POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Machine Learning-enhanced Refinement and Agglomeration Algorithms for Polytopal Finite Element Methods

Doctoral programme in
Mathematical Models and Methods in Engineering

Doctoral Dissertation of: **Enrico Manuzzi**

Advisor: Prof. Paola Francesca Antonietti
Chair of the Doctoral Program: Prof. Michele Correggi
Department of Mathematics

2023 - Cycle XXXV

Abstract

Many geophysical and engineering applications, such as fluid-structure interaction, crack and wave propagation problems, and flow in fractured porous media, are characterized by a strong complexity of the physical domain, possibly involving thousands of fractures, heterogeneous media, moving geometries and complex topographies. Whenever "classical" Finite Element Methods (FEMs) are employed to discretize the underlying differential model, the process of generating and handling the computational mesh can be the bottleneck of the whole simulation, as computational grids can be composed only of tetrahedral, hexahedral, or prismatic elements. To overcome this limitation, many numerical methods that support computational meshes composed of general polygonal and polyhedral (polytopal, for short) elements have been developed in the last decade, such as for example the Virtual Element Method (VEM), the Polytopal Discontinuous Galerkin method (PolyDG), the mimetic finite differences method, the hybridizable discontinuous Galerkin method and the Hybrid High-Order method. However, since elements may have any shape, there are no well-established strategies to efficiently handle polytopal mesh refinement, i.e., partitioning mesh elements into smaller elements to produce a finer grid, and agglomeration, i.e., merging mesh elements to obtain coarser grids. Refinement is used to adaptively construct the grid in order to improve the accuracy of the solution, while agglomeration is used to reduce the number of degrees of freedom or in combination with multigrid solvers, which exploit a hierarchy of grids with different resolutions to accelerate the convergence of iterative algebraic algorithms. In order to perform these operations effectively, it is therefore extremely important to preserve the geometrical structure and quality of the initial grid, at a low computational cost. In this thesis, we propose to use Machine Learning (ML) based strategies to tackle the open problems of performing effectively mesh refinement and agglomeration. In two dimensions, we develop new strategies to handle polygonal grids refinement based on Convolutional Neural Networks (CNNs). We show that CNNs can be successfully employed to identify the "shape" of a polygonal element correctly so that ad-hoc refinement criteria can be applied. In this way, CNNs can be used to enhance existing refinement strategies, at a low online computational cost. In three dimensions, we extend the two-dimensional framework for mesh

refinement, combining CNNs also with the k-means clustering algorithm, to partition the points of the polyhedron to be refined, in order to make the approach more robust with respect to unstructured grids. In order to deal with the agglomeration of polygonal grids, we propose the use of Graph Neural Networks (GNNs) to partition the connectivity graph of mesh elements. GNNs can naturally and effectively process both the graph and the geometrical information, featuring strong generalisation capabilities and fast inference. The effectiveness of the proposed strategies is demonstrated in terms of quality metrics, mesh complexity, computational cost and performance when applied to VEMs, PolyDG methods and multigrid solvers. Finally, as a complementary contribution to this work, we explore the use Variational Physics-Informed Neural Networks (VPINNs) to directly solve the one-dimensional Helmholtz impedance problem, which is a mesh-less approach.

Keywords: Convolutional Neural Networks, Graph Neural Networks, K-means clustering, Virtual Element method, Polytopal Discontinuous Galerkin method, Multigrid solvers.

Sommario

Molte applicazioni geofisiche e ingegneristiche, come problemi d'interazione fluido-struttura, di propagazione di fessure e onde o di flusso di un fluido in mezzi porosi fratturati, sono caratterizzate da una forte complessità del dominio fisico, che può coinvolgere migliaia di fratture, mezzi eterogenei, geometrie mobili e topografie complesse. Ogni volta che si utilizzano approcci classici basati sugli elementi finiti per discretizzare il modello differenziale sottostante, il processo di generazione e gestione della griglia di calcolo può essere il collo di bottiglia dell'intera simulazione, poiché tali mesh possono essere composte solo da elementi tetraedrici, esaedrici o prismatici. Per ovviare a questa limitazione, nell'ultimo decennio sono stati sviluppati molti metodi numerici che supportano griglie computazionali composte da elementi poligonali e poliedrici generici (abbreviato politopali), come il metodo degli Elementi Virtuali Politopali, il metodo di Galekin Discontinuo per griglie Politopali, il metodo delle differenze finite mimetiche, il metodo di Galerkin discontinuo ibridizzabile e il metodo Ibrido di Alto Ordine. Tuttavia, poiché gli elementi possono avere qualsiasi forma, non esistono strategie consolidate per gestire in modo efficiente il raffinamento delle mesh poliedriche, ossia la suddivisione degli elementi della griglia in elementi più piccoli per produrre una mesh più fine, e agglomerazione, ossia l'unione degli elementi della griglia per ottenere una mesh più lasca. Il raffinamento viene utilizzato per costruire in modo adattivo la griglia al fine di migliorare l'accuratezza della soluzione, mentre l'agglomerazione viene utilizzata per ridurre il numero di gradi di libertà o in combinazione con i solutori multi-griglia, che sfruttano una gerarchia di mesh con risoluzioni diverse per accelerare la convergenza di algoritmi algebrici iterativi. Per eseguire queste operazioni in modo efficace è quindi estremamente importante preservare la struttura geometrica e la qualità della griglia iniziale, ad un basso costo computazionale. In questa tesi proponiamo di utilizzare strategie basate sull'Apprendimento Automatico per affrontare i problemi aperti di eseguire efficacemente il raffinamento e l'agglomerazione delle griglie. In due dimensioni, sviluppiamo nuove strategie per gestire il raffinamento delle mesh poligonali basate su Reti Neurali Convoluzionali. Dimostriamo che le Reti Neurali Convoluzionali possono essere impiegate con successo per identificare correttamente la "forma" di un elemento poligonale, in modo da poter applicare criteri di raffinamento ad-hoc. In questo

modo, le Reti Neurali Convoluzionali possono essere utilizzate per migliorare le strategie di raffinamento esistenti, a un basso costo computazionale online. In tre dimensioni, estendiamo l'approccio bidimensionale per il raffinamento della mesh, combinando le Reti Neurali Convoluzionali anche con l'algoritmo k-means, per partizionare i punti del poliedro da raffinare, al fine di rendere l'approccio più robusto rispetto a griglie non strutturate. Per far fronte all'agglomerazione di mesh poligonali, proponiamo l'uso di Reti Neurali per Grafi per partizionare il grafo di connettività degli elementi della griglia. Le Reti Neurali per Grafi sono in grado di elaborare in modo naturale ed efficace sia il grafo che le informazioni geometriche, con una forte capacità di generalizzazione e una rapida inferenza di nuovi input. L'efficacia delle strategie proposte è dimostrata in termini di metriche di qualità, complessità della griglia, costo computazionale e prestazioni quando vengono applicate al metodo degli Elementi Virtuali Politopali, al metodo di Galekin Discontinuo per griglie Politopali e a solutori multi-griglia. Infine, come contributo complementare a questo lavoro, esploriamo l'uso delle Reti Neurali Variazionali Informate sulla Fisica del problema per risolvere direttamente il problema d'impedenza di Helmholtz unidimensionale, che è un approccio privo di mesh.

Parole chiave: Reti Neurali Convoluzionali, Reti Neurali per Grafi, algoritmo K-means, metodo degli Elementi Virtuali Politopali, metodo di Galekin Discontinuo per griglie Politopali, solutori multi-griglia.

Ringraziamenti

Vorrei ringraziare la mia relatrice Prof. Paola F. Antonietti, per avermi guidato e sostenuto in questi tre anni, per avermi sempre trattato con umanità e per aver sempre creduto in me, a volte più di quanto io stesso non abbia fatto. Ringrazio il Prof. Marco Verani e il Dr. Domenico S. Brunetto per avermi sostenuto nella mia prima esperienza d'insegnamento, ed avermi mostrato quanto questo possa essere appagante e divertente. Ringrazio il Prof. Ilario Mazzieri e il Dr. Michele Botti per aver avuto fiducia in me e avermi aiutato a dare voce alla mia ricerca. Ringrazio il Prof. Franco Dassi, che con la sua vitalità e dedizione ha sempre reso sempre piacevole lavorare assieme. Ringrazio i miei studenti Nicola, Gabriele e Luca, che con il loro duro lavoro hanno contribuito a questo lavoro di tesi. Auguro loro il meglio per il futuro.

Ringrazio il Prof. Jan S. Heasthaven e il suo gruppo di ricerca per avermi accolto all'École Polytechnique Fédérale de Lausanne. E' stata un'esperienza che mi ha arricchito molto sia scientificamente che umanamente. Ringrazio in particolare il Dr. Fernando J. P. Henriquez Barazza, con cui ho avuto la fortuna di lavorare fianco a fianco per i tre mesi che ho passato a Losanna. Seppure in così poco tempo, abbiamo stretto un'amicizia che credo vada ben oltre la collaborazione scientifica e spero di continuare a divertirmi e a lavorare con lui.

Ringrazio il Ministero dell'Università e della Ricerca (MUR), per aver parzialmente sostenuto questa lavoro con le borse PRIN numero 201744KLJL e 20204LN5N5 (titolare del fondo Prof. Paola F. Antonietti).

Ringrazio i miei amici di "Milano", che poi molti a Milano non si trovano più. Grazie ai ragazzi del Tender, per avermi accolto in ufficio come uno di famiglia. Sono tutte persone straordinarie e mi mancherà mangiare con loro in sala mella, sentirli esporre bizzarre teorie, festeggiare e arrampicare assieme. Un grazie speciale a Giuseppe, con cui ho cominciato questa avventura, per aver sempre sostenuto tutti incondizionatamente ed essersi fatto carico dei problemi degli altri, sia emotivi che pratici. Grazie a tutti i miei coinquilini, dal primo anno di triennale all'ultimo di dottorato, per avermi sopportato e condiviso con me sia gli spazi che bei momenti. Grazie a tutti gli amici fantastici che

ho incontrato durante l'università e che sono sempre rimasti al mio fianco. Con loro ho davvero capito di poter mettere radici anche lontano da casa. Un grazie speciale ad alcuni di loro che mi sono stati particolarmente vicini durante il dottorato. Grazie a Gilda e Ale, per essermi sempre stati accanto e aver gioito e pianto con me, anche quando potevo più prendere che dare, per avermi strappato a me stesso tutte le volte che mi sono chiuso, per avermi ospitato un'infinità di volte, per tutte le risate e i giochi da tavolo. Di amici come loro ne esistono pochi al mondo. Grazie a Lorenzo, mio amico e collega sia durante la triennale che durante il dottorato, per essere stato sempre positivo, aperto ed entusiasta, specialmente nei confronti di quello che non conosce, per aver sempre guardato a quanto poteva dare per aiutare i suoi amici, senza risparmiarsi, e mai a quanto poteva ricevere. Grazie ad Andre, per avermi accolto in casa durante il primo anno di dottorato e in terra straniera a Bruxelles. Spero di vivere altre mille follie assieme a lui, incluso il periplo con la canoa, e che il fuoco della sua intraprendenza non si spenga mai. Grazie al mio amico GGGiò per aver condiviso con me tante esperienze e passioni, dai concerti metal alle maratone di Jojo, dalla scrivania d'ufficio alle lamentele della proprietaria (come dimenticare quando esplose l'asse del water). E' stato un vero spasso. Grazie a Fabio, per avermi portato con se ovunque sia andato, dall'Abruzzo al Belgio e all'Olanda, per aver mantenuto viva la nostra amicizia nonostante la distanza con telefonate regolari, per avermi incluso nella sua famiglia di cui ha condiviso con me sia le cose belle e che i dispiaceri, per essere un piccolo treno di positività che a volte deraglia ma di sicuro non si ferma. Grazie a Borja, averlo incontrato mi ha indubbiamente cambiato. Grazie per avermi consigliato e ispirato, per i pomeriggi di videogiochi, per il Pata Negra, per avermi ospitato a Madrid e a Londra, per avermi aiutato a trovare la strada nel realizzare il mio modello della felicità, per avermi mostrato che anche io potevo fare il dottorato. Aveva ragione. Auguro a lui e a Raquel il radioso futuro insieme che meritano. Grazie a Nicola, con cui ho condiviso l'appartamento a Losanna, e che ho ammirato dal primo giorno sia come uomo di scienza che come amico, per come è in grado di tirare fuori il meglio dalle persone, semplicemente ascoltandole davvero, con umiltà e genuino interesse. Spero davvero tanto che le nostre strade continueranno ad incrociarsi. Grazie alla mia amica Camilla, sia per i momenti belli che per quelli brutti, ma soprattutto per avermi insegnato ad abbracciare tutte le emozioni che la vita ha da offrire.

Grazie ai miei amici di Cesena, con cui sono cresciuto, per aver sempre fatto il tifo per me, per aver sempre trattato le miei idee come se fossero vive e per esserci sempre voluti bene per come siamo. Un grazie speciale ad alcuni di loro per avermi fatto da faro quando non sapevo dove andare. A Porch, che conosco da tutta la vita. Per quanto la strada possa portarci in luoghi distanti o farci crescere in modo diverso, sarà sempre parte indissolubile

della mia famiglia. A Miki, che mi ha insegnato che l'arma più forte di un vero guerriero è la capacità di continuare a sorridere nelle difficoltà, non importa quante volte si cada. A Saso, per avermi dimostrato che il duro lavoro, l'intraprendenza e la fiducia nelle persone alla fine ripagano sempre.

Un grazie dal più profondo del mio cuore alla mia famiglia. Sono la colonna portante della mia vita e non saprei come fare senza di loro. A mio babbo Mario, che con la sua grandezza e nobiltà d'animo si è sempre fatto carico dei problemi di tutti con un sorriso. A mia mamma Cristiana, che con il suo modo solare e la sua positività ha sempre spazzato via l'oscurità che le difficoltà della vita possono portare. A mio fratello Riccardo, per avermi teso la mano quando ero troppo stanco e avermi consigliato col suo modo pratico e pulito, per aver cucinato per me quando non volevo mangiare. A mia sorella Beatrice, che con la sua bontà e leggerezza di cuore non ha mai esitato a farsi da parte per me ed è sempre riuscita a farmi ridere.

Un grazie a tutti i parenti, che mi hanno sostenuto durante questa esperienza e sono sempre stati orgogliosi di me, in particolare ai nonni. A Carlo, che anche se ci ha lasciati tempo fa e a volte mi manca, spero possa vedermi e guidarmi ancora. A Wilmen, che con la forza di un uragano ha sempre tenuto unita la famiglia, ricordandoci quanto sia bello e importante volersi bene. A Edda, che con la sua grande sensibilità si è sempre preoccupata prima degli altri che di se stessa.

Infine, anche se può sembrare strano, vorrei ringraziare me stesso, per non essermi mai arreso, per aver capito quanto è importante essere fieri di se stessi al di là dei propri successi e fallimenti, per non aver mai smesso di sognare e amare la matematica anche quando a volte mi sembrava impossibile.

Mi spiace davvero non poter rendere giustizia a tutti in queste righe, ma sappiate solo che vi ho pensato, dal primo all'ultimo. Quindi semplicemente grazie.

Contents

Introduction

Many applications in the fields of Engineering and Applied Sciences, such as fluid-structure interaction problems, flow in fractured porous media, crack and wave propagation problems, are characterized by a strong complexity of the physical domain, possibly involving moving geometries, heterogeneous media, immersed interfaces and complex topographies. Whenever classical Finite Element Methods (FEMs) are employed to discretize the underlying differential model, the process of grid generation can be the bottleneck of the whole simulation, as computational meshes can be composed only of tetrahedral, hexahedral, or prismatic elements. To overcome this limitation, in the last years there has been a great interest in developing FEMs that can employ general polygons and polyhedra as grid elements for the numerical discretizations of partial differential equations. We mention the mimetic finite difference method [32, 47, 48, 102], the hybridizable discontinuous Galerkin method [59–62], the Polyhedral Discontinuous Galerkin (PolyDG) method [10, 13, 18, 28, 50, 53, 99], the Virtual Element Method (VEM) [19, 30, 31, 33, 34, 36] and the Hybrid High-Order method [70–74]. This calls for the need to develop effective algorithms to handle polygonal and polyhedral (polytopal, for short) grids and to assess their quality (see e.g. [23]). For a comprehensive overview we refer to the monographs and special issues [19, 32, 36, 53, 74, 75] and the references therein.

Among the open problems, there is the issue of efficiently handling polytopal mesh refinement [37, 100, 111], i.e., partitioning mesh elements into smaller elements to produce a finer grid, and agglomeration, i.e., merging mesh elements to obtain coarser grids [16, 28, 56, 89, 124]. Mesh refinement can be used to obtain a finer approximation of the differential problem at hand, in order to improve the accuracy of the solution. On the contrary, mesh agglomeration can be used to obtain a coarser approximation of the differential problem at hand, in order to reduce the number of degrees of freedom and therefore the computational effort. These operations can be naturally performed in the context of polygonal and polyhedral grids, because of the flexibility in the definition of the shape of mesh elements. They can be used in combination to attain an optimal convergence of the employed numerical method, properly balancing the trade-off between accuracy of the solution and computational effort related to the number of degrees of freedom. Appli-

cations also include generating a hierarchy of grids with different resolution for a domain of interest, that are at the basis of multigrid solvers [12, 15, 28, 29, 43, 45, 77, 91], to accelerate the converge of algebraic iterative solvers. Moreover, these operations can be exploited with domain decomposition techniques [7, 11, 84, 141] to obtain a meaningful decomposition of the original domain, given a suitable initial discretization.
As for standard triangular and quadrilateral meshes, during either refinement or agglomeration it is important to preserve the quality of the underlying mesh, since this might affect the overall performance of the method in terms of stability and accuracy. Indeed a suitable adapted mesh may allow to achieve the same accuracy with a much smaller number of degrees of freedom when solving the numerical problem, hence saving memory and computational power. However, since in such a general framework mesh elements may have any shape, there are no well established strategies to achieve effective refinement or agglomeration with a fast, robust and simple approach, contrary to classical tetrahedral, hexahedral and prismatic meshes. Moreover, grid agglomeration is a topic quite unexplored, because it is not possible to develop such kind of strategies within the framework of classical FEMs.

In recent years there has been a great development of Machine Learning (ML) algorithms, a framework which allows to extract information automatically from data, to enhance and accelerate numerical methods for scientific computing [17, 98, 127–129, 131–133]. In this thesis, we propose to use ML-based strategies to efficiently handle polytopal mesh refinement and agglomeration, in order to fully exploit all of the benefits of polytopal finite element methods, such as geometrical flexibility, convergence properties and others. The core concept lies in learning the "shape" of mesh elements in order to perform the desired operations accordingly. Such a learning needs to be performed in an automatic and flexible way, because of the too high variability of the geometries of interest, tailoring the approach to a wide range of different possible situations, and avoiding, as much as possible, online geometric checks that are expensive to be carried out for large-scale applications where the under-laying grid might be composed of million of elements. Rather than simply trying to decide a priori criteria to perform refinement and agglomeration, which would inevitably result in poor performance or high computational cost due to the impossibility of capturing all of the possible situations, ML strategies exploit and process automatically the huge amount of available data to learn only the distribution of the features of interest for the application, leading to high performance and computational efficiency. By combining the a priori approach of classical numerical methods, with the a posteriori approach of ML-based strategies, it is possible to not only to boost existing algorithms but also to develop new algorithms capable to work in more general frameworks. In particular, we propose

the use of Deep Neural Networks [114], powerful function approximators, to exploit the information coming from artificially generated databases of meshes, in order to drive the processes of refinement and agglomeration. This approach is fully generalizable as it is not dependent of the differential problem under investigation, but depends only on the "shape" of the local mesh elements.

In the following, we provide an overview of the contents of each chapter of the thesis.

In Chapter 1 we introduce the numerical methods that employ general polygonal and polyhedral grid, that we employ as benchmark for the proposed refinement and agglomeration strategies. In particular, we will describe the VEM and the PolyDG method, with the latter being employed also in a multigrid framework. Applications include the advection-diffusion-reaction problem and the Stokes problem.

In Chapter 2 we introduce different ML techniques that can be employed for grid refinement and agglomeration. In particular, we will introduce Convolutional Neural Networks (CNNs) for the problem of image classification, related to mesh refinement, Graph Neural Networks (GNNs) for the problem of graph partitioning, related to mesh agglomeration, and the k-means algorithm in a general framework.

