Andrea Torsello
Luca Rossi
Luca Cosmo
Giorgia Minello (Eds.)

LNCS 15444

Structural, Syntactic, and Statistical Pattern Recognition

Joint IAPR International Workshops S+SSPR 2024
Venice, Italy, September 9–10, 2024
Revised Selected Papers

Springer

Lecture Notes in Computer Science **15444**

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

Andrea Torsello · Luca Rossi · Luca Cosmo ·
Giorgia Minello
Editors

Structural, Syntactic, and Statistical Pattern Recognition

Joint IAPR International Workshops, S+SSPR 2024
Venice, Italy, September 9–10, 2024
Revised Selected Papers

Springer

Editors
Andrea Torsello
Ca' Foscari University of Venice
Venice, Italy

Luca Rossi
The Hong Kong Polytechnic University
Kowloon, Hong Kong

Luca Cosmo
Ca' Foscari University of Venice
Venice, Italy

Giorgia Minello
Ca' Foscari University of Venice
Venice, Italy

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-80506-6 ISBN 978-3-031-80507-3 (eBook)
https://doi.org/10.1007/978-3-031-80507-3

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at the joint IAPR International Workshops on Structural and Syntactic Pattern Recognition (SSPR 2024) and Statistical Techniques in Pattern Recognition (SPR 2024). S+SSPR is a joint event organized by Technical Committee 1 (Statistical Pattern Recognition Techniques) and Technical Committee 2 (Structural and Syntactical Pattern Recognition) of the International Association of Pattern Recognition (IAPR). In a break with the tradition of co-locating S+SSPR with ICPR, S+SSPR 2024 was held spatially and temporally separate from ICPR 2024. In fact, S+SSPR 2024 was held in Venice, Italy, in the frescoed "Aula Magna Silvio Trentin" in the Venetian nobiliar palace Ca' Dolfin. The event had a rich two-day format spanning September 9–10, 2024.

The reviewing process was single blind, where each submission was reviewed by at least two and usually three Program Committee members. We received 27 submissions from 15 countries across 5 continents, from which 19 were eventually selected for presentation at the workshop.

The accepted papers cover the major topics of current interest in pattern recognition, including classification and clustering, deep learning, structural matching and graph-theoretic methods, and multimedia analysis and understanding.

During the conference we were delighted to host two IAPR keynote speakers: Emanuele Rodolà from Università la Sapienza, Rome, Italy, presenting a talk titled "Unlocking Neural Composition", and the 2024 IAPR TC1 Pierre Devijver Award winner Thomas G. Dietterich from Oregon State University, Corvallis, OR, USA, giving a presentation titled "Integrating machine learning into safety-critical systems".

We would like to thank all the Program Committee members for their help in the review process and express our appreciation to Springer for publishing this volume. More information about the workshops and organization can be found on the website: https://iapr.org/ssspr2024/

October 2024

Andrea Torsello
Luca Rossi
Luca Cosmo
Giorgia Minello

Organization

General Chairs

Andrea Torsello	Ca' Foscari University of Venice, Italy
Luca Rossi	The Hong Kong Polytechnic University, Hong Kong

Local Chairs

Filippo Bergamasco	Ca' Foscari University of Venice, Italy
Luca Cosmo	Ca' Foscari University of Venice, Italy
Giorgia Minello	Ca' Foscari University of Venice, Italy
Mara Pistellato	Ca' Foscari University of Venice, Italy

SPR Chair

Ambra Demontis	University of Cagliari, Italy

SSPR Chair

Xiao Bai	Beihang University, China

Program Committee

Adam Krzyzak	Concordia University, Canada
Alessandro Bicciato	Ca' Foscari University of Venice, Italy
Ambra Demontis	University of Cagliari, Italy
Andrea Torsello	Ca' Foscari University of Venice, Italy
Benoit Gaüzère	Normandie Université, INSA de Rouen, LITIS, France
Carlo Sansone	University of Naples Federico II, Italy
Chenyu Liu	The Hong Kong Polytechnic University, Hong Kong
Emilian Postolache	Ca' Foscari University of Venice, Italy

Ethem Alpaydin	Özyeğin University, Turkey
Filippo Bergamasco	Ca' Foscari University of Venice, Italy
Francesc J. Ferri	University of Valencia, Spain
Francesc Serratosa	Universitat Rovira i Virgili, Spain
Francisco Escolano	University of Alicante, Spain
Frank-Michael Schleif	University of Applied Sciences Würzburg-Schweinfurt, Germany
Gady Agam	Illinois Institute of Technology, USA
Giorgia Minello	Ca' Foscari University of Venice, Italy
Giorgio Fumera	University of Cagliari, Italy
Giorgio Piras	University of Cagliari, Italy
Henry Senior	Queen Mary University of London, UK
Jing-Hao Xue	University College London, UK
Juan Humberto Sossa Azuela	CIC-IPN, Mexico
Laurent Heutte	Université de Rouen Normandie, France
Lichi Zhang	Shanghai Jiao Tong University, China
Lingfeng Zhang	The Hong Kong Polytechnic University, Hong Kong
Luca Cosmo	Ca' Foscari University of Venice, Italy
Luca Rossi	The Hong Kong Polytechnic University, Hong Kong
Manuel Curado	University of Alicante, Spain
Manuele Bicego	University of Verona, Italy
Mara Pistellato	Ca' Foscari University of Venice, Italy
Mauricio Orozco-Alzate	Universidad Nacional de Colombia, Colombia
Nikolaos Nikolaou	University College London, UK
Nikunj Oza	NASA, USA
Punam Kumar Saha	University of Iowa, USA
Richard Wilson	University of York, UK
Salvatore Tabbone	Université de Lorraine, France
Silvia Biasotti	CNR - IMATI, Italy
Simone Marinai	University of Florence, Italy
Srishti Gupta	University of Cagliari, Italy
Walter G. Kropatsch	TU Wien, Austria
Xiao Bai	Beihang University, China
Xiaohan Yu	Macquarie University, Australia
Xiaoyi Jiang	University of Münster, Germany

Sponsors


VENETIARVM VNIVERSITAS
IN DOMO FOSCARI
dAIS
Department of
Environmental Science,
Informatics and Statistics


THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學


IAPR


IAPR
TC1


IAPR
TC2

Contents

A Differentiable Approximation of the Graph Edit Distance

Julia Wallnig[1(✉)], Luc Brun[2], Benoit Gaüzère[3], Sébastien Bougleux[2], Florian Yger[4], and David B. Blumenthal[1]

[1] Biomedical Network Science Lab, Department Artificial Intelligence in Biomedical Engineering, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
{julia.wallnig,david.b.blumenthal}@fau.de
[2] Normandie Univ, ENSICAEN, CNRS, UNICAEN, GREYC, Caen, France
luc.brun@ensicaen.fr, sebastien.bougleux@unicaen.fr
[3] Normandie Univ, INSA Rouen Normandie, LITIS, Rouen, France
benoit.gauzere@insa-rouen.fr
[4] PSL-Université Paris-Dauphine, CNRS, LAMSADE, Paris, France
florian.yger@lamsade.dauphine.fr

Abstract. To determine the similarity of labeled graphs, the graph edit distance (GED) is widely used due to its metric properties on the graph space and its interpretability. It is defined as the minimal cost of a sequence of edit operations transforming one graph into another one, with the cost of each edit operation being a parameter of the distance. Although calculating GED is NP-hard, various heuristics exist which, in practice, typically yield tight upper or lower bounds. Since determining appropriate edit operation costs for a given dataset or application can be challenging, it is attractive to learn these costs from the data, e.g., using metric learning architectures. However, for this approach to be feasible, a differentiable algorithm to approximate the GED is required. In this work, we present such an algorithm and show via an empirical evaluation on three datasets that the obtained distances closely match the distances computed by a state-of-the-art combinatorial GED heuristic.

Keywords: Graph Edit Distance · Sinkhorn Algorithm · Differentiable Solver

1 Introduction

The *graph edit distance* (GED) [10] is a classical approach to quantify the dissimilarity of two labeled graphs G and H. It involves transforming a graph G into a graph isomorphic to H through a sequence of six elementary edit operations (node or edge insertion, deletion, or substitution), which all come with associated *edit costs*. A sequence of edit operations transforming G into H is called *edit path*, and its cost is defined as the sum of the individual operation costs. The GED from G to H is defined as the minimal cost of an edit path from

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 1–10, 2025.
https://doi.org/10.1007/978-3-031-80507-3_1

G to H. Computing GED is NP-hard [19], but various well-performing heuristics exist which provide upper or lower bounds [2].

In most GED heuristics, the edit operation costs are treated as parameters to be provided by the user. Since defining sensible costs for a given dataset can be challenging, it has been suggested to learn the edit costs from the data [11,14]. However, in existing approaches, a ground-truth mapping from the nodes of G to the nodes of H is required which is often not available. Moreover, even if such a mapping exists, it may not be unique, and *a priori* fixing one mapping may induce a bias in the learning process. In view of recent advances in the field of metric learning [18], it is hence attractive to learn the edit costs in an end-to-end fashion. For this, however, a differentiable operator to (approximately) compute GED is required which currently does not exist. Here, we close this gap.

Our differentiable GED operator relies on two building blocks. The first one is the widely used GED heuristic BRANCH [15], which computes both upper and lower bounds for GED [3]. BRANCH transforms the problem of computing GED to a *linear sum assignment problem with error-correction* (LSAPE), which is then solved using variants of the Hungarian algorithm [7,8]. The other building block is a recent work [9] which shows that the (differentiable) Sinkhorn algorithm [16] can be adapted such that it yields a differentiable approximation of LSAPE. We combine these building blocks into a differentiable operator to compute GED by replacing the Hungarian algorithm in BRANCH with the adapted Sinkhorn algorithm. Tests on three datasets show that our differentiable GED approximator achieves equally good results as BRANCH. Our results hence pave the way for incorporating GED into differentiable metric learning frameworks, thereby facilitating future advances in end-to-end learning of the edit costs.

2 Preliminaries

Basic Notations. We consider undirected, possibly labeled graphs $G = (V^G, E^G)$ and $H = (V^H, E^H)$, set $n{:=}|V^G|$ and $m{:=}|V^H|$, and use the notations $[n]{:=}\{1,\dots,n\}$ and $[m]{:=}\{1,\dots,m\}$ to denote the index sets for the nodes of G and H, respectively. In addition, for any matrix X of size $n \times m$, $x_{i,j}$ with $i \in [n]$ and $j \in [m]$ will always denote the entries of the matrix.

Graph Edit Distance and Node Maps. While GED is classically defined in terms of edit paths as explained above, most algorithmic approaches rely on an alternative definition in terms of node maps. A *node map* $\pi \subseteq (V^G \cup \{\epsilon\}) \times (V^H \cup \{\epsilon\})$ specifies for all nodes and, derived from this, for all edges if they are substituted, inserted, or deleted. Applying π's induced edit operations yields an *induced edit path* P_π, with $c(\pi)$ denoting the sum of the costs of all its edit operations. Under reasonable constraints that hold in almost all applications, minimizing $c(\pi)$ over all node maps π from G to H yields the GED between these graphs [4,6].

The Linear Sum Assignment Problem with Error Correction. LSAPE is an extension of the well-known bipartite matching a.k.a. linear sum assignment

problem (LSAP), where two additional dummy nodes $n+1$ and $m+1$ are inserted to model insertions and deletions. Given a cost matrix C of size $(n+1)\times(m+1)$ with $c_{n+1,m+1}=0$, LSAPE asks to solve the discrete minimization problem

$$\min_X \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} c_{i,j}x_{i,j} =: C(X), \tag{1}$$

where $X \in \{0,1\}^{(n+1)\times(m+1)}$ is an *error-correcting permutation matrix* (a.k.a. ϵ-permutation matrix), i.e., a binary matrix with $\sum_{i=1}^{n+1} x_{i,j}=1$ for all $j\in[m]$, and $\sum_{j=1}^{m+1} x_{i,j}=1$ for all $i\in[n]$ [7] (note that there are no constraints on $i=n+1$ and $j=m+1$). Each ϵ-permutation matrix X induces an *error-correcting matching* (also called ϵ-matching)

$$\pi_X := \{(i,j)\in[n+1]\times[m+1] \mid x_{i,j}=1\}, \tag{2}$$

and we use the notation $C(\pi_X):=C(X)$ to denote the matching cost of π_X. The Hungarian algorithm [12] is a widely used method for optimally solving LSAP and can also be extended to optimally solve LSAPE [7,8].

The Sinkhorn Algorithm. The Sinkhorn-Knopp theorem states that, for each positive matrix S of size $n\times n$, there are positive diagonal matrices D_1 and D_2 such that $X:=D_1SD_2\in\mathbb{R}^{n\times n}$ is bistochastic (i.e., all rows and columns of X sum up to 1). The Sinkhorn algorithm produces a series of matrices that converge to X via alternate scaling of S's rows and columns to unit sum. If applied on an exponentiated cost matrix

$$S := \exp(-\varepsilon^{-1}C), \tag{3}$$

the Sinkhorn algorithm converges to the optimal solution of the *optimal transport problem with entropic regularization* ($\varepsilon>0$ is the regularization constant) [13]. This problem, in turn, can be seen as a continuous relaxation of LSAP.

While the classical Sinkhorn algorithm is applicable only to matrices S of size $n\times n$, it has recently been shown that that it can be generalized to positive rectangular matrices of size $(n+1)\times(m+1)$ [9] and then yields an *ϵ-bistochastic matrix* $X\in\mathbb{R}^{n+1\times m+1}$, where the firsts n row sums, the first m column sums, and the entry $x_{n+1,m+1}$ equal 1. Since ϵ-bistochastic matrices are continuous relaxations of ϵ-matchings, running the adapted Sinkhorn algorithm on an exponentiated cost matrix yields a differentiable approximation of LSAPE [9].

Heuristics for Graph Edit Distance Computation Based on the Linear Sum Assignment with Error Correction. Various GED heuristics exist that are based on LSAPE [2]. These algorithms decompose the input graphs G and H into local structures rooted at their nodes and then construct an instance C of LSAPE, where the last row $C_{n+1,\bullet}$ and the last column $C_{\bullet,m+1}$ contain, respectively, local structure insertion and deletion costs and the remaining cells $C_{i,j}$ contain local structure substitution costs. C is then solved optimally, e.g., with a variant of the Hungarian algorithm, which yields an ϵ-matching $\pi_X\subseteq[n+1]\times[m+1]$.

Since π can be interpreted as a node map from V^G to V^H, its induced edit cost $c(\pi_X)$ is an upper bound for GED. Moreover, for some LSAPE-based GED heuristics, the matching cost $C(\pi_X)$ is a lower bound for GED. One such approach is the widely used algorithm BRANCH [15]. In this algorithm, the local structures are defined as branches (i.e., root nodes together with their adjacent edges) and the LSAPE instance C is defined in terms of the elementary node and edge edit costs required to transform a branch rooted at node $u \in V^G$ into a branch rooted at node $v \in V^H$ (see [2,3,15] for details). For this article, we tested our differentiable GED approximator with the LSAPE instance C as constructed by BRANCH but the underlying approach is compatible with any LSAPE-based GED heuristic. Moreover, for our algorithm to be differentiable, we assume that C is provided as input. This assumption can be lifted by using the Sinkhorn algorithm not only as an approximate solver for C but also for the approximate computation of the branch edit costs during the construction of C.

3 A Differentiable Graph Edit Distance Approximator

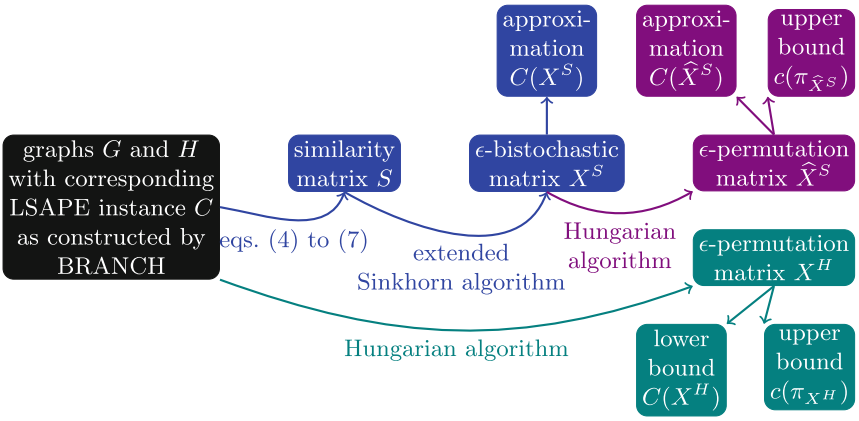


Fig. 1. Overview of our differentiable GED approximator (blue) with discrete post-processing (violet) in comparison to BRANCH (turquoise). (Color figure online)

Figure 1 provides an overview of our continuous GED approximator. Given two graphs G and H and a corresponding LSAPE instance C as constructed by BRANCH, we start by transforming C into a similarity matrix S as follows: We first compute a transformed cost matrix C' by setting

$$c'_{i,j} := \begin{cases} c_{i,j} - \min\{c_{i,j'} \mid j' \in [m+1]\} & i \in [n] \\ c_{i,j} & i = n+1 \end{cases} \tag{4}$$

for all $(i,j) \in [n+1] \times [m+1]$, and further transform C' into C'' by defining

$$c''_{i,j} := \begin{cases} c'_{i,j} - \min\{c'_{i',j} \mid i' \in [n+1]\} & j \in [m] \\ c'_{i,j} & j = m+1 \end{cases} \tag{5}$$

for all $(i,j) \in [n+1] \times [m+1]$. Let us note that Eqs. (4) and (5) correspond to the initialization steps of the LSAPE algorithm [7,8]. The matrices C'' and C correspond to equivalent LSAPE problems. Within our context, these simplifications of the cost matrix ensure that the minimum of each row and each column is set to 0. Using the construction scheme described below (Eq. (7)), the corresponding similarity values are set to 1 independently of the entropic regularization.

Next, we compute a scaling factor

$$T := \min\left\{\alpha, \beta \cdot \log(\alpha) \cdot \max(C'')^{-1}\right\}, \tag{6}$$

where $\alpha, \beta > 0$ are scaling hyper-parameters. The constant T can be interpreted as the inverse of the regularization constant ε in Eq. (3). Equipped with C'' and T, we then compute a similarity matrix as follows:

$$S := \exp(-T \cdot C'') \tag{7}$$

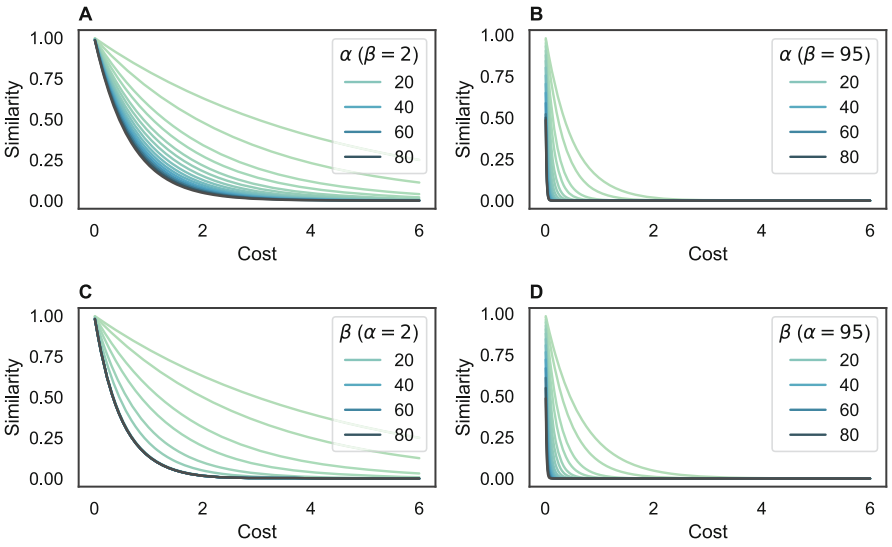


Fig. 2. Shape of the cost-to-similarity transformation function for different values of the parameters α and β. (A, B) Shape for varying α and fixed values of β. (C, D) Shape for varying β and fixed values of α.

Let us note that the division by $\max(C'')$ allows to normalize all cost values between 0 and 1. The resulting cost-to-similarity function in dependence on β

and α is visualized in Fig. 2: For small values of α and β, the function has a near-linear behaviour. For larger β and α, it assumes the shape of a step function.

After constructing S, we feed it into the extended Sinkhorn algorithm proposed in [9]. This yields an ϵ-bistochastic matrix X^S, whose matching cost $C(X^S)$ (see Eq. (1)) constitutes our differentiably computable approximation of the GED from G to H. $C(X^S)$ is the analogue to the matching cost $C(X^H)$ of the ϵ-permutation matrix X^H computed by BRANCH, but—unlike $C(X^H)$—is not guaranteed to constitute a lower bound for GED. Moreover, X^S is in general not discrete and hence does not constitute a node map from G to H.

To enable comparison also w.r.t. BRANCH's upper bound $c(\pi_{X^H})$, we further project X^S to an ϵ-permutation matrix $\widehat{X}^S$, using the Hungarian algorithm (violet boxes and arrows in Fig. 1). The node map $\pi_{\widehat{X}^S}$ induced by $\widehat{X}^S$ then yields an upper bound $c(\pi_{\widehat{X}^S})$ for GED, and the matching cost $C(\widehat{X}^S)$ gives us another approximation for GED. Note that since the computation of $\widehat{X}^S$ relies on the non-differentiable Hungarian algorithm, neither $c(\pi_{\widehat{X}^S})$ nor $C(\widehat{X}^S)$ should be viewed as direct output of our differentiable GED approximator.

4 Empirical Evaluation

Datasets and Edit Costs. We used the datasets Acyclic and MAO from GREYC's Chemistry Dataset (https://brunl01.users.greyc.fr/CHEMISTRY/), which contain graphs representing molecular compounds. Acyclic contains 183 acyclic graphs; MAO contains 68 graphs including cycles. Both datasets contain node labels; edge labels are uniform. To also consider variability in edge labels, we used a dataset presented in [17], containing 72 graphs with non-uniform node and edge labels, representing pseudoknotted RNA secondary structures. We used a pre-processed version of the data provided by the authors of [5] (https://github.com/bionetslab/edge-preservation-similarity/). As elementary edit costs, we choose $(c^s_V, c^d_V, c^i_V, c^s_E, c^d_E, c^i_E) = (2, 4, 4, 1, 1, 1)$, as proposed in [1].

Evaluation Metrics. As evaluation metrics, we calculated the relative diversions of the matching costs $C(X^S)$ and $C(\widehat{X}^S)$ w.r.t. the matching cost (lower bound for GED) $C(X^H)$ of the ϵ-permutation matrix computed by BRANCH:

$$div(X^S) := \frac{C(X^S) - C(X^H)}{C(X^H) + 1} \quad div(\widehat{X}^S) := \frac{C(\widehat{X}^S) - C(X^H)}{C(X^H) + 1} \tag{8}$$

Since neither $C(X^S)$ nor $C(\widehat{X}^S)$ are provable lower or upper bounds for GED, it is unclear whether higher or lower values $C(X^S)$ and $C(\widehat{X^S})$ are closer to GED. This is why we use the term "diversion" in the context of comparison of the matching costs. If $div(X^S)$ and $div(\widehat{X}^S)$ are close to 0, $C(X^S)$ and $C(\widehat{X}^S)$ constitute approximations of GED which are close-to-equivalent to the lower bound $C(X^H)$ computed by BRANCH. To quantify the quality of the upper bound $c(\pi_{\widehat{X}^S})$, we computed the relative error:

$$error(\widehat{X}^S) := \frac{c(\pi_{\widehat{X}^S}) - c(\pi_{X^H})}{c(\pi_{X^H}) + 1} \tag{9}$$

Here, we use the term "error" because $c(\pi_{\widehat{X}^S})$ is an upper bound for GED and so a lower value is always preferable. If $error(\widehat{X}^S)$ is negative, the upper bound $c(\pi_{\widehat{X}^S})$ is tighter than the upper bound provided by BRANCH.

Implementation, Availability, and Hardware Specification. We implemented our algorithm in Python and relied on PyTorch to implement the adapted Sinkhorn algorithm suggested in [9]. The source code is available on GitHub (https://github.com/juliawal/continuous_ged_approximation). Tests were run on a MacBook with an Apple M1 processor and 8GB of RAM.

Quality of the Differentiable Approximation. Figure 3 shows the diversions of the matching costs of the ϵ-bistochastic matrices X^S, split across datasets (Fig. 3A) and for varying hyper-parameters α and β (Fig. 3B). For all three datasets, the median is around zero, with individual outliers that are largest for the RNA dataset and smallest for Acyclic. Larger values of α and β result in a smaller diversions. Overall, this means that our differentiable GED approximation is close to the lower bound for GED computed by BRANCH.

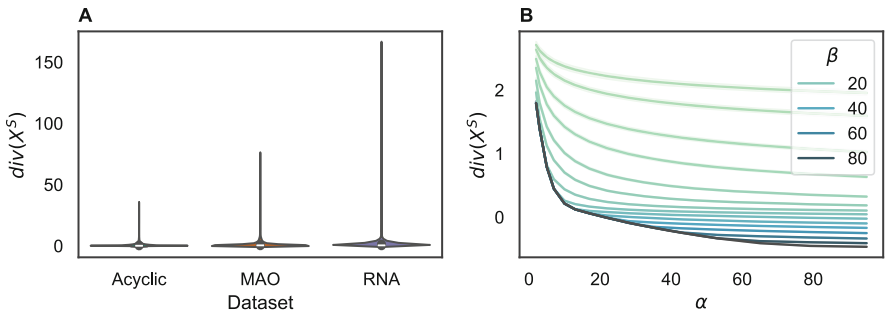


Fig. 3. Relative diversions of the matching costs of the ϵ-bistochastic matrices X^S from the matching costs of the ϵ-permutation matrices X^H computed by BRANCH. A: Violin plots showing distributions of relative diversions $div(X^S)$ split by datasets. B: Relative diversions $div(X^S)$ in dependence of α and β.

Quality of the Projection. Figure 4 shows the diversions of the matching costs of the projections $\widehat{X}^S$, split across datasets (Fig. 4A) and for varying hyper-parameters α and β (Fig. 4B). While we observe a minor increase in diversion for increasing α and β, diversions are overall tiny and even smaller than for the ϵ-bistochastic matrices.

The quality of the projections' induced edit costs $c(\pi_{\widehat{X}^S})$ (upper bound for GED) is visualized in Fig. 5. For all three datasets, the median relative error is around 0 (Fig. 5A). While there are individual outliers for the RNA dataset, for larger values of α and β, the upper bound $c(\pi_{\widehat{X}^S})$ is actually tighter than the upper bound computed by BRANCH for around 70 to 75% of all instances (Fig. 5B). Errors decrease with increasing values of α and β (Fig. 5C).

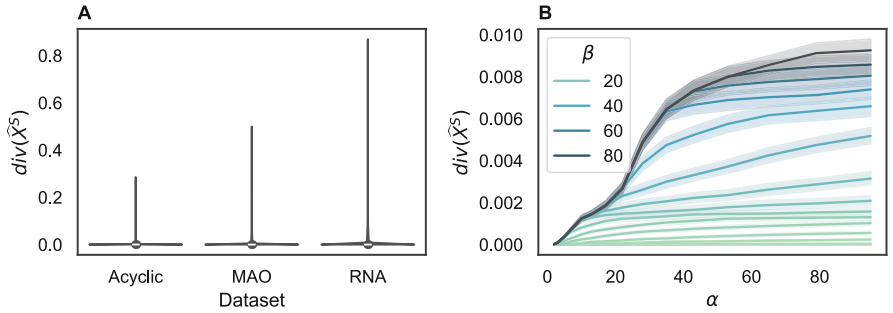


Fig. 4. Relative diversions of the matching costs of the projections $\widehat{X}^S$ of the ϵ-bistochastic matrices X^S from the matching costs of the ϵ-permutation matrices X^H computed by BRANCH. A: Violin plots showing distributions of relative diversions $div(\widehat{X}^S)$ split by datasets. B: Relative diversions $div(\widehat{X}^S)$ in dependence of α and β.

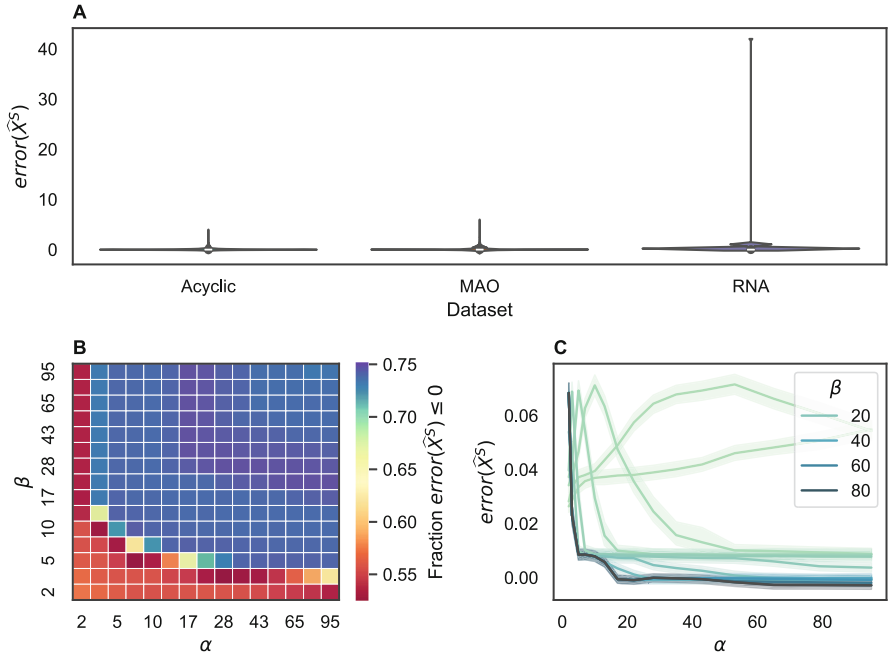


Fig. 5. Relative errors of the induced edit costs of the projections $\widehat{X}^S$ of the ϵ-bistochastic matrices X^S from the induced edit costs of the ϵ-permutation matrices X^H computed by BRANCH. A: Violin plots showing distributions of relative errors split by datasets. B: Fractions of instances with negative or zero relative errors. C: Relative errors in dependence of α and β.

Runtime. Figure 6 shows the runtime of the Sinkhorn algorithm when fed with similarity matrices constructed according to Eq. (7) in dependence of α and β. Larger values of α and β lead to longer runtimes, independently of the dataset.

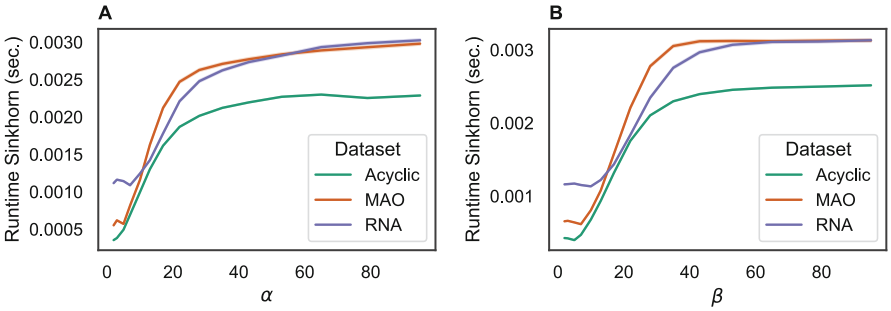


Fig. 6. Runtime of the Sinkhorn algorithm step within our continuous GED approximator for varying hyper-parameters α (A) and β (B).

5 Summary and Outloook

In this article, we presented a differentiable approximator for GED based on a reduction to LSAPE and an extended version of the Sinkhorn algorithm. Having access to such an approximator is desirable because it allows to incorporate the approximation of GED into deep metric learning frameworks, which may pave the way for a data-driven inference of elementary edit costs. Tests on three datasets showed that our approach performs similar to the widely used GED heuristic BRANCH. Our work opens up various avenues for future work: For instance, it would be interesting to assess the effect of replacing the Hungarian algorithm by the Sinkhorn algorithm also in the construction of the LSAPE instance C, which we here treated as part of the input. It may also be possible to formulate an error-correcting version of the optimal transport problem with entropic regularization, which may provide a strong theoretical justification for using the Sinkhorn algorithm as an approximate solver for LSAPE.

References

1. Abu-Aisheh, Z., et al.: Graph edit distance contest: results and future challenges. Pattern Recogn. Lett. **100**, 96–103 (2017). https://doi.org/10.1016/J.PATREC.2017.10.007
2. Blumenthal, D.B., Boria, N., Gamper, J., Bougleux, S., Brun, L.: Comparing heuristics for graph edit distance computation. VLDB J. **29**(1), 419–458 (2020). https://doi.org/10.1007/s00778-019-00544-1
3. Blumenthal, D.B., Gamper, J.: Improved lower bounds for graph edit distance. IEEE Trans. Knowl. Data Eng. **30**(3), 503–516 (2018). https://doi.org/10.1109/TKDE.2017.2772243
4. Blumenthal, D.B., Gamper, J.: On the exact computation of the graph edit distance. Pattern Recogn. Lett. **134**, 46–57 (2020). https://doi.org/10.1016/j.patrec.2018.05.002
5. Boria, N., Kiederle, J., Yger, F., Blumenthal, D.B.: The edge-preservation similarity for comparing rooted, unordered, node-labeled trees. Pattern Recogn. Lett. **167**, 189–195 (2023). https://doi.org/10.1016/j.patrec.2023.02.017

6. Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gaüzère, B., Vento, M.: Graph edit distance as a quadratic assignment problem. Pattern Recogn. Lett. **87**, 38–46 (2017). https://doi.org/10.1016/j.patrec.2016.10.001
7. Bougleux, S., Gaüzère, B., Blumenthal, D.B., Brun, L.: Fast linear sum assignment with error-correction and no cost constraints. Pattern Recogn. Lett. **134**, 37–45 (2020). https://doi.org/10.1016/j.patrec.2018.03.032
8. Bougleux, S., Gaüzère, B., Brun, L.: A Hungarian algorithm for error-correcting graph matching. In: Foggia, P., Liu, C.-L., Vento, M. (eds.) GbRPR 2017. LNCS, vol. 10310, pp. 118–127. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58961-9_11
9. Brun, L., Gaüzère, B., Renton, G., Bougleux, S., Yger, F.: A differentiable approximation for the linear sum assignment problem with edition. In: ICPR 2022, pp. 3822–3828. IEEE (2022). https://doi.org/10.1109/ICPR56361.2022.9956203
10. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recogn. Lett. **1**(4), 245–253 (1983). https://doi.org/10.1016/0167-8655(83)90033-8
11. Cortés, X., Conte, D., Cardot, H.: Learning edit cost estimation models for graph edit distance. Pattern Recogn. Lett. **125**, 256–263 (2019). https://doi.org/10.1016/j.patrec.2019.05.001
12. Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1–2), 83–97 (1955). https://doi.org/10.1002/nav.3800020109
13. Peyré, G., Cuturi, M.: Computational optimal transport: with applications to data science. Found. Trends® Mach. Learn. **11**(5–6), 355–607 (2019). https://doi.org/10.1561/2200000073
14. Rica, E., Álvarez, S., Serratosa, F.: On-line learning the graph edit distance costs. Pattern Recogn. Lett. **146**, 55–62 (2021). https://doi.org/10.1016/j.patrec.2021.02.019
15. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(7), 950–959 (2009). https://doi.org/10.1016/j.imavis.2008.04.004
16. Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pac. J. Math. **21**(2), 343–348 (1967). https://doi.org/10.2140/pjm.1967.21.343
17. Wang, F., Akutsu, T., Mori, T.: Comparison of pseudoknotted RNA secondary structures by topological centroid identification and tree edit distance. J. Comput. Biol. **27**(9), 1443–1451 (2020). https://doi.org/10.1089/CMB.2019.0512
18. Yoshida, T., Takeuchi, I., Karasuyama, M.: Distance metric learning for graph structured data. Mach. Learn. **110**(7), 1765–1811 (2021). https://doi.org/10.1007/s10994-021-06009-3
19. Zeng, Z., Tung, A.K.H., Wang, J., Feng, J., Zhou, L.: Comparing stars: on approximating graph edit distance. Proc. VLDB Endow. **2**(1), 25–36 (2009). https://doi.org/10.14778/1687627.1687631

Learning Graph Similarity by Counting Holes in Simplicial Complexes

Kalvin Dobler(✉) and Kaspar Riesen

Institute of Computer Science, University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
{kalvin.dobler,kaspar.riesen}@unibe.ch

Abstract. A major objective of graph-based pattern recognition is to quantify the similarity between two graphs. One method, *error-tolerant graph matching*, assigns a cost for structural differences. More recent approaches involve *graph kernels* or *graph neural networks*. In this paper, we introduce a novel approach using graph neural networks to embed graph nodes into real vector spaces. We then compute topological invariants over nested *simplicial complexes* and derive a similarity value. Extensive experiments show that our method can achieve comparable classification performance of other algorithms while significantly reducing computation time.

Keywords: Structural Pattern Recognition · Graph Similarity Computation · Graph Representation Learning · Simplicial Homology

1 Introduction

Research in structural pattern recognition can be broadly divided into three distinct eras. The first era is centered around the graph matching paradigm [1]. The second era focuses on adapting the kernel theory, initially developed for vectorial representations, to graphs (e.g., by means of *Random Walk Kernels* [2], or similar). With the advent of neural networks, there was a notable shift towards adapting and generalizing deep neural networks to non-Euclidean spaces. This shift marked the start of the third era of structural pattern recognition, characterized by the development and maturation of graph neural networks, such as *Graph Convolution Networks* (GCN) [3], or similar.

In a recent paper [4], it has been proposed to bridge the gap between the first and third era of structural pattern recognition. That is, a novel approach to speed up the computation of *graph edit distance* (GED) with *graph neural networks* (GNNs) has been proposed. In this method, a GNN is trained in an end-to-end fashion for graph classification and node embeddings from the last message passing layer of the trained GNN model are then extracted. Next, an explicit node-to-node assignment between two graphs is defined by solving a *linear sum assignment problem* (LSAP) on the node embeddings. Finally, the

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 11–20, 2025.
https://doi.org/10.1007/978-3-031-80507-3_2

similarity of two graphs is defined as the cost of the globally consistent node and edge edit operations implied by the assignment of node embeddings.

The drawback of the method proposed in [4] is, however, that computing node-to-node assignment for each combination of the underlying graph sets is highly redundant. That is, the process requires (1) computing pairwise distances between node embeddings for each graph pair, (2) solving an optimization problem, and then (3) calculating the cost of the edit operations implied by the assignment. Consider, for instance, a test and training set of 100 graphs each. This means computing $10,000$ cost matrices, each of them followed by an optimization problem and a calculation of corresponding edit operations. In the present paper we research an alternative approach to this process for reducing the runtime of the graph similarity computation.

A natural way to avoid redundant computations might be to rely on the node embeddings of each graph independently. We thus propose to consider the node embeddings as point clouds, then compute associated topological invariants, and compare these invariants between graphs to derive a similarity value (as depicted in Fig. 1). In the above mentioned example, this means computing topological invariants for each of the 200 training and test graphs once, which can then be compared by means of any similarity measure defined in $\mathbb{R}^n$.

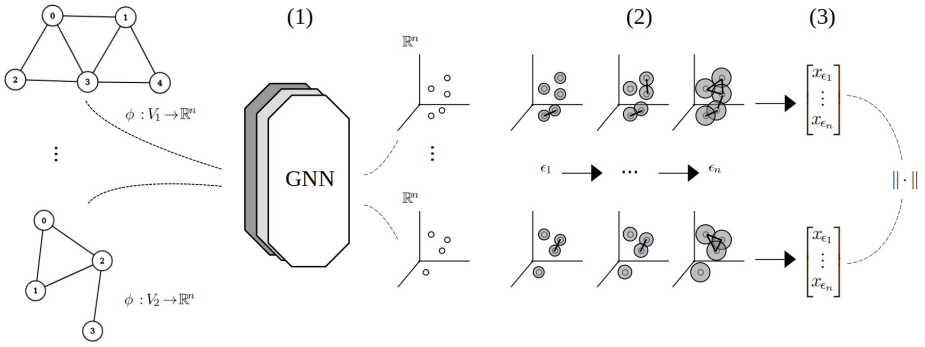


Fig. 1. Overview of the proposed approach. It involve (1) learning node embeddings with GNNs, (2) computing topological invariants over nested simplicial complexes, and (3) deriving a similarity value from these invariants.

The proposed approach is somehow related to previous works that leverage deep learning for graph similarity computation. For example, GMN [5] and SimGNN [6] use GNNs as encoders and compute graph similarities based on the resulting graph embeddings. Another related research direction involves unsupervised graph similarity learning through contrastive learning, such as CGMN [7]. In addition, topological descriptors, i.e., numerical quantifiers of graphs, can be used to derive graph similarities as well (e.g., Polarity number, Wiener index, or similar [8]). Our approach differs from these methods in several important ways. First, instead of relying solely on the embedding vector generated by the GNN

model, we intend to directly use the node embeddings in order to derive graph similarities. Secondly, unlike topological descriptors, the message passing mechanism of GNNs provides us with rich structural information encoded in the node embeddings, which is in turn captured in the proposed topological invariants.

The remainder of this paper is organized as follows. First, in Sect. 2, we provide the necessary background to comprehend our novel approach. Then, in Sect. 3, we describe the details of the proposed graph similarity computation procedure. In Sect. 4, we present the experimental setup and provide the results of our experimental evaluation. Finally, in Sect. 5, we derive some conclusions and discuss future research avenues.

2 Background

To keep the paper self-contained, we review two concepts, viz. *Graph Neural Networks* and *Simplicial Homology*, actually relevant for our procedure.

2.1 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) generalize the set of methods used in deep neural networks to graph-structured data. At a high level, GNNs aim to produce effective representations of nodes in a graph by leveraging the graph structure and the relationships between nodes. This allows GNNs to effectively capture local patterns and relationships within a graph, while also considering the overall graph structure, to make predictions. This is accomplished through an iterative process of information propagation across the graph, where each node $v \in V$ updates its representation based on the information received from its neighbors $u \in \mathcal{N}(v)$. Typically, the architecture of a GNN model consists of several layers, each of which contains a message passing mechanism, aggregation functions, and learnable parameters. Formally, these can be expressed by

$$h_v^{t+1} = \text{UPDATE}\left(h_v^{(t)}, \text{AGGREGATE}\left(\{h_u^{(t)}, \forall u \in \mathcal{N}(v)\}\right)\right),$$

where $h_v^{(t)}$ corresponds to the representation of node v at the t-th layer. During each message-passing step, the AGGREGATE function takes as input the set of representations of the neighbors $u \in \mathcal{N}(v)$ of node v to generate a new message. Common aggregation methods include sum, mean, and max. The UPDATE function then combines the aggregated information with the current node representation of v to produce a new representation. The specific form of the UPDATE function depends on the architecture and design of the GNN. It usually involves a neural network layer, which may contain parameters such as weights and biases that are learned during the learning process. This new representation is then passed on to the next GNN layer.

The final node representations h_v^T can be used for various tasks. For node-level tasks, they predict some property for each node. For graph-level tasks, they predict some property for the entire graph, typically using pooling layers that aggregate a vectorial representation for a graph.

2.2 Simplicial Homology

Simplicial Complexes. We denote a k-simplex σ as the convex hull of $k+1$ points, which lie in an n-dimensional vector space. The collection of simplices defines a *simplicial complex* K, such that the intersection of any two simplices is either empty, or in a single simplex.

Given a metric space (M, d), where M is a set and d is a metric on M, and threshold parameter $\epsilon > 0$, the *Vietoris-Rips complex* $VR_\epsilon(M)$ is the simplicial complex that contains a finite simplex $\sigma \subseteq M$ whenever $d(p, q) \leq \epsilon$ for every pair $p, q \in \sigma$ [9]. We denote a sequence of nested simplicial complexes as a *filtration* where $\emptyset \subseteq K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n$.

Chain Complexes and Homology Groups. A chain complex is a sequence of vector spaces together with a sequence of linear transformations δ, defined as

$$\ldots \to C_{k+1} \xrightarrow{\delta_{k+1}} C_k \xrightarrow{\delta_k} C_{k-1} \to \ldots \to C_1 \xrightarrow{\delta_1} C_0 \xrightarrow{\delta_0} 0$$

Each chain C_k defines a vector space with basis consisting of the set of k-simplices. Formally, we write $c = \sum_i a_i \sigma_i$, where a_i refers to the i^{th} coefficients and σ_i refers to the k-simplices. We denote the k^{th} chain group C_k as the *free abelian* group generated by the k-chains on K [10]. For any $k \in \mathbb{Z}$, the linear map δ_k between chain groups is called the *boundary map* $\delta_k : C_k \to C_{k-1}$, which is defined as an alternating sum

$$\delta_k(\sigma) = \sum_{i=0}^{k} (-1)^i \{v_0, \ldots, \hat{v}_i, \ldots, v_k\}$$

where $\hat{v}_i$ indicates that the node v_i is omitted. The coefficients are defined over $\mathbb{Z}_2$ which allows us to disregard the orientation of simplices. It can be shown that the composition of two boundary maps is zero, that is, for any $k \in \mathbb{Z}$,

$$\delta_k \circ \delta_{k+1} = 0.$$

This equation implies the following inclusion

$$\text{Im } \delta_{k+1} \subseteq \text{Ker } \delta_k$$

where Im and Ker refer to image and kernel, respectively. Consequently, we can define the n^{th} homology group of the chain complex to be the quotient group $H_k = \text{Ker } \delta_k / \text{Im } \delta_{k+1}$. Elements of Ker δ_k are called *cycles* and denoted by Z_k, while elements of Im δ_{k+1} are called *boundaries* and denoted by B_k [10].

H_k refers to the number of k-dimensional holes in the simplicial complex. In other words, $\dim(H_0)$ counts the number of connected components of the space, $\dim(H_1)$ counts the number of holes, $\dim(H_2)$ counts the number of voids, and so on. Intuitively, we are interested in cycles that are not the boundary of any higher dimensional simplex. In other words, by modding out by the boundary, we sent elements of Z_k, that are the boundary of a higher dimensional hole, to 0.

We further denote the rank of the n^{th} homology group as the n^{th} *Betti number* β_n, which refers to the number of distinct holes, i.e., the dimension of this vector space. The Euler characteristic $\mathcal{X}(K)$ is defined as the alternating sum of the number of k-simplices in the simplicial complex K.

3 Graph Matching Through Simplicial Homology

The novel graph matching approach that we present in this paper consists of three fundamental steps. First, we train a *Graph Isomorphism Network* (GIN) [11] in an end-to-end fashion for graph classification (other GNN architectures could also be used). Second, we extract the node embeddings of the last message passing layer of the trained GIN. Third, we compute topological invariants of the point clouds associated with each graph. In particular, we compute the Euler characteristics $\mathcal{X}$ and the Betti numbers β for each point cloud representing a graph (see Sect. 2).

In practice, one would be interested in analyzing topological features across different scales ϵ, as the correct choice of ϵ is not given *a priori*. However, computing a filtration on $\mathbb{Z}$ is computationally expensive and memory intensive. Our strategy is thus to consider a plausible set of ϵ values in order to compute a sample of filtered simplicial complexes. To this end, for each data set, we compute the distribution of all pairwise distances of node embeddings between training graphs. This distribution contains information about the frequency of all distances between points for a given data set. The idea is then to determine a set of ϵ values based on the training graphs only. This seems reasonable as the embeddings produced by the GIN can be considered to come from the same space, provided that the network architecture, training process, and input consistency are maintained. Hence, by considering a sample of these distances, we ensure that a sufficiently large number of distances covering a substantial part of the space is available. We then consider ϵ values as the $\{0.15, 0.30, 0.45\}$-quantiles of these distributions.

For our application, we hypothesize that a sample of filtered simplicial complexes may suffice to derive meaningful similarity values, as the node embeddings implicitly encode wide structural information derived by the message passing mechanism of GNNs. That is, two similar graphs should have similar point clouds and a sample of filtered simplicial complexes should provide sufficient information to capture this similarity.

The set of quantiles is an important hyperparameter. The more quantiles we consider, the more simplicial complexes we build, which in turn increases the computational cost. For each ϵ value, we compute the Vietoris-Rips complex of the point clouds and associated topological invariants. In particular, we consider the Euler characteristics for simplicial 1-complex and 2-complex as well as the Betti numbers up to β_2. For instance, if we consider computing Euler characteristics for the mentioned set of quantiles, we would obtain a vector of dimensionality 3, whereas we would obtain vectors of dimensionality 6 and 9 for $\beta_{0:1}$ (β_0 to β_1) and $\beta_{0:2}$ (β_0 to β_2), respectively. Finally, we compute the

similarity between each pair of graphs using the Euclidean distance of these vectors.

From now on, we term the graph similarity computation computed by this procedure as GNN-TDA (where TDA stands for *Topological Data Analysis*).

4 Experimental Evaluation

4.1 Reference Systems

The goal of the experimental evaluation is to compare the classification performance and runtime of our novel approach GNN-TDA with two reference systems, viz. *Bipartite Graph Edit Distance* (BP-GED) [12] and *Graph Neural Network Graph Edit Distance* (GNN-GED) [4].

BP-GED is an approximate algorithm for the computation of the GED in a substantially faster way than traditional methods. In this approach, the *quadratic assignment problem* (QAP) of GED computation is reduced to an instance of an LSAP. As this approach neglects the structural relationships between the nodes in a global way, the minimum sum of edge edit operation costs, implied by the corresponding node operation, is integrated in the LSAP solving problem. BP-GED can be used to derive a lower and an upper bound on the true GED [13] (in the present paper, we use the somewhat more common upper-bound approximation of the true GED).

GNN-GED leverages GNNs to capture the structural information of a graph. Specifically, the node embeddings extracted from the GNN are optimally mapped using an LSAP solver. The edit cost associated with this assignment defines the similarity value between the graphs. The advantage of GNN-GED over BP-GED is twofold. First, its computation is substantially faster. Second, it solves the assignment on node embeddings, which inherently encode wide structural knowledge through the message passing mechanism of GNNs.

4.2 Data Sets

We employ five data sets from the TUDataset graph repository [14] for our evaluation. Three of these data sets are related to small molecules: MUTAG, AIDS, and NCI1. The MUTAG data set includes *mutagenic* and *non-mutagenic* compounds, the AIDS data set contains *confirmed active* and *confirmed inactive* molecules based on their efficacy in protecting human cells from HIV, and the NCI1 data set features *active* and *inactive* molecules according to their ability to inhibit the growth of non-small cell lung cancer. The other two data sets are from the bioinformatics domain. The ENZYMES data set categorizes enzymes according to the six EC top-level classes, while the OHSU data set includes functional brain networks identified based on hyperactive-impulsive regions.

4.3 Experimental Setup

For each dataset, we train a GIN model in an end-to-end fashion to classify the corresponding graphs [11]. The model architecture consists of GIN layers, each implemented as a linear layer, batch normalization, ReLU activation, linear layer, and ReLU activation. We then concatenate the graph embeddings and feed them into a final multilayer perceptron (MLP) composed of two linear layers. We concatenate them instead of adding them to help preserving information from previous layers.

We consider the number of GIN layers $\in \{3, 4, 5\}$, hidden node feature dimensions $\in \{32, 64, 128\}$, and learning rates $\in \{0.0001, 0.001, 0.01\}$, as hyperparameters. To fine-tune these parameters, we perform a 10-fold cross-validation on each dataset. We produce splits of each data set into training and test sets, using a $80\% - 20\%$ split size. Specifically, the former split is used for 10-fold cross-validation and subsequent training of the models, while the latter split is held out from the beginning and only considered for final testing. In Table 1, we present the best-performing hyperparameters for each dataset.

Table 1. Best performing hyperparameters for all data sets.

Dataset	# GIN layers	Node feature dimension	Learning rate
MUTAG	3	64	$1e-3$
AIDS	3	64	$1e-4$
NCI1	4	64	$1e-4$
ENZYMES	5	64	$1e-4$
OHSU	5	64	$1e-3$

To recall, we use the GIN as an auxiliary encoder to learn node embeddings. We consider these embeddings independently as point clouds for computing topological invariants, which are finally used to derive similarity values between graphs (as described in Sect. 3). For final classification, we employ a k-nearest neighbor classifier (k-NN) and we optimize $k \in \{3, 5, 7, 11\}$ based on the training graphs only.

4.4 Classification Accuracy and Runtime Computation

First, we investigate the differences of the accuracies of a k-NN classifier that either operates on the reference distances (calculated by means of BP-GED [12] and GNN-GED [4]) or on our novel distances obtained by GNN-TDA. In Table 2 we report the F1 measures for all variants. We observe that our novel approach achieves comparable recognition accuracies on two data sets (MUTAG and AIDS), while it is rather suboptimal for the other data sets. Note that the classification accuracy depends on the number of filtered simplicial complexes

considered and thus directly to the number of ϵ values. We assume that considering only a very small sample of filtered simplicial complexes can still achieve a meaningful classification accuracy. It is worth noting that we limit our consideration of topological invariants up to H_2. However, theoretically, we could extend this consideration up to H_n where $n > 2$, to achieve more detailed insights into the space, albeit with increased computational costs.

Table 2. F1-scores (%) for all data sets. $\mathcal{X}_k$ refers to the Euler characteristics up to simplicial k-complexes, whereas β_n refers to the Betti numbers up to the n^{th} Betti number.

Dataset	Reference Systems		GNN-TDA			
	BP-GED	GNN-GED	$\mathcal{X}_1$	$\mathcal{X}_2$	β_1	β_2
MUTAG	90.5	89.8	89.7	90.0	89.7	90.0
AIDS	99.7	100.0	98.8	97.7	99.1	98.2
NCI1	74.8	77.5	61.8	58.9	61.5	59.9
ENZYMES	36.6	41.6	25.0	25.9	22.5	23.4
OHSU	66.6	52.6	37.5	50.0	37.5	50.0

Next, in Table 3, we compare the runtime for computing all pairwise similarity values on one specific dataset (averaged over five runs). Overall, the runtime computations of our novel approach GNN-TDA is substantially lower than the runtime of both BP-GED and GNN-GED. We provide speed-up values corresponding to the ratio between the reference methods and fastest GNN-TDA runtimes. We observe maximum speed-up factors ranging from at least 6 to at most 252.

Table 3. Total computation time in minutes (averaged over five runs). $\mathcal{X}_k$ refers to the Euler characteristics up to simplicial k-complexes, whereas β_n refers to the Betti numbers up to the n^{th} Betti number.

Dataset	Reference Systems		GNN-TDA				Speed-up w.r.t.	
	BP-GED	GNN-GED	$\mathcal{X}_1$	$\mathcal{X}_2$	β_1	β_2	BP-GED	GNN-GED
MUTAG	0.070	0.020	0.001	0.007	0.004	0.100	70	20
AIDS	7.300	1.800	0.084	0.201	0.163	8.200	87	21
NCI1	84.500	20.200	0.336	1.134	0.829	69.803	252	60
ENZYMES	2.100	0.400	0.012	0.207	0.073	51.447	175	33
OHSU	0.200	0.030	0.005	1.159	0.079	297.160	40	6

5 Conclusion and Future Work

We propose and research a novel graph matching framework using GNNs and simplicial homology. The proposed approach directly builds up on our previous work [4] which aims at speeding-up GED computation. In particular, we first train a GNN in an end-to-end fashion in order to learn node representations. The node representations are then considered as point clouds and topological invariants are computed for few filtered simplicial complexes only. In an experimental evaluation, we observe that our approach substantially improves the runtime of graph similarity computation while achieving comparable classification accuracies on two data sets. However, it performs suboptimally on the other datasets. Interestingly, we notice that a small number of filtered simplicial complexes might be sufficient to achieve meaningful classification accuracies, although this depends on the dataset and the topological invariants being considered.

We see diverse promising avenues to be pursued in future work. First, other neural network architectures rather than GIN may be employed. Second, the graph-focused task for classification currently serves as a proxy to extract node representations. We plan to develop training losses that align more closely with our original objective. Third, this work raises interesting research questions regarding the procedure for determining ϵ values and the optimal number of such values to consider, albeit with increased computational cost.

Acknowledgments. This study is funded by the Swiss National Science Foundation (SNSF) Project Nr 200020_219388.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. J. Pattern Recogn. Artif. Intell. **18**(3), 265–298 (2004). https://doi.org/10.1142/S0218001404003228
2. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: Fawcett, T., Mishra, N. (eds.) Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), Washington, DC, USA, 21–24 August 2003, pp. 321–328. AAAI Press (2003). http://www.aaai.org/Library/ICML/2003/icml03-044.php
3. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017, Conference Track Proceedings. OpenReview.net (2017). https://openreview.net/forum?id=SJU4ayYgl
4. Dobler, K., Riesen, K.: Learning graph matching with graph neural networks (submitted for publication). In: Artificial Neural Networks in Pattern Recognition (ANNPR) (2024)
5. Li, Y., Gu, C., Dullien, T., Vinyals, O., Kohli, P.: Graph matching networks for learning the similarity of graph structured objects. CoRR, abs/1904.12787 (2019). http://arxiv.org/abs/1904.12787

6. Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., Wang, W.: SimGNN: a neural network approach to fast graph similarity computation. In: Shane Culpepper, J., Moffat, A., Bennett, P.N., Lerman, K. (eds.) Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, 11–15 February 2019, pp. 384–392. ACM (2019). https://doi.org/10.1145/3289600.3290967
7. Jin, D., et al.: CGMN: a contrastive graph matching network for self-supervised graph similarity learning. In: De Raedt, L. (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23–29 July 2022, pp. 2101–2107. ijcai.org (2022). https://doi.org/10.24963/IJCAI.2022/292
8. Todeschini, R., Consonni, V.: Handbook of Molecular Descriptors. Wiley-VCH (2000)
9. Dey, T.K., Wang, Y.: Computational Topology for Data Analysis. Cambridge University Press, Cambridge (2022)
10. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
11. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. OpenReview.net (2019). https://openreview.net/forum?id=ryGs6iA5Km
12. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(7), 950–959 (2009). https://doi.org/10.1016/J.IMAVIS.2008.04.004
13. Riesen, K.: Structural Pattern Recognition with Graph Edit Distance. ACVPR, Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27252-8. ISBN 978-3-319-27251-1
14. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tudataset: a collection of benchmark datasets for learning with graphs. CoRR, abs/2007.08663 (2020). https://arxiv.org/abs/2007.08663

Community-Hop: Enhancing Node Classification Through Community Preference

Ahmed Begga[1(✉)], Waqar Ali[2], Gabriel Niculescu[1], Francisco Escolano[1], Thilo Stadelmann[3,4], and Marcello Pelillo[2,3]

[1] University of Alicante, Alicante, Spain
ahmed.begga@ua.es
[2] Ca' Foscari University of Venice, Venice, Italy
[3] ECLT European Centre for Living Technology, Venice, Italy
[4] ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland

Abstract. In recent years, Graph Neural Networks (GNNs) have demonstrated significant influence on the analysis of graph structures by leveraging message-passing mechanisms to aggregate neighborhood information and perform various graph-related tasks from node classification to link prediction. Recently, GNNs have mostly been developed to deal with different types of graph structures, such as homophily (similar labels among connected nodes) and heterophily (dissimilar labels among connected nodes). However, existing methods lack the ability to combine node features and graph topology optimally to deal with heterophily. This paper proposes a Community-HOP-based GNN model for dealing with homophilic and heterophilic graph structures. Specifically, we incorporate valuable insights from the graph community structure to guide the feature aggregation process of the GNN layer to learn diverse graph properties and improve performance on node-level tasks. Extensive experiments on six node-level datasets under standard metrics demonstrate that the Community-HOP method surpasses existing baselines.

