, 2008.
30. Sanfeliu, A.; Fu, K.S. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst. Man Cybern.* **1983**, *SMC-13*, 353–362. [CrossRef]
31. Gao, X.; Xiao, B.; Tao, D.; Li, X. A survey of graph edit distance. *Pattern Anal. Appl.* **2010**, *13*, 113–129. [CrossRef]
32. Martino, A.; Rizzi, A. An Enhanced Filtering-Based Information Granulation Procedure for Graph Embedding and Classification. *IEEE Access* **2021**, *9*, 15426–15440. [CrossRef]
33. Baldini, L.; Martino, A.; Rizzi, A. Relaxed Dissimilarity-based Symbolic Histogram Variants for Granular Graph Embedding. In Proceedings of the 13th International Joint Conference on Computational Intelligence—NCTA, Singapore, 10–13 November 2021; pp. 221–235. [CrossRef]
34. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
35. Lovász, L. Random walks on graphs: A survey. *Combinatorics* **1993**, *2*, 1–46.
36. Göbel, F.; Jagers, A.A. Random walks on graphs. *Stoch. Process. Their Appl.* **1974**, *2*, 311–336. [CrossRef]
37. Pedrycz, W.; Homenda, W. Building the fundamentals of granular computing: A principle of justifiable granularity. *Appl. Soft Comput.* **2013**, *13*, 4209–4218. [CrossRef]

38. Wang, G.; Yang, J.; Xu, J. Granular computing: From granularity optimization to multi-granularity joint problem solving. *Granul. Comput.* **2017**, *2*, 105–120. [CrossRef]
39. Yao, Y.Y. The rise of granular computing. *J. Chongqing Univ. Posts Telecommun.* **2008**, *20*, 299–308.
40. Yao, Y.; Zhao, L. A measurement theory view on the granularity of partitions. *Inf. Sci.* **2012**, *213*, 1–13. [CrossRef]
41. Baldini, L.; Martino, A.; Rizzi, A. Towards a Class-Aware Information Granulation for Graph Embedding and Classification. In *Computational Intelligence, Proceedings of the 11th International Joint Conference, IJCCI 2019, Vienna, Austria, 17–19 September 2019, Revised Selected Papers*; Merelo, J.J., Garibaldi, J., Linares-Barranco, A., Warwick, K., Madani, K., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 263–290. [CrossRef]
42. Tichy, N. An Analysis of Clique Formation and Structure in Organizations. *Adm. Sci. Q.* **1973**, *18*, 194–208. [CrossRef]
43. Luce, R.D.; Perry, A.D. A method of matrix analysis of group structure. *Psychometrika* **1949**, *14*, 95–116. [CrossRef] [PubMed]
44. Bron, C.; Kerbosch, J. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* **1973**, *16*, 575–577. [CrossRef]
45. Moon, J.W.; Moser, L. On cliques in graphs. *Isr. J. Math.* **1965**, *3*, 23–28. [CrossRef]
46. Baldini, L.; Martino, A.; Rizzi, A. Exploiting Cliques for Granular Computing-based Graph Classification. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–9. [CrossRef]
47. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
48. Grandini, M.; Bagli, E.; Visani, G. Metrics for Multi-Class Classification: An Overview. *arXiv* **2020**, arXiv:2008.05756.
49. Riesen, K.; Bunke, H. IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In *Structural, Syntactic, and Statistical Pattern Recognition*; da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 287–297.
50. Morris, C.; Kriege, N.M.; Bause, F.; Kersting, K.; Mutzel, P.; Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020). *arXiv* **2020**, arXiv:2007.08663.
51. Webb, E.C. *Enzyme Nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes*, 6th ed.; Academic Press: Cambridge, MA, USA, 1992.
52. Dosch, P.; Valveny, E. Report on the Second Symbol Recognition Contest. In *Graphics Recognition. Ten Years Review and Future Perspectives*; Liu, W., Lladós, J., Eds.; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2006; pp. 381–397.
53. Kazius, J.; McGuire, R.; Bursi, R. Derivation and Validation of Toxicophores for Mutagenicity Prediction. *J. Med. Chem.* **2005**, *48*, 312–320. [CrossRef] [PubMed]
54. Debnath, A.K.; Lopez de Compadre, R.L.; Debnath, G.; Shusterman, A.J.; Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *J. Med. Chem.* **1991**, *34*, 786–797. [CrossRef] [PubMed]
55. Kriege, N.; Mutzel, P. Subgraph Matching Kernels for Attributed Graphs. In Proceedings of the 29th International Coference on International Conference on Machine Learning, Edinburgh, UK, 27 June–3 July 2012; Omnipress: Madison, WI, USA, 2012; Volume ICML'12, pp. 291–298.
56. Dobson, P.D.; Doig, A.J. Distinguishing Enzyme Structures from Non-enzymes without Alignments. *J. Mol. Biol.* **2003**, *330*, 771–783. [CrossRef]
57. Shervashidze, N.; Schweitzer, P.; van Leeuwen, E.J.; Mehlhorn, K.; Borgwardt, K.M. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.* **2011**, *12*, 2539–2561.