In Chapter 3 we propose new strategies to handle polygonal grids refinement based on CNNs, that are function approximatiors particularly well suited for image classification. Indeed, they have been successfully applied in many areas, especially computer vision [2, 83, 110, 114, 140]. By viewing mesh elements as images, where pixels represent volume units of the considered polytope, CNNs can be employed to correctly classify the "shape" of both polygons and polyhedra, without resorting to the explicit calculation of geometric properties, which may be expensive, especially in three dimensions. This information can then be exploited to design tailored strategies for different families of polygons and polyhedra [4, 6]. According to the classification, the most suitable refinement strategy is then selected among the available ones, therefore enhancing performance. The proposed approach has several advantages:

- it helps preserving structure and quality of the mesh, since it can be easily tailored for different types of elements;
- it can be combined with suitable user-defined refinement criteria;
- it is independent of the differential model at hand and of the numerical method employed for the discretization;
- the overall computational cost is kept low, since neural networks can be trained offline once and for all.

We test the proposed idea in two dimensions considering PolyDG methods and VEMs. We demonstrate that the proposed algorithms can greatly improve the performance of the discretization schemes both in terms of accuracy and quality of the underlying grids.

In Chapter 4 we propose new strategies based on ML techniques to handle polyhedral grid refinement, to be possibly employed within an adaptive framework, as a generalization to three dimensions of the approach proposed in Chapter 3. While in two dimensions an approach based solely on CNNs seems to be sufficient to handle the complexity of mesh elements [4], this is not the case in three dimensions [6], where the shape variability of polyhedra is too high to be captured withing a single database. This calls for the need of an unsupervised learning approach that, contrary to the supervised one which employs CNNs, does not require prior knowledge in form of labelled data to solve a new instance of the problem. In particular, we employ the k-means clustering algorithm [95, 115] to partition the points of the polyhedron to be refined. Learning a clustered representation allows to better handle situations where elements have no particular structure, while preserving the quality of the grid. This strategy is a variation of the well known Centroidal Voronoi Tessellation (CVT) [82, 117]. We also propose the use of CNNs to classify the shape of an element so that ad-hoc refinement criteria can be defined, as in the two dimensional case. This strategy can be used to enhance existing refinement strategies, including the k-means strategy. We test the proposed algorithms considering the VEM and the PolyDG method. We demonstrate that these strategies do preserve the structure and the quality of the underlaying grids, reducing the overall computational cost and mesh complexity.

In Chapter 5 we propose a geometrical deep learning-based approach to handle polygonal grids agglomeration via GNNs [85, 86, 94], that are architectures specifically meant to work with graph-structured data. GNNs can be viewed as the generalisation of CNNs, since the latter are designed to work on specific lattice graphs, while the former can applied to graph any sort of irregular sparsity pattern. The problem of mesh agglomeration can be re-framed as a graph partitioning problem, by exploiting the connectivity structure of the mesh. In particular, the graph representation of the mesh is obtained by assigning a node to each element of the mesh, and connecting with an edge the pair of nodes which are relative to adjacent elements in the original mesh. By exploiting such a representation, GNNs can be applied to solve a node classification problem, where each element is assigned to a cluster, which corresponds an element of the agglomerated mesh. GNNs can process naturally and simultaneously both the graph structure of mesh and the geometrical information that can be attached to the nodes, such the elements areas or their barycentric coordinates. This is not the case of other approaches such as METIS [106], a standard solver for graph partitioning which can process only the graph infor-

mation, or k-means, which can process only the geometrical information. The proposed GNN-based algorithms exploit an unsupervised training procedure, where parameters are set to minimize the expected value of the normalized cut, over a database of polygonal grids. Performance in terms of quality metrics is enhanced for ML strategies, with GNNs featuring a lower computational cost online. Such models also show a good degree of generalization when applied to more complex geometries, such as brain MRI scans, and the capability of preserving the quality of the grid. The effectiveness of these strategies is accessed also when applied to MG solvers in a PolyDG framework.

In Chapter 6, as a further development, we propose a preliminary work on the use of Variational Physics-Informed Neural Networks (VPINNs) to solve the one-dimensional Helmholtz impedance problem. In particular, neural networks are used to minimize the residual of the weak formulation of the Helmholtz problem. In this case, neural networks are employed to directly compute the solution of the differential problem, therefore substituting classical finite element schemes. This is a different approach to the one adopted so far in this thesis, where ML strategies were employed to enhance numerical methods. One of the main differences of this approach with respect to classical finite element schemes is the fact of being "mesh-less", as it is not required to perform a discretization of the physical domain. However, some analogies can be drawn since the neural network automatically induces a partition of the domain. Therefore, this work can be seen as a complementary contribution to the problem of handling polytopal meshes, in the attempt to highlight advances, disadvantages, similarities and synergies of both mesh-based and mesh-less approaches. The main reason to use artificial neural networks to tackle the Helmholtz problem resides in their powerful approximation properties for highly oscillatory functions, which are of particular interest for the application under investigation. Indeed, the required number of parameters of the network, in particular the number of layers, scales logarithmically with respect to the considered frequency. The main bottleneck of this approach is the expensive training of the network. We therefore analyze different training strategies, aiming at simplifying the under-laying optimization problem.

In Chapter 7 we summarize the achieved results and discuss further developments and open problems for future research.

1 | Numerical methods for polytopal grids

In this chapter, following [8, 20, 51, 66], we introduce numerical methods that employ general polygons and polyhedra as grid elements for the numerical discretizations of partial differential equations. In particular, we will describe the Virtual Element Method (VEM) and the Polytopal Discontinuous Galrkin (PolyDG) method, with the latter being employed also in a multigrid framework. Applications include the advection-diffusion-reaction problem and the Stokes problem.

The VEM was introduced in [30, 31] as a generalization of the Finite Element Method (FEM) that allows for very general polygonal and polyhedral meshes, which also includes non convex and very distorted elements. The VEM is not based on the explicit construction and evaluation of the basis functions, as standard FEM, but on a proper choice and use of the degrees of freedom in order to compute the operators involved in the discretization of the problem. The employed basis functions are virtual, since they follow a rigorous definition, include (but are not restricted to) standard polynomials but are not computed in practice. The accuracy of the method is guaranteed by the polynomial part of the virtual space. Using such approach introduces other potential advantages, such as exact satisfaction of linear constraints [67] and the possibility to build easily discrete spaces of high global regularity [14, 46].

The Discontinuous Galerkin (DG) method was first introduced in [130] and then proposed to deal with elliptic and parabolic problems [21, 26, 142]. As the discrete polynomial space can be defined locally on each mesh element, DG methods feature a high-level of intrinsic parallelism. The DG methods can be extended to computational grids characterized by polytopic elements [10, 13, 15, 18, 28, 50, 53, 99]. In particular, employing a bounding box approach for each element [50, 88], together with a proper discontinuity penalization, allow for polytopic elements that can be characterized by faces of arbitrarily small measure and possibly by an unbounded number of faces [15, 52].

Employing general polygonal e polyhedral grids allows for flexibility in the definition of

the element shape, which couples very well with the possibility of defining agglomerated meshes, a key ingredient for the development of multigrid algorithms [5, 12, 15, 28, 29, 43, 45, 55, 77, 91, 144]. Multigrid methods, such as the two-level, the V-cycle or the W-cycle, exploit a hierarchy of discretizations of the physical domain in order to accelerate the convergence of classical numerical methods for the solution of partial differential equations. In particular, through a practice known as relaxation, they perform a global correction of the fine grid solution approximation by solving the problem on a coarser grid. The definition of the subspaces associated with the agglomerated grids is straightforward, since inter-element continuity is not required. Agglomeration must done with care, since it may lead to coarser grids with an increasing number of faces and this might affect the conditioning of the coarser components of the solver and the overall efficiency.

In order to effectively exploit of all the benefits of the described algorithm, such as the geometrical flexibility during mesh generation, the capability of handling hanging nodes naturally and the convergence properties, it is therefore important to design proper refinement and agglomeration strategies for polytopal grids, capable of preserving the grid quality at a low computational cost.

1.1. The virtual element discretization of the diffusion-reaction problem

In this section we give a brief overview of the VEM in three space dimensions for the simple model problem of diffusion-reaction in primal form. More details on the method for this same model and formulation can be found in [1, 30, 31, 66] while extension to variable coefficients is presented in [33]. Here k will denote a positive integer number, associated to the "polynomial degree" of the virtual element scheme.

1.1.1. Notation

In the following, E will denote a polygon and P a polyhedron, while faces, edges and vertices will be indicated by f, e, and v respectively.

If P is a polyhedron in $\mathbb{R}^3$, we will denote by $\boldsymbol{x}_P, h_P$ and $|P|$ the centroid, the diameter, and the volume of P, respectively. The set of polynomials of degree less than or equal to s in P will be indicated by $\mathcal{P}_s(P)$. If $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$ is a multiindex, we will indicate by m_α the scaled monomial

$$m_\alpha = \left(\frac{x - x_P}{h_P}\right)^{\alpha_1} \left(\frac{y - y_P}{h_P}\right)^{\alpha_2} \left(\frac{z - z_P}{h_P}\right)^{\alpha_3}$$

and we will denote by $\mathcal{P}_s^{\text{hom}}(P)$ the space of scaled monomials of degree exactly (and no less than) s :

$$\mathcal{P}_s^{\text{hom}}(P) = \text{span}\{m_{\boldsymbol{\alpha}}, |\boldsymbol{\alpha}| = s\}$$

where $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \alpha_3$. The case of a polygon $E \subset \mathbb{R}^2$ is completely analogous.

A face f of a polyhedron is treated as a two-dimensional set, using local coordinates (x, y) on the face. Edges of polyhedra and polygons are treated in an analogous way as one-dimensional set.

1.1.2. The diffusion-reaction model problem

Let $\Omega \subset \mathbb{R}^3$ represent the domain of interest (that we assume to be a polyhedron) and let Γ denote a subset of its boundary, that we assume for simplicity to be given by a union of some of its faces. We denote by $\Gamma' = \partial\Omega/\Gamma$.

We consider the simple diffusion-reaction problem

$$\begin{cases} -\Delta u + u = f & \text{in } \Omega \\ u = r & \text{on } \Gamma \\ \dfrac{\partial u}{\partial n} = g & \text{on } \Gamma' \end{cases}$$

where $f \in L^2(\Omega)$ and $g \in L^2(\Gamma')$ denote respectively the applied load and Neumann boundary data, and $r \in H^{1/2}(\Gamma)$ is the assigned boundary data function. The variational form of our model problem reads

$$\begin{cases} \text{Find } u \in H^1_\Gamma(\Omega) \text{ such that} \\ \int_\Omega \nabla u \cdot \nabla v + \int_\Omega uv = \int_\Omega fv + \int_{\Gamma'} gv \quad \forall v \in H^1_{\Gamma,0}(\Omega), \end{cases}$$

where

$$H^1_\Gamma = \left\{v \in H^1(\Omega) : v|_\Gamma = r\right\}, \quad H^1_{\Gamma,0} = \left\{v \in H^1(\Omega) : v|_\Gamma = 0\right\}.$$

1.1.3. Virtual elements on polygons

We start by defining the virtual element space on polygons. Given a generic polygon E, let the preliminary virtual space

$$\widetilde{V}^k(E) = \left\{v \in H^1(E) \cap C^0(E) : v|_e \in \mathcal{P}_k(e)\ \forall e \in \partial E, \Delta v \in \mathcal{P}_k(E)\right\}$$

with e denoting a generic edge of the polygon.

For any edge e, let the points $\{\nu_e^i\}_{i=1}^{k-1}$ be given by the $k-1$ internal points of the Gauss-Lobatto integration rule of order $k+1$ on the edge. We now introduce three sets of linear operators from $\widetilde{V}^k(E)$ into real numbers. For all v in $\widetilde{V}^k(E)$:

O1: evaluation of $v(\nu) \quad \forall \nu$ vertex of E;

O2: evaluation of $v(\nu_e^i) \quad \forall e \in \partial E, i = \{1, 2, \ldots, k-1\}$;

O3: moments $\int_E v p_{k-2} \quad \forall p_{k-2} \in \mathcal{P}_{k-2}(E)$.

The following projector operator $\Pi_E^\nabla : \widetilde{V}^k(E) \to \mathcal{P}_k(E)$ will be useful in the definition of our space and also for computational purposes. For any $v \in \widetilde{V}^k(E)$, the polynomial $\Pi_E^\nabla v \in \mathcal{P}_k(E)$ is defined by (see [30])

$$\begin{cases} \int_E \nabla \left(v - \Pi_E^\nabla v\right) \cdot \nabla p_k = 0 \quad \forall p_k \in \mathcal{P}_k(E) \\ \text{for } k = 1: \quad \sum_{\nu \text{ vertex of } E} \left(v(\nu) - \Pi_E^\nabla v(\nu)\right) = 0 \\ \text{for } k \geq 2: \quad \int_E \left(v - \Pi_E^\nabla v\right) = 0. \end{cases}$$

Note that, given any $v \in \widetilde{V}^k(E)$ the polynomial $\Pi_E^\nabla v$ only depends on the values of the operators O1-O3. Indeed, an integration by parts easily shows that the values of the above operators applied to v are sufficient to uniquely determine $\Pi_E^\nabla v$ and no other information on the function v is required [30, 31].

We are now ready to present the two dimensional virtual space:

$$V^k(E) = \left\{ v \in \widetilde{V}^k(E) : \int_E v q = \int_E \left(\Pi_E^\nabla v\right) q \text{ for all } q \in \mathcal{P}_{k-1}^{\text{hom}}(E) \cup \mathcal{P}_k^{\text{hom}}(E) \right\}.$$

It is immediate to check that $\mathcal{P}_k(E) \subseteq V^k(E) \subseteq \widetilde{V}^k(E)$. Moreover the following lemma holds (the proof can be found in [1, 33]).

Lemma 1.1. *The operators O1-O3 constitute a set of degrees of freedom for the space* $V^k(E)$.

Finally note that, since any $p_k \in \mathcal{P}_k(E)$ can be written in an unique way as $p_k = p_{k-2} + q$, with $p_{k-2} \in \mathcal{P}_{k-2}(E)$ and $q \in \mathcal{P}_{k-1}^{\text{hom}}(E) \cup \mathcal{P}_k^{\text{hom}}(E)$, it holds

$$\int_E v p_k = \int_E v \left(p_{k-2} + q\right) = \int_E v p_{k-2} + \int_E \left(\Pi_E^\nabla v\right) q \tag{1.1}$$

for any $p_k \in \mathcal{P}_k(E)$. The first term on the right hand side above can be calculated recalling O3 while the second one can be computed directly by integration. This shows that we can actually compute $\int_E v p_k$ for any $p_k \in \mathcal{P}_k(E)$ by using only information on the degree

of freedom values of v.

1.1.4. Virtual elements on polyhedrons

Let Ω_h be a partition of Ω into non-overlapping and conforming polyhedrons. We start by defining the virtual space V^k locally, on each polyhedron $P \in \Omega_h$. Note that each face $f \in \partial P$ is a two-dimensional polygon. Let the following boundary space

$$\mathcal{B}^k(\partial P) = \left\{ v \in C^0(\partial P) : v|_f \in V^k(f) \text{ for all } f \text{ face of } \partial P \right\}.$$

The above space is made of functions that on each face are two-dimensional virtual functions, that glue continuously across edges. Recalling Lemma 1.1, it follows that the following linear operators constitute a set of degrees of freedom for the space $\mathcal{B}^k(\partial P)$

O4 evaluation of $v(\nu) \quad \forall \nu$ vertex of P

O5 evaluation of $v(\nu_e^i) \quad \forall e$ edge of $\partial P, i = \{1, 2, .., k-1\}$;

O6 moments $\int_f v p_{k-2} \quad \forall p_{k-2} \in \mathcal{P}_{k-2}(f), \forall f$ face of ∂P.

Once the boundary space is defined, the steps to follow in order to define the local virtual space on P become very similar to the two dimensional case. We first introduce a preliminary local virtual element space on P

$$\widetilde{V}^k(P) = \left\{ v \in H^1(P) : v|_{\partial P} \in \mathcal{B}^k(\partial P), \Delta v \in \mathcal{P}_k(P) \right\}$$

and the "internal" linear operators

- moments $\int_P v p_{k-2} \quad \forall p_{k-2} \in \mathcal{P}_{k-2}(P)$. We can now define the projection operator $\Pi_P^\nabla : \widetilde{V}^k(P) \to \mathcal{P}_k(P)$ by

$$\begin{cases} \int_P \nabla \left(v - \Pi_P^\nabla v \right) \cdot \nabla p_k = 0 \quad \forall p_k \in \mathcal{P}_k(P) \\ \text{for } k = 1 : \quad \sum_{\nu \underset{\text{of } P}{\text{vertex}}} \left(v(\nu) - \Pi_P^\nabla v(\nu) \right) = 0, \\ \text{for } k \geq 2 : \quad \int_P \left(v - \Pi_P^\nabla v \right) = 0. \end{cases}$$

An integration by parts and observation (1.1) show that the projection Π_P^∇ only depends on the operator values O4-O6. Therefore we can define the local virtual space

$$V^k(P) = \left\{ v \in \widetilde{V}^k(P) : \int_P v q = \int_P \left(\Pi_P^\nabla v \right) q \text{ for all } q \in \mathcal{P}_{k-1}^{\text{hom}}(P) \cup \mathcal{P}_k^{\text{hom}}(P) \right\}$$

The proof of the following lemma mimicks the two dimensional case, see for instance [6].

Lemma 1.2. *The operators O4-O6 constitute a set of degrees of freedom for the space* $V^k(P)$.

It is immediate to verify that $\mathcal{P}_k(P) \subseteq V^k(P)$, that is a fundamental condition for the approximation properties of the space. Moreover, again due to the observation above, the projection operator $\Pi_P^\nabla : V^k(P) \to \mathcal{P}_k(P)$ is computable only on the basis of the degree of freedom values O4-O6. In addition, by following the same identical argument as in (1.1) we obtain that $\int_P v p_k$ is computable for any $p_k \in \mathcal{P}_k(P)$ by using the degrees of freedom. Therefore also the L^2 projection operator $\Pi_P^0 : V^k(P) \to \mathcal{P}_k(P)$, defined for any $v \in V^k(P)$ by

$$\int_P \left(v - \Pi_P^0 v\right) q_k = 0 \quad \forall q_k \in \mathcal{P}_k(P),$$

is computable by using the degree of freedom values.

Finally, the global virtual space $V^k \subset H^1(\Omega)$ is defined by using a standard assembly procedure as in finite elements. We define

$$V^k = \left\{ v \in H^1(\Omega) : v|_P \in V^k(P) \text{ for all } P \in \Omega_h \right\}.$$

The associated (global) degrees of freedom are the obvious counterpart of the local ones introduced above, i.e.

- evaluation of $v(\nu) \quad \forall \nu$ vertex of $\Omega_h \backslash \Gamma$
- evaluation of $v\left(\nu_e^i\right) \quad \forall e$ edge of $\Omega_h \backslash \Gamma, i = \{1, 2, .., k-1\}$;
- moments $\int_f v p_{k-2} \quad \forall p_{k-2} \in \mathcal{P}_{k-2}(f), \forall f$ face of $\Omega_h \backslash \Gamma$;
- moments $\int_P v p_{k-2} \quad \forall p_{k-2} \in \mathcal{P}_{k-2}(P), \forall P \in \Omega_h$.

1.1.5. Discretization of the problem

We start by introducing the discrete counterpart of the involved bilinear forms. Given any polyhedron $P \in \Omega_h$ we need to approximate the local forms

$$a_P(v, w) = \int_P \nabla v \cdot \nabla w, \quad m_P(v, w) = \int_P vw.$$

We follow [30, 31]. We first introduce the stabilization form

$$s_P(v, w) = \sum_{i=1}^{N_{\text{dof}}^p} \Xi_i(v) \Xi_i(w) \quad \forall v, w \in V^k(P),$$

where $\Xi_i(v)$ is the operator that evaluates the function v in the ith local degree of freedom and N_{dof}^P denotes the number of such local degrees of freedom, see O4-O6. We then set, for all $v, w \in V^k(P)$,

$$a_P^h(v,w) = \int_P \left(\nabla\Pi_P^\nabla v\right)\cdot\left(\nabla\Pi_P^\nabla w\right) + h_P s_P\left(v-\Pi_P^\nabla v, w-\Pi_P^\nabla w\right),$$
$$m_P^h(v,w) = \int_P \left(\Pi_P^0 v\right)\left(\Pi_P^0 w\right) + |P| s_P\left(v-\Pi_P^0 v, w-\Pi_P^0 w\right).$$

The above bilinear forms are consistent and stable in the sense of [30]. The global forms are given by, for all $v, w \in V^k$,

$$a^h(v,w) = \sum_{P\in\Omega_h} a_P^h(v,w), \quad m^h(v,w) = \sum_{P\in\Omega_h} m_P^h(v,w).$$

Let now the discrete space with boundary conditions and its corresponding test space

$$V_\Gamma^k = \left\{v \in V^k : v|_\Gamma = r_I\right\}, \quad V_0^k = \left\{v \in V^k : v|_\Gamma = 0\right\}$$

where r_I is, face by face, an interpolation of r in the virtual space $V^k(f)$.