Keywords: Graph Neural Networks · Node Classification · Spectral Clustering

1 Introduction

GNNs have proven to be powerful methods for analyzing graph-based data, finding use in diverse areas such as social network analysis and predicting molecular properties [8,9,18]. However, conventional GNN architectures often face challenges in capturing complex structural information and relationships between nodes at various distances, particularly in graphs with heterophilic properties (where connected nodes have dissimilar labels) [21]. Recent advancements in GNN design have addressed these limitations by incorporating higher-order neighborhood information. Models such as MixHop [1] and FSGNN [12] have

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 21–30, 2025.
https://doi.org/10.1007/978-3-031-80507-3_3

demonstrated the effectiveness of aggregating features from nodes at different hop distances. These approaches allow for more flexible feature mixing and improved performance on various graph datasets.

While these methods have shown promise, they often treat all neighbors equally within each hop distance, potentially overlooking important structural information encoded in the graph's community structure. Community detection in graphs has long been a subject of study in network science [11,15], with spectral methods providing powerful tools for identifying clusters of densely connected nodes [3].

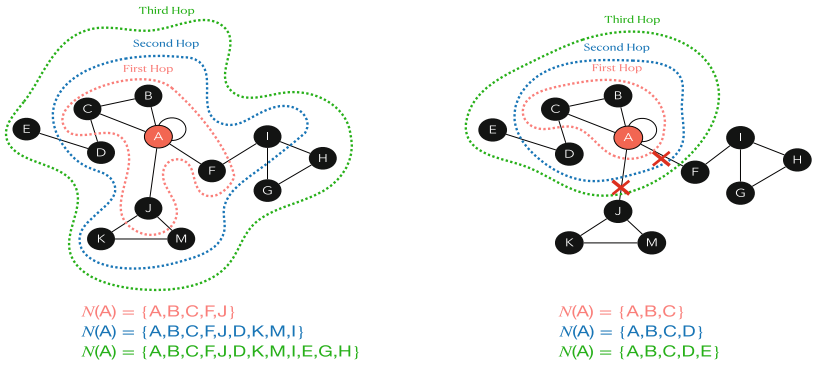


Fig. 1. Traditional Hop vs Community Hop. Evolution of the neighborhood of A, $N(\mathrm{A})$, in traditional hops (left) and in our approach, Community Hop (right).

This study develops a novel GNN layer that leverages graph community structure to guide the feature aggregation process. By combining spectral community detection techniques [11] with a modified transition matrix for inter-community hops, our approach aims to prioritize information flow within and between communities in a more meaningful way. This community-aware feature aggregation strategy allows the model to capture both local (by just combining community nodes) and global (by using the same GNN for all the clusters) graph structures more effectively.

2 Related Work

Recent years have seen significant advancements in GNN architectures, particularly in addressing the challenges of heterophily and over-smoothing. These innovations have largely focused on modifying the feature aggregation process and leveraging higher-order neighborhood information. Several approaches have been proposed to tackle the heterophily problem, where connected nodes may have dissimilar features or labels. H2GCN [21] introduced ego- and neighbor-embedding separation, along with the exploration of higher-order neighborhood

structures. GPR-GNN [5] minimizes over-smoothing by integrating the PageRank method with GNNs. Furthermore, GGCN [20] addresses heterophily and over-smoothing issues by utilizing degree corrections and signed messages.

Interestingly, studies have revealed that basic models like Multi-Layer Perceptrons (MLPs) and LINK [10] can occasionally surpass conventional GNN architectures when dealing with heterophilic datasets. This observation has led to the development of hybrid methods that merge node features with graph-based representations. A prominent example is LINKX [10], which integrates MLPs for node features with LINK regression, showing promising performance on heterophilic graphs.

Another line of research has focused on aggregating features from neighbors at different distances. MixHop [1] and FSGNN [12] utilize the transition matrix's powers to capture multi-hop neighborhood information. FSGNN uses a regularizer method, such as softmax and L2-Normalization in GNN's layers.

Recent work has also explored novel ways to address the over-smoothing problem in deeper GNN architectures. Ordered GNN [16] proposes an approach that aligns the hierarchy of a rooted-tree with ordered neurons in node embeddings, effectively preserving information from different neighborhood depths.

While these advancements have significantly improved GNN performance on various graph types, there remains room for innovation in leveraging graph structure more effectively, particularly in the context of community detection and inter-community information flow. Our proposed method builds on these insights by incorporating spectral clustering and community hops, offering a novel approach to enhance GNN performance across diverse node-level predictions.

3 Preliminaries

In this section, we define the mathematical notations used in this study. Let $G = (V, E)$ represent an undirected input graph, where V denotes the set of nodes and $E \subseteq V \times V$ represents the set of edges. We use the adjacency matrix $A \in \{0, 1\}^{n \times n}$ to capture the graph's topological structure, where $A_{ij} = 1$ if $(i, j) \in E$ and $A_{ij} = 0$ otherwise.

To account for self-loops, we modify the adjacency matrix to $\tilde{A} = A + I$, where I denotes the identity matrix. The features of each node are now represented by a matrix $F \in \mathbb{R}^{n \times k}$, where k indicates the dimension of the feature space.

Additionally, we utilize the diagonal degree matrix D for the graph G, where $D_{ii} = d_i$ denotes the degree of node i, calculated by $d_i = \sum_j A_{ij}$. The normalized transition matrix P is then defined as $P = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$.

3.1 Spectral Clustering

Spectral clustering is a robust method that utilizes the spectral properties of graph Laplacians to achieve clustering [11]. This section covers the essential matrices and concepts that are fundamental to spectral clustering techniques.

A key component in spectral clustering is the graph Laplacian, which comes in two primary forms. The unnormalized graph Laplacian is given by $L = D - W$, where W represents the weighted adjacency matrix and D is the diagonal degree matrix, with $D_{ii} = \sum_j w_{ij}$.

Furthermore, the normalized graph Laplacian can be represented by the following formula: $\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$. Here, $\mathcal{L}$ provides a normalized version of the Laplacian that adjusts for the degree of nodes, facilitating more effective clustering.

These Laplacians have several important properties [6]:

1. They are symmetric and positive semi-definite.
2. The smallest eigenvalue is 0, with corresponding eigenvector $\mathbf{1}$ for L and $D^{1/2}\mathbf{1}$ for $\mathcal{L}$.
3. They have n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

For any vector $f \in \mathbb{R}^n$, we have:

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2; \quad f^T \mathcal{L} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 \tag{1}$$

The multiplicity k of the eigenvalue 0 equals the number of connected components in the graph. The eigenspace of 0 is spanned by the indicator vectors of these components for L, and by $D^{1/2}$-scaled indicator vectors for $\mathcal{L}$ [11].

Spectral clustering is closely related to the Normalized Cut (NCut) problem [15]. Given a partition of V into k disjoint subsets $A_1, \ldots, A_k$, the NCut is defined as:

$$\mathrm{NCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, V \setminus A_i)}{\mathrm{vol}(A_i)} \tag{2}$$

where $\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$ and $\mathrm{vol}(A) = \sum_{i \in A} d_i$.

Minimizing NCut is NP-hard, but it can be relaxed to a tractable eigenvalue problem. This relaxation leads to the spectral clustering algorithm, which computes the first k eigenvectors $u_1, \ldots, u_k$ corresponding to the k smallest eigenvalues of $\mathcal{L}$ (or generalized eigenvectors of $Lu = \lambda D u$) [11,15].

These eigenvectors form a matrix $U \in \mathbb{R}^{n \times k}$, where each row represents a node's k-dimensional embedding. This embedding enhances cluster properties in the data [11], allowing for easier separation in the new representation.

The final step involves clustering these embeddings, typically using the k-means algorithm, to obtain the approximate solution to the NCut problem. This approach effectively captures important graph properties such as communities and structural characteristics through the spectrum of the Laplacian, providing a powerful tool for graph partitioning [6,11,15].

3.2 Graph Neural Networks

Graph Neural Networks (GNNs) have emerged as a significant technique for handling data that is structured as graphs. These models adapt the concepts of

convolutional neural networks to the non-Euclidean nature of graph data. The core idea behind GNNs is to iteratively update node representations by collecting and processing information from neighboring nodes. A typical GNN layer can be formulated as:

$$H^{(i+1)} = \sigma(AH^{(i)}W^{(i)}), \tag{3}$$

where $H^{(i+1)}$ denotes the updated node features matrix at layer $i+1$ after applying the layer transformation,$H^{(i)}$ represents the node features matrix before the transformation, $W^{(i)}$ is a matrix of learnable parameters and σ is a non-linear activation function. In the literature [9], it is common to use the normalized adjacency matrix, which is denoted as $\bar{A} = D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}$.

4 Methodology

Our methodology addresses the limitations of existing GNN approaches by combining spectral graph theory with flexible multi-hop neighborhood aggregation. Figure 1 illustrates the difference between traditional hops and our community-based approach, which mitigates oversmoothing by emphasizing communal connections [4].

The foundation of our approach leverages spectral graph clustering to uncover global community structure. We begin with the eigendecomposition of the normalized Laplacian $\mathcal{L}$ [6]:

$$\mathcal{L} = U\Lambda U^T, \tag{4}$$

where U is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues. The spectral properties of $\mathcal{L}$ are intimately connected to the graph's structure, with eigenvalues in the interval $[0, 2]$ [6].

We focus on the spectral gap, defined as $\gamma = \lambda_2 - \lambda_1$, where $\lambda_1 = 0$ and λ_2 is the smallest non-zero eigenvalue. This gap is related to the graph's connectivity and mixing time [17]. Specifically, the Cheeger constant $h(G)$, which measures the "bottleneckedness" of the graph, is bounded by the spectral gap through the Cheeger inequality:

$$\frac{\lambda_2}{2} \leq h(G) \leq \sqrt{2\lambda_2}, \tag{5}$$

This relationship, known as the Lovász bound [2,11], provides crucial insights into the graph's community structure. A small Cheeger constant indicates the presence of well-defined communities, while a large constant suggests a more uniformly connected graph [7].

We select the k leading eigenvectors corresponding to the smallest non-zero eigenvalues, where k is a hyperparameter. The choice of k can be guided by examining subsequent spectral gaps ($\lambda_{i+1} - \lambda_i$), with a large gap suggesting a natural number of clusters [11].

To identify communities, we apply k-means clustering to the rows of the truncated eigenvector matrix U_k. This spectral embedding tends to separate nodes into more linearly distinguishable clusters than in the original graph space.

We then introduce an edge-pruning mechanism to emphasize intra-community connections:

$$\mathcal{A}_{ij} = A_{ij} \cdot [C(i) = C(j)], \tag{6}$$

where $C(i)$ denotes the cluster assignment of node i. This pruning creates a block-diagonal structure in $\mathcal{A}$, aligning with the theoretical expectation of an ideal community structure in the spectral clustering framework. Following MixHop [1] and FSGNN [12], we will use the transition matrix to perform hops but this time with $\mathcal{A}$. Now the transition matrix can be defined as $\mathcal{P} = D^{-\frac{1}{2}}\mathcal{A}D^{-\frac{1}{2}}$

Building on this community-aware structure, we incorporate a multi-hop aggregation scheme with attention-like learnable parameters, inspired by recent GNN advancements [1,12]. The feature update rule for the $(i+1)$-th layer is:

$$H^{(i+1)} = \left[\alpha_1 A H^{(i)} W_1^{(i)} \| \alpha_2 \mathcal{P}^1 H^{(i)} W_2^{(i)} \| \alpha_3 \mathcal{P}^2 H^{(i)} W_3^{(i)} \| \cdots \| \alpha_{j+1} \mathcal{P}^j H^{(i)} W_{j+1}^{(i)}\right], \tag{7}$$

where j is the number of hops, $H^{(i)} \in \mathbb{R}^{n \times d_i}$ is the node feature matrix at the i-th layer, with n nodes and d_i features. $\mathcal{P} \in \mathbb{R}^{n \times n}$ is our community-aware transition matrix, and $W_{j+1}^{(i)} \in \mathbb{R}^{d_i \times d_{out}}$ are learnable weight matrices for each hop distance j at layer i, and $\|$ denotes column-wise concatenation.

We introduce learnable attention-like parameters α_{j+1} for each hop embedding and the original node features and adjacency, allowing the model to weigh the importance of different neighborhood scales adaptively. Importantly, these attention parameters are constrained to sum to 1: $\sum_{k=1}^{j+1} \alpha_k = 1, \quad \alpha_k \geq 0 \quad \forall k \in \{0, 1, \ldots, m\}$. This constraint ensures that the attention mechanism is a proper weighting system across different hop distances.

This multi-hop aggregation allows the model to simultaneously capture and weigh information from various neighborhood scales, as we illustrate in Fig. 2. For instance, if $j = 2$, the model considers the initial adjacency ($\alpha_1 A H^{(i)} W_1^{(i)}$), its immediate neighbors ($\alpha_2 \mathcal{P}^1 H^{(i)} W_1^{(i)}$), and its 2-hop neighbors ($\alpha_3 \mathcal{P}^2 H^{(i)} W_2^{(i)}$) in each layer. The use of different weight matrices $W_{j+1}^{(i)}$ and attention parameters α_{j+1} for each hop distance and the initial adjacency, enables the model to learn the relative importance of information from different scales while maintaining a balanced aggregation.

This approach generalizes the power iteration method often used in spectral clustering, allowing the capture of higher-order relationships in the graph while maintaining the ability to differentiate between local and global structural information. By learning to assign different importance to various neighborhood scales and preserving the original feature information, our model effectively captures complex patterns of node similarity and dissimilarity, adapting to both homophilic and heterophilic graph structures.

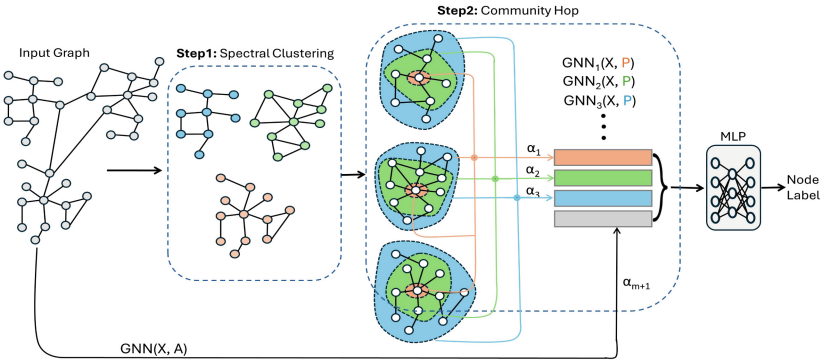


Fig. 2. Illustration of the spectral clustering and GNN propagation process. The input graph undergoes spectral clustering to identify communities (Step 1). Then, a community-aware multi-hop aggregation is performed (Step 2), where information is propagated within communities. The obtained node representations are concatenated and then passed through a MLP for the final prediction of node labels.

4.1 Computational Complexity

Our approach's computational complexity is divided into two primary processes: preprocessing and processing.

The preprocessing phase involves calculating the k leading eigenvectors of the normalized Laplacian matrix. For a graph with n nodes and m edges, this computation has a worst-case time complexity of $O(n^3)$ and a space complexity of $O(n^2)$. However, performance can be enhanced by employing optimized algorithms tailored for sparse graphs.

During the processing phase, k-means clustering and GNN propagation are performed. This step has a time complexity is $O(nk^2 + LHmF)$, where k denotes the number of clusters and the dimension of spectral embedding, L is the number of GNN layers, H represents the number of hops, and F is the number of features. The space complexity for the processing phase is $O(n(k + F) + m)$.

Despite the preprocessing step being computationally intensive, especially for larger graphs, it provides a comprehensive basis for identifying global community structures. This trade-off between computational cost and structural insight allows our approach to effectively capture both global and local patterns in the graph, enabling robust performance on both homophilic and heterophilic graph structures.

5 Experiments and Discussions

This section evaluates the proposed method's performance on six node classification benchmarks. The experimental results reveal that the Community-HOP method achieved the highest performance on four out of six datasets compared to

Table 1. Node-classification accuracies. Top three models are highlighted: **First**, **Second**, **Third**.

	Texas	**Wisconsin**	**Cornell**	**Citeseer**	**Pubmed**	**Cora**
Hom level	0.11	0.21	0.30	0.74	0.80	0.81
# Nodes	183	251	183	3,327	19,717	2,708
# Edges	295	466	280	4,676	44,324	5,278
# Classes	5	5	5	7	3	6
MLP	80.81 ± 4.75	85.29 ± 6.40	81.89 ± 6.40	74.02 ± 1.90	75.69 ± 2.00	87.16 ± 0.37
GCN [9]	55.14 ± 5.16	51.76 ± 3.06	60.54 ± 5.30	76.50 ± 1.36	88.42 ± 0.50	86.98 ± 1.27
GAT [18]	52.16 ± 6.63	49.41 ± 4.09	61.89 ± 5.05	76.55 ± 1.23	87.30 ± 1.10	86.33 ± 0.48
GraphSAGE [8]	82.43 ± 6.14	81.18 ± 5.56	75.95 ± 5.01	76.04 ± 1.30	88.45 ± 0.50	86.90 ± 1.04
H2GCN [21]	84.86 ± 7.23	87.65 ± 4.89	82.70 ± 5.28	77.11 ± 1.57	89.49 ± 0.38	87.87 ± 1.20
Geom-GCN [13]	66.76 ± 2.72	64.51 ± 3.66	60.54 ± 3.67	78.02 ± 1.15	89.95 ± 0.47	85.35 ± 1.57
LINKX [10]	74.60 ± 8.37	75.49 ± 5.72	77.84 ± 5.81	73.19 ± 0.99	87.86 ± 0.77	84.64 ± 1.13
GGCN [20]	84.86 ± 4.55	86.86 ± 3.29	85.68 ± 6.63	77.14 ± 1.45	89.15 ± 0.37	87.95 ± 1.05
CGNN [19]	71.35 ± 4.05	74.31 ± 7.26	66.22 ± 7.69	76.91 ± 1.81	87.70 ± 0.49	87.10 ± 1.35
MixHop [1]	77.84 ± 7.73	75.88 ± 4.90	73.51 ± 6.34	76.26 ± 1.33	85.31 ± 0.61	87.61 ± 0.85
FSGNN [12]	87.30 ± 5.29	87.84 ± 3.37	85.13 ± 6.07	77.40 ± 1.90	77.40 ± 1.93	87.93 ± 1.00
GPRGNN [5]	78.38 ± 4.36	82.94 ± 4.21	80.27 ± 8.11	77.13 ± 1.67	87.54 ± 0.38	87.95 ± 1.18
Community-HOP	89.46 ± 5.72	89.01 ± 3.84	82.70 ± 3.00	78.30 ± 2.13	89.50 ± 0.47	88.22 ± 1.29

baselines. This section describes the datasets and experimental settings, followed by a comprehensive comparison of the results and a detailed analysis (Table 1).

To evaluate the efficacy of the Community-HOP, we selected six small to medium real-world node classification benchmark datasets: Cora, Cornell, PubMed, Texas, and Wisconsin [14]. A statistical summary of these datasets, including the edge homophily ratio (HOM LEVEL) [21], offers insight into the dataset's heterophily. A lower HOM LEVEL indicates greater heterophily, posing a challenge for vanilla GNN models, which typically perform worse under these conditions.

For the node classification experiments, we utilized the dataset splits provided by [14]. Each split includes 48% of the data for training, 32% for validation, and 20% for testing. The performance metrics are reported as the average accuracy with standard deviation across 10 different splits. All models were trained for a total of 3000 epochs using the Adam optimizer and cross-entropy loss function. To optimize the Community-HOP method, hyperparameter tuning was carried out, focusing on parameters such as learning rate, dropout rate, number of clusters, number of hops, and hidden dimensions. A grid search strategy was used to examine different combinations of these hyperparameters. Detailed information on the hyperparameter settings for each dataset is available at the following link.

Our experimental findings show that Community-HOP markedly exceeds the performance of existing methods in accuracy across four diverse datasets, showcasing its effectiveness and adaptability in node classification tasks with varying degrees of heterophily. The analysis of edge homophily ratios (HOM LEVEL)

emphasizes the difficulties encountered with higher heterophily levels and highlights the improvements offered by Community-HOP over conventional GNNs and other state-of-the-art techniques.

However, our method does not achieve superior performance on the Cornell and PubMed datasets. In the case of Cornell, this limitation can be attributed to difficulties in accurately computing spectral clusters, exacerbated by significant gaps in the dataset's homophily structure. For PubMed, the high volume of nodes and edges impedes our ability to effectively capture the underlying community structure. Consequently, inadequate clustering results in suboptimal performance for community-specific hops.

Our method focuses on community nodes by executing multiple hops within the community and shows promising results, particularly in homophilic environments where neighbors share the same label. This characteristic is advantageous as it aligns with the assumption that nodes within such environments exhibit high intra-community homophily. Notably, our Community-HOP also demonstrates effective performance in heterophilic contexts, suggesting that it can adeptly manage heterophily within communities. This adaptability contributes to its overall improved classification performance across various datasets, showing its potential for broader applicability in diverse graph-based tasks.

6 Conclusion and Future Work

In this study, we introduced a Community-HOP-based GNN model designed to address the challenge of heterophily in graphs. Central to our approach is the Community-Hop method, which leverages community structural information to refine the feature aggregation process within the GNN layers. This technique enhances the relevance of information flow both within and across communities by integrating spectral community detection with an adapted transition matrix for inter-community hops. However, a significant limitation of our approach is its computational cost, particularly for large-scale graphs and the determination of an optimal parameter k. Future work will focus on addressing these challenges by advancing spectral clustering techniques and developing methods for automatic optimization of k.

Acknowledgement. The authors acknowledge the support from the Spanish Government under project PID2022-142516OB-I00.

References

1. Abu-El-Haija, S., et al.: Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning (2019)
2. Arnaiz-Rodríguez, A., Begga, A., Escolano, F., Oliver, N.: Diffwire: inductive graph rewiring via the lovász bound. In: Learning on Graphs Conference, LoG 2022, 9–12 December 2022, Virtual Event. Proceedings of Machine Learning Research (2022)

3. Bianchi, F.M., Grattarola, D., Alippi, C.: Mincut pooling in graph neural networks. CoRR abs/1907.00481 (2019)
4. Cai, C., Wang, Y.: A note on over-smoothing for graph neural networks. CoRR abs/2006.13318 (2020)
5. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. In: International Conference on Learning Representations (2021)
6. Chung, F.R.: Spectral Graph Theory. American Mathematical Society (1997)
7. Deng, S., Ling, S., Strohmer, T.: Strong consistency, graph laplacians, and the stochastic block model. J. Mach. Learn. Res. **22**, 117:1–117:44 (2021)
8. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (2017)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
10. Lim, D., et al.: Large scale learning on non-homophilous graphs: new benchmarks and strong simple methods. In: Advances in Neural Information Processing Systems (2021)
11. von Luxburg, U.: A tutorial on spectral clustering. CoRR abs/0711.0189 (2007)
12. Maurya, S.K., Liu, X., Murata, T.: Improving graph neural networks with simple architecture design (2021)
13. Pei, H., Wei, B., Chang, K.C., Lei, Y., Yang, B.: Geom-GCN: geometric graph convolutional networks. CoRR abs/2002.05287 (2020)
14. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-GCN: geometric graph convolutional networks (2020)
15. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
16. Song, Y., Zhou, C., Wang, X., Lin, Z.: Ordered GNN: ordering message passing to deal with heterophily and over-smoothing. In: The Eleventh International Conference on Learning Representations (2023)
17. Topping, J., Giovanni, F.D., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. In: International Conference on Learning Representations (2022)
18. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
19. Yamamoto, T.: Crystal graph neural networks for data mining in materials science (2019)
20. Yan, Y., Hashemi, M., Swersky, K., Yang, Y., Koutra, D.: Two sides of the same coin: heterophily and oversmoothing in graph convolutional neural networks. CoRR abs/2102.06462 (2021)
21. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Beyond homophily in graph neural networks: current limitations and effective designs. In: Proceedings of the 34th International Conference on Neural Information Processing Systems (2020)

Spatio-Temporal Graph Neural Networks for Water Temperature Modeling

Benjamin Fankhauser[1(✉)], Vidushi Bigler[2], and Kaspar Riesen[1]

[1] Institute of Computer Science, University of Bern, Bern, Switzerland
{benjamin.fankhauser,kaspar.riesen}@unibe.ch
[2] Institute for Optimisation and Data Analysis, Bern University of Applied Sciences, Biel, Switzerland
vidushi.bigler@bfh.ch

Abstract. River water temperature modeling – a time series problem where spatial relations matter – is important to understand our environment. Currently two research directions are present to tackle this problem, viz. Recurrent Neural Networks, namely long short-term memory (LSTM), and Graph-based approaches, which exploit the natural tree structure of rivers. In the present paper, we extend a state-of-the-art LSTM method for water temperature modeling with a Graph Convolutional Network and a Graph Isomorphism Network. This novel combination results in a spatio-temporal neural network which can be applied for node predictions in any graph having nodes with unique identifiers. In the present paper, we apply the novel procedure on the Swiss River Network (a data set with decades of measurements of river temperature and atmospheric variables in Switzerland). In an experimental evaluation we show that the proposed method is robust in convergence and improves the state-of-the-art result by several percentage points in terms of Root Mean Squared Error.

Keywords: River Water Temperature · LSTM · Recurrent Neural Network · GCN · GIN · Graph Neural Network · Spatio Temporal Model

1 Introduction

River water temperature is an important variable in our ecosystem, as a large number of ecological processes are heavily dependent by it. Mainly higher river water temperatures are critical as the hydrological ecosystems are sensitive to raising water temperatures [1]. In the worst case, such a rise results in the extinction of species as well as the deterioration of water quality [2]. The present paper researches novel approaches for water temperature modeling.

An accurate and elaborated modeling of the river water temperature has two benefits. First, with future air temperature projections, one can model river water temperature more precisely and in turn detect river sections with critical

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 31–40, 2025.
https://doi.org/10.1007/978-3-031-80507-3_4

water temperature more precisely. Second, having more sophisticated models allows us to adapt them to more complex problems, such as, for instance, the investigation of the effects of anthropological buildings on the river infrastructure [3,4].

River water temperature modeling is an interesting real world application for machine learning as there are several non linear effects contributing together. Solar radiation, for instance, is absorbed by the river bed or particles in the water, which has a major influence on the temperature. The river bed itself creates friction (which also leads to heat). Snow melt, ground water inflow, city sewage, or rain water also influence the river water temperature [5].

The *Federal Office for the Environment of Switzerland* (FOEN) is running a water temperature monitoring for several decades in many rivers of Switzerland. Additionally, the *Swiss Meteorological Institute* (MeteoSwiss) measures air temperature and other atmospheric variables in the vicinity of these water stations. Recently, data from these water and weather stations in combination with their connectivity has been published as the *Swiss River Network* data set [6]. This data set is unique, yet somehow related to other data sets, e.g. data for rain-fall run-off models [7], or weather parameter prediction [8].

For modeling the water temperature, the use of the air temperature as well as the discharge as predictor variables has been proposed. For instance, physically inspired methods, like Air2Stream [9], show success based on statistical models using these variables. More recently, the application of long short-term memory (LSTM) [10,11] has been proposed to further improve the prediction precision. LSTMs are a special type of Recurrent Neural Network (RNN) and are used to model time series [12]. Also other deep learning architectures, some taking the neighboring relation of water stations into account, provide competitive results [13,14]. In order to model neighboring water stations, the use of graph structures, where the nodes represent water stations and the edges represent relations between stations based on the river sections, has also been proposed [15].

In the present work, we propose to combine one of the latest RNN methods for water temperature modeling [16] with Graph Neural Networks (GNNs). The proposed procedure is particularly designed for node predictions in a node-with-id network, namely a graph where each node has a unique identifier and is thus permutation invariant by definition.

The remainder of this paper is structured as follows. In Sect. 2, we formally introduce the problem of temperature modeling. Additionally, we briefly review the latest RNN methods for water temperature modeling and the GNNs actually used in our procedure. In Sect. 3, we introduce the proposed spatio-temporal neural network suited for a node-with-id network like the Swiss River Network. Section 4 contains a thorough evaluation of the proposed method. Finally, we draw conclusions and propose future research directions in Sect. 5.

2 Related Work

In this work we focus on water temperature modeling based on air temperature. Formally, for a given time series of T air temperatures $at_k^{(1)}, ..., at_k^{(T)}$ at a specific water station indexed k, the goal is to model f_k with

$$f_k(at_k^{(1)}, ..., at_k^{(t)}) = \hat{wt}_k^{(t)}, \quad \forall t \in \{1, \ldots, T\}$$

so that $\hat{wt}_k^{(t)}$ is the predicted water temperature at time step t (with $1 \leq t \leq T$) at water station k.

The method proposed in this paper combines two state-of-the-art directions in water temperature modeling, namely RNNs and GNNs which are reviewed in the next two subsections.

2.1 LSTM Based Water Temperature Modeling

LSTM (a special type of an RNN) has been deployed to many time series problems [12]. The *Station-Specific LSTM* [6,10] for water temperature modeling, for instance, uses one vanilla LSTM for each available water station in order to model f_k. The drawback of this approach is that it results in many models to converge and maintain.

Water temperature modeling LSTMs have recently been extended with an embedding per water station [16], termed LSTM-E. Formally, the LSTM-E method uses the following algorithm:

$$\begin{aligned}
\text{LSTM-E}(at_k^{(t)}, e_k) := LSTM(at_k^{(t)} || e_k) : \\
f^{(t)} &= \sigma(\boldsymbol{W}_f a_k^{(t)} + \boldsymbol{V}_f e_k + \boldsymbol{U}_f h^{(t-1)} + \boldsymbol{b}_f) \\
i^{(t)} &= \sigma(\boldsymbol{W}_i a_k^{(t)} + \boldsymbol{V}_i e_k + \boldsymbol{U}_i h^{(t-1)} + \boldsymbol{b}_i) \\
o^{(t)} &= \sigma(\boldsymbol{W}_o a_k^{(t)} + \boldsymbol{V}_o e_k + \boldsymbol{U}_o h^{(t-1)} + \boldsymbol{b}_o) \\
\tilde{c}^{(t)} &= \theta(\boldsymbol{W}_c a_k^{(t)} + \boldsymbol{V}_c e_k + \boldsymbol{U}_c h^{(t-1)} + \boldsymbol{b}_c) \\
c^{(t)} &= f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \\
h^{(t)} &= o^{(t)} \odot \theta(c^{(t)}) \\
\hat{wt}^{(t)} &= MLP(h^{(t)})
\end{aligned}$$

As in the vanilla LSTM, $\sigma(z)$ denotes the sigmoid activation function and $\theta(z)$ is the tanh function. The two variables h and c are the hidden state and propagated through time. The bold symbols are the learnable network weights, which are shared among all water stations.

Roughly speaking, the LSTM-E corresponds to one single global LSTM where every water station k has an assigned embedding e_k which is in turn learned during training time. It has been shown that this method not only improves the predictive performance but also lowers the trainable parameters by several orders of magnitude compared to station-specific LSTMs [16].

2.2 Graph Neural Networks (GNNs)

In several contributions, it has been shown that combining information of neighboring water stations is beneficial to hydrological models [13,15,17]. In order to make use of the spatial connectivity between water stations, we use the graph introduced in the Swiss River Network [6] which actually corresponds to a node-with-id network.

Formally, a node-with-id network is a graph $G = (V, E)$ where V are the nodes and E the edges. Additionally, there is an identifier function $id : V \rightarrow \mathbb{N}$, which assigns a unique and ordinal identifier to each node on the network. This introduces a deterministic ordering of nodes in the graph. Note, however, that the graph still has an irregular structure, as the degree of each node can vary. Thus, GNNs with message passing mechanisms are suited to this data structure.

The generalized message passing algorithm expects a feature vector $\boldsymbol{x}_i$ at each node $v_i \in V$. In a first step, this feature vector is transformed by a function $g(\boldsymbol{x}_i)$ and then transmitted along the edges of node v_i. In a second step, each node v_i receives $|\mathcal{N}(v_i)|$ incoming messages and aggregates them to $\boldsymbol{z}_i$ (where $\mathcal{N}(v_i)$ refers to the set of neighboring nodes of v_i). In a third step, each node updates its own feature vector using a function $h(\boldsymbol{x}_i, \boldsymbol{z}_i)$, resulting in a new state at each node. The complete message passing process can be applied M-times (increasing the receptive field of each node). By adding an activation function in between executions of message passing we finally obtain a GNN [18].

More formally, Eq. 1 shows the generalized message passing framework for the update step of node v_i with its neighbor nodes $v_j \in \mathcal{N}(v_i)$. The aggregation function *aggr* is usually implemented as mean or sum (yet, other methods are available).

$$\boldsymbol{x}_i^{(m)} = h(\boldsymbol{x}_i^{(m-1)}, aggr_{v_j \in \mathcal{N}(v_i)}(g(\boldsymbol{x}_j^{(m-1)}, \boldsymbol{x}_i^{(m-1)}))) \tag{1}$$

In Eq. 1 we use index m to indicate the m-th execution of message passing. In the present work, we use two state-of-the-art implementations of the message passing framework, viz. *Graph Convolutional Networks* (GCN) [18] and *Graph Isomorphic Networks* (GIN) [19].

– The GCN [18] uses the following implementation of the message passing algorithm (with an edge weight of 1

$$\boldsymbol{x}_i^{(m)} = \underbrace{\Theta^T}_{h} \underbrace{\sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}}}_{aggr} \underbrace{\frac{1}{\sqrt{\hat{d}_j \hat{d}_i}} \boldsymbol{x}_j^{(m-1)}}_{g}$$

$$\hat{d}_i = 1 + deg(v_i)$$

where Θ are the learnable filter weights and the aggregation corresponds to the adjacency matrix with inserted self loops ($deg(v_i)$ refers to the degree of node v_i).

– The GIN [19] improves the expressiveness of GCN by approximating an injective multi set aggregation function using MLPs. In its reduced form it uses the following implementations of the message passing algorithm:

$$\boldsymbol{x}_i^{(m)} = \underbrace{MLP}_{h}\left((1+\epsilon)\boldsymbol{x}_i^{(m-1)} + \underbrace{\sum_{v_j \in \mathcal{N}(v_i)}}_{aggr} \underbrace{\boldsymbol{x}_j^{(m-1)}}_{g}\right)$$

As in the literature proposed we use an MLP during the node update step and ϵ is a free parameter.

3 Spatio-Temporal Nodes-with-Id Network

Major contribution of this paper is that we combine the RNN model LSTM-E (detailed in Sect. 2.1) and both GNN architectures GCN and GIN (detailed in Sect. 2.2) for water temperature modeling. In particular, we propose the following architecture that describes the method at water station k at time step t in the time series for $m \in \{1, \ldots, M\}$ message passing steps. (See also Fig. 1 which illustrates the general architecture of the proposed spatio-temporal network).

$$\boldsymbol{x}_k^{(t,0)} = \text{LSTM-E}(at_k^{(t)}, e_k), \qquad \text{stacked } D\text{-times} \tag{2}$$

$$\boldsymbol{x}_k^{(t,m)} = \text{GNN}(X^{(t,m-1)}, E), \qquad \text{repeated } M\text{-times} \tag{3}$$

$$\hat{wt}_k^{(t)} = \text{MLP}(\boldsymbol{x}_k^{(t,M)}) \tag{4}$$

First (in Eq. 2), we apply the LSTM-E Method at each node of the network in order to transform the air temperature into a hidden state which has high predictive power. In Eq. 2 at_k is the air temperature at water station k and e_k refers to the station specific embedding. In a second step (in Eq. 3), we apply the message passing algorithm, so that the network can take information of its surrounding nodes into account and further refine the hidden state. This message passing step takes as input all the hidden states $\boldsymbol{x}_i \in X$ of all water stations and edges E of the Swiss River Network graph. In a last step (in Eq. 4), the hidden state is projected to a water temperature using linear transformations.

Note that the LSTM-E, GNN, MLP are all global models with shared parameters among all water stations, and only the embedding e_k is station-specific.

4 Experimental Evaluation

4.1 Experimental Setup

For our evaluation, we use the catchment area of the river Rhine of the Swiss River Network dataset G_{2010} [6]. This results in one connected component with

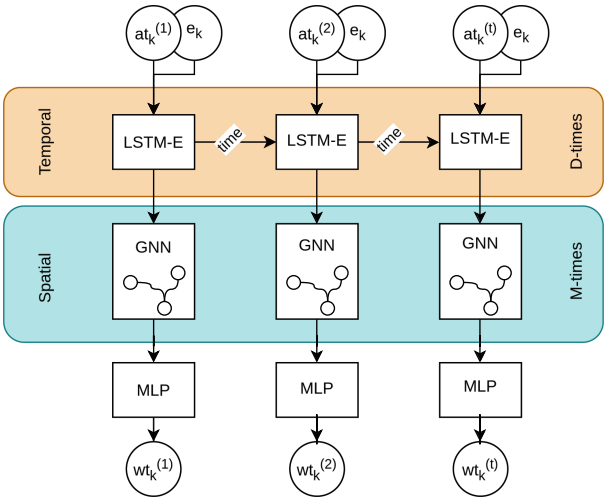


Fig. 1. The proposed method at water station k within a node-with-id network. The air temperature at_k is transformed using the LSTM-E Method [16], an embedding based LSTM. The hidden state is then collected in the graph structure of the Swiss River Network [6]. The GNN refines the hidden state using message passing among neighboring nodes. At the latest stage, an MLP predicts the final water temperature $\hat{wt}_k^{(t)}$ at time step t.

50 water stations and corresponding air temperature from the years 2010 to the end of 2020. We use daily averaged temperatures.[1]

Data from the years 2010 to 2018 are used as training set. From these years we use a 90/10% training/validation split. The validation set is used for model selection only, as we run an exhaustive grid search over the hyperparameters. Data from the two years 2019 and 2020 are used as test set and are not considered during training time. In this work we only report metrics obtained on the test set.

We compare the proposed method to two state-of-the-art systems, viz. the station-specific LSTM [10] and the embedding based LSTM [16].

- The *Station-Specific LSTM* uses one vanilla LSTM for each water station [10]. Each station-specific LSTM is trained in isolation and a grid search is run to find the best performing hyperparameters for each water station [6].
- The *Embedding LSTM* corresponds to the state-of-the-art RNN LSTM-E and does not use neighboring information, yet learned embeddings per station [16].

To compare and asses the predictive performance, we report the widely used *Root Mean Squared Error* (RMSE), the *Mean Average Error* (MAE) and the *Nash-Sutcliffe model Efficiency Coefficient* (NSE) as defined in Table 1. For both

[1] Code and data is made available under https://swiss-river-network.github.io.

RMSE and MAE, values closer to 0 and for the NSE values closer to 1 indicate better model performance.

Table 1. The three metrics used for evaluation of the methods ($wt^{(t)}$ is the ground truth value for the water temperature and $\hat{wt}^{(t)}$ refers to our prediction).

RMSE	MAE	NSE
$\sqrt{\frac{1}{T}\sum_{t=1}^{T}(wt^{(t)} - \hat{wt}^{(t)})^2}$	$\frac{1}{T}\sum_{t=1}^{T}\lvert wt^{(t)} - \hat{wt}^{(t)}\rvert$	$1 - \frac{\sum_{t=1}^{T}(wt^{(t)} - \hat{wt}^{(t)})^2}{\sum_{t=1}^{T}(wt^{(t)} - \bar{wt})^2}$

4.2 Training and Hyperparameter Tuning

As proposed in the LSTM-E method [16], we optimize the embeddings e_k during training time. We use a grid search on the learning rate, the embedding dimension, the amount of stacked LSTMs D, the amount of message passing steps M, the width of the hidden space used in the LSTM as well the GNN. Table 2 shows the search space of the grid search for all five parameters. As gradient descent based optimizer we use the Adam optimizer [20] and model selection is based on the lowest RMSE on the validation set.

Table 2. Space of the grid search among all embedding and graph based methods.

Hyperparameter	Values	
	Min	Max
Learning rate	0.005	0.01
Embedding dimensions	5	10
Stacked LSTMs D	1	2
Message passing steps M	1	2
Hidden space width	8	16

4.3 Empirical Results

We report the metrics (RMSE, MAE, and NSE) on the hold-out test set in Table 3. The test set contains 50 water stations and we report the average values of each metric over all 50 stations. To encounter random artifacts, we rerun the grid search several times and report the standard deviation in brackets.

The proposed method that combines the LSTM-E with GNNs shows an improvement in all metrics for both instances of the GNN. If we compare our approach with the station-specific LSTM, we observe an improvement in the RMSE

and MAE of around 10% (while the NSE is only slightly improved). When comparing with the Embedding LSTM, we observe only slight improvements (for the RMSE we see an absolute improvement of 0.01 and 0.02 for GCN and GIN, respectively) and for the MAE an absolute improvement of 0.01 is observed for both models). What is noticeable, however, is the significant decrease in the standard deviation for the GIN model. Compared to the Embedding LSTM, these deviations are approximately seven times, five times and 20 times smaller than for the Embedding LSTM (according to RMSE, MAE and NSE), which corresponds to a massive improvement in the robustness of the model.

Table 3. The average metrics reported on the hold-out test set over 50 water stations. Brackets indicate the standard deviation after multiple runs of the grid search. For each metric the best result is marked in bold.

	Method	Metric		
		RMSE	MAE	NSE
Reference	Station-Specific LSTMs [10]	0.80	0.62	0.95
	Embedding LSTM [16]	0.74 (± 0.007)	0.57 (± 0.005)	0.96 (± 0.002)
Ours	Spatio Temporal (GCN)	0.73 (± 0.012)	**0.56** (± 0.009)	**0.97** (± 0.001)
	Spatio Temporal (GIN)	**0.72** (± 0.001)	**0.56** (± 0.001)	**0.97** (± 0.00001)

5 Conclusion

Water temperature of rivers plays an important role in future climate change and thus quite an effort is made in monitoring and prediction. However, further research is needed due to two reasons. First, to improve the predictive performance of the current models, and second to flexibly adapt the models to future tasks. In the present paper, we propose to use a state-of-the-art RNN in water temperature modeling and extend it with a graph based network using the message passing mechanism. This leads to a spatio-temporal neural network.

In an empirical evaluation we demonstrate that the use of the spatial information improves the modeling performance in three different metrics, setting a new state-of-the-art baseline in water temperature modeling. We also observe a beneficial improvement in robustness of model convergence.

Major goal of this paper is to improve the accuracy of water temperature modeling. Yet, working with graphs enables more flexible approaches than collecting data for a decade until an isolated LSTM converges. Moreover, graph modeling opens a variety of other research directions. The proposed framework might be suited, for instance, to model dynamic changes on the underlying river network (for example, if a new water station is built or a short-term measurement is made). Some parts of Switzerland do have more water stations, this means we can also nest finer resolutions into the current graph, or explore the

relation between neighboring stations in more detail. Last but not least, the proposed method makes use of a node-with-id network, and thus it easily generalizes to other real world applications like transportation, computer, or social media networks.

Acknowledgments.. This project is supported by the Swiss National Science Foundation (SNSF) Grant Nr. PT00P2 206252. Data are kindly provided by the Federal Office for the Environment and MeteoSwiss. Calculations were performed on UBELIX (https://www.id.unibe.ch/hpc), the HPC cluster at the University of Bern.

References

1. Dahlke, F.T., Wohlrab, S., Butzin, M., Pörtner, H.O.: Thermal bottlenecks in the life cycle define climate vulnerability of fish. Science **369**(6499), 65–70 (2020)
2. Caissie, D.: The thermal regime of rivers: a review. Freshw. Biol. **51**(8), 1389–1406 (2006)
3. Reid, P.C., et al.: Global impacts of the 1980s regime shift. Glob. Change Biol. **22**(2), 682–703 (2016)
4. Woolway, R.I., Dokulil, M.T., Marszelewski, W., Schmid, M., Bouffard, D., Merchant, C.J.: Warming of central European lakes and their response to the 1980s climate regime shift. Clim. Change **142**, 505–520 (2017)
5. Piccolroaz, S., Calamita, E., Majone, B., Gallice, A., Siviglia, A., Toffolon, M.: Prediction of river water temperature: a comparison between a new family of hybrid models and statistical approaches. Hydrol. Process. **30**(21), 3901–3917 (2016)
6. Fankhauser, B., Bigler, V., Riesen, K.: Graph-based deep learning on the swiss river network. In: International Workshop on Graph-Based Representations in Pattern Recognition, pp. 172–181. Springer (2023)
7. Kratzert, F., Klotz, D., Brenner, C., Schulz, K., Herrnegger, M.: Rainfall-runoff modelling using long short-term memory (LSTM) networks. Hydrol. Earth Syst. Sci. **22**(11), 6005–6022 (2018)
8. Zanetta, F., Nerini, D., Beucler, T., Liniger, M.A.: Physics-constrained deep learning postprocessing of temperature and humidity. Artif. Intell. Earth Syst. **2**(4), e220089 (2023)
9. Toffolon, M., Piccolroaz, S.: A hybrid model for river water temperature as a function of air temperature and discharge. Environ. Res. Lett. **10**(11), 114011 (2015)
10. Qiu, R., et al.: River water temperature forecasting using a deep learning method. J. Hydrol. **595**, 126016 (2021)
11. Moshe, Z., Metzger, A., Elidan, G., Kratzert, F., Nevo, S., El-Yaniv, R.: Hydronets: leveraging river structure for hydrologic modeling. arXiv preprint arXiv:2007.00595 (2020)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
13. Jia, X., et al.: Physics-guided recurrent graph model for predicting flow and temperature in river networks. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), pp. 612–620. SIAM (2021)
14. Chen, S., Zwart, J.A., Jia, X.: Physics-guided graph meta learning for predicting water temperature and streamflow in stream networks. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2752–2761 (2022)

15. Zhao, Q., et al.: Joint spatial and temporal modeling for hydrological prediction. IEEE Access **8**, 78492–78503 (2020)
16. Fankhauser, B., Bigler, V., Riesen, K.: Leveraging LSTM embeddings for river water temperature modeling (in review)
17. Fankhauser, B., Bigler, V., Riesen, K.: Impute water temperature in the swiss river network using LSTMs. In: International Conference on Pattern Recognition Applications and Methods, pp. 732–738. Scitepress (2024)
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017, Conference Track Proceedings. OpenReview.net (2017). https://openreview.net/forum?id=SJU4ayYgl
19. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. OpenReview.net (2019). https://openreview.net/forum?id=ryGs6iA5Km
20. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint (2014)

Enhancing IoT Network Security with Graph Neural Networks for Node Anomaly Detection

Vincenzo Carletti(✉), Pasquale Foggia, Francesco Rosa, and Mario Vento

Department of Information Engineering, Electrical Engineering and Applied Mathematics, University of Salerno, Fisciano, Italy
{vcarletti,pfoggia,frosa,mvento}@unisa.it
https://mivia.unisa.it

Abstract. The widespread deployment of Internet of Things (IoT) devices in homes, industries, and public spaces presents significant cybersecurity challenges, particularly due to their limited computational capabilities and often insecure configurations. Detecting infected devices through network analysis presents a significant challenge given the diversity of network protocols and behaviors. Traditional methods, reliant on packet statistics or binary signatures, show their limits in these complex environments. Recent advancements in machine learning and deep learning offer promising alternatives, also through the use of graph-based representations that capture network topology and facilitate the detection of complex attack patterns.

This paper presents a comprehensive analysis in a realistic setup of two state-of-the-art Graph Neural Networks (GNNs) designed for node anomaly detection, applied to a large-scale dataset of IoT network traffic. The dataset, comprising over 240,000 graphs extracted from IoT23, IoTID20, and IoT-Traces, includes both benign and malicious communications. In the analysis we take into account the impact of varying snapshot durations and graph-based representations on the performance achieved by the GNNs.

The results suggest that using a state-of-the-art graph autoencoder (DOMINANT) with a computationally efficient representation (TDG) is the best trade-off among the considered constraints and variables.

Keywords: Network Anomaly Detection · IoT Networks · GraphNeural Networks

1 Introduction

Nowadays, IoT devices are ubiquitous in domestic and industrial environments, serving a wide range of purposes. These devices, which include everything from simple lamps and thermostats to electrical appliances controlled by smart voice

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 41–51, 2025.
https://doi.org/10.1007/978-3-031-80507-3_5

assistants, are constantly connected and thus exposed to several kind of threats. Indeed, these devices are among the most preferred targets for malicious users due to the limited computational capabilities of typical IoT devices, that prevent from the adoption of conventional countermeasures like antivirus software, and the inexperience of most users, which leads to weak passwords and insecure configurations. For instance, attackers exploit these weaknesses to set up botnets and launch distributed attacks, such as Distributed Denial of Service (DDoS), against other systems across the internet. [5,19]

One of the most effective approaches is to detect infected IoT devices by analyzing their communication patterns over the network. However, the wide range of network protocols and behaviors in IoT environments makes the task challenging, particularly when using traditional approaches that rely on analyzing statistics from network connections, packet header information, or binary signatures in the packet payload [14]. With the purpose to overcome the limitation of these approaches, in recent years, machine learning and deep learning methods have gained interest, thanks to their ability to handle more complex network scenarios and to generalize with respect to unseen communication patterns [1,13,14,16].

Different methods proposed in the literature are based on vector representations of network traffic obtained by extracting statistical features from network flows, which are sequences of packets sharing the same transport protocol, source and destination address and port. The main limitation of this representation is that it does not directly account for the network topology. Consequently, these methods can only detect network patterns arising from the flows of hosts that are directly communicating with each other, ignoring patterns that may involve groups of hosts or those communicating through intermediary hosts, as seen in attacks using lateral movements [4,12]. Therefore, graph-based approaches and representations have been proposed in [6,12,23,25] to overcome this limitation.

Several works in the state-of-the-art that face the detection and classification of network attacks reports remarkable results [25], but the proposed experiments only consider old datasets without IoT devices and are based on the assumption that malicious communication patterns related to attacks are known in advance. As discussed in [4] the latter is not a realistic hypothesis, indeed, in most of the networks a traffic analysis system (such as an Intrusion Detection System) will face with a large quantity heterogeneous devices and protocols, thus, it is impossible to know all the potential threats and their evolution within a network in advance. On the other hand, it is reasonable to expect that the system can collect a large volume of normal traffic. Under this hypothesis, we face the problem as a *node anomaly detection task*, where the aim is to identify nodes whose attributes and structure deviates from that belonging to a normal traffic model learned by the systems.

In this paper we present an extensive analysis on the exploitation of two state-of-the-art Graph Neural Network designed for node anomaly detection on a very large dataset of graphs extracted from three recent dataset collected from real IoT network communications and attacks, i.e. IoT23 [17], IoTID20 [20], and

IoT-Traces [18]. Each graph represents the network communications among the hosts in a specific time window. To evaluate the impact of the window size we also have built four versions of the dataset considering interval sizes of 2 m30 s, 2 m00 s, 1 m30 s, and 1 m00 s respectively, composing a dataset that comprises over 240,000 graphs ranging from 8 to 3,000 nodes.

The overall paper is organized as follows: in Sect. 2 we introduce the most recent graph-based representations of network traffic and the GNN proposed for node anomaly detection, in Sect. 3 we describe the experimental setup, the dataset and the procedure proposed to extract the graphs from network traffic, in Sect. 4 we discuss the results of the proposed analysis, and finally in Sect. 5 we provide the final comments and the future works.

2 Related Works

2.1 Graph-Based Representations of Network Traffic

Graph-based representations of network traffic have been proposed in [6,23–25]. Among them, the most popular representations are: *Similarity Graphs* (SG) [6], *Traffic Trajectory Graphs* (TTG) [23,24], and *Extended Traffic Dispersion Graphs* (e-TGD) [25].

The process to generate the graph, for all the representations, begin by extracting vector representations of network flows, which are then used as initial feature vectors assigned to nodes in the graph. These vectors are typically obtained by dividing the overall communication into time snapshots of a given duration. For each snapshot, the raw packets in the network flows within that interval are processed to extract statistical measures such as the average packet size and flow duration, as well as non-statistical information like IP and MAC addresses. In this paper, we utilized NFStream [2], a publicly available tool, to extract the vector representation.

In SG, nodes represent communication flows, and their attributes are the corresponding feature vectors. The similarity among all nodes is computed using the cosine metric. For each node, the k closest neighbors are connected via undirected edges.

Similar to SG, in TTG, each node represents a network flow. However, nodes are connected by undirected edges if the corresponding network flows share the IP address (e.g., the source of the first flow is the same device as the destination of the second flow).

Differently form the latter, in TDG and e-TDG each node is transport-level endpoints that communicate over the network identified by the couple IP address and port, while the edges are the network flows related to the endpoint. Therefore, the flow features vectors are initially attached to edges instead of nodes, then, once the graph is built, the label of the latter are computed by averaging the feature vectors of the edges. The e-TDG representation is, then, obtained starting from the TDG and augmenting the node feature vector with graph metrics such as degree, centrality, betweenness, closeness, and eccentricity.

2.2 Node Anomaly Detection Methods

When considering the anomaly detection problem, traditional approaches are based on well-known machine learning methods such as k-NN [8], OC-SVM [15], and Isolation Forests [9]. Although these methods are easy to use and do not require complex, time-consuming training procedures, they often struggle with generalization, particularly when applied to different networks that exhibit varying communication patterns and attacks. For this reason, researchers have shifted their focus to deep learning methods such as autoencoders or graph neural networks. In particular, these latter naturally model communication flows using nodes (e.g., communicating entities) and edges (e.g., communication links between entities). Generally, all approaches share the same base concepts: learning a function that assigns an *anomaly score* to each node in the graph. The main differences among the methods lie in how the node embedding is learned and how the score is evaluated based on the computed node representation.

In [7], the authors propose DOMINANT, an encoder-decoder architecture that computes the anomaly score by leveraging reconstruction loss on both the graph structure and the node attributes. DOMINANT consists of two main components: an encoder and a decoder. The encoder takes the original graph, in terms of adjacency matrix and node features, and produces an encoded representation for each node. The decoder then uses the encoded representation of the graph to reconstruct both the adjacency and the node feature matrices. The underlying idea is that if a node deviates from the training distribution, it will exhibit a high reconstruction error. The latter is directly used as an anomaly score; indeed, by comparing the error with a properly selected threshold, a node can be classified as either anomalous (if the anomaly score exceeds the threshold) or normal (if the anomaly score is below the threshold). The value of the threshold can be adjusted during the validation process by setting the maximum allowable number of false detections.

A different approach is proposed in [21], where the authors introduce OC-GNN (One-Class Graph Neural Network). This method is a variant of OC-SVM [15], specifically designed for one-class classification. In OC-GNN, a Graph Neural Network (GNN) is used as the backbone to build node representations. These representations are then used to learn the minimum volume hypersphere that contains the embeddings of normal nodes. Nodes outside this hypersphere are considered abnormal. The anomaly score is the distance of the node embeddings from the sphere's centroid; if this distance exceeds the hypersphere's radius, the node is considered anomalous, otherwise, it is not.

Other approaches leverage contrastive learning to learn discriminative representations that maximize the distance between positive and negative samples in the embedding space [10]. Notable methods in this category include CoLA [11] and CONAD [22]. CoLA assumes that anomalies in attributed graphs appear as discrepancies between a node and its neighbors. It defines the embedding of a node's neighborhood subgraph as a positive sample and generates a negative sample from a random subgraph excluding the target node. Conversely, CONAD introduces an augmented graph G_{ano} with synthetically created anoma-

lous nodes. This graph is used to train the encoder with a contrastive loss, enhancing the similarity of embeddings for normal nodes while increasing dissimilarity for anomalous nodes in G_{ano}.

In preliminary work such as [4], a review of the presented models was conducted. Specifically, [4] focused on identifying both the best representation and the best method. Differently, this work aims to find the optimal trade-off between snapshot duration, representation complexity, and the overall performance, comparing these factors across different methods.

3 Experiments

As introduced in Sect. 1, the purpose of the proposed analysis is to asses the effectiveness of state-of-the-art GNNs across different datasets and under realistic working hypothesis where we have trained and validated the method using exclusively fully benign graphs and we have constrained them to ensure a false positive rate below the 1% during the validation. Additionally, since the duration of the snapshot may affect the performance of the GNNs we have extracted the graphs considering four different time durations. More details about the dataset are provided in Sect. 3.1.

Among the different methods introduced briefly in Sect. 2, our analysis focuses on DOMINANT and OC-GNN because, unlike contrastive learning methods, they do not assume to have knowledge about malign nodes and patterns and do not require adding artificially created anomalies. The GNNs have been trained in an inductive setting, which differs from the transductive setting used in the original papers in which they have been proposed. For the sake of completeness, we trained the models for up to 300 epochs, with early stopping after 20 epochs. We used the AdamW optimizer, with a learning rate of $1e^{-4}$ for DOMINANT and $3e{-}4$ for OC-GNN.

As for the graph-based representations, since we aim to identify infected devices (and processes) through their traffic patterns, we have chosen TDG and e-TDG because they are the most suitable for our purpose. In general, TDG can be obtained from raw network traffic very efficiently, while e-TDG can be computationally expensive due to the structural features that need to be computed. Therefore, in the analysis we have also evaluated if the computational overhead can provide advantages to the task at hand.

3.1 Dataset

As previously mentioned, the analysis has been conducted using three recent publicly available datasets of IoT network traffic: IoT23 [17], IoTID20 [20], and IoT-Traces [18].

The IoT23 dataset consists of both benign and malicious traffic across 23 different scenarios: 3 benign and 20 malicious. While the benign traffic has been generated by well-known home devices, the malicious one has been simulated using a Raspberry Pi and comprises different attacks, such as DDoS and botnets.

The captures of each scenario typically last 24 h, except when packet generation accelerates excessively. IoTID20 includes traffic from smart home devices, smartphones, and laptops. It includes 9 types of attacks, some of which are not present in IoT23, including various DoS and DDoS attacks, ARP spoofing, and scanning, providing a comprehensive view of potential security threats in a smart home network. Finally, IoT-Traces is a dataset containing only benign traffic from 28 IoT and non-IoT devices, including smartphones, tablets, and laptops, collected over a span of 28 weeks.

We have used the benign graphs extracted from IoT23 dataset for training, validation and testing; while the graphs with malign samples of IoT23 and all the graphs extracted from IoTID20 and IoT-Traces have been used exclusively for testing. In particular, the last two dataset have been used to evaluate the GNNs' capability to generalize across different network scenarios.

In our analysis, we have considered four different snapshot durations: 2 m30 s, 2 m, 1 m30 s, and 1 m. For the sake of clarity, for each duration we had to rerun the graph generation procedure from scratch since this parameters affect the flows falling in a given time snapshot. Therefore, we obtained a different number of samples in the training, validation and test sets for each duration.

Table 1 shows details about the number of samples belonging to the different sets. It is important to point out that the number of graphs reported in the table are the same for both the representations since the e-TDG have been extracted starting from the original TDG by adding the extended structural features. This allowed us to collect experimental results that are comparable between the graph-based representations.

In conclusion, for clarity we provide an idea of the time involved in generating the IoT23 graph dataset, consider the following: for a snapshot duration of 2 m30 s, the e-TDG representation requires approximately 4.88 h to compute, whereas the TDG representation takes only 57 s. Similarly, for a 1 m00 s snapshot, the e-TDG representation takes 2 h, while the TDG representation requires just 54 s.

For more details about the dataset and to download it, please visit the webpage [3].

4 Results

In Table 2 we report the results on graphs with only benign nodes, that have been extracted from scenarios without infected devices, while in Table 3 we show how the GNNs performs on the general test set with scenarios that include both benign and malicious network communications.