58. Wale, N.; Karypis, G. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. In Proceedings of the Sixth International Conference on Data Mining (ICDM'06), Hong Kong, China, 18–22 December 2006; pp. 678–689. [CrossRef]
59. Schomburg, I.; Chang, A.; Ebeling, C.; Gremse, M.; Heldt, C.; Huhn, G.; Schomburg, D. BRENDA, the enzyme database: Updates and major new developments. *Nucleic Acids Res.* **2004**, *32*, D431–D433. [CrossRef] [PubMed]
60. Borgwardt, K.M.; Ong, C.S.; Schönauer, S.; Vishwanathan, S.V.N.; Smola, A.J.; Kriegel, H.P. Protein function prediction via graph kernels. *Bioinformatics* **2005**, *21*, i47–i56. [CrossRef]
61. Deza, M.M.; Deza, E. *Encyclopedia of Distances*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2009.
62. Hagberg, A.A.; Schult, D.A.; Swart, P.J. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*; Varoquaux, G., Vaught, T., Millman, J., Eds.; Los Alamos National Lab.: Pasadena, CA, USA, 2008; pp. 11–15.
63. Rozemberczki, B.; Kiss, O.; Sarkar, R. Little Ball of Fur: A Python Library for Graph Sampling. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), Online, 19–23 October 2020; pp. 3133–3140.
64. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
65. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef]
66. Baldini, L.; Martino, A.; Rizzi, A. Stochastic Information Granules Extraction for Graph Embedding and Classification. In Proceedings of the 11th International Joint Conference on Computational Intelligence, NCTA, (IJCCI 2019), Hendaye, France, 4–6 September 2019; Volume 1, pp. 391–402. [CrossRef]

67. Riesen, K.; Bunke, H. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vis. Comput.* **2009**, *27*, 950–959. [CrossRef]
68. Conte, D.; Ramel, J.Y.; Sidère, N.; Luqman, M.M.; Gaüzère, B.; Gibert, J.; Brun, L.; Vento, M. A Comparison of Explicit and Implicit Graph Embedding Methods for Pattern Recognition. In *Graph-Based Representations in Pattern Recognition*; Kropatsch, W.G., Artner, N.M., Haxhimusa, Y., Jiang, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 81–90. [CrossRef]
69. Da San Martino, G.; Navarin, N.; Sperduti, A. Ordered Decompositional DAG kernels enhancements. *Neurocomputing* **2016**, *192*, 92–103. [CrossRef]
70. Martino, A.; Rizzi, A. (Hyper)graph Kernels over Simplicial Complexes. *Entropy* **2020**, *22*, 1155. [CrossRef] [PubMed]
71. Yanardag, P.; Vishwanathan, S. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2015; Volume KDD '15, pp. 1365–1374. [CrossRef]
72. Riesen, K.; Bunke, H. Graph Classification by Means of Lipschitz Embedding. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* **2009**, *39*, 1472–1483. [CrossRef] [PubMed]
73. Gibert, J.; Valveny, E.; Bunke, H. Dimensionality Reduction for Graph of Words Embedding. In *Graph-Based Representations in Pattern Recognition*; Jiang, X., Ferrer, M., Torsello, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 22–31.
74. Martino, A.; Giuliani, A.; Rizzi, A. (Hyper)Graph Embedding and Classification via Simplicial Complexes. *Algorithms* **2019**, *12*, 223. [CrossRef]
75. Ma, Y.; Wang, S.; Aggarwal, C.C.; Tang, J. Graph Convolutional Networks with EigenPooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; Association for Computing Machinery: New York, NY, USA, 2019; Volume KDD '19, pp. 723–731. [CrossRef]
76. Bacciu, D.; Errica, F.; Micheli, A. Contextual graph markov model: A deep and generative approach to graph processing. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 10–15 July 2018; Volume 1, pp. 495–504.
77. Martineau, M.; Raveaux, R.; Conte, D.; Venturini, G. Learning error-correcting graph matching with a multiclass neural network. *Pattern Recognit. Lett.* **2020**, *134*, 68–76. [CrossRef]
78. Kajla, N.I.; Missen, M.M.S.; Luqman, M.M.; Coustaty, M. Graph Neural Networks Using Local Descriptions in Attributed Graphs: An Application to Symbol Recognition and Hand Written Character Recognition. *IEEE Access* **2021**, *9*, 99103–99111. [CrossRef]
79. Riba, P.; Dutta, A.; Lladós, J.; Fornés, A. Graph-Based Deep Learning for Graphics Classification. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 13–15 November 2017; Volume 2, pp. 29–30. [CrossRef]
80. Sidère, N.; Héroux, P.; Ramel, J.Y. Vector Representation of Graphs: Application to the Classification of Symbols and Letters. In Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, 26–29 July 2009; pp. 681–685. [CrossRef]
81. Luqman, M.M.; Ramel, J.Y.; Lladós, J.; Brouard, T. Fuzzy multilevel graph embedding. *Pattern Recognit.* **2013**, *46*, 551–565. [CrossRef]
82. Gibert, J.; Valveny, E.; Bunke, H. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognit.* **2012**, *45*, 3072–3083. [CrossRef]
83. Brun, L.; Conte, D.; Foggia, P.; Vento, M. A Graph-Kernel Method for Re-identification. In *Image Analysis and Recognition*; Kamel, M., Campilho, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 173–182.
84. Gaüzère, B.; Brun, L.; Villemin, D. Two New Graphs Kernels in Chemoinformatics. *Pattern Recogn. Lett.* **2012**, *33*, 2038–2047. [CrossRef]
85. Gaüzère, B.; Brun, L.; Villemin, D.; Brun, M. Graph kernels based on relevant patterns and cycle information for chemoinformatics. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012; pp. 1775–1778.