We can finally state the discrete problem

$$\begin{cases} \text{Find } u_h \in V_\Gamma^k \quad \text{such that} \\ a^h\left(u_h, v_h\right) + m^h\left(u_h, v_h\right) = \int_\Omega f_h v_h + \int_{\Gamma'} g_h v_h \quad \forall v_h \in V_0^k \end{cases}$$

where the approximate loading f_h is the L^2-projection of f on piecewise polynomials of degree k, and where g_h is the L^2-projection of g on piecewise polynomials (still of degree k) living on Γ'. Note that all the forms and operators appearing above are computable in terms of the degree of freedom values of u_h and v_h.

We close this section by recalling a convergence result. The main argument for the proof can be found in [30], while the associated interpolation estimates were shown in [120] for two dimensions and extended in [54] to the three dimensional case.

Let now $\{\Omega_h\}_h$ be a family of meshes, satisfying the following assumption. There exists a positive constant γ such that all elements P of $\{\Omega_h\}_h$ and all faces of ∂P are star-shaped with respect to a ball of radius bigger than or equal to γh_P; moreover all edges $e \in \partial P$, for all $P \in \{\Omega_h\}_h$ have length bigger than or equal to γh_P.

Theorem 1.1. *Let the above mesh assumptions hold. Then, if the data and solution is*

sufficiently regular for the right hand side to make sense, it holds

$$\|u - u_h\|_{H^1(\Omega)} \leq Ch^{s-1} \left(|u|_{H^s(\Omega_h)} + |f|_{H^{s-2}(\Omega_h)} + |g|_{H^{s-3/2}(\Gamma_h')} \right),$$

where $2 \leq s \leq k+1$*, the real* h *denotes the maximum element diameter size and the constant* C *is independent of the mesh size. The norms appearing on the right hand side are broken Sobolev norms with respect to the mesh (or its faces).*

The above result applies also if $1 \leq s < 2$, but in that case the regularities on the data f, g need to be modified. If the domain Ω is convex (or regular) then under the same assumptions and notations it also holds [33]

$$\|u - u_h\|_{L^2(\Omega)} \leq Ch^s \left(|u|_{H^s(\Omega_h)} + |f|_{H^{s-2}(\Omega_h)} + |g|_{H^{s-3/2}(\Gamma_h')} \right).$$

Finally, we note that an extension of the results to more general mesh assumptions could be possibly derived following the arguments for the two-dimensional case shown in [35].

1.2. Polytopal discontinuous Galerkin discretization of the advection-diffusion-reaction problem

In this section we introduce the DG method for polytopal grids for the solution of the advection-diffusion-reaction model problem, applied in two and three dimensions. For more details see [50, 51].

1.2.1. Preliminaries

For a Lipschitz domain $\omega \subset \mathbb{R}^d, d \geq 1$, we denote by $H^s(\omega)$ the Hilbertian Sobolev space of index $s \geq 0$ of real-valued functions defined on ω, endowed with the seminorm $|\cdot|_{H^s(\omega)}$ and norm $\|\cdot\|_{H^s(\omega)}$. Furthermore, we let $L_p(\omega), p \in [1, \infty]$, be the standard Lebesgue space on ω, equipped with the norm $\|\cdot\| L_p(\omega)$. Finally, $|\omega|$ denotes the d-dimensional Hausdorff measure of ω.

1.2.2. The advection-diffusion-reaction model problem

Let Ω be a bounded open polyhedral domain in $\mathbb{R}^d, d = 2, 3$, and let Γ signify the union of its $(d-1)$ dimensional open faces. We consider the advection-diffusion-reaction equation

$$\mathcal{L}u \equiv -\nabla \cdot (a \nabla u) + \mathbf{b} \cdot \nabla u + cu = f, \quad \text{in } \Omega, \tag{1.2}$$

where $c \in L_\infty(\Omega), f \in L_2(\Omega)$, and $\mathbf{b} := (b_1, b_2, \ldots, b_d)^\top \in [W^1_\infty(\Omega)]^d$. Here, $a = \{a_{ij}\}_{i,j=1}^d$ is a symmetric positive semidefinite tensor whose entries a_{ij} are bounded, piecewise continuous, real-valued functions defined on $\bar{\Omega}$, with

$$\xi^\top a(x)\xi \geq 0 \quad \forall \xi \in \mathbb{R}^d, \qquad \text{a.e.} \qquad x \in \bar{\Omega}.$$

Under the above hypothesis, (1.2) is termed a partial differential equation with nonnegative characteristic form.
We denote by $\mathbf{n}(x) = \{n_i(x)\}_{i=1}^d$ the unit outward normal vector to Γ at $x \in \Gamma$ and introduce the Fichera function $\mathbf{b} \cdot \mathbf{n}$ to define

$$\Gamma_0 = \left\{x \in \Gamma : \mathbf{n}(x)^\top a(x)\mathbf{n}(x) > 0\right\}$$
$$\Gamma_- = \left\{x \in \Gamma \backslash \Gamma_0 : \mathbf{b}(x) \cdot \mathbf{n}(x) < 0\right\}, \quad \Gamma_+ = \left\{x \in \Gamma \backslash \Gamma_0 : \mathbf{b}(x) \cdot \mathbf{n}(x) \geq 0\right\}.$$

The sets $\Gamma_-$and $\Gamma_+$are referred to as the inflow and outflow boundary, respectively. Note that $\Gamma_0 = \Gamma_0 \cup \Gamma_- \cup \Gamma_+$. If Γ_0 is nonempty, we subdivide it into two disjoint subsets Γ_D and Γ_N whose union is Γ_0, with Γ_D nonempty and relatively open in Γ, on which we consider the boundary conditions for (1.2):

$$u = g_\mathrm{D} \quad \text{on} \quad \Gamma_\mathrm{D} \cup \Gamma_-, \quad \mathbf{n} \cdot (a\nabla u) = g_\mathrm{N} \quad \text{on} \quad \Gamma_\mathrm{N}, \tag{1.3}$$

and also adopt the hypothesis that $\mathbf{b}\cdot\mathbf{n} \geq 0$ on Γ_N, whenever Γ_N is nonempty. Additionally, we assume that the following positivity hypothesis holds: there exists a positive constant γ_0 such that

$$c_0(x)^2 := c(x) - \frac{1}{2}\nabla \cdot \mathbf{b}(x) \geq \gamma_0 \quad \text{a.e. } x \in \Omega$$

The well-posedness of the boundary value problem (1.2),(1.3) has been studied in [101].

1.2.3. Finite element spaces

Let $\mathcal{T}$ be a subdivision of the computational domain Ω into disjoint open polygonal ($d = 2$) or polyhedral ($d = 3$) elements κ such that $\bar{\Omega} = \cup_{\kappa\in\mathcal{T}}\bar{\kappa}$ and denote by h_κ the diameter of $\kappa \in \mathcal{T}$; *i.e.*, $h_\kappa := \mathrm{diam}(\kappa)$. In the absence of hanging nodes/edges, we define the interfaces of the mesh $\mathcal{T}$ to be the set of $(d-1)$-dimensional facets of the elements $\kappa \in \mathcal{T}$. To facilitate the presence of hanging nodes/edges, which are permitted in $\mathcal{T}$, the interfaces of $\mathcal{T}$ are defined to be the intersection of the $(d-1)$-dimensional facets of neighbouring elements. In the case when $d = 2$, the interfaces of a given element $\kappa \in \mathcal{T}$ will always consist of line segments (($d-1$) dimensional simplices). For $d = 3$, we assume that

each interface of an element $\kappa \in \mathcal{T}$ may be subdivided into a set of co-planar triangles. With this in mind we use the terminology 'face' to refer to a $(d-1)$-dimensional simplex (line segment or triangle for $d = 2$ or 3 , respectively), which forms part of the boundary (interface) of an element $\kappa \in \mathcal{T}$. For $d = 2$, the face and interface of an element $\kappa \in \mathcal{T}$ necessarily coincide with each other, while in three-dimensions this may no longer be the case, since the boundary of a general polyhedron may consist of planar polygons which are not triangular.

As in [50], we assume that a sub-triangulation into faces of each mesh interface is given if $d = 3$, and denote by $\mathcal{E}$ the union of all open mesh interfaces if $d = 2$ and the union of all open triangles belonging to the subtriangulation of all mesh interfaces if $d = 3$. In this way, $\mathcal{E}$ is always defined as a set of $(d-1)$-dimensional simplices. Further, we write $\mathcal{E}_{\text{int}}$ to denote the union of all open $(d-1)$-dimensional element faces $F \subset \mathcal{E}$ that are contained in Ω, and let $\Gamma_{\text{int}} := \{x \in \Omega : x \in F, F \in \mathcal{E}_{\text{int}}\}$. Further assumptions on the class of admissible meshes will be outlined later on in Section 1.2.6.

Given $\kappa \in \mathcal{T}$, we write p_κ to denote the (positive) polynomial degree of the element κ, and collect the p_κ in the vector $\mathbf{p} := (p_\kappa : \kappa \in \mathcal{T})$. We then define the finite element space $S^{\mathrm{p}}_{\mathcal{T}}$ with respect to $\mathcal{T}$ and $\mathbf{p}$ by

$$S^{\mathrm{p}}_{\mathcal{T}} := \{u \in L_2(\Omega) : u|_\kappa \in \mathcal{P}_{p_\kappa}(\kappa), \kappa \in \mathcal{T}\},$$

where $\mathcal{P}_{p_\kappa}(\kappa)$ denotes the space of polynomials of total degree p_κ on κ. As in [50], we point out that the local elemental polynomial spaces employed within the definition of $S^{\mathrm{p}}_{\mathcal{T}}$ are defined in the physical coordinate system, without the need to map from a given reference or canonical frame. We define the broken Sobolev space $H^{\mathrm{s}}(\Omega, \mathcal{T})$ with respect to the subdivision $\mathcal{T}$ up to composite order s as follows

$$H^{\mathrm{s}}(\Omega, \mathcal{T}) = \{u \in L_2(\Omega) : u|_\kappa \in H^{s_\kappa}(\kappa) \quad \forall \kappa \in \mathcal{T}\}.$$

For $u \in H^1(\Omega, \mathcal{T})$, we define the broken gradient $\nabla_h u$ by $(\nabla_h u)|_\kappa = \nabla(u|_\kappa), \kappa \in \mathcal{T}$, which will be used to construct the forthcoming DG-FEM.

1.2.4. Trace operators

For any element $\kappa \in \mathcal{T}$, we denote by $\partial\kappa$ the union of $(d-1)$-dimensional open faces of κ. Then, the inflow and outflow parts of $\partial\kappa$ are defined as follows

$$\partial_-\kappa = \{x \in \partial\kappa, \quad \mathbf{b}(x) \cdot \mathbf{n}_\kappa(x) < 0\}, \quad \partial_+\kappa = \{x \in \partial\kappa, \quad \mathbf{b}(x) \cdot \mathbf{n}_\kappa(x) \geq 0\},$$

respectively, where $\mathbf{n}_\kappa(x)$ denotes the unit outward normal vector to $\partial\kappa$ at $x \in \partial\kappa$. Given $\kappa \in \mathcal{T}$, the trace of a function $v \in H^1(\Omega, \mathcal{T})$ on $\partial_-\kappa$, relative to κ, is denoted by v_κ^+. Further, if $\partial_-\kappa \backslash \Gamma$ is nonempty, then for $x \in \partial_-\kappa \backslash \Gamma$ there exits a unique $\kappa' \in \mathcal{T}$ such that $x \in \partial_+\kappa'$; with this notation, we denote by $v_\kappa^-$the trace of $v|_{\kappa'}$ on $\partial_-\kappa \backslash \Gamma$. Hence the upwind jump of the (scalar-valued) function v across a face $F \subset \partial_-\kappa \backslash \Gamma$ is denoted by

$$\lfloor v \rfloor := v_\kappa^+ - v_\kappa^-.$$

Next, we introduce some additional trace operators. Let κ_i and κ_j be two adjacent elements of $\mathcal{T}$ and let x be an arbitrary point on the interior face $F \subset \Gamma_{\text{int}}$ given by $F = \partial\kappa_i \cap \partial\kappa_j$. We write $\mathbf{n}_i$ and $\mathbf{n}_j$ to denote the outward unit normal vectors on F, relative to $\partial\kappa_i$ and $\partial\kappa_j$, respectively. Furthermore, let v and $\mathbf{q}$ be scalarand vector-valued functions, which are smooth inside each element κ_i and κ_j. By $(v_i, \mathbf{q}_i)$ and $(v_j, \mathbf{q}_j)$, we denote the traces of $(v, \mathbf{q})$ on F taken from within the interior of κ_i and κ_j, respectively. The averages of v and $\mathbf{q}$ at $x \in F$ are given by

$$\{\!\{v\}\!\} := \frac{1}{2}(v_i + v_j), \quad \{\!\{\mathbf{q}\}\!\} := \frac{1}{2}(\mathbf{q}_i + \mathbf{q}_j),$$

respectively. Similarly, the jump of v and $\mathbf{q}$ at $x \in F \subset \Gamma_{\text{int}}$ are given by

$$[\![v]\!] := v_i\mathbf{n}_i + v_j\mathbf{n}_j, \quad [\![\mathbf{q}]\!] := \mathbf{q}_i \cdot \mathbf{n}_i + \mathbf{q}_j \cdot \mathbf{n}_j,$$

respectively. On a boundary face $F \subset \Gamma$, such that $F \subset \partial\kappa_i, \kappa_i \in \mathcal{T}$, we set

$$\{\!\{v\}\!\} = v_i, \quad \{\!\{\mathbf{q}\}\!\} = \mathbf{q}_i, \quad [\![v]\!] = v_i\mathbf{n}_i \quad [\![\mathbf{q}]\!] = \mathbf{q}_i \cdot \mathbf{n}_i,$$

with $\mathbf{n}_i$ denoting the unit outward normal vector on the boundary Γ.

Remark 1.1. *The jump operator $[\![\,]\!]$ is independent of face orientation, while the sign of the upwind jump operator $\lfloor\cdot\rfloor$ depends on the direction of the flow.*

1.2.5. Interior penalty discontinuous Galerkin method

In this section, we introduce the *hp*-version DG-FEM discretization of the model problem (1.2), (1.3). For simplicity of presentation, we suppose that the entries of the diffusion tensor a are constant on each element $\kappa \in \mathcal{T}$, i.e.,

$$a \in \left[S_{\mathcal{T}}^{\mathbf{0}}\right]_{\text{sym}}^{d\times d}$$

Our results can easily be extended to the case of general $a \in L_\infty(\Omega)^{d\times d}_{\text{sym}}$ based on employing the modified DG-FEM proposed in [87]. In the following, $\sqrt{a}$ denotes the (positive semidefinite) square-root of the symmetric tensor a; further, $\bar{a}_\kappa := |\sqrt{a}|_2^2|_\kappa$, where $|\cdot|_2$ denotes the l_2-norm.

The interior penalty DG-FEM is given by: find $u_h \in S^{\mathbf{p}}_{\mathcal{T}}$ such that

$$B\left(u_h, v_h\right) = \ell\left(v_h\right)$$

for all $v_h \in S^{\mathbf{p}}_{\mathcal{T}}$. Here, the bilinear form $B(\cdot,\cdot) : S^{\mathrm{p}}_{\mathcal{T}} \times S^{\mathbf{p}}_{\mathcal{T}} \to \mathbb{R}$ is defined as the sum of two parts:

$$B(u,v) := B_{\mathrm{ar}}(u,v) + B_{\mathrm{d}}(u,v)$$

where the bilinear form $B_{\mathrm{ar}}(\cdot,\cdot)$ accounts for the advection and reaction terms:

$$\begin{aligned} B_{\mathrm{ar}}(u,v) :=& \sum_{\kappa\in\mathcal{T}} \int_\kappa (\mathbf{b}\cdot\nabla u + cu)v \,\mathrm{d}x \\ &- \sum_{\kappa\in\mathcal{T}} \int_{\partial_-\kappa\setminus\Gamma} (\mathbf{b}\cdot\mathbf{n})\lfloor u\rfloor v^+ \mathrm{d}s - \sum_{\kappa\in\mathcal{T}} \int_{\partial_-\kappa\cap(\Gamma_{\mathrm{D}}\cup\Gamma_-)} (\mathbf{b}\cdot\mathbf{n})u^+v^+\mathrm{d}s. \end{aligned} \tag{1.4}$$

The bilinear form $B_{\mathrm{d}}(\cdot,\cdot)$ takes care of the diffusion term:

$$\begin{aligned} B_{\mathrm{d}}(u,v) :=& \sum_{\kappa\in\mathcal{T}} \int_\kappa a\nabla u\cdot\nabla v \,\mathrm{d}x + \int_{\Gamma_{\mathrm{int}}\cup\Gamma_{\mathrm{D}}} \sigma[\![u]\!]\cdot[\![v]\!]\mathrm{d}s \\ &- \int_{\Gamma_{\mathrm{int}}\cup\Gamma_{\mathrm{D}}} \left(\{\!\{a\nabla_h u\}\!\}\cdot[\![v]\!] + \{\!\{a\nabla_h v\}\!\}\cdot[\![u]\!]\right)\mathrm{d}s \end{aligned} \tag{1.5}$$

Furthermore, the linear functional $\ell : S^{\mathrm{p}}_{\mathcal{T}} \to \mathbb{R}$ is defined by

$$\begin{aligned} \ell(v) :=& \sum_{\kappa\in\mathcal{T}} \int_\kappa fv \,\mathrm{d}x - \sum_{\kappa\in\mathcal{T}} \int_{\partial_-\kappa\cap(\Gamma_{\mathrm{D}}\cup\Gamma_-)} (\mathbf{b}\cdot\mathbf{n})g_{\mathrm{D}}v^+\mathrm{d}s \\ &- \int_{\Gamma_{\mathrm{D}}} g_{\mathrm{D}}\left((a\nabla_h v)\cdot\mathbf{n} - \sigma v\right)\mathrm{d}s + \int_{\Gamma_{\mathrm{N}}} g_{\mathrm{N}}v \,\mathrm{d}s \end{aligned} \tag{1.6}$$

The nonnegative function $\sigma \in L_\infty\left(\Gamma_{\mathrm{int}} \cup \Gamma_{\mathrm{D}}\right)$ appearing in (1.5) and (1.6) is referred to as the discontinuity penalization parameter and it depends on the diffusion tensor a and the discretization parameters. In particular, let $\sigma : \Gamma\setminus\Gamma_{\mathrm{N}} \to \mathbb{R}_+$be defined facewise by

$$\sigma(x) := \begin{cases} \alpha \max_{\kappa\in\{\kappa_1,\kappa_2\}} \left\{ C_{\mathrm{INV}}\left(p_\kappa,\kappa,F\right)\frac{\bar{a}_\kappa p_\kappa^2|F|}{|\kappa|}\right\}, & x\in F\subset\Gamma_{\mathrm{int}}, F=\partial\kappa_1\cap\partial\kappa_2, \\ \alpha\, C_{\mathrm{INV}}\left(p_\kappa,\kappa,F\right)\frac{\bar{a}_\kappa p_\kappa^2|F|}{|\kappa|}, & x\in F\subset\Gamma_{\mathrm{D}}, F=\partial\kappa\cap\Gamma_{\mathrm{D}}, \end{cases}$$

with $\alpha > 0$ large enough, and independent of $p_\kappa, |F|$, and $|\kappa|$. C_{INV} is the inverse trace inequality constant defined as in [51].

1.2.6. Mesh assumptions

Assumption 1.1. *The subdivision $\mathcal{T}$ is shape regular in the sense of [58], i.e., there exists a positive constant C_{shape}, independent of the mesh parameters, such that:*

$$\forall \kappa \in \mathcal{T}, \quad \frac{h_\kappa}{\rho_\kappa} \leq C_{shape},$$

with ρ_κ denoting the diameter of the largest ball contained in κ.

Assumption 1.2. *There exists a positive constant C_F , independent of the mesh parameters, such that*

$$\max_{\kappa \in \mathcal{T}} (card\{F \subset \Gamma \cup \Gamma_{int} : F \subset \partial\kappa\}) \leq C_F.$$

Remark 1.2. *We note that Assumption 4.2 naturally imposes the condition that the number of hanging nodes and the number of faces that each element κ in the finite element mesh $\mathcal{T}$ possesses is uniformly bounded under mesh refinement.*

As in [50], we require the existence of the following coverings of the mesh.