The first evidence is that, in most cases, DOMINANT outperforms OC-GNN in both the test sets and shows the most balanced performance across different snapshot durations and graph-based representations. It achieves the highest accuracy on the benign test set (e.g., 0.999 on IoT23 and 0.976 on IoT Traces) and very high F1-score values on scenarios with malign samples (e.g., 0.988 on IoT23 and 0.939 on IoTID20). In terms of generalization capability, DOMINANT also shows the best performance in both the test sets. Although OC-GNN

Table 1. Number of graphs (# Graphs) and nodes (# Nodes) for the Training (Train.), Validation (Val) and Test sets. The Training and Validation sets contain exclusively benign samples. The ratio (Ratio) between number of benign and malign samples refers to the Test set.

		# Graphs			# Nodes			
		Train	Val	Test	Train	Val	Test	Ratio
2 m30 s	IoT23	664	169	5392	4886	1517	4008133	0.134
	IoTID20	-	-	70	-	-	124469	0.180
	IoT Traces	-	-	32458	-	-	3124487	-
2 m00 s	IoT23	521	134	6937	3936	1341	4075521	0.133
	IoTID20	-	-	85	-	-	124749	0.168
	IoT Traces	-	-	40573	-	-	3366134	-
1 m30 s	IoT23	622	159	9035	4086	1344	4147394	0.131
	IoTID20	-	-	104	-	-	125042	0.185
	IoT Traces	-	-	54095	-	-	3444113	-
1 m00 s	IoT23	402	104	13556	2498	720	4229430	0.130
	IoTID20	-	-	142	-	-	125372	0.187
	IoT Traces	-	-	81131	-	-	3543945	-

achieves the best scores on IoTID20 for the general test set, the improvement over DOMINANT is negligible in many cases. Furthermore, if we analyze the accuracy of OC-GNN on IoT Traces and IoTID20 over the benign test set it is evident the lower capability to generalize with respect to DOMINANT.

Regarding the representation, the additional features of the e-TDG can lead to an improvement in the performance of the GNNs. OC-GNN generally performs better with e-TDG, likely because the additional features may help the GNN to obtain node embeddings more effective to build the hypersphere around the benign samples. In the case of DOMINANT, it typically achieves better results with TDG; however, the performance difference between the two representations is often negligible. Moreover, using e-TDG, DOMINANT generalizes better on the IoTID20 datasets in all scenarios. Therefore, for OC-GNN, e-TDG is the best representation. However, when considering DOMINANT, we can choose between TDG and e-TDG to accommodate additional constraints, such as the need for real-time processing.

Finally, analyzing the snapshot duration, we can note that it do not significantly affect the performance of DOMINANT, although a shorter duration seems to improve the generalization capability of this GNN. Indeed, the lower is the duration the higher is the generalization capability on the benign test set across all the dataset. A similar trend is observed in the general test set, particularly on the IoTID20 dataset. Differently, OC-GNN is more sensitive to variations of the snapshot duration, which can significantly affects its generalization capabil-

Table 2. Accuracy achieved by the GNNs on benign samples in the test set. The results are arranged according to the four snapshot durations. For each duration and dataset we have highlighted the best result.

Model	Repr	Set	2 m30 s	2 m00 s	1 m30 s	1 m00 s
DOMINANT	TDG	IoT23	**0.986**	**0.992**	**0.990**	**0.999**
		IoTID20	0.267	0.427	0.358	0.558
		IoT Traces	**0.945**	**0.967**	**0.956**	**0.957**
	e-TDG	IoT23	0.970	0.970	0.976	0.950
		IoTID20	**0.705**	**0.559**	0.533	**0.669**
		IoT Traces	0.766	0.930	0.868	0.917
OC-GNN	TDG	IoT23	0.899	0.908	0.939	0.824
		IoTID20	0.433	0.530	0.486	0.439
		IoT Traces	0.843	0.812	0.819	0.695
	e-TDG	IoT23	0.586	0.838	0.847	0.977
		IoTID20	0.565	0.509	**0.713**	0.376
		IoT Traces	0.604	0.512	0.644	0.879

Table 3. Performance of the GNNs on the general test set of IoT23 and IoTID20 in terms of Accuracy (Acc), Precision (Prec), Recall (Rec) and F1-score (F1). The results are arranged according to the four snapshot durations. For each duration and dataset we have highlighted the best result.

Model	Repr	Set	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
			2 m30 s				2 m00 s			
DOMINANT	TDG	IoT23	0.978	0.989	0.985	**0.987**	0.978	0.990	0.985	**0.988**
		IoTID20	0.772	0.880	0.846	0.863	0.794	0.874	0.885	0.879
	e-TDG	IoT23	0.880	0.889	0.988	0.936	0.882	0.891	0.988	0.937
		IoTID20	0.902	0.900	0.994	0.945	0.892	0.890	0.996	0.940
OC-GNN	TDG	IoT23	0.887	0.889	0.996	0.940	0.886	0.892	0.991	0.939
		IoTID20	0.806	0.891	0.878	0.885	0.249	0.708	0.190	0.300
	e-TDG	IoT23	0.882	0.885	0.995	0.937	0.230	0.676	0.247	0.361
		IoTID20	0.919	0.915	0.997	**0.954**	0.913	0.908	0.998	**0.951**
			1 m30 s				1 m00 s			
DOMINANT	TDG	IoT23	0.981	0.990	0.989	**0.989**	0.874	0.998	0.859	0.923
		IoTID20	0.735	0.857	0.823	0.840	0.893	0.900	0.982	**0.939**
	e-TDG	IoT23	0.883	0.892	0.988	0.937	0.780	0.887	0.861	0.874
		IoTID20	0.888	0.886	0.996	0.937	0.906	0.904	0.993	0.947
OC-GNN	TDG	IoT23	0.776	0.886	0.857	0.871	0.890	0.894	0.994	**0.941**
		IoTID20	0.600	0.865	0.623	0.724	0.393	0.794	0.377	0.511
	e-TDG	IoT23	0.779	0.885	0.861	0.873	0.783	0.886	0.865	0.876
		IoTID20	0.934	0.931	0.996	**0.962**	0.938	0.936	0.994	**0.964**

ity. This is evident from the drop in accuracy and F1-score on IoTID20 in the general test set with durations of 2m00s and 1m00s.

To conclude, we do not observe a significant advantage in using a long snapshot duration such as at 2 m30 s. Instead, we achieve better or at least comparable results with shorter snapshots. Therefore, the choice of the duration, as for the representations, can be guided also by other requirements, such as to have the highest throughput possible. Indeed, while considering Intrusion Detection Systems (IDS) the throughput can significantly affect the performance of the network under analysis.

5 Conclusions

In this paper, we faced the problem of detecting malicious traffic in IoT networks as a node anomaly detection problem in a realistic setup. To this purpose, we have selected two state-of-the-art GNNs and adapted the training procedure to fit for the chosen structural representations and the specificity of the graphs obtained. The analysis was performed on a large set of graphs extracted from three recent datasets of IoT network traffic that include both benign and malign communication among the devices. We conducted a comparison by imposing the false positive rate to be lower that 1% during the training and analyzing two relevant variables that may affect the performance of a system: the duration of the snapshots and the graph-based representation.

The results of our experiments suggest that good trade-off among all the considered constraints and variables can be achieved by using a state-of-the-art graph autoencoder (DOMINANT), with a computationally efficient representation (TDG).

References

1. Abbasi, M., Shahraki, A., Taherkordi, A.: Deep learning for network traffic monitoring and analysis (NTMA): a survey. Comput. Commun. **170**, 19–41 (2021). https://doi.org/10.1016/j.comcom.2021.01.021
2. Aouini, Z., Pekar, A.: Nfstream: a flexible network data analysis framework. Comput. Netw. **204**, 108719 (2022)
3. Carletti, V., Foggia, P., Rosa, F., Vento, M.: IoT network analysis. https://mivia.unisa.it/datasets/iot-network-analysis/
4. Carletti, V., Foggia, P., Vento, M.: Detecting abnormal communication patterns in IoT networks using graph neural networks. In: International Workshop on Graph-Based Representations in Pattern Recognition, pp. 127–138. Springer (2023)
5. Carletti, V., Saggese, A., Foggia, P., Greco, A., Vento, M.: Machine learning methodologies for preventing malware obfuscation, pp. 99–116. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-21940-5_6
6. Deng, A., Hooi, B.: Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4027–4035 (2021)

7. Ding, K., Li, J., Bhanushali, R., Liu, H.: Deep anomaly detection on attributed networks. In: Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 594–602. SIAM (2019)
8. Fix, E., Hodges, J.L.: Discriminatory analysis. nonparametric discrimination: consistency properties. Int. Stat. Rev./Revue Internationale de Statistique **57**(3), 238–247 (1989)
9. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422. IEEE (2008)
10. Liu, X., et al.: Self-supervised learning: generative or contrastive. IEEE Trans. Knowl. Data Eng. **35**(1), 857–876 (2021)
11. Liu, Y., Li, Z., Pan, S., Gong, C., Zhou, C., Karypis, G.: Anomaly detection on attributed networks via contrastive self-supervised learning. IEEE Trans. Neural Netw. Learn. Syst. **33**(6), 2378–2392 (2021)
12. Lo, W.W., Layeghy, S., Sarhan, M., Gallagher, M., Portmann, M.: E-graphsage: a graph neural network based intrusion detection system for IoT. In: 2022 IEEE/IFIP Network Operations and Management Symposium, NOMS 2022, pp. 1–9. IEEE (2022)
13. Lotfollahi, M., Siavoshani, M.J., Zade, R.S.H., Saberian, M.: Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft. Comput. **24**(3), 1999–2012 (2019). https://doi.org/10.1007/s00500-019-04030-2
14. Macas, M., Wu, C., Fuertes, W.: A survey on deep learning for cybersecurity: progress, challenges, and opportunities. Comput. Netw. **212**, 109032 (2022). https://doi.org/10.1016/j.comnet.2022.109032
15. Müller, K.R., Mika, S., Tsuda, K., Schölkopf, K.: An introduction to kernel-based learning algorithms. In: Handbook of Neural Network Signal Processing, pp. 4–1. CRC Press (2018)
16. Pacheco, F., Exposito, E., Gineste, M., Baudoin, C., Aguilar, J.: Towards the deployment of machine learning solutions in network traffic classification: a systematic survey. IEEE Commun. Surv. Tutor. **21**(2), 1988–2014 (2019). https://doi.org/10.1109/COMST.2018.2883147
17. Parmisano, A., Garcia, S., Erquiaga, M.J.: A labeled dataset with malicious and benign IoT network traffic. Stratosphere Laboratory, Praha, Czech Republic (2020)
18. Sivanathan, A., et al.: Classifying IoT devices in smart environments using network traffic characteristics. IEEE Trans. Mob. Comput. **18**(8), 1745–1759 (2019). https://doi.org/10.1109/TMC.2018.2866249
19. Swessi, D., Idoudi, H.: A survey on Internet-of-Things security: threats and emerging countermeasures. Wireless Pers. Commun. (2022). https://doi.org/10.1007/s11277-021-09420-0
20. Ullah, I., Mahmoud, Q.H.: A scheme for generating a dataset for anomalous activity detection in IoT networks. In: Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, 13–15 May 2020, Proceedings 33, pp. 508–520. Springer (2020)
21. Wang, X., Jin, B., Du, Y., Cui, P., Tan, Y., Yang, Y.: One-class graph neural networks for anomaly detection in attributed networks. Neural Comput. Appl. **33**, 12073–12085 (2021)
22. Xu, Z., Huang, X., Zhao, Y., Dong, Y., Li, J.: Contrastive attributed network anomaly detection with data augmentation. In: Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, 16–19 May 2022, Proceedings, Part II, pp. 444–457. Springer (2022)

23. Zheng, J., Li, D.: GCN-TC: combining trace graph with statistical features for network traffic classification. In: 2019 IEEE International Conference on Communications (ICC), ICC 2019, pp. 1–6. IEEE (2019)
24. Zheng, J., Zeng, Z., Feng, T.: GCN-ETA: high-efficiency encrypted malicious traffic detection. Secur. Commun. Netw. **2022**, 1–11 (2022)
25. Zola, F., Segurola-Gil, L., Bruse, J.L., Galar, M., Orduna-Urrutia, R.: Network traffic analysis through node behaviour classification: a graph-based approach with temporal dissection and data-level preprocessing. Comput. Secur. **115**, 102632 (2022)

LSTM Networks and Graph Neural Networks for Predicting Events of Hypoglycemia

Fabian Hüni[1(✉)], Jose Garcia-Tirado[2], and Kaspar Riesen[1]

[1] Institute of Computer Science, University of Bern, Bern, Switzerland
{fabian.hueni,kaspar.riesen}@unibe.ch
[2] University Clinic for Diabetes, Endocrinology, Nutritional Medicine, and Metabolism, Inselspital – University Hospital Bern, University of Bern, Bern, Switzerland
jose.garcia@unibe.ch

Abstract. Diabetes disrupts the regulation of blood glucose levels, causing *hyperglycemia* (high glucose levels) and *hypoglycemia* (low glucose levels). Hypoglycemia, in particular, can result in cognitive impairment, seizures, and loss of consciousness, making its prediction and avoidance crucial for enhancing patients' quality of life. With the help of pattern recognition techniques and the relatively widespread use of *continuous glucose monitoring* (CGM) devices, intelligent systems can be developed, which in turn can predict hypoglycemic events in advance. In the present paper we propose to use a *Long Short-Term Memory* (LSTM) model and a *Graph Attention Network* (GAT) for this task and we evaluate both models on a real-world CGM dataset of 37 type 1 diabetes persons. In an empirical evaluation, we observe that the LSTM model excels in short-term predictions (15 min), whereas the GAT model performs better for longer horizons (30 and 60 min). Additionally, the GAT model demonstrates more stable performance across all horizons and lower variability across different datasets compared to the LSTM.

Keywords: Type 1 diabetes · LSTM · Graph Attention Network

1 Introduction

Diabetes mellitus is a condition marked by high blood glucose levels due to disorders in insulin secretion or insulin action. There are two main types of diabetes, viz. type 1, where the body does not produce insulin, and type 2, where the body is resistant to insulin. Type 1 diabetes is treated with insulin injections, which help muscles absorb glucose from the blood [1]. Without enough insulin, blood sugar levels can become too high (termed *hyperglycemia*), and too much insulin can cause blood sugar to drop too low (termed *hypoglycemia*). Symptoms of hypoglycemia can range from headaches and sweating to unconsciousness or

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 52–61, 2025.
https://doi.org/10.1007/978-3-031-80507-3_6

even death. The present paper focuses on predicting hypoglycemia using statistical and structural pattern recognition techniques in a binary classification scenario. That is, we aim at predicting whether or not blood glucose will fall below a certain threshold within a certain time interval (blood glucose can be measured in mg/dl (or mmol/L), and levels below 70 mg/dl are considered dangerous [2]).

Devices for *Continuous Glucose Monitoring* (CGM) can help persons manage their blood glucose level by providing real-time data. Moreover, the data output by these devices allow for the research and development of intelligent systems to predict hyperglycemia and hypoglycemia events [3]. Predicting hypoglycemia, especially during sleep (*nocturnal hypoglycemia*), is actually crucial for timely preventive actions.

Current research on hypoglycemia prediction shows a variety of approaches and objectives. Roughly speaking, studies can be categorized into regression tasks, which predict actual blood glucose levels, and classification tasks, which identify hypoglycemia events [4]. In both scenarios, the prediction horizons vary from short-term (less than 60 min) to mid-term (60 to 240 min) and long-term (over 240 min). Most research focuses on type 1 diabetes due to its more predictable nature, although type 2 diabetes, which has more variable blood glucose levels, affects more patients. Data for these studies comes from clinical trials using CGM devices or is generated by simulators like UVA/PADOVA [5]. Different machine learning models, ranging from statistical methods to deep neural networks, are used, and studies vary in the features they utilize, with some using only blood glucose levels and others incorporating additional data such as insulin doses and electrocardiograms.

Bertachi et al. [6], for instance, use a *Multi-Layer Perceptron* (MLP) and *Support Vector Machine* (SVM) to predict nocturnal hypoglycemia in 12 patients using CGM and FitBit data. Iacono et al. [7] develop and research a personalized hypoglycemia alarm system based on *Recurrent Neural Networks* using in silico data from the UVA/PADOVA simulator. Also *Transformer*-based models [8] and *Random Forest* models [9] have been proposed for hypoglycemia prediction.

In the present paper, two models are used for prediction of hypoglycemia, namely the *Long Short-Term Memory* (LSTM) [10] and the *Graph Attention Network* (GAT) [11]. LSTM models handle time series data and its dependencies, while GAT models, a type of *Graph Neural Network*, work on graph-structured data using attention mechanisms to weigh the importance of different nodes. We compare the performance of both models using real-time CGM data across different window sizes and prediction horizons (the window size refers to the number of consecutive time steps considered for one prediction, and the prediction horizon is the number of future time steps for which a prediction is made). The main research question addressed by this paper is, how GAT models compare to LSTM models in predicting hypoglycemia events using CGM data. Moreover, we aim at researching the impact of different window sizes and prediction horizons on the models' performance.

The remainder of this paper is structured as follows. Section 2 describes in detail the underlying time series data and the construction of the graph from the time series (to be eventually fed into the LSTM and GAT, respectively). Section 3 presents the two models used for prediction and discusses the architectures employed. The empirical evaluation is presented and discussed in Sect. 4, and in Sect. 5, we draw some conclusions and outline possibilities for future research activities.

2 Dataset and Graph Construction

2.1 CGM Time Series

In this paper, we use a recent dataset [12] with CGM data from 37 individual persons. From now on, the term *dataset* is used for a single set of CGM measurements from one person. For each dataset the CGM measurements are collected over a timespan of approximately 30 days in an interval of five minutes. We define the *window size* and *prediction horizon* as follows.

The window size determines the number of consecutive samples from the CGM time series that are considered together for one prediction. Prediction horizon, on the other hand, refers to the number of samples in the future for which a prediction is made. Both the window size and the prediction horizon can be expressed in terms of the number of included consecutive samples and the time span they cover. For instance, six samples correspond to a 30 min time span (as the data is collected at five-minute intervals).

Given the objective of the present paper, each sample value in the datasets can be labeled as follows.

- A positive label (*True*) signifies that the patient will be affected by hypoglycemia in the prediction horizon. That is, at least one sample in the prediction horizon falls below the threshold of 70 mg/dl.
- A negative label (*False*) signifies that there is no hypoglycemia event in the considered prediction horizon.

In Fig. 1 an example is provided with a visual explanation of the window size and prediction horizon. The sample of the current time would be labeled as a positive hypoglycemia event since multiple measurements within the prediction horizon fall below the defined threshold.

To evaluate the performance on unseen data, the samples of each dataset are split into disjoint training and test sets (by taking the first (in term of the time) 70% and last 30% of the samples for training and testing, respectively). For hyperparameter tuning, we decide to select one of the 37 datasets as an extra validation dataset. To identify this validation dataset, we apply the following heuristic.

1. We select datasets from all 37 datasets that have less than 1% difference in the percentage of positive labels in the training and test set.

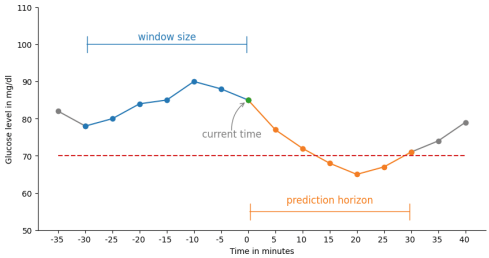


Fig. 1. Illustration of CGM data with annotations for window size, prediction horizon, and current time. Blue points show measurements within the window size used for prediction input. Orange points show measurements within the prediction horizon for label generation. The red dashed line marks the hypoglycemia threshold of 70 mg/dl. (Color figure online)

2. From the selected datasets from step 1, we select the three datasets with the lowest deviation from the mean percentage of positive labels.
3. From the top three datasets, we select the one with the highest number of samples as validation dataset.

2.2 Representing CGM Time Series as Graph

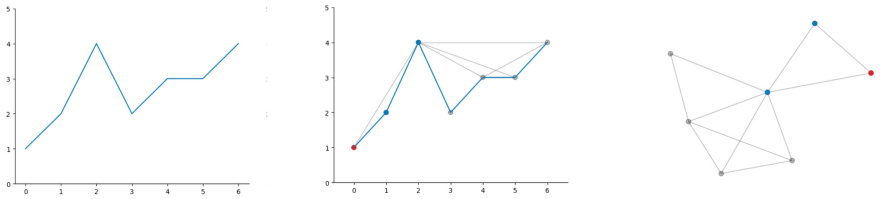
So far the datasets are represented as time series data, with each sample comprising a timestamp t_i and a corresponding CGM value y_i. We propose to transform this time series data into graphs (similar as defined in [13]). This transformation is actually essential for utilizing GNNs in the downstream classification task.

To this end, we adopt the concept of *visibility graphs* [14] for our application. In a visibility graph, each CGM sample is interpreted as a node for the resulting graph. The visibility relationship between each pair of nodes is then evaluated, and an edge is added between them if they are visible to each other. Formally, given two arbitrary samples (t_a, y_a) and (t_b, y_b) in the time series, where t_a, t_b refer to the time stamps and y_a, y_b to glucose measurements at the corresponding time stamps. Then, the corresponding nodes are visible to each other, if any other sample (t_c, y_c) placed between them (i.e., $t_a < t_c < t_b$) fulfills the following equation

$$y_c < y_b + (y_a - y_b)\frac{t_b - t_c}{t_b - t_a} \tag{1}$$

Each node is labeled with two features, viz. the CGM value and its position in the time series data sequence. The inclusion of the node's order might enable the model to capture the temporal relationships between nodes. Additionally, the Euclidean distance between two nodes, connected by an edge, is included as a feature of the corresponding edge.

In Fig. 2 an illustration of the transformation of the time series to a visibility graph is shown (the differently colored nodes help identifying the correspondence between time stamps and nodes).



(a) Time Series (b) Visibility edges (c) Visibility graph

Fig. 2. Illustration of transforming a times series into a visibility graph.

3 Classification Models

We use two state-of-the-art methods for learning prediction models from data. The first is particularly well suited for time series data, while the second is perfectly suited for graph-based data. For the sake of completeness, both concepts are briefly reviewed in two separate subsections.

3.1 Long Short-Term Memory Neural Networks

Long Short-Term Memory (LSTM) [10] neural networks are a special type of *Recurrent Neural Networks* (RNN) particularly designed to address the limitations of traditional RNNs in handling sequential data. One of the primary challenges with standard RNNs is their inability to effectively learn long-term dependencies due to issues such as vanishing and exploding gradients. These issues arise during the training process, where the gradients used to update the network's weights diminish or grow exponentially, leading to poor performance on longer sequences.

LSTMs overcome these challenges with a unique architecture that includes memory cells and three types of gates, viz. input, forget, and output gates, that regulate the flow of information. The input gate determines which new information should be added to the cell state, effectively deciding what is relevant from the current input. The forget gate controls which information should be discarded from the cell state, ensuring that the network does not retain irrelevant data that could hinder learning. Finally, the output gate decides what part of the cell state should be used to compute the output at each time step. By dynamically adjusting these gates during training, LSTMs can learn to prioritize and retain important information over long sequences, making them highly effective for tasks involving time series prediction [15], natural language processing [16], or speech recognition [17].

In the present paper, we use two consecutive LSTM layers, one dropout layer for regularization and a dense output layer with a sigmoid activation function to map the values to probabilities. We employ the *focal cross-entropy loss* for both hyperparameter tuning and training. Hyperparameter tuning is performed on the pre-defined validation dataset with a fixed window size and a fixed prediction

Table 1. Hyperparameters for the LSTM model along with their respective search spaces. The best performing parameters are shown in bold face.

Hyperparameter	Values
Batch Size	**16**, 32, 64, 128, 256, 512, 1028
Learning Rate	0.00001, **0.0001**, 0.001, 0.01
Weight Decay	0.00001, 0.0001, **0.001**, 0.01
# Units LSTM-Layer 1	32, 64, 128,256, **512**
# Units LSTM-Layer 2	**32**, 64, 128, 256, 512
Output Activation Function	ReLu, **Sigmoid**, Linear
Dropout Rate	0.3, **0.4**, 0.5

horizon. The considered hyperparameters for the LSTM model along with their respective search spaces is shown in Table 1 (the best performing parameters are shown in bold face).

3.2 Graph Attention Networks

Graph Attention Networks (GAT) [11] are a class of neural networks designed to operate on graph-structured data by leveraging attention mechanisms. Earlier network-based methods for processing graph data, such as *Graph Convolutional Networks* (GCN) [18], sometimes struggle to adequately capture the varying importance of neighboring nodes when aggregating information. GATs address this limitation by applying self-attention at each node, allowing the model to weigh the significance of its neighbors dynamically. This is achieved through an attention layer that computes a set of attention coefficients for each node, reflecting the importance of its neighbors' features. These coefficients are then used to aggregate the neighbors' information, resulting in a weighted sum that emphasizes the most relevant nodes. GATs have shown significant promise in a wide range of applications (e.g., from social network analysis [19] to fraud detection [20], to name just two examples).

The GAT model employed in this work comprises embedding generation, message-passing, pooling, and an output activation function. First, the features of nodes and edges are transformed into node and edge embeddings. Next, multiple rounds of message passing are applied, where connected nodes exchange information. Then, the pooling layer reduces the number of nodes in the graph for a more compact representation. Finally, the output activation function maps the output of the pooling layer to a prediction for the classification task. Preliminary experiments have shown that, unlike the LSTM model, the focal cross-entropy loss does not improve performance for the GAT model. Therefore, the loss function is fixed to *binary cross-entropy*.

The considered hyperparameters for the GAT model along with their respective search spaces is summarized in Table 2 (the best performing parameters are shown in bold face). Note that in addition to the GAT-specific hyperparameters,

Table 2. Hyperparameters for the GNN model along with their respective search spaces. The best performing parameters are shown in bold face.

Hyperparameter	Values
Batch Size	16, 32, 64, **128**, 256
Learning Rate	0.00001, 0.0001, **0.001**, 0.01
Weight Decay	**0.00001**, 0.0001, 0.001, 0.01
# Node Dense Units	8, 16, 32, 64, 128, **256**
# Edge Dense Units	**8**, 16, 32, 64, 128, 256
# GAT-Layers	**2**, 3, 4, 5, 6
# Heads per GAT-layer	3, 4, 5, **6**
# Channels per head	6, 7, **8**, 9, 10
Edge dropout	0.1, 0.2, 0.3, **0.4**, 0.5
Pool reduce type	Mean, **Min**, Max

the batch size, output activation function, weight decay and learning rate are also optimized as presented in Sect. 3.1.

4 Experimental Evaluation

The goal of the experimental evaluation is twofold. First, we aim at evaluating the performance of the LSTM and GAT models on the hypoglycemia prediction task and compare their results. Second, we aim at assessing the impact of varying prediction horizons and window sizes on the performance of both models.

Table 3. Precision (p), Recall (r) and the F_2 measure for both LSTM and GAT model achieved with different window sizes and prediction horizons. The better performing model per metric and window/horizon pair is highlighted in light gray.

Window	Model	Prediction Horizon								
		15			30			60		
		p	r	F_2	p	r	F_2	p	r	F_2
30	LSTM	0.211	0.868	0.503	0.146	0.745	0.376	0.103	0.626	0.260
	GAT	0.133	0.779	0.388	0.147	0.755	0.408	0.142	0.770	0.403
60	LSTM	0.214	0.846	0.501	0.144	0.703	0.365	0.105	0.568	0.262
	GAT	0.150	0.767	0.412	0.141	0.772	0.400	0.140	0.779	0.398
120	LSTM	0.224	0.805	0.502	0.143	0.650	0.348	0.105	0.502	0.255
	GAT	0.139	0.770	0.397	0.148	0.760	0.409	0.143	0.761	0.396

In Table 3 the precision (p), recall (r), and the F_2 measure for both LSTM and GAT model achieved with different window sizes and prediction horizons are shown. Regarding the results, the following two observations can be made.

- The LSTM outperforms GAT for 15-min prediction horizons, but the GAT performs better for 30 and particularly for 60-min horizons in general.
- Even the best performing model with a recall of 0.868 and a precision of 0.211 for a 15-min horizon is not yet production-ready due to a high false positive rate, making it less reliable for patient support.

Another interesting insight concerns the standard deviation of the F_2 measure distribution over all datasets as shown in Fig. 3. In the examined combination of window size and prediction horizon (both 30 min in this illustration), the LSTM model exhibits a standard deviation of 0.175 while the distribution of the GNN model has a standard deviation of only 0.034. It is clearly visible that the distribution of the GNN is much more compact than that of the LSTM. This indicates that the GNN model is substantially better than the LSTM model in producing stable and robust results over all scenarios and datasets.

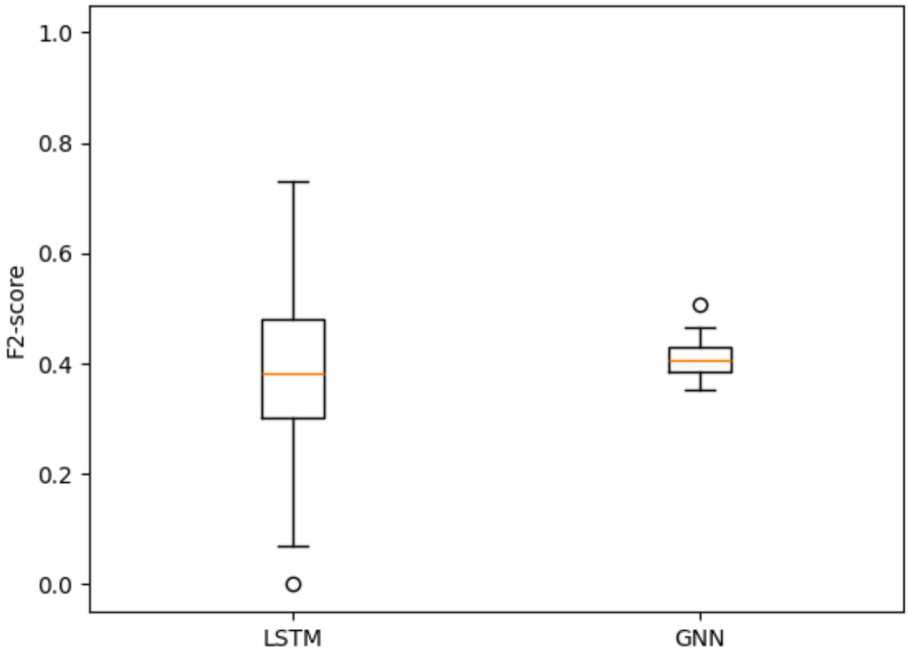


Fig. 3. Distribution of the F2-Scores across all datasets for both the LSTM and GNN models.

5 Conclusion and Future Work

Advances in technologies for continuous glucose monitoring and novel methods stemming from artificial intelligence have opened new possibilities for developing intelligent systems for diabetes management. This paper researches an LSTM and a GNN model for the specific task of hypoglycemia prediction. Both models are evaluated across 37 real-world patient datasets with window sizes of 30, 60, and 120 min and prediction horizons of 15, 30, and 60 min. In an empirical evaluation, we observe that the LSTM surpasses the GAT for 15-min prediction horizons, whereas the GAT excels over the LSTM for 30-min and especially for 60-min horizons. Furthermore, the GAT model exhibits more consistent performance across all horizons and shows lower variability across different datasets when compared to the LSTM.

We see several rewarding avenues to be pursued in future work. In this paper we use an average patient data for tuning, applying the resulting hyperparameters across all models. Future studies might benefit from tuning hyperparameters for each dataset individually or using combined validation sets to potentially enhance performance. Moreover, we could assess other algorithms for transforming the time series to visibility graphs (e.g., horizontal visibility graphs). Additionally, one could enrich the node and edge embeddings with features such as node degree and connection slope, or apply heuristics to limit the edges to the local neighborhood only. Last but not least, generating additional data from existing time series or using artificial data from a diabetes simulator like UVA/PADOVA [7] could provide a cost- and time-efficient means to increase the size of the training data.

References

1. Pathak, V., Pathak, N.M., O'Neill, C.L., Guduric-Fuchs, J., Medina, R.J.: Therapies for type 1 diabetes: current scenario and future perspectives. Clin. Med. Insights Endocrinol. Diab. **12**, 1179551419844521 (2019)
2. Seaquist, E.R., et al.: Hypoglycemia and diabetes: a report of a workgroup of the American diabetes association and the endocrine society. Diabetes Care **36**(5), 1384–1395 (2013)
3. Rodbard, D.: Continuous glucose monitoring: a review of successes, challenges, and opportunities. Diab. Technol. Therapeutics **18**(Suppl. 2(S2)), S3–S13 (2016)
4. Mujahid, O., Contreras, I., Vehi, J.: Machine learning techniques for hypoglycemia prediction: trends and challenges. Sensors **21**(2) (2021)
5. Man, C.D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., Cobelli, C.: The UVA/PADOVA type 1 diabetes simulator: new features. J. Diab. Sci. Technol. **8**(1), 26–34 (2014). PMID: 24876534
6. Bertachi, A., et al.: Prediction of nocturnal hypoglycemia in adults with type 1 diabetes under multiple daily injections using continuous glucose monitoring and physical activity monitor. Sensors **20**(6) (2020)
7. Iacono, F., Magni, L., Toffanin, C.: Personalized LSTM-based alarm systems for hypoglycemia and hyperglycemia prevention. Biomed. Signal Process. Control **86**, 105167 (2023)

8. Lee, S.-M., Kim, D.-Y., Woo, J.: Glucose transformer: forecasting glucose level and events of hyperglycemia and hypoglycemia. IEEE J. Biomed. Health Inform. **27**(3), 1600–1611 (2023)
9. Dave, D., et al.: Feature-based machine learning model for real-time hypoglycemia prediction. J. Diabetes Sci. Technol. **15**(4), 842–855 (2021). PMID: 32476492
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
11. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018, Conference Track Proceedings. OpenReview.net (2018)
12. Garcia-Tirado, J., et al.: Assessment of meal anticipation for improving fully automated insulin delivery in adults with type 1 diabetes. Diab. Care **46** (2023)
13. Tastan, A., Escorihuela-Altaba, C., Garcia-Tirado, J., Riesen, K.: Clustering time series data for personalized type 1 diabetes management. To be published in the Proceedings of ICPRAI 2024
14. Lacasa, L., Luque, B., Ballesteros, F., Luque, J., Nuo, J.C.: From time series to complex networks: the visibility graph. Proc. Natl. Acad. Sci. **105**(13), 4972–4975 (2008)
15. Fankhauser, B., Bigler, V., Riesen, K.: Impute water temperature in the swiss river network using LSTMs. In: Santana, M.C., De Marsico, M., Fred, A.L.N. (eds.) Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2024, Rome, Italy, 24–26 February 2024, pp. 732–738. SCITEPRESS (2024)
16. Yao, L., Guan, Y.: An improved LSTM structure for natural language processing. In: 2018 IEEE International Conference of Safety Produce Informatization (IICSPI), pp. 565–569. IEEE (2018)
17. Graves, A., Jaitly, N., Mohamed, A.: Hybrid speech recognition with deep bidirectional LSTM. In: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 273–278. IEEE (2013)
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR, abs/1609.02907 (2016)
19. Song, W., Xiao, Z., Wang, Y., Charlin, L., Zhang, M., Tang, J.: Session-based social recommendation via dynamic graph attention networks. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 555–563 (2019)
20. Liu, C., Sun, L., Ao, X., Feng, J., He, Q., Yang, H.: Intention-aware heterogeneous graph attention networks for fraud transactions detection. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 3280–3288 (2021)

Evaluation Metrics in Saliency Maps Applied to Graph Regression

Natàlia Segura-Alabart[1(✉)], Alberto Fernández[2], Alexander Kensert[3], Deirdre Cabooter[3], and Francesc Serratosa[1]

[1] Departament d'Enginyeria Informática i Matemátiques, Universitat Rovira i Virgili, 43007 Tarragona, Spain
natalia.segura@urv.cat
[2] Departament d'Enginyeria Química, Universitat Rovira i Virgili, 43007 Tarragona, Spain
[3] Department for Pharmaceutical and Pharmacological Sciences, University of Leuven, 3000 Leuven, Belgium

Abstract. Graph Convolutional Networks (GCNs) excel at learning and extracting meaningful features from data represented by graphs. However, their decision-making processes are complex and not readily interpretable. Post-hoc explanation methods, such as saliency map generators (SMG), like Grad-CAM or Grad-CAM++, address this issue by highlighting areas of input data that influence the model decisions. Although initially applied to image classification, SMG can be adapted to interpret the roles of nodes and edges in graph-based regression or classification tasks. This paper introduces Single Step Metrics (STM), novel metrics designed to evaluate the performance of SMG in graph regression tasks, offering a new approach for quantitatively assessing the interpretability of GCN models. By applying these metrics, we demonstrate that the STM results align with the insights provided by SMG. Specifically, we compare the performance of Grad-CAM and Grad-CAM++ across three chemical datasets, proving that STM can effectively differentiate between SMG methods and identify the one better suited for a given problem. Our findings indicate that both SMG and STM consistently show that Grad-CAM is more suited for analyzing graph-based chemical data.

Keywords: Saliency map · Graph regression · Explainability

1 Introduction

Graph Convolutional Networks (GCNs) [1] are deep neural models renowned for their ability to extract meaningful features in prediction tasks where data is represented by graphs. However, understanding the decision-making processes of such models is a difficult task. This is because they are typically characterised by multiple layers of non-linear transformations, wherein the mapping between input and output is complex and not readily interpretable.

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 62–71, 2025.
https://doi.org/10.1007/978-3-031-80507-3_7

Saliency map generators (SMG) are techniques initially applied to image classification, which indicate the areas of the input data that played an important role in the model decision [2]. These areas are called saliency maps. Thus, they have become indispensable tools for interpreting the predictions of deep neural models. SMG, when adapted to GCNs, can be used for the interpretation of nodes and edges in regression or classification applications where the data is characterised by graphs, as demonstrated in [3,4].

Various metrics exist to measure the correlation between the saliency map explanation and the model's prediction abilities given an image, as described in [2,5,6]. These metrics compare the model's predictions using the original input data to those using modified input data. The modifications are based on the saliency map generated from the original input data through a process known as masking. The quality of the output depends on which part of the input data has been modified. If the modified data corresponds to the areas highlighted by the saliency map as important, the decrease in output quality should be more significant compared to modifications in areas deemed unimportant by the saliency map. The input data masking can be executed either in a single step or iteratively. This paper focuses on the former case, known as Single Step Metrics (STM) [2].

The aim of this paper is to introduce STM designed for graph inputs, with a focus on graph regression, to evaluate the performance of SMG. These metrics measure the effectiveness of the SMG in identifying the importance of each node in the regression results.

2 Related Work

2.1 Saliency Map Generators: Post-hoc Explanation Methods

SMG are methods encompassed in post-hoc explanation methods [2], which are general approaches to generate explanations of the decisions made by any prediction model without requiring retraining. Specifically, for SMG, Class Activation Mapping (CAM) [7] and its successors, such as Grad-CAM [8] and Grad-CAM++ [5] are some of the most popular methods. Interestingly, Grad-CAM was adapted to generate the saliency maps for prediction models based on GCN in [3], which inspired our proposal. In the next subsections, we describe Grad-CAM and Grad-CAM++ in detail.

Grad-CAM or Gradient-weighted Class Activation Mapping is a SMG based on deep convolutional neural networks. Nevertheless, this method deduces the saliency map taking into consideration only the last layer of the classification model, in a similar way as CAM.

Formally, for any class c within a dataset, the gradient of the score for class c, y^c (prior to a softmax or any activation function), is calculated with respect to the activation of feature maps in a convolutional layer k, A^k. Subsequently, these resulting gradients are global average pooled across the width and height

dimensions of the data (denoted as i and j, respectively). The weights w_k^c for a feature map A^k and class c are:

$$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{1}$$

where Z is the number of elements in the feature map A^k. Finally, a weighted combination of w_k^c with A^k is performed to obtain the class-specific saliency map, L_{ij}^c:

$$L_{ij}^c = ReLU(\sum_k w_k^c A^k) \tag{2}$$

Grad-CAM++ is a generalization of the Grad-CAM algorithm to improve the handling of multiple occurrences of a localized object in an image and improve the localization accuracy. The main difference is how the weights w_k^c for a feature map A^k and class c are computed. First, the gradient weights α_{ij}^{kc} are computed as:

$$\alpha_{ij}^{kc} = \frac{\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}} \tag{3}$$

where *(i, j)* and *(a, b)* are iterators over the same A^k to avoid confusion. Then, the weights w_k^c are reformulated from Eq. (1) as:

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} ReLU(\frac{\partial Y^c}{\partial A_{ij}^k}) \tag{4}$$

2.2 Single Step Metrics for Classification

STM are evaluation metrics designed to assess the correctness of saliency maps generated by a SMG, such as Grad-CAM. These metrics are based on applying a usually slight modification to the input data and later measuring the consequential impact on the final prediction.

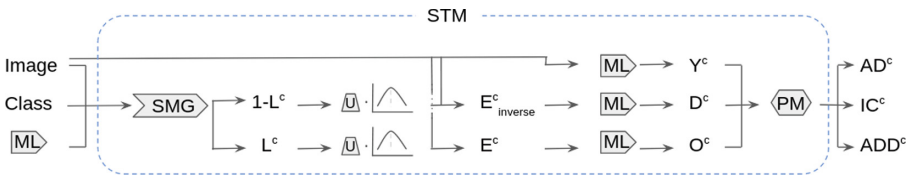


Fig. 1. STM in image classification. ML stands for Machine Learning, 1-L^c represents the inverse masking operation in I, U denotes Upsampling and Normalization, and PM represents Performance Metrics.

Figure 1 provides an overview of the STM process for an image classification model. The process begins with creating the saliency map L^c using the SMG based on the input data or image i, a class c, and the model's trained weights. Next, with the input data or image and L^c, the explanation map E^c is computed. E^c involves a masking operation, where the saliency maps are point-wise multiplied with the original image I. This operation is defined as:

$$E^c = s(u(L^c)) \circ I \tag{5}$$

where $u(\cdot)$ indicates the upsampling into the original data dimensions, $s(\cdot)$ is the min-max normalization function and $\circ$ is the Hadamard product. E^c only preserves the information of I in the pixels that are considered important and reduces the ones that are not. Then, using this modified image a new classification score O^c for a class c is predicted using the trained ML model. A higher value of O^c is expected to correspond to an increased confidence in the model's prediction of the image belonging to a specific class.

An inverse masking operation is also performed on I. This involves creating an inverse explanation map $E^c_{inverse}$ for a given class c, calculated by point-wise multiplication of the inverse of the saliency maps with the original image I, as follows:

$$E^c_{inverse} = [1 - s(u(L^c))] \circ I \tag{6}$$

In this case, $E^c_{inverse}$ preserves the information in the pixels of I that are not considered important for the final classification and reduces the information in the pixels that are important (i.e. *deletion*). Using this modified image, a new classification score, D^c, is predicted using the trained ML model.

Consequently, there are three different classification scores for the same image, depending on the masking applied: Y^c for no masking, O^c for direct masking, and D^c for inverse masking. Finally, these scores are used to evaluate the STM using specific performance metrics. The three performance metrics are Average drop percentage (AD), Increase in Confidence (IC), and Average Drop in Deletion percentage (ADD) [5,6], which are explained in Sect. 2.3.

2.3 Performance Metrics

Average Drop (AD) is the average percentage drop in the model's confidence for a given class c in an image classification value when comparing it to O^c, defined as:

$$AD = \frac{1}{N} \sum_{n=1}^{N} \frac{max(0, Y_i^c - O_i^c)}{Y_i^c} \times 100 \tag{7}$$

We use max to handle cases where Y_i^c is higher than O_i^c and N denotes the number of elements in the dataset. In case of equal output values, it can be deduced that the saliency map is insensitive to the model-learned parameters. The AD value is computed per image and then averaged over the entire dataset. A lower AD value indicates better performance.

Increase in Confidence (IC) is complementary to the previous metric, AD. IC measures the number of times in the entire dataset that O_i^c is higher than Y_i^c. We define IC as follows:

$$IC = \frac{1}{N} \sum_{n=1}^{N} 1_{[Y_i^c < O_i^c]} \times 100 \tag{8}$$

where 1_x is an indicator function that returns 1 when the argument is true, $Y_i^c < O_i^c$. The IC value is computed per image and then averaged over the entire dataset. A higher IC value indicates better performance.

Average Drop in Deletion (ADD) modifies AD within a specific context by an inverse operation, $E_{inverse}^c$, to compute the classification score, D_i^c. It evaluates the average percentage drop in the model's confidence caused by the inverse masking on the final prediction. ADD can be defined as:

$$ADD = \frac{1}{N} \sum_{n=1}^{N} \frac{Y^c - D^c}{Y_c} \times 100 \tag{9}$$

The ADD value is computed per image and then averaged over the entire dataset. A higher ADD value indicates better performance.

3 Method

3.1 Model Architecture

The GCN model's architecture begins with an input layer that accepts graph structures. This is followed by two Graph Convolutional layers, which capture local neighborhood information. A readout layer aggregates the graph-level information, which is then passed through three Fully Connected layers.

The final output is a single value representing the regression value. Weights in the model are randomly initialized. The training process involves minimizing the mean squared error (MSE) between the predicted and actual global property using the Adam optimizer.

3.2 Single Step Metrics for Graph Regression

STM were initially designed for assessing image classification models. In this study, we propose to extend the application of STM to evaluate graph regression models, thereby introducing a novel evaluation metric for this purpose.

Figure 2 provides an outline of the STM process for a graph regression model. This process is very similar to the one explained in Sect. 2.2 but two modifications have been incorporated. The first one involves the use of graphs as input data instead of images, while the second one implies the use of STM in regression models. Accordingly, we define the explanation map, E_i, for a given graph as

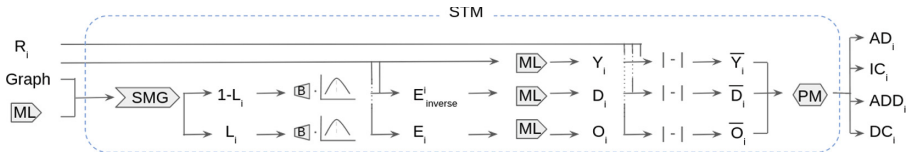


Fig. 2. STM in graph regression. ML stands for Machine Learning, 1-L_i represents the inverse masking operation in the graph, B denotes Broadcasting and Normalization, and PM represents Performance Metrics.

the point-wise multiplication of the saliency maps L_i with the node features of the original graph X_i:

$$E_i = b(|s(L_i)|) \circ X_i \tag{10}$$

where $|s(\cdot)|$ represents the normalization function applied to the saliency map values, transforming them to a range between 0 and 1 before being transformed into their absolute values, *b(.)* represents the broadcasting function into the original dimensions of X_i. Note that, since we are working with GCNs, the graph edges are not considered in this process. Thus, each element of E_i reflects the importance of a node rather than a pixel. And the inverse explanation map $E^i_{inverse}$ as the point-wise multiplication of the inverse of the saliency maps with X_i as:

$$E^i_{inverse} = [1 - b(|s(L_i)|)] \circ X_i \tag{11}$$

Note that if we change the *b(.)* function into a *u(.)* function we can work with image data as in Eq. (5) and Eq. (6).

Another change is the use of the absolute difference between the real value R_i and the predicted values for Y_i,O_i, and D_i ($\bar{Y}_i$, $\bar{O}_i$,$\bar{D}_i$, respectively), defined as follows:

$$\bar{W}_i = |R_i - W_i| \tag{12}$$

where W_i represents the predicted score. The reason is that in the context of classification models, a higher value of W_i is expected to correspond to an increased confidence in the model's prediction of the data belonging to a specific class. Contrarily, in regression models, the objective is to achieve a prediction the closest to the actual value. Therefore, using the actual predicted value to compare its performance is not meaningful.

3.3 Performance Metrics

The performance metrics have been adapted to accommodate regression, and we have introduced a fourth metric, Drop in Confidence (DC). In each of these metrics, the goal is to attain higher values, indicating enhanced performance.

Average Drop (AD) is the average increase in the model's confidence for the prediction value, $\bar{Y}_i$ when comparing it to $\bar{O}_i$, defined as:

$$AD = \frac{1}{N} \sum_{n=1}^{N} \frac{|\bar{Y}_i - \bar{O}_i|}{max(\bar{Y}_i, \bar{O}_i)} \tag{13}$$

Increase in Confidence (IC) quantifies the frequency with which $\bar{O}_i$ is lower than $\bar{Y}_i$ across the entirety of the dataset, defined as:

$$IC = \frac{1}{N} \sum_{n=1}^{N} 1_{[\bar{Y}_i > \bar{O}_i]} \tag{14}$$

Average Drop in Deletion (ADD) is the comparison of the prediction value, $\bar{Y}_i$, to $\bar{D}_i$, defined as:

$$ADD = \frac{1}{N} \sum_{n=1}^{N} \frac{max(0, \bar{D}_i - \bar{Y}_i)}{\bar{D}_i} \tag{15}$$

We use *max* to handle cases where $\bar{Y}_i$ is higher than $\bar{D}_i$, as well as when $\bar{Y}_i$ equals $\bar{O}_i$.

Drop in Confidence (DC) is a new STM evaluation metric that complements ADD. This metric is designed to provide additional information similar to how IC complements AD. DC quantifies the frequency in the entire dataset where $\bar{Y}_i$ is lower than $\bar{D}_i$. We define DC as:

$$DC = \frac{1}{N} \sum_{n=1}^{N} 1_{[\bar{Y}_i < \bar{D}_i]} \tag{16}$$

where 1_x is an indicator function that returns 1 when the argument is true, $\bar{Y}_i < \bar{D}_i$.

4 Practical Application

4.1 Datasets

To validate our approach, we applied it to three distinct chemical datasets, which were defined to test models that predict the retention time of various molecules. These datasets were obtained from [4]. Retention time is the time a compound remains in the stationary phase of a chromatography system before being detected, indicating its identity and concentration. These datasets consists of chemical compounds represented as graphs, derived from SMILE strings, where nodes represent the chemical atoms and edges represent the chemical bonds. The HILIC (Hydrophilic Interaction Liquid Chromatography) dataset

consists of 1400 molecules with node sizes ranging from 4 to 91 and a retention time between 0.89 and 10.28 min. The RIKEN dataset consists of 862 molecules with node counts varying from 8 to 100 nodes and retention times spanning from 1.52 to 10.4 min. Lastly, the SMRT dataset contains 77980 molecules with node sizes between 8 and 50 and retention times between 5.67 and 24.53 min. Note that both the RIKEN and SMRT datasets were acquired under Reversed-Phase Liquid Chromatography (RPLC) conditions.

4.2 Architecture Configuration

The input data, consisting of SMILE strings, was first converted into graph representations. In our experiments, we used two GCNs, each with 256 neurons, residual connections, and ReLU activation functions, followed by a readout layer. After that, three Fully Connected dense layers were employed, comprising 1024, 1024, and 1 neuron, respectively, all utilizing ReLU activation functions.

The training occurred over 50 or 150 epochs, with batch sizes of 32 or 128 for the HILIC and RIKEN datasets, and the SMRT dataset, respectively. The training and validation was done using 90% of the data, with approximately 90% of that subset used for training and the remaining 10% for validation. We used the Adam optimizer and a learning rate scheduler that reduced the learning rate by a factor of 0.1, with a minimum learning rate of 10^{-6}. Early stopping halted training after 10 epochs without validation loss improvement.

4.3 Predictive Performance

To evaluate the overall performance of the model, the MSE, and coefficient of determination (R^2) were calculated on all datasets for both the training and testing sets. For the HILIC dataset, the MSE values were 2.14 for the training set and 2.11 for the testing set, with corresponding R^2 values of 0.71 and 0.69, respectively. The RIKEN dataset showed better performance, with MSE values of 0.64 for the training set and 0.70 for the testing set, and R^2 values of 0.84 and 0.80, respectively. For the SMRT dataset, the MSE values were 0.94 for the training set and 1.34 for the testing set, while the R^2 values were 0.88 and 0.84, respectively. These results highlight the superior performance of the RIKEN dataset in terms of MSE, and the strong predictive capabilities of the RIKEN and SMRT datasets as indicated by their higher R^2 values.

4.4 STM Evaluation

We selected a molecule to visualize the saliency maps. As shown in Fig. 3, graph *a* exhibits inverted red and green regions compared to the other graphs (*b, c*) in the same rows with a positive contribution of the more polar atoms in graph *a*, versus a negative contribution of these polar atoms in graphs *b* and *c*. The remaining shared molecules exhibit the same behavior. These findings align with the chemical insights derived from the saliency maps. This is because

the chromatographic mechanism (RPLC) for the RIKEN and SMRT datasets operate on similar principles, while HILIC operates on opposite principles as such, polar functional groups will contribute positively to retention in HILIC. In contrast, Grad-CAM++ highlights only the important regions, all shown in green, without indicating the direction of their importance.

Table 1 presents the evaluation metrics for the HILIC, RIKEN, and SMRT datasets using Grad-CAM and Grad-CAM++ SMG methods. Overall, the results highlight that Grad-CAM is more sensitive in capturing significant features compared to Grad-CAM++ across the datasets, as illustrated by the saliency maps. Specifically, Grad-CAM exhibits higher IC, DC and ADD values in the RIKEN and SMRT datasets. In contrast, for the HILIC dataset, Grad-CAM++ demonstrates a higher IC value, while the DC and ADD values are similar on the two methods.

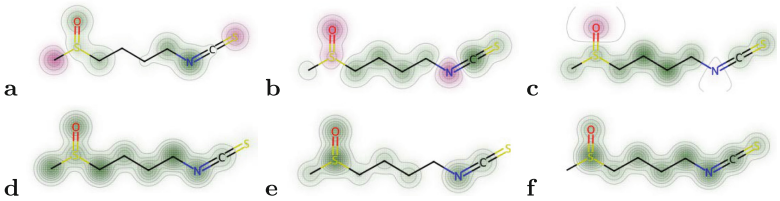


Fig. 3. Saliency maps generated using Grad-CAM (first row) and GradCAM++ (second row) methods for HILIC (first column: a, d), RIKEN (second column: b, e), and SMRT (third column: c, f). The green regions indicate positive contributions and the red regions indicating negative contributions. (Color figure online)

Table 1. SMG performance metrics

Database	SMG	IC	AD	DC	ADD
HILIC	GradCAM	0.14	0.72	0.63	0.39
	GradCAM++	0.41	0.49	0.64	0.42
RIKEN	GradCAM	0.17	0.74	0.97	0.80
	GradCAM++	0.02	0.82	0.79	0.56
SMRT	GradCAM	0.14	0.72	0.97	0.82
	GradCAM++	0.09	0.77	0.85	0.64

5 Conclusions and Future Work

Understanding the predictions generated by deep learning models is crucial for their practical application in real-world contexts. While visualizations of

saliency maps offer promise in improving model interpretability, understanding these maps often requires additional analysis and domain-specific knowledge. This study highlights the importance of interpretability in GCNs by introducing four different STM metrics for quantitatively assessing SMG in graph regression tasks. Our comparison of Grad-CAM and Grad-CAM++ across three chemical datasets using STM demonstrates that Grad-CAM is more suited for analyzing graph-based chemical data, as confirmed by both SMG and STM evaluations. In future work, a comprehensive comparison with other SMG techniques is needed to further validate these findings.

Acknowledgements. This research is supported by the Martí i Franqués program of Universitat Rovira i Virgili, the AGAUR research group 2021SGR-00111: 'ASCLEPIUS: Smart Technology for Smart Healthcare', and the Spanish project NextPandemics (PID2022-138327OB-I00). A.K.'s is funded by a joint initiative of the Research Foundation Flanders (FWO) and the Walloon Fund for Scientific Research (FNRS) (EOS - research project "Chimic" (EOS ID: 30897864)).

References

1. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
2. Gomez, T., Mouchére, H.: Computing and evaluating saliency maps for image classification: a tutorial. J. Electron. Imaging **32**(2), 020801 (2023)
3. Pope, P.E., Kolouri, S., Rostami, M., Martin, C.E., Hoffmann, H.: Explainability methods for graph convolutional neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10764–10773 (2019)
4. Kensert, A., Bouwmeester, R., Efthymiadis, K., Van Broeck, P., Desmet, G., Cabooter, D.: Graph convolutional networks for improved prediction and interpretability of chromatographic retention data. Anal. Chem. **93**(47), 15633–15641 (2021)
5. Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-CAM++: generalized gradient-based visual explanations for deep convolutional networks. In: IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 839–847 (2018)
6. Jung, H., Oh, Y.: Towards better explanations of class activation mapping. In: IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1316–1324 (2021)
7. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
8. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. In: IEEE International Conference on Computer Vision (ICCV), pp. 618–626 (2017)

LESI-GNN: An Interpretable Graph Neural Network Based on Local Structures Embedding

Giorgia Minello[1(✉)], Lingfeng Zhang[2], Alessandro Bicciato[1], Luca Rossi[2], Andrea Torsello[1], and Luca Cosmo[1]

[1] Ca' Foscari University of Venice, Venice, Italy
{giorgia.minello,alessandro.bicciato,andrea.torsello,luca.cosmo}@unive.it
[2] The Hong Kong Polytechnic University, Hung Hom, Hong Kong
lingfeng.zhang@connect.polyu.hk, luca.rossi@polyu.edu.hk

Abstract. In recent years, deep learning researchers have been increasingly interested in developing architectures able to operate on data abstracted as graphs, *i.e.*, Graph Neural Networks (GNNs). At the same time, there has been a surge in the number of commercial AI systems deployed for real-world applications. At their core, the majority of these systems are based on black-box deep learning models, such as GNNs, greatly limiting their accountability and trustworthiness. The idea underpinning this paper is to exploit the representational power of graph variational autoencoders to learn an embedding space where a "convolution" between local structures and latent vectors can take place. The key intuition is that this embedding space can then be used to decode the learned latent vectors into more interpretable latent structures. Our experiments validate the performance of our model against widely used alternatives on standard graph benchmarks, while also showing the ability to probe the model decisions by visualising the learned structural patterns.

Keywords: Graph Neural Network · Interpretability · Autoencoder

1 Introduction

Deep learning has become an indispensable tool for a wide range of applications, from biomedical image analysis [9] to natural language processing [21] and speech recognition [16]. Despite this, deep learning-based systems are not yet considered reliable in many fields due to their black-box nature, which prevents users to derive human-intelligible explanations of the model predictions. The resulting lack of accountability and trustworthiness in turn greatly limits the use of these systems in applications where fairness, privacy, and safety play a critical role [29]. Consider for example the case of loan applications and hiring decisions, where understanding why a model makes a certain prediction is crucial to ensuring the system fairness [19].

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 72–81, 2025.
https://doi.org/10.1007/978-3-031-80507-3_8

In recent years, Graph Neural Networks (GNNs) have attracted increasing attention as an effective way to learn from graph data and solve tasks such as node classification, graph classification, and link prediction. The popularity of these networks stems from the rich nature of graph data as well as the wide range of systems that can be modelled using graphs, from molecules [27] to social networks [14] and images [21]. Widely used GNNs include Graph Convolutional Networks (GCNs) [13], GraphSAGE [11], and Graph Attention Networks (GATs) [25]. These models typically work by combining feature and structural information by recursively passing messages along the edges of the input graph.

Much like other deep learning approaches, GNNs also lack a clear mechanism to explain their predictions. To complicate things further, endowing GNNs with such a mechanism presents several additional challenges due to the distinctive characteristics of graph data. Most notably, the non-vectorial nature of graphs hinders the ability to apply existing explanation methods, which are often designed to deal with image or text data.

In this paper, we introduce a novel GNN model that attempts to overcome this limitation by making use of a latent space which allows us to interpret the patterns extracted by the model. To this end, we propose to leverage recent advancements in Graph Variational Autoencoders (GVAEs) [23]. With a pretrained GVAE to hand, we map the neighborhood of each node of an input graph to the latent space defined by the encoder. More specifically, for each node neighborhood the encoder outputs the parameters of a probability distribution. Given a set of latent vectors, we then define the output signal for the input node as a function of the probability of the latent vectors being sampled from its corresponding distribution. Crucially, this allows us to map the learned latent vectors onto graph structures using the decoder, which in turn allows us to interpret the model decisions.

The remainder of this paper is organized as follows. Section 2 reviews the related work, while Sect. 3 introduces the relevant background. The proposed model is presented in Sect. 4, while its performance on a graph classification and the interpretability mechanism are tested in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related Work

In the early work of Sperduti and Starita [24], recursive operators are used to adapt neural networks to operate on graphs. However, the vast majority of existing GNNs work by exchanging feature and structural information between neighboring nodes through a variation of what is commonly known as the *message passing* mechanism [1,10,13]. The common idea underpinning these methods is that the input node feature can be enhanced by allowing neighboring nodes to exchange structural and feature information, thus allowing the nodes to learn about the structural patterns and feature distribution of potentially distant neighbors. Despite its popularity, the discriminating power of the message passing model has been brought into question after it has been shown to be

formally equivalent to the WL graph isomorphism test [17,26]. Another major drawback of this model is the lack of a clear way to interpret the model predictions. In fact, despite the apparent simplicity of the message passing mechanism, there is no intuitive way to visualize the information learned by the network, similarly for example to what can be done with the convolutional filters of CNNs.

In an effort to boost the power of GNNs, a number of researchers have started to move away from the standard message passing. For example, Bouritsas *et al.* [5] show that one can improve the ability of the network to discriminate between non-isomorphic graphs by augmenting the graph nodes attributes with positional encoding. Other works propose using different positional features [20], as centrality measures [15]. Similarly, Bevilaqua *et al.* [2] propose to represent graphs as bags of substructures in order to boost the discriminative power of their model. However, more recently, other researchers [3,4,6,8,18] have taken a radically different approach, which not only improves the expressive power of GNNs but also adds an inherent interpretability mechanism in the network architecture. This is achieved by using (graph) kernels [6] to define a mechanism similar to the convolution of CNNs, *i.e.*, to measure the similarity between the input graph and a set of latent (structural) filters. While these approaches provide a relatively easy way to interpret the model decisions by visualising the learned latent structures, this often comes at the cost of an increased complexity.

3 Background

Consider an undirected, unweighted graph $\mathcal{G} = (V, E)$, where V represents a set of n nodes and E represents the set of edges connecting these nodes. The graph is characterized by its adjacency matrix $\mathbf{A}$, a symmetric binary matrix. In $\mathbf{A}$, $\mathbf{A}_{i,j} = 1$ if there is an edge between the i-th and j-th nodes, and $\mathbf{A}_{i,j} = 0$ otherwise. Each node $v \in V$ can also have an associated feature vector $\mathbf{x}_v \in \mathbb{R}^d$, where d denotes the feature dimension. These feature vectors form the node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, with the i-th row of $\mathbf{X}$ corresponding to the feature vector of the i-th node.

Graph Neural Network. A GNN is a neural network designed to learn from graph data by learning meaningful representations of graph nodes ($\mathbf{h}_v$) or of entire graphs ($\mathbf{h}_G$). This in turn is achieved by leveraging both structural and feature information. The information is mainly processed through two steps: (1) an aggregation step, where each node gathers features from its neighbors, and (2) an update step, usually involving a linear transformation followed by a nonlinear activation function, where the nodes update their representations using the information they have gathered in the previous step. In this context, if $\mathbf{h}_v^{(i)}$ represents the representation of node v at layer i, $\mathbf{h}_v^{(i+1)}$ indicates the updated one.

Variational AutoEncoder. Autoencoders are neural network architectures used to learn efficient data representations, consisting of an encoder and a decoder. In the context of graphs, given an input $\mathcal{G}$ the encoder compresses

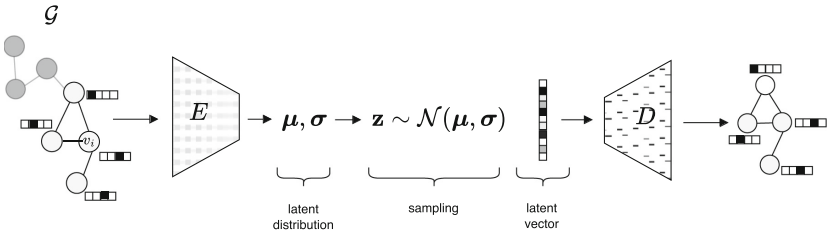


Fig. 1. Graph Variational AutoEncoder. Starting from an attributed graph, the egonet of node v is extracted. This egonet includes the features of node v and its neighboring nodes. The graph encoder processes the subgraph to produce the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ of the normal distribution underlying the latent space, from which a latent variable $\mathbf{z}$ is sampled. The obtained continuous representation $\mathbf{z}$ of the egonet is then used by the decoder to reconstruct the input subgraph.

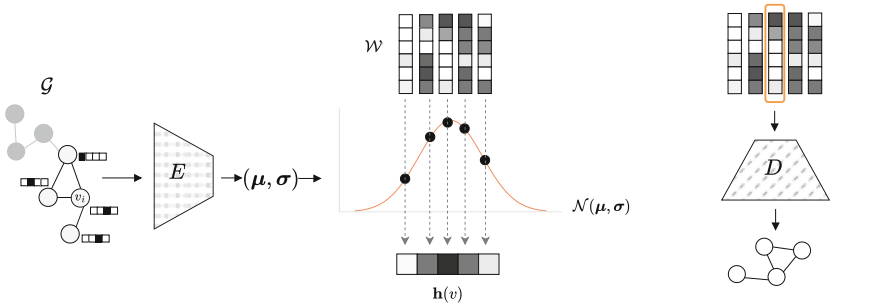
its adjacency matrix $\mathbf{A}$ and node feature matrix $\mathbf{X}$ into a latent space representation $\mathbf{Z}$. The decoder then attempts to reconstruct the original adjacency and feature matrices from $\mathbf{Z}$, aiming to minimize the reconstruction error. A Variational AutoEncoder (VAE) [12] extends this by introducing a generative approach. Instead of mapping the input directly to a latent space, the encoder produces parameters for a probability distribution in the latent space, specifically the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of a multivariate normal distribution with independent components. A latent variable $\mathbf{z}$ is sampled from this distribution and used by the decoder to reconstruct the input. The loss function in a VAE includes both the reconstruction loss and a Kullback-Leibler (KL) divergence term, which measures how closely the learned latent distribution approximates the prior distribution (usually a standard normal distribution), *i.e.*,

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}|\mathbf{A},\mathbf{X})}[-\log p(\mathbf{A},\mathbf{X}|\mathbf{z})] + \mathrm{KL}(q(\mathbf{z}|\mathbf{A},\mathbf{X})\|p(\mathbf{z})), \tag{1}$$

where $q(\mathbf{z}|\mathbf{A},\mathbf{X})$ is the variational posterior and $p(\mathbf{z})$ is the prior. This ensures that the VAE not only reconstructs the input data accurately but also generates meaningful latent representations [23].

4 Our Method

The idea underpinning our approach is to express a novel graph "convolutional" operator, *i.e.*, an operator that updates the feature representation of the graph nodes by taking into account both structural and feature information. Specifically, we aim to define this as a function operating in the latent space of graphs. This in turn allows our method to achieve a high degree of interpretability by learning latent representations and identifying influential input graph patterns while being significantly faster than interpretable GNNs based on graph kernels [6]. This aspect is especially useful in fields like molecular science, where understanding the defining atoms and bonds of molecule families is crucial. In



(a) Convolution Operation (b) Interpretation

Fig. 2. (a) Proposed Convolution Operation. The convolution updates the node representation by evaluating the likelihood that a set of learned latent vectors $\mathcal{W}$ is drawn from the Gaussian distribution defined by the encoder outputs $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. (b) Interpretation step. Learned latent vectors are decoded into graphs.

the following we describe the four main ingredients of our model, namely the pretraining of (1) the encoder-decoder architecture that allows us to map graphs to the latent space, (2) the convolution operation based on this space, (3) the mechanism to interpret the learned latent vectors, (4) and the loss.

Graph Latent Space. Our model relies on learning a latent space that enables both the embedding of existing graph substructures and the generation of a graph from its latent representation. We achieve this using a GVAE [23]. As shown in Fig. 1, the GVAE is trained to encode a set of subgraphs extracted from the dataset into a normal distribution in the latent space, represented by a mean $\boldsymbol{\mu}$ and a standard deviation $\boldsymbol{\sigma}$. This distribution is later used to sample a latent variable $\mathbf{z}$, which is decoded into a new graph, ideally isomorphic to the input one. In our implementation, we extracted the subgraphs by considering the induced k-hop subgraph centered at each node of all the graphs in the training set. The architecture of our GVAE is largely similar to [23], with the exception of minor differences in the implementation details. We refer the reader to [23] for details on the GVAE architecture. Moreover, we make the code of our model (including the GVAE) publicly available online in the interest of reproducibility[1].

Graph Convolution in the GVAE Latent Space. With the GVAE to hand, we develop LESI-GNN Latent Embedding of Subgraphs for Interpreting Graph Neural Networks, a simple yet effective graph classification model. It consists of a single convolutional layer and leverages the key features of the GVAE. Specifically, given a node of the input graph and the k-hop subgraph centered on it (*i.e.*, the node egonet), the proposed convolution updates the node representation by (1) mapping the egonet to its corresponding probability distribution using the GVAE and (2) measuring the probability that a set of learned latent vectors belongs to this distribution.

[1] https://github.com/giorgiamean/graphvae4motifs.

More specifically, for each node $v \in \mathcal{G}$, we extract the egonet $\mathcal{S}_v^k$ of node v. This subgraph is composed by node v (the center), all its neighborhood nodes within a distance of k hops, and the edges connecting them. The egonet includes the features of node v and its neighboring nodes. Given a pretrained GVAE, we use its encoder to process the egonets, mapping each egonet to the parameters $\boldsymbol{\mu}_v, \boldsymbol{\sigma}_v$ of its corresponding latent distribution, *i.e.*,

$$(\boldsymbol{\mu}_v, \boldsymbol{\sigma}_v) = \text{Encoder}(\mathcal{S}_v^k).$$

Next, we compute the convolution response as the *unnormalized* probability of the i-th learned vector $\mathbf{w}_i$ to belong to that particular distribution, leading to the updated node representation $\mathbf{h}(v) = [\mathbf{h}_1(v), \ldots, \mathbf{h}_M(v)]$, with the i-th element $\mathbf{h}_i(v)$ computed as

$$\mathbf{h}_i(v) = \exp\left(-\frac{1}{2}(\mathbf{w}_i - \boldsymbol{\mu}_v)\boldsymbol{\Sigma}_v^{-1}(\mathbf{w}_i - \boldsymbol{\mu}_v)^\top\right), \tag{2}$$

where $\boldsymbol{\Sigma}_v$ is a diagonal matrix with diagonal values equal to $\boldsymbol{\sigma}_v^2$. Equation (2) measures the probability that the learned latent vectors belong to the distribution created by the GVAE, as shown in Fig. 2a. Note that the M weight latent vectors $\mathbf{h}_1(v), \ldots, \mathbf{h}_M(v)$ are the only parameters of our convolution. Once all the node representations $\mathbf{h}_v$ are computed, they are aggregated to produce an overall representation $\mathbf{h}_\mathcal{G}$ for the graph $\mathcal{G}$, which is then used to complete the task at hand (*i.e.*, graph classification, in the context of this paper).

Normal Distribution Loss. While we interpret the learned vectors as compact graph representations in the latent space of the GVAE, during the optimization they could drift outside of the subspace in which the GVAE encodes the graphs. In particular, the KL loss used during the training of the autoencoder promotes the latent distribution of the graphs to resemble a standard normal distribution. To ensure that also the learned latent vectors follow a similar prior, we regularize the training of the classifier with an additional loss $\mathcal{L}_{dist}$, which minimizes the squared difference between the standard normal and the kernel density estimation of the distribution of the learned vectors, *i.e.*,

$$\mathcal{L}_{dist} = \int \left(\phi_{\mathbf{0},I}(x) - \frac{1}{M}\sum_{i=1}^{M}\phi_{\mathbf{w}_i,\boldsymbol{\sigma}^2 I}(x)\right)^2 dx =$$

$$\frac{1}{\sqrt{(2\pi)^D}}\left(\frac{1}{\sqrt{2^D}} - \frac{2}{M}\sqrt{\frac{(\boldsymbol{\sigma}^2)^D}{(1+\boldsymbol{\sigma}^2)^D}}\sum_{i=1}^{M} e^{-\frac{\|\mathbf{w}_i\|^2}{2+2\boldsymbol{\sigma}^2}} + \frac{\sqrt{(\boldsymbol{\sigma}^2/2)^D}}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M} e^{-\frac{\|\mathbf{w}_i-\mathbf{w}_j\|^2}{4\boldsymbol{\sigma}^2}}\right),$$

where $\phi_{\boldsymbol{\mu},\boldsymbol{\sigma}^2}$ is the multivariate normal distribution with $\boldsymbol{\mu}$ mean and a diagonal covariant matrix with $\boldsymbol{\sigma}^2$ diagonal entries, and $\phi_{\mathbf{0},I}$ is the standard normal distribution. For simplicity, we assume unit variances. D is the dimensionality of the latent space. $\mathcal{L}_{dist}$ is added to the cross entropy loss to train the network.

Latent Vectors Interpretation. The proposed architecture allows us to interpret the predictions made by LESI-GNN in terms of the substructures learned

Table 1. Avg. classification accuracy ± std error on the 5 datasets considered in this study. The best (second best) model for each dataset is highlighted in bold (underlined). The results reported for GKNN are from [6].

Dataset	MUTAG	PTC	PROTEINS	NCI1	AIDS
GCN	75.99 ± 3.5	56.11 ± 2.8	71.60 ± 0.8	74.53 ± 0.7	94.55 ± 0.4
DGCNN	85.03 ± 2.7	50.60 ± 2.2	73.86 ± 0.7	74.38 ± 0.7	<u>99.40 ± 0.1</u>
DiffPool	81.35 ± 2.5	57.83 ± 2.3	71.25 ± 1.5	**79.32 ± 0.9**	99.15 ± 0.1
ECC	80.82 ± 4.2	50.87 ± 2.8	71.17 ± 1.0	75.82 ± 0.5	96.05 ± 0.2
GIN	80.26 ± 3.1	58.69 ± 1.6	<u>74.76 ± 0.9</u>	<u>78.89 ± 0.6</u>	99.35 ± 0.2
GraphSAGE	72.22 ± 2.8	55.78 ± 2.7	71.25 ± 1.5	77.44 ± 0.7	97.40 ± 0.3
GKNN	**85.73 ± 2.7**	<u>59.29 ± 2.5</u>	**74.94 ± 1.1**	73.60 ± 1.3	NA
LESI-GNN (ours)	<u>85.61 ± 2.3</u>	**60.0** ± 1.8	74.49 ± 1.5	67.61 ± 1.3	**99.60 ± 0.1**

by the network. While the vectors themselves are hard to interpret, our architecture allows us to convert them into graphs using the decoder of the pretrained GVAE. This in turn is made possible by effectively nudging the latent vectors to lie in the latent space of GVAE. The result is the simple yet effective interpretation mechanism shown in Fig. 2b, where the learned latent substructures play a role equivalent to that of the convolutional filters learned by a CNN.

5 Experiments

We compare our model against six state-of-the-art GNNs: GCN [13], DGCNN [30], DiffPool [28], ECC [22], GIN [26], and GraphSAGE [11]. To these we add GKNN [6], a recently introduced kernel-based interpretable GNN. We compare these methods on five commonly used biological/chemical datasets for graph classification tasks: MUTAG, PTC, PROTEINS, NCI1, and AIDS All node features are encoded using one-hot encoding.

We evaluate the performance of each model using 10-fold cross-validation, with a 90/10 training+validation/test split, further splitting train+validation into training (90) and validation (10). We use the training/validation sets to select the optimal model (lowest loss on validation set) configurations through grid search. To establish the hyperparameter search space, we follow the ranges specified in [7]. For our model, we vary: pooling (mean, sum), KL weight (1, 10), and number of hops (1, 2), for the GVAE pretraining; model dimension (16, 32, 64) and number of MLP layers (2, 3) for the classifier part.

We report the average accuracy ± standard error in Table 1. With the exception of NCI1, our model performs favourably in all the datasets considered, with the added benefit of allowing us to interpret the model predictions by visualizing the latent structures learned by the network. This is also possible in GKNN, however it comes at a higher computational cost, as GKNN operates directly in the graph space (rather than a vectorial space) and to do so requires expensive numerical approximations of the gradients (see Sect. 5.2).

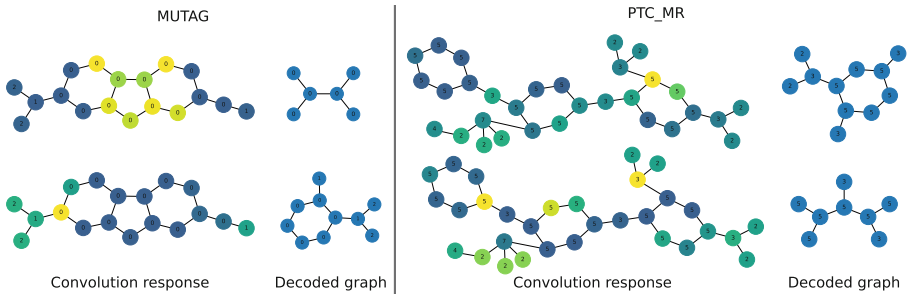


Fig. 3. Decoded graphs and convolution response of some learned latent vectors on MUTAG and PTC datasets

5.1 Interpretability

As discussed in Sect. 4 and illustrated in Fig. 2b, the weights learned by our method are vectors in the graph latent space. This allows us to generate the corresponding graph using the GVAE decoder. In Fig. 3, we visualize some of the decoded latent vectors learned during the training on MUTAG and PTC. We show both the graph generated by decoding the latent vector and the response of our convolution on the egonet centered on each node of one of the training graphs as described in Eq. (2).

5.2 Runtime

We compare the runtime of LESI-GNN with that of GKNN, which provides a similar level of interpretability. We are significantly faster during both training and inference time. Our training time on MUTAG is 1.24 and 1.23 s per epoch for the GVAE and the classifier, respectively, compared to 10.01 s for GKNN. On the larger dataset, NCI1, our training time per epoch is 18.32 (GVAE) and 1.66 s (classifier), versus 83.45 s on for GKNN. Similarly, our inference time for one sample in both datasets is below 1 ms, compared to 16.62 ms and 8.02 ms for GKNN.

6 Conclusion

We introduced LESI-GNN, an interpretable GNN where the core idea is that of performing the convolution between local substructures and latent vectors in the embedding space learned by a GVAE. This enables us to visualize the learned latent vectors as structural patterns in order to interpret the model decision. While allowing a certain degree of interpretability, LESI-GNN is still able to perform better or on-par with widely used GNNs. The major limitation of the proposed architecture is that it only allows to perform one layer of convolutions. Future work will investigate ways to overcome this limitation.

Acknowledgments. This project is partially supported by the European Union with the program Next-GenerationEU. In particular, L.C. and A.B. are supported by PNRR - M.4 C.2, Investment 1.1 PRIN 2022 - project n. 2022AL45R2, EYE-FI.AI, CUP H53D2300350-0001. G.M. is supported by iNEST - Interconnected Nord-Est Innovation Ecosystem (iNEST ECS_00000043 - CUP H43C22000540006). A.T.'s work was partially supported by the project "Perturbation problems and asymptotics for elliptic differential equations: variational and potential theoretic method" - PRIN 2022 - grant n. 2022SENJZ3.

References

1. Bai, L., Cui, L., Jiao, Y., Rossi, L., Hancock, E.: Learning backtrackless aligned-spatial graph convolutional networks for graph classification. IEEE Trans. Pattern Anal. Mach. Intell. (2020)
2. Bevilacqua, B., et al.: Equivariant subgraph aggregation networks. In: International Conference on Learning Representations (2022)
3. Bicciato, A., Cosmo, L., Minello, G., Rossi, L., Torsello, A.: Classifying me softly: a novel graph neural network based on features soft-alignment. In: S+SSPR, pp. 43–53. Springer (2022)
4. Bicciato, A., Cosmo, L., Minello, G., Rossi, L., Torsello, A.: GNN-LoFI: a novel graph neural network through localized feature-based histogram intersection. Pattern Recogn. **148**, 110210 (2024)
5. Bouritsas, G., Frasca, F., Zafeiriou, S.P., Bronstein, M.: Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Trans. Pattern Anal. Mach. Intell. (2022)
6. Cosmo, L., et al.: Graph kernel neural networks. IEEE Trans. Neural Netw. Learn. Syst. (2024)
7. Errica, F., Podda, M., Bacciu, D., Micheli, A.: A fair comparison of graph neural networks for graph classification. arXiv preprint arXiv:1912.09893 (2019)
8. Feng, A., You, C., Wang, S., Tassiulas, L.: Kergnns: interpretable graph neural networks with graph kernels. arXiv preprint arXiv:2201.00491 (2022)
9. Gallagher-Syed, A., et al.: Multi-stain self-attention graph multiple instance learning pipeline for histopathology whole slide images. arXiv preprint arXiv:2309.10650 (2023)
10. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (2017)
11. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
14. Lima, A., Rossi, L., Musolesi, M.: Coding together at scale: github as a collaborative social network. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 8, pp. 295–304 (2014)
15. Lockhart, J., Minello, G., Rossi, L., Severini, S., Torsello, A.: Edge centrality via the holevo quantity. In: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+ SSPR 2016, Mérida, Mexico, 29 November–2 December 2016, Proceedings, pp. 143–152. Springer (2016)

16. Mehrish, A., Majumder, N., Bharadwaj, R., Mihalcea, R., Poria, S.: A review of deep learning techniques for speech processing. Inf. Fusion **99** (2023)
17. Morris, C., et al.: Weisfeiler and leman go neural: higher-order graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4602–4609 (2019)
18. Nikolentzos, G., Meladianos, P., Tixier, A.J.P., Skianis, K., Vazirgiannis, M.: Kernel graph convolutional neural networks. In: International Conference on Artificial Neural Networks, pp. 22–32. Springer (2018)
19. Roscher, R., Bohn, B., Duarte, M.F., Garcke, J.: Explainable machine learning for scientific insights and discoveries. IEEE Access **8**, 42200–42216 (2020)
20. Said, A., Ahmad, O.U., Abbas, W., Shabbir, M., Koutsoukos, X.: Improving graph machine learning performance through feature augmentation based on network control theory. arXiv preprint arXiv:2405.03706 (2024)
21. Senior, H., Slabaugh, G., Yuan, S., Rossi, L.: Graph neural networks in vision-language image understanding: a survey. Vis. Comput. 1–26 (2024)
22. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3693–3702 (2017)
23. Simonovsky, M., Komodakis, N.: Graphvae: towards generation of small graphs using variational autoencoders. In: Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, 4–7 October 2018, Proceedings, Part I 27, pp. 412–422. Springer (2018)
24. Sperduti, A., Starita, A.: Supervised neural networks for the classification of structures. IEEE Trans. Neural Netw. **8**(3), 714–735 (1997)
25. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. Stat **1050**(20), 10–48550 (2017)
26. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
27. Yi, H.C., You, Z.H., Huang, D.S., Kwoh, C.K.: Graph representation learning in bioinformatics: trends, methods and applications. Briefings Bioinform. **23**(1), bbab340 (2022)
28. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
29. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: a taxonomic survey. IEEE Trans. Pattern Anal. Mach. Intell. **45**(5), 5782–5799 (2022)
30. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

Mixture of Variational Graph Autoencoders

Alessandro Bicciato[1(✉)], Edwin Hancock[2], Richard Wilson[2], and Andrea Torsello[1]

[1] Ca' Foscari University of Venice, Venice, Italy
alessandro.bicciato@unive.it
[2] University of York, York, North Yorkshire, UK

Abstract. Autoencoders, a type of unsupervised model, are capable of learning effective latent representations of data without supervision, only requiring the decoder to be able to reconstruct the original datapoint from its latent representation obtained through the encoder. When dealing with structured data, graph-autoencoders are one of the few effective ways to obtain such latent representations of graphs. However, when dealing with large, diverse graph datasets, autoencoders struggle to adapt to varying structures, leading to suboptimal encoding. In our paper, we introduce a novel approach called Mixture of Variational Graph Autoencoders, which addresses this limitation by introducing a mixture of encoder/decoder models which provide multiple local and class-specific models that better adapt to different patches of the dataspace. An exhaustive experimental evaluation shows that our approach greatly outperforms the state of the art in reconstruction precision (Code: https://github.com/gdl-unive/MVGAE).

Keywords: Variational Autoencoder · Deep learning

1 Introduction

In the current high-demand landscape for machine learning algorithms, the need to annotate large datasets is more apparent than ever. Unsupervised methods, which allow to train models on unlabeled data, offer a potential solution to this challenge.

Autoencoders are an example of unsupervised models trained to reconstruct the input data. Comprising two essential components, the encoder and the decoder, the autoencoder system compresses and encodes data in the encoding phase, transforming the original data into an alternative (latent) representation space and extracting meaningful information for feature extraction. On the other hand, the decoder is tasked with restoring the encoded data to its original representation, minimizing the error between the original and the decoded data. Through the encoder, autoencoders learn an effective representation, thus addressing the issues of inadequate feature extraction encountered in manual

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 82–91, 2025.
https://doi.org/10.1007/978-3-031-80507-3_9

approaches while mitigating the risk of overfitting. The classical autoencoder was initially introduced by Rumelhart et al. [16], with a comprehensive description provided by Bourlard et al. [1].

Despite their advantages, autoencoders exhibit some limitations, including the prolonged training time of deep models due to layer-by-layer training, limited interpretability of extracted features, and, arguably more poignantly, struggle to adapt to varying structures when dealing with large, diverse graph datasets, leading to suboptimal encoding.

In this paper we introduce a new approach called mixture of variational graph autoencoders (MVGAE). This approach incorporates a mixture model of encoders during the encoding process, allowing for multiple independent models to better deal with a different part of the data-space. All encoders encode to the same latent space and, thus, can be decoded by the same decoding process. The encoder to use for each graph is selected utilizing a neural network that is trained to map graphs into the different mixture components. The main goal of the mixture model is to effectively group the graphs into different clusters and allow for specialized encoders for each group. This way we compose simple local models that work well in different areas of the data-space, in order to provide a globally-optimal encoding.

Further, under the assumption that graphs belonging to the same class share similar structures, we push each component to be class-specific by associating a label model to each component. Under this model the decoded label is independent on the input graph/latent vector, conditioned on the mixture component, thus pushing each component to encode graphs belonging to a single class, thus inducing a task-driven component separation.

The paper is organized as follows: Sect. 2 reviews the current state of the art and related methods. In Sect. 3, we describe the proposed model, and in Sect. 4, we discuss the results of our experimental evaluations. Finally, in Sect. 5, we offer some conclusions.

2 State of the Art

In recent years, the conventional autoencoder model has undergone great development, from denoising autoencoders to sparse or convolutional ones. The autoencoder model is an efficient coding approach that facilitates learning and extracting crucial features. Implemented as a neural network, it is designed to reconstruct its input signal, operating as an unsupervised learning model that does not require data labels.

The *denoising autoencoder* (DAE) is an extension of the traditional autoencoder [18]. It is inspired by human perceptual capabilities and involves introducing a noise layer following the input layer. This allows the model to learn to reconstruct clean data from noisy inputs. The *sparse autoencoder* [12] introduces a sparsity constraint to enforce a level of sparsity in the activation of neurons within the hidden layer.

With the emergence of convolutional neural networks, *convolutional autoencoders* have also been developed [15] where convolutions are introduced to replace

the fully connected layers in conventional autoencoders, and down(up)-sampling offer the equivalen of a pooling layer. These operations effectively preserve spatial information.

Kingma et al. introduced the Variational Autoencoder (VAE) [8], which operates on the fundamental principle of mapping a dataset into an ideal Gaussian distribution using an encoder. Subsequently, samples drawn from this Gaussian distribution are fed into the decoder to generate reconstructed data. VAE is a data generation model grounded in variational Bayesian inference, in which the encoder processes the input X to obtain μ and σ, introducing a Gaussian distribution ε to derive the probability encoding Z. Finally, the decoder decodes Z to reconstruct X.

Kipf and Welling recently introduced variational autoencoders to the graph domain [10]. Variational Graph Autoencoders (VGAE) offer a framework for unsupervised learning on graph-structured data. This model makes use of latent variables and is capable of learning interpretable latent representations for undirected graphs. The model [10] provides a node-covariant autoencoder, *i.e.*, a model that encodes each node in a common latent space. Under this model the decoder is not trained and performs the edge prediction by taking the dot product between the encoded vectors. More precisely, let $A = (a_{ij})$ be the decoded adjacency matrix, we have

$$P(a_{ij} = 1|Z) = \text{sigmoid}\left(\mathbf{z}_i^T \mathbf{z}_j\right), \tag{1}$$

where $\mathbf{z}_i$ is the encoded vector for node i. The role of the encoder is then to produce encoded vectors such that this decoding procedure reproduces the input graph.

This model shifts all the complexity on the encoder and works well for relatively simple problems, however it exhibit a scaling and generalization limit when dealing with more complex tasks. In fact, due to the highly multimodal nature of the datasets, a single simple model often fails to capture all their characteristics and variations of the data, and larger and more complex models are required to encode the nodes with the necessary information to correctly estimate the links. However, these models are harder to train and are subject to over-fitting.

Despite recent simplifications and improvements [2,4,7,14,17], self-supervised GAEs have not progressed as much as contrastive learning. So far, no GAEs have been able to clearly outperform contrastive SSL methods, particularly in node and graph classifications, which have been significantly enhanced by neural encoders like graph neural networks.

Over the years, several challenges have come to light. Many graph autoencoders (GAEs) use link reconstruction as the primary objective to promote closeness between neighboring nodes [2,5,10,13,17,19]. Consequently, previous GAEs excel at link prediction and node clustering but fall short in node and graph classifications. Additionally, feature reconstruction without corruption may lack robustness. Despite the use of feature reconstruction in GAEs [2,10,13,14,17], many still rely on basic architectures, posing the risk of learning trivial solutions. However, denoising autoencoders [18], which corrupt input and then attempt to

recover it, are widely used in NLP [11], and this approach may also be applicable to graphs.

One approach to address these challenges is GraphMAE [3], a masked graph autoencoder, which has been developed to mitigate the difficulties encountered in generative self-supervised graph pretraining. Unlike conventional methods that focus solely on reconstructing graph structures, GraphMAE places emphasis on reconstructing features using a masking strategy and scaled cosine error.

3 Method

In order to mitigate the complexity/generalization problem of VGAEs, we subdivided the learned model into smaller local models, where each local model captures a specific segment of the data space. We do this by modeling the variational encoder as a mixture of simple Kipf-Welling encoders, each designed to explain a small portion of the data space. This approach helps capture the dataset's diverse components by assigning different models to different graphs, improving overall interpretability and performance.

Here we concentrate on graph-encoding problems where the goal is not just link prediction, but where we naturally have a class label associated with each graph. We map each graph to one mixture component through a GCN classifier [9], which is trained alongside the autoencoder. We facilitate the separation of the graphs by positing that each component should model graphs that belong to one particular class; while more components can be assigned to the same class, and there can be spatial overlap in the latent space between distinct components. We obtain this by adding a conditionally independent class-label model to each component so that the decoder can produce a class label alongside the graph, but the label depends only on the originating component, which only indirectly depends on the input graph through the classifier. The MVGAE is then trained by enforcing the reconstruction of both the graph structure and the class label of the input graph.

Note that the class label is not used by the encoder or the classifier, thus the label is used in training but not needed in prediction. Further, the label produced by the decoder is used to facilitate task-driven component separation and is not meant as a classifier for the generated graph, and can safely be discarded during prediction.

Figure 1 illustrates our autoencoding scheme. The GCN classifier produces a probability distribution π, balanced with a Gumbel Softmax [6] that, in the training step, has a variable temperature τ, which is a dynamic hyper-parameter.

The essence of a variational autoencoder lies in the use of a variational distribution q_ϕ for which we choose the parameters ϕ to best fit the target distribution $p(\mathbf{Z})$ of the latent vector $\mathbf{z}$. We also fit the data likelihood to best reconstruct the input from the proposed latent space distribution, resulting in the variational lower bound

$$\mathcal{L}(\theta, \phi; \mathbf{X}, \mathbf{A}) = -D_{KL}(q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{A}), p_\theta(\mathbf{Z})) + \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{A}|\mathbf{Z})\right], \tag{2}$$

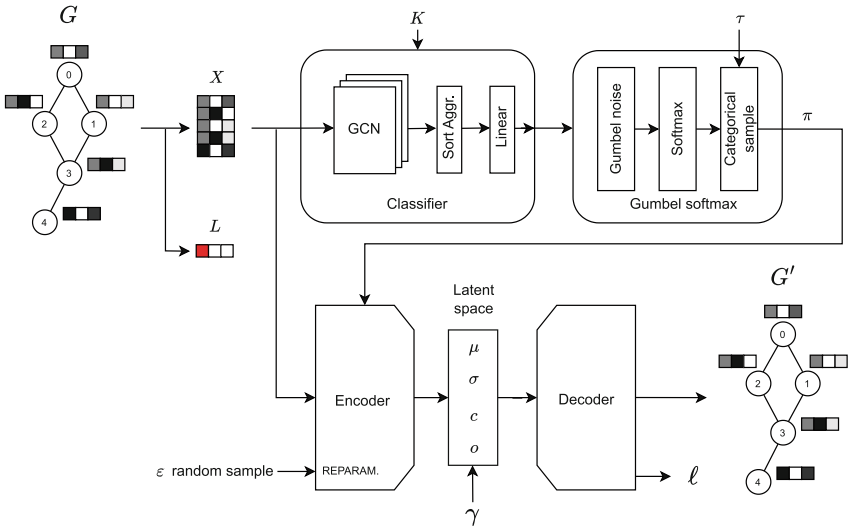


Fig. 1. Schema for the mixture variational autoencoder. For each graph G, we have X and L, representing the input data of the encoder and the one hot encoded graph label, respectively. In the mixture part, we have a classifier that gets in input X and K (the number of mixture components), and to the output is computed a Gumbel Softmax with variable temperature τ, producing the weight vectors π. In the latent space, we will have μ and σ from the encoding of X projected to a Gaussian distribution ε, the mixture components c, and the classification of the mixture components o. We introduce the learned vector γ for each label of L associated with the mixture component. The decoder will generate G' and the observed label ℓ.

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ represents the D features of the N nodes of a graph and $\mathbf{A}$ its adjacency matrix. The first term ensures the closeness of q_ϕ to the prior $p_\theta(\mathbf{Z})$ and the second term that the input is correctly generated from the latent space. In what follows, first we introduce the model with the mixture of encoders, and then we add label encoding.

3.1 Mixture Model

We want to build a mixture model to help the encoding, *i.e.*, we augment the latent space with a class variable $\mathbf{c} = \{0, 1\}^K$, such that $c_k = 1$ if the graph belongs to mixture component k. Under these assumptions, we identify the distributions that define the encoder, the decoder, and the latent space. The latent space distribution $p_\theta(\mathbf{Z}, \mathbf{c})$ is determined to be isotropic and assumes all mixture components are equally probable *apriori*:

$$p_\theta(\mathbf{Z}, \mathbf{c}) = \prod_i \mathcal{N}(\mathbf{z}_i; \mathbf{0}, \mathbf{I}) \prod_k \left(\frac{1}{K}\right)^{c_k}. \tag{3}$$

The decoder distribution follows the Kipf and Welling definition for reconstructing the latent space by computing the dot product

$$p_\theta(\mathbf{A}|\mathbf{Z},\mathbf{c}) = \prod_{i,j} \text{sigmoid}\left(\mathbf{z}_i^T \mathbf{z}_j\right), \tag{4}$$

while for the encoder distribution we have

$$q_\phi(\mathbf{Z},\mathbf{c}|\mathbf{X},\mathbf{A}) = q_{\phi,1}(\mathbf{Z}|\mathbf{c},\mathbf{X},\mathbf{A}) q_{\phi,2}(\mathbf{c}|\mathbf{X},\mathbf{A}). \tag{5}$$

The first term is the probability of the encoding, given adjacency matrix $\mathbf{A}$ and the node features $\mathbf{X}$

$$q_{\phi,1}(\mathbf{Z}|\mathbf{c},\mathbf{X},\mathbf{A}) = \prod_i \mathcal{N}\left(\mathbf{z}_i; \boldsymbol{\mu}_i(\mathbf{c},\mathbf{X},\mathbf{A}), \text{diag}\left(\mathbf{v}_i(\mathbf{c},\mathbf{X},\mathbf{A})\right)\right). \tag{6}$$

We assume the distribution to be Normal with average $\boldsymbol{\mu}_i$ and variance $\text{diag}(\mathbf{v}_i)$, all conditioned to the mixture component and the graph.

The second term is the distribution over the mixture components

$$q_{\phi,2}(\mathbf{c}|\mathbf{X},\mathbf{A}) = \prod_k \pi_k(\mathbf{X},\mathbf{A})^{c_k}, \tag{7}$$

where π is the assignment to the components operated by the classifier. Here the use of a GCN and sort-pooling guarantee invariance to node order.

The Kullback-Leibler divergence term of the variational lower bound then becomes

$$\begin{aligned} D_{KL}(q_\phi||p(\mathbf{Z},\mathbf{c})) &= \sum_c \int q_\phi(\mathbf{Z},\mathbf{c}|\mathbf{X},\mathbf{A}) \left[\log q_\phi(\mathbf{Z},\mathbf{c}|\mathbf{X},\mathbf{A}) - \log p(\mathbf{Z},\mathbf{c})\right] d\mathbf{Z} \\ &= \sum_k \pi_k \log \pi_k + \log K + \frac{1}{2} \sum_{ij} \left(\mu_{ij}^2 + \sigma_{ij}^2 - 1 - \log \sigma_{ij}^2\right). \end{aligned} \tag{8}$$

3.2 Label Encoding

In order to use task-specific information to drive component separation and enforce that each component encodes graphs from a single class, we add a label model to each component. To this end, we augment the latent space with a class variable $\mathbf{o} = \{0,1\}^{K \times L}$, with $\forall k, \sum_l \mathbf{o}_{k,l} = 1$, *i.e.*, $\mathbf{o}_{k,l} = 1$ if the k-th component generates graphs with label l, 0 otherwise. The constraint is one-sided, so we can have more than one mixture component generating the same label.

Note that the generated graph depends on $\mathbf{c}$, but not on $\mathbf{o}$. On the other hand, the *observed* label ℓ depends only on which mixture component the graph is generated from, and can be obtained as $\ell_l = \sum_k \mathbf{c}_k \mathbf{o}_{k,l}$, while it is independent to the encoded graph, conditioned to the component. Of course, the chosen component depends on the input graph through the classifier, which, however,

does not see the ground-truth labels, and is trained to separate the labels by the gradients back-propagated from each component's label model.

Hence, we define the decoder probability as

$$p_\theta(\mathbf{A}, \ell|\mathbf{Z}, \mathbf{c}, \mathbf{o}) = \left(\prod_{i,j} \text{sigmoid}(\mathbf{z}_i^T \mathbf{z}_j)\right) \cdot \left(\sum_k \mathbf{c}_k \mathbf{o}_{k,l}\right). \tag{9}$$

Note that the decoder produces both the graph and the labels. The graph is conditionally independent of $\mathbf{c}$ and $\mathbf{o}$, given $\mathbf{Z}$, while the label, depending only on the mixtures, can be obtained by marginalizing $\mathbf{co}$ over the mixtures. Adding the labels, we assume the prior

$$p_\theta(\mathbf{Z}, \mathbf{c}, \mathbf{o}) = \prod_i \mathcal{N}(\mathbf{z}_i; \mathbf{0}, \mathbf{I}) \prod_k \prod_l \left(\frac{1}{KL}\right)^{\mathbf{c}_k \mathbf{o}_{k,l}}, \tag{10}$$

which is isotropic and assumes all mixtures and all labels are equally probable *apriori*.

For the encoder distribution, we have

$$q_\phi(\mathbf{Z}, \mathbf{c}, \mathbf{o}|\mathbf{X}, \mathbf{A}) = q_{\phi,1}(\mathbf{Z}|\mathbf{c}, \mathbf{X}, \mathbf{A}) q_{\phi,2}(\mathbf{o}, \mathbf{c}|\mathbf{X}, \mathbf{A}). \tag{11}$$

The first term is identical to the model without the labels, and gives the distribution of the latent variable $\mathbf{Z}$

$$q_{\phi,1}(\mathbf{Z}|\mathbf{c}, \mathbf{X}, \mathbf{A}) = \prod_i \mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}_i(\mathbf{c}, \mathbf{X}, \mathbf{A}), \text{diag}(\mathbf{v}_i(\mathbf{c}, \mathbf{X}, \mathbf{A}))). \tag{12}$$

The second term $q_{\phi,2}(\mathbf{o}, \mathbf{c}|\mathbf{X}, \mathbf{A})$ is the mixture-label predictor distribution, *i.e.*,

$$q_{\phi,2}(\mathbf{o}, \mathbf{c}|\mathbf{X}, \mathbf{A}) = \prod_k \prod_l \left(\pi_k(\mathbf{X}, \mathbf{A}) \gamma_{k,l}\right)^{\mathbf{c}_k \mathbf{o}_{k,l}}, \tag{13}$$

where π is the component assignment operated by the classifier network, and $\gamma_{k,l}$ encodes the probability of generating label l from the component k. Note that this is just a property of each mixture component and is independent on the observed graph.

The Kullback-Leibler divergence term of the variational lower bound then becomes

$$\begin{aligned} D_{KL}(q_\phi||p(\mathbf{Z}, \mathbf{c})) &= \sum_c \int q_\phi(\mathbf{Z}, \mathbf{c}, \mathbf{o}|\mathbf{X}, \mathbf{A}) [\log q_\phi(\mathbf{Z}, \mathbf{c}, \mathbf{o}|\mathbf{X}, \mathbf{A}) - \log p(\mathbf{Z}, \mathbf{c}, \mathbf{o})] d\mathbf{Z} \\ &= \sum_k \pi_k \log \pi_k + \log K + \sum_k \pi_k \sum_l \gamma_{k,l} \log \gamma_{k,l} + K \log L + \\ &\quad \frac{1}{2} \sum_{ij} (\mu_{ij}^2 + \sigma_{ij}^2 - 1 - \log \sigma_{ij}^2). \end{aligned} \tag{14}$$

4 Results

We compare our model (MVGAE) to two state-of-the-art methods: the original VGAE of Kipf and Welling [10] and a modified version of GraphMAE [3] to perform a link prediction task. Most autoencoder algorithms that perform link prediction in state-of-the-art are usually designed for dictionary-like datasets and are not well-suited for our task with graph datasets.

We used five graph datasets commonly used in graph classification literature and on graph augmentation. The typology is mixed to prove a good general result. The chosen datasets are:

- ENZYMES: a bioinformatics dataset composed of 600 graphs. They are natively divided into six classes, have node labels and attributes, with an average number of nodes of 32.63 connected by an average of 62.14 edges.
- IMDB-Multi: this social dataset counts 1500 graphs divided into three classes. They do not have node or edge labels, but they have complex connectivity, having an average of 65.94 edges and an average of 13 nodes per graph.
- MSRC_21: one of the oldest and classic datasets for semantic labeling. Twenty-one different categories of surfaces are considered. It is composed of 563 graphs divided into 20 classes. The mean edge count is 198.32 for an average of 77.52 nodes.
- MUTAG: it is the most straightforward dataset used. It defines small molecules in 188 graphs divided into two categories. The node average is 17.93, and the edge average is 19.79.
- NCI1: another small molecules dataset comprising 4110 graphs in 2 classes. The graphs have an average of 29.87 nodes and 32.30 edges.

We conducted a comprehensive set of experiments to evaluate the performance of our proposed model. We performed a 10-fold cross-validation on every dataset. For each fold, we divided the training set into training and validation sets with a 9:1 ratio. We used the validation set to select the best model within each fold. It is important to note that the folds and the splits of train, validation, and test were consistent among all the methods. The hyperparameters explored were the number of components in the mixture: 2, 3, 4, 6, 8, 10, 20, and for MSRC_21 also 30 and 40 considering the number of classes of the dataset; the dropout in the GCN of the mixture: 0, 0.1; and finally the number of hidden features of the GCN: 16, 32, 64, 128. We train the models for 2000 epochs in all the experiments, using the Adam optimizer, and with a learning rate of 0.01.

In Table 1, we reported the results of the experiments with AUC-ROC and average precision. MVGAE performs very well in all the datasets, achieving over 93% in the AUC-ROC curve and over 92% in the average precision. When compared to the classic variational autoencoder, MVGAE is largely better on all datasets; the biggest increase is on the IMDB-Multi dataset, almost doubling the precision on both metrics and the slimmest in MSRC_21 with a +13.62% in AUC-ROC and +10.87% in average precision.

GraphMAE, even if not built for link prediction, performs well compared to the baseline. However, our method performs better, even if for MSRC_21 is just ∼4–5% away from us in both metrics.

Table 1. Results with the standard errors on the 5 datasets considered for the link prediction task. We report the AUC-ROC curve and the average precision for the baseline Variational Autoencoder, as well as the comparison with GraphMAE, and MVGAE.

	ENZYMES	IMDB-Multi	MSRC_21	MUTAG	NCI1
AUC-ROC					
VAE	69.72 ± 0.523	50.31 ± 0.092	85.44 ± 0.338	73.01 ± 0.415	65.02 ± 0.352
GraphMAE	72.44 ± 0.910	84.56 ± 1.116	95.54 ± 0.072	56.50 ± 0.363	52.65 ± 0.063
MVGAE	**93.84 ± 1.961**	**98.82 ± 0.239**	**99.06 ± 0.068**	**96.42 ± 1.623**	**96.77 ± 1.283**
Average Precision					
VAE	71.83 ± 0.517	55.19 ± 0.109	87.72 ± 0.226	71.76 ± 0.465	68.27 ± 0.417
GraphMAE	71.38 ± 0.593	81.28 ± 0.851	93.08 ± 0.086	62.39 ± 0.220	59.38 ± 0.105
MVGAE	**92.54 ± 2.332**	**98.63 ± 0.344**	**98.59 ± 0.101**	**95.33 ± 2.025**	**95.77 ± 1.571**

5 Conclusion

In our paper, we introduced a new type of variational autoencoder called Mixture of Variational Graph Autoencoder (MVGAE), which combines and blends behaviors of autoencoders from different tasks. This method extends the original Variational Graph Autoencoder by Kipf and Welling [10]. Alongside the encoder part, we connect a mixture component to classify graphs into a predefined number of groups, which may not strictly align with the actual classification of the chosen dataset in order to underline other substructural properties.

Our results showed that this method holds promise in capturing different graph information, assigning it to the most similar groups, and reflecting this information in the structural connections. Still, further exploration is needed to investigate changes in hyperparameters and improve the method further.

Acknowledgment. Alessandro Bicciato is supported by the PRIN 2022 project n. 2022AL45R2 (EYE-FI.AI, CUP H53D2300350-0001).

Andrea Torsello's work was partially supported by the project "Perturbation problems and asymptotics for elliptic differential equations: variational and potential theoretic method" funded by the program "NextGenerationEU" and by MUR-PRIN, grant 2022SENJZ3.

References

1. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biol. Cybern. **59**(4–5), 291–294 (1988)
2. Cui, G., Zhou, J., Yang, C., Liu, Z.: Adaptive graph encoder for attributed graph embedding. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 976–985 (2020)
3. Hou, Z., et al.: Graphmae: self-supervised masked graph autoencoders. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 594–604 (2022)

4. Hu, W., et al.: Strategies for pre-training graph neural networks. arXiv preprint arXiv:1905.12265 (2019)
5. Hu, Z., Dong, Y., Wang, K., Chang, K.W., Sun, Y.: GPT-GNN: generative pre-training of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1857–1867 (2020)
6. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with Gumbel-Softmax. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017, Conference Track Proceedings. OpenReview.net (2017). https://openreview.net/forum?id=rkE3y85ee
7. Jin, W., et al.: Self-supervised learning on graphs: deep insights and new direction. arXiv preprint arXiv:2006.10141 (2020)
8. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. IEEE Trans. Neural Netw. **5**(1), 61–80 (2016)
10. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
11. Lee, J., Toutanova, K.: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, vol. 3, no. 8 (2018)
12. Ng, A., et al.: Sparse autoencoder. CS294A Lect. Notes **72**(2011), 1–19 (2011)
13. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. arXiv preprint arXiv:1802.04407 (2018)
14. Park, J., Lee, M., Chang, H.J., Lee, K., Choi, J.Y.: Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6519–6528 (2019)
15. Ranjan, R., Patel, V.M., Chellappa, R.: Hyperface: a deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. IEEE Trans. Pattern Anal. Mach. Intell. **41**(1), 121–135 (2017)
16. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
17. Salehi, A., Davulcu, H.: Graph attention auto-encoders. arXiv preprint arXiv:1905.10715 (2019)
18. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1096–1103 (2008)
19. Wang, C., Pan, S., Long, G., Zhu, X., Jiang, J.: MGAE: marginalized graph autoencoder for graph clustering. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 889–898 (2017)

Multimodality Calibration in 3D Multi Input-Multi Output Network for Dementia Diagnosis with Incomplete Acquisitions

Adriano De Simone[1,2(✉)], Michela Gravina[1], and Carlo Sansone[1]

[1] Department of Electrical Engineering and Information Technology (DIETI), University of Naples Federico II, Naples, Italy
{adriano.desimone3,michela.gravina,carlo.sansone}@unina.it
[2] Department of Public Health, University of Naples Federico II, Naples, Italy

Abstract. Alzheimer's disease (AD) is one of the leading causes of cognitive decline and death worldwide, necessitating ongoing research in diagnosis, prognosis, and treatment. Advances in Artificial Intelligence, particularly Deep Learning (DL), have enabled the development of sophisticated diagnostic models using medical imaging data. In clinical trials, Magnetic Resonance Imaging (MRI) and Positron Emission Tomography (PET) are predominantly used, offering valuable metabolic and structural insights. Integrating these modalities through Multimodal Deep Learning (MDL) approaches enhances diagnostic accuracy and robustness. However, implementing MDL in medical imaging presents challenges, such as the difficulty in obtaining complete multimodal data for each patient and effectively integrating the characteristics of multiple modalities. To address these issues, we propose a 3D Multi Input-Multi Output (3D-MIMO) neural network that employs an efficient training strategy to manage data absence while using a Transfer Module (TM) to synergistically fuse PET and MRI data. We validate our approach using the Open Access Series of Imaging Studies (OASIS) and Alzheimer's Disease Neuroimaging Initiative (ADNI) datasets, demonstrating the efficacy of our 3D-MIMO neural network in handling multimodal data.

Keywords: Multimodal Deep Learning · Transfer Module · Incomplete acquisitions

1 Introduction

Alzheimer's Disease (AD) is a neurodegenerative disorder characterized by the gradual decline of cognitive faculties. It is currently recognized as the most commonly diagnosed form of dementia and one of the leading causes of death worldwide. Although a cure to halt its pathological progression remains elusive, research makes significant strides in diagnosis, prognosis, and treatment to support patients and their families.

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 92–101, 2025.
https://doi.org/10.1007/978-3-031-80507-3_10

With technological advancements, particularly in Artificial Intelligence (AI), Deep Learning (DL) architectures have emerged as the leading approach for developing diagnostic models based on medical image data [8]. Training a DL model for AD diagnosis typically involves heterogeneous data sources with diverse informational content [8]. The primary investigative tools for studying the onset and monitoring the progression of AD using an image-based approach include Positron Emission Tomography (PET) and Magnetic Resonance Imaging (MRI). PET provides metabolic information, while MRI offers structural insights. Integrating these modalities enhances the richness of information, leading to a clearer clinical picture and increased diagnostic accuracy and robustness. Consequently, Multimodal Deep Learning (MDL) approaches are becoming increasingly prevalent for AD diagnostic tasks, enabling the integration of multiple and heterogeneous data sources.

Although widely researched and yielding promising results, the implementation of an MDL solution in medical image computing faces several challenges. One significant issue is the difficulty in obtaining images from all relevant modalities for the same patient within a multimodal dataset (paired acquisition). Indeed, in practical scenarios, patients often have incomplete acquisitions, where some modalities are missing. Furthermore, the successful integration of multiple modalities hinges on accurately accounting for the complex relationships between data from diverse sources. Different data-acquisition methods may emphasize various features with differing levels of importance, necessitating the reduction of less discriminative information and the enhancement of pertinent patterns. To address these challenges, we propose a 3D Multi Input-Multi Output (3D-MIMO) neural network that utilizes a Transfer Module (TM) [5] to facilitate synergistic fusion of PET and MRI modalities, thus improving learning performance. In addition, we implement an efficient training strategy to manage data absence, ensuring robust performance even with incomplete or highly unbalanced datasets. To demonstrate the efficacy of our methodology, we utilize two publicly available datasets, the Open Access Series of Imaging Studies (OASIS) [11] and the Alzheimer's Disease Neuroimaging Initiative (ADNI) [6], both of which are widely used in related studies and include MRI and PET acquisitions.

The rest of the paper is organized as follows: Sect. 2 briefly describes the literature; Sect. 3 illustrates the proposed methodology; Sect. 4 details the implemented experiments; Sect. 5 discusses the obtained results; Sect. 6 provides some conclusions.

2 Related Works

Multimodal data fusion techniques are typically classified into early, intermediate, and late fusion [17]. Early fusion (EF) involves integrating multiple data sources into a single structure before entering them into a learning model. Late fusion (LF) combines decisions coming from multiple classifiers, each trained on separate modalities, according to a specific rule. Intermediate fusion (IF), or

joint learning, utilizes deep neural networks to transform raw inputs into higher-level representations. This process often involves merging units from multiple modality-specific paths into a single layer [17].

In our previous works [3,4], we investigated the role of MDL for dementia assessment, comparing different fusion strategies in scenarios with incomplete data acquisitions. Additionally, recent literature has increasingly utilized IF to analyze complementary imaging modalities, employing DL approaches such as convolutional neural networks (CNNs) in both 2D [14] and 3D formats [2,9]. The methods proposed in the literature [1,2,9,18] demonstrate an accuracy greater than 90% in distinguishing normal brain conditions from damaged ones. However, to address incomplete acquisitions, authors have suggested methods such as replacing the missing modality with black images [14], or removing the unpaired volumes [2,9,18].

The success of integrating multiple modalities depends on effectively modeling the complex interactions between heterogeneous data sources [12]. Different acquisition tools may reveal features with varying degrees of relevance for a given task, which requires the reduction of redundant or less useful information while focusing on critical patterns [12]. To address this challenge, various methods have been proposed, including adaptive and learnable modules [7,19], as well as approaches that leverage feature correlations [13].

3 Methodology

Among all MDL approaches, IF strategy stands out for its exceptional flexibility, making it highly adaptable to various scenarios and architectures. However, effectively combining features of different natures remains challenging due to the intrinsic characteristics of the sources, particularly in scenarios with incomplete acquisitions [10,12]. To address these issues, this paper proposes the integration of a Multi Input-Multi Output 3D neural network (3D-MIMO) with a learnable Transfer Module (TM). This approach aims to enhance the integration of features extracted from different modalities while also increasing the robustness of the solution to missing data.

The proposed architecture, depicted in Fig. 1, employs 3D convolutional layers to fully exploit the volumetric characteristics of both imaging modalities, specifically brain MRI and PET. It features two separate inputs, each corresponding to one of the acquisitions, which follow parallel, symmetric modality-specific pathways. Each pathway includes six convolutional blocks, each consisting of a convolutional layer with a $3 \times 3 \times 3$ kernel, stride set to 2, and padding set to 1, followed by batch normalization and a ReLU activation function. The first convolutional block generates 16 output channels, while each successive block doubles the number of channels. These modality-specific pathways collectively form the Convolutional Core (Conv-C) of the proposed architecture, which encompasses the layers responsible for feature computation prior to the definition of the shared representation. The classification core (CC) of the proposed 3D MIMO network consists of three feature-specific cores: one acting on the

characteristics extracted from the MRI (MRI-CC), another on the PET (PET-CC), and a third on the shared representation (PAIRED-CC), which is obtained by concatenating the results of the two pathways in the Conv-C. Specifically, the MRI-CC and PET-CC consist of a global average pooling (G) layer and a fully connected (f_c) layers, while the PAIRED-CC includes a global average pooling layer followed by two fully connected layers separated by a ReLU function.

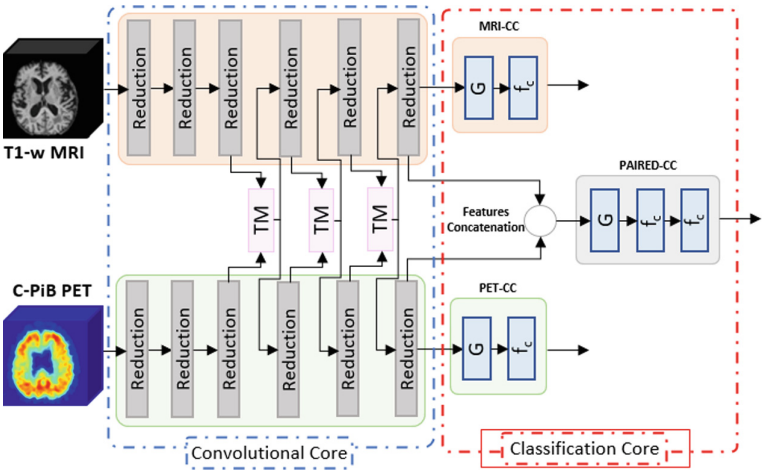


Fig. 1. 3D MIMO architecture, consisting of two inputs and three outputs. It is possible to distinguish three different sub-networks, the MRI-NET, the PET-NET, and the PAIRED-NET. The proposed transfer module in presented in the box TM

TM improves the integration of heterogeneous modalities by allowing them to influence each other during feature extraction. This process, called mode calibration, takes advantage of the complementarity of the inputs and uses control to minimize the contribution of the least relevant features in each specific feature map. Using descriptors from all inputs, this mechanism facilitates mutual influence between modalities throughout the training process. The proposed module is inserted into the Conv-C of the 3D MIMO network, between convolutional blocks belonging to different modality-specific pathways. Let $F_j^i \in \mathbb{R}^{X_j^i \times Y_j^i \times Z_j^i \times C_j^i}$ represent the feature map extracted by the i-th convolutional block within the j-specific pathways, with $j \in \{T1; PET\}$. Here, $X_j^i \times Y_j^i \times Z_j^i$ indicates the spatial dimensions and ${C_j}^i$ denotes the number of channels. The TM integrated into layer i is a multi input-multi output module that takes two feature maps F_j^i and provides two outputs $\tilde{F}_j^i$, corresponding to their calibrated versions, achieved through a gating procedure. The TM operates through two distinct stages, namely *shared vector computation* and *multimodal calibration*, as shown in Fig. 2.