Definition 1.1. *A (typically overlapping) covering $\mathcal{T}_\# = \{\mathcal{K}\}$ related to the polytopic mesh T is a set of shape-regular d-simplices $\mathcal{K}$, such that for each $\kappa \in \mathcal{T}$, there exists a $\mathcal{K} \in \mathcal{T}_\#$, with $\kappa \subset \mathcal{K}$. Given $\mathcal{T}_\#$, we denote by $\Omega_\#$ the covering domain given by $\Omega_\# := (\cup_{\mathcal{K} \in \mathcal{T}_\#} \bar{\mathcal{K}})^\circ$, where, for a closed set $D \subset \mathbb{R}^d$, D° denotes the interior of D.*

Assumption 1.3. *There exists a covering $\mathcal{T}_\#$ of $\mathcal{T}$ and a positive constant $\mathcal{O}_\Omega$, independent of the mesh parameters, such that the subdivision $\mathcal{T}$ satisfies*

$$\max_{\kappa \in \mathcal{T}} \mathcal{O}_\kappa \leq \mathcal{O}_\Omega,$$

where, for each $\kappa \in \mathcal{T}$,

$$\mathcal{O}_\kappa := card\ \{\kappa' \in \mathcal{T} : \kappa' \cap \mathcal{K}' = \emptyset, \mathcal{K} \in \mathcal{T}_\# such\ that \kappa \subset \mathcal{K}\}.$$

As a consequence, we deduce that

$$diam(K) \leq C_{diam} h_\kappa,$$

for each pair $\kappa \in \mathcal{T}$, $\mathcal{K} \in \mathcal{T}_{\#}$, with $\kappa \subset \mathcal{K}$, for a constant $C_{diam} > 0$, uniformly with respect to the mesh size.

We note that Assumption 1.3 ensures that the amount of overlap present in the covering $\mathcal{T}_{\#}$ remains bounded as the computational mesh $\mathcal{T}$ is refined. The proceeding hp-approximation results and inverse estimates for polytopic elements are based on referring back to d-dimensional simplices, where standard results can be applied; see, for example, [24, 25, 57, 122].

Definition 1.2. *For each element κ in the computational mesh $\mathcal{T}$, we define the family $\mathcal{F}_b^\kappa$ of all possible d-dimensional simplices contained in κ and having at least one face in common with κ. The notation κ_b^F will be used to indicate a simplex belonging to $\mathcal{F}_b^\kappa$ and sharing with $\kappa \in \mathcal{T}$ a given face F. Functions defined on Ω can be extended to the covering domain $\Omega_{\#}$ based on employing the following extension operator, cf. [137].*

Definition 1.3. *We let $\tilde{\mathcal{T}}$ denote the subset of elements $\kappa, \kappa \in \mathcal{T}$, such that each $\kappa \in \tilde{\mathcal{T}}$ can be covered by at most $m_{\mathcal{T}}$ shape-regular simplices $K_i, i = 1, \ldots, m_{\mathcal{T}}$, such that*

$$\operatorname{dist}(\kappa, \partial K_i) > C_{as} \operatorname{diam}(K_i) / p_\kappa^2,$$

and

$$|K_i| \geq c_{as} |\kappa|$$

for all $i = 1, \ldots, m_{\mathcal{T}}$, for some $m_{\mathcal{T}} \in \mathbb{N}$ and $C_{as}, c_{as} > 0$, independent of κ and $\mathcal{T}$, where p_κ denotes the polynomial degree associated with element $\kappa, \kappa \in \mathcal{T}$.

Assumption 1.4. *Every polytopic element $\kappa \in \mathcal{T} \backslash \tilde{\mathcal{T}}$, admits a sub-triangulation into at most $n_{\mathcal{T}}$ shaperegular simplices $\mathfrak{s}_i, i = 1, 2, \ldots, n_{\mathcal{T}}$, such that $\bar{\kappa} = \cup_{i=1}^{n_{\mathcal{T}}} \bar{\mathfrak{s}}_i$ and*

$$|\mathfrak{s}_i| \geq \hat{C} |\kappa|$$

for all $i = 1, \ldots, n_{\mathcal{T}}$, for some $n_{\mathcal{T}} \in \mathbb{N}$ and $\hat{c} > 0$, independent of κ and $\mathcal{T}$.

1.2.7. Error analysis

We introduce the DG-FEM-norm $|||\cdot|||_{\mathrm{DG}}$ as the sum of two parts as follows:

$$|||v|||_{\mathrm{DG}}^2 := |||v|||_{\mathrm{ar}}^2 + |||v|||_{\mathrm{d}}^2,$$

where

$$|||v|||_{\mathrm{ar}}^2 := \sum_{\kappa\in\mathcal{T}} \left(\|c_0 v\|_{L_2(\kappa)}^2 + \frac{1}{2}\left\|v^+\right\|_{\partial_-\kappa\cap(\Gamma_{\mathrm{D}}\cup\Gamma_-)}^2 + \frac{1}{2}\left\|v^+ - v^-\right\|_{\partial_-\kappa\setminus\Gamma}^2 + \frac{1}{2}\left\|v^+\right\|_{\partial_+\kappa\cap\Gamma}^2 \right)$$

with c_0 as in (2.5), and

$$|||v|||_{\mathrm{d}}^2 := \sum_{\kappa\in\mathcal{T}} \|\sqrt{a}\nabla v\|_{L_2(\kappa)}^2 + \int_{\Gamma_{\mathrm{int}}\cup\Gamma_{\mathrm{D}}} \sigma|[\![v]\!]|^2 \,\mathrm{d}s.$$

Here, $\|\cdot\|_\tau, \tau\subset\partial\kappa$, denotes the (semi)norm associated with the (semi)inner product $(v,w)_\tau = \int_\tau |\mathbf{b}\cdot\mathbf{n}| vw \,\mathrm{d}s$. Under the considered assumtions it is possible to prove that

$$|||u-u_h|||_{\mathrm{DG}} \le C\frac{h^{s-1}}{p^{l-\frac{3}{2}}}\|u\|_{H^l(\Omega)},$$

where u is the exact solution (1.2) and (1.3), u_h is the numerical solution obtained employing the PolyDG scheme, C is a suitable constant, h is the meshsize, p is the polinomial degree and $s=\min\{p+1,l\}$, $l>1+d/2$. This coincides with the analogous result derived in [36] for standard meshes consisting of simplices or tensorproduct elements. This error bound is h optimal and p suboptimal by $p^{1/2}$.

1.3. Polytopal discontinuous Galerkin discretization of the transient Stokes problem

We consider the transient Stokes problem which reads as follows: given a final time $T>0$, f a (regular) forcing term, and μ and ρ the viscosity and density of the fluid, find the velocity $u=u(t)$ and the pressure $p=p(t)$ such that, for all $t\in(0,T]$,

$$\begin{cases} \rho\partial_t\mathbf{u} - \mu\Delta\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \\ \operatorname{div}\mathbf{u} = 0 & \text{in } \Omega \\ \mathbf{u} = \mathbf{0} & \text{on } \partial\Omega \end{cases} \tag{1.7}$$

Problem (1.7) is supplemented with sufficiently regular initial conditions $\boldsymbol{u}(\boldsymbol{x},0)=\boldsymbol{u}^0(\boldsymbol{x})$ in Ω. To guarantee the well-posedness of the problem, we prescribe that $p\in L_0^2(\Omega)$, where $L_0^2(\Omega)$ is the space of $L^2(\Omega)$ functions with zero average over Ω.

We introduce the functional spaces

$$\boldsymbol{V} = \left\{ \boldsymbol{v}\in\left[H^1(\Omega)\right]^d, d=2,3, \text{ such that } \boldsymbol{v}|_{\partial\Omega}=0 \right\}$$

and $Q = L^2_0(\Omega)$ and endow them with the norms

$$\|\boldsymbol{v}\|_{\boldsymbol{V}} := \left\|\mu^{\frac{1}{2}}\nabla\boldsymbol{v}\right\|_{L^2(\Omega)} \quad \text{and} \quad \|q\|_Q := \|q\|_{L^2(\Omega)}.$$

The weak formulation of problem (1.7) reads as follows: find $(\boldsymbol{u}, p) \in \boldsymbol{V} \times Q$, such that, for all $t \in (0, T]$

$$(\rho\partial_t\boldsymbol{u}, \boldsymbol{v})_\Omega + a(\boldsymbol{u}, \boldsymbol{v}) + b(p, \boldsymbol{v}) - b(q, \boldsymbol{u}) = (\boldsymbol{f}, \boldsymbol{v})_\Omega \quad \forall(\boldsymbol{v}, q) \in \boldsymbol{V} \times Q, \tag{1.8}$$

where

$a : \boldsymbol{V} \times \boldsymbol{V} \to \mathbb{R}$, $\quad a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \mu\nabla\boldsymbol{u} : \nabla\boldsymbol{v}$, $\quad b : Q \times \boldsymbol{V} \to \mathbb{R}$, $\quad b(p, \boldsymbol{v}) = -\int_\Omega p\nabla\cdot\boldsymbol{v}$,

and $(\cdot,\cdot)_\Omega$ denotes the L^2-inner product over the domain Ω. It is well-known that the bilinear form $b(\cdot,\cdot)$ satisfies a continuous inf-sup condition; see, e.g., [39]. More precisely, there exists a universal positive constant depending only on Ω such that, to all $q \in L^2_0(\Omega)$, we associate a function $\boldsymbol{v}_q \in \boldsymbol{V}$ satisfying $\nabla\cdot\boldsymbol{v}_q = q$ and

$$\beta\,\|\boldsymbol{v}_q\|_V \le \|q\|_{L^2(\Omega)}.$$

1.3.1. Polygonal discontinuous Galerkin semi-discrete approximation

First, we introduce the necessary notation and key analytical results required for the definition and analysis of PolyDG semi-discrete approximation of the transient Stokes problem. For more details see [20].

We introduce a mesh $\mathcal{T}_h$ composed of polytopic elements K of arbitrary shape. We indicate with h_K the diameter of the element K. We define an interface to be either the intersection of the $(d-1)$-dimensional facets of two neighboring elements or the intersection of the $(d-1)$ dimensional facets of an element with the boundary of Ω. When $d = 2$, interfaces coincide with edges and consist of line segments; in presence of hanging nodes, a single line segment can contain more than one interface. When $d = 3$, we assume that each interface consists of a general planar polygon that we assume that can be further decomposed into a set of co-planar triangles, denoted as faces. With this notation, we collect all the $(d-1)$-dimensional faces in the set $\mathcal{F}_h$, i.e., any face $F \in \mathcal{F}_h$ is always defined as a set of $(d-1)$-dimensional simplices (line segments or triangles); cf. [50]. We also decompose the faces $\mathcal{F}_h$ into $\mathcal{F}_h = \mathcal{F}_h^i \cup \mathcal{F}_h^b$, where $\mathcal{F}_h^i$ denotes the set of interior faces and $\mathcal{F}_h^b$ denotes the set of boundary faces. To avoid technicalities, in the following we assume that ρ and

μ are piecewise constant over the mesh.

For given integers $\ell, m \geq 1$, we introduce the DG finite element spaces

$$\mathbf{V}_h^\ell = \left\{ \boldsymbol{v} \in \left[L^2(\Omega)\right]^d : v|_K \in \left[\mathcal{P}^\ell(K)\right]^d \forall K \in \mathcal{T}_h \right\},$$
$$Q_h^m = \left\{ q \in L_0^2(\Omega) : q|_K \in \mathcal{P}^m(K) \forall K \in \mathcal{T}_h \right\},$$

where $\mathcal{P}^k(K), k \geq 1$, denotes the space of polynomials defined over the element $K \in \mathcal{T}_h$ of total degree at most k. In practice, the shape functions and the degrees of freedom are Springer directly generated on the physical element $K \in \mathcal{T}_h$ with the "bounding box" technique; see, e.g., [50].

Given $s > 1/2$, associated with any mesh $\mathcal{T}_h$, we introduce the broken Sobolev space $H^s(\mathcal{T}_h) = \{v \in L^2(\Omega) | v|_K \in H^s(K) \text{ for all } K \in \mathcal{T}_h\}$. The standard Dirichlet trace operator is well defined on the skeleton of the mesh for functions in $H^s(\mathcal{T}_n)$. We define the stabilization functions $\sigma_v \in L^\infty(\mathcal{F}_h)$ and $\sigma_p \in L^\infty(\mathcal{F}_h)$ as follows.

Definition 1.4. *We define the functions $\sigma_v : \mathcal{F}_h \to \mathbb{R}$ and $\sigma_p : \mathcal{F}_h^i \to \mathbb{R}$ as*

$$\sigma_v|_F = \begin{cases} \gamma_v \max_{K^+,K^-} \left\{ \frac{\ell^2 \mu}{h_K} \right\} & F \in \mathcal{F}_h^i, \\ \gamma_v \frac{\ell^2 \mu}{h_K} & F \in \mathcal{F}_h^b, \end{cases} \qquad \sigma_p \Big|_F = \gamma_p \min_{K^+,K^-} \left\{ \frac{h_K}{m} \right\} \quad F \in \mathcal{F}_h^i,$$

where γ_v and γ_p are two universal positive constants, and ℓ and m denote the polynomial approximation degrees for the velocity and the pressure, respectively.

Next, we introduce three bilinear forms that are instrumental for the construction of the DG method. More precisely, we define

$$\begin{aligned} a_h(\boldsymbol{u}, \boldsymbol{v}) &= \int_\Omega \mu \nabla_h \boldsymbol{u} : \nabla_h \boldsymbol{v} - \sum_{F \in \mathcal{F}_h} \int_F \mu \{\!\{\nabla_h \boldsymbol{u}\}\!\} : [\![\boldsymbol{v}]\!] \\ &\quad - \sum_{F \in \mathcal{F}_h} \int_F \mu [\![\boldsymbol{u}]\!] : \{\!\{\nabla_h \boldsymbol{v}\}\!\} + \sum_{F \in \mathcal{F}_h} \int_F \sigma_v [\![\boldsymbol{u}]\!] : [\![\boldsymbol{v}]\!] \\ b_h(p, \boldsymbol{v}) &= - \int_\Omega p \nabla_h \cdot \boldsymbol{v} + \sum_{F \in \mathcal{F}_h} \int_F \{\!\{p \boldsymbol{I}\}\!\} : [\![\boldsymbol{v}]\!], \\ s_h(p, q) &= \sum_{F \in \mathcal{F}_h^i} \int_F \sigma_p [\![p]\!] \cdot [\![q]\!] \end{aligned}$$

where ∇_h is the piecewise broken gradient operator, while the average operator $\{\!\{\cdot\}\!\}$ and jump operator $[\![\cdot]\!]$ were defined previously in Section 1.2.4. The bilinear form $s_h(p, q)$ represents the pressure stabilization term and σ_p plays the role of penalty parameter.

Given $f \in [L^2(\Omega)]^d$, the semi-discrete PolyDG approximation of (1.8) reads as follows: for any $t \in (0,T]$, find $(\boldsymbol{u}_h, p_h) \in \mathbf{V}_h^\ell \times Q_h^m$ such that

$$(\rho \partial_t \boldsymbol{u}_h, \boldsymbol{v}_h)_\Omega + a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) + b_h(p_h, \boldsymbol{v}_h) - b_h(q_h, \boldsymbol{u}_h) + s_h(p_h, q_h) = (\boldsymbol{f}, \boldsymbol{v}_h)_\Omega$$

for all $(\boldsymbol{v}_h, q_h) \in \mathbf{V}_h^\ell \times Q_h^m$.

1.3.2. Error analysis of the stationary Stokes problem

We now consider the stationary Stokes problem, i.e (1.7) with $\rho = 0$. On the product space $\mathbf{V}_h^\ell \times Q_h^m$, we define the norm

$$\|(\boldsymbol{v}_h, q_h)\|_{\mathrm{E}}^2 = \|\boldsymbol{v}_h\|_{\mathbf{V}_h^\ell}^2 + \|q_h\|_{Q_h^m}^2 \quad \forall\, (\boldsymbol{v}_h, q_h) \in \mathbf{V}_h^\ell \times Q_h^m,$$

where

$$\begin{aligned} \|\boldsymbol{v}_h\|_{\mathbf{V}_h^\ell}^2 &= \textstyle\sum_{K\in\mathcal{T}_k} \left\|\mu^{1/2}\nabla_h \boldsymbol{v}_h\right\|_{L^2(K)}^2 + \left\|\sigma_v^{1/2}[\![\boldsymbol{v}_h]\!]\right\|_{L^2(\mathcal{F}_h)}^2 \quad \forall \boldsymbol{v}_h \in \mathbf{V}_h^\ell, \\ \|q_h\|_{Q_h^m}^2 &= \|q_h\|_{L^2(\Omega)}^2 + |q_h|_j^2, \quad |q_h|_j^2 = s_h(q_h, q_h) \quad \forall q_h \in Q_h^m. \end{aligned}$$

We also introduce the spaces

$$\mathcal{X} = \mathbf{V}_h^\ell + \left(\left[H^2(\Omega)\right]^d \cap \left[H_0^1(\Omega)\right]^d\right), \quad \mathcal{M} = Q_h^m + \left(H^1(\Omega) \cap L_0^2(\Omega)\right)$$

for the velocity and pressure, respectively. We endow $\mathcal{X} \times \mathcal{M}$ with the energy norm

$$\|(\boldsymbol{v}, q)\|_{\mathcal{X}\times\mathcal{M}}^2 = \|(\boldsymbol{v}, q)\|_{\mathrm{E}}^2 + \sum_{F\in\mathcal{F}_h} \left\|\sigma_v^{-1/2}\{p\boldsymbol{I}\}\right\|_{L^2(F)}^2 + \sum_{F\in\mathcal{F}_h} \left\|\sigma_v^{-1/2}\{\nabla_h \boldsymbol{u}\}\right\|_{L^2(F)}^2.$$

The following error estimate holds

$$\|(\boldsymbol{u} - \boldsymbol{u}_h, p - p_h)\|_{\mathcal{X}\times\mathcal{M}} \lesssim \frac{1}{\beta_h} \frac{h^{s-1}}{m^{r-3/2}} \left(\|\mathcal{E}\boldsymbol{u}\|_{H^r\left(\cup_{K\in\mathcal{T}_h}\mathcal{K}\right)} + \|\mathcal{E}p\|_{H^{r-1}\left(\cup_{K\in\mathcal{T}_h}\mathcal{K}\right)}\right)$$

where m is the polynomial degrees of the discretization, $s = \min\{m+1, r\}$, β_h is the inf-sup constant described in [20] and $\mathcal{E} : H^\ell(\Omega) \to H^\ell(\mathbb{R}^d)$, $\ell \geq 0$ is the Stein extension operator for Sobolev spaces on Lipschitz domains introduced in [137].

1.4. Polygonal discontinuous Galerkin multigrid formulation of the Poisson problem

We consider the weak formulation of the Poisson problem, subject to homogeneous Dirichlet boundary conditions: find $u \in V = H^2(\Omega) \cap H_0^1(\Omega)$ such that

$$\mathcal{A}(u,v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x = \int_\Omega f v \, \mathrm{d}x \quad \forall v \in V, \tag{1.9}$$

with $\Omega \subset \mathbb{R}^d, d = 2, 3$, a convex polygonal/polyhedral domain with Lipschitz boundary and $f \in L^2(\Omega)$. The unique solution $u \in V$ of problem (1.9) satisfies

$$\|u\|_{H^2(\Omega)} \leq C \|f\|_{L^2(\Omega)}.$$

In view of the forthcoming multigrid analysis, let $\{\mathcal{T}_j\}_{j=1}^J$ be a sequence of tessellation of the domain Ω, each of which is characterized by disjoint open polytopic elements κ of diameter h_κ, such that $\bar{\Omega} = \bigcup_{\kappa \in \mathcal{T}_j} \bar{\kappa}, j = 1, \ldots, J$. The mesh size of $\mathcal{T}_j$ is denoted by $h_j = \max_{\kappa \in \mathcal{T}_j} h_\kappa$. For the sake of simplicity, we assume that on each level the mesh $\mathcal{T}_j$ is quasi-uniform. To each $\mathcal{T}_j$ we associate the corresponding discontinuous finite element space V_j, defined as

$$V_j = \left\{ v \in L^2(\Omega) : v|_\kappa \in \mathcal{P}_{p_j}(\kappa), \kappa \in \mathcal{T}_j \right\},$$

where $\mathcal{P}_{p_j}(\kappa)$ denotes the space of polynomials of total degree at most $p_j \geq 1$ on $\kappa \in \mathcal{T}_j$.