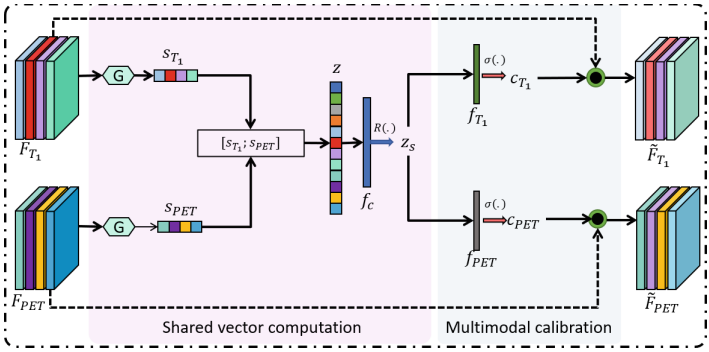


Fig. 2. Architecture of the proposed TM for two different image modalities

The *shared vector computation* determines the shared representation z_s^i of the layer $i-th$ that integrates vectors derived from the feature maps. Each channel descriptor vector $s_j^i \in \mathbb{R}^{1\times C_j^i}$ for the feature map is calculated using a global average pooling, ensuring an equal contribution of all elements within each channel to its characterization. Then, the concatenation of all the s_j^i creates $z^i \in \mathbb{R}^{1\times C^i}$ ($z^i = [s_{T1}^i \frown s_{PET}^i]$), where $C^i = C_{T1}^i + C_{PET}^i$. The shared representation z_s^i is derived from z_s^i by passing it through a fully connected layer f_c, followed by the ReLU activation function $R(\cdot)$ ($z_s^i = R(f_c(z^i))$), where $z_s^i \in \mathbb{R}^{1\times C^t}$. Here, C^t is defined as $C^t = C^i/r$ with r representing the reduction ratio, that is set to 4 as proposed in [5,7]. The inclusion of the ReLU enables the TM able to effectively capture the complex and nonlinear relationships among the elements of the input vector z^i, which comprises descriptors from various image modalities.

The *multimodal calibration* step utilizes the shared representation z_s^i to adjust the feature maps F_{T1}^i and F_{PET}^i, thus integrating information from different data sources. z_s^i serves as input to two separate fully connected layers f_{T1} and f_{PET}, each specific to a modality. The sigmoid activation function $\sigma(\cdot)$ is applied to confine the outputs within the range [0,1], producing two calibration vectors $c_{T1} \in \mathbb{R}^{1\times C_{T1}^i}$ ($c_{T1} = \sigma(f_{T1}(z_s^i))$), and $c_{PET} \in \mathbb{R}^{1\times C_{PET}^i}$ ($c_{PET} = \sigma(f_{PET}(z_s^i))$). These vectors aim to attenuate the influence of selected channels. Importantly, C_{T1}^i and C_{PET}^i correspond to the number of channels in F_{T1}^i and F_{PET}^i, respectively, such that each value in the calibration vector denotes the importance of the corresponding channel in the feature map. Consequently, the calibrated feature maps $\tilde{F}_{T1}^i$ and $\tilde{F}_{PET}^i$ are obtained through a channel-wise product ($\odot$) between the input F_{T1}^i and F_{PET}^i and their respective calibration vectors c_{T1} and c_{PET}, weighting each channel according to its significance level.

Our implementation follows the methodology proposed in [5,7], placing the TM in the latter stages of the network (Fig. 1). This placement choice aims to avoid excessive calibration of feature maps that are strongly influenced by specific image modalities.

The integration of multiple inputs and outputs in our proposed architecture facilitates the development of an effective training strategy designed to accommodate incomplete data acquisitions. As depicted in Fig. 1, the architecture consists of three distinct sub-networks. MRI-NET and PET-NET incorporate modality-specific pathways within Conv-C and feature-specific cores within CC for MRI and PET, respectively. In contrast, PAIRED-NET integrates the entire Conv-C and PAIRED-CC for combined modalities. This modular design proves advantageous for handling incomplete acquisitions: MRI-NET and PET-NET are utilized when only a single modality is available, whereas PAIRED-NET leverages paired acquisitions. The implemented training strategy involves adapting the network weight updates based on input characteristics. Specifically, when both modalities are present, updates are applied to layers within PAIRED-NET. Conversely, in cases where a modality is missing, only one sub-network (e.g., MRI-NET if PET data is absent) participates in the training process. It is worth noting that the TM modules inserted between convolutional blocks belonging to different modality-specific pathways can be updated only in the case of paired acquisitions.

4 Experimental Set-Up

In this study, we evaluate the generalization ability of our methodology using two publicly available datasets that include both MRI and PET acquisitions. The first dataset is the OASIS-3 collection [11], which comprises images and clinical data for 1098 patients. We utilize T1-weighted (T1) MRI and Pittsburgh Compound B (C-PiB) PET scans. The Clinical Dementia Rating (CDR) score is used to categorize the data into two distinct classes, as recommended by various studies in the literature [15,16]: a CDR score of 0 ($CDR = 0$) indicates normal cognitive function, defining the cognitive normal class (C), while a CDR score greater than 0.5 ($CDR > 0.5$) indicates dementia status (DS).

The second dataset is ADNI [6], a public database project of the University of Southern California. In this study, we included ADNI-1, ADNI-2, ADNI-GO and ADNI-3 collections. Among all the possible modalities, we considered the T1-weighted (T1) MRI and the fluorodeoxyglucose (FDG) PET scans. Each acquisition is associated with a diagnostic group. We selected the volumes declared as cognitively normal (C class), and included in the DS class the acquisitions designed as mild cognitive impairment or Alzheimer's disease.

Scan sessions and clinical data acquisition do not always align. A common method is to consider MRI and PET images as part of the same session if the difference in days between them is less than one year. This approach yields three distinct datasets: MRI, PET, and PAIRED. The MRI and PET datasets comprise all MRI and PET volumes, respectively, along with their corresponding class labels. The PAIRED dataset includes only pairs of MRI-PET that belong to the same session. Table 1 provides details of the datasets generated from both the OASIS-3 [11] and ANDI [6] collections, including the number of volumes for each class.

Table 1. Information about generated datasets in terms of number of volumes for each class, cognitive normal (C) and Dementia status (DS). Notably, the MRI and PET datasets also include the volumes in the PAIRED dataset.

OASIS-3				ADNI			
Dataset	Classes		TOTAL	Dataset	Classes		TOTAL
	C	DS			C	DS	
MRI	926	337	1263	MRI	973	1569	2542
PET	627	91	718	PET	236	514	750
Paired	556	72	628	Paired	225	737	962

The pre-processing steps involve motion correction, skull-stripping to remove non-brain tissue, and intensity normalization. To further reduce the inclusion of non-brain tissue, we extract the smallest 3D cubical box that encloses the brain from each MRI and PET volume. These extracted volumes are then resized to $128 \times 128 \times 128$ and normalized to the [0,1] range. This ensures that the network processes images on the same scale across different acquisitions.

Although we focus on IF strategy including TM (IF-TM configuration), we provide a comparison with different competitors, namely the Unimodal approach (U), and other multimodal fusion approaches, such as the EF, the LF, and the IF without the module.

As depicted in Fig. 1, the 3D MIMO architecture comprises three distinct sub-networks: MRI-NET, PET-NET, and PAIRED-NET. This configuration allows for the performance assessment of the implemented network across three separate datasets, corresponding to each classification core. Additionally, the performance can be evaluated on a composite dataset, created by combining the MRI, PET, and PAIRED datasets, referred to as the COMPLETE dataset. When the input is an MRI (or PET) volume, the output of MRI-NET (or PET-NET) is utilized. Conversely, for an MRI-PET pair, the outputs of the PAIRED-CC are considered. Consequently, this setup enables a comprehensive performance evaluation across all available volumes. The EF and LF approaches are applicable only to the PAIRED dataset, as EF requires a two-channel input and LF integrates results from two networks. However, performance evaluation using the COMPLETE dataset can be achieved by leveraging the network trained with the U approach to handle incomplete inputs. We evaluated performance using metrics such as Accuracy (ACC), Precision, Recall for each class, and the Area Under the ROC Curve (AUC). We assigned 70% of the patients to the training set, 15% to the validation set, and the remaining 15% to the test set, ensuring that there was no overlap of subject volumes between the training and evaluation phases. All experiments were conducted using PyTorch (version 1.10) for network training and MATLAB 2020b for preprocessing. The hardware setup included a workstation equipped with two NVIDIA RTX 3090 GPUs, an Intel(R) Core(TM) i7-10700KF CPU, and 64 GB of DDR4 RAM.

5 Results and Discussions

Table 2 presents the results of the experiments carried out on both the OASIS-3 [11] and ADNI [6] collections, detailing the fusion modality (Mod.), the data involved in the evaluation (Data), and the performance metrics. For the MRI and PET datasets, the proposed 3D MIMO architecture with TM outperforms the unimodal approach in terms of ACC for both collections. Notably, there is a significant performance improvement in the PET results for both OASIS-3 [11] and ADNI [6] datasets across all metrics. When considering the PAIRED dataset for OASIS-3 [11], the IF-TM configuration achieves the highest values across all metrics, surpassing both EF and LF strategies. Additionally, we compared the proposed 3D MIMO methodology with two state-of-the-art solutions [2,4] that utilize a multi-input network for the IF strategy. The results demonstrate the superior performance of our 3D MIMO architecture with TM. For the ADNI [6] dataset, our proposed architecture surpasses the solution in [4] in terms of ACC, precision for DS, and recall for C classes. Moreover, it outperforms the approach in [18], which focuses on a multi-input 3D CNN for distinguishing between normal (C) acquisitions and those indicative of dementia status (DS). However, it is worth noting that although the IF-TM solution performs well, an EF-based approach achieves slightly better values in some metrics. Examining the COMPLETE dataset, TM performs sensibly better with OASIS-3 [11], whereas with ADNI [6], it achieves the best results in terms of ACC but has the lowest AUC value. We hypothesize that the differing performance of the proposed architecture on the OASIS-3 [11] and ADNI [6] datasets may be attributed to the varying levels of imbalance present in these collections. As detailed in Table 1, OASIS-3 [11] exhibits a more pronounced class imbalance, whereas ADNI [6] shows a greater imbalance between the number of PET and MRI acquisitions. Regardless of the dataset, Table 2 underscores the superiority of the MDL approach over

Table 2. Performance of the experiments on both OASIS-3 and ADNI collections.

	OASIS-3							ADNI						
Data	Mod.	ACC	Precision		Recall		AUC	Mod.	ACC	Precision		Recall		AUC
			C	DS	C	DS				C	DS	C	DS	
MRI	U	76.89	87.57	57.89	**78.72**	72.37	80.98	U	67.05	52.90	**76.16**	**64.06**	66.81	**74.05**
	IF-TM	**79.37**	**93.24**	**59.62**	76.67	**86.11**	**84.04**	IF-TM	**69.63**	**56.37**	71.21	18.90	**93.20**	70.23
PET	U	77.93	94.17	38.10	78.86	72.73	77.05	U	94.83	88.09	98.65	97.37	93.59	95.44
	IF-TM	**83.22**	**99.05**	**45.45**	**81.25**	**95.24**	**95.50**	IF-TM	**96.49**	**89.02**	**99.50**	**98.65**	**95.73**	**97.43**
PAIRED	EF	85.16	89.74	36.36	93.75	25.00	78.96	EF	**97.04**	89.74	**99.23**	**97.22**	96.99	98.66
	LF	82.48	96.65	37.33	83.09	77.78	87.37	LF	92.44	90.95	95.59	85.22	97.74	98.98
	IF-TM	**96.90**	**97.37**	**93.33**	99.11	**82.35**	**95.62**	IF-TM	95.86	89.19	97.72	91.67	96.99	98.97
	IF [4]	91.34	96.36	58.82	93.81	71.43	88.24	IF [4]	94.08	**93.34**	94.24	77.78	**98.49**	**99.10**
	IF [2]	95.00	-	-	**100.00**	83.33	94.00	IF [18]	94.11	-	-	94.27	93.33	-
COMPLETE	EF	80.78	86.07	67.50	86.93	65.85	83.31	EF	73.60	60.20	**84.40**	**72.69**	74.82	80.53
	LF	79.72	89.27	59.58	82.32	72.47	81.25	LF	73.06	60.40	82.40	71.69	73.82	**80.93**
	IF-TM	**84.93**	**93.79**	**68.42**	**84.69**	**85.53**	**87.57**	IF-TM	**75.26**	**72.17**	76.03	43.01	**91.60**	78.95

the unimodal scenario due to the integration of complementary information. The IF method leverages the ability of neural networks to create hierarchical, learnable representations of input data, facilitating integration at multiple levels of abstraction [17]. This approach effectively mitigates the limitations of EF, where predefined shared representations can overlook feature correlations and redundancies. Additionally, compared to LF, which relies on a final decision step, IF enables simultaneous contributions from heterogeneous data sources during the decision-making process. It improves the integration of the feature transfer module by reducing the influence of the less discriminating ones. The training strategy effectively exploits all available information, allowing flexible management of incomplete data. The module, designed for learning from different sources, is easily integrated and generalizable, making it useful for analyzing other diseases in various medical data.

6 Conclusion

In this paper, our objective is to improve the dependency between modality-specific pathways by incorporating a transfer module (TM) within a 3D MIMO network, capable of handling incomplete acquisitions. This module is designed to emphasize the most discriminative features while attenuating less informative ones, thereby improving the definition of a shared representation. Moreover, to make the proposed architecture applicable in scenarios with incomplete acquisitions, we implemented an effective training strategy, that update the weights of the network according to the characteristics of the inputs. To the best of our knowledge, this paper presents the first solution incorporating the TM within a MIMO architecture designed for a multimodal scenario with incomplete acquisitions. Future work will explore the role of the transfer module in various tasks, such as multimodal segmentation in medical imaging. Furthermore, further analysis will focus on the explainability of the implemented solutions, including the significance of features identified by the TM in the calibration vectors.

Acknowledgements. This work has been supported by PNRR MUR project PE0000013-FAIR.

References

1. Abdelaziz, M., Wang, T., Elazab, A.: Alzheimer's disease diagnosis framework from incomplete multimodal data using convolutional neural networks. J. Biomed. Inform. **121**, 103863 (2021)
2. Castellano, G., Esposito, A., Lella, E., Montanaro, G., Vessio, G.: Automated detection of Alzheimer's disease: a multi-modal approach with 3D MRI and amyloid PET. Sci. Rep. **14**(1), 5210 (2024)
3. De Simone, A., Sansone, C.: A multimodal deep learning based approach for Alzheimer's disease diagnosis. In: International Conference on Image Analysis and Processing, pp. 131–139. Springer (2023)

4. Gravina, M., García-Pedrero, A., Gonzalo-Martín, C., Sansone, C., Soda, P.: Multi input-multi output 3D CNN for dementia severity assessment with incomplete multimodal data. Artif. Intell. Med. **149**, 102774 (2024)
5. Gravina, M., Santucci, D., Cordelli, E., Soda, P., Sansone, C.: Cross-modality calibration in multi-input network for axillary lymph node metastasis evaluation. IEEE Trans. Artif. Intell. (2024)
6. Jack, C.R., Jr., et al.: The Alzheimer's disease neuroimaging initiative (ADNI): MRI methods. J. Magn. Reson. Imaging: Official J. Int. Soc. Magn. Reson. Medi. **27**(4), 685–691 (2008)
7. Joze, H.R.V., Shaban, A., Iuzzolino, M.L., Koishida, K.: MMTM: multimodal transfer module for CNN fusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13289–13299 (2020)
8. Khojaste-Sarakhsi, M., Haghighi, S.S., Ghomi, S.F., Marchiori, E.: Deep learning for Alzheimer's disease diagnosis: a survey. Artif. Intell. Med. **130**, 102332 (2022)
9. Kim, S.K., Duong, Q.A., Gahm, J.K.: Multimodal 3D deep learning for early diagnosis of Alzheimer's disease. IEEE Access (2024)
10. Lahat, D., Adali, T., Jutten, C.: Multimodal data fusion: an overview of methods, challenges, and prospects. Proc. IEEE **103**(9), 1449–1477 (2015)
11. LaMontagne, P.J., et al.: OASIS-3: longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and Alzheimer disease. MedRxiv (2019)
12. Liu, K., Li, Y., Xu, N., Natarajan, P.: Learn to combine modalities in multimodal deep learning. arXiv preprint arXiv:1805.11730 (2018)
13. Ma, W., et al.: A novel adaptive hybrid fusion network for multiresolution remote sensing images classification. IEEE Trans. Geosci. Remote Sens. **60**, 1–17 (2021)
14. Massalimova, A., Varol, H.A.: Input agnostic deep learning for Alzheimer's disease classification using multimodal MRI images. In: 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), pp. 2875–2878. IEEE (2021)
15. Puente-Castro, A., Fernandez-Blanco, E., Pazos, A., Munteanu, C.R.: Automatic assessment of Alzheimer's disease diagnosis based on deep learning techniques. Comput. Biol. Med. **120**, 103764 (2020)
16. Qiu, S., et al.: Development and validation of an interpretable deep learning framework for Alzheimer's disease classification. Brain **143**(6), 1920–1933 (2020)
17. Ramachandram, D., Taylor, G.W.: Deep multimodal learning: a survey on recent advances and trends. IEEE Signal Process. Mag. **34**(6), 96–108 (2017). https://doi.org/10.1109/MSP.2017.2738401
18. Song, J., Zheng, J., Li, P., Lu, X., Zhu, G., Shen, P.: An effective multimodal image fusion method using MRI and PET for Alzheimer's disease diagnosis. Front. Digit. Health **3**, 637386 (2021)
19. Zhou, T., Fu, H., Chen, G., Shen, J., Shao, L.: Hi-net: hybrid-fusion network for multi-modal MR image synthesis. IEEE Trans. Med. Imaging **39**(9), 2772–2781 (2020)

The Effect of Class Distribution on Multi-Modal Medical Images Classification in Meta-Learning

Zainab Almugbel[1,2(✉)]

[1] Computing Department, Applied College, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
[2] Computer Science and Engineering, School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
zhalmugbel@iau.edu.sa,zainab.almugbel@mdu.se

Abstract. Medical images are available in small datasets with different modalities and for various organs. Although Transfer learning is a promising approach for training models on small datasets, further studies are required on using them with several image modalities and body parts. This paper explores two dominant meta-learning algorithms: a metric-based algorithm, namely Prototypical Network, and an optimization-based algorithm, namely MAML. The algorithms trained on a small multi-modal medical dataset (i.e., Slake), and a general dataset of the same size (i.e., Tiny-imagenet) with different class distribution methods: Random, Logical-based, and Statistical-based class distribution. The aim is to apply diversity among the classes in the meta-training set and similarity between the classes in the meta-training, and meta-testing or meta-validation sets. The results validate the importance of class distribution on the accuracy of the algorithms. MAML is hard to be trained on Tiny-imagenet but it shows good accuracy on Slake in specific cases. In statistical distribution, the Prototypical Network shows high accuracy on the datasets, especially when the similarity between the meta-training and meta-validation sets is considered for Tiny-imagenet.

Keywords: Meta-learning · Class distribution · Medical images

1 Introduction

The availability of small medical image datasets makes it insufficient to train deep learning models. Image feature extraction is essential for image classification because the features hold the important aspects of the images. These features can enhance image classification [15]. Since deep learning models require large datasets for training, transfer learning is a promising approach for training models on small datasets to extract image features [16]. However, it is confirmed that the transfer learning models give good results in their target tasks but further studies are required on using them with several image modalities and organs

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 102–111, 2025.
https://doi.org/10.1007/978-3-031-80507-3_11

[16]. The limitation of data issue could be solved by collecting medical data from the internet to further train the model before using it on the medical datasets [9].

This has two effects. First, data credibility would be varied since data is taken from different resources. Second, unlike general images, medical images have several modalities that might not be available on the internet, namely Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Positron Emission Tomography (PET), and Medical Hyperspectral Image (HSI). Choosing the appropriate pre-trained model for a specific task is influenced by the image type and the organ category [16]. Utilizing an existing pre-trained model and applying transfer learning might not be useful.

This motivates checking the meta-learning approaches that are designed specifically for small datasets. Many medical image classification approaches focus on specific diseases or image modalities. These approaches separately train and evaluate the models on each dataset. This work evaluates the training from scratch of two dominant meta-learning algorithms on a tiny dataset that includes multi-modal medical images for different body parts, and diseases on different class distribution methods. The purpose is to show the class distribution's effect on the meta-learning algorithms' performance.

2 Background

2.1 Meta-Learning

Meta-learning is a machine-learning approach to train models in small datasets. It is the ability to adapt the prior experience to a new task in a "systematic and data-driven" way in which a set of parameters are updated [36]. Each task is a subset of the dataset that includes data samples arranged in N-ways K-shots, where N is the number of classes and K is the number of samples of each class.

The tasks are generated randomly to form support and query sets of meta-training, meta-validation, and meta-testing sets. First, the learner uses the support set for training to get initial values for the parameters. Second, the learner updates the values of the parameters to learn a new objective taken from the query set. Thus, the learner can adapt to a new sample of the query set (a disjoint set of the support set) during the test time via transferring the "information from the most similar task to the new task" [36].

MAML. MAML is a model and task-agnostic algorithm for meta-learning that aims to obtain high performance on a small number of gradient updates [10]. It has two levels of parameter updates: one on the current task level and a second on the all tasks level [10]. MAML takes a set of tasks that consists of a support set and a query set. It is trained on the support set, then it quickly adapts its learning to the query set to minimize the loss function [17].

Prototypical Network. Unlike MAML, Prototypical Network [29] is based on the similarity between the embeddings of a sample taken from the query set and a prototype. The prototype is calculated by taking the mean of the embeddings of each class's samples in the support set. After obtaining the embeddings of the query's sample and the prototype, any similarity measurement method, such as Euclidean distance, can be used to measure the distance between the two embeddings. The prototype with the lowest distance is the class for the sample.

2.2 Few-Shot Medical Image Classification

The research in medical image classification targets two issues: dataset scarcity and imbalance dataset/long tail class distribution. The approaches can be categorized as follows: The first category includes the integration of different models, such as the integration of Prototypical Network, Pretrained ResNet34, and Siamese Network [1], Variational Auto-encoder and Prototypical Network [12], Prototypical Network with auxiliary Supervised Contrastive network [23], Graph Neural Network and MAML [22], and MAML with a knowledge graph-based disease diagnosis model [19]

The second category integrates different models with different augmentation techniques as an additional step, such as Siamese-Prototypical network with augmentation techniques [32]. The third category re-configures a dataset to investigate cross-domain shift (clinic/patient shift) [4].

The fourth category modifies the model to enhance its feature extraction ability as follows: MAML's cross-entropy function is replaced with dice loss [34]. Siamese Neural Networks is integrated with different data balancing and weighted loss techniques [11]. The loss function is modified with augmentation techniques [20]. Different convex optimization models are used as classifiers in the meta-learning setting as alternatives to classical meta-learning [21]. Advanced regularization techniques are applied to the Reptile algorithm [28]. This last method outperforms transfer learning [28].

The last category uses public and domain-specific datasets with different integrated models as follows: Variational Autoencoder (VAE) is used to reconstruct the extracted feature of CNN with public and medical data sets [6]. Prototypical Network and model agnostic meta-learning (MAML) algorithms are used to classify ultrasound breast cancer images in a cross-domain approach using public dataset in meta-training and domain-specific dataset in meta-testing [14]. Two transfer learning techniques (pre-training in imagenet and a multi-source domain generalization method from person re-identification field) are used, in sequence, to enhance feature extraction before applying Prototypical Network for classification [33].

The research either attempts to increase the amount of data or develops a complex system. One work focuses on the dataset reconfiguration for cross-domain shift evaluation [4]. Another work [33] uses a multi-modal (X-RAY, CT, MRI) medical dataset with transfer learning. The dataset is built by taking medical images from 10 public medical image datasets (lung, chest, eye, brain tumor, and human tissue).

Since this research re-configures a dataset to be used in few-shot settings, it studies the impact of that dataset design on Few-shot image classification on tiny datasets. This has been previously studied with large datasets containing thousands of samples in [27]. They validate that the design of the dataset can be a better option to enhance the performance than coming up with a new advanced algorithm [27]. In their research, they compare the performance of two Metric-based algorithms, namely Prototypical Network [29] and Matching Network [30]. The size of the smallest used dataset is CUB [31] with a total of 5885 images for 100 classes. The study also only validates its output on non-medical datasets imagenet [8] and CUB (both animal-based datasets). The output confirms the effect of the dataset design on the selected algorithms on a small number of classes. When the number of classes is big, the effect is small. One additional work [35] assumes that some classes are not helpful or might hurt the performance. For that, it proposes learning the selection of classes based on optimizing a function called the similarity ratio (a metric that measures the class "similarities with novel classes and diversity in base classes".

This research follows the work [27] to validate how diversity and similarity can enhance the performance of meta-learning algorithms. While class diversity means the measurement of similarity among images in one class [27], the diversity, here, is defined as the dis-similarity among the classes in the meta-training set. The similarity, here, is measured between the meta-training set and the meta-testing as in [27] or meta-validation sets. Other differences are the size and type of datasets, and the selected meta-learning algorithms belonging to different categories.

3 Methodology

This section presents the datasets and the experiments' settings. First, medical and general datasets are described. Second, the chosen backbone, the hyperparameters, and the methods of class distribution are presented.

3.1 Datasets Configuration

Medical Dataset. SLAKE [18] is a publicly clinician-oriented Visual Question Answering dataset of different image modalities (MRI, X-rays, or CT). To re-configure SALKE in a few-shot setting, the images are extracted and classified based on the content types of their questions, where content types are organs or abnormalities avoiding image duplication and binary questions. This gives only 336 images distributed equally among 21 classes: [Abdomen, Lung, Chest, Cardiomegaly, Atelectasis, Pneumonia, Left Lung, Right Lung, Liver, Liver Cancer, Chest_lung, Spleen, Heart, Small Bowel, Brain_Tissue, Brain_Face, Brain, Neck, Head, Pelvic Cavity, Lung Cancer].

Non-medical Dataset. Tiny-imagenet has been randomly created from Mini-imageNet [26] to compare the performance of the chosen meta-learning algorithms on medical and general datasets of the same size. Tiny-imagenet includes 336 images of 21 classes: [house_finch, robin, triceratops, green_mamba, harvestman, toucan, jellyfish, dugong, Walker_hound, Saluki, Gordon_setter, goose, Ibizan_hound, white_wolf, meerkat, rhinoceros_beetle, nematode, king_crab, golden_retriever, malamute, dalmatian].

3.2 Experimental Setting

MAML and Prototypical Network use Resnet 12 as a backbone with the same Hyperparameters. The class distribution assigns different classes in the meta-training, meta-validation, and meta-testing sets.

Hyperparameters. Optuna [2] is applied on MAML to choose the hyperparameters, see Table 1. They are also applied to the Prototypical Network.

Table 1. Hyper-parameters Settings.

hyper parameters	value	hyper parameters	value
epoch	40	optimizer	adam
n-way [class number]	4	k-shot [support set]	1, 5, 8
k-shot [query set]	8	learning rate	0.452
episodes train no.	200	episodes validation no.	400

Class Distribution. The classes are split into 60% meta-training: 11 classes, and 20% to meta-validation and meta-testing sets: 5 classes each. The distribution of these classes affects the tasks distribution which is important for accuracy. The research compares three methods of class distribution: random, logical-based, and statistical-based distribution to validate the impact of meta-training set diversity and its similarity to the meta-validation or meta-testing sets.

Random Class Distribution. The classes are randomly split among three sets: meta-training, meta-validation, and meta-testing sets. In this case, the meta-testing set might include a new group, where a group is a set of classes that are related logically. This means no class of that group presented in meta-training or meta-validation sets.

Logical-Based Class Distribution. Two steps are followed to avoid presenting a new group in the meta-testing set. The classes are grouped based on the images' content. For instance, a group includes a set of organs and diseases belonging to

the respiratory system. Then, the relationships among the classes are identified. At least one class of each group is included in the meta-training set. While diversity is implemented by including classes of different groups in the meta-training set, The relationships are used to measure similarity among the classes. The remaining classes that have direct relationships to the meta-training classes are included in meta-validation or meta-testing set depending on the experiment settings. The logical class distribution is done manually since the datasets are small, but it could be automated by creating a knowledge graph of the classes. Then, the relationship degrees are used for applying diversity and similarity for the class distribution.

Statistical-Based Class Distribution. Since the Prototypical Network is a metric-based algorithm, this method has been selected to show its effect on the accuracy. The color-based multi-dimensional histogram [25] is randomly selected to represent image features. Next, the class feature is calculated by taking the mean of its images' features. Finally, Euclidean distance [7] and Cross-correlation [5] measure the distances among the classes. Unlike the logical split which applies relationships for class distribution, the statistical split applies distances. The meta-training set contains the classes showing the highest distances. The classes with the lowest distances to the meta-training set are included in the meta-validation or meta-testing for similarity implementation.

4 Results and Discussions

Two main experiments are conducted considering the meta-training set diversity: the first considers its similarity to the meta-testing set, and the second considers its similarity to the meta-validation set. The experiments' results, see Tables 2, 3 and 4, are the mean accuracy of 100 tasks of three runnings. The highest accuracy of each distribution method is in bold.

Friedman's test (FR) [24] is calculated on Tables 2 and 3 to check if a significant difference exists among the results, where the n-ways k-shots are the blocks and the class distribution methods are the treatments. The null hypothesis states that the medians of the four class distribution methods are equal, hence no significant difference exists. FR is calculated four times: one per model and dataset. The values are equal to 9 and 8.2. When the Chi-square distribution table with 2 degrees of freedom is checked, the null hypothesis is retained with a considerably low significant level of 1%.

The results show that learning the classes in Tiny-imagenet is more difficult than Slake. This might be because of the features of the images. Tiny-imagenet includes animals with backgrounds of different shapes, colors, and structures. The deep learning model starts with learning the basic shapes in the first layers; then, it evolves in the deeper layers [15]. Although the small dataset makes it harder for the model to learn these features, Prototypical Network has a better learning in the meta-training meta-validation similarity setting.

In general, the Prototypical Network outperforms MAML on both datasets. The Prototypical Network calculates the similarity between the class representation and the test sample for the prediction while MAML depends on the gradient update. When MAML is trained to learn diverse and distinct tasks (set of classes and their samples), MAML lacks transferring the learned knowledge or generalizing its learning to other tasks [3]. This is because some of the tasks might require a major modification to the gradients which causes forgetting the previously learned tasks (catastrophic forgetting) [13].

Table 2. Performance in Random Split.

Model	Dataset	4-way 1-shot	4-way 5-shot	4-way 8-shot
MAML	SLAKE	0.6687	0.7547	0.7875
MAML	Tiny-imagenet	0.2684	0.2834	0.2834
Proto-Network	SLAKE	**0.719**	**0.775**	**0.812**
Proto-Network	Tiny-imagenet	**0.348**	**0.449**	**0.484**

Table 3. Meta-training and Meta-testing Similarity Performance.

Model	Dataset	Logical Split			Euclidean Distance			Cross-correlation		
		4-way 1-shot	4-way 5-shot	4-way 8-shot	4-way 1-shot	4-way 5-shot	4-way 8-shot	4-way 1-shot	4-way 5-shot	4-way 8-shot
MAML	SLAKE	**0.7212**	**0.7769**	**0.7975**	0.4628	0.4462	0.4597	0.5153	0.5291	0.5216
	Tiny-imagenet	0.2709	0.2863	0.3119	0.4222	0.4378	0.4469	0.2878	0.2938	0.3075
Proto-Network	SLAKE	0.711	0.770	0.787	**0.881**	**0.934**	**0.952**	**0.891**	**0.935**	**0.942**
	Tiny-imagenet	0.308	0.389	0.407	0.372	**0.474**	**0.507**	**0.398**	**0.473**	**0.503**

In Table 3, the similarity is considered between the classes in the meta-training and meta-testing sets. MAML shows the highest performance when no new group is introduced in the meta-validation and meta-testing sets (logical distribution on Slake and Euclidean Distance on Tiny-imagenet). The statistical approach benefits the Prototypical Network, even when a new group presents in the meta-testing set.

In Table 4, the similarity is considered between the classes in the meta-training and meta-validation sets. In this setting, the classes of the meta-validation set and replaced with the classes in the meta-testing set. MAML shows more benefits from the statistical approach on Slake. In addition, the results show close performance on both datasets. While the Prototypical Network performance goes higher with the logical split, it shows close performance with

Table 4. Meta-training and Meta-validation Similarity Performance.

Model	Dataset	Logical Split			Euclidean Distance			Cross-correlation		
		4-way 1-shot	4-way 5-shot	4-way 8-shot	4-way 1-shot	4-way 5-shot	4-way 8-shot	4-way 1-shot	4-way 5-shot	4-way 8-shot
MAML	SLAKE	0.6678	0.7134	0.7488	0.7566	0.7809	0.7863	0.7366	0.7606	0.7753
	Tiny-imagenet	0.2575	0.2547	0.2587	0.2847	0.2863	0.2875	0.2812	0.2809	0.2594
Proto-Network	SLAKE	**0.874**	**0.906**	**0.902**	**0.826**	**0.893**	**0.899**	**0.867**	**0.9**	**0.906**
	Tiny-imagenet	**0.945**	**0.981**	**0.988**	**0.943**	**0.981**	**0.989**	**0.938**	**0.981**	**0.986**

the other splits on SLAKE. In Tiny-imagenet, Prototypical Network outperform MAML on all the settings. Applying the meta-training meta-validataion similarity wins over the meta-training meta-testing similarity in many cases and show a more stable performance. Thus, the effect of the class distribution depends on the algorithm and the dataset characteristics.

5 Conclusion

This research studies the effect of class distribution on training meta-learning algorithms on small domain-specific and general datasets. The design of the dataset affects the performance of MAML and Prototypical Network. In general, the diversity of the meta-training classes and the similarity between the meta-training meta-validation sets show close performance in both algorithms unlike the meta-training meta-testing similarity. Although the class distribution shows a big impact on the accuracy, it might not be useful in real-life applications. A new approach is required to produce an accuracy that mitigates the effect of the class distribution on the performance.

References

1. Abbas, Q.: An intelligent medical image classification system using few-shot learning. Concurrency Comput. Pract. Experience **35**(2), e7451 (2023)
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
3. Baik, S., Choi, J., Kim, H., Cho, D., Min, J., Lee, K.M.: Meta-learning with task-adaptive loss function for few-shot learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9465–9474 (2021)
4. Chamarthi, S., Fogelberg, K., Gawlikowski, J., Brinker, T.J.: Few-shot learning for skin lesion classification: a prototypical networks approach. Inf. Med. Unlocked **48**, 101520 (2024)
5. Crochiere, R.E., Rabiner, L.R.: Multirate Digital Signal Processing, vol. 18. Prentice-hall Englewood Cliffs, NJ (1983)

6. Dai, Z., et al.: PFEMed: few-shot medical image classification using prior guided feature enhancement. Pattern Recogn. **134**, 109108 (2023)
7. Danielsson, P.E.: Euclidean distance mapping. Comput. Graphics Image Process. **14**(3), 227–248 (1980)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
9. Eslami, S., de Melo, G., Meinel, C.: Does clip benefit visual question answering in the medical domain as much as it does in the general domain? arXiv preprint arXiv:2112.13906 (2021)
10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning, pp. 1126–1135. PMLR (2017)
11. Galán-Cuenca, A., Gallego, A.J., Saval-Calvo, M., Pertusa, A.: Few-shot learning for COVID-19 chest x-ray classification with imbalanced data: an inter vs. intra domain study. Pattern Anal. Appl. **27**(3), 69 (2024)
12. Guo, Q., et al.: Plug-and-play feature generation for few-shot medical image classification. In: 2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1096–1103. IEEE (2023)
13. Gupta, G., Yadav, K., Paull, L.: La-MAML: look-ahead meta learning for continual learning. arXiv:2007.13904 (2020)
14. Işık, G., Paçal, İ: Few-shot classification of ultrasound breast cancer images using meta-learning algorithms. Neural Comput. Appl. **36**, 12047–12059 (2024)
15. Jogin, M., Madhulika, M., Divya, G., Meghana, R., Apoorva, S.: Feature extraction using convolution neural networks (CNN) and deep learning. In: 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), pp. 2319–2323. IEEE (2018)
16. Kora, P., et al.: Transfer learning techniques for medical image analysis: a review. Biocybernetics Biomed. Eng. **42**(1), 79–107 (2022)
17. Kotia, J., Kotwal, A., Bharti, R., Mangrulkar, R.: Few shot learning for medical imaging. In: Das, S., Das, S., Dey, N., Hassanien, A.E. (eds.) Machine Learning Algorithms for Industrial Applications, pp. 107–132 (2021)
18. Liu, B., Zhan, L.M., Xu, L., Ma, L., Yang, Y., Wu, X.M.: Slake: a semantically-labeled knowledge-enhanced dataset for medical visual question answering. In: 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), pp. 1650–1654. IEEE (2021)
19. Liu, Q., et al.: A few-shot disease diagnosis decision making model based on meta-learning for general practice. Artif. Intell. Med. **147**, 102718 (2024)
20. Liu, X., et al.: A difficulty-aware and task-augmentation method based on meta-learning model for few-shot diabetic retinopathy classification. Quant. Imaging Med. Surg. **14**(1), 861 (2024)
21. Lu, L., Cui, X., Tan, Z., Wu, Y.: MedOptNet: meta-learning framework for few-shot medical image classification. IEEE/ACM Trans. Comput. Biol. Bioinf. **21**(4), 725–736 (2023)
22. Mohanta, A., Dey Roy, S., Nath, N., Bhowmik, M.K.: Domain adapted few-shot learning for breast histopathological image classification. In: International Conference on Pattern Recognition and Machine Intelligence, pp. 407–417. Springer (2023)
23. Ouahab, A., Ben Ahmed, O.: ProtoMed: prototypical networks with auxiliary regularization for few-shot medical image classification. Image Vis. Comput. 154, 105337 (2025)

24. Pereira, D.G., Afonso, A., Medeiros, F.M.: Overview of friedman's test and post-hoc analysis. Commun. Stat. Simul. Comput. **44**(10), 2636–2653 (2015)
25. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. **40**, 99–121 (2000)
26. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vision **115**, 211–252 (2015)
27. Sbai, O., Couprie, C., Aubry, M.: Impact of base dataset design on few-shot image classification. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16, pp. 597–613. Springer (2020)
28. Singh, R., Bharti, V., Purohit, V., Kumar, A., Singh, A.K., Singh, S.K.: MetaMed: few-shot medical image classification using gradient-based meta-learning. Pattern Recogn. **120**, 108111 (2021)
29. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
30. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
31. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-UCSD birds-200-2011 dataset (2011)
32. Yan, J., Feng, K., Zhao, H., Sheng, K.: Siamese-prototypical network with data augmentation pre-training for few-shot medical image classification. In: 2022 2nd International Conference on Frontiers of Electronics, Information and Computation Technologies (ICFEICT), pp. 387–391. IEEE (2022)
33. Zhang, B., Gao, B., Liang, S., Li, X., Wang, H.: A classification algorithm based on improved meta learning and transfer learning for few-shot medical images. IET Image Proc. **17**(12), 3589–3598 (2023)
34. Zhang, C., Cui, Q., Ren, S.: Few-shot medical image classification with MAML based on dice loss. In: 2022 IEEE 2nd International Conference on Data Science and Computer Application (ICDSCA), pp. 348–351. IEEE (2022)
35. Zhou, L., Cui, P., Jia, X., Yang, S., Tian, Q.: Learning to select base classes for few-shot classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4624–4633 (2020)
36. Zhu, W., Wang, X.: Automated Machine Learning and Meta-learning for Multimedia. Springer (2021)

From Semantic Segmentation of Natural Images to Medical Image Segmentation Using ViT-Based Architectures

Alexandru Valentin Pătraşcu[1(✉)], Ciprian-Mihai Ceauşescu[1], and Bogdan Alexe[1,2]

[1] University of Bucharest, Bucharest, Romania
patrascuvalentinalexandru@gmail.com,
ciprian-mihai.ceausescu@drd.unibuc.ro
[2] Gheorghe Mihoc-Caius Iacob Institute of Mathematical Statistics and Applied Mathematics of the Romanian Academy, Bucharest, Romania
bogdan.alexe@fmi.unibuc.ro

Abstract. We address the problem of medical image segmentation in the context of limited training data. Our approach builds on the capabilities of the Vision Transformer (ViT) and the recent Segmenter model, adapting them for the task of medical image segmentation. By leveraging Segmenter models pre-trained on moderately-sized datasets like ADE20K, we demonstrate their effectiveness when fine-tuned on smaller and scarce medical imaging datasets, specifically those for skin lesions and polyps. Employing our proposed training strategy, the adapted Segmenter model both matches and surpasses the current state-of-the-art on three key medical image datasets: ISIC2018 for skin lesions, and CVC-ClinicDB and ETIS-LaribPolypDB for polyps, while maintaining competitive performance on Kvasir-SEG and CVC-ColonDB.

Keywords: Medical image segmentation · Vision transformers · Segmenter model

1 Introduction

Medical image segmentation poses the problem of partitioning precisely and reliably a medical image into multiple segments for the purposes of diagnosis, treatment planning, and quantitative analysis. Usually, Convolutional Neural Networks (CNNs) [12,13] and various common architectures have been used for achieving this goals. More recently, Vision Transformers (ViTs) [7] have revolutionized computer vision, enhancing image classification and segmentation by capturing long-range interactions. Their capacity to capture global dependencies among semantic segments is primarily due to their robust attention mechanisms which typically require pre-training on substantial datasets. The limited availability of annotated training data in medical imagery presents a significant challenge to the development of complex imaging tools relying only

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 112–121, 2025.
https://doi.org/10.1007/978-3-031-80507-3_12

on medical data. To overcome this limitation, researchers have used pre-trained models to greatly enhance performance and efficiency by employing the existing knowledge from other datasets. The authors of [21] demonstrate that a common strategy to address the challenges of scarce medical datasets is to start from a pre-trained model on ImageNet and fine-tune it further on a medical dataset, in order to shift the distribution of ImageNet features learned from natural images to a feature distribution specific to medical images. Similarly, the work of [4] uses PatchCore [16], a state-of-the-art anomaly detection method specialized for visual inspection of industrial natural images, tailored to the task of classifying and segmenting lung cancer from CT scan images and brain tumor from MRI images. We follow this paradigm and consider a state-of-the-art ViT architecture for semantic image segmentation, the Segmenter model proposed by [18]. The ViT architecture contained by the Segmenter model captures global image context without using convolutions and attains competitive performance on standard image segmentation benchmarks for natural images. Our proposed method for medical image segmentation explores the benefits of using the Segmenter model, pre-trained on ADE20K [24] for the task of semantic segmentation in natural images and fine-tuned for specific tasks in medical imaging such as polyp and skin lesion segmentation. In summary our contribution is twofold: (1) we explore the benefits of using Segmenter model for semantic segmentation in medical images; (2) we thoroughly evaluate the fine-tuned Segmenter model for the task of segmenting polyp and skin lesion images and provide an in-depth analysis of using various setups wrt state-of-the art methods for several datasets.

2 Datasets

Integrating data from multiple sources is necessary to improve diagnosis accuracy, to make better and informed decisions. To evaluate the performance of the proposed method, we use five publicly available 2D medical image datasets, one for skin lesions, one for gastrointestinal polyps and three for colon polyps:

1. ISIC2018 dataset [5] designed for skin lesion analysis to improve melanoma diagnosis, consisting of 3694 images of sizes ranging from 540×576 to 4499×6748 pixels. The images were extracted from the Task 1 of the competition.
2. Kvasir-SEG dataset [11] contains 1000 images of gastrointestinal polyps along with corresponding segmentation masks and bounding boxes. These were manually annotated by a doctor and validated by a gastroenterology expert. The size of the images vary from 332×487 to 1920×1072.
3. CVC-ClinicDB dataset [1] originates from clinical cases at a hospital in Barcelona, Spain, and is derived from 23 standard white light colonoscopy intervention videos. It comprises 612 high-resolution color images, each with a resolution of 576×768 pixels, obtained from clinical colonoscopy examinations. Each image is accompanied by a manual annotation file that precisely marks the location of the polyp.

4. CVC-ColonDB dataset [2] is maintained by the Computer Vision Center (CVC) in Barcelona. It includes 380 colonoscopy images, each with a resolution of 500×574 pixels, along with manually annotated segmentation masks that precisely mark the location of polyps.
5. ETIS-LaribPolypDB dataset [3] collects early colorectal polyp images, consisting of 196 polyp instances of size 966×1225.

3 Related Work

There are several works tailored to the specific medical image segmentation task that they tackle. We present the most representative of them, the ones that we also compare to in Sect. 5.

Works on Polyp Segmentation. A typical architecture, U-Net [15], originally designed for biomedical image segmentation, that follows an encoder-decoder model, delivers strong results in different medical image segmentation tasks. PraNet [10], a parallel reverse attention network for precise segmentation of colorectal polyps in colonoscopy images, addresses challenges such as diverse polyp characteristics and indistinct boundaries. It significantly increases segmentation accuracy and real-time efficiency, validating the model across multiple datasets and metrics. In [23], the authors proposed FCN-Transformer, a new colonoscopy polyp segmentation architecture combining transformer-based feature extraction with a fully convolutional branch to predict full-size segmentation maps from high-resolution images. This approach achieves high performance across Kvasir-SEG and CVC-ClinicDB datasets, demonstrating superior generalization by training on one dataset and evaluating on another. DUCK-Net [9], is a novel convolutional neural network, employing an encoder-decoder structure with a residual downsampling mechanism and custom convolutional blocks to effectively process multi-resolution image data. By utilizing robust data augmentation techniques, DUCK-Net achieves very good results on major benchmarks (Kvasir-SEG, CVC-ClinicDB, CVC-ColonDB, ETIS-LaribPolypDB), demonstrating strong generalization capabilities despite limited training data. We compare to all these methods on the four datasets for polyp segmentation in Sect. 5.

Works on Skin Lesion Segmentation. MDViT [14], is a novel multi-domain Vision Transformer for medical image segmentation that uses domain adapters and mutual knowledge distillation to enhance performance and mitigate negative knowledge transfer across multiple small datasets. TransFuse [22], employs a parallel-in-branch network that combines Transformers and CNNs to efficiently capture global dependencies and low-level spatial details, achieving promising results with fewer parameters and faster inference on various medical image sets. BAT [20] enhances skin lesion segmentation by integrating a boundary-wise attention gate (BAG) into transformers, effectively capturing global dependencies and local details to address challenges like melanoma variation and ambiguous boundaries, outperforming other methods. Swin UNETR [19] uses a hierarchical Swin transformer encoder to effectively capture long-range information

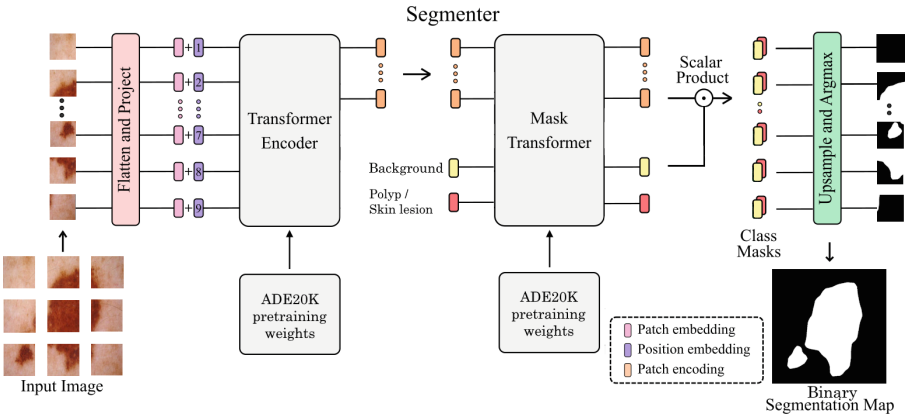


Fig. 1. Overview of the Segmenter model on the skin lesion segmentation task. *Encoder (left): the image patches are transformed into a sequence of embeddings, which are then processed by a transformer. Decoder (right): The mask transformer uses the encoder's output and class embeddings to predict segmentation masks. Image adapted from* [18].

and an FCNN-based decoder with skip connections to achieve high performance on brain tumor segmentation tasks. We compare to all these methods on the ISIC2018 dataset for skin lesion segmentation in Sect. 5.

4 Method

We build our method on top of the state-of-the-art Segmenter model adapting it for the task of segmenting medical images (Fig. 1). The Segmenter encoder closely follows the original ViT encoder, maintaining its core principles and architectural design. Specifically, the image is divided into patches, which are then flattened and linearly projected into embeddings. Positional embeddings are added to retain spatial context, forming the input sequence for the transformer. The transformer encoder, composed of multiple layers, employs multi-head self-attention and point-wise Multi-Layer-Perceptron (MLP) blocks, each with layer normalization and residual connections. The sequence of patch encodings $z_L \in \mathbb{R}^{N \times D}$ is transformed into a segmentation map $s \in \mathbb{R}^{H \times W \times K}$, where K represents the number of classes, N represents the number of patches, D is the dimensionality of the patch embeddings, H and W refer to the height and width of the original input image. The decoder maps these patch-level encodings to class scores, which are then upsampled by bilinear interpolation to achieve pixel-level accuracy. Two different decoders were implemented in the original paper, a point-wise linear layer that was used as a baseline, and a mask transformer. For the transformer-based decoder, a set of K learnable class embeddings $cls = [cls_1, \ldots, cls_K] \in \mathbb{R}^{K \times D}$ is introduced, where K is the number of classes, and D the latent vector size. These embeddings are processed with patch encodings z_L by the decoder, consisting of M layers. The mask transformer generates K masks by computing the scalar product between L2-normalized patch

embeddings $z'_M \in \mathbb{R}^{N \times D}$ and class embeddings $c \in \mathbb{R}^{K \times D}$. The class masks are computed as:

$$\text{Masks}(z'_M, c) = z'_M c^T \tag{1}$$

where $\text{Masks}(z'_M, c) \in \mathbb{R}^{N \times K}$ is a set of patch sequences. Each sequence is reshaped into a 2D mask $s_{\text{mask}} \in \mathbb{R}^{H/P \times W/P \times K}$, where (P, P) represents the patch size, and upsampled to $s \in \mathbb{R}^{H \times W \times K}$ using bilinear interpolation. A softmax is applied on the class dimension to obtain pixel-wise class scores, forming the final segmentation map. In our case K = 2, the background and foreground classes.

5 Experimental Evaluation

We evaluate our method for medical image segmentation of polyps and skin lesions. We compare our approach to several baselines using both convolutional and transformer-based approaches.

Evaluation Measures. We employ commonly used evaluation metrics tailored to medical image segmentation tasks. Specifically, we use two key evaluation metrics: (i) Jaccard Index, also known as Intersection over Union (IoU) which compares the similarity between the predicted segmentation and the ground truth; (ii) Dice Similarity Coefficient (DSC) which quantifies the overlap between the predicted segmentation and the ground truth segmentation.

Loss Functions. In addition, three distinct loss functions were investigated to evaluate their effectiveness in medical image segmentation tasks: (i) Binary Cross-Entropy (BCE) loss which measures the discrepancy between the predicted probabilities and the actual binary labels; (ii) DSC loss which maximizes the overlap between predicted and ground truth segmentations by minimizing the Dice Coefficient; (iii) Hausdorff Distance (HD) loss which focuses solely on the largest error.

5.1 Segmenter Models

We use the small (Seg-S), base (Seg-B), and large (Seg-L) variants of the Segmenter model, each of them based on the ViT architecture, as described in Table 1. The variable parameters within the transformer encoder include the number of layers and the token size. The head size of a multi-headed self-attention (MSA) block remains constant at 64. The number of attention heads is determined by dividing the token size by the head size. Additionally, the hidden layer size of the MLP that follows the MSA is set to be four times the token size. The Segmenter models have their ViT encoders pre-trained on ImageNet-21k using robust data augmentation and regularization techniques [17]. The models are initially pre-trained at an image resolution of 224 × 224 and subsequently fine-tuned on ImageNet-1k at a resolution of 384 × 384. The pre-trained Segmenter models

Table 1. Segmenter variants configurations.

Model	Backbone	Layers	Token size	Heads	Params
Seg-S	ViT-S	12	384	6	22M
Seg-B	ViT-B	12	768	12	86M
Seg-L	ViT-L	24	1024	16	307M

were further fine-tuned [18] on various semantic segmentation datasets. The one that we consider in our work is ADE20K [24]. The ADE20K dataset serves as a comprehensive benchmark for semantic segmentation offering a rich diversity of annotations, covering a wide range of scenes, objects, parts of objects, and even sub-parts. The diversity and richness of ADE20K make it an ideal choice for pre-training, as it allows models to learn complex visual patterns and relationships, thereby enhancing their generalization and performance on other segmentation tasks with scarce data, as in our case. To fine-tune the pre-trained Segmenter models, we use the open-source semantic segmentation toolbox *mmsegmentation* [6], which provides a standardized framework for implementing, training, and evaluating state-of-the-art segmentation models. For the loss functions, we use the BCE and DSC loss functions either individually, combined together, or in conjunction with the HD loss. As an optimizer, we employ the Stochastic Gradient Descent (SGD) with a base learning rate γ_0 and a weight decay set to 0. Also, we use the poly learning rate decay, $\gamma = \gamma_0(1 - \frac{N_{\text{iter}}}{N_{\text{total}}})^{0.9}$, where N_{iter} and N_{total} represent the current iteration number and the total iteration number, respectively. We use a base learning rate γ_0 of 10^{-3}, and train the models for 30K iterations with a batch size of 8.

5.2 Experiments on Polyp Datasets

We report results on the four polyp datasets described in Sect. 2, namely the Kvasir-SEG, CVC-ClinicDB, CVC-ColonDB and ETIS-LaribPolypDB datasets. In our experiments we follow the train-validation-test split proposed in [8] employing also their evaluation script which is publicly available. Table 2 show the results by comparing the three variants of our method (small, base and large Segmenter models) to the different baselines described in Sect. 3. Our models show in general a very good performance, with all of the three variants performing almost equally good. On average we notice a slightly 1% difference between the best performing variant (Seg-B) and the worse variant (Seg-S). In particular, Seg-B achieves the highest performance on CVC-ClinicDB dataset (IoU: 0.902, Dice: 0.948) and ETIS-LaribPolypDB dataset (IoU: 0.879, Dice: 0.936). On average, our Seg-B method is within 1% IoU and Dice score to the best performing method DUCK-Net which uses 34 filters. These results underline the efficacy of our models in achieving high-quality medical image segmentation. In conclusion, Segmenter models pre-trained on ADE20K and fine-tuned on specific polyp datasets match the state-of-the-art methods designed for polyp segmentation.

Table 2. Quantitative results. *Datasets: Kvasir-SEG (KV-SEG), CVC-ClinicDB (ClinicDB), CVC-ColonDB (ColonDB), ETIS-LaribPolypDB (ETIS). The two DUCK-Net variants mean:* [1] *model 17 filters,* [2] *model 34 filters.*

	KV-SEG		ClinicDB		ColonDB		ETIS		average	
Method	IoU↑	Dice↑	IoU↑	Dice↑	IoU↑	Dice↑	IoU↑	Dice↑	IoU↑	Dice↑
U-Net [15]	0.763	0.866	0.617	0.763	0.704	0.803	0.697	0.798	0.695	0.807
PraNet [10]	0.834	0.909	0.777	0.874	0.840	0.913	0.790	0.883	0.810	0.894
FCN-Transf. [23]	0.855	0.922	0.874	0.933	0.830	0.907	0.846	0.916	0.851	0.919
DUCK-Net [1] [9]	0.877	0.934	0.895	0.945	**0.879**	**0.935**	0.873	0.932	0.881	0.936
DUCK-Net [2] [9]	**0.905**	**0.950**	0.901	**0.948**	0.857	0.923	**0.879**	0.935	**0.885**	**0.939**
our-Seg-S	0.878	0.935	0.884	0.939	0.854	0.921	0.845	0.916	0.865	0.927
our-Seg-B	0.886	0.940	**0.902**	**0.948**	0.852	0.920	**0.879**	**0.936**	0.879	0.936
our-Seg-L	0.899	0.947	0.883	0.938	0.849	0.918	0.850	0.921	0.870	0.931

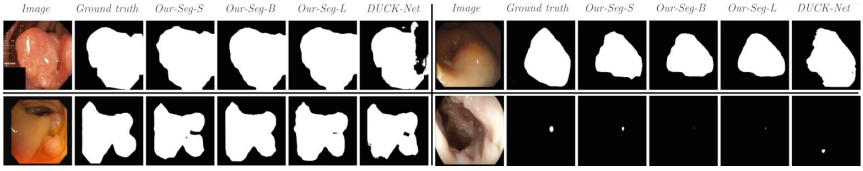


Fig. 2. Qualitative analysis on polyp datasets. *We present results of different methods for several datsets: Kvasir-SEG (top-left), CVC-ClinicDB (bottom-left), CVC-ColonDB (top-right), ETIS-LaribPolypDB (bottom-right).*

In Fig. 2, we present qualitative analysis on all four polyp datasets highlighting the segmentation accuracy of our models (Seg-S, Seg-B, Seg-L) compared to DUCK-Net. The figure illustrates that our models not only match, but often surpass the DUCK-Net in terms of segmentation precision, clearly delineating polyp boundaries and maintaining high similarity to the ground truth images.

5.3 Experiments on Skin Lesion Dataset

In our experiments we follow the training split procedure described in [14], i.e. 5-fold cross-validation on the ISIC2018 training set. Table 3 shows the results, comparing our two variants of Segmenter models (small and base) to the various baselines described in Sect. 3. Firstly, we notice that both our variants perform equally well. Our best Segmenter model, pre-trained on ADE20K, achieves an IoU of 0.841 with a standard deviation of 0.014 and an Dice of 0.911 with a standard deviation of 0.010, indicating consistent and robust performance across the 5-fold cross-validation setup. Secondly, we observe, that

Table 3. Quantitative results for skin lesion dataset ISIC2018.

Method	IoU ↑	Dice ↑
MDViT [14]	0.830	0.903
TransFuse [22]	0.832	0.904
BAT [20]	0.830	0.905
Swin UNETR [19]	0.829	0.903
our-Seg-S	**0.841**	**0.911**
our-Seg-B	0.840	0.910

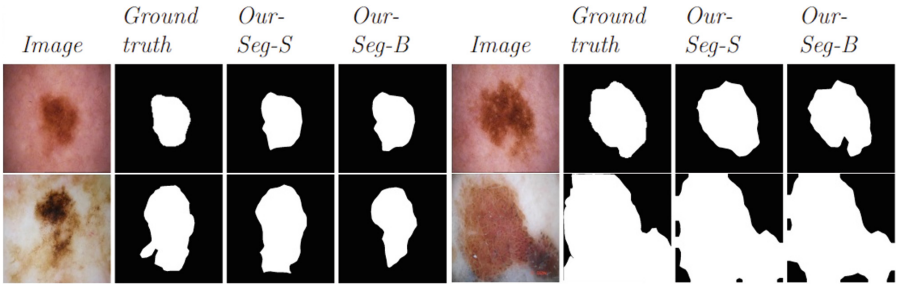


Fig. 3. Qualitative analysis on ISIC2018 dataset of our method.

compared to other models reported in the work of [14], Segmenter outperforms all listed methods in terms of both IoU and Dice. Notably, it surpasses MDViT [14], TransFuse [22], and BAT [20], which are among the top-performing models, showcasing Segmenter's superior capability in capturing and accurately segmenting the intricate details in skin lesion images.
Figure 3 presents qualitative results on the ISIC2018 dataset highlighting the segmentation accuracy of our models, the small and base variants of the Segmenter model. Both variants demonstrate strong segmentation capabilities, closely approximating the ground truth. Although there are minor variations in detail capture and edge definition among the variants, these differences are slight, underscoring the models' overall effectiveness in accurate skin lesion segmentation.