Remark 1.3. *For the sake of brevity we use the notation $x \lesssim y$ to mean $x \leq Cy$, where $C > 0$ is a constant independent from the discretization parameters. Similarly we write $x \gtrsim y$ in lieu of $x \geq Cy$, while $x \approx y$ is used if both $x \lesssim y$ and $x \gtrsim y$ hold.*

A suitable choice of $\{\mathcal{T}_j\}_{j=1}^J$ and $\{V_j\}_{j=1}^J$ leads to the non-nested *hp*-multigrid schemes. This method is based on employing a set of non-nested polytopic partitions $\{\mathcal{T}_j\}_{j=1}^J$, such that the coarse level $\mathcal{T}_{j-1}$ is independent from $\mathcal{T}_j$, with the only constraint

$$h_{j-1} \lesssim h_j \leq h_{j-1} \quad \forall j = 2, \ldots, J.$$

We also assume that the polynomial degree varies from one level to another such that

$$p_{j-1} \leq p_j \lesssim p_{j-1} \quad \forall j = 2, \ldots, J.$$

wereAdditional assumptions on the grids $\{\mathcal{T}_j\}_{j=1}^J$ were outlined in Section 1.2.6.

1.4.1. Discontinuous Galerkin formulation

Let $\mathcal{R}_j : [L^1(\mathcal{F}_j)]^d \to [V_j]^d$ be the lifting operator defined as

$$\int_\Omega \mathcal{R}_j(\mathbf{q}) \cdot \eta = -\int_{\mathcal{F}_j} \mathbf{q}\{\!\{\eta\}\!\} \mathrm{d}s \quad \forall \eta \in [V_j]^d,$$

cf. [22], where $\mathcal{F}_j$ is the set of faces of mesh $\mathcal{T}_j$. We introduce the bilinear form $\mathcal{A}_j(\cdot,\cdot) : V_j \times V_j \to \mathbb{R}$ corresponding to the symmetric interior penalty DG method on the j-th level is defined by

$$\begin{aligned}\mathcal{A}_j(u,v) = &\sum_{\kappa\in\mathcal{T}_j} \int_\kappa [\nabla u \cdot \nabla v + \mathcal{R}_j([\![u]\!]) \cdot \nabla v + \mathcal{R}_j([\![v]\!]) \cdot \nabla u] \,\mathrm{d}x \\ &+ \sum_{F\in\mathcal{F}_j} \int_F \sigma_j [\![u]\!] \cdot [\![v]\!] \mathrm{d}s,\end{aligned}$$

where, according to [8, 81], $\sigma_j \in L^\infty(\mathcal{F}_j)$ denotes the interior penalty stabilization function, which is defined by

$$\sigma_j(x) = C_\sigma^j \frac{p_j^2}{\{h_\kappa\}_H}, \quad x \in F \in \mathcal{F}_j,$$

with $C_\sigma^j > 0$ independent of $p_j, |F|$ and $|\kappa|$, and where $\{\cdot\}_H$ is the harmonic average given by

$$\{h_\kappa\}_H = \begin{cases} \frac{2h_{\kappa^+}h_{\kappa^-}}{h_{\kappa^+}+h_{\kappa^-}}, & F \in \mathcal{F}_j^I, \bar{F} \subset \partial\overline{\kappa^+} \cap \partial\overline{\kappa^-}, \\ h_\kappa, & F \in \mathcal{F}_j^B, F \subset \partial\bar{\kappa} \cap \partial\bar{\Omega}. \end{cases}$$

Remark 1.4. *Formulation (5) is based on the lifting operators $\mathcal{R}_j$ and allows to introduce the discrete gradient operator $\mathcal{G}_j : V_j \to [V_j]^d$, defined as*

$$\mathcal{G}_j(v) = \nabla_j v + \mathcal{R}_j([\![v]\!]) \quad \forall j = 1, \ldots, J,$$

where ∇_j is the piecewise gradient operator on the space V_j.

Our goal is to develop non-nested V-cycle multigrid schemes to solve the following problem posed on the finest level V_J : find $u_J \in V_J$ such that

$$\mathcal{A}_J(u_J, v_J) = \int_\Omega f v_J \,\mathrm{d}x \quad \forall v_J \in V_J. \tag{1.10}$$

1.4.2. The BPX-framework for the V-cycle algorithms

The analysis presented in this section is based on the general multigrid theoretical framework of [42] for multigrid methods with non-nested spaces and non-inherited bilinear forms. In order to develop a geometric multigrid, the discretization at each level V_j follows the one already presented in [12], where a W-cycle multigrid method based on nested subspaces is considered. For more details see [8]. The key ingredient in the construction of our proposed multigrid schemes is the inter-grid transfer operators.

First, we introduce the operators $A_j : V_j \to V_j$, defined as

$$(A_j w, v)_{L^2(\Omega)} = \mathcal{A}_j(w, v) \quad \forall w, v \in V_j, \quad j = 1, \dots, J, \tag{1.11}$$

and we denote by $\Lambda_j \in \mathbb{R}$ the maximum eigenvalue of $A_j, j = 2, \dots, J$. Moreover, let Id_j be the identity operator on the level V_j. The smoothing scheme, which is chosen to be the Richardson iteration, is given by

$$B_j = \Lambda_j \, \mathrm{Id}_j \quad j = 2, \dots, J.$$

The prolongation operator connecting the coarser space V_{j-1} to the finer space V_j is denoted by I_{j-1}^j. Since the two spaces are non-nested, i.e. $V_{j-1} \not\subset V_j$, it cannot be chosen as the natural injection operator. The most natural way to define the prolongation operator is the L^2-projection, i.e. $I_{j-1}^j : V_{j-1} \to V_j$

$$\left(I_{j-1}^j v_H, w_h\right)_{L^2(\Omega)} = (v_H, w_h)_{L^2(\Omega)} \, \forall w_h \in V_j,$$

The restriction operator $I_j^{j-1} : V_j \to V_{j-1}$ is defined as the adjoint of I_{j-1}^j with respect to the $L^2(\Omega)$-inner product, i.e.,

$$\left(I_j^{j-1} w_h, v_H\right)_{L^2(\Omega)} = \left(w_h, I_{j-1}^j v_H\right)_{L^2(\Omega)} \quad \forall v_H \in V_{j-1}.$$

For our analysis, we also need to introduce the operator $P_j^{j-1} : V_j \to V_{j-1}$

$$\mathcal{A}_{j-1}\left(P_j^{j-1} w_h, v_H\right) = \mathcal{A}_j\left(w_h, I_{j-1}^j v_H\right) \quad \forall v_H \in V_{j-1}, w_h \in V_j.$$

According to (1.11), problem (1.10) can be written in the following equivalent form: find $u_J \in V_J$ such that

$$A_J u_J = f_J, \tag{1.12}$$

where $f_J \in V_J$ is defined as $(f_J, v)_{L^2(\Omega)} = \int_\Omega f v dx \; \forall v \in V_J$. Given an initial guess $u_0 \in V_J$, and choosing the parameters $m_1, m_2 \in \mathbb{N}$, the multigrid V-cycle iteration algorithm for the approximation of u_J is outlined in Algorithm 1.1.

Algorithm 1.1 Multigrid V-cycle iteration for the solution of problem (1.12)

Initialize $u_0 \in V_j$
for k = 0,1,... **do**
  $u_{k+1} = \mathrm{MG}_{\mathcal{V}}(J, f_J, u_k, m_1, m_2)$
  $u_k = u_{k+1}$
end for

Algorithm 1.2 One iteration of the Multigrid V-cycle scheme

if $j = 1$ **then**
  $\mathrm{MG}_\nu(1, f, z_0, m_1, m_2) = A_1^{-1} f$
else
  Pre-smoothing:
  for $i = 1, ..., m_1$ **do**
    $z^{(i)} = z^{(i-1)} + B_j^{-1}(g - A_j z^{(i-1)})$
  end for
  Coarse grid correction:
  $r_{j+1} = I_j^{j-1}(f - A_j z^{(m_1)})$
  $e_{j+1} = \mathrm{MG}_\nu(j-1, r_{j-1}, 0, m_1, m_2)$
  $z^{(m_1+1)} = z^{(m_1)} + I_{j-1}^{j} e_{j-1}$
  Post-smoothing:
  for $i = m_1 + 2, ..., m_1 + m_2 + 1$ **do**
    $z^{(i)} = z^{(i-1)} + B_j^{-1}(g - A_j z^{(i-1)})$
  end for
  $\mathrm{MG}_\nu(J, f, z_0, m_1, m_2) = z^{m_1+m_2+1}$
end if

In particular, $\mathrm{MG}_{\mathcal{V}}(J, f_J, u_k, m_1, m_2)$ represents the approximate solution obtained after one iteration of our non-nested V-cycle scheme, which is defined by induction: if we consider the general problem of finding $z \in V_j$ such that

$$A_j z = g, \tag{1.13}$$

with $j \in \{2, \ldots, J\}$ and $g \in L^2(\Omega)$, then $\mathrm{MG}_{\mathcal{V}}(j, g, z_0, m_1, m_2)$ represents the approxi-

mate solution of (1.13) obtained after one iteration of the non-nested V-cycle scheme with initial guess $z_0 \in V_j$ and m_1, m_2 pre-smoothing and post-smoothing steps, respectively. The recursive procedure is outlined in Algorithm 1.2, where we also observe that on the level $j = 1$ the problem is solved exactly.

1.4.3. Additive Schwarz smoother

In order to improve the performance of our V-cycle algorithm, we define in this section a domain decomposition preconditioner that can be used as a smoothing operator in place of the Richardson one. To this aim, let $\mathcal{T}_j$ and $\mathcal{T}_{j-1}$ be a pair of consecutive (non-nested) coarse/fine meshes, satisfying the grid assumptions given in Section 1.2.6. We next introduce the local and coarse solvers, that are the key ingredients in the definition of the smoother on the space $V_j, j = 2, \dots, J$.

Local Solvers. Let us consider the finest mesh $\mathcal{T}_j$ with cardinality N_j, then for each element $\kappa_i \in \mathcal{T}_j$, we define a local space V_j^i as the restriction of the DG finite element space V_j to the element $\kappa_i \in \mathcal{T}_j$:

$$V_j^i = V_j|_{\kappa_i} \equiv \mathcal{P}_{p_j}(\kappa_i) \quad \forall i = 1, \dots, N_j,$$

and for each local space, the associated local bilinear form is defined by

$$\mathcal{A}_j^i : V_j^i \times V_j^i \to \mathbb{R}, \quad \mathcal{A}_j^i(u_i, v_i) = \mathcal{A}_j\left(R_i^T u_i, R_i^T v_i\right) \quad \forall u_i, v_i \in V^i,$$

where $R_i^T : V_j^i \to V_j$ denotes the classical extension by-zero operator from the local space V_j^i to the global V_j.

Coarse Solver. The natural choice in our contest is to define the coarse space V_j^0 to be exactly the same used for the Coarse grid correction step of the V-cycle algorithm introduced in Sect. 4, that is

$$V_j^0 = V_{j-1} \equiv \left\{ v \in L^2(\Omega) : v|_\kappa \in \mathcal{P}_{p_{j-1}}(\kappa), \kappa \in \mathcal{T}_{j-1} \right\},$$

the bilinear form on V_j^0 is then given by

$$\mathcal{A}_j^0 : V_j^0 \times V_j^0 \to \mathbb{R}, \quad \mathcal{A}_j^0(u_0, v_0) = \mathcal{A}_{j-1}(u_0, v_0) \quad \forall u_0, v_0 \in V_j^0$$

Here, we define the injection operator from V_j^0 to V_j as the prolongation operator in-

troduced in Sect. 4, that is $R_0^T : V_j^0 \to V_j, R_0^T = I_{j-1}^j$. By introducing the projection operators $P_i = R_i^T \tilde{P}_i : V_j \to V_j, i = 0, 1, \ldots, N_j$, where

$$\tilde{P}_i : V_j \to V_j^i, \quad \mathcal{A}_j^i \left(\tilde{P}_i v_h, w_i \right) = \mathcal{A}_j \left(v_h, R_i^T w_i \right) \quad \forall w_i \in V_j^i, \quad i = 1, \ldots, N_j,$$
$$\tilde{P}_0 : V_j \to V_j^0, \quad \mathcal{A}_j^0 \left(\tilde{P}_0 v_h, w_0 \right) = \mathcal{A}_j \left(v_h, R_0^T w_0 \right) \quad \forall w_0 \in V_j^0$$

the additive Schwarz operator is defined by $P_{ad} = \sum_{i=0}^{N_j} \left(R_i^T \left(A_j^i \right)^{-1} R_i \right) A_j \equiv B_{ad}^{-1} A_j$, where $B_{ad}^{-1} = \sum_{i=0}^{N_j} \left(R_i^T \left(A_j^i \right)^{-1} R_i \right)$ is the preconditioner. Then, the Additive Schwarz smoothing operator with m steps consists in performing m iterations of the Preconditioned Conjugate Gradient method using B_{ad} as preconditioner. In Algorithm 1.3 we show the V-cycle multigrid method using P_{ad} as a smoother

Algorithm 1.3 One iteration of Multigrid V-cycle scheme with AS-smoother

if $j = 1$ **then**
  $\mathrm{MG}_{\mathcal{AS}}(1, f, z_0, m_1, m_2) = A_1^{-1} f$
else
  Pre-smoothing:
  $z^{(m_1)} = ASPCG(A_j, z_0, g, m_1)$
  Coarse grid correction:
  $r_{j+1} = I_j^{j-1}(f - A_j z^{(m_1)})$
  $e_{j+1} = \mathrm{MG}_{\mathcal{AS}}(j-1, r_{j-1}, 0, m_1, m_2)$
  $z^{(m_1+1)} = z^{(m_1)} + I_{j-1}^j e_{j-1}$
  Post-smoothing:
  $z^{(m_1+m_2+1)} = ASPCG(A_j, z^{m_1+1}, g, m_2)$
  $\mathrm{MG}_\nu(J, f, z_0, m_1, m_2) = z^{m_1+m_2+1}$
end if

Here, $M_{\mathcal{AS}}(j, g, z_0, m_1, m_2)$ denotes the approximate solution of $A_j z = g$ obtained after one iteration, with initial guess z_0 and m_1, m_2 pre- and post-smoothing steps, respectively. The smoothing step is given by the algorithm $ASPCG$, i.e., $z = ASPCG(A, z_0, g, m)$ represents the output of m steps of Preconditioned Conjugate Gradient method applied to the linear system of equations $Ax = g$, by using B_{as} as preconditioner and starting from the initial guess z_0.

2 | Machine learning techniques for refinement and agglomeration

In this chapter we introduce different ML techniques that can be employed for grid refinement and agglomeration. The former can be re-framed as an image classification problem, while the latter can be recast as a graph partitioning problem, as we will see more in detail in the next chapters. In particular, we will introduce different architectures of neural networks and the k-means algorithm, to be employed within supervised and unsupervised learning frameworks.

2.1. Supervised learning for image classification

Consider a two dimensional gray-scale image, represented by a tensor $B \in \mathbb{R}^{m\times n},\ m, n \geq 1$, and the corresponding label vector $y = ([y]_j)_{j=1,..,\ell} \in [0,1]^\ell$, where $\ell \geq 2$ is the total number of classes, and $[y]_j$ is the probability of B to belong to the class j for $j = 1 : \ell$. In a supervised learning framework, we are given a dataset of desired input-output couples $\{(B_i, y_i)\}_{i=1}^N$, where N is number of labelled data. We consider then an image classifier represented by a function of the from $F : \mathbb{R}^{m\times n} \rightarrow\ (0,1)^\ell$, which in our case will be a CNN, parameterized by $w \in \mathbb{R}^M$ where $M \geq 1$ is the number of parameters. Our goal is to tune w so that F minimizes the data misfit, i.e.

$$\min_{w\in\mathbb{R}^M} \sum_{i\in I} l(F(B_i), y_i),$$

where I is a subset of $\{1, 2, ..., N\}$ and l is the cross-entropy loss function defined as

$$l(F(B), y) = \sum_{j=1}^{\ell} -[y]_j \log[F(B)]_j.$$

This optimization phase is also called “learning" or “training phase". During this phase, a known shortcoming is “overfitting": the model fits very well the data used in the training phase, but performs poorly on new data. For this reason, the data set is usually splitted into: i) training set: used to tune the parameters during the training phase; ii) validation set: used to monitor the model capabilities on different data during the training phase. The training is halted if the error on the validation set starts to increase; iii) test set: used to access the actual model performance on new data after the training.
While the training phase can be computationally demanding, because of the large amount of data and parameters to tune, it needs to be performed off line once and for all. Instead, classifying a new image using a pre-trained model is computationally fast: it requires only to evaluate F on a new input. The predicted label is the one with the highest estimated probability.

2.1.1. Convolutional neural networks

CNNs are parameterized functions, in our case of the form

$$\text{CNN}: \ \mathbb{R}^{m\times n} \ \to (0,1)^{\ell}, \quad m,n,\ell \geq 1,$$

constructed by composition of simpler functions called “layers of neurons". We are now going to introduce different types of layers:

Image convolutional layers. Linear mappings of the form $\textsc{Conv} : \mathbb{R}^{m\times n\times c} \to \mathbb{R}^{m\times n\times \bar{h}}$ with $m,n,c,\bar{h} \geq 1$, where m and n are the size of the input image, c is the number of channels, e.g. $c = 3$ for a colored image, and $\bar{h}$ is the number of features maps. For an image $B \in \mathbb{R}^{m\times n}$ and a kernel $K \in \mathbb{R}^{(2k+1)\times(2k+1)}$ the convolution operator $*$ is defined as

$$[K * B]_{i,j} = \sum_{p,q=-k}^{k} [K]_{k+1+p,k+1+q}[B]_{i+p,j+q}, \quad i = 1:m, \ j = 1:n,$$

with zero padding, i.e. $B_{i+p,j+q} = 0$ when indexes are out of range. This operation can be viewed as a filter scanning through image B, extracting local features that depend only on small subregions of the image. This is effective because a key property of images is that close pixels are more strongly correlated than distant ones. The scanning filter mechanism provides the basis for the invariance of the output to translations and distortions of the

input image [38]. The convolutional layer is defined as

$$[\textsc{Conv}(B)]_i = \sum_{j=1}^{c} [K]_{:,:,i,j} * [B]_{:,:,j} + [b]_i \mathbf{1}, \quad i = 1 : \bar{h},$$

where the colon index denotes that all the indexes along that dimension are considered, $\mathbf{1} \in \mathbb{R}^{m \times n}$ is the $m \times n$ matrix with all entries equal to 1, $K \in \mathbb{R}^{(2k+1)\times(2k+1)\times\bar{h}\times c}$ is a kernel matrix and $b \in \mathbb{R}^{\bar{h}}$ is a bias vector of coefficients to be tuned.

Batch normalization. Batch normalization layers (BNorm) are linear mappings used to speed up training and reduce the sensitivity to network initialization [103].

Pooling layers. Mappings used to perform down-sampling, such as

$$\textsc{Pool} : \mathbb{R}^{m\times n\times c} \to \mathbb{R}^{\lceil \frac{m}{s} \rceil \times \lceil \frac{n}{s} \rceil \times c} \quad m, n, c \geq 1,$$

$$[\textsc{Pool}(B)]_{i,j,t} = \max_{p,q=1:k} B_{s(i-1)+p, s(j-1)+q, t}$$

with $s \geq 1$ and zero padding. They improve the invariance of the output with respect to translations of the input.

Activation functions. Mappings used to introduce non-linearity, such as the rectified linear unit $[\mathrm{RELU}(x)]_i = \max(0, x_i)$ or the hyperbolic tangent $[\tanh(x)]_i = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}}$.

Dense layers. Generic linear mappings of the form $\textsc{Linear} : \mathbb{R}^{m\times n\times c} \to \mathbb{R}^{\ell}$, $n, m, c, \ell \geq 1$ defined by parameters to be tuned. They are used to separate image features extracted in the previous layers.

Softmax. We define the function $\textsc{Softmax}\mathbb{R}^{\ell} \to (0, 1)^{\ell}$, where $\ell \geq 1$ is the number output classes, $[\textsc{Softmax}(x)]_i = \frac{e^{x_i}}{\sum_{j=1}^{\ell} e^{x_j}}$. They are used to assign a probability to each class.

All of the above operations (convolution, pooling, ...) naturally extend to the three dimensional case, i.e. when dealing with images $\mathbf{B} \in \mathbb{R}^{n\times n\times n}$ made of voxels and not pixels. In practise, subsequent application of convolutional, activation and pooling layers may be used to obtain a larger degree of invariance to input transformations such as rotations, distortions, etc.

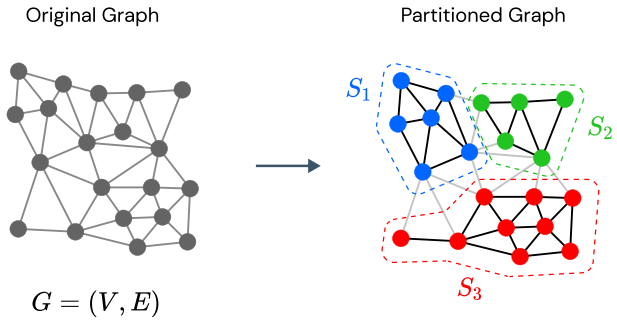

Figure 2.1: Example of graph partitioning into three sets. Left: original graph. Right: partitioned graph into three subsets S_1, S_2, S_3.