5.4 Ablation Studies

Tables 4 and 5 provide a series of ablation studies for assessing the importance of various components and methodologies employed in our segmentation framework. All the four ablation experiments presented in Tables 4 and 5 were performed using the official ISIC2018 dataset split, consisting of 2594/100/1000 images for train/validation/test. In terms of loss functions to be used, our exper-

Table 4. Ablation studies for assessing the importance of different components used in our framework. *We study: (i) the use of different loss functions and their combinations; (ii) the importance of using pre-trained models on ADE20K; (iii) the impact of the model size.*

Loss functions			*Segmenter pre-training*			*Segmenter size*		
Loss	IoU ↑	Dice ↑	Pre-training (PT)	IoU ↑	Dice ↑	Model	IoU ↑	Dice ↑
Dice	0.811	0.887	No PT (Seg-S)	0.827	0.897	Seg-S	**0.836**	**0.902**
BCE	**0.836**	**0.902**	ADE20K PT (Seg-S)	**0.836**	**0.902**	Seg-B	0.832	0.900
Dice ⊕ BCE	0.819	0.893	No PT (Seg-B)	0.809	0.886	Seg-L	0.832	0.901
Dice ⊕ HD	0.763	0.857	ADE20K PT (Seg-B)	**0.832**	**0.900**			
BCE ⊕ HD	0.801	0.880						

Table 5. Optimizer variants configurations.

Optimizer	IoU ↑	Dice ↑
Adam	0.805	0.892
SGD	**0.836**	**0.902**
AdamW	0.818	0.900

imental results done on the Seg-S variant indicate that BCE loss provides best results, while combining it with the Dice or HD losses is harmful. Pre-training on ADE20K is helpful, providing an increase between 0.9 and 2.3% in terms of IOU while different variants of the Segmenter (Small, Medium, Large) provide very similar performance in terms of both IoU or Dice scores when trained with the BCE loss. Additionally, the optimizer ablation study presented in Table 5 shows that SGD outperforms Adam and AdamW, achieving higher IoU and Dice scores. Despite Adam-based optimizers offering faster convergence, SGD's slower, more controlled optimization seems to generalize better.

6 Conclusions and Future Work

In this paper we proposed a method to leverage a high-performing segmentation model pre-trained on a comprehensive natural images dataset (ADE20K) and further fine-tuning it on medical image datasets of polyps and skin lesions. In future work we would like to investigate the importance of the dataset used for pre-training by incorporating in our framework datasets such as Cityscapes and Pascal-Context instead of ADE20K [18].

References

1. Bernal, J., Sanchez, F.J., Fernández-Esparrach, G., Gil, D., Rodríguez, C., Vilariño, F.: Wm-dova maps for accurate polyp highlighting in colonoscopy: validation vs. saliency maps from physicians. Comput. Med. Imaging Graph. **43**, 99–111 (2015)
2. Bernal, J., Sánchez, J., Vilarino, F.: Towards automatic polyp detection with a polyp appearance model. Pattern Recogn. **45**(9), 3166–3182 (2012)
3. Bernal, J., et al.: Comparative validation of polyp detection methods in video colonoscopy: results from the MICCAI 2015 endoscopic vision challenge. IEEE Trans. Med. Imaging **36**(6), 1231–1249 (2017)
4. Ceauşescu, C.M., Alexe, B., Volpi, R.: Coreset based medical image anomaly detection and segmentation. In: Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, pp. 549–558. INSTICC, SciTePress (2024)
5. Codella, N., et al.: Skin lesion analysis toward melanoma detection 2018: a challenge hosted by the international skin imaging collaboration (ISIC) arXiv:1902.03368 (2018)

6. Contributors, M.: MMSegmentation: openmmlab semantic segmentation toolbox and benchmark (2020). https://github.com/open-mmlab/mmsegmentation
7. Dosovitskiy, A., et al.: An image is worth 16×16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
8. Dumitru, R.G.: Duck-net (2023). https://github.com/RazvanDu/DUCK-Net. Accessed 01 Jun 2024
9. Dumitru, R.G., Peteleaza, D., Craciun, C.: Using duck-net for polyp image segmentation. Sci. Rep. **13**(1), 9803 (2023)
10. Fan, D.P., et al.: PraNet: parallel reverse attention network for polyp segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 263–273. Springer (2020)
11. Jha, D., et al.: Kvasir-SEG: a segmented polyp dataset. In: International Conference on Multimedia Modeling, pp. 451–462. Springer (2020)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in NIPS, vol. 25 (2012)
13. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
14. Liu, X., et al.: MDViT: multi-domain vision transformer for medical image segmentation. arXiv preprint arXiv:2307.02100 (2023)
15. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18, pp. 234–241. Springer (2015)
16. Roth, K., Pemula, L., Zepeda, J., Scholkopf, B., Brox, T., Gehler, P.: Towards total recall in industrial anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14298–14308 (2022). https://doi.org/10.1109/CVPR52688.2022.01392
17. Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your ViT? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270 (2021)
18. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: transformer for semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7262–7272 (2021)
19. Tang, Y., et al.: Self-supervised pre-training of swin transformers for 3D medical image analysis. In: CVPR (2022)
20. Wang, J., Wei, L., Wang, L., Zhou, Q., Zhu, L., Qin, J.: Boundary-Aware Transformers for Skin Lesion Segmentation. Springer International Publishing (2021)
21. Xie, Y., Richmond, D.: Pre-training on grayscale imagenet improves medical image classification. In: Leal-Taixé, L., Roth, S. (eds.) Computer Vision - ECCV 2018 Workshops, pp. 476–484. Springer International Publishing, Cham (2019)
22. Zhang, Y., Liu, H., Hu, Q.: Transfuse: Fusing transformers and CNNs for medical image segmentation. In: MICCAI 2021. Springer (2021)
23. Zheng, S., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR (2021)
24. Zhou, B., et al.: Semantic understanding of scenes through the ADE20K dataset. Int. J. Comput. Vis. **127**, 302–321 (2019)

Chronic Wound Assessment with Semi-supervised Hierarchical CNNs

Shahram Ghahremani(✉)

Department of Electrical Engineering and Computer Science, York University, Toronto, ON M3J 1P3, Canada
shg@yorku.ca

Abstract. The assessment of chronic wound healing often depends on visual inspection, but subjective methods can introduce variability and hinder reliable predictions. Current computer-aided detection (CAD) tools encounter limitations in automated assessment, particularly in the fine-grained analysis of tissues with irregular textures. This paper introduces a new approach, a semi-supervised hierarchical convolutional neural network (SHCNN), for robust and efficient chronic wound assessment. SHCNN simultaneously handles two crucial tasks: accurate wound size measurement and fine-grained segmentation of seven tissue types. The model is initially trained using limited pixel-level annotations, and later relies on image-level labels, alleviating the need for scarce and arduously obtained pixel-level annotations. Evaluation on a dataset with 1200 image-level and 600 pixel-level annotations demonstrates the effectiveness of SHCNN. It achieves an average pixel-wise accuracy of 81% for tissue segmentation and measurement errors of 2% (small wounds) and 3% (large wounds). These findings suggest that SHCNN has the potential to replace manual assessment, providing a valuable tool for automated chronic wound management.

Keywords: Automatic wound measurements · Automatic wound segmentation · Convolutional neural networks

1 Introduction

Chronic wounds, characterized by a prolonged healing process extending beyond a month, impose a substantial burden on individuals and healthcare systems [23]. In the USA, these wounds affect about 2% of the population, impacting 5.7 million people and costing over US$20 billion annually [9]. This cost is expected to grow due to aging populations, rising healthcare costs, and the global increase in diabetes and obesity. Manual wound assessment is time-consuming and prone to errors, making a shift to automated approaches essential. Automation could reduce clinicians' workload and improve accuracy and consistency in wound assessment.

Assessing chronic wounds involves measuring wound size and analyzing tissue types. Accurate measurement is key for tracking healing, but no single method

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 122–132, 2025.
https://doi.org/10.1007/978-3-031-80507-3_13

is considered best due to the difficulty of measuring complex wound shapes. Traditional methods using rulers risk contamination and may not accurately measure irregular shapes, making them unsuitable for many chronic wounds.

Analyzing tissue types is also crucial for assessing healing, providing important information about the wound's stage. Manual examination is subjective and slow, relying on visual assessment. While image processing can automate this analysis, it faces challenges in feature extraction and engineering.

In this paper, we introduce a hierarchical semi-supervised convolutional neural network (HSCNN) for automated wound assessment. HSCNN performs two tasks: segmenting both a predefined marker and the wound area for measurement, and segmenting seven distinct tissue types from the wound area.

Expert pixel-level annotations for clinical images are challenging to obtain due to time constraints on wound nurses. To address this, our architecture combines easily available image-level labels with more time-consuming pixel-level annotations. By using both weakly and strongly labeled data, HSCNN trains a robust segmentation network for accurate wound assessment.

Following are the major contributions of our work:

- Multitask efficiency: HSCNN accurately measures wound size and performs fine-grained segmentation of seven tissue types, offering clinicians valuable insights for tailored treatment decisions.
- Reduced annotation burden: Using a semi-supervised approach, HSCNN efficiently utilizes image-level labels with limited pixel-level annotations, reducing the need for extensive manual labeling and easing the workload on clinicians.
- High accuracy and performance: HSCNN, evaluated on a dataset with 1200 image-level and 600 pixel-level annotations, achieves 81% accuracy for tissue segmentation and low measurement errors, showing its potential to replace manual assessment.

2 Background and Related Work

In modern medical practice, chronic wound tissues are classified into seven types: healthy granulating, unhealthy granulating, hypergranulating, infected, necrotic, sloughy, and epithelizing [18]. Healthy granulating tissue shows new tissue forming, with tiny blood vessels on a light red or pink moist surface. Unhealthy granulating tissue, which can appear dark red, bluish, or pale, indicates problems like infection or poor blood supply. Hypergranulating tissue grows above the wound margin due to prolonged healing phases. Infected tissue is greenish, with a foul odor, showing bacterial infection. Necrotic tissue is black, indicating dead skin cells. Sloughy tissue, a wet necrotic type, detaches from the wound, appearing white, yellow, or grey. Epithelizing tissue is made up of cells forming protective layers over granulating tissue.

Research on automating chronic wound assessment mainly falls into two categories: wound area measurement and wound tissue segmentation. The following subsections review related work in these areas.

2.1 Wound Area Measurement

Wound measurement is key for tracking healing, assessing treatment, and identifying stagnant wounds. Traditional methods, like ruler-based measurements, are accessible but often inaccurate, especially with irregular wound shapes and infection risks [25].

To overcome these issues, various image processing techniques have been proposed [5]. However, robust and accurate automated wound measurement remains challenging due to tissue complexities.

Recent AI advances, especially deep learning, offer promising solutions for wound assessment [12]. Wang et al. used fully convolutional networks (FCN), achieving 64.2% mean Dice accuracy [24]. Goyal et al. proposed FCN-16, reaching 79.4% Dice accuracy but struggled with small and irregular wounds [10]. Liu et al. introduced an FCN architecture, achieving 91.6% Dice accuracy with semi-automatic annotations [16]. Wang et al. developed a MobileNetsV2-based framework balancing efficiency and accuracy [25].

In our study, we propose an efficient and accurate framework based on U-Net architecture [22]. Details of our model are discussed in Sect. 3.

2.2 Wound Tissue Segmentation

Wound tissue segmentation involves pixel-wise classification within the wound bed, crucial for wound management. Traditional methods use color clustering, thresholding, or machine learning for patch-based classification [6,18]. While these methods offer more objectivity than visual assessment, they are often computationally intensive, time-consuming, and sensitive to image variations like lighting and angle.

Recent advances in deep learning, especially semantic segmentation models, offer promising solutions by autonomously learning features in medical images. Li et al. [15] combined dynamic color thresholding with a convolutional neural network for wound tissue segmentation, requiring pre-segmentation of the wound area.

Ramachandram et al. [21] proposed a cascaded deep learning approach for wound detection and tissue classification, using a deep convolutional encoder-decoder for wound detection, followed by EfficientNetB0 for tissue segmentation. Brayan et al. [17] introduced a similar dual-stage approach using YOLOv4 for wound detection and a U-Net model for segmentation.

Our study builds on these advancements with a fully automated cascaded model for wound area segmentation (first stage) and wound tissue segmentation (second stage). To address challenges with pixel-level annotations, we use a semi-supervised learning approach, improving the precision and efficiency of wound measurement and tissue segmentation.

3 Methodology

We propose a hierarchical semi-supervised convolutional neural network (HSCCN) to address the challenges of measuring wound area and segmenting

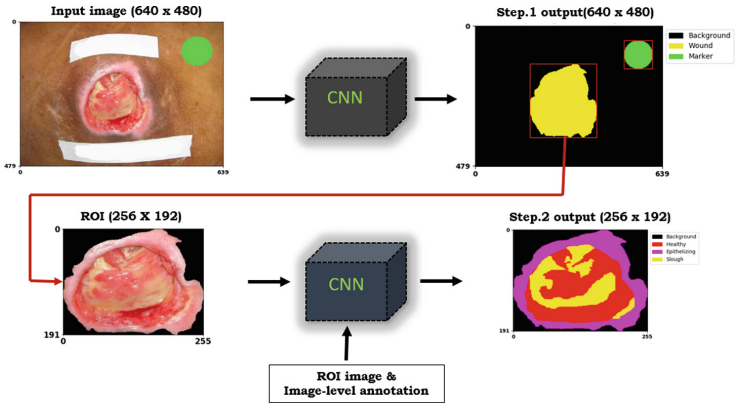


Fig. 1. Semi-supervised Hierarchical Convolutional Neural Network(HSCNN).

wound tissues. The HSCNN architecture, illustrated in Fig. 1, consists of two distinct stages:

- Stage 1: Wound area measurement
  Leveraging an encoder-decoder U-Net architecture [22], the HSCCN performs semantic segmentation on a 640×480 input image, precisely delineating the wound area and a reference marker with known dimensions from the background. This facilitates robust pixel-to-inch conversion calculations for accurate wound size quantification.
- Stage 2: Wound tissue segmentation
  Focusing on the segmented wound area (region of interest, ROI) extracted from Stage 1, a subsequent CNN model performs fine-grained tissue type segmentation on a 256×192 input image. This cascaded architecture fosters specialization, with the first stage adept at holistic wound detection and segmentation within the full clinical image, and the second stage meticulously differentiating individual tissue types within the extracted ROI. The hierarchical structure further mitigates false positive tissue classifications by suppressing irrelevant background information.

Training the second-stage model necessitates pixel-wise labels for ROI images. However, acquiring such detailed annotations poses challenges in clinical settings due to workload constraints. To address this, the HSCCN employs a semi-supervised learning approach, effectively leveraging both pixel-level and image-level annotations to enhance tissue segmentation efficiency.

This dual-stage framework, coupled with the strategic implementation of semi-supervised learning, empowers the HSCCN to deliver comprehensive wound assessment capabilities, encompassing both precise size quantification and fine-grained tissue analysis. Subsequent sections will delve deeper into the intricacies of pixel-wise tissue type segmentation within stage 2, elucidating the specific

techniques and tools employed by the HSCCN to become a valuable asset in wound care management.

3.1 Semi-supervised Tissue Segmentation Learning

Figure 2 illustrates the block diagram of the second stage of the proposed HSCNN. In this depiction, we actively incorporate a small number of pixel-level annotations while emphasizing the utilization of image-level annotations across a substantial quantity of images to enhance segmentation performance, establishing a semi-supervised learning setting.

Our hierarchical model integrates both strongly (pixel-level) and weakly (image-level) annotated images in the second stage, facilitating the learning of segmentation networks by sharing parameters. In this innovative approach, the ROIs from both weakly and strongly annotated images serve as inputs to a shared CNN model.

The shared CNN model processes inputs from both sources and aims to learn similar features present in the two wound input images (ROIs). In the last layer, the shared CNN produces two branches, each learning specific features corresponding to its ground truth. Notably, confidence maps corresponding to image-level labels are extracted for inferring segmentation masks in the upper (loss 1) branch of Fig. 2. Meanwhile, in the lower (loss 2) branch of Fig. 2, the model engages in a typical semantic segmentation task with pixel-wise labels.

This semi-supervised approach not only significantly alleviates the need for laborious pixel-level annotations, a common bottleneck in medical image segmentation, but also unlocks broad application potential beyond chronic wound tissue segmentation. Its flexibility extends to various visual image analysis tasks, promising wider contributions in the realm of medical imaging.

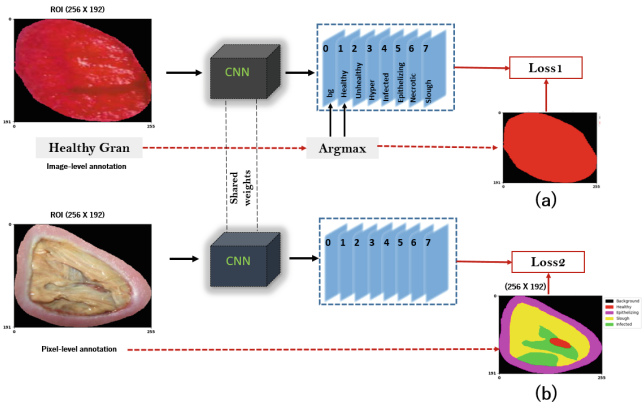


Fig. 2. Details of training semantic segmentation in semi- supervised manner with our proposed approach. In particular, (a) is the online predicted segmentation mask; (b) is the human annotated segmentation mask.

4 Experimental Evaluation

4.1 Dataset

Our dataset comprises 1800 images collected from various sources, including publicly available wound databases, including the Medetec Wound Database [3], as well as online sources such as BioElectronics Clinical Studies [1], an Indian community-based epidemiological study of wounds [11], Medscape [4], and Disease Picture [2]. Among these images, 1,200 are weakly annotated (image-level annotation), while the remaining 600 images, are meticulously annotated with pixel-level detail.

All the images in our dataset are captured under different conditions, including variations in illumination, pose, etc. Furthermore, they are taken with different camera devices and exhibit diverse resolutions, ranging from 216×158 to 4208×2368.

4.2 Validation and Evaluation Metrics

To validate the effectiveness of the proposed model, the complete dataset is split into training (80%) and test (20%) datasets. Later, we choose randomly 80% of data from the training dataset to train the proposed model and rest of 20% is used as validation dataset for cross validation to avoid overfitting. We assess the performance of our proposed model using two approach; quantitative metrics analysis and qualitative analysis. Two main tasks including wound size measurement task, and 7 tissue type segmentation task are evaluated in the two validation approaches. We perform the quantitative analysis using the Dice score, also known as F1-score as:

$$DICE(A;B) = \frac{2|A \cap B|}{|A| + |B|} \tag{1}$$

where the Dice score is in the interval [0,1]. We refer to the foreground object in the ground truth as object A, and object B for the predicted object. A perfect segmentation yields a Dice score of 1.

5 Results

In this section we present the performance of our proposed method using the quality and quantity metrics for the two main tasks; wound measurement and pixel-wise tissue segmentation.

5.1 Wound Measurement Result

Figure 3 shows the individual wound area measurements for 20 wounds, ordered from largest to smallest, using our proposed technique, the HSV method [8], and the ruler method [14] as the ground truth. The average wound area using

the proposed approach is 41.94 cm^2 and using HSV was 42.74 cm^2, while the ruler method has a mean of 41.45 cm^2. The mean difference in wound area between the proposed technique and the ruler method is 0.49 cm^2, indicating the effectiveness of the proposed approach for the wound measuring task.

5.2 Qualitative and Quantitative Results of Tissue Segmentation

The proposed architecture is evaluated on the chronic wound dataset to compare its performance with conventional methods. The corresponding results are given in Table 1. In [8], authors have shown that the conventional feature extractor methods such as HSV as color descriptors and the LBP as a texture descriptor, revealed a high discriminative power in the three tissue types scenario. However, Table 1 clearly show that HSV [8] and LBP [19] failed to reach an acceptable level of performance when tested in the seven class scenario and thus cannot be used for clinical purposes. While in three class scenarios each class can be separated using simple features like color and texture, in realistic seven class tissue types more powerful features are needed. The deep learning patch based

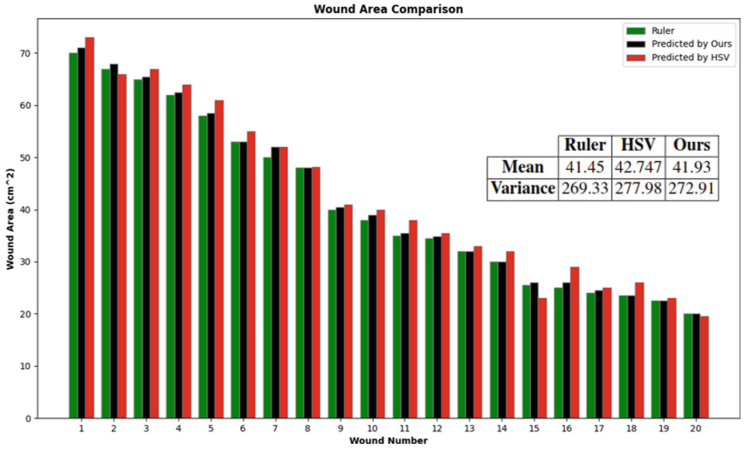


Fig. 3. A comparison of wound area measurement using the HSV and the proposed technique.

Table 1. Quantitative segmentation results of the chronic wound tissues on the wound dataset

Approach	Tissue type						
	Necrotic(%)	Slough(%)	Healthy Gran(%)	Unhealthy Gran(%)	Hyper Gran(%)	Infected(%)	Epithelizing(%)
HSV [8]	63.2	38.6	29.1	28.3	25.2	28.7	28.8
LBP [19]	64.3	47.2	31.7	33.6	24.7	27.4	26.8
AlexNet [18]	74.5	65.3	64.1	62.3	65.2	70.4	70.6
U-Net [22]	82.3	78.4	74.1	73.8	74.6	75.5	78.9
Attention U-Net [20]	84.3	76.9	75.7	75.6	75.4	76.7	79.1
Cascaded-FCNN [7]	85.7	77.1	77.1	77.4	77.5	78.5	80.2
HSCNN(Ours)	**86.1**	**80.5**	**78.9**	**80.5**	**79.8**	**80.1**	**81.6**

method which uses Alexnet model [13] to classify wound image patches [18] outperforms the conventional HSV and LBP techniques, however this approach is computationally expensive because each patch is desperately fed into the neural network and also the network does not reuse the shared features between the overlapping patches.

U-Net model [22] address the issue of patch based model by applying learnable upscaling the depth layers in AlexNet and also providing a superior design pattern of skip connections between different stages of the neural network. A further improvement can be obtained by applying attention layers in U-Net structure [20]. The attention approach allows the CCN layers to be more specific to local regions. Cacaded CNN [7] results show that extracting the region of interest (ROI) and training the ROI by second model outperforms the U-Net attention. Incorporating both weakly and strongly labels in the training of HSCNN performs consistently better in all tissue types indicating that removing irrelevant information and focusing on the interested area of the image in the training process can improve the accuracy of the model. According to the objective function defined in Sect. 4.2, the proposed hierarchical semi-supervised CNN model (HSCNN) achieves 86.1%, 80.5%, 78.9%, 80.5%, 79.8%, 80.1%, 81.6% accuracy for pixel wise segmentation of necrotic, sloughy, healthy granulating, unhealthy granulating, hyper granulating, infected, and epithelizing tissues, respectively.

The qualitative results of the tissue segmentation task are presented in Fig. 4. The complex and heterogeneous structure of the wound and all tissues were

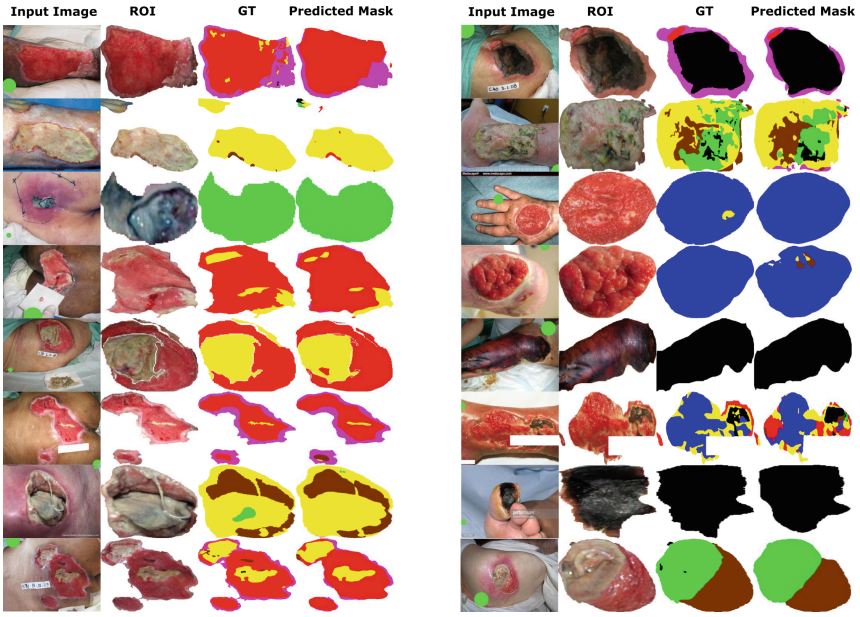


Fig. 4. Automatic chronic wound tissue segmentation with semi-supervised hierarchical convolutional neural network(SS-HCNN).

detected in the shown images. While in three class scenarios each class can be separated using simple features like color and texture, in realistic seven class tissue types more powerful features are needed. It can be observed that the proposed model is able to reach an acceptable level of performance when tested in the seven class scenario and thus can be used for clinical purposes.

6 Conclusion

This paper introduced a hierarchical semi-supervised convolutional neural network (HSCNN) for accurate wound size measurement and fine-grained segmentation of seven tissue types. By combining image-level labels with limited pixel-level annotations, HSCNN reduces clinician workload and ensures efficient training, making it suitable for clinical use. HSCNN achieves 81% pixel-wise segmentation accuracy and low measurement errors, showing its potential to improve chronic wound management by replacing subjective assessments with objective, automated analyses.

A limitation is the use of two CNN models, which challenges deployment on devices with limited computing resources. While a single CNN with two branches is a potential solution, cascaded models still perform better. Future work will focus on enhancing the model to balance accuracy with computational demands.

References

1. Chronic Wounds - BioElectronics Corporation. http://www.bielcorp.com/clinical-evidence/chronic-wounds/
2. Disease Pictures - Images, & Photos of ill. https://diseaeseshows.com/
3. Medetec Surgical Dressings and Wound Management Resource Centre - Home page. http://www.medetec.co.uk/index.html
4. Wound Care: Background, Epidemiology, Etiology (2020). https://emedicine.medscape.com/article/194018-overview, publication: Medscape - eMedicine
5. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
6. Chakraborty, C.: Computational approach for chronic wound tissue characterization. Inf. Med. Unlocked **17**, 100162 (2019). https://www.sciencedirect.com/science/article/pii/S235291481830203X
7. Christ, P.F., et alutomatic liver and tumor segmentation of CT and MRI volumes using cascaded fully convolutional neural networks. arXiv preprint arXiv:1702.05970 (2017)
8. Fauzi, M.F.A., Khansa, I., Catignani, K., Gordillo, G., Sen, C.K., Gurcan, M.N.: Segmentation and automated measurement of chronic wound images: probability map approach. In: Medical Imaging 2014: Computer-Aided Diagnosis, vol. 9035, p. 903507. International Society for Optics and Photonics (2014)
9. Frykberg, R.G., Banks, J.: Challenges in the treatment of chronic wounds. Adv. Wound Care **4**(9), 560–582 (2015). https://doi.org/10.1089/wound.2015.0635
10. Goyal, M., Yap, M.H., Reeves, N.D., Rajbhandari, S., Spragg, J.: Fully convolutional networks for diabetic foot ulcer segmentation. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 618–623. IEEE (2017). https://ieeexplore.ieee.org/document/8122675

11. Gupta, N., Gupta, S.K., Shukla, V.K., Singh, S.P.: An Indian community-based epidemiological study of wounds. J. Wound Care **13**(8), 323–325 (2004)
12. Kim, P.J., Homsi, H.A., Sachdeva, M., Mufti, A., Sibbald, R.G.: Chronic wound telemedicine models before and during the COVID-19 pandemic: a scoping review. Adv. Skin Wound Care **35**(2), 87–94 (2022). https://pubmed.ncbi.nlm.nih.gov/35050917/
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25 (2012). https://doi.org/10.1145/3065386
14. Langemo, D., Spahn, J., Spahn, T., Pinnamaneni, V.C.: Comparison of standardized clinical evaluation of wounds using ruler length by width and scout length by width measure and scout perimeter trace. Adv. Skin Wound Care **28**(3), 116 (2015)
15. Li, F., Wang, C., Liu, X., Peng, Y., Jin, S.: A composite model of wound segmentation based on traditional methods and deep neural networks. Comput. Intell. Neuroscience **2018**, 4149103 (2018)
16. Liu, X., Wang, C., Li, F., Zhao, X., Zhu, E., Peng, Y.: A framework of wound segmentation based on deep convolutional networks. In: 2017 10th International Congress on image and Signal Processing, Biomedical Engineering and Informatics (CISP-BMEI), pp. 1–7. IEEE (2017). https://ieeexplore.ieee.org/abstract/document/8302184/
17. Monroy, B., Bacca, J., Sanchez, K., Arguello, H., Castillo, S.: Two-step deep learning framework for chronic wounds detection and segmentation: a case study in Colombia. In: 2021 XXIII Symposium on Image, Signal Processing and Artificial Vision (STSIVA), pp. 1–6. IEEE (2021). https://ieeexplore.ieee.org/document/9592008
18. Nejati, H., et al.: Fine-grained wound tissue analysis using deep neural network. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1010–1014. IEEE (2018)
19. Noguchi, H., Kitamura, A., Yoshida, M., Minematsu, T., Mori, T., Sanada, H.: Clustering and classification of local image of wound blotting for assessment of pressure ulcer. In: 2014 World Automation Congress (WAC), pp. 427–432. IEEE (2014)
20. Oktay, O., et al.: Attention U-Net: learning where to look for the pancreas. arXiv preprint arXiv:1804.03999 (2018)
21. Ramachandram, D., Ramirez-GarciaLuna, J.L., Fraser, R.D., Martínez-Jiménez, M.A., Arriaga-Caballero, J.E., Allport, J.: Fully automated wound tissue segmentation using deep learning on mobile devices: cohort study. JMIR mHealth uHealth **10**(4), e36977 (2022). https://mhealth.jmir.org/2022/4/e36977/
22. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 234–241. Springer (2015)
23. Sen, C.K.: Human wound and its burden: updated 2020 compendium of estimates. Adv. Wound Care **10**(5), 281–292 (2021). https://doi.org/10.1089/wound.2021.0026
24. Wang, C., et al.: A unified framework for automatic wound segmentation and analysis with deep convolutional neural networks. In: 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 2415–2418. IEEE (2015)

25. Wang, C., et al.: Fully automatic wound segmentation with deep convolutional neural networks. Sci. Rep. **10**(1), 21897 (2020). https://www.nature.com/articles/s41598-020-78799-w

ZIRACLE: Zero-Shot Composed Image Retrieval with Advanced Component-Level Emphasis

Florian Erdösi(✉), Kilian Weishaupt, and Khanlian Chung

Vector Informatik, Stuttgart, Germany
florian.erdoesi@vector.com, kilian.weishaupt@vector.com,
khanlian.chung@vector.com

Abstract. Image retrieval systems often struggle to align with a user's search intent, particularly when the query image is composed of multiple semantic concepts. To address this challenge, we propose a novel tempered-weighting query-fusion approach that leverages the capabilities of pre-trained vision-language models (VLMs) to focus on specific image components expressed through textual descriptions. Notably, our method utilizes the BLIP-2 model's embeddings without requiring additional training, enabling zero-shot Composed Image Retrieval (CoIR). For systematic evaluation of our approach and comparison, we introduce the Focus-CoIR dataset, a novel resource derived from the BDD100K dataset that adopts a focusing setting and provides multiple ground truth positive labels for each target image and corresponding textual description. Our experimental results on Focus-CoIR demonstrate the effectiveness of our approach, outperforming state-of-the-art CoIR methods across all evaluated ranking metrics. This work highlights the potential of pre-trained VLMs for CoIR and contributes to the advancement of CoIR research. We make the Focus-CoIR dataset and our implementation publicly available to support future research in this domain.

Keywords: CoIR · Multimodal · ZS-CIR · training free

1 Introduction

The ever faster-growing availability of digital visual data enables us to generate large data sets and thus train and deploy data-hungry applications such as large deep learning models. However, it is hard and tedious work to extract samples with specific characteristics.

Content-based image retrieval (CBIR) is a field of research in the deep learning community focusing on automatically retrieving images semantically similar to a query image. The models used for CBIR quantify the semantic overlap of two images by comparing the images' latent space vectors, for example. One challenge is that a query image often contains more than one semantic concept.

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 133–145, 2025.
https://doi.org/10.1007/978-3-031-80507-3_14

For example, does a user want to find images with a specific feature in the foreground or the background?

Composed Image Retrieval (CoIR) aims to ease this problem by combining images with text. Here, a target image is used in combination with a textual description to steer the retrieval process. Often, the textual descriptions are used to specify modifications: For example, when using an image of a black dress as a target in combination with the text "change color to red", the retrieval should result in images of red dresses.

In this paper, we do not aim to transform the semantics during retrieval. Instead, we investigate how CoIR can be employed to extract the images most faithful and true to a user-intended search concept. We demonstrate how textual descriptions enable an AI to focus on specific features and concepts, like daytime or vehicle appearance, while simultaneously ensuring the retrieved images accurately match the query image. To achieve this we mix the image and text embeddings of the BLIP-2 [11] model. We call our method "Zero-shot composed Image Retrieval with Advanced Component-Level Emphasis" (Ziracle).

Our research contributions are:

- We present a novel zero-shot, training-free method to fuse text and visual embeddings for retrieving images with specific characteristics.
- We present a BDD100k-based data set for CoIR with focus on retrieving images with specific characteristics.
- We demonstrate our method and compare it to state-of-the-art CoIR methods.

2 Related Work

2.1 Vision Language Models

Vision language models (VLMs) are capable of dealing with natural language and visual inputs (such as images or videos). One way to process vision and textual inputs is to align both individual latent spaces. OpenAI demonstrated this with the introduction of contrastive learning in CLIP [17] as well as ALIGN [8]. BLIP [12] and BLIP-2 [11] extended this approach by attention modules capable of taking image encodings from a frozen image encoder and text tokens from a frozen text encoder into account. BLIP-2 deploys a transformer-based module in which the information between the two modalities can be bi-directionally shared to ensure an early data fusion.

These multimodal capabilities make VLMs an ideal starting point for CoIR tasks because they can deal with textual and visual information by design.

2.2 Composed Image Retrieval

CoIR aims to retrieve images from a database using a query image and textual prompt or description. Utilizing two modalities helps to guide the retrieval process. Chen et al. [3] introduced a masked language modeling similar approach in

the CLIP image encoder training. The image encoder output of a masked image is fused with the text encoder output. Contrastive learning is used to minimize the latent space distance between the fused latent vector and the latent vector of the unmasked image which improves the more precise combined textual and visual search. Baldrati et al. [1,2] extended the CLIP approach with a combiner network to fuse text and image encoding to a unified representation. Pic2Word [18] first introduced a mapping network for CLIP-based architectures. The mapping network learned to map the latent space embedding from a frozen vision encoder to a single text token. This token can be inserted into a prompt and then be digested by a (frozen) text encoder. This approach allows users to exploit highly sophisticated, publicly available visual models and text models such as Visual Transformers [7] and BERT [6] without the need for training such encoders themselves.

A mapping network can also build upon a transformer/attention-based architecture, either to generate a text token [19] to be inserted as a textual token embedding or to generate a common `[CLS]` token, containing both text and vision information [10]. Tian et al. [20] and Liu et al. [16] have shown that the textual and vision information can be summarized via aggregation modules which handle the data fusion from frozen image and text encoders.

Alternatives to optimizing the information fusion have also been proposed. Karthik et al. [9] solved the multi-modality problem by using an image captioner. This way no image encoder is needed and mapping from image to text or vice versa is completely circumvented. Instead, the caption is fused with the text query into one single text-based prompt which is passed to an LLM. Liu et al. [16] proposed bi-directional training and modules. In contrast to other mapping approaches, their approach allows for bi-directional mapping and in particular text-to-vision mapping and performs CoIR primarily with image embeddings. Levy et al. [10] identified the ambiguity of competing semantic concepts as one of the main challenges in CoIR. Their approach utilized a chatbot to specify the search. By iteratively refining the textual input of the user, the model tries to find the most fitting semantic concept. To optimize the order of possible candidates for a query, Liu et al. [16] and Chen et al. [4] introduced re-ranking modules in their architectures. The re-ranker is a post-processing step to re-order the candidates to make the first ones more relevant. Zhao et al. [24] introduced a deep learning module to first aggregate the shared semantic multi-modal concepts in a latent space representation. This is then used for CoIR tasks. [22] share the same idea but go even further by extracting common and shared concepts on the feature level via a decomposing network.

2.3 Data Sets

For the evaluation of CoIR, several evaluation methods and data sets have been proposed. The tasks involve processing textual and visual information and in many cases also some sort of semantic transformation. Therefore, popular data sets contain pairs or sets of linked images with corresponding captions. For

instance, the CIRR data set utilizes relative captions to describe the relationship between the images [14]. Another example is the FashionIQ data set [21]. It focuses on a narrow semantic domain and provides not only relative captions between images but also absolute captions for each image in the form of textual descriptions as well as attributes. The SIMAT data set [5] has more emphasis on text-driven image transformations. Therefore, it introduces transformation queries that describe the changed attributes between two images.

3 Method

We propose an approach to utilize a pretrained VLM, namely BLIP-2 [11], for CoIR, without the need for additional training. Specifically, our approach Ziracle allows the user to focus the retrieval on specific components in the target image using textual descriptions. To make use of both the image and text modalities, we employ a novel strategy to combine BLIP-2 embeddings for a target image and text during retrieval, which we call tempered-weighting query-fusion.

BLIP-2 leverages a Q-Former adapter module between a frozen image encoder and a frozen LLM to improve on tasks like image captioning. The Q-Former is a BERT-style [6] transformer encoder that takes input text tokens and 32 learned query embeddings, which interact with the frozen image encoder outputs via cross-attention layers. Through this, they serve as feature extractions to capture important image features. The output token embeddings from the Q-Former are utilized as input tokens for text generation with an LLM, after passing them through a linear projection layer. We employ the feature extraction model variant of BLIP-2, which omits the language model and provides the Q-Former and image encoder, as it showed improved retrieval performance in our preliminary studies.

Our tempered-weighting query-fusion strategy combines the Q-Former image and text embeddings to a single unified representation, which enables us to perform CoIR: Given a target image and describing text, which focuses on the most relevant image components, we retrieve the most relevant candidate images from a given image database.

3.1 Combining Text and Image Embeddings for Focused Retrieval

Our tempered-weighting query-fusion strategy assumes that each of the 32 components in the BLIP-2 image embedding encodes unique characteristics of the input image. This is based on the mechanisms proposed in the original BLIP-2 paper [11], which suggests that the embeddings derived from the learned queries capture a range of visual features. Thus, we propose that focusing on specific image contents can be achieved through suitable weighting of the 32 vector components during the calculation of a similarity score between two image embeddings.

To achieve this, we combine the target image and text embeddings to form a weighting vector, which is used to adjust the similarity score. We use the BLIP-2

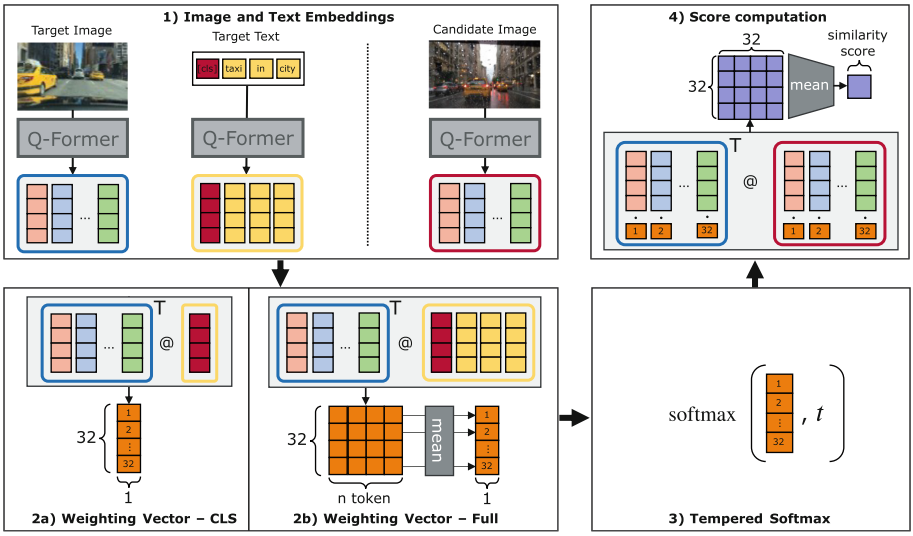


Fig. 1. Overview - Tempered-weighting Query Fusion

Q-Former to extract base embeddings for both the target image and text, as well as the database images used for retrieval (Fig. 1, part 1). The embeddings are generated in separate model calls, allowing us to employ our own combination strategy based on the separate image and text embeddings.

The combination of the image and text modalities is done through matrix multiplication of the target image embedding with either the `[CLS]` token embedding for the "cls" variant (Fig. 1, part 2a), or all text token embeddings of the target text for the "full" variant (Fig. 1, part 2b). In the "full" variant we then employ a rowwise mean function, such that both variants result in a weight vector with 32 entries. Semantically, this corresponds to an individual comparison of each of the image embeddings' 32 vector components to the aggregated text representation in the form of the token embeddings, resulting in a weighting score for each vector component.

To employ the weight vector in the similarity computation, we perform a weighted matrix multiplication of the image embeddings, using the computed weights to reweight the different vector components during the multiplication (Fig. 1, part 4). This focuses the similarity computation on the vector components most important according to the interplay between target text and image. The final similarity score is determined through averaging the values of the score matrix.

In practice, the raw weights are close to equally distributed, which can result in no actual difference between using weights or not. To alleviate this issue, we employ a tempered softmax function to make the differences in the weight vector more pronounced (Fig. 1, part 3). By choosing a small temperature, we can

obtain more differentiated weights, changing the balance during the similarity computation.

This query fusion approach is comparable to the well-known attention mechanism. Though, unlike attention, which typically computes weights based on pairwise similarities between query and key embeddings, our approach derives weights from the combined image and text embeddings, allowing for a more focused weighting of components based on their relevance to the text-image interplay, instead of using attention to form a weighted sum of embeddings.

3.2 Data Set Generation

This section details the generation of a novel labeled dataset, Focus-CoIR, specifically designed for focusing CoIR. We justify the creation of a new dataset due to two limitations of existing CoIR benchmarks: they focus on the transformation setting, where text descriptions modify image components, and they have few ground truth positive labels per reference image. In contrast, Focus-CoIR evaluates methods that utilize **text descriptions to focus** on existing image components and provides **multiple ground truth positive** labels per reference image.

To generate Focus-CoIR, we leverage the publicly available BDD100K dataset [23] and propose a labeling process that minimizes manual labeling efforts. The process involves:

1. Selecting a target image and creating two textual descriptions: a short one for CoIR retrieval and a longer one for manual labeling.
2. Retrieving the top 500 candidate images for each of three CoIR methods (BLIP4CIR [15], TransAgg [13], and our proposed method).
3. Combining and deduplicating the retrieved images to create a list of 500 unique candidates.
4. Manually labeling the candidates as "relevant" or "irrelevant" with respect to the target image and long description.

We identified this assisted manual labeling to be the only feasible approach with the BDD100K dataset, as automated labeling approaches could not be applied successfully due to poor label accuracy. The labeling process results in a dataset containing 102 target images, each with 500 labeled candidate images. We intend to publicly release Focus-CoIR to enable reproducibility and encourage further research in this area.

4 Evaluation

This section details the evaluation procedure and performance of our proposed tempered-weighting query-fusion method for CoIR.

We assess Ziracle's effectiveness on our newly generated Focus-CoIR dataset. We do not evaluate on existing CoIR benchmarks due to the reasons discussed in

Sect. 3.2. For each target image and its textual description within Focus-CoIR, retrieval is performed against the pool of all 500 candidate images with associated ground truth labels corresponding to the target image. The retrieval performance is evaluated using several standard ranking metrics, including Average Precision at 20 (AvgPrec@20), Precision at 20 (Prec@20), and Precision at 50 (Prec@50).

The hyperparameter settings for Ziracle were determined through empirical evaluation. Specifically, a temperature value of 0.001 was selected, and the "full" setting was employed, wherein all token embeddings were utilized during query-fusion. A comprehensive examination of the effects of these hyperparameters is presented in Appendix B, which provides a detailed analysis of our hyperparameter studies.

We establish a competitive benchmark by comparing our method against several well-regarded zero-shot CoIR approaches known for achieving state-of-the-art performance on the CIRR benchmark dataset. These include TransAgg [13], Candidate Set Re-ranking [16], BLIP4CIR [15], and Context-I2W [19]. Additionally, we include a baseline evaluation, using unaltered BLIP-2 image embeddings for retrieval and ignoring the textual description.

As our approach does not include any training on the evaluation dataset, we use released checkpoints for all external methods and refrain from further finetuning the models, to ensure a fair comparison. The aforementioned evaluation protocol using Focus-CoIR is applied consistently to all compared methods, though we use the specific multi-modal retrieval process for each external method, as specified in the sources.

4.1 Results on Focus-CoIR

The results of our evaluation on the Focus-CoIR dataset, for our Ziracle method as well as the external models for comparison, are given in Table 1. We evaluate the methods in two settings, using both the short and the long description defined for each target image in the dataset.

Our method, Ziracle, achieves improvements over the included state-of-the-art methods and the baseline across all metrics in both settings. In the short description setting, Ziracle shows a relative improvement of 7.9% in terms of AvgPrec@20 compared to the second-best performing method, Candidate-Set Re-ranking. This improvement is also observed for Prec@20 with a gain of 4.3%, highlighting Ziracle's effectiveness in retrieving relevant images even with concise textual descriptions. The margin for improvement is slightly smaller for Prec@50 at 2.2%, suggesting that both methods perform well when retrieving a larger set of candidate images. These findings are consistent across the long description setting as well.

Notably, the results obtained with the short description setting are equal to or surpass those obtained with the long description setting across all metrics and methods. This suggests that all methods benefit from more concise textual descriptions. This is likely due to the noisy image descriptions in the long description, as long description not only focus on desired elements but also provide additional context and general image descriptions.

Table 1. Result comparison on the Focus-CoIR dataset.

	Short description			Long description		
	AvgPrec@20	Prec@20	Prec@50	AvgPrec@20	Prec@20	Prec@50
BLIP-2 baseline	55.42	44.90	40.94	55.42	44.90	40.94
BLIP4CIR	51.56	45.34	42.53	51.26	45.49	43.33
Cand.-Set-Rerank	56.62	46.72	44.14	56.65	47.40	43.45
Context-I2W	54.98	46.86	44.22	55.87	45.29	43.27
TransAgg	55.14	45.20	41.08	52.17	43.58	40.55
Ziracle (ours)	**64.51**	**51.08**	**46.41**	**64.34**	**50.64**	**46.31**

In conclusion, the evaluation results support the efficacy of our proposed Ziracle method for composed image retrieval. Ziracle consistently outperforms existing methods across all metrics and settings, demonstrating its effectiveness in utilizing textual descriptions for accurate retrieval of relevant candidate images.

4.2 Qualitative Retrieval Results

In addition to the quantitative evaluation on Focus-CoIR, we present qualitative retrieval results of our proposed Ziracle method for a target in the Focus-CoIR dataset. We compare Ziracle's performance to the image-only BLIP-2 baseline. For both methods, we display the top-5 retrieved images.

Figure 2 demonstrates Ziracle's effectiveness in focusing on the target concept. By incorporating the textual description, all images retrieved by Ziracle contain the prompted green cycle path. In contrast, the BLIP-2 baseline retrieves two images with a red special lane instead. This demonstrates how a user can guide the CoIR to retrieve the most faithful concepts in the candidates.

Fig. 2. Qualitative Top-5 Retrieval Results - Target: "Green cycle path to the right." (Color figure online)

Further examples of qualitative retrieval results are provided in Appendix A for additional insight into the performance of our method.

5 Conclusion

In this research study, our primary objective is to address the challenge of CoIR by enhancing the alignment between the image retrieval process and a user's actual search intent. Specifically, we aim to achieve this alignment by focusing on specific image components, as expressed through textual descriptions.

To this end, we present a novel tempered-weighting query-fusion approach that leverages embeddings generated by a pretrained VLM, specifically BLIP-2, for zero-shot CoIR. Notably, our method achieves this without the need for additional training, using only the pretrained weights released with BLIP-2.

Furthermore, we introduce the Focus-CoIR dataset, which distinguishes itself from existing image retrieval datasets in two key ways. First, it adopts a focusing setting rather than a transformative one with respect to the text query. Second, it provides multiple ground truth positive labels for each target image and textual description. This unique dataset enables the evaluation of CoIR methods that utilize user-provided textual input to focus on existing image components.

Through empirical evaluation using the Focus-CoIR dataset, we demonstrate the feasibility of our query-fusion method to perform image retrieval in line with a user's intent. Comparative analysis against several state-of-the-art CoIR methods reveals a significant improvement across all evaluated ranking metrics when employing our proposed approach.

Looking ahead, we envision several possible directions for further research. Firstly, while our current method is primarily based in the BLIP-2 model, exploring the application of this concept for other vision language models could provide valuable insights into the unique characteristics of their respective embedding spaces. Secondly, incorporating iterative refinement of retrieved concepts based on user feedback could represent a significant step towards better aligning retrieval results with user intent. By allowing users to provide feedback on initial retrieval results, we can refine and improve the retrieval process in a dynamic and interactive way.

To facilitate further research in this domain, we publicly release the constructed Focus-CoIR dataset along with a comprehensive implementation of our tempered-weighting query-fusion method.

A Qualitative Retrieval Results

In this appendix, we provide additional qualitative retrieval results of our proposed Ziracle method for multiple targets in the Focus-CoIR dataset to supplement the main results presented in Sect. 4. We compare Ziracle's performance to the image-only BLIP-2 baseline. For both method, we display the top-5 retrieved images.

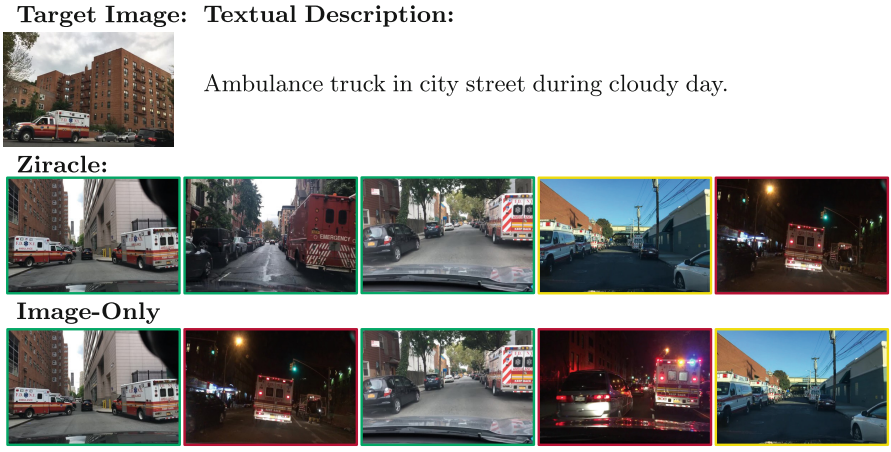


Fig. 3. Target: "Ambulance truck in city street during cloudy day."

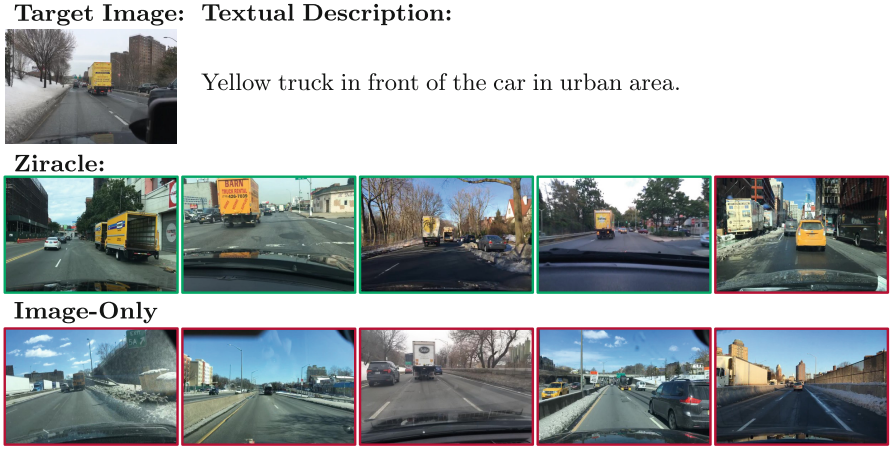


Fig. 4. Target: "Yellow truck in front of the car in urban area." (Color figure online)

Figure 3 shows a similar trend as the example in Fig. 2, but also reveals a limitation of Ziracle. While the retrieval results improve compared to the baseline, the last retrieved image remains a nighttime scene. In the baseline, the same image is retrieved in second position. This suggests that Ziracle does indeed improve on the baseline through emphasizing the prompted "cloudy day" setting, but may still miss some concepts due to the semantic limitations of the vanilla BLIP-2 embeddings employed.

Figure 4 showcases an interesting instance where the image-only BLIP-2 baseline retrieves images that are conceptually similar in terms of general image features (like scene, weather, road type). However, these images miss the central element of the yellow truck. By incorporating the focus through the textual

description, Ziracle successfully retrieves fitting images containing yellow trucks in four out of the five retrieved images.

B Hyperparameter Studies

To optimize our method's performance, we conducted a hyperparameter study focusing on two key aspects: the temperature of the tempered softmax function and the text embedding used for weight calculation. The results of the hyperparameter studies are presented in Table 2 and Table 3.

The temperature of the tempered softmax function controls the "peakedness" of the distribution used to estimate component weights during query-fusion. A higher temperature leads to a more uniform distribution, while a lower temperature concentrates the weight on a smaller number of components. Our study revealed a clear trend of improved retrieval performance with lower temperatures, with the best performance achieved at temperatures of 0.001 and below. This suggests that a more focused weight distribution across components is beneficial for retrieval performance.

The text embedding for weight calculation refers to the specific text representation used to compute the raw component weights. We compared two variants: "cls", which utilizes only the embedding of the `[CLS]` token, and "full", which leverages embeddings from all tokens within the input text sequence. Our study found that the choice of text embedding has a marginal impact on performance, with both variants achieving comparable results. This indicates that the model can effectively leverage information from both the entire sequence and the condensed `[CLS]` token representation.

Our hyperparameter studies demonstrate the effectiveness of the tempered softmax function in improving retrieval performance and provide valuable insights for improving the parametrization of our query-fusion strategy.

Table 2. Hyperparameter study: Softmax temperature

Temperature	AP@20	P@20	P@50
0.00001	63.68	50.65	46.77
0.0001	63.71	50.64	46.69
0.001	63.70	50.67	46.61
0.01	63.20	50.62	45.89
0.1	56.83	45.87	41.38
1.0	55.62	45.21	40.82

Table 3. Hyperparameter study: Text embedding

Text Emb.	AP@20	P@20	P@50
cls	60.64	48.87	44.81
full	61.61	49.02	44.58

References

1. Baldrati, A., Bertini, M., Uricchio, T., del Bimbo, A.: Composed image retrieval using contrastive learning and task-oriented clip-based features (2023)
2. Baldrati, A., Bertini, M., Uricchio, T., Bimbo, A.D.: Effective conditioned and composed image retrieval combining clip-based features, pp. 21434–21442 (2022). https://doi.org/10.1109/CVPR52688.2022.02080
3. Chen, J., Lai, H.: Pretrain like your inference: masked tuning improves zero-shot composed image retrieval (2023)
4. Chen, J., Lai, H.: Ranking-aware uncertainty for text-guided image retrieval (2023)
5. Couairon, G., Cord, M., Douze, M., Schwenk, H.: Embedding arithmetic for text-driven image transformation. arXiv preprint arXiv:2112.03162 (2021)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding (2019)
7. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale (2021)
8. Jia, C., et al.: Scaling up visual and vision-language representation learning with noisy text supervision (2021)
9. Karthik, S., Roth, K., Mancini, M., Akata, Z.: Vision-by-language for training-free compositional image retrieval. arXiv preprint arXiv:2310.09291 (2023)
10. Levy, M., Ben-Ari, R., Darshan, N., Lischinski, D.: Chatting makes perfect: chat-based image retrieval (2023)
11. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models (2023)
12. Li, J., Li, D., Xiong, C., Hoi, S.: Blip: bootstrapping language-image pre-training for unified vision-language understanding and generation (2022)
13. Liu, Y., Yao, J., Zhang, Y., Wang, Y., Xie, W.: Zero-shot composed text-image retrieval. arXiv preprint arXiv:2306.07272 (2023)
14. Liu, Z., Rodriguez-Opazo, C., Teney, D., Gould, S.: Image retrieval on real-life images with pre-trained vision-and-language models (2021)
15. Liu, Z., Sun, W., Hong, Y., Teney, D., Gould, S.: Bi-directional training for composed image retrieval via text prompt learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 5753–5762 (2024)
16. Liu, Z., Sun, W., Teney, D., Gould, S.: Candidate set re-ranking for composed image retrieval with dual multi-modal encoder. arXiv preprint arXiv:2305.16304 (2023)
17. Radford, A., et al.: Learning transferable visual models from natural language supervision (2021)
18. Saito, K., et al.: Pic2word: mapping pictures to words for zero-shot composed image retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 19305–19314 (2023)
19. Tang, Y., et al.: Context-i2w: mapping images to context-dependent words for accurate zero-shot composed image retrieval (2023)
20. Tian, Y., Newsam, S., Boakye, K.: Image search with text feedback by additive attention compositional learning (2022)
21. Wu, H., et al.: The fashion IQ dataset: retrieving images by combining side information and relative natural language feedback. In: CVPR (2021)
22. Yang, X., Liu, D., Zhang, H., Luo, Y., Wang, C., Zhang, J.: Decompose semantic shifts for composed image retrieval (2023)

23. Yu, F., et al.: Bdd100k: a diverse driving dataset for heterogeneous multitask learning (2020)
24. Zhao, S., Xu, H.: Neucore: neural concept reasoning for composed image retrieval (2023)

Improving Object Detection Performance on Low-Quality Images Using Histogram Matching and Model Stacking

Yohanes Nuwara[1(✉)] and Quoc-Huy Trinh[2]

[1] Politecnico di Milano Graduate School of Management, Bovisa, Milan, Italy
yohanes.nuwara@gsom.polimi.it
[2] SpexAI GmBH, Dresden, Germany
huy@spex.ai

Abstract. Imaging is very important in many industries, however, it often suffers from low light and colour shift issues. For tasks that require object detection, quality issue can reduce model accuracy. This work aimed to enhance the performance of few object detectors such as Mask R-CNN, YOLOv5, and YOLOv8 on low-quality images using a combined histogram matching and model stacking techniques. We used 2 datasets which exhibit image quality issues, namely Roboflow100 underwater object and oil palm fruit (*Elaeis guineensis*) images collected from a plantation in Indonesia. The images were processed through two distinct gates where two models were trained on original images (M_1) and on histogram matched images (M_2). The output from the two gates were stacked and applied Non-Maximum Suppression (NMS) to reduce the overlapping. The predicted class from the second model shows good agreement with ground truth due to colour correction by histogram matching (0.651 $F1$-score). Hence, we modified the NMS to assign the class from M_2 and presented final result. The result demonstrated a notable improvement in accuracy up to $F1$-score of 0.811, confirming the effectiveness of the proposed method in enhancing object detection performance on low-quality images.