2.2. Unsupervised learning for graph partitioning

Let $N > 0$ be the number of graph nodes. Given a graph $G = (V, E)$, where $V = \{v_i\}_{i=1}^N$ and $E = \{e(v_i, v_j) : v_i, v_j \in V\}$ are the sets of nodes and the set of edges, respectively. The problem of graph partitioning consists in finding M disjoint sets of nodes $S_1, ..., S_M$ such that $\cup_{i=1}^M S_i = V$ and $\cap_{i=1}^M S_i = \emptyset$, see, e.g., Figure 2.1. We restrict our framework to undirected graphs, i.e. $e(v_i, v_j) \in E \iff e(v_j, v_i) \in E$. We will use the following notation:

- $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix, i.e. $A_{i,j} = 1$ if $e(v_i, v_j) \in E$ and 0 otherwise;
- $X \in \mathbb{R}^{N \times F}$ is the features matrix of the nodes, which may represent any information such as coordinates, where F is the number of features;
- $\mathcal{N}(v_i) = \{v_j \in V : e(v_i, v_j) \in E\}$ denotes the neighborhood of the i-th node, containing the nodes v_j directly adjacent to v_i.

We can define the *cut* of a graph, representing the number of edges connecting the disjoint sets of nodes resulting from the paritioned graph. In the case of two partitions, it can be defined as:

$$\text{cut}(S_1, S_2) = |\{e(v_i, v_j) \in E : v_i \in S_1, v_j \in S_2\}|, \tag{2.1}$$

and can be easily generalized to the case of M partitions, as

$$\text{cut}(S_1, ..., S_M) = \frac{1}{2} \sum_{k=1}^{M} \text{cut}(S_k, S_k^C) \qquad \text{with} \quad S_k^C = V \setminus S_k. \tag{2.2}$$

The *normalized cut* is defined as

$$\text{Ncut}(S_1, ..., S_M) = \sum_{k=1}^{M} \frac{\text{cut}(S_k, S_k^C)}{\text{vol}(S_k, V)}, \tag{2.3}$$

where the *volume* of the partition S_k is defined as

$$\text{vol}(S_k, V) = |\{e(v_i, v_j) \in E : v_i \in S_k, v_j \in V\}|. \tag{2.4}$$

It represents the total degree of all nodes of the k-th partition. Depending on the application of interest, different notions of *volume* can be used. The *normalized cut*, contrary to the standard *cut*, allows to take into account partitions where the number of nodes is balanced between the different sets. Let Y_{ij} be the probability for node i of belonging to partition j. The *expected cut*, given two partitions S_k and its complement S_k^C, is defined as

$$\mathbb{E}[\text{cut}(S_k, S_k^C)] = \sum_{i=1}^{N} \sum_{v_j \in \mathcal{N}(v_i)} Y_{ik}(1 - Y_{jk}) = \sum_{i=1}^{N} \sum_{j=1}^{N} Y_{ik}(1 - Y_{kj^T})A_{ij}. \tag{2.5}$$

Let D be the column vector of the degrees of the nodes, i.e. $D_i = \text{vol}(\{v_i\}, V)$. Then

$$\mathbb{E}[\text{vol}(S_k, V)] = (Y^T D)_k = \Gamma_k. \tag{2.6}$$

The *expected normalized cut* can be defined as

$$\mathbb{E}[\text{Ncut}(S_1, ..., S_M)] = \sum (Y \oslash \Gamma)(1 - Y)^T \odot A, \tag{2.7}$$

where $\oslash$ and $\odot$ denote the element-wise division and multiplication, respectively, and the summation is over all the entries of the resulting matrix.

In a unsupervised learning framework, we are given an unlabelled dataset, in our case of the form $\{(A^i, X^i)\}_{i=1}^{N_G}$ where $N_G \geq 1$ is the number of graphs in the database, for which we want to find a different representation. For the case of mesh agglomeration, these graphs represent the elements connectivity of different computational grids. We consider then a node classifier F, which in our case will be a GNN, parameterized by a set of weights W, that takes as input the adjacency matrix A and the nodes features X of a graph and outputs the probability Y_{ij} for node i of belonging to partition j. Our goal is to tune W so that F minimizes the following *loss* function

$$\mathcal{L} = \sum_{i=1}^{N_G} \ell(A^i, Y^i; W), \tag{2.8}$$

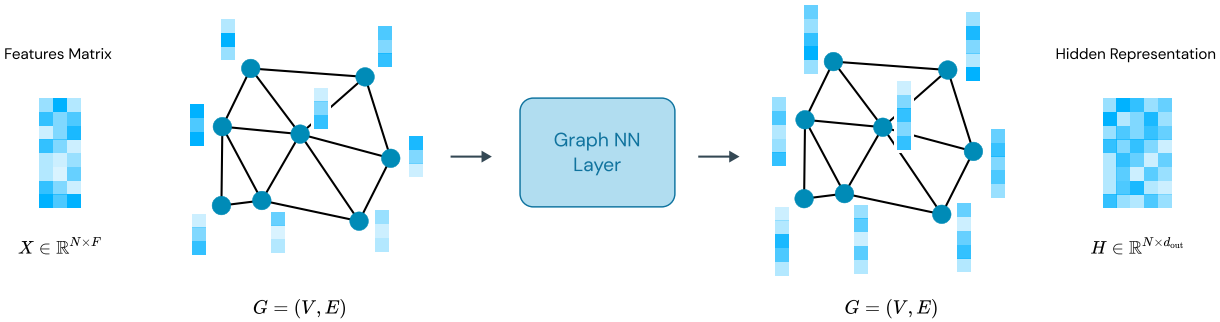

Figure 2.2: General GNN framework, consisting of an input graph G together with its features matrix X, and an output hidden representation H related to the same graph structure.

where $Y^i = F(A^i, X^i; W)$ and

$$\ell(A, Y; W) = \sum_{k=1}^{M} \sum_{i,j=1}^{N} \frac{Y_{ik}(1 - Y_{kj}^T)A_{ij}}{\Gamma_k} \tag{2.9}$$

is the expected normalized cut of the graph.

2.2.1. Graph neural networks

GNNs are deep learning architectures specifically meant to work with graph-structured data within the framework of Geometrical Deep Learning, that concerns the application of neural networks to non-Euclidean data structures. The recent success of GNNs was mainly caused by the central role played by CNNs architectures for many applications, especially computer vision [2, 114]. Indeed, GNNs can be viewed as the generalisation of CNNs, since the latter are designed to work on specific lattice graphs, while the former can applied to graph any sort of irregular sparsity pattern. In addition to the mappings defined in Section 2.1.1, the following layers can also be combined to construct the GNN architecture.

Graph convolutional layers. These layers take as input a graph $G = (V, E)$, consisting of an adjacency matrix A together with features attached to nodes, edges or the global graph, and returns a graph with the same connectivity structure while progressively transforming the information of the features, as shown in Figure 2.2. These mappings are permutation-invariant with respect to the order of the nodes. Let X denote the nodes features matrix, or the initial representations, related of the input graph G, and let H^k denote the hidden representation of those features after applying the k-th convolutional

layer. At step k, the hidden representation H_i^k for the node v_i is computed as:

$$a_i^k = \Phi^k(\{H_j^{k-1} : v_j \in \mathcal{N}(v_i)\}), \tag{2.10}$$

$$H_i^k = \Psi^k(H_i^{k-1}, a_i^k), \tag{2.11}$$

where $H^0 = X$ for the initial layer, Φ is the *aggregation function* that defines how the information coming from the neighborhood $\mathcal{N}(v_i)$ of node v_i is aggregated, and Ψ is the *combination function* that defines how the aggregated information a_i^k is combined with the one stored in v_i. Different definitions of aggregation and combination functions leads to different GNN architectures. In particular, we re-frame equations (2.10) and (2.11) by considering the mean aggregation function, as follows:

$$H_i^{l+1} = \sigma(H_i^l W_1^l + (\text{mean}_{j\in\mathcal{N}(v_i)} H_j^l) W_2^l), \tag{2.12}$$

where $\sigma(\cdot)$ is a non-linear activation function, such as the REctified Linear Unit (ReLU) or the hyperbolic tangent (tanh), and $W_1^k, W_2^k \in \mathbb{R}^{F_k \times F_{k+1}}$ are weight matrices associated to the l-th layer, representing a trainable linear transformation, where F_k and F_{k+1} are the features dimensions for the current and next layers, respectively. We refer to (2.12) as SAmpling-and-aggreGatE Convolutional (SAGECONV) layer [94] followed by activation function σ. Such layers also account for the possibility of sub-sampling the neighbourhood of a node during the aggregation process, leveraging the information coming from features to better generalize to unseen nodes, while keeping the computational cost under control.

Input normalization layer. We consider a mapping of the form INORM : $\mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F}$, where N is the number of nodes and F is the number of features, that normalizes the input feature matrix $X = [x_1|\dots|x_F]$. The normalization is performed differently for each type of feature, i.e. column-wise: for strictly positive features, such as the mesh elements areas, we simply rescale to $[0, 1]$

$$\tilde{x}_i = \frac{x_i}{\max(x_i)}, \quad i = 1, ..., F,$$

while for other features, such as barycentric coordinates, we first center them, by subtracting the mean, and then rescale to $[-1, 1]$

$$\tilde{x}_i = \frac{y_i}{\max(|y_i|)}, \quad y_i = x_i - \text{mean}(x_i), \quad i = 1, ..., F.$$

2.3. The k-means clustering algorithm

The k-means clustering algorithm [95, 115, 118], is an iterative procedure for partitioning a set of input points in $\mathbb{R}^n$, where $n \geq 1$ is the number of features, into k parts, as described in Algorithm 2.1. It is an unsupervised learning algorithm, as no labels are provided together with the data. It relies on the definition of k centroids and each point is assigned to the closest centroid. The location of the centroids is chosen in such a way to minimize the euclidean distance between centroids and points within the clusters.

Algorithm 2.1 k-means clustering

Input: set of points $\{x_i\}_{i=1}^N \subset \mathbb{R}^n$, number of clusters k.
Output: set of labels for each point $\{\ell_i\}_{i=1}^N$.

1: Randomly sample k points and label them centroids $c_1, c_2, ..., c_k$.
2: Compute labels $\{\ell_i\}_{i=1}^N$ by assigning each point to the cluster with the closest centroid in L^2 norm.
3: Compute the average of points in each cluster to obtain new centroids $c_1, c_2, ..., c_k$.
4: Repeat steps 3 and 4 until cluster assignments do not change, or the maximum number of iterations is reached.

3 | Refinement of polygonal grids using convolutional neural networks

In this chapter[1], we propose a new strategy to handle polygonal grid refinement based on Convolutional Neural Networks (CNNs). CNNs are powerful function approximators used in ML, that are particularly well suited for image classification when clearly defined rules cannot be deduced. We show that CNNs can be successfully employed to identify correctly the "shape" of a polygonal element without resorting to any geometric property. This information can then be exploited to apply tailored refinement strategies for different families of polygons. We reacall that this approach has several advantages:

- It helps preserving the mesh quality, since it can be easily tailored for different types of elements.
- It can be combined with suitable (user-defined) refinement strategies.
- It is independent of the numerical method employed to discretize the underlying differential model.
- The overall computational costs are kept low, since the training phase of a CNN is performed off line and it is independent of the differential model at hand.

The proposed approach is general and can be extended in three dimension, provided that suitable refinement strategies are available for polyhedra. We show that CNNs can be used effectively to boost either existing refinement criteria, such as the Mid-Point (MP) strategy, that consists in connecting the edges midpoints of the polygon with its centroid, and we also propose a refinement algorithm that employs pre-defined refinement rules on regular polygons. We refer to these paradigms as CNN-enhanced refinement strategies. To demonstrate the capabilities of the proposed approach we consider a second-order model problem discretized by either PolyDG methods and VEMs and we test the two CNN-

[1]The results of the thesis are original and are contained in [4].

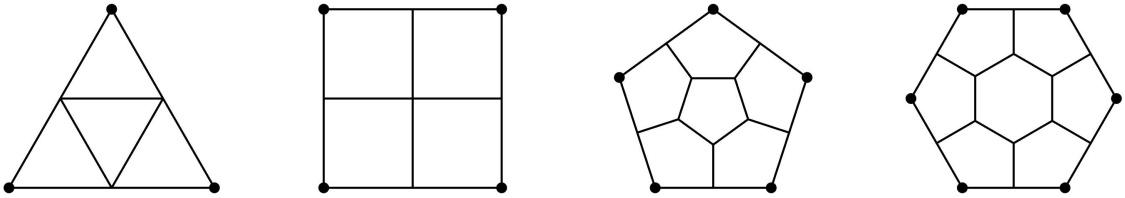

Figure 3.1: Refinement strategies for triangular, quadrilateral, pentagonal and hexagonal polygons. The vertices of the original polygon K are marked with black dots.

enhanced refinement strategies based on polygons shape recognition. For both the CNNs-enhanced refinement strategies we demonstrate their effectiveness through an analysis of quality metrics and accuracy of the discretization methods.

3.1. Polygon classification using convolutional neural networks

In this section we discuss the problem of correctly identify the "shape" of a general polygon, in order to later apply a suitable refinement strategy according to the chosen label of the classification. We start by observing that for polygons with "regular" shapes, e.g. triangles, squares, pentagons, hexagons and so on, we can define ad-hoc refinement strategies. For example, satisfactory refinement strategies for triangular and quadrilateral elements can be designed in two dimensions, as shown in Figure 3.1 (left). If the element K is a triangle, the midpoint of each edge is connected to form four triangles; if K is a square, the midpoint of each edge is connected with the centroid of the vertices of the polygon (to which we will refer as "centroid", for short), i.e. the arithmetical average of the vertices coordinates, to form four squares. For a regular polygon K with more than fours edges, suitable refinement strategies can also be devised, see e.g. [111] and Figure 3.1 (right). The idea is to:

1. Construct a suitably scaled and rotated polygon $\hat{K}$ with centroid that coincide with the centroid of the initial polygon K.
2. Connect the vertices of $\hat{K}$ with the midpoints of the edges K, forming a pentagon for each vertex of the original polygon K.

This strategy induces a partition with as many elements as the number of vertices of the original polygon K plus one. The above refinement strategies for regular polygons have the following advantages:

- They produce regular structures, thus preserving mesh quality.

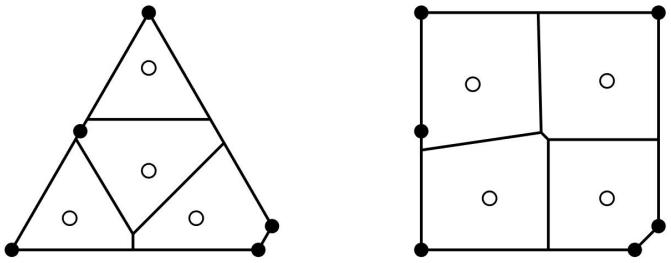

Figure 3.2: Refinement using Voronoi tessellation. The vertices of the original polygon K are marked with full dots, while seeds are marked with empty dots.

- They enforce a modular structure, as the new elements have the same structure of the original one, easing future refinements.
- They keep mesh complexity low, as they add few vertices and edges.
- They are simple to be applied and have a low computational cost.

The problem of refining a general polygonal element is still subject to ongoing research. A possible strategy consist of dividing the polygon along a chosen direction into two sub-elements [37]; this strategy is very simple and has an affordable computational cost, and it also seems to be robust and to preserve elements regularity. Because of its simplicity, however, this strategy can hardly exploit particular structures of the initial polygon.

Another possibility is to use a Voronoi tessellation [100], where some points, called seeds, are chosen inside the polygon K and each element of the new partition is the set of points which are closer to a specific seed, as shown in Figure 3.2. It is not obvious how many seeds to use and where to place them, but the resulting mesh elements are fairly rounded. The overall algorithm has a consistent but reasonable computational cost.

Another choice is to use the Mid-Point (MP) strategy, which consist in connecting the midpoint of each edge of K with the centroid of K, as shown in Figure 3.3 (top). This strategy is very simple and has a low computational cost. Modularity is enforced, in the sense that the resulting elements of the mesh are all quadrilaterals. Notice that nodes are added to adjacent elements, therefore chaining the way those elements are refined unless suitable (geometric) checks are included, as shown in Figure 3.4. The main drawback of this strategy is that it potentially disrupts mesh regularity and the number of mesh elements increases very rapidly. Therefore, which refinement strategy is the most effective depends on the problem at hand and the stability properties of the numerical scheme used for its approximation.

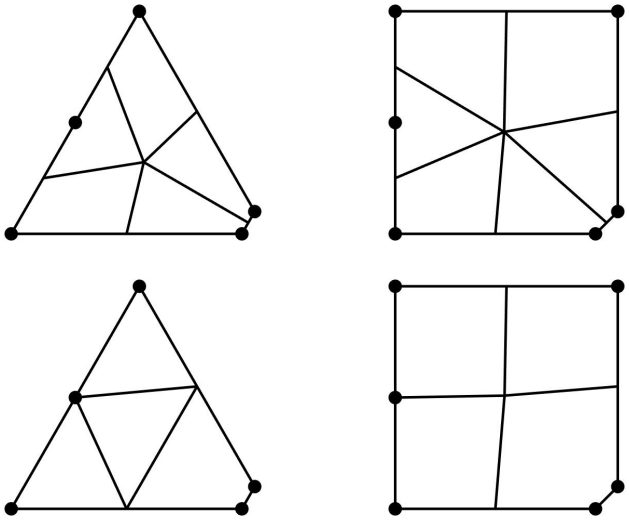

Figure 3.3: Top: the two polygons have been refined based on employing the "plain" MP rules. Bottom: the two polygons have been first classified to belong to the class of "triangular" and "quadrilateral" element, respectively, and then refined accordingly. The vertices of the original polygons are marked with black dots.

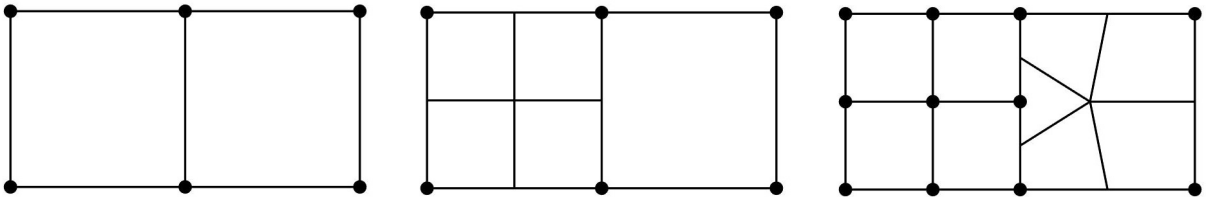

Figure 3.4: Left: the initial grid. Middle: the MP refinement strategy has been applied to the square on the left, therefore adding one node to the adjacent square on the right. Right: the MP refinement strategy has been applied also to the square on the right, dividing it into five sub-elements. The vertices of the grids that are employed to apply the MP refinement strategy of each stage are marked with black dots.

In order to suitably drive these refinement criteria, we propose to use CNNs to predict to which "class of equivalence" a given polygon belongs to. For example in Figure 3.3 (top) we show two elements refined using the MP strategy. They are clearly a quadrilateral and a pentagon, respectively, but their shapes are very similar to a triangle and a square respectively, and hence they should be refined as in Figure 3.3 (bottom). Loosely speaking, we are trying to access whether the shape of the given polygon is "more similar" to a triangle, or a square, or a pentagon, and so on. The general algorithm is the following:

1. Let $\mathcal{P} = \{P_1,\ P_2,\ \ldots\ P_m\}$ be a set of possible polygons to be refined and let $\mathcal{R} = \{R_1,\ R_2,\ \ldots\ R_n\}$ be a set of possible refinement strategies.

2. We build a classifier $F : \mathcal{P} \to \mathcal{R}$ in such a way that suitable mesh quality metrics are

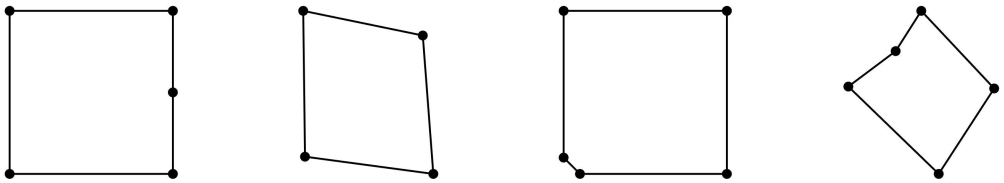

Figure 3.5: Illustrative examples of polygons belonging to the class of "squares". They have been obtained by adding small distortions or extra aligned vertices to the reference square, plus rotations and scalings in some cases. The vertices are marked with black dots.

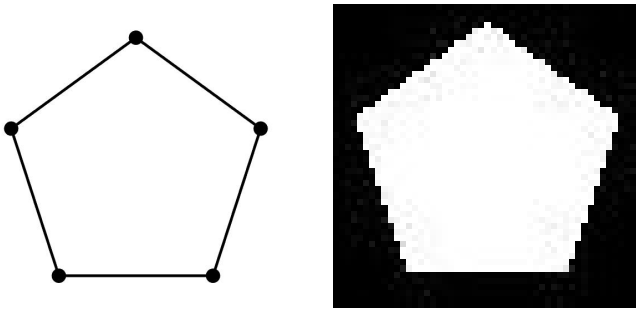

Figure 3.6: Binary image of a pentagon of size 64×64 pixels. Each pixel has value 1 (white pixel) if it is inside the polygon and 0 otherwise (black pixel). The binary images of each mesh element are then employed to classify the shape of the element, avoiding the automatic and exclusive use of geometric information.

preserved [23, 136] (quality metrics will be described later in Section 3.3). The set of all polygons mapped into the same refinement strategy is a "class of equivalence".

In principle, any classifier F may be used. However, understating the specific relevance of different geometric features of a general polygon (e.g. number of edges, area, etc...) might not be enough to operate a suitable classification. Instead, we can construct a database of polygons that can be used to "train" our classifier F. In order to construct such database we proceed as follows. Starting from the "reference" polygon in a class (e.g. the reference triangle for the class "triangles", the reference square for the class "squares" and so on) we generate a set of perturbed elements that still belong to the same class and are obtained by adding new vertices and/or adding noise to them, introducing rotations and so on. An illustrative example in the case of the class "squares" is show in Figure 3.5. Training a function with labelled data is known as "supervised learning", and CNNs are particularly well suited to deal with image classification problems in this context. Indeed, because of the inherently "image classification" nature of this problem, polygons can be easily converted into binary images: each pixel assumes value 1 if it is inside the polygon and 0 otherwise, as shown in Figure 3.6.

A binary image of a polygon can be efficiently generated as follows:

1. Scale and translate the original polygon into the reference box $(0, 1)^2$.
2. Construct a grid with the desired number of pixels.
3. From each edge of the polygon sample properly spaced points and assign value 1 to pixel containing those points. This will define an inner and an outer region of pixels.
4. Assign value 1 to each pixels in the inner region: starting from an inner pixel, recursively assign value 1 to neighbouring pixels until other pixels with value 1 are met.

For the sake of computational efficiency, this process may also be performed approximately to a certain extent. Indeed, the neural network directly learns from the image representation of the polygon and, if trained properly, it should be robust with respect to small distortions.

Identifying the "shape" of a polygonal element based on its geometrical properties is also possible using the so-called Shape Analysis (see e.g. [80]). This approach relies on rotating and translating the original element so as to minimize a suitable distance function, called "Procrustes distance", from a reference shape. The reference shape with the smallest Procrustes distance is then selected. This is a viable option in two dimensions, because the vertices of a polygon can be naturally ordered clock-wise, whereas in three dimensions it has much higher computational cost. Despite the fact that different classification algorithms may be more effective for different situations, depending on the requirements on accuracy, generalization and computational costs, we decided to use a ML based approach because:

- The behaviour of the classifier may be tuned using examples. Therefore, an explicit description of the reference shape is not required. This allows to define very general "classes of equivalence". For example, an alternative database may be generated as follows:
  - pick a polygon and apply different refinement strategies
  - rank the outcomes according to some criterion;
  - assign the label corresponding to the refinement strategy which attained the highest score.
- If a new set of elements is not classified as desired, a simple solution is to label thoses samples and include them in the training process of the CNN. Solving this problem using Shape Analysis is more complex, because the set of possible reference

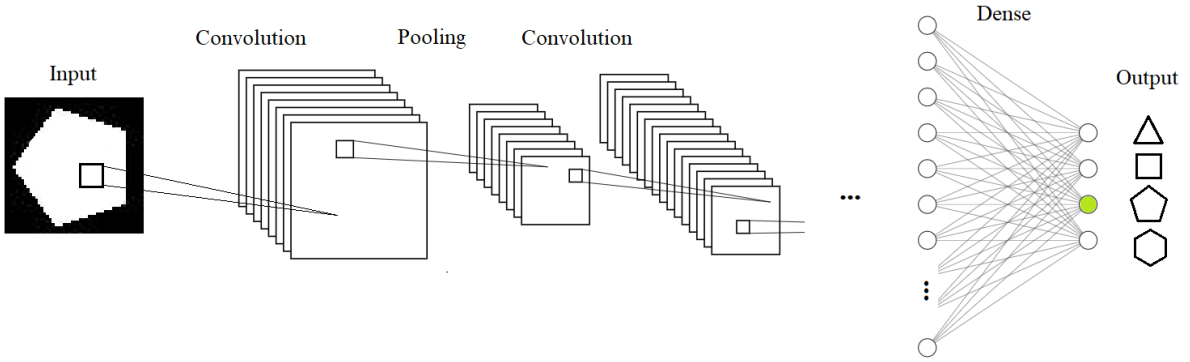

Figure 3.7: Simplified scheme of a CNN architecture employed for classification of the shape of polygons. The large squares represent the image channels after applying a convolution layer or a pooling layer, while the small squares represent layer filters scanning through every channel. After applying a convolution layer the number of channels is multiplied by the number of feature maps, while after applying a pooling layer the size of each channel diminishes accordingly. Circles represent neurons, one for each input and output, and connections represent linear dependencies. Finally, labels are represented using geometrical shapes.

shapes need to be re-design.

- Labeling a new sample using a neural network is computationally efficient, especially when dealing with multiple classes: the CNN extracts key features from the image, which are then used to assign the label. Instead, using a Shape Analysis approach the new sample is compared with every reference shape separately."

In particular, we adopt the supervised learning approach for image classification described in section 2.1. In this case, for polygons classification $B \in \{0,1\}^{m\times n}$, i.e images are binary. Moreover, in our case the classes are given by the label "triangle", "square", "pentagon" and so on. Finally, a visual representation of a CNN is shown in Figure 3.7 for the case of regular polygons classification.

3.2. Convolutional neural networks-enhanced refinement strategies

In this section we present two strategies to refine a general polygon that exploit a pre-classification step of the polygon shape. More specifically, we assume that a CNN for automatic classification of the polygon label is given. The first strategy consists in enhancing the classical MP algorithm, whereas the second strategy exploits the refinement criteria for regular polygons illustrated in Section 3.1.

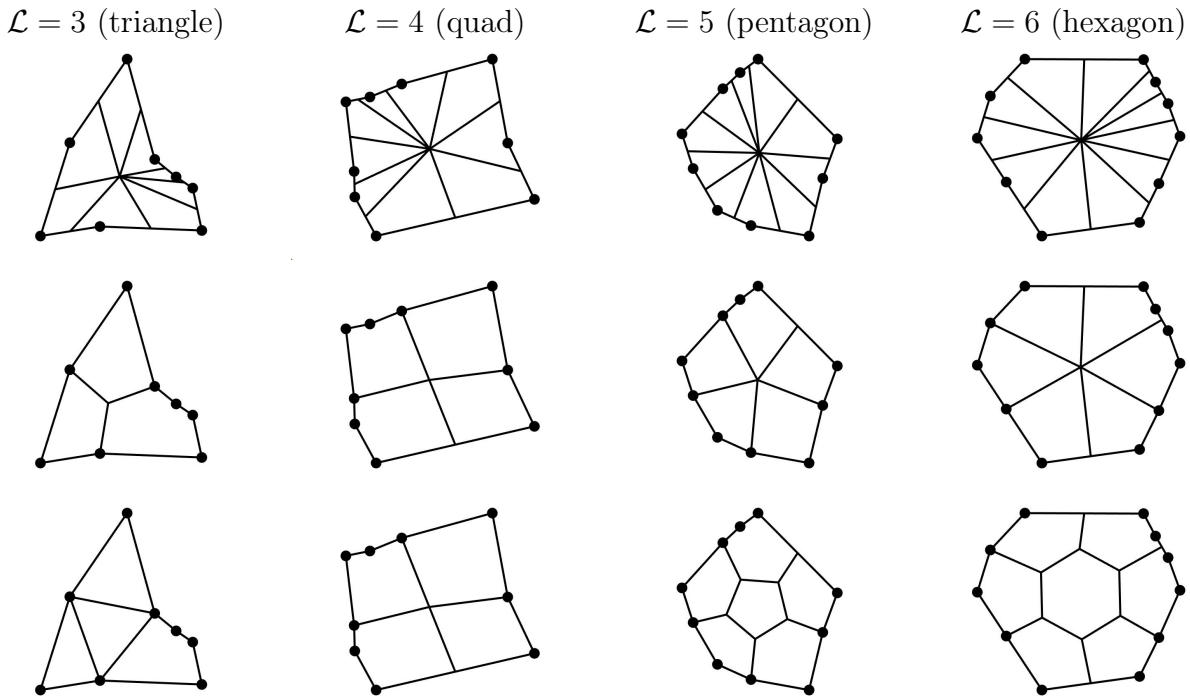

Figure 3.8: Samples of polygons refined using the MP (top), CNN-MP (middle) and CNN-RP (bottom) refinement strategies. The elements have been classified using the CNN algorithm with labels $\mathcal{L} = 3, 4, 5, 6$, respectively. The vertices of the original polygons are marked with black dots.

3.2.1. A convolutional neural network-enhanced midpoint strategy

Assume we are given a general polygon P to be refined and its label $\mathcal{L}$, obtained using a CNN for classification of polygon shapes. Here $\mathcal{L} \geq 3$ is an integer, where $\mathcal{L} = 3$ corresponds to the label "triangle", $\mathcal{L} = 4$ corresponds to the label "square", and so on. If the polygon P has a large number of (possibly aligned) vertices, applying the MP strategy may lead to a rapid deterioration of the shape regularity of the refined elements. In order to reduce the number of elements produced via refinement and to improve their quality, a possible strategy is to enhance via CNNs the MP refinement strategy, and apply the MP refinement strategy not to the original polygon P but to a suitable approximate polygon $\hat{P}$ with a number of vertices $\mathcal{L}$ identified by the CNN classification algorithm, as described in Algorithms 3.1 and 3.2.

Algorithm 3.1 CNN-enhanced Mid-Point (CNN-MP) refinement strategy

Input: polygon P

Output: partition of P into polygonal sub-elements

1: Convert P, after a proper scaling, to a binary image as shown in Figure 3.6.
2: Apply a CNN for classification of the polygon shape and obtain its label $\mathcal{L} \geq 3$.
3: Based on $\mathcal{L}$, identify the refinement points on the boundary of P, as described in Algorithm 3.2.
4: Connect the refinement points of P to its centroid c_P.

Algorithm 3.2 Identification of the refinement points

Input: polygon P, label $\mathcal{L} \geq 3$

Output: refinement points on the boundary of P

1: Build a polygon $\hat{P}$ suitably approximating P: select $\mathcal{L}$ vertices $\hat{v}_1, \hat{v}_2, ...\hat{v}_{\mathcal{L}}$, among the vertices of P, that maximize $\sum_{i,j=1}^{\mathcal{L}} ||\hat{v}_i - \hat{v}_j||$.
2: Compute the centroid c_P of P, the edge midpoints of P and the edge midpoints $\{\hat{m}_i\}_{i=1}^{\mathcal{L}}$ of $\hat{P}$.
3: For every edge midpoint $\hat{m}_i$ of $\hat{P}$: find the closest point to $\hat{m}_i$ and c_P, among the vertices and the edges midpoints of P.

Examples of refined polygons using the MP strategy are shown in Figure 3.8 (top) whereas the analogous ones obtained employing the CNN-enhanced Mid-Point (CNN-MP) refinement strategy are shown in Figure 3.8 (middle). We point out that the computational cost of the CNN-MP strategy is very low. Moreover, parallelism is enforced in a stronger sense: the CNN does not distinguish between one edge of a polygon and two aligned edges, solving the problem of refining adjacent elements pointed out in Figure 3.4.

3.2.2. A new "reference polygon" based refinement strategy

Assume, as before, that we are given a general polygon P to be refined together with its label $\mathcal{L} \geq 3$, that can be obtained employing a CNN classification algorithm. If the given polygon is a reference polygon we could refine it based on employing the refinement strategies described in Section 3.1 and illustrated in Figure 3.1, where the cases $\mathcal{L} = 3, 4, 5, 6$ are reported. Our goal is to extend these strategies so that they can be applied to general polygons. In order to do that, the idea of the algorithm is to first compute the refinement points of P, as before, and then connect them using the refinement strategy for the class $\mathcal{L}$. More precisely, our new CNN-enhanced Reference Polygon (CNN-RP) refinement

strategy is described in Algorithms 2 and 3 and illustrated in Figure 3.8 (bottom). Notice that lines 1-2-3 in Algorithm 3.3 are the same as in Algorithm 3.1.

Algorithm 3.3 CNN-enhanced Reference Polygon (CNN-RP) refinement strategy

Input: polygon P

Output: partition of P into polygonal sub-elements

1: Convert P, after a proper scaling, to a binary image as shown in Figure 3.6.
2: Apply a CNN for classification of the polygon shape and obtain its label $\mathcal{L} \geq 3$.
3: Based on $\mathcal{L}$, identify the refinement points on the boundary of P, as described in Algorithm 3.2.
4: **if** $\mathcal{L} = 3$ **then**
5: Connect the refinement points of P so as to form triangular sub-elements.
6: **end if**
7: **if** $\mathcal{L} = 4$ **then**
8: Connect the refinement points of P to its centroid, so as to form quadrilateral sub-elements.
9: **end if**
10: **if** $\mathcal{L} \geq 5$ **then**
11: Construct inside of P a suitably scaled and rotated regular polygon with $\mathcal{L}$ vertices and with the same centroid of P. Connect the vertices of the inner regular polygon with the refinement points of the outer polygon P, so as to form sub-elements as shown in Figure 3.1.
12: **end if**

This strategy can be applied off line with a low computational cost and has the advantage to enforce parallelism as each mesh element can be refined independently.

Notice that for a non-convex polygon, the CNN-RP and the CNN-MP strategies do not guarantee in general to generate a valid refined element, because the centroid could lie outside the polygon. In practise, they work well even if the polygon is "slightly" non-convex. However, in case of a non-valid refinement, it is always possible to first subdivide the polygon into two elements, possibly of comparable size, by connecting two of its vertices. In general, non-valid refinements may be detected because they are not valid partitions, i.e. when the new elements are overlapping or when they do not cover the original element. In practise, one may simply check whether or not some points lie inside the original polygon and whether or not edges are intersecting each other.

3.3. Quality metrics

In order to evaluate the quality of the proposed refinement strategies, we recall some of the quality metrics introduced in [23]. The diameter of a domain $\mathcal{D}$ is defined, as usual, as $\text{diam}(\mathcal{D}) := \sup\{|x-y|,\ x,y \in \mathcal{D}\}$. Given a polygonal mesh, i.e. a set of non-overlapping polygonal regions $\{P_i\}_{i=1}^{N_P}$, $N_P \geq 1$, that covers a domain Ω, we can define the mesh size $h = \max_{i=1:N_P} \text{diam}(P_i)$. For a mesh element P_i, the Uniformity Factor (UF) is defined as $\text{UF}_i = \frac{\text{diam}(P_i)}{h}$, $i = 1, ..., N_P$. This metric takes values in $[0, 1]$.

For a polygon P, we also introduce the following quality metrics, taken from [23]:

1. Circle Ratio (CR): ratio between the radius of the inscribed circle and the radius of the circumscribed circle of P:
$$\frac{\max_{\{B(r)\subset P\}} r}{\min_{\{P\subset B(r)\}} r},$$
where $B(r)$ is a ball of radius r. For the practical purpose of measuring the roundness of an element the radius of the circumscribed circle has been approximated with $\text{diam}(P)/2$.

2. Area-Perimeter Ratio (APR):
$$\frac{4\pi\ \text{area}(P)}{\text{perimeter}(P)^2}.$$

3. Minmum Angle (MA): minimum inner angle of P, normalized by 180°.

4. Edge Ratio (ER): ratio between the shortest and the longest edge of P.

5. Normalized Point Distance (NPD): minimum distance between any two vertices of P, divided by the diameter of the circumscribed circle of P.

These metrics are scale-independent and take values in $[0, 1]$. The more regular the polygons are, the larger CR, APR and MA are. Large values of ER and NPD also indicate that the polygon is well proportioned and not skewed. However, small values of ER and NPD do not necessarily mean that the element is not shaped-regular, as shown in Figure 3.9.
Notice that some quality metrics are not as important as others. For example, for several polytopal methods it is known that a small Edge Ratio does not necessarily deteriorates the accuracy of the method [35, 44, 79, 121].

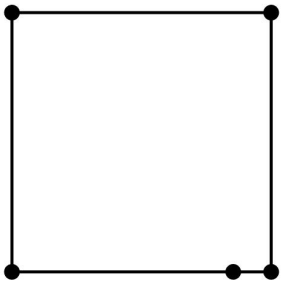

Figure 3.9: A polygon with a small edge. Although its shape is regular, ER and NPD metrics assume small values, depending on the size of the smallest edge.

3.4. Convolutional neural network training

The CNN architecture we used for polygons classification is given by
$\text{CNN} : \{0,1\}^{64\times 64} \to (0,1)^{\ell}$, where

$$\begin{aligned}\text{CNN} = \textsc{Conv}(k=1,\bar{h}=2) \ &\to \text{BN}\textsc{orm} \to \text{ReLU} \to \textsc{Pool}(k=2,s=2) \to \\ &\textsc{Conv}(k=1,\bar{h}=4) \to \text{BN}\textsc{orm} \to \text{R}\textsc{eLU} \to \textsc{Pool}(k=2,s=2) \to \\ &\textsc{Conv}(k=1,\bar{h}=8) \to \text{BN}\textsc{orm} \to \text{R}\textsc{eLU} \to \textsc{Pool}(k=2,s=2) \to \\ &\textsc{Conv}(k=1,\bar{h}=16) \to \text{BN}\textsc{orm} \to \text{R}\textsc{eLU} \to \textsc{Pool}(k=2,s=2) \to \\ &\textsc{Conv}(k=1,\bar{h}=32) \to \text{BN}\textsc{orm} \to \text{R}\textsc{eLU} \to \textsc{Pool}(k=2,s=2) \to \\ &\textsc{Conv}(k=1,\bar{h}=64) \to \text{BN}\textsc{orm} \to \text{R}\textsc{eLU} \to \textsc{Linear} \to \textsc{Softmax},\end{aligned}$$

where $k, \bar{h}, s$ are defined as in Section 2.1.1. This choice of a deep architecture is motivated by the fact that many convolutional and pooling layers are needed to enforce the invariance to any possible rotation of the input polygon. The size of the images was selected to be 64×64 pixels, i.e. a resolution large enough to apply five pooling layers, whose effect is to down-sample the image. It was empirically observed that a larger resolution was not needed for the datasets we are going to consider. This is motivated by the fact that the approximation properties of neural networks generally depend on the dimension of the manifold where data are, and not on the dimension of the input space (see e.g. [126]). A larger resolution maybe needed in order to be able to differentiate between polygons characterized by smaller angles, e.g. to correctly distinguish between a regular polygon with 9 sides an one with 10 sides.
For each class, we generated 20.000 images transforming regular polygons by adding edges and noise to the vertices. Since we will consider datasets with $\ell = 4$ and $\ell = 6$ classes, the total number of images will be respectively 80.000 and 120.000. We set training, validation and test sets equal to 60%-20%-20% of the whole dataset, respectively. To train

Figure 3.10: Confusion matrices for polygons classification. Left: the number of classes is $\ell = 6$ and the target classes vary from $\mathcal{L} = 3$ (triangles) to $\mathcal{L} = 8$ (octagons). Right: the number of classes is $\ell = 4$ and the target classes vary from $\mathcal{L} = 3$ (triangles) to $\mathcal{L} = 6$ (hexagons). The prediction accuracy for each target class decreases as more target classes are considered.

the neural networks we used the Adam (adaptive moment estimation) optimizer, see e.g. [109].

Initially we selected the number of target classes to be equal to $\ell = 6$, i.e. polygons are sampled from triangles to octagons. We show the confusion matrix in Figure 3.10 (left). The same results obtained with $\ell = 4$, i.e. target classes varying from $\mathcal{L} = 3$ (triangles) to $\mathcal{L} = 6$ (hexagons), are shown in Figure 3.10 (right). From these results it seems that the prediction accuracy is better in the case of a smaller set of target classes. This is expected, as for example a regular octagon is much more similar, in terms of angles amplitude and edges length, to a regular heptagon than to a regular triangle. Moreover, for polygons with many edges more pixels might be required in order to appreciate the differences between them. In the following numerical experiments we have decided to choose $\ell = 4$, as this choice seems to balance the effectiveness of our classification algorithm with the computational cost. We also remark that for the following reasons:

- Refining heptagons and octagons as if they were hexagons does not seem to affect dramatically the quality of the refinement.