Keywords: Histogram matching · Image quality · Model stacking · Non-maximum suppression · Object detection

1 Introduction

In the development of the computer vision, object detection is one of the notable applications of deep learning due to its real-world applications. In the early stage, several detection methods such as Fast-RCNN, Faster-RCNN, SSD512, Mask R-CNN were proposed to solve this challenge. Following this, several YOLO family models from YOLOv1 up to YOLOv10, and DETR were proposed to deal with the limitations of the small and multi objects from previous works. Although the previous works achieved impressive results, they were still having

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 146–155, 2025.
https://doi.org/10.1007/978-3-031-80507-3_15

some constraints when applied to low-quality images, which could make the models failed in several real-world applications that relied on imaging system.

This literature review explores the effects of color variability on deep learning models for computer vision. In one study, 27 different RGB distortions were applied to ImageNet images and it observed that yellow color distortions had the most negative impact, reducing ResNet-152's top-1 accuracy from 88.92% to 56.47% [1]. Similarly, another study focused on improving color constancy networks for object recognition under varying illumination. By isolating objects against a black background, their approach enhanced model performance in difficult lighting conditions [2].

Several studies addressed the challenges of object detection in complex environments. A super pixel-based Bag-of-Words model was proposed to improve object segmentation under variable lighting, combining it with Conditional Random Fields for precise detection [3]. Likewise, an interesting study tackled poor object detection under low-light industrial settings by combining adaptive retinex and shadow removal techniques, demonstrating improvements in recognizing objects affected by shadows and poor lighting [4].

In addition to color variability, many studies emphasized the role of histogram equalization in enhancing image quality and detection accuracy. The contrast limited adapted histogram equalization (CLAHE) was applied to improve image analysis in medical fields like CT and MRI [5]. Another study introduced a method that preserves entropy while enhancing image details, outperforming traditional equalization techniques [6]. Moreover, more studies [7,8], and [9] combined histogram equalization with other methods like color balancing and filtering, demonstrating improvements in object detection and visual clarity in medical, industrial, and satellite images [10]. These advancements underscore the importance of managing lighting and contrast for improving the robustness of computer vision models across diverse applications.

Motivated by these studies, we proposed an approach that employs the histogram matching and model stacking to deal with these problems, specifically applied in agriculture. In summary, we contributed the following in our proposed approach. Firstly, we proved the effectiveness of histogram matching technique to improve the quality of image when illumination issue occurred. Secondly, we introduced our approach using model stacking in order to improve the performance of any object detection model.

2 Methods

In this study, we developed our proposed workflow on a benchmark dataset called Roboflow100 Underwater Objects, released by Roboflow [11]. This dataset consisted of 7,600 labeled data with 5 classes of underwater fauna. This data suffered from low lighting issue that appeared turbid greenish. Under this condition, object detection system was expected to give low accuracy results, which was a perfect use case for our method.

Then, we applied and tested this technique on a real use case in agricultural industry. We had collected 4,000 photographs of oil palm (*Elaeis guineensis*)

fruits harvested from palm trees. The fruits were grouped into 6 classes according to classification by MJMPOM[1] namely abnormal bunch, unripe bunch, underripe bunch, overripe bunch, empty bunch, and damaged bunch. The fruit bunch had been annotated manually according to the classification. Out of the 4,000 photographs collected, 30% have image issues, such as yellow cast, overexposure, and occlusion (dark shadow).

To solve the problem, we proposed a workflow shown in Fig. 1. It consists of histogram matching (HM), two-gate model training, model stacking, and an adjusted non-maximum suppression (NMS) to improve the detection accuracy on image with quality issues.

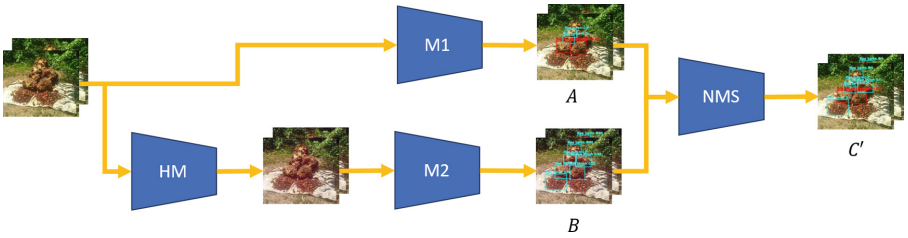


Fig. 1. The pipeline we proposed for enhanced object detection on low-quality images. Note: HM is histogram matching, M_1 is first object detection model, M_2 is second object detection model, and NMS is stacking and non-maximum suppression

2.1 Histogram Matching

In histogram matching (HM), we selected one image as a reference (or destination) image, denoted as d. This reference image was a normal image with a good lighting. The intensity histogram was extracted from d. The image with quality issue was considered as source image, denoted as s. The cumulative distribution function (CDF) based on the histogram was calculated from both source image s and destination image d, then the CDF of s was mapped to d. The transformed image is denoted as s'.

Figure 2 illustrates the application of histogram matching to three source images with distinct lighting issues: yellow cast, overexposure, and occlusion (dark shadows). Each row presented an original image (first column), its histogram (second column), the transformed image after histogram matching (third column), and the histogram of the transformed image (fourth column). The histograms were calculated by unraveling all RGB pixel values into a single intensity distribution.

The image with a yellow cast had a skewed intensity histogram concentrated in the mid to high range. The overexposed image initially showed a peak in high-intensity values, indicating excessive brightness. The image with occlussions had

[1] MJM (Palm Oil Mill) Sdn Bhd published their classification in https://www.mjmpom.com/ffb-grading-guideline.

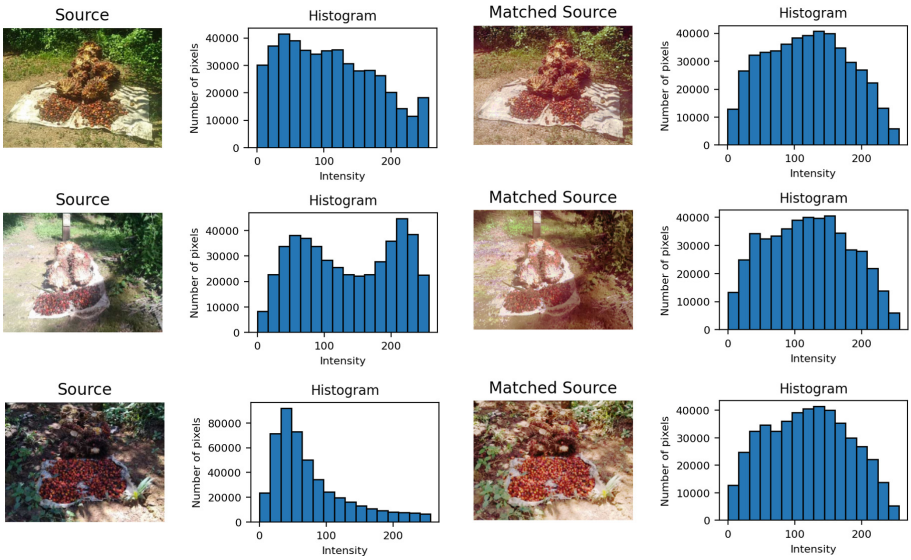


Fig. 2. Original and transformed images after HM. Column 1: original image, column 2: histogram of the original image, column 3: transformed images after HM, column 4: histogram of the transformed images. Images are divided into 3 issues. Row 1: image with yellow cast issue, row 2: image with overexposure issue, and row 3: image with occlusion issue. (Color figure online)

a histogram dominated by low-intensity values. HM brightens the image and balances the intensity distribution, enhancing visibility in low quality areas of the image.

2.2 Model Training

In this stage, we had 2 different sets of images, namely original images and HM-transformed images. Object detection model was then trained on each of the sets to classify different object classes. We compared 3 models, namely Mask R-CNN, YOLOv5, and YOLOv8.

Mask R-CNN is built on Faster R-CNN by adding a segmentation branch, enabling it to detect objects and generate pixel-level masks for instance segmentation. It leverages a region proposal network (RPN) to predict object locations and corresponding masks.

YOLOv5 and YOLOv8 follow the "You Only Look Once" (YOLO) architecture, where object detection is performed in a single pass through the network, facilitating real-time detection. YOLOv5 introduces anchor boxes and mosaic augmentation to refine detection accuracy, while YOLOv8 further optimizes performance by integrating dynamic anchor boxes and transformer layers, improving both speed and accuracy.

In the first gate, we trained first model (M_1) on original sets, and in the separate gate, we trained second model (M_2) on HM-transformed sets. Object detection model resulted an $M \times 6$ matrix where M represents the detected objects, and 6 variables namely object class (c), bounding box center x-coordinate (x), center y-coordinate (y), bounding box width (w), bounding box height (h), and confidence score (p). Matrices from both model should have different bounding box detected locations and classes, therefore we stacked both results.

2.3 Model Stacking and Adjusted Non-Maximum Suppression

Both models M_1 and M_2 had different detection result. Most of the objects might be missed in M_1 but detected in M_2 thanks to the colour correction. However, it might happen otherwise although it was not a dominant case. Hence, the results from both models were stacked together. To reduce the overlapping bounding boxes, we used the non-maximum suppression (NMS) algorithm.

However, it was interesting to observe from the result that due to colour correction, M_2 detected the class more accurately. Hence, we adjusted the NMS algorithm by taking the class from M_2. This was done by reducing confidence scores of M_1 result, for example 5×10^{-2} so that it would always take the bounding boxes with confidence scores from M_2.

Algorithm 1. Model Stacking and adjusted Non-maximum Suppression

Input: Bounding boxes from model M_1 (matrix A) and model M_2 (matrix B)
Output: Processed bounding boxes after NMS
Let $\mathbf{A} = (x_i, y_i, w_i, h_i, p_i^{(1)})_{i=1}^{n_1}$
Let $\mathbf{B} = (x_j, y_j, w_j, h_j, p_j^{(2)})_{j=1}^{n_2}$
Set $p_j^{(2)} = 0.05$ for all $j \in 1, 2, \ldots, n_2$
Stack to form matrix $\mathbf{C} = \mathbf{A} \cup \mathbf{B} = (x_k, y_k, w_k, h_k, s_k)_{k=1}^{n_1+n_2}$
Apply Non-maximum Suppression (NMS) on $\mathbf{C}$:
for each bounding box $(x_i, y_i, w_i, h_i, p_i)$ in $\mathbf{C}$ **do**
    for each bounding box $(x_j, y_j, w_j, h_j, p_j)$ in $\mathbf{C}$ **do**
        if IoU$((x_i, y_i, w_i, h_i), (x_j, y_j, w_j, h_j)) >$ threshold **then**
            if $p_i > p_j$ **then**
                Select bounding box $(x_i, y_i, w_i, h_i, p_i)$
                Discard bounding box $(x_j, y_j, w_j, h_j, p_j)$
            else
                Select bounding box $(x_j, y_j, w_j, h_j, p_j)$
                Discard bounding box $(x_i, y_i, w_i, h_i, p_i)$
            end if
        end if
    end for
end for
Return Final bounding boxes

3 Experimental Results

In the following discussion, we will show how our approach improved object detection result. We initiated the analysis on the RF100 Underwater Objects dataset. Figure 3 shows the comparison of results. We could barely see objects shown in the 4 figures on the left, due to very poor lighting making the photos look greenish under the sea. The object detection had missed few objects. After we implemented our approach consisting of histogram matching and model stacking, the object detection result improved. Significant colour correction of the figures on the right impacted in detection improvement.

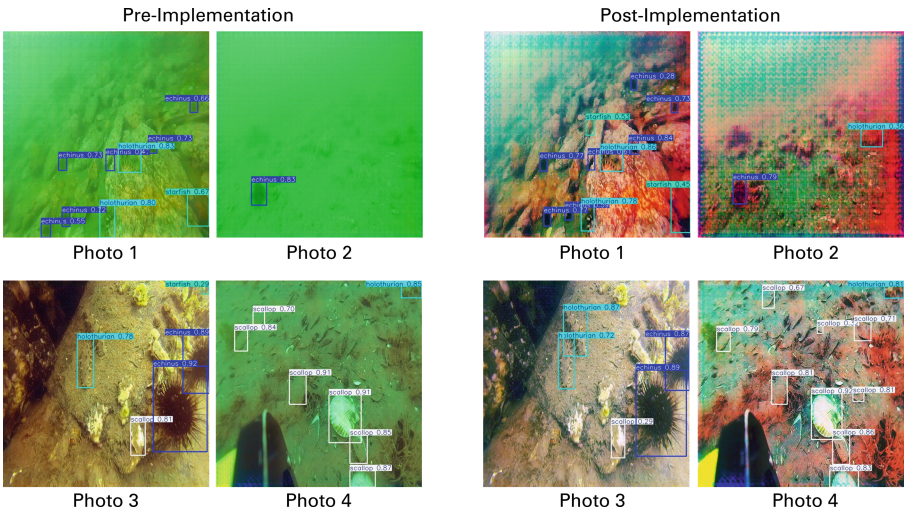


Fig. 3. Comparison of object detection result on low-quality images of RF100 underwater objects dataset (Left) and result on enhanced quality image using our approach (Right)

Next, we analyzed our implementation result on palm oil dataset. We divided our analysis into 3 discussions each for different conditions, namely yellow-cast, overexposed, and occlusion issue. Through out this article, we coined variables such as M_1 as model trained on original images and M_2 as model trained on transformed images after histogram matching (HM).

3.1 Image with Yellow-Casted Issue

Our approach effectively enhanced the detection of fruit bunches in images exhibiting a yellow color cast as shown in Fig. 4. Model M_1 successfully identified ripe bunches (cyan bounding boxes) and even detected 2 overripe bunches (red bounding boxes). However, it exhibited limitations in detecting some fruit

bunches, particularly those with a darker appearance due to significant color distortion (last row). This limitation was addressed by model M_2, which detected the missed bunches in the transformed image (where color fidelity was restored) but introduced false positives by classifying overripe bunches as ripe. By stacking both models outputs, we achieved a final detection of 5 ripe bunches and 2 overripe bunches.

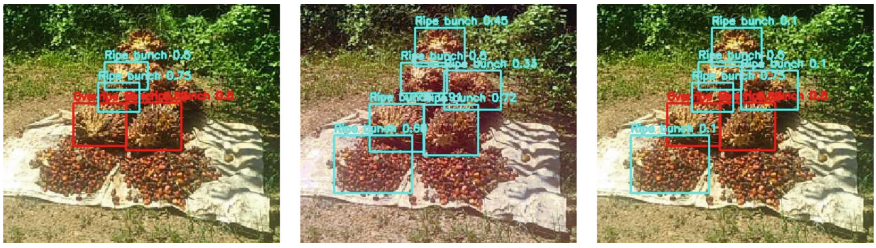


Fig. 4. Object detection result on yellow cast images using (a) the first model on untransformed (original) images, (b) the second model on transformed images with histogram matching, and (c) the final model after stacking the first and the second model. Notice the color difference in (b). (Color figure online)

3.2 Image with Overexposure

Our proposed method also effectively mitigated the impact of overexposure on fruit bunch detection as shown in Fig. 5. Overexposed images exhibit increased exposure, leading to color distortion in the fruit bunches. Model M_1 failed to detect 1 ripe bunch due to this color shift. The excessive brightness reduced the model's ability to accurately recognize the fruit bunches. While model M_1 successfully detected 4 ripe bunches (indicated by cyan bounding boxes), HM was employed to normalize the color. This correction enabled model M_2 to detect 1 additional ripe bunch that was missed by model M_1. Notably, model M_2 achieved a detection of 5 ripe bunches. By stacking both model outputs, final detection of 5 ripe bunches was obtained.

3.3 Image with Occlusion

Our proposed algorithm demonstrated efficacy in improving the performance of object detection in images with occlusion shown in Fig. 6. Insufficient sunlight and the presence of surrounding objects such as palm trees could lead to shadows. Consequently, model M_1 exhibited underdetection, only identifying 2 ripe bunches. To address this limitation, histogram matching (HM) was applied. HM effectively enhanced image contrast. Notably, model M_2 achieved successful detection of all bunches following HM application. It identified 6 ripe bunches and 1 damaged bunch. This demonstrated significant impact of HM in mitigating the shadow. By stacking both model outputs, the final result yielded accurate detection of 6 ripe bunches and 1 damaged bunch.

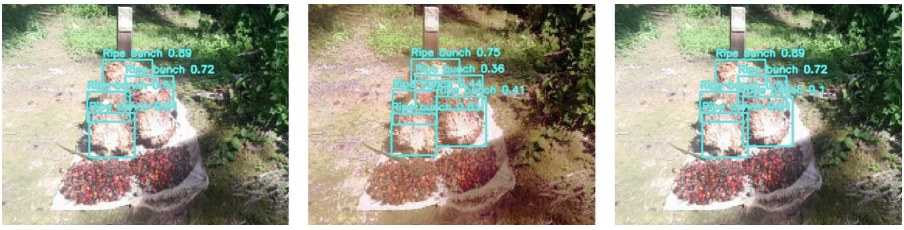


Fig. 5. Object detection result on overexposed images using (a) the first model on untransformed (original) images, (b) the second model on transformed images with histogram matching, and (c) the final model after stacking the first and the second model. Notice the color difference in (b).

Fig. 6. Object detection result on too-dark (shadow) images using (a) the first model on untransformed (original) images, (b) the second model on transformed images with histogram matching, and (c) the final model after stacking the first and the second model. Notice the color difference in (b).

Overall, we evaluated the models both on RF100 (our benchmark) and our palm oil dataset to analyze the improvement made by our approach. Table 1 below presented the comparison of results of model M_1, M_2, and stacked models. Regardless of the object detection models (Mask R-CNN, YOLOv5, and YOLOv8) used in this experiment, there was always accuracy improvement using the stacked model approach both on RF100 and our dataset. Implemented on RF100, the $F1$-score accuracies of stacked models using Mask R-CNN, YOLOv5, and YOLOv8 were 0.779, 0.821, and 0.847. Implemented on our dataset, the $F1$-score accuracies of stacked models using Mask R-CNN, YOLOv5, and YOLOv8 were 0.803, 0.793, and 0.811. YOLOv8 has the best performance compared to Mask-RCNN and YOLOv5.

Hence, we focused our analysis on YOLOv8. On our dataset, the stacked model had significant improvement compared to model M_1 and M_2. The final model with our approach achieved $F1$-score accuracy of 0.811 compared to the model trained on original images which achieved 0.571 and model trained on HM-transformed images which achieved 0.651. This experiment proved the effectiveness of our approach using the combination of histogram matching and model stacking in improving object detection result on low-quality images.

Table 1. Performance comparison of different architectures on two datasets.

Dataset	Architecture	Training	Average $F1$-score	mAP@50-90
R$F100$	Mask R-CNN	M_1	0.671	0.398
		M_2	0.700	0.410
		Stacked model	0.779	0.623
	YOLOv5	M_1	0.711	0.497
		M_2	0.726	0.611
		Stacked model	0.821	0.730
	YOLOv8	M_1	0.701	0.501
		M_2	0.739	0.615
		Stacked model	0.847	0.733
Ours	Mask R-CNN	M_1	0.451	0.460
		M_2	0.623	0.544
		Stacked model	0.803	0.688
	YOLOv5	M_1	0.524	0.447
		M_2	0.623	0.544
		Stacked model	0.793	0.701
	YOLOv8	M_1	0.571	0.503
		M_2	0.651	0.612
		Stacked model	0.811	0.798

4 Conclusion

The experimental results showed that histogram matching significantly improved illumination in images with lighting issues such as yellow cast, overexposure, and occlusions. Histogram matching corrected illumination issues, leading to more balanced images and better visibility of objects. We trained 2 models, each using different architectures namely Mask R-CNN, YOLOv5, and YOLOv8, on the original images (called as M_1) and HM-transformed images (called as M_2). Model stacking and adjusted NMS was applied to combine results from both M_1 and M_2. As a result, our evaluation showed that M_1 had an $F1$-score of 0.571 and mAP of 0.503, M_2 improved to 0.651 $F1$-score and 0.612 mAP, while the stacked model achieved the highest scores with 0.811 $F1$-score and 0.798 mAP using YOLOv8 model on our palm oil dataset. This improvement was also evident on RF100 Underwater Objects dataset with final $F1$-score of 0.847 after stacking. In summary, our approach was proven effective in improving object detection model performance on images with low-quality issues.

Another key information that was not implemented in our oil palm classification is the level of loose fruit. This is the standard of many oil palm industries that we need to apply in our future work. We saw many possibilities of implementing our proposed workflow in other industrial use cases, such as manufacturing, petroleum, food and beverages, and so on. As imaging system becomes

more advanced nowadays, it is useful to consider the use of special cameras such as Infrared or RGB-Depth to improve detection task. Therefore, future studies could be implementing this workflow for different image input or leveraging use of other colour space such as HSV to investigate any improvement on model performance that could occur.

References

1. Hu, C., Shi, W.: Impact of Color Variation on Robustness of Deep Neural Networks. arXiv (2023)
2. Sugiyama, S., Manabe, Y., Yata, N.: Improving the accuracy of the color constancy network by the object detection. In: International Workshop on Advanced Imaging Technology on SPIE Conference Proceedings, pp. 1–2. SPIE, Hong Kong (2022)
3. Shu, G., Dehghan, A., Shah, M.: Improving an Object Detector and Extracting Regions using Superpixels. CVPR (2013)
4. Yunanto, A., Kartowisastro, I.H.: Diminishing variant illumination factor in object recognition. In: Nguyen, N.T., Tojo, S., Nguyen, L.M., Trawiński, B. (eds.) ACIIDS 2017. LNCS (LNAI), vol. 10192, pp. 561–571. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54430-4_54
5. Manjunath, A., Neelapa, Prakash, Yatnalli, V., Bhusare, S.S.: Performance analysis of graph theory-based contrast limited adaptive histogram equalization for image enhancement. WSEAS Trans. Syst. (2023)
6. Bataineh, B.: Image contrast enhancement for preserving entropy and image visual features. Int. J. Adv. Intell. Inform. (IJAIN) **9**(9), 1–2 (2023)
7. Dawar, J., Raheja, P., Vashisth, U., Cheng, I., Basu, A.: Color balanced histogram equalization for image enhancement. In: IEEE International Conference on Multimedia & Expo Workshops (ICMEW) on IEEE Xplore, pp. 1–2. IEEE, Taipei (2020)
8. Al-Azzawi, Z.H.N., Mahdi, W.H., Ahmed, S.K., Yasin, W.S.: Medical image enhancement using histogram equalization techniques. In: Al-Khadum 2nd International Conference on Modern Applications of Information and Communication Technology on AIP Conference Proceedings, pp. 1–2. AIP, Baghdad (2023)
9. Singh, S., Godara, A.: Gaurav: detection of partial invisible objects in images using histogram equalization. Int. J. Comput. Appl. **85**(9), 40–44 (2014)
10. Zhou, X., Geng, Q., Hu, Y., Hao, J.: The application of improved histogram equalization and image segmentation in satellite image. DEStech Trans. Comput. Sci. Eng. (2013)
11. Ciaglia, F., Zuppichini, F.S., Guerrie, P., McQuade, M., Solawetz, J.: Roboflow 100: A Rich, Multi-Domain Object Detection Benchmark. arXiv (2022)

Comparing Learning Methods to Enhance Decision-Making in Simulated Curling

Michael Brunner(✉) and Kaspar Riesen

Institute of Computer Science, University of Bern, Bern, Switzerland
{michael.brunner,kaspar.riesen}@unibe.ch

Abstract. This paper researches strategic decision-making in simulated curling using advanced computational methods. A simulated solution could revolutionize tactical discussions in curling and enhance the strategy of professional curlers. By focusing on position analysis and the integration of supervised learning algorithms, the study aims to enhance the accuracy and effectiveness of strategic calls in curling simulations. The research reveals that supervised neural networks and graph neural networks outperform heuristic approaches and exhibit high accuracy in complex game scenarios. However, the study also shows that current methods and datasets seem to be insufficient within the realm of machine learning and pattern recognition to enable an artificial system to consistently outperform professional curlers (particularly in complex mid-end positions). The findings also indicate that a hybrid approach, combining methods from statistical and structural pattern recognition, might better address the challenges of the continuous space of curling situations.

Keywords: Curling Simulation · Neural Networks · Graph Neural Networks

1 Introduction

In recent years, advances in the field of artificial intelligence in general and pattern recognition in particular have created new opportunities to improve strategic decision-making in sports, e.g. in rugby [1] or other team sports [2]. Curling, with its intricate blend of strategy and precision, presents a unique challenge and a fertile ground for applying these technologies. The present paper researches the potential of advanced computational methods to improve strategic decision-making in curling, focusing on position analysis and supervised learning. Significant contributions in related work [3–5] have explored similar approaches, yet there is still no framework available that can be used in professional curling to analyze and optimize decision-making processes.

To develop a framework capable of making strategic decisions in curling, several challenges must be addressed. First, curling operates in a continuous space where the position and trajectory of each stone can vary infinitely, making it difficult for a computational framework to predict and evaluate outcomes

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 156–165, 2025.
https://doi.org/10.1007/978-3-031-80507-3_16

precisely. Second, categorizing possible calls is inherently complex due to the vast number of potential strategies and shots available in any given situation. This complexity is compounded by the difficulty a computer faces in understanding the nuanced impact of specific shots on the overall game strategy.

The primary goal of this paper is to establish a novel baseline for further research into the application of machine learning and pattern recognition algorithms in curling strategy and performance. The main focus of this work is position analysis of curling stones, specifically the task of determining possible calls from a given stone position and ultimately identifying the best strategic call using supervised learning methods (such as neural network models [6] or graph neural networks [7]). These methods are eventually trained to evaluate game positions and suggest calls based on patterns learned from expert opinions and simulated gameplay.

The remainder of this paper is organized as follows. Next, in Sect. 2, the curling simulation environment and the generic curling bot are briefly reviewed. Next, in Sect. 3, we introduce three different bots to generate candidate calls in curling simulations. In Sect. 4, we thoroughly evaluate these three bots in various scenarios and in Sect. 5, we conclude the paper and make suggestions for future research.

2 Curling Simulation Environment and Curling Bot

This section briefly describes the framework used to implement the curling simulation environment and the generic curling bot developed for this paper.

2.1 Curling Simulation Environment

Our general framework includes core components such as the game logic of curling, a physics engine, and an encoding for player accuracy, which collectively create a robust and dynamic environment for testing and developing strategic decision-making in curling. The game logic models the rules and mechanics of curling, managing turn sequences, strategy implementation, and scoring to emulate a full curling match. This ensures that all actions adhere to the established rules, maintaining the integrity and realism of the simulation.

For this work, we develop a physics engine which is designed to realistically simulate the physical interactions and movements of curling stones on the ice. It updates stones' positions iteratively, simulates curling motion, and accounts for velocity reduction and collisions. This level of detail is essential for creating a realistic and immersive simulation, providing a solid foundation for testing strategic decisions. The physics constants are refined through expert evaluation to mirror real-world curling dynamics accurately, and shots are rendered at accelerated speeds to approximate real-time action while maintaining efficient algorithmic performance.

To replicate human performance variability, Gaussian noise is introduced to each shot in the simulation, modeling the inherent uncertainties observed in

actual curling games. This noise affects the weight and trajectory of the shots, reflecting different accuracy levels and the difficulty in controlling various types of throws. By adjusting each shot's intended line and weight based on these noise models, the simulation ensures that even skilled virtual players exhibit realistic performance variations. This approach enhances the authenticity of the simulation, creating a challenging environment for strategic analysis and training, and testing the robustness of the bots' strategies.

2.2 Generic Curling Bot

For the present research, we initially develop a generic bot that can readily be employed for making strategic decisions in curling simulations. The bot analyzes the current game position to determine the best possible calls, calculates the required curl for specific shots, and checks for free paths. The core mechanism involves ranking candidate calls based on factors like score, end, hammer, shot, and stone locations, ensuring a comprehensive evaluation of potential strategies.

In each iteration, the bot has several shot options, including draw, hit and stay, freeze, tap, hit and roll, double take-out, and run back. Each shot requires precise calculations for the line, weight, and handle. This flexibility allows the bot to adapt to various strategic scenarios during gameplay. Ensuring the path is unobstructed is crucial, and involves a two-step process, viz. trajectory computation and obstacle detection. The bot calculates the trajectory based on the shot's initial settings and checks for any obstructions along the path to ensure the shot can be executed as planned.

This generic setup allows us to build various bots that differ how they determine the best possible call given a certain situation. The different strategies actually implemented in the present paper are reviewed next.

3 Generate Candidate Calls

We propose three different bots to generate candidate calls in curling simulations. These bots employ a range of approaches, from heuristic methods with predefined rules to advanced neural network strategies. Each bot assesses the current position on the ice, considering factors like the score and hammer advantage, to propose strategic moves. Only the top calls are considered, excluding impossible calls. The first bot uses rigid heuristic-based approaches for quick decision-making, while the other two bots utilize a list of 305 predefined calls to adapt to more complex scenarios effectively.

3.1 The Heuristic Bot

The *Heuristic Bot* employs a set of predefined rules based on the current score and game stage. It adapts its strategy during the end, using different sets of calls for opening moves, mid-end plays, and the last stone to maximize scoring opportunities.

One key distinction in finding an adequate rule is based on whether the bot is leading or trailing in the game. When leading, the bot tends to prefer take-out shots to simplify the end and maintain its lead by reducing the number of opponent stones in play. Conversely, if trailing, the bot adopts more aggressive strategies, utilizing calls that increase complexity and potential scoring opportunities to change the game's momentum. Another notable difference lies in the bot's use of the hammer. With it, the bot often has more options to play on the sides of the rink, leveraging the last-stone advantage to secure scoring positions away from the center. Without the hammer, the strategy focuses more on central play, aiming to control the key scoring area and limit the opponent's opportunities.

3.2 The Neural Network Bot

The *Neural Network Bot* improves upon the heuristic approach by using extensive simulated gameplay to train neural network models for each shot within an end. The network architecture includes two linear layers and a comprehensive input feature vector capturing various aspects of the game position.

The input vector for the neural network is meticulously designed to capture a wide array of features from the game's position, amounting to 3,888 distinct inputs. These inputs provide a rich, detailed view of the game's state, facilitating sophisticated analysis and decision-making. Key features encoded within the input vector include, for instance, the *score* and the *score range*, i.e., a detailed listing of each team's stones in the house, organized from the closest to the furthest from the pin, accompanied by additional information about the positional structure or path Information (i.e., an indication of whether paths are blocked or playable, providing strategic insights into possible calls).

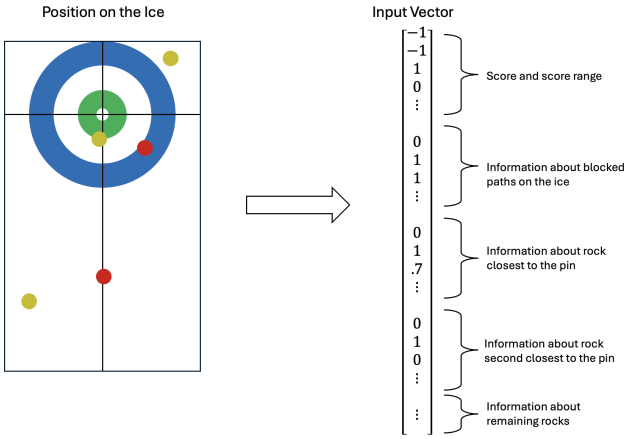


Fig. 1. Conversion of a position on the ice to its corresponding feature vector.

Figure 1 visually illustrates how positions on the ice are transformed into this comprehensive feature vector. Each stone, in play or not, is represented by 142 neurons. For stones not currently in play, all corresponding neurons are set to zero to preserve the accuracy of the data representation. The arrangement of these stones in the vector is determined by their proximity to the pin.

During the data collection phase, over 12,000 unique game positions are simulated to encompass a broad spectrum of potential game scenarios. Each of these positions is manually and rigorously evaluated by seasoned curlers who have identified the best call based on their knowledge and strategic insights. This expert-guided methodology ensures that the training data reflects advanced strategic thinking and adhere to the competitive standards of curling.

To ensure adaptability by both teams, the dataset focuses exclusively on the team with the red stones. To simulate a play for the team with the yellow stones, the bot is programmed to invert the colors of all stones on the ice and recalculate the feature vector accordingly. This color inversion allows for training a single model that is capable of making optimal decisions regardless of the team color.

Furthermore, to augment the training data without extending the data collection period, each game position is mirrored along the centerline and paired with its corresponding strategic call. This data augmentation technique effectively doubles the dataset size, providing a more varied set of scenarios for model training.

The training process for this neural network involves fine-tuning the model to align closely with expert-labeled outputs, effectively learning to replicate expert strategic decisions. The NN Bot is equipped with 16 individual models, each specifically fine-tuned for distinct shots within the game. This targeted approach ensures that the bot's strategies are both precise and tailored to the various stages of the game, in the same way a human would strategize.

3.3 Graph Neural Network Bot

The *Graph Neural Network (GNN) Bot* builds on the supervised data but employs a graph-based model architecture to better analyze relational dynamics between stones. That is, we use the same extensively annotated dataset as described in Sect. 3.2. Moreover, similar to the NN Bot, the GNN Bot is equipped with 16 distinct models, with each dedicated to optimizing decision-making for a specific shot in the end. This differentiation ensures that strategies are finely tuned to the unique tactical requirements of each shot, from the opening to the final positions of an end.

The key difference of the GNN bot to the NN bot lies in the representation of the position: here, the position is converted into a graph, and a graph neural network is eventually trained to accurately respond with the optimal call. More precisely, the actual stones are represented as nodes in a graph. Key attributes for each stone include, for instance, the location (i.e., the coordinates of the stone on the ice), the guarded status (two numerical values ranging from 0 to 1, reflecting how well the stone is guarded), and various others.

Weighted edges are established between nodes to encode the relational dynamics among stones. The weight of an edge is determined by the strength of the relationship between stones. That is, heavier weights are assigned to edges where stones exhibit a stronger connection, such as being in similar lines which may indicate a guarding relationship or when stones are in close proximity to each other. This graph structure allows the bot to analyze the position and interpret complex stone relationships. Figure 2 provides an example of the conversion from the current position into a graph.

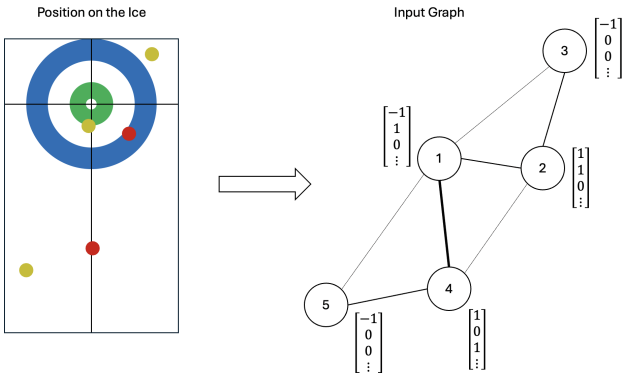


Fig. 2. Conversion of an example position to its input graph.

The GNN Bot utilizes a specialized architecture designed to process graph-structured data (potentially enabling a sophisticated analysis of the relationships between the stones). The core of the model consists of two graph convolution layers that transform the input features (attributes of stones and their positions) into higher-level abstract representations. These layers leverage the message passing paradigm to facilitate the flow of information between nodes, thereby capturing the intricate relationships among the stones. After processing through the graph convolution layers, node features are aggregated using global mean pooling, which condenses the information from all nodes into a single vector. This vector is then processed through a linear layer to project the features into an output space suitable for classification. The final output is processed by a softmax function, which converts the linear layer outputs into a probability distribution over possible strategic calls, indicating the most advantageous calls given the current game state.

4 Empirical Evaluation

4.1 Evaluation of the Learning Bots

The first evaluation involves testing and comparing both the NN bot and the GNN bot. Each bot's decision-making skills are tested across 100 positions

selected by top curlers, focusing on different stages of an end (i.e., we differentiate the positions depending on which stone it is, e.g. the fifth stone or the last shot). These positions are chosen to ensure high-level strategic thinking, with the "correct" and "acceptable" calls defined by experts, thereby providing a robust benchmark for assessing the bots' performance.

The performance of each bot in correctly identifying the optimal or acceptable calls is quantified and presented in Table 1. The table shows the percentage of times each bot makes a call that matches the curlers' chosen strategies under four test conditions.

Table 1. Percentage of correct and acceptable calls made by each bot.

Test Condition	NN		GNN	
	Correct	Acceptable	Correct	Acceptable
Fifth stone	41%	80%	32%	79%
Ninth stone	33%	69%	35%	73%
Second Last Shot	82%	90%	83%	88%
Last Shot	93%	96%	89%	96%

Both the NN and the GNN bot show similar performance compared to each other during all stages of the end. The accuracy generally increases at later stages as the strategy in the later stages is usually more straightforward, focusing mainly on either scoring more points or limiting the opponent's ability to score points. In contrast, the performance tends to decline during the middle stages of the end. This is attributed to the more complex strategies involved, making it more challenging for the bots to make optimal decisions. The complexity and variability of potential outcomes in these stages require more sophisticated analysis, which the current systems might struggle to handle effectively. The accuracy for earlier shots often shows a significant gap between choosing the optimal call and an acceptable call. This is primarily due to the fewer stones involved, making it easier to identify an acceptable call.

When examining individual positions, it is clear that bots have more difficulty recognizing good calls in more complex scenarios. This is because there are more possible options to consider, and some calls are not detectable by the bot due to the limited number of predefined calls. Overall, the results underscore the complexity of strategic decision-making in curling, where later stages of the game allow for more precise and effective calls.

Table 2 illustrates the accuracy of each bot during model training on the chosen test set, specifically analyzing the accuracy for only the last stone and second-to-last stone in various configurations. Again, the results indicate similar performance levels for both NN and GNN, with NN slightly outperforming GNN in simpler scenarios. However, as the complexity of the positions increases with more stones, the performance of the NN bot declines more sharply compared to

the performance of the GNN bot. This trend suggests that GNNs may be better suited for handling complex positions, whereas NNs could be more effective in simpler scenarios with fewer stones.

Table 2. Comparison of NN and GNN position classification accuracy.

Number of stones	Shots remaining	Accuracy NN	Accuracy GNN
$r < 4$	1	95%	90%
$4 \leq r < 8$	1	92%	88%
$r \geq 8$	1	87%	88%
$r < 4$	2	87%	86%
$4 \leq r < 8$	2	80%	80%
$r \geq 8$	2	71%	79%

Overall, we observe the following substantial advantages and disadvantages of the two models.

- GNNs do not require a specific ordering of input features, providing flexibility in dynamic game situations where the importance and interrelations of stones can change throughout play.
- GNNs leverage the graph structure to model complex relationships between stones, allowing for a more nuanced understanding of the game state. However, this complexity necessitates advanced data processing techniques.
- While GNNs excel at analyzing relationships between individual stones, they may not capture the holistic strategic dynamics dependent on the overall positioning of all stones as effectively as NNs.

4.2 Bot Comparison

Next, we evaluate each bot's performance in gameplay scenarios. To this end, each of the three bots plays against each other – 10 times with and 10 times without hammer in the first end. This results in 60 games in total, with 10 wins guaranteed (against oneself) and a maximum of 50 wins. The results of this tournament matches are succinctly summarized in the win table, depicted in Table 3.

The NN Bot and the GNN Bot exhibit the strongest overall performance (with 44 and 34 wins, respectively). Conversely, the Heuristic bot shows limited success. In a detailed analysis, we also observe that the Heuristic bot demonstrates better performance when leading but struggles when behind, suggesting that its strategic rules may not be robust with an offensive setup, where the opponent has more stones in play.

As the NN Bot seems to be the most successful bot in the previous test, it is further evaluated through matches against human players of diverse skill levels.

Table 3. Number of wins made by each bot in the tournament.

Bot	# Wins
NN	44
GNN	34
Heuristic	12

By including games against human players, the evaluation puts the bot's capabilities into perspective, offering insights into how well it can compete against experienced and novice curlers alike. The results of these encounters are summarized in Table 4, which details the bot's performance across different categories of player expertise.

Table 4. Performance of the NN bot against human players of varying expertise.

Opponent	# Games	# Wins	Win Rate
Professional Curler	19	0	0%
Experienced Amateur	12	2	17%
Casual Player	9	4	44%
Non-Curler	10	10	100%

While the bot is unable to secure wins against professional curlers, indicating a gap in handling complex strategic scenarios, its performance improves against less experienced players. The bot achieves a 17% win rate against experienced amateurs and 44% against casual players, suggesting it competes effectively in mid-level strategic contexts. Notably, its 100% success against non-curlers demonstrates the bot's capability to outperform complete novices.

5 Conclusions and Future Work

This paper explores the development of strategic decision-making in curling using advanced computational methods. The primary focus is on position analysis, leveraging supervised learning with various data collection techniques to evaluate and rank possible calls. In particular, we propose two representations to formalize arbitrary game situations and stone positions – one with a rich feature vector and one with a labeled graph. Based on these two data structures, we then train a neural network and a graph neural network that are capable to derive the next call. Both frameworks are integrated in a generic curling bot. Both the NN bot and the GNN bot achieve quite high win rates when compared with a heuristic bot that operates on expert rules. Furthermore, we observe that the NN Bot can achieve a win rate of 17% against experienced amateurs, 44% against casual

players, and 100% against non-curlers. However, the best bot obtained in this study is still not able to win against professional curlers, indicating a gap in handling complex strategic scenarios.

We see several rewarding avenues to be pursued in future research. For instance, future improvements could involve using GNNs for more complex positions and NNs for simpler ones, or potentially integrating both networks into a hybrid model. This hybrid approach would combine the strengths of both architectures: GNN's adeptness at handling relational data and NN's efficiency in processing positional features. Integrating these architectures could enable the model to capture both detailed interactions among stones and the overall strategic layout, potentially leading to richer and more precise game analysis.

References

1. Watson, N., Hendricks, S., Stewart, T., Durbach, I.: Integrating machine learning and decision support in tactical decision-making in rugby union. J. Oper. Res. Soc. **72**(10), 2274–2285 (2021)
2. Bunker, R., Susnjak, T.: The application of machine learning techniques for predicting match results in team sport: a review. J. Artif. Intell. Res. **73**, 1285–1322 (2022)
3. Lee, K., Kim, S.A., Choi, J., Lee, S.W.: Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In: International Conference on Machine Learning, pp. 2937–2946. PMLR (2018)
4. Masui, F., Otani, H., Yanagi, H., Ptaszynski, M.: Study on game information analysis for support to tactics and strategies in curling. In: Cabri, J., Pezarat-Correia, P., Vilas-Boas, J. (eds.) icSPORTS 2016-2017. CCIS, vol. 975, pp. 128–149. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14526-2_9
5. Han, Y., Zhou, Q., Duan, F.: A game strategy model in the digital curling system based on NFSP. Complex Intell. Syst. 1–7 (2021)
6. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
7. Wu, L., Cui, P., Pei, J., Zhao, L., Guo, X.: Graph neural networks: foundation, frontiers and applications. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 4840–4841 (2022)

An Empirical Characterization of the Stability of Isolation Forest Results

Alberto Azzari(✉) and Manuele Bicego

University of Verona, Verona, Italy
{alberto.azzari,manuele.bicego}@univr.it

Abstract. Isolation Forests (IForest), a specific variant of Random Forests tailored for anomaly detection, operate by isolating points through recursive partitioning. Despite their widespread use and enhancements in splitting rules, training schemes, and anomaly scoring, an often overlooked aspect is their stability due to the inherent randomness. Surprisingly, most studies and empirical evaluations report results based on a single execution or on the average of a few executions, potentially overlooking significant variability due to this randomness. This paper presents a detailed investigation of the stability of IForests' outcome, proposing some empirical evidence that there may be substantial differences in results across different runs. By exploiting concepts from the field of Ensemble Classifiers, we propose a possible explanation and a strategy to mitigate this instability. Even if we limit our examination to the original IForest model using standard parameters and datasets from the foundational papers, our study underscores the importance of accounting for the random nature of IForests and offers insights and recommendations for practitioners.

Keywords: Isolation Forest · Ensemble Learning · Anomaly Score

1 Introduction

Isolation Forests (IForest) [18,20] represent a particular kind of Random Forests [2] specifically designed for anomaly detection, which usefulness has been shown in many different scenarios [1,6,7]. The main idea behind these methods is that it is possible to measure the isolation of a point by looking at the depth it reaches in a decision tree built on a set of samples of the considered problem. Points that are well inside the distribution would need more splits to be isolated, i.e., would reach a deeper leave in the tree; on the contrary, an outlier, being separated by definition, will be isolated in few splits, i.e., it would reach a leaf in fewer steps. Since their introduction in 2008, isolation forests have received increasing attention from researchers and have been extended in different directions. A non-exhaustive list includes improvements in the splitting rule [11], in the training scheme [10,14], in the computation of the anomaly score [19,21] or even in the applicability to non-vectorial data [22,27].

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 166–176, 2025.
https://doi.org/10.1007/978-3-031-80507-3_17

One of the crucial ingredients of IForests is represented by its intrinsic random nature. In addition to being a Random Forest (thus inheriting the randomicity intrinsic in the bootstrapping procedure), these methods are based on Extremely Randomized Trees (ERT - [9]), i.e., Decision Trees in which randomization is driven to the extreme. Assuming the general scenario of a node's splitting rule defined with a threshold on a feature, within ERT, the rule is defined i) by randomly choosing the feature and ii) by randomly selecting the threshold in the domain of such feature. With this extreme randomization, one may argue about the stability of the results of these methods in practical scenarios. Surprisingly, this aspect is rather underestimated in the literature. In the original paper of Isolation Forests [18], there is no citation to this aspect, and reported empirical results are obtained with only one run of the method. Similarly, classical surveys on anomaly detection, which include Isolation Forests, such as [6] or [7], do not consider this aspect, presenting results of a single run, and the same holds for more recent surveys in specific application scenarios, such as [1] (industrial manufacturing), [13] (natural gas), or [8] (intrusion detection). When this aspect is taken into account (e.g., the journal version of Isolation Forests [20] or very recent approaches – e.g., [26]), the solution is to repeat 10-15 times the execution and take the average, without explicitly examining the specific issue.

This paper is aimed at explicitly studying this aspect, namely at investigating the stability of the results of Isolation Forests. To the best of our knowledge, the only other paper[1] that explicitly considers this aspect is the very recent [4]. In such a paper, the authors conducted a quite large study on the impact on the results of different aspects of IForests, such as parameters (sample size, number of trees), the presence of outliers in the training set, the dimension of the data, and others. Crucially, they also analyze the "Randomness effect", i.e., the stability of IForest results due to the high randomization present in the model. To do this, the authors checked the standard deviation of the accuracy on 10 repetitions of IForest. The analysis, based on a single dataset, led to the conclusions that *"These results illustrate the stability of Isolation Forest despite its randomness. We can, therefore, rely on a single execution of Isolation Forest"*. We will show here that such a conclusion is rather superficial, showing that results among different runs may be rather different. We will also provide an interpretation of this instability by reasoning on the specific way the anomaly score is computed from the forest, borrowing concepts and ideas from the Ensemble Classifier field; following such view, we will also propose a way to reduce such impact partially. In our study, we focus on the original version of IForest, [18,20], employing the standard parametrization and the datasets used in those papers. Even if we do not provide a thorough analysis of different extensions of IF and different datasets, we are convinced that our results provide clear evidence that considering the random nature of IForests is definitely crucial.

[1] Some recent studies (e.g. [25]) tackled the problem of stability of classic Random Forests; even these results are of general interest, they are thought for the classical classification/regression training schemes, and thus do not apply to Isolation Forests, which are based on a very peculiar (and highly randomized) training scheme (ERT).

The rest of the paper is organized as follows. In Sect. 2, we will briefly summarize Isolation Forests, mainly to set up the notation. In Sect. 3, we present the empirical evidence about stability, providing a possible interpretation and (partial) solution in Sect. 4. Finally, in Sect. 5, we conclude our paper.

2 Background

The Isolation Forest (IForest) [18,20] represents an ensemble of Isolation Trees (iTrees). An iTree is a proper binary tree where each node has exactly zero (a leaf) or two (an internal node) children nodes. Each internal node is equipped with a test, which allows an object to go to the left or to the right child according to the test result. Like in classical classification and regression decision trees [3], a test t on an object $\mathbf{x} = x_1, \cdots x_m$ is defined by a tuple $t = (f, v)$ so that the object $\mathbf{x}$ follows the left branch if $\mathbf{x}_f < v$, the right one otherwise, with $\mathbf{x}_f$ being the value of the feature f in the object $\mathbf{x}$.

Training Isolation Trees and Forest. To train an Isolation Tree, we follow a classical top-down, recursive strategy. We start with the whole training set $D = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(L)}\}$ composed by L objects, and choose a test $t = (f, v)$ for the root. The test splits D into D_L and D_R, according to the answer of the test. The sets D_L and D_R are then used to determine the tests in the left and right children. This is recursively done in each node until the tree reaches a height limit or a node contains a minimum number of samples. To select the test $t = (f, v)$ within a node, Isolation Trees use the Extremely Randomized Trees (ERT—[9]) paradigm. Specifically, given a set $D = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(L_k)}\}$ of objects reaching a node, we randomly select a feature f and then randomly generate a split threshold v within the range (f_{min}, f_{max}). Here, f_{min} and f_{max} represent the minimum and maximum values of the feature f among the objects in D, defined as:

$$f_{min} = \min_{i=1\cdots L_k} \mathbf{x}_f^{(i)}, \quad f_{max} = \max_{i=1\cdots L_k} \mathbf{x}_f^{(i)}$$

where $\mathbf{x}_f^{(i)}$ is the value of feature f for the object $\mathbf{x}^{(i)}$.

The Isolation Forest (IForest) represents an ensemble of iTrees. Following the classical recipe of Random Forests [2], each iTree is created, with the procedure described in the previous subsection, starting from a random sub-sampling of the training set.

Computing the Anomaly Score. Given a trained Isolation Forest, the anomaly score $a_{IF}(\mathbf{x})$ of an object $\mathbf{x}$ is computed as follows. First of all, the average path length $\bar{h}(\mathbf{x})$ of $\mathbf{x}$ over multiple trees is computed:

$$\bar{h}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} h_t(\mathbf{x})$$

where T is the number of trees in the forest, and $h_t(\mathbf{x})$ is the number of edges along the path $\mathbf{x}$ follows in the t-th tree. The anomaly score $a_{IF}(\mathbf{x})$ for the object $\mathbf{x}$ is then computed as:

$$a_{IF}(\mathbf{x}) = 2^{-\frac{\bar{h}(\mathbf{x})}{c(n)}} \tag{1}$$

where $c(n)$ is a normalization coefficient representing the expected path length of unsuccessful searches in Binary Search Trees, derived from Isolation Forest [18] and computed with the following equation:

$$c(n) = 2\,H(n-1) - \left(\frac{2(n-1)}{n}\right)$$

where $H(i)$ is the i-th harmonic number ($H(i) = \ln(i) + 0.5772156649$ **(Euler's constant)**) and n is the number of instances in the given dataset.

3 Stability of Isolation Forest Results

In this section, we analyze the stability of the results of the IForest method; as in many outlier detection problems, to quantify the performances we employed the ROC-AUC measure, computed starting from the anomaly scores defined in Eq. (1). We focus here on the standard configuration of IForest, suggested in the original papers [18,20] and adopted in many other settings (e.g., [4]); in particular each Isolation Tree is built with a random subsample of 256 objects, growing the three until the maximum depth is 8 (i.e. $\log(n)$, $n = 256$). The forest is composed of 100 trees. Our empirical analysis is based on the datasets employed in the original papers of Isolation Forests [18,20] – except for the Annthyroid problem, for which we were not able to recover the same version (in the web version we found[2] the dataset contains 7200 objects, whereas in the Isolation Forest paper 6832, without citation to the preprocessing). A summary of the datasets is reported in Table 1.

Table 1. Datasets employed in the experiments.

Name	Number of objects	Number of features	Percentage of outliers
Http	567497	3	0.4%
Cover	286048	10	0.9%
Mulcross	262144	4	10%
Smtp	95156	3	0.03%
Shuttle	49097	9	7%
Mammography	11183	6	2%
Annthyroid	7200	6	2%
Satellite	6435	36	32%
Pima	768	8	35%
Breastw	683	9	35%
Arrhythmia	452	274	15%
Ionosphere	351	32	36%

[2] https://odds.cs.stonybrook.edu/annthyroid-dataset/.

To analyse the stability, we tried to answer to the following practical question: suppose that I get one result with IForest, how different would be the result if I make another run? In order to quantify this, we adopted the following strategy:

1. Train the first IForest, compute the anomaly scores and determine the AUC (AUC 1^st run).
2. Train another IForest, computing again the anomaly scores and the AUC.
3. compare the two AUCs and two sets of anomaly scores to assess if there is a statistically significant difference among them. To do that, we exploited the equivalence between the AUC and the Wilcoxon or Mann-Whitney U test statistic, as explained in [12]. The employed test also returns a p-value[3]
4. Repeat the above procedure 200 times, counting the number of times the two runs were considered different with a statistical significance; as significance level, we employed $\alpha = 0.05$, also performing a Bonferroni Correction for multiple tests. In order to reduce the computational burden of training so many IForests, we created 2000 Isolation Trees, randomly sampling from this superset the 100 trees of each Forest.

3.1 Results

The obtained results are shown in Table 2 for the different datasets: in particular, we reported the number of times, over the total, it happened that a run of IForest was different than the first run with a statistical significance (Num. S.S.D.). To have a clearer idea, we also reported the AUC of the first run and the averaged AUC of the 200 other runs (with min and max).

Table 2. Results for standard configuration of IForest.

Problem	Num. S.S.D.	AUC 1^{st} run	Avg AUC other runs (min-max)
Http	169/200	1.000	0.999 (0.997–1.000)
Cover	176/200	0.886	0.886 (0.796–0.949)
Mulcross	197/200	0.947	0.956 (0.924–0.978)
Smtp	12/200	0.906	0.907 (0.892–0.925)
Shuttle	180/200	0.996	0.997 (0.995–0.999)
Mammography	75/200	0.861	0.862 (0.842–0.886)
Annthyroid	161/200	0.832	0.816 (0.776–0.865)
Satellite	127/200	0.685	0.697 (0.669–0.746)
Pima	20/200	0.669	0.675 (0.642–0.706)
Breastw	0/200	0.989	0.986 (0.982–0.990)
Arrhythmia	0/200	0.800	0.804 (0.777–0.832)
Ionosphere	1/200	0.855	0.854 (0.839–0.866)

[3] Code is available here: https://github.com/alistairewj/auroc-matlab.

The most evident observation that can be derived from the table is that for many different datasets, the number of statistically different runs is very high, being almost 0 in only 3 cases. Actually, if we consider the AUC, we can observe that within the 200 runs, values are rather different than the AUC of the first run (for example, for the Cover dataset, the minimum is 0.09 lower than the first run). From these numbers, it seems evident that the randomness effect should be carefully considered and that we cannot show the results deriving from a single run of IForest. The phenomenon is more evident for large datasets, whereas it is less relevant for datasets with less than 1000 objects. This seems reasonable since with smaller datasets, the sets of objects used to train each tree (i.e., via subsampling) are more overlapped, thus resulting in fewer diverse trees.

3.2 Results with More Trees

Results presented in the previous section clearly show that randomicity in IForests can lead to unstable results. One possible (and rather obvious) solution would be to increase the number of trees, thus exploiting the ensemble effect to reduce variability. This option would clearly increase the computational complexity, but we are interested here in a different question: does this solution increase the stability in all cases? To investigate this aspect, we repeated the experiments of the previous section by increasing the number of trees, i.e., by considering 200, 400, and 800 trees. We aggregate the results by taking averages over groups of datasets in order to understand the behavior when considering the following specific aspects: i) size of the dataset (small vs. large problems), ii) dimensionality of the feature space (high dimensional vs. low dimensional problems), iii) AUC 1^{st} run (difficult vs. easy problems) and iv) percentage of outliers (problems with few or many outliers). For each aspect, we split the 12 datasets into two groups of 6: for example, for the size, we considered in one group the 6 smallest datasets (Ionosphere, Arrhythmia, Breastw, Pima, Satellite, and Annthyroid), whereas in the other group, the remaining ones. We then took the averages of the number of statistically significant different runs for an increasing number of trees, reporting such values in Fig. 1 for the 4 different aspects.

From the figures, we can observe that, as expected, increasing the number of trees reduces the instability problem in many cases; however, problems may still appear when working with difficult problems involving many outliers.

4 A Possible Explanation

In this section, we provide some considerations on the possible origin of the instability of the IForest results. In particular, we start from the definition of the anomaly score computed from the forest, as given in Eq. (1), and we interpret it using concepts and ideas of the Ensemble Classifier field [17]. Specifically, it is easy to see that Eq. (1) can be rewritten as follows (for the sake of readability,

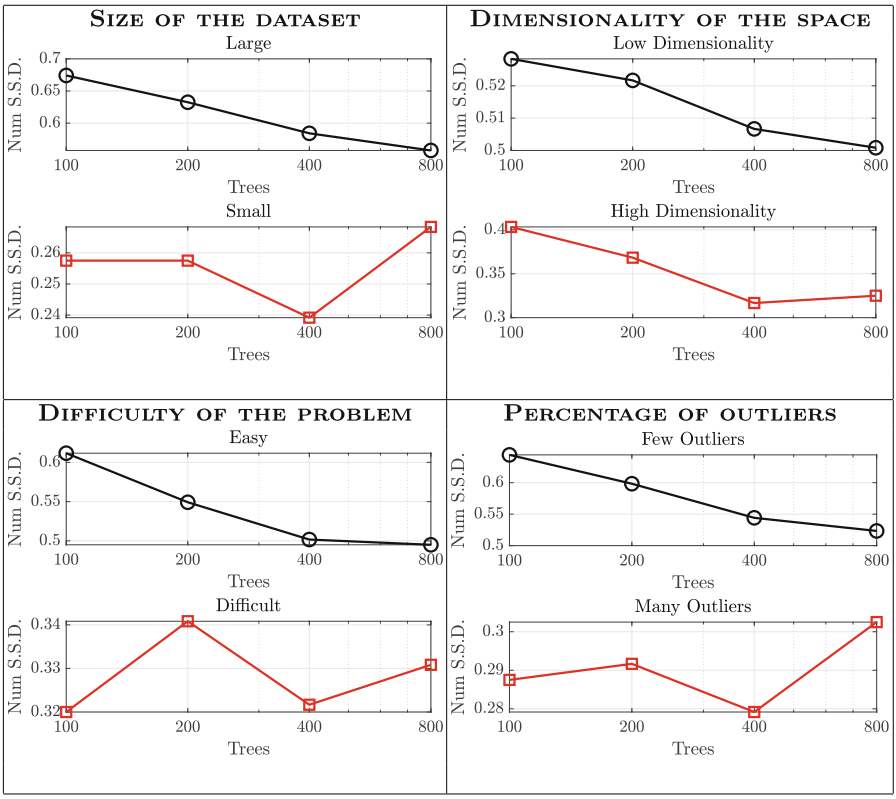


Fig. 1. Results when growing the number of trees.

we removed the $c(n)$ factor, which is constant over different trees, and does not impact our reasoning):

$$a_{\text{IF}}(x) = 2^{-\bar{h}(x)} = 2^{-\frac{1}{T}\sum_{t=1}^{T} h_t(x)} = \prod_{t=1}^{T} 2^{-\frac{h_t(x)}{T}} \tag{2}$$

By considering that each isolation tree t returns its own anomaly score:

$$a_{\text{IT}}(x) = 2^{-\frac{h_t(x)}{T}} \tag{3}$$

we can observe that the original anomaly score $a_{\text{IF}}(x)$ simply represents the output of an ensemble of classifiers, in which the individual outputs are combined together with the PROD rule, one of the combination rules which go under the name of nontrainable (or fixed) combiners [17]. In the ensemble learning community, the PROD rule has been largely studied, both from a theoretical [15,16] and an empirical perspective [23,24], especially in comparison with the SUM rule. From all these studies, it emerges that the PROD rule is rather a severe combiner rule, being a proper choice only in the absence of estimation errors, but

being more sensitive if such errors are present; in such cases, more benevolent rules (such as the SUM rule) represent a more appropriate choice. In particular, authors of [16] suggested that the PROD rule may be less accurate since a single bad score can change the overall output due to the restricted behavior of the product. This means that, in the IForest case, one single bad tree may drastically change the final output; since trees are randomly created, bad trees may be generated.

This possible explanation of the instability of IForests opens the route to the investigation of alternative combiners, which may reduce the instability of IForest. We propose here some preliminary experiments along this direction, employing some classic rules, such as the SUM[4], the MEDIAN, the MIN, the MAX and the TRIMMEDSUM (i.e., the sum rule in which the 5% larger and smaller scores are removed [17]). Results are shown in Table 3, in which we reported the Num. S.S.D. over three datasets: Shuttle, Mammography, and Pima, belonging to different stability category (low, medium, high, respectively).

Table 3. Comparison with other combiners, showing Shuttle, Mammography and Pima datasets.

Method	Shuttle	Mammography	Pima
Original (PROD)	180/200	75/200	20/200
SUM	150/200	50/200	0/200
TRIMMED-SUM	150/200	48/200	4/200
MEDIAN	140/200	64/200	14/200
MIN	200/200	80/200	30/200
MAX	158/200	34/200	0/200

From the Table 3, it seems evident that alternative rules may improve the stability of IForest results, this being especially true for the SUM rule. The only exception is the MIN rule, which, however, can be considered as an approximation of the PROD rule [15]. To include the AUC in the perspective, we report in Fig. 2 a critical diagram showing the result of a Friedman test, followed by the post-hoc Nemenyi test [5], which compares the AUC of the different aggregation rules over the different datasets: the plot reports rankings – the lower the better–, with the difference between methods connected by a line being not statistically different. We can observe that the performances of most of the alternatives are equivalent to those of the original score, except for the MIN and the MAX rule, which results are drastically lower. Thus we can confirm the general findings of Kuncheva, who, in her book [17] says that *"The current understanding is that*

[4] Please note that a novel anomaly score for IForest, which can be seen as equivalent to the SUM rule, has also been investigated in the recent [21]; in such paper, however, it was introduced mainly from a probabilistic perspective, with experiments focused only on improving the AUC.

the average (i.e., the sum), in general, may be less accurate than the product for some problems, but is the most stable of the two".

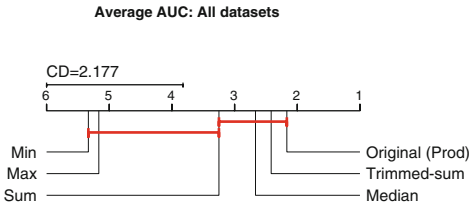


Fig. 2. Critical difference diagram showing the ranks of the employed combiners based on the average AUC across all datasets.

5 Conclusions

This study delves into the stability of Isolation Forests, a widely used anomaly detection technique, with a focus on understanding how its inherent randomness affects the repeatability of its results. Our empirical analysis reveals significant variability in the Area Under the Curve (AUC) scores across multiple runs of the IForest algorithm. This highlights a critical issue: despite their popularity and effectiveness, the random nature of IForest can lead to unstable outcomes, which can undermine the reliability of conclusions drawn from a single execution. Increasing the number of trees can reduce this variability, but doesn't entirely eliminate it. Alternative methods for combining anomaly scores, like the sum, offer improved stability over the default product rule, suggesting that these should be considered to enhance the reliability of IForest outcomes.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Barbariol, T., Chiara, F.D., Marcato, D., Susto, G.A.: A review of tree-based approaches for anomaly detection. In: Tran, K.P. (ed.) Control Charts and Machine Learning for Anomaly Detection in Manufacturing. SSRE, pp. 149–185. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-83819-5_7
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
3. Breiman, L., Friedman, J.H., Olshen, R., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
4. Chabchoub, Y., Togbe, M.U., Boly, A., Chiky, R.: An in-depth study and improvement of isolation forest. IEEE Access **10**, 10219–10237 (2022)
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)

6. Domingues, R., Filippone, M., Michiardi, P., Zouaoui, J.: A comparative evaluation of outlier detection algorithms: experiments and analyses. Pattern Recogn. **74**, 406–421 (2018)
7. Emmott, A.F., Das, S., Dietterich, T., Fern, A., Wong, W.K.: Systematic construction of anomaly detection benchmarks from real data. In: Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, pp. 16–21 (2013)
8. Falcão, F., et al.: Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 318–327 (2019)
9. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**, 3–42 (2006)
10. Guha, S., Mishra, N., Roy, G., Schrijvers, O.: Robust random cut forest based anomaly detection on streams. In: International Conference on Machine Learning, pp. 2712–2721. PMLR (2016)
11. Hariri, S., Kind, M.C., Brunner, R.J.: Extended isolation forest. IEEE Trans. Knowl. Data Eng. **33**(4), 1479–1489 (2019)
12. Hernández-Orallo, J., Flach, P., Ferri Ramírez, C.: A unified view of performance metrics: translating threshold choice into expected classification loss. J. Mach. Learn. Res. **13**, 2813–2869 (2012)
13. Jha, H., Khanal, A., Seikh, H., Lee, W.: A comparative study on outlier detection techniques for noisy production data from unconventional shale reservoirs. J. Nat. Gas Sci. Eng. **105**, 104720 (2022)
14. Karczmarek, P., Kiersztyn, A., Pedrycz, W., Al, E.: K-means-based isolation forest. Knowl.-Based Syst. **195**, 105659 (2020)
15. Kittler, J.: Combining classifiers: a theoretical framework. Pattern Anal. Appl. **1**, 18–27 (1998)
16. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **20**(3), 226–239 (1998)
17. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley (2004)
18. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422. IEEE (2008)
19. Liu, F.T., Ting, K.M., Zhou, Z.-H.: On detecting clustered anomalies using SCiForest. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS (LNAI), vol. 6322, pp. 274–290. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15883-4_18
20. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data (TKDD) **6**(1), 1–39 (2012)
21. Mensi, A., Bicego, M.: Enhanced anomaly scores for isolation forests. Pattern Recogn. **120**, 108115 (2021)
22. Mensi, A., Tax, D.M., Bicego, M.: Detecting outliers from pairwise proximities: proximity isolation forests. Pattern Recogn. **138**, 109334 (2023)
23. Tax, D.M., Duin, R.P., Van Breukelen, M.: Comparison between product and mean classifier combination rules. In: Proceedings of Workshop on Statistical Pattern Recognition, Prague, Czech, p. 39. Citeseer (1997)
24. Tax, D.M., Van Breukelen, M., Duin, R.P., Kittler, J.: Combining multiple classifiers by averaging or by multiplying? Pattern Recogn. **33**(9), 1475–1485 (2000)
25. Wang, Y., Wu, H., Nettleton, D.: Stability of random forests and coverage of random-forest prediction intervals. In: Advances in Neural Information Processing Systems, vol. 36 (2024)

26. Xiang, H., et al.: Optiforest: optimal isolation forest for anomaly detection. In: Proceedings of International Joint Conference on Artificial Intelligence (2023)
27. Zhang, X., et al.: Lshiforest: a generic framework for fast tree isolation based ensemble anomaly analysis. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 983–994. IEEE (2017)

Automated Classification of Android Games Using Word Embeddings

Elena Flondor(✉) and Marc Frincu

Faculty of Mathematics and Computer Science, West University of Timisoara, bv. Vasile Parvan, Timisoara, Romania
elena.flondor97@e-uvt.ro, mfrincu@info.uvt.ro

Abstract. Application stores allow developers to organise their applications into general categories, such as social, music, games, etc. For the games category, they suggest a list of subcategories. The need to automate the games classification process to avoid misclassification is increasing as the mobile gaming industry continues to grow. However, there are few researchers focusing on game classification. As their results are poor, this research compares five methods to identify the category of mobile games: Decision Tree, Random Forest, Extreme Gradient Boosting, Gradient Boosting Machines, and Artificial Neural Networks. We use a large collection of 147,128 mobile games to ensure diversity in our data set. We extract features from game descriptions, manifest encodings and from application packages. To capture the semantic meaning of words, we use four embedding methods: Word2Vec, fastText, BERT, MiniLM. For each embedding method, we evaluate the performance of the classifiers. Our results show that Extreme Gradient Boosting can outperform existing work, achieving 81% precision in classification.

Keywords: Supervised Machine Learning · Word Embeddings · Classification · Android Games · Word2Vec · BERT · fastText · MiniLM

1 Introduction

A significant proportion of the population has access to smartphones and uses them for a variety of purposes. Their wide availability and functionality created a new platform for the gaming industry. Statistics show that in the first quarter of 2022, a total of 14.4 billion mobile games were downloaded globally across both, the Google Play Store (GPS) and the App Store [17]. Mobile application stores provide numerous types of games to satisfy the users' need to play their favourite games anywhere, anytime. Some leading mobile app publishers allow developers to choose from pre-defined categories for their games [1,2].

With thousands of applications classified under each category, existing work is focused on automating the classification of Android applications to avoid misclassification. To accomplish this task, applications and their metadata were

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 177–187, 2025.
https://doi.org/10.1007/978-3-031-80507-3_18

parsed for feature extraction and Supervised to Unsupervised Machine Learning techniques were considered (Sect. 2). However, only a few researchers studied the problem of games classification (Sect. 2).

In this work, *our focus is on automating the process of classifying mobile games to help users select games and avoid misclassifying them on the markets.* We extract features from both market metadata - game descriptions [4,14,15], and game packages [12,24] - manifest components, permissions, etc. (Sect. 3.1). Descriptions provide context and intended use, while game package features provide concrete technical details. We use natural language processing (NLP) techniques to prepare descriptions for embedding, as these representations have proven effective in classification tasks [11]. To encode the manifest components, we adapted our previous work based on a bag-of-words approach [12]. Embedded descriptions, encoded manifest components and the array of features from application packages are passed to powerful classification algorithms, such as: Decision Tree (DT) [7], Random Forest (RF) [6], Extreme Gradient Boosting (XGB) [8], Gradient Boosting Machine (GBM) [13], and Artificial Neural Network (ANN) [18] to evaluate their ability to detect game types. In summary, we highlight our key contributions as follows:

- We propose an automated method for classifying mobile games based on the game descriptions, manifest features, permissions, and package features.
- We show through experiments that econding descriptions with Word2Vec helps to achieve the highest classification performance.
- We achieve 81% precision and 80% accuracy, recall, and f1-score with XGB by combining the extracted features, and using a large data set. We also show that XGB classifies is more time-efficiently than other used classifiers.

2 Literature Survey

This section reviews the existing work focused on classification of mobile games.

Classifying mobile applications has been a topic of interest for researchers in the past. In addition, several researchers focused on the classification of mobile games. For example, Babatunde [4] used a collection of 5,174 games (17 categories) and studied the performance of the Naive Bayes classifier. The researchers used descriptions, content ratings, whether the application is free, whether it has in-app purchases. The descriptions were cleaned by removing stop-words, numbers, and punctuation. Then applied TF-IDF [3] to reduce the influence frequently appearing tokens, and 67% accuracy was achieved for game classification. Wang et al. [25] extracted multiple features from application packages (permissions, intent filters, API calls, hardware features) and achieved 66% accuracy in classifying 8 categories of games (18,866 samples). Other researchers [24] used a set of 25,001 games and built a Convolutional Neural Network to classify them using icons and screenshots. The results showed an accuracy of 55.3%. Horppu et al. [15] used a collection of 2,443 iOS App Store games to classify them based on their descriptions and titles. The descriptions were cleaned by

removing punctuation, special characters, noise words, company names, and by excluding advertising for other games. Latent Semantic Indexing was also used to reduce the curse of dimensionality, and Support Vector Machine achieved 77% accuracy.

Compared to previous work, we use a large dataset of about 150,000 mobile games to have a wide variety. We use descriptions from the store and features extracted form the applications' package: manifest components, permissions, features related to file types, etc. We avoid the use of content rating, price and the presence of in-app purchases as we do not consider these to be relevant to a game category. To process the descriptions, we use NLP techniques similar to those used in the works discussed (lowering, removal of special characters, stop-words, URLs and numerical characters). We also removed emojis, emails, entity names, words that occur only once in the entire description corpus, and corrected misspelled and stuck words. Instead of TF-IDF [4], we encoded the cleaned descriptions as word embeddings (Word2Vec [19], fastText [5], Bert [10], miniLM [26]) to capture semantic relationships between words in the descriptions of mobile games. They help to overcome the syntactic limitations of these descriptions. Manifest components are encoded using a bag-of-words based method [12] and used as an array of features. We also extract numeric features from AndroidManifest.xml and file type based features. All these features are used to automatically classify games. We achieve better results with XGB and compare them with several algorithms mentioned in previous work.

3 Proposed Framework

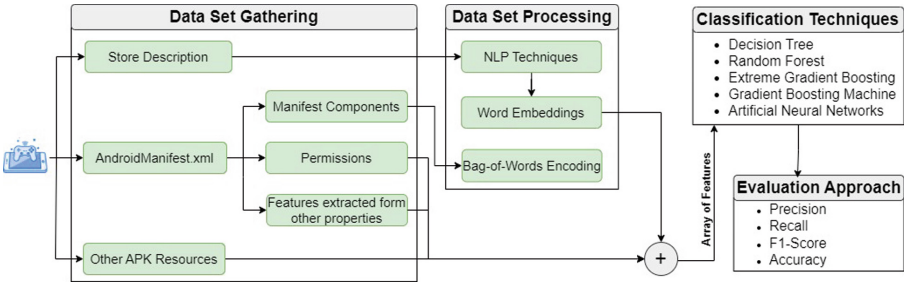


Fig. 1. Research framework.

In this research work, we studied the performance of several classification algorithms (DT, RF, XGB, GBM, ANN) for the classification of mobile games. The proposed framework for the process is shown in Fig. 1. We adapted (Sect. 3.2, Sect. 3.3) previous work [4,12,14,15] and obtained improved results by considering a wider range of NLP techniques for description processing and by using features extracted from application packages (Sect. 3.2).

The first step is the **Data Set Gathering** process (Sect. 3.1). Second, we move to the **Data Set Processing** (Sect. 3.2). Thirdly, we use **Classification Techniques** (Sect. 3.3). Finally, we apply the **Evaluation Approach** (Sect. 3.4).

3.1 Data Set Gathering

The data set includes mobile games published on the GPS from 2019 to the present. The store offers 17 sub-categories of games. To ensure variation in our data set, we considered 147,182 mobile games from 9 categories, exceeding the number of samples in the data sets of previous works [4,14,15,24]. The categories and the number of samples in each one are: Racing - 8,892; Educational - 12,597; Puzzle - 44,826; Board - 8,635; Action - 19,185; Casino - 5,075; Sports - 3,954; Word - 8,374; Music - 4,082. For each game, we gathered the English descriptions from the market and the features from the application package. We used APKTool [16] to decompile the games to get the *AndroidManifest.xml* files and to access file resources to extract other features, for which we obtained promising results in classifying Android applications in our previous work [12].

The *AndroidManifest.xml* file declares the permissions required by an Android application (internet access, camera access, etc.), it lists the components of the application (activities, services, broadcast receivers and content providers), and it specifies the API level supported and the application's capabilities and features (hardware and software features), along with the application's metadata. We extracted: the names of the manifest components - which will be encoded for classification process (Sect. 3.2), list of used permissions, numerical features representing: the number of manifest entry points, the number of each type of components, the size of manifest version name, the number of manifest resources, the occurrence of specific keywords (e.g., view, send) in the names of actions. We also extracted numerical features representing the number of different file types in the application packages (number of audio files, of image files, of video files, etc.), and features related to the directories of application packages: number of files in the resources directory, number of directories, etc. The size of the mobile games is also used as a feature in our classification process.

3.2 Data Set Processing

For data set processing we followed two main directions.

The first direction proposes NLP techniques and word embeddings to process game descriptions to improve algorithms' ability to understand, interpret and classify text effectively. We use NLP techniques from previous work (Sect. 2), but also consider other techniques to improve the quality of the descriptions:

Lowering and Special Characters Removal. The first step in descriptions processing is text lowering, followed by special characters removal.

Stop-Words, URLs, Emails, Emojis Removal. We remove URLs, emails, and emojis as they do not offer relevant information for our classification task and they can increase storage and processing costs.

Correction of Misspelled and Stuck Words. By analysing the games descriptions, we identified the presence of approximately 475,000 misspelled or nonsense words (e.g. rivalrours, pw8eqbr97n6htnnu, anceokexhuobi, etc.) and approximately 2,246,000 stuck words (e.g. additionsubtractionmultiplicationdivisionnow, dictionarysupported, takescare, ourmission, etc.). Since they represent noise and lack of information in the learning process, we tried to correct them using *pyspellchecker 0.8.1* and *wordsegment 1.3.1* from *Python*, and removed those for which we could not find a suitable replacement.

Named Entity and Numerical Values Removal. We consider names of places, organisations, products, numerical values or ordinals to be noise words. We remove them because such words do not reflect the behaviour of the games (e.g. instagram, facebook, telegram, italy, afghan, bulgaria, 2024, first, secondly, etc.).

In additon, we *removed words that occur only once in the description corpus* to reduce dimensionality and noise, and to prevent overfitting.

We used well-known word embeddings to prepare clean descriptions for the game classification process. We use **Word2Vec** [19] - a two-layer neural network that uses a continuous bag-of-words model and a skip-gram model to generate word embeddings; **fastText** [5] - is based on the Word2Vec skip-gram model and it generates the vector representations of a single word by averaging the vectors of its n-grams; **BERT** [10] - uses an encoder-only architecture and uses a bidirectional method that considers both the left and right context of words in a sentence rather than analysing the text sequentially; and **MiniLM** [26] - compresses BERT's knowledge through a dual approach of distillation, using a smaller number of layers, making it more efficient.

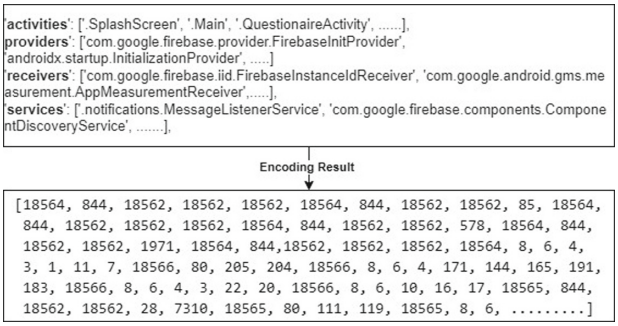


Fig. 2. Example of manifest component encoding.

The second direction follows the processing of manifest components, which we adapted from our previous work [12]. Words from manifest components extracted from *AndroidManifest.xml* were filtered out. We built a vocabulary of words that occur only in games of a single category and with words that occur in more than 10 game descriptions. Therefore, we reduced the vocabulary

from 306,103 to 18,568 words to embed only the most relevant ones. We created vector representations for each game manifest: the beginning of each component name was marked with a specific indicator, words not present in the vocabulary were marked with a different indicator, we set the length of the encoding to 200 based on the median length of the words in the manifest component names, and we padded the encoding if the number of manifest components did not reach the fixed size. Figure 2 shows the result of the encoding process for a given manifest.

We concatenated each description embedding, with manifest encodings, and with the other features extracted from the games' package, and used them as an array of features in the classification process.

3.3 Classification Techniques

In this step, we classify the games in our data set using the features sets obtained during data set processing (Sect. 3.2). We use a supervised multi-class strategy to classify our mobile games, and we experiment with several classification algorithms: DT [7], RF [6], XGB [8], GBM [13] and ANN [18]. These methods are usually used to classify Android applications [14,21–23] due to their capacity to work with short texts such as descriptions.

Our analysis is carried out using the Python library *Scikit-learn 1.5.0* [20] which integrates a wide range of Machine Learning algorithms for supervised and unsupervised classification problems. We use *DecisionTreeClassifier*, *RandomForestClassifier*, and *GradientBoosingClassifier*. For Extreme Gradient Boosting we use *XGBClassifier* from *xgboost* Python library [9]. For all previous algorithms, we used the default hyperparameter configuration. For the implementation of the ANN, we used *keras* from *tensorflow 2.14.0*, and through experiments we observed that the architecture that leads to the best classification results that can be achieved with an ANN in this context consists of: five dense layers with 512, 256, 128, 64, and 32 neurons and *relu* activation function. Each dense layer is followed by a dropout with a rate of 0.5 to reduce overfitting.

3.4 Evaluation Approach

To evaluate the performance of the algorithms mentioned in Sect. 3.3 we applied: $F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$. We define: TP (true positives) - the number of the games correctly predicted; TN (true negatives) - the number of games correctly predicted as not belonging to inappropriate categories; FP (false positives) - the number of games incorrectly classified; FN (false negatives) - the number of games incorrectly excluded from appropriate categories. We define accuracy as: $Accuracy = \frac{correct\_predictions}{all\_predictions}$.

4 Results and Analysis

Here we discuss the results obtained in the classification of mobile games using the set of features obtained after concatenating each description embedding with

manifest encoding and with the set of features extracted from the game package (Sect. 3.2). As mentioned above, we used five classifiers: DT, RF, XGB, GBM and ANN (Sect. 3.3). For each classifier we used cross-validation to get 25% of the data to test and 75% to train the models. We also used class weights, to avoid biasing the models towards the more frequent classes (Sect. 3.1). We calculated them using *class_weight* from *Scikit-learn 1.5.0*.

The results of classifying our set of mobile games are shown in Table 1. Our classification algorithms performed best when the game descriptions were vectorised using Word2Vec. In particular, XGB was able to achieve the best result, 81% precision and 80% accuracy, recall, f1-score in separating the game categories. Figure 3 shows the confusion matrix. The algorithm performed best at recognising casino games (86.89% accuracy) and music games (86.83% accuracy), but also the results for action, racing, word and educational games were also above 80% accuracy. It performs worst for board games, with about 67.47% accuracy; 15% of these games are incorrectly classified as puzzle games. The classifier also identifies 8% of the educational games as being from the puzzle category. Misclassification also occur in the case of racing and sports games: 10%, respectively about 9% of these games are classified as action games.

XGB obtained comparable results when embedding the descriptions using fastText (79% precision) and BERT (78% precision). We noticed that using all four vectorisation methods, we obtained second best classification results with the proposed ANN (e.g., 80% precision using Word2Vec). DT classifier performed worst in all classification cases, barely achieving 63% precision with Word2Vec.

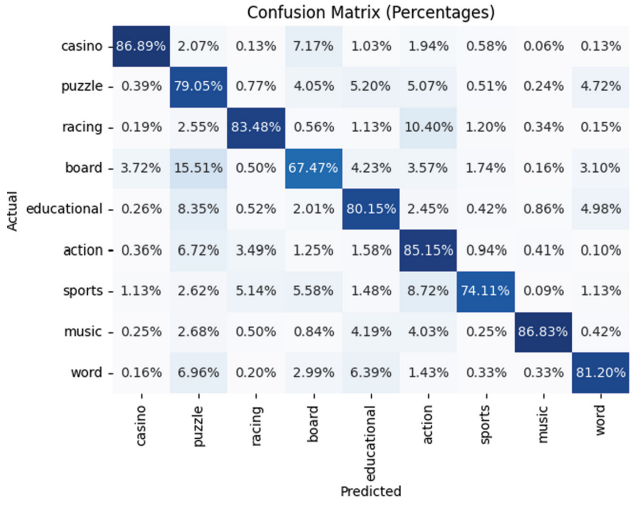


Fig. 3. XGB classifier confusion matrix.

Since our best results were obtained for description embedding with Word2Vec, we analysed the time costs during the training process in the case of

Table 1. Evaluation metrics values of classifiers using different embeddings.

Description Embedding Method	Classifier	Classification Metrices			
		Precision	Recall	F1-Score	Accuracy
Word2Vec	DT	63%	63%	63%	63%
	RF	78%	77%	77%	77%
	XGB	**81%**	**80%**	**80%**	**80%**
	GBM	78%	76%	76%	76%
	ANN	80%	77%	78%	77%
fastText	DT	57%	57%	57%	57%
	RF	76%	72%	71%	72%
	XGB	**79%**	**79%**	**79%**	**79%**
	GBM	76%	74%	74%	74%
	ANN	77%	76%	76%	76%
BERT	DT	53%	54%	54%	54%
	RF	75%	69%	68%	69%
	XGB	**78%**	**78%**	**78%**	**78%**
	GBM	75%	73%	73%	73%
	ANN	77%	74%	74%	74%
MiniLM	DT	48%	48%	48%	48%
	RF	73%	65%	62%	65%
	XGB	**75%**	**74%**	**75%**	**74%**
	GBM	71%	68%	68%	68%
	ANN	75%	72%	73%	72%

this method (Table 2). The analysis shows that XGB was the most efficient in terms of time cost, with a training time of approximately 16 s. The less efficient was GBM with more than 1 h for training.

The results obtained in our work outperformed existing work mentioned in Sect. 2. Compared to previous researchers who used data sets with a smaller number of samples divided into several categories [4,15,24,25], which may not cover the variety of games in each category, our data set consists of 147,182 samples divided into 9 categories. In terms of classification performance, previous researchers achieved 55.3% [24], 66% [25], 67% [4], and 77% [15] accuracy with their methods. We managed to achieve an accuracy of 80% (81% precision), surpassing their work. Looking more closely at the results, we observe that using three out of four of the proposed methods for the encoding the game description (Word2Vec, fastText, BERT) with manifest encodings, and the set of features extracted from application packages helped to overachieve works discussed in Sect. 2 by applying XGB. Furthermore, the results show that we outperform several of the previous works by all the applied methods except DT.

Table 2. Time cost of classifiers using Word2Vec embedding during training process.

Classifier	DT	RF	XGB	GBM	ANN
Time Costs	44.85 s	2 m 21.24 s	**15.95 s**	1 h 10 m 10 s	2 m 22.43 s

As our initial data set is unbalanced, we evaluated our proposed approach on a balanced one. We randomly selected 3000 samples for each of the nine categories, creating a data set of 27,000 samples. Since XGB gave the best classification results, we trained it on 75% of this data and used 25% to evaluate its performance. The experiment showed 81% for accuracy, precision, recall, and f1-score. The confusion matrix (Fig. 4) showed that the classifier correctly identified 90.66% of the music games, 88.99% of the casino games, 85.78% of the word games. It performed worse in identifying board games, 70% correctly identified.

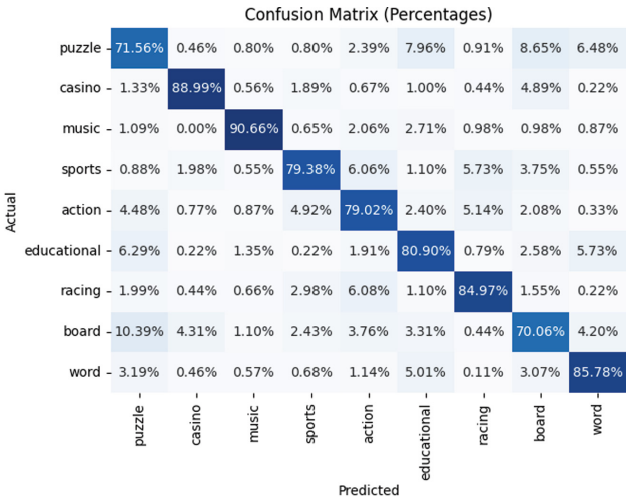


Fig. 4. XGB classifier confusion matrix for balanced data set.

5 Conclusion and Future Work

This paper discussed the performance of several classification algorithms (Sect. 3.3) in the context of identifying categories of mobile games with similar functionalities using their descriptions published on markets, manifest components, permissions, and other features extracted from applications' package (Sect. 3.1). We used NLP techniques to prepare the descriptions for the embedding step, and used four different embedding methods (Word2Vec, fastText, BERT, MiniLM) to encode game descriptions. We adapted our previous work

[12] and built a vocabulary of the most relevant words found in the manifest components names to encode them as an array of features (Sect. 3.2).

According to our results, XGB outperformed classifiers such as DT, RF, GBM, and our architecture of ANN for each encoding method applied to the game descriptions (Table 1). It was also the best in terms of efficiency, as its training process took less than 16 s (Table 2). All our classifiers were able to outperform some of the previous works. Moreover, the proposed classification is accurate enough to provide useful information to those involved in the mobile gaming industry, and it can be used to analyse the market trends.

For future work, we plan to improve the classification results by extracting features from the games' source code and resources, and by dynamic analysis. We also plan to extend the number of the categories to make our solution applicable to all game categories provided by the stores.

References

1. Apple app store - choosing a category. https://developer.apple.com/app-store/categories/. Accessed 18 June 2023
2. Google play store - choose a category and tags for your app or game. https://support.google.com/googleplay/android-developer/answer/9859673?hl=en. Accessed 14 Feb 2023
3. TF-IDF, pp. 986–987 (2010). https://doi.org/10.1007/978-0-387-30164-8_832
4. Babatunde, O.: Applying naive bayes classification to google play apps categorization. CoRR abs/1608.08574 (2016). http://arxiv.org/abs/1608.08574
5. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017). https://doi.org/10.1162/tacl_a_00051
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
7. Breiman, L., Friedman, J., Olshen, R.A., Stone, C.J.: Classification and Regression Trees, 1st edn. Chapman and Hall/CRC (1984). https://doi.org/10.1201/9781315139470
8. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: KDD 16, pp. 785–794. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2939672.2939785
9. Chen, T., Guestrin, C.: XGBoost documentation (2024). https://xgboost.readthedocs.io/en/stable/. Accessed 26 June 2024
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019). https://doi.org/10.18653/v1/N19-1423
11. Ebrahimi, F., Tushev, M., Mahmoud, A.: Classifying mobile applications using word embeddings. ACM Trans. Softw. Eng. Methodol. **31** (2021). https://doi.org/10.1145/3474827
12. Flondor, E.: Exploring androidmanifest.xml for automated android apps classification. In: 2023 IEEE International Conference on Big Data (BigData), pp. 6145–6147 (2023). https://doi.org/10.1109/BigData59044.2023.10386962

13. Friedman, J.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001)
14. Guendouz, M., Amine, A., Hamou, R.: Machine learning based classification of android apps through text features (2017)
15. Horppu, I., Nikander, A., Buyukcan, E., Mäkiniemi, J., Sorkhei, A., Ayala-Gómez, F.: Automatic classification of games using support vector machine. CoRR abs/2105.05674 (2021). https://arxiv.org/abs/2105.05674
16. iBotPeaches: Apktool: a tool for reverse engineering android APK files (2016). https://ibotpeaches.github.io/Apktool/
17. Clement, J.: App store and google play mobile game downloads worldwide 2018-2022 (2024). https://www.statista.com/statistics/661553/global-app-stores-mobile-game-downloads/
18. Jain, A., Mao, J., Mohiuddin, K.: Artificial neural networks: a tutorial. Computer **29**(3), 31–44 (1996). https://doi.org/10.1109/2.485891
19. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, 2–4 May 2013, Workshop Track Proceedings (2013). http://arxiv.org/abs/1301.3781
20. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
21. Reyhani Hamedani, M.: AndroClass: an effective method to classify android applications by applying deep neural networks to comprehensive features. Wirel. Commun. Mob. Comput. **2018**, 21 (2018). https://doi.org/10.1155/2018/1250359
22. Rungta, M., Sherki, P.P., Dhaliwal, M.P., Tiwari, H., Vala, V.: Two-phase multimodal neural network for app categorization using APK resources. In: 2020 IEEE 14th International Conference on Semantic Computing (ICSC), pp. 162–165 (2020). https://doi.org/10.1109/ICSC.2020.00032
23. Shimagaki, J., Kamei, Y., Ubayashi, N., Hindle, A.: Automatic topic classification of test cases using text mining at an android smartphone vendor, pp. 1–10 (2018). https://doi.org/10.1145/3239235.3268927
24. Suatap, C., Patanukhom, K.: Game genre classification from icon and screenshot images using convolutional neural networks, pp. 51–58 (2019). https://doi.org/10.1145/3375959.3375988
25. Wang, W., Li, Y., Wang, X., Liu, J., Zhang, X.: Detecting android malicious apps and categorizing benign apps with ensemble of classifiers. Futur. Gener. Comput. Syst. **78**, 987–994 (2018)
26. Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: MINILM: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS 2020. Curran Associates Inc., Red Hook (2020)

An Interesting Property of Random Forest Distances with Respect to the Curse of Dimensionality

Manuele Bicego(✉) and Ferdinando Cicalese

University of Verona, Verona, Italy
{manuele.bicego,ferdinando.cicalese}@univr.it

Abstract. Random forest distances represent a powerful class of data-dependent similarity measures whose usefulness has been shown in many different scenarios. In this paper, we discuss an interesting property of these measures with respect to the curse of dimensionality, i.e., the set of problems that may arise when the feature space is too large with respect to the number of available objects. Starting from a recent theoretical characterization of two RF-distances defined on an ensemble of Extremely Randomized Trees (ERT), we provide some empirical evidence that such distances are indeed robust to the curse of dimensionality, improving their performances when increasing the dimensionality of the space. Further, we empirically show that this behavior is not restricted to the ERT-based RF-distances, but in general, it also holds with alternative training schemes.

Keywords: Random Forest distances · Curse of dimensionality · RatioRF

1 Introduction

Random Forests (RF) [13] represent successful classification and regression tools, based on an ensemble of decision trees. In recent years, RFs have also been successfully exploited to derive meaningful measures of (dis)similarity [4], to be employed in different types of contexts, like nearest-neighbor classification, distance-based anomaly detection, and mostly, Random Forest Clustering [4,8,21,24,26]. In a RF-distance, the main idea is that it is possible to measure the similarity between two objects by looking at the way they traverse each tree of the forest. For example, in the simplest and most used RF-distance defined by Breiman [13,21], two objects are highly similar if, for many of the trees of the forest, they fall in the same leave, since they are answering in the same way to all the tests found in the root-to-leave path. This concept has be refined along different directions, leading to several definitions of RF-based similarity measures [4,8,24,26].

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 188–198, 2025.
https://doi.org/10.1007/978-3-031-80507-3_19

Typically, to derive a RF-distance, we follow two steps: in the first, we learn a forest using the available data; in the second, we exploit the trained forest to define the similarity for a pair of objects x, y, by using the information contained in the paths x, y traverse in each tree. For what concerns the training, a widely adopted solution is to use *Extremely Randomized Trees* (ERT) [16], a class of decision trees in which randomization is taken to the extreme: in every node, the test is defined by choosing a random feature and a random threshold in the domain of that feature. This training scheme has been largely and successfully employed for RF-distance computation [4,9,24], being an unsupervised, simple, and efficient way to derive a forest, is shown to outperform alternatives even in supervised scenarios [23]. Recently, a theoretical justification of the good behavior of some ERT-based RF-distances has been proposed in [10]: in particular, assuming there exists a "true" distance, and assuming that there is a proper representation (i.e., a vectorial representation of the objects which satisfies the *Compactness Hypothesis* formulated by Arkadev and Braverman in 1967 [3]), [10] shows that near objects in the true distance are, with high probability, also near in the RF-distances computed with ERT forests. In the derivation of the theorems in [10], the assumption is that the tests on the same root-to-leaf path are on distinct features: if not making this restricting assumption and assuming that features can be reused several times on the same root-to-leaf path, [10] suggested a worst approximation guarantee of the RF-distance with respect to the true distance.

In this paper, we show another interesting property of these RF-distances. We started from the following reinterpretation of the findings in [10]: in problems with few features, the probability of re-using the same feature in different tests of the tree is higher than in problems with many features, since in each node of an ERT the feature on which to compute the split is randomly chosen among the available features. Therefore, we may argue that having more features would permit us to get a better approximation of the RF-distances with respect to the true distance. This intriguing property would be clearly in contrast with the Curse of Dimensionality, which says that in too high dimensional spaces, measures of similarity become meaningless: due to the increasing sparsity of the data, under reasonable assumptions, the ratio between the distances of the nearest and farthest neighbors tends to 1 for a variety of distance functions (the so-called concentration effect) [1,22]. In this paper, we empirically investigate this issue, providing some evidence that this appealing behavior for RF-distances is actually true, i.e., a better behavior is obtained when using more features.

In particular, we show our point with classification tests and *feature curves*, namely curves measuring the generalization error of a classifier exploiting a RF-distance defined on a varying number of features. We used the Nearest Neighbor rule, measuring the generalization error with the Leave One Out error protocol. We started from some datasets for which the feature curves, computed using the classic Euclidean Distance, show the expected curse of dimensionality behavior (i.e., the error starts to increase after a certain critical number of features). We then compute on the same datasets the feature curves using some well-kwown RF-distances, showing that instead the generalization error keeps on having

a decreasing trend also when adding more and more features. Furthermore, we provide evidence of two other interesting facts: i) this behavior is not restricted to the distances for which the theorems in [10] hold, but also for other RF-distances; ii) the behavior mainly persists even if we use training schemes alternative to ERTs, such as those typically employed for clustering [9,21].

The rest of the paper is organized as follows: in Sect. 2 we recap the main ideas behind Random Forest distances, whereas in Sect. 3 we summarize the issues related to the curse of dimensionality, especially in the distance case. Then, Sect. 4 contains the main empirical findings, and, finally, Sect. 5 concludes the paper with some discussion.

2 Random Forest Distances

Random Forest distances represent powerful and flexible data dependent measure of similarity [4], shown to be very useful in different tasks [8,9,24,26], also in the presence of missing data [11]. Typically, a RF-distance is defined in two steps: i) an RF is trained on the available data; ii) a distance between two objects is defined through the trained RF, typically by making the two objects traverse all the trees of the trained Forest, and comparing the answers they provide.

Let us summarize here the two distances employed in [10]. Given an object x, and a tree t, let us call $\ell_t(x)$ the leaf reached by the object x after traversing the tree t. Let us also denote as $P_t(x)$ the *path* of x from the root to its leaf. The first measure, which we call *Shi*, is the RF distance introduced by Breiman in his seminal paper [13] and then employed by Shi and colleagues for Random Forest Clustering [21]. The distance simply counts in how many trees, over the total number of trees, two objects do not fall in the same leave:

$$d_{Shi}(x,y) = \sqrt{\frac{1}{T}\sum_t (1 - \mathrm{I}(\ell_t(x), \ell_t(y)))} \tag{1}$$

where T represents the number of trees in the forest, and $\mathrm{I}(a,b)$ is the indicator function:

$$\mathrm{I}(a,b) = \begin{cases} 1 & \text{if} a = b \\ 0 & \text{if} a \neq b \end{cases} \tag{2}$$

The second measure is the recent RatioRF [8], which compares two objects on the basis of the answers they provide to all the tests contained in the two paths $P_t(x)$ and $P_t(y)$. In more detail, calling S_t^x the set of tests in the path $P_t(x)$ of x, we define as S_t^{xy} the set of all tests in the two paths, $S_t^{xy} = S_t^x \cup S_t^y$, and A_t^{xy} as the set of tests in S_t^{xy} on which x and y agree. The RatioRF distance is then defined as

$$d_{RRF}(x,y) = \sqrt{1 - \frac{1}{T}\sum_t \frac{|A_t^{xy}|}{|S_t^{xy}|}} \tag{3}$$

For these two distances, results in [10] show that when i) the Compactness Hypothesis [3] holds, i.e. the problem representation is such that similar objects

have close representations (i.e. they are close for most of the features), and ii) the forest is composed by Extremely Randomized Trees [16], then there exists a constant c such that if two objects are ϵ-close in the true distance, then with high probability they are $(c \cdot \epsilon)$-close in the RF-distances.

3 Curse of Dimensionality

Curse of dimensionality [2] can be defined as "the severe difficulty that can arise in spaces of many dimensions" [12]. This phenomenon, first spotted by Bellman in [6], has gained a renewed interest in recent years, mainly due to the surprising findings in the neural network community related to the so-called double descent [5,19]. In a few words, the curse of dimensionality represents a set of problems that may arise when employing PR/ML tools in problems with too many features with respect to the training objects. A dual (and more general) version of the phenomenon postulates that the Curse of Dimensionality may arise when the model is *too complex* with respect to the number of training objects, like for example, when employing a polynomial model with a too large degree – within this more general view, the number of features is no more than a measure of "complexity" of a model.

Among the different problems that may occur, let us cite two examples: i) when increasing the dimension, keeping fixed the number of objects, the space becomes almost empty, making density estimation impossible; ii) a model trained in a too high dimensional space –or, more intuitively, a too complex model– can be easily overtrained, not being able to generalize. Also distances have been largely studied with respect to the curse of dimensionality: in this case, the main result is the so-called "concentration effect" [1,22], which says that a distance that depends on too many independent features – i.e. a distance in a too high dimensional space – is almost constant. Some authors [7,17,20] also discussed the effects of this problem on the computation of the Nearest Neighbor (or the K-neaerest Neighbor) in high dimensional spaces, postulating the conditions under which such computation is meaningful. These aspects may affect the generalization capabilities of distance-based classifiers (like K-Nearest Neighbor): luckily, the concentration effect in practical cases is not as severe as the theory says, the generalization depending on the intrinsic dimensionality of the dataset [18], and, crucially, also on the distance. In this respect, [15] showed that the concentration does not derive from the finiteness of the dataset, but is an intrinsic property of the distance; in this paper, we provide some more empirical findings on this aspect, showing that RF-distances are robust with respect to the dimension of the feature space.

4 Experiments

In this section our empirical analysis is provided. We first introduce the experimental details, followed by results, comments and extensions.

Table 1. Summary of the employed Datasets

Name	Source	# objects	# features	# classes
ACSF1	UCR-TS (ACSF1)	200	1460	10
BeetleFly	UCR-TS (BeetleFly)	40	512	2
BirdChicken	UCR-TS (BirdChicken)	40	512	2
Gait	UCI-ML (Gait Classification)	48	321	16
Gastro-WL	UCI-ML (Gastrointestinal Lesions - White Light)	76	698	3
Gastro-NBI	UCI-ML (Gastrointestinal Lesions - Narrow Band Imaging)	76	698	3
HouseTwenty	UCR-TS (HouseTwenty)	159	2000	2
Herring	UCR-TS (Herring)	128	512	2
ORL-2Sub	Kaggle (ORL face dataset)	20	4096	2

4.1 Experimental Details

As we have seen in Sect. 3, the curse of dimensionality is defined as the set of problems that may occur when *the number of features is too large with respect to the number of objects*. Thus, to observe this effect, we searched for some problems with a relatively small number of objects and a remarkably high number of features: this is a very common scenario in biomedical domains, where few patients are typically characterized by a very large number of features (think, for example, to spectra or expression data). We also employed some datasets from sequence benchmarks, since in typical scenarios like the UCR Time series classification Archive [14] all sequences have the same length, and can be then considered as vectors – actually in such scenarios all the baseline errors are computed with the Nearest Neighbor rule and the Euclidean Distance. All the datasets are summarized in Table 1, where in the "source" column UCI-ML indicates the UCI Machine Learning Repository[1] CVR-TS the UCR Time Series Classification Archive[2] and Kaggle the Kaggle repository[3] All datasets were used as provided, except for the ORL-2Sub; in this case, the images have been resized to 64×64 and vectorized, and we used only subject 1 and subject 36, representing the most difficult pair of subjects to be classified, according to a LOO nearest neighbor evaluation.

To estimate the feature curves, we compute the Leave One Out error of the nearest neighbor classifier which uses in input the distance. LOO errors have been computed with the number of features n equal to 4, 8, 16, 32, ... and so on up to the whole dimension of the dataset. For every such choice of n, n random features have been extracted and used. In the subsequent step (i.e. when

[1] https://archive.ics.uci.edu/.
[2] https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.
[3] https://www.kaggle.com/datasets.

extracting $2n$ features), the previous set has been maintained (i.e., the set with $2n$ features is obtained by adding n novel random features to the existing set) – this being in line with classical settings in learning curves [25]. RF distances have been computed using forests with 100 trees, each built on a random 50% of the dataset, and grown until its maximum depth. The whole procedure has been repeated 500 times, and averaged results (with standard errors of the mean) are computed. To ensure a fair comparison among distances, the same random selection of features has been used for all distances.

4.2 Results

As a preliminary result, we show in Fig. 1 the feature curves computed by using the Euclidean Distance. In this case, it is evident that the NN based on Euclidean distance suffers from the curse of dimensionality, showing the classical U-shaped curve: at the beginning, adding more features is useful, but after a certain level, it makes the classifier more confused.

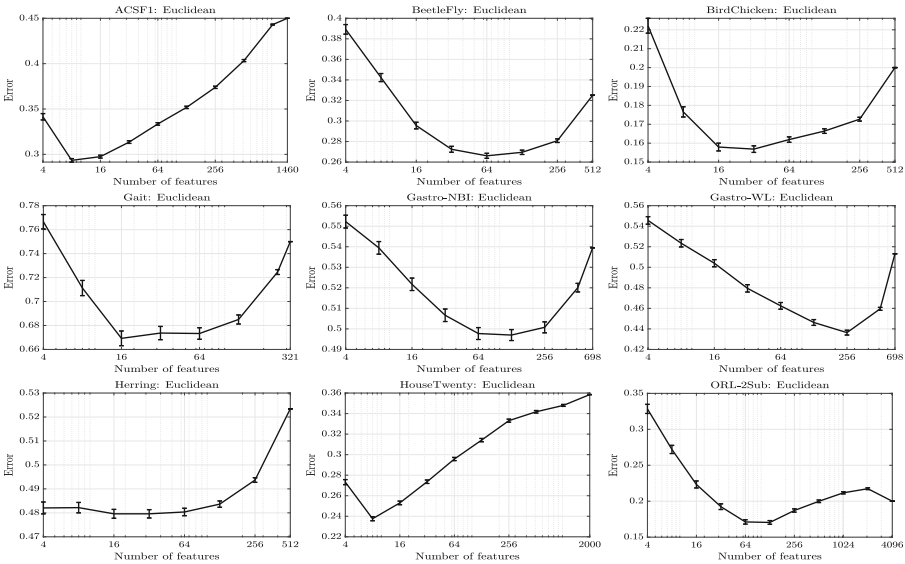


Fig. 1. Feature curves with Euclidean Distances

Then we computed the curves for the two RF-distances considered in the theoretical study of [10], namely the Shi distance and the RatioRF distance. Results are shown in Fig. 2: it seems evident that the curse of dimensionality is not an issue anymore, as the generalization error appears always to decrease when increasing the dimensionality of the dataset. These results confirm and extend the theoretical findings of [10], which hold under the restricting assumptions of the presence of a "proper representation" and a "true" distance: with

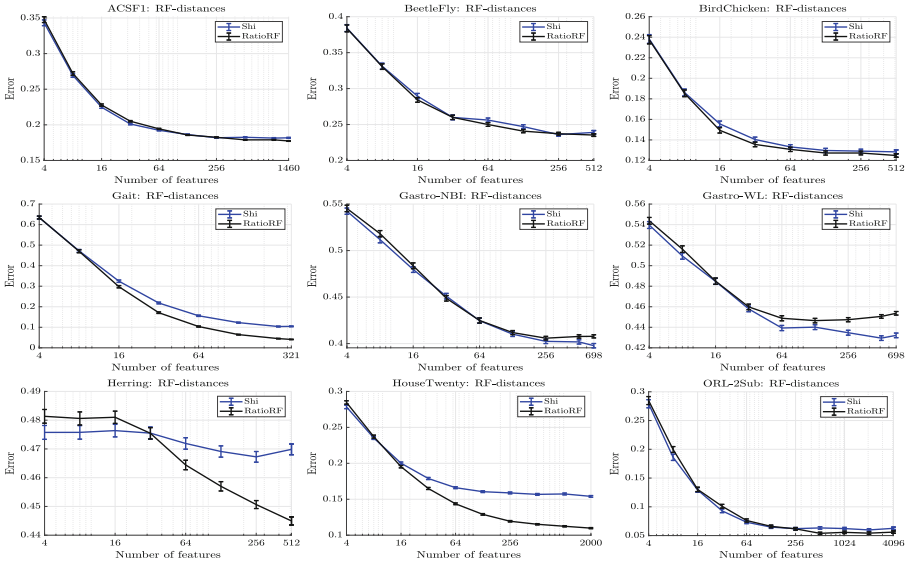


Fig. 2. Feature curves with RatioRF and Shi distances

more features, we have a better approximation of the true distance with the RF-distance, which, empirically, corresponds to a better generalization of a classifier exploiting it.

Let us extend the scope of the empirical analysis: theorems in [10] have been shown for two RF-distances (Shi and RatioRF) built on forests trained with the ERT paradigm, and one may wonder if the good behavior with respect to the curse of dimensionality depends on the particular distance and training employed, or if it generalizes to other RF-distances and other training schemes. To investigate this aspect, we first repeated the same set of experiments using RF-distances alternative to Shi and RatioRF, computed again on ERT ensembles: the "Zhu2" distance, the second variant introduced in [26] (called *ClustRF-Strct-Unfm* in such paper), the "Ting" measure, a mass-based Random Forest distance defined in [24], and the "Aryal" measure, a more recent RF-distance defined in [4] which implements an extension of the class of m_p distances.

Then, we repeated the experiments with an alternative training scheme, the so-called "negative-sampling" method, which represents the first and most classic solution for computing RF distances for RF-clustering [9,21,26]. Within this paradigm, a classification forest is trained on a two classes problem: one class contains the training points, the second contains a set of synthetically generated points, obtained by random sampling from the product of empirical marginal distributions (to remove the dependency between features). The results for alterna-

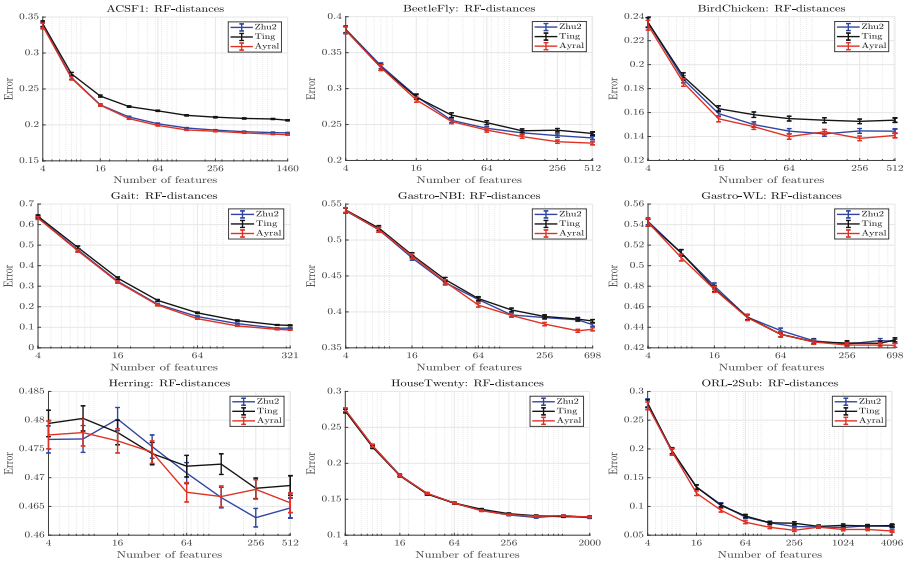


Fig. 3. Feature curves with other RF distances, defined on ERT forests.

tive distances, based on ERT forests, are reported in Fig. 3, whereas the results with the negative-sampling training schemes are shown in Figs. 4, for all RF-distances (in this case, due to the high computational requirements of this training scheme, we repeated the experiments only 100 times). For what concerns the ERT trained distances, we can observe that also in this case we have a good behavior of generalization errors, which do not show the classical U-shape of curse of dimensionality. When changing the training scheme, we can observe a generally good trend, even if, in some cases, not as good as when trained with ERT (see, for example, the ACSF1 and the Herring dataset).

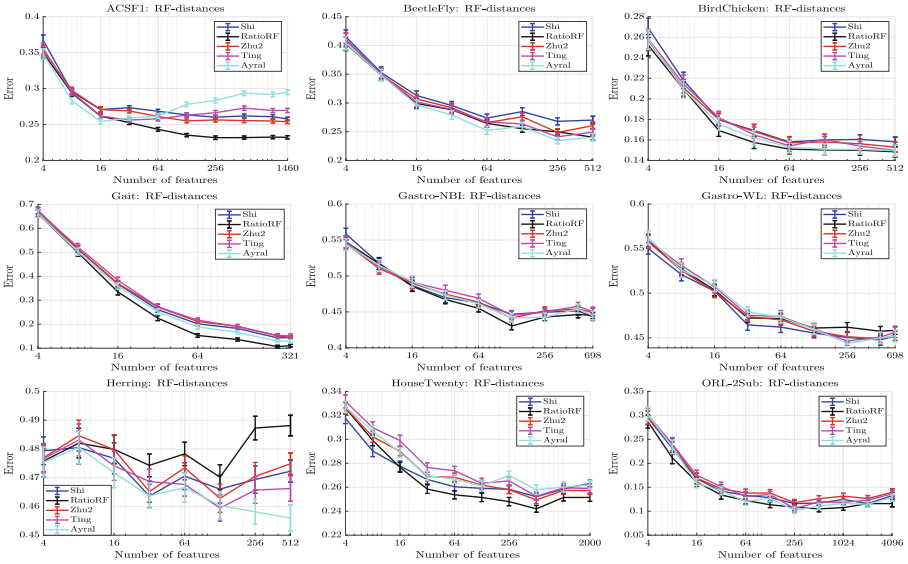


Fig. 4. Feature curves with all RF distances with the "Sample Negative" training scheme.

5 Discussion and Conclusion

In this paper, we provided some empirical evidence that RF-distances are not affected by the curse of dimensionality, especially if trained with the ERT scheme. This represents another empirical confirmation of the goodness of these distances in facing ML/PR problems. From a methodological perspective, why should RF-distances be less prone to the curse of dimensionality? The first explanation derives from the investigation of [10], which focused on showing that the distance computed by the forest does not diverge too much from the actual distance (assumed to be well represented in the data), and in fact indicating that high dimensionality data are more likely to guarantee this. More than this, our intuition is that the natural function of a decision tree is to separate objects, hence, to possibly magnify the distance even between very similar objects. Quantitatively, a few tests with different result over a reasonably short tree (compared to a possibly very high total number of features) translate into a significant difference. In this sense, a decision tree-based distance is less likely to become "uniform", i.e., it is less prone to suffer from the concentration effect.

Acknowledgments. M.B. would like to thank Tom Viering (TUDelft) for suggesting the connection between the work in [10] and the curse of dimensionality.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Proceedings of the International Conference on Database Theory (ICDT), pp. 420–434 (2001)
2. Altman, N., Krzywinski, M.: The curse (s) of dimensionality. Nat. Methods **15**(6), 399–400 (2018)
3. Arkadev, A.G., Braverman, E.M.: Teaching computers to recognize patterns. Transl. from the Russian by W. Turski and JD Cowan. Academic (1967)
4. Aryal, S., Ting, K., Washio, T., Haffari, G.: A comparative study of data-dependent approaches without learning in measuring similarities of data objects. Data Min. Knowl. Discov. **34**(1), 124–162 (2020)
5. Belkin, M., Hsu, D., Ma, S., Mandal, S.: Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proc. Natl. Acad. Sci. **116**(32), 15849–15854 (2019)
6. Bellman, R.: Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton (1961)
7. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Proceedings of the International Conference on Database Theory (ICDT), pp. 217–235. (1999)
8. Bicego, M., Cicalese, F., Mensi, A.: RatioRF: a novel measure for random forest clustering based on the Tversky's ratio model. IEEE Trans. Knowl. Data Eng. **35**(1), 830–841 (2023)
9. Bicego, M., Escolano, F.: On learning random forests for random forest-clustering. In: Proceedings of the International Conference on Pattern Recognition (ICPR), pp. 3451–3458. IEEE (2021)
10. Bicego, M., Cicalese, F.: On the good behavior of extremely randomized trees in random forest-distance computation. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), pp. 645–660 (2023)
11. Bicego, M., Cicalese, F.: Computing random forest-distances in the presence of missing data. ACM Trans. Knowl. Discov. Data **18**(7) (2024)
12. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
13. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
14. Dau, H.A., et al.: The UCR time series archive. IEEE/CAA J. Autom. Sinica **6**(6), 1293–1305 (2019)
15. François, D., Wertz, V., Verleysen, M.: The concentration of fractional distances. IEEE Trans. Knowl. Data Eng. **19**(7), 873–886 (2007)
16. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006)
17. Hinneburg, A., Aggarwal, C.C., Keim, D.A.: What is the nearest neighbor in high dimensional spaces? In: Proceedings of the International Conference on Very Large Data Bases, pp. 506–515 (2000)
18. Kpotufe, S.: k-NN regression adapts to local intrinsic dimension. In: Advances in Neural Information Processing Systems, vol. 24 (2011)
19. Loog, M., Viering, T., Mey, A., Krijthe, J.H., Tax, D.M.: A brief prehistory of double descent. Proc. Natl. Acad. Sci. **117**(20), 10625–10626 (2020)
20. Radovanovic, M., Nanopoulos, A., Ivanovic, M.: Hubs in space: popular nearest neighbors in high-dimensional data. J. Mach. Learn. Res. **11**(sept), 2487–2531 (2010)

21. Shi, T., Horvath, S.: Unsupervised learning with random forest predictors. J. Comput. Graph. Stat. **15**(1), 118–138 (2006)
22. Talagrand, M.: A new look at independence. Ann. Probab. 1–34 (1996)
23. Ting, K.M., Zhu, Y., Zhou, Z.H.: Isolation kernel and its effect on SVM. In: Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD), pp. 2329–2337 (2018)
24. Ting, K., Zhu, Y., Carman, M., Zhu, Y., Zhou, Z.H.: Overcoming key weaknesses of distance-based neighborhood methods using a data dependent dissimilarity measure. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1205–1214 (2016)
25. Viering, T., Loog, M.: The shape of learning curves: a review. IEEE Trans. Pattern Anal. Mach. Intell. **45**(6), 7799–7819 (2022)
26. Zhu, X., Loy, C., Gong, S.: Constructing robust affinity graphs for spectral clustering. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, CVPR 2014, pp. 1450–1457 (2014)

Author Index

A. Torsello et al. (Eds.): S+SSPR 2024, LNCS 15444, pp. 199–200, 2025.
https://doi.org/10.1007/978-3-031-80507-3