- Ad-hoc refinement strategies for polygons with many edges seem to be less effective because more sub-elements are produced.
- A considerable additional computational effort might be required to include more classes.
- The more classes we use, the easier the possibility of a misclassification error is and hence to end up with a less robust refinement procedure.

Considering polygon classes ranging from triangles to hexagons yields a satisfactory accuracy of 93.2% as shown in Figure 3.10 (right). Consider also that an accuracy close to 100% is not realistic because when distorting for example a pentagon by adding noise to its vertices there is a considerable probability to turn it into something indistinguishable from a slightly distorted "square", which therefore we would like to classify as "square" even though the original label is "pentagon".
"Hexagons" and "pentagons" are the most misclassified elements, when classified as "pentagons" and "squares" by the CNN, respectively. In both cases, applying the wrong refinement strategy should not yield dramatic differences in terms of elements quality, as for example it would instead misclassifying an "hexagon" as a "triangle". Moreover, an effective refinement algorithm should be robust to some extent with respect to misclassification errors. These remarks will be confirmed by numerical experiments, shown in Section 3.6
Thanks to the limited number of dataset samples and network parameters, the whole algorithm (dataset generation, CNN training and testing) took approximately six minutes using MATLAB2019b on a Windows OS 10 Pro 64-bit, Intel(R) Core(TM) i7-8750H CPU (2.20GHz / 2.21GHz) and 16 GB RAM memory.
Improving the accuracy of the CNN using more data and/or a network with more layers is possible. However, the dataset will probably not be fully representative of mesh elements on which the CNN is going to be applied, because it is artificially generated. Therefore, attempting to achieve a very high accuracy may cause the CNN to overfit such dataset and hence to generalize poorly on real mesh elements. A different approach would be using directly real mesh elements to train the CNN. However, in this case one should be able to design a strategy to label elements automatically.

3.5. Validation on a set of polygonal meshes

In this section we compare the performance of the proposed algorithms. We consider four different coarse grids of the domain $(0, 1)^2$: a grid of triangles, a Voronoi grid, a smoothed Voronoi grid obtained with Polymesher [139], and a grid made of non-convex elements. In

# mesh elements	triangles	Voronoi	smoothed Voronoi	non-convex
initial grid	32	9	10	14
MP	6371	2719	3328	4502
CNN-RP	2048	685	784	1574
CNN-MP	1201	536	667	849

Table 3.1: Final number of elements for each mesh shown in Figure 3.11: a grid of triangles, a Voronoi grid, a smoothed Voronoi grid and a grid made of non-convex elements have been uniformly refined using the Midp-Point (MP), the CNN-enhanced Mid-Point (CNN-MP) and the CNN-enhanced Reference Polygon (CNN-RP) strategies. On average, the MP strategy produced 3 or 4 times more elements than the CNN-RP strategy, and 5 times more elements than CNN-MP strategy.

Figure 3.11 these grids have been successively refined uniformly, i.e. each mesh element has been refined, for three times using the MP, the CNN-MP and the CNN-RP strategies. The final number of mesh elements is shown in Table 3.1. We observe that on average the MP strategy produced 3 or 4 times more elements than the CNN-RP strategy, and 5 times more than CNN-MP strategy.

In Figure 3.12 we show the computed quality metrics described in Section 3.3 on the grids of Figure 3.11 (triangles, Voronoi, smoothed Voronoi, non-convex). Despite the fact that the performance are considerably grid dependent, the CNN-RP strategy and the CNN-MP strategy seem to perform in a comparable way. Moreover, the CNN-RP and the CNN-MP strategies perform consistently better than the MP strategy, since their distributions are generally more concentrated toward the value 1.

3.6. Testing CNN-based refinement strategies with PolyDG and virtual elements discretizations

In this section we test the effectiveness of the proposed refinement strategies, to be used in combination with polygonal finite element discretizations. To this aim we consider PolyDG and Virtual Element discretizations of the following model problem: find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v \quad \forall v \in H_0^1(\Omega), \tag{3.1}$$

with $f \in L^2(\Omega)$ a given forcing term. The workflow is as follows:

1. Generate a grid for Ω.
2. Compute numerically the solution of problem (3.1) using either the VEM [30, 31,

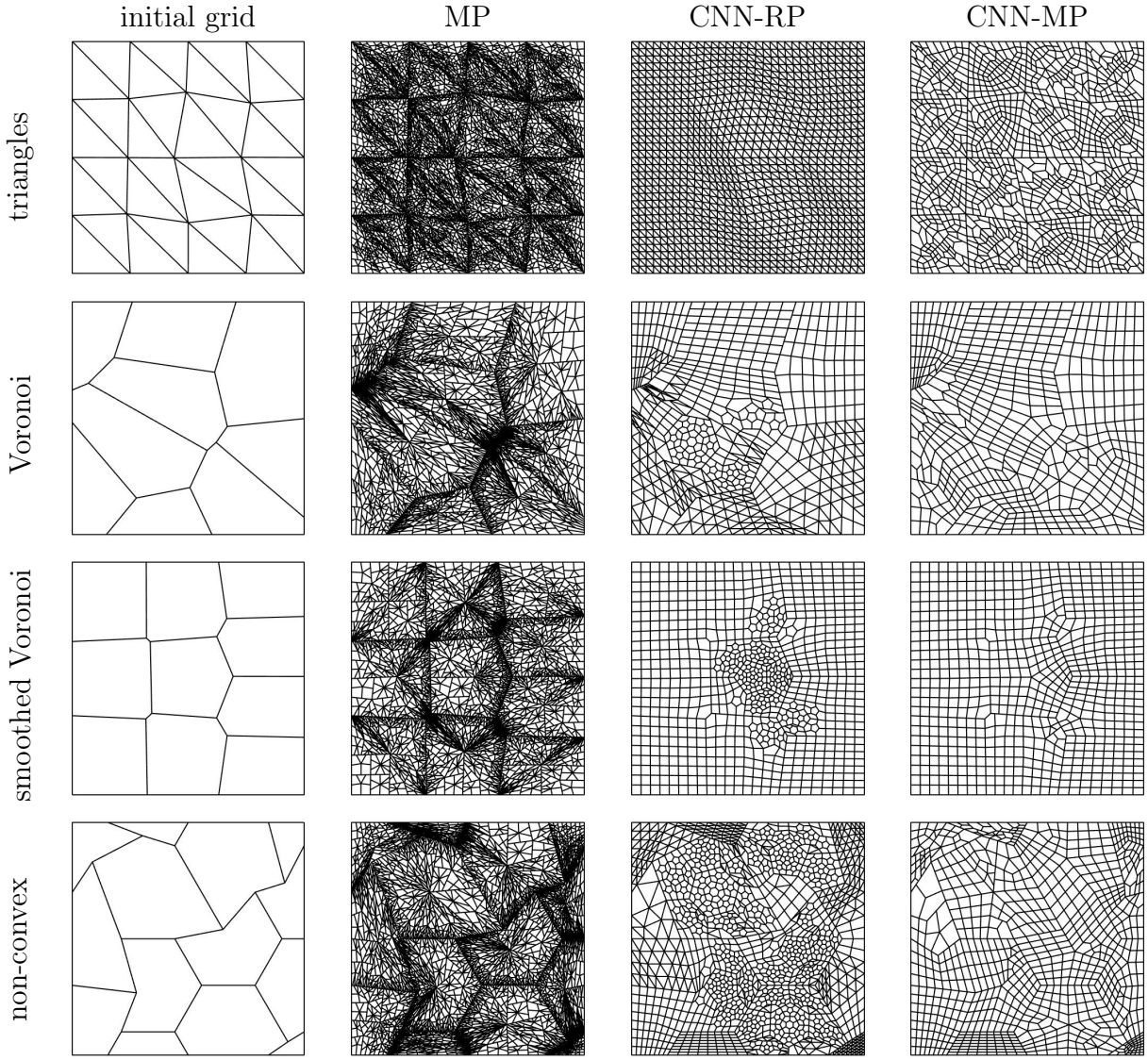

Figure 3.11: In the first column, coarse grids of the domain $\Omega = (0,1)^2$: a grid of triangles, a Voronoi grid, a smoothed Voronoi grid, and a grid made of non-convex elements. Second to fourth columns: refined grids obtained after three steps of uniform refinement based on employing the MP (second column), the CNN-RP (third column) and the CNN-MP (fourth column) strategies. Each row corresponds to the same initial grid, while each column corresponds to the same refinement strategy.

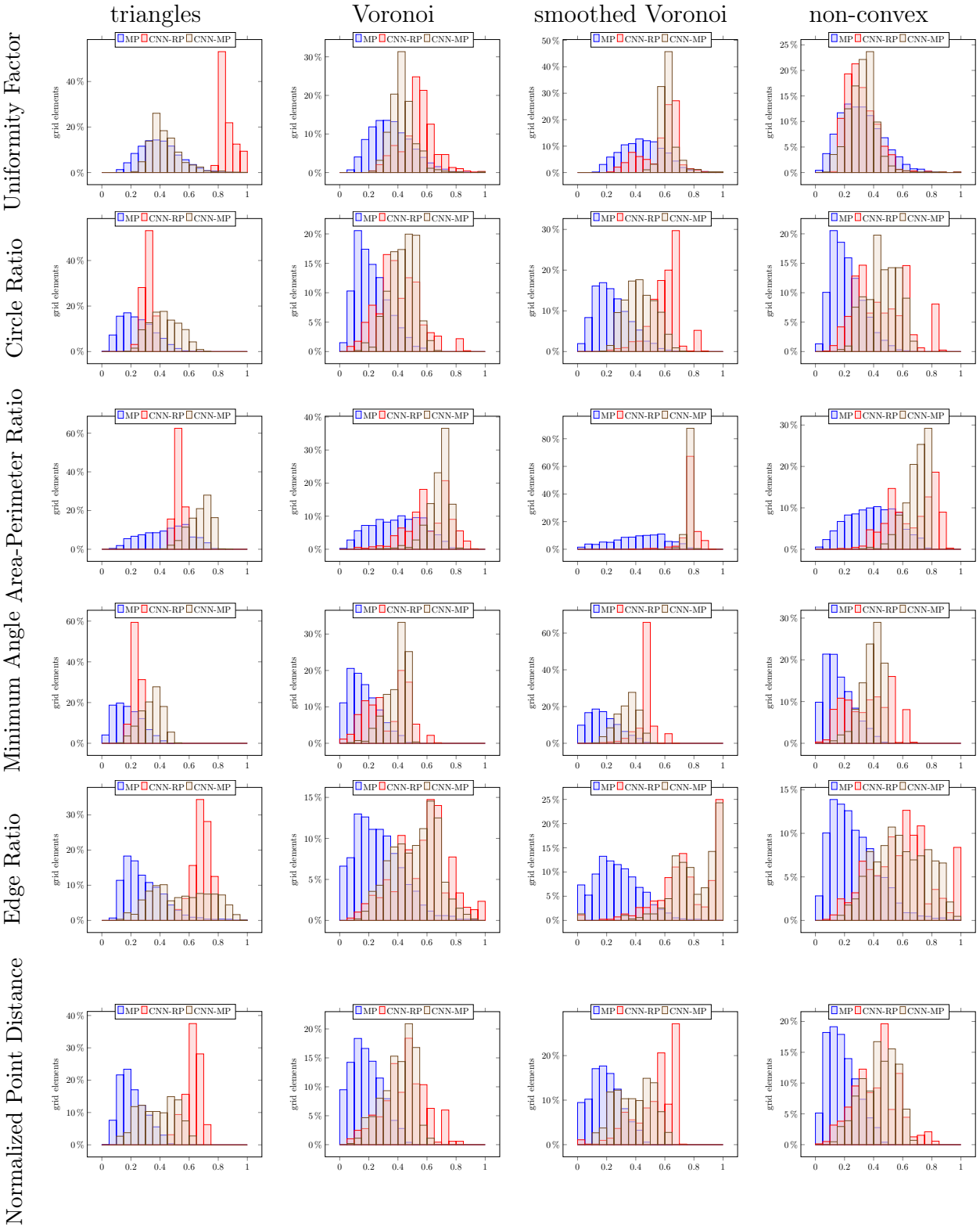

Figure 3.12: Computed quality metrics (Uniformity Factor, Circle Ratio, Minimum Angle, Edge Ratio and Normalized Point Distance) for the refined grids reported in Figure 3.11 (second to fourth column) and obtained based on employing different refinement strategies (MP, CNN-MP, CNN-RP).

33, 34] or the PolyDG method [10, 13, 28, 50, 53, 99].

3. Compute the error. In the VEM case the error is measured using the discrete H_0^1-like norm (see [31], for details), while in the PolyDG case the error is computed using the DG norm (see [22, 63], for details)

$$\|v\|_{\mathrm{DG}}^2 = \sum_P \|\nabla v\|_{L^2(P)}^2 + \sum_F \|\gamma^{1/2}[\![v]\!]\|_{L^2(F)}^2,$$

where γ is the stabilization function (that depends on the discretization parameters and is chosen as in [50]), P is a polygonal mesh element and F is an element face. The jump operator $[\![\cdot]\!]$ is defined as in [22].

4. Use the fixed fraction refinement strategy to refine a fraction r of the number of elements. To refine the marked elements we employ one of the proposed strategies. Here, in order to investigate the effect of the proposed refinement strategies, we did not employ any a posteriori estimator of the error, but we computed element-wise the local error based on employing the exact solution.

3.6.1. Uniformly refined grids

When $r = 1$, the grid is refined uniformly, i.e. at each refinement step each mesh element is refined. The forcing term f in (3.1) is selected in such a way that the exact solution is given by $u(x, y) = \sin(\pi x)\sin(\pi y)$. The grids obtained after three steps of uniform refinement are those already reported in Figure 3.11. In Figure 3.13 we show the computed errors as a function of the number of degrees of freedom. We observe that the CNN-enhanced strategies (both MP and RP ones) outperform the plain MP rule. The difference is more evident for VEMs than for PolyDG approximations. This different sensitivity to mesh distortions may be due to the fact that VEM are hybrid methods with unknowns on the elements boundary.

Generating the binary image of a polygon has a computational cost that scales linearly with the number of edges of a polygon and with the squared root of the total number of pixels. Evaluating online the CNN has a computational cost that depends on the number pixels of the input image (and on the classification problem at hand). With our current implementation and for the considered benchmarks, classifying the shape of every mesh element takes on average approximately 3% the time of solving the numerical problem over the refined grid. However, this ratio will decrease if meshes with more elements are considered, because except for particular cases the average number of edges of a mesh element will remain approximately constant and the number of pixels will remain constant,

while solving the numerical problem has a cost which in general scales more than linearly with the number of degrees of freedom.

In Figure 3.14 we compare the performance of the currently used CNN with accuracy 93% and of a CNN with accuracy 80%. As we can see, the PolyDG method does not seem very sensitive to the CNN accuracy, while the VEM seems more sensitive but performance seem not to always improve consistently over the selected grids.

3.6.2. Adaptively refined grids

In this case we selected $r = 0.3$. The forcing term f in (3.1) is selected in such a way that the exact solution is $u(x, y) = (1 - e^{-10x})(x - 1)\sin(\pi y)$, that exhibits a boundary layer along $x = 0$. Figure 3.15 shows the computed grids after three steps of refinement for the PolyDG case. Very similar grids have been obtained with Virtual Element discretizations.

In Figure 3.16 we show the computed errors as a function of the number of degrees of freedom for both Virtual Element and PolyDG discretizations. The CNN-enhanced strategies (both MP and RP ones) outperform the plain MP rule. The difference is more evident for VEMs than for PolyDG approximations.

3.6.3. Application to an advection-diffusion problem

We now consider the following advection-diffusion problem:

$$\begin{cases} -\Delta u + \operatorname{div}(\boldsymbol{\beta} u) = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

where $\Omega = (0, 1)^2$ and $\boldsymbol{\beta} = [1\ 0]^T$ is a constant velocity filed. We choose the forcing term $\mathbf{f}$ in such a way that the exact solution is $(x - 1)(1 - \exp(2x))\sin(\pi y)$, and solve this problem using PolyDG method, both in the uniform refinement case (coarse grids in the first column of Figure 3.11), and in the adaptive refinement case with fixed refinement fraction of 30% (fine grids in the first column of Figure 3.15). In Figure 3.17 we show the computed errors using the DG norm as a function of the number of degrees of freedom. As we can see, the CNN-enhanced strategies (both MP and RP ones) perform better than the plain MP rule.

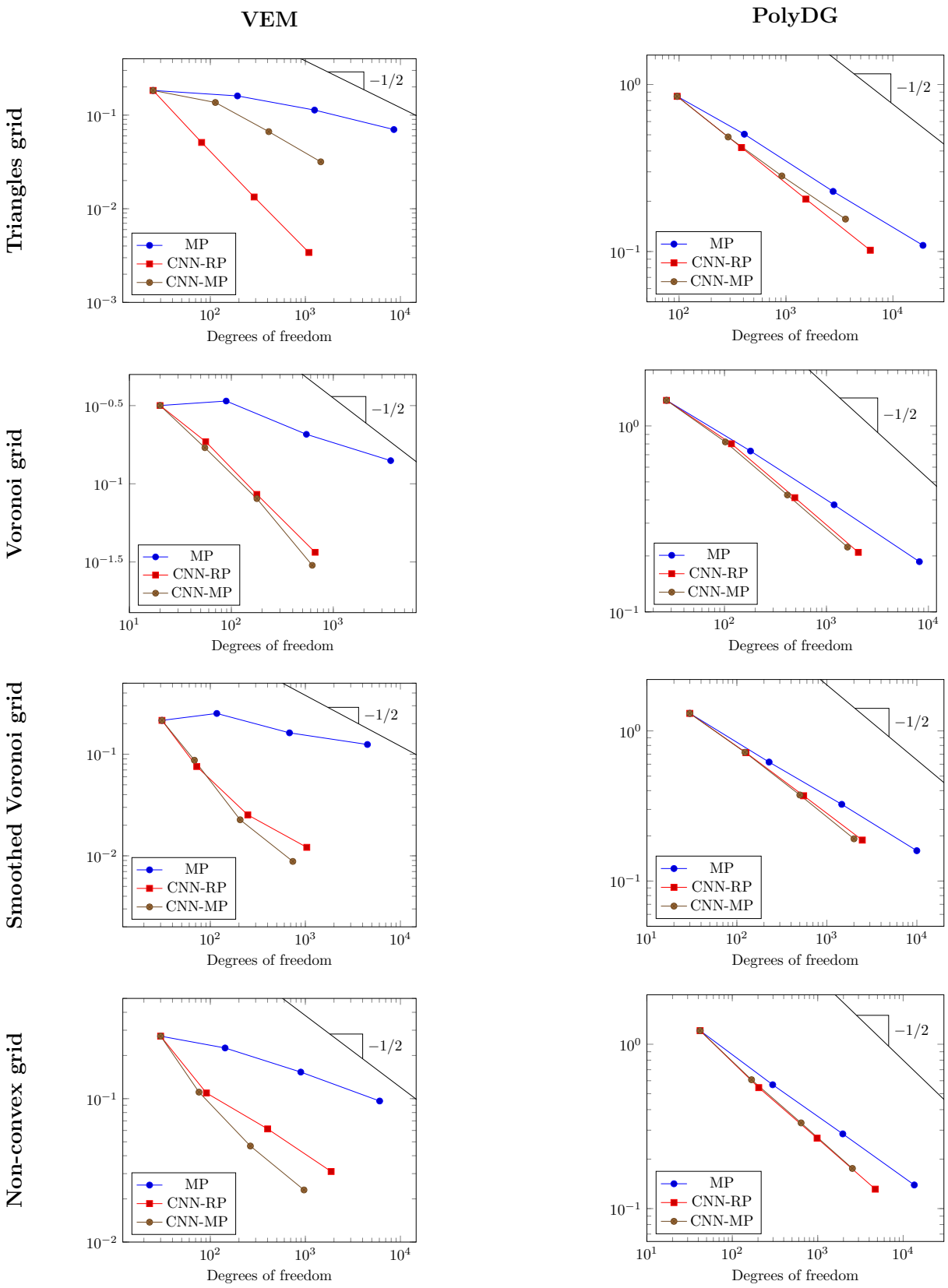

Figure 3.13: Test case of Section 3.6.1. Computed errors as a function of the number of degrees of freedom. Each row corresponds to the same initial grid (triangles, Voronoi, smoothed Voronoi, non-convex) refined uniformly with the proposed refinement strategies (MP, CNN-RP and CNN-MP), while each column corresponds to a different numerical method (VEM left and PolyDG right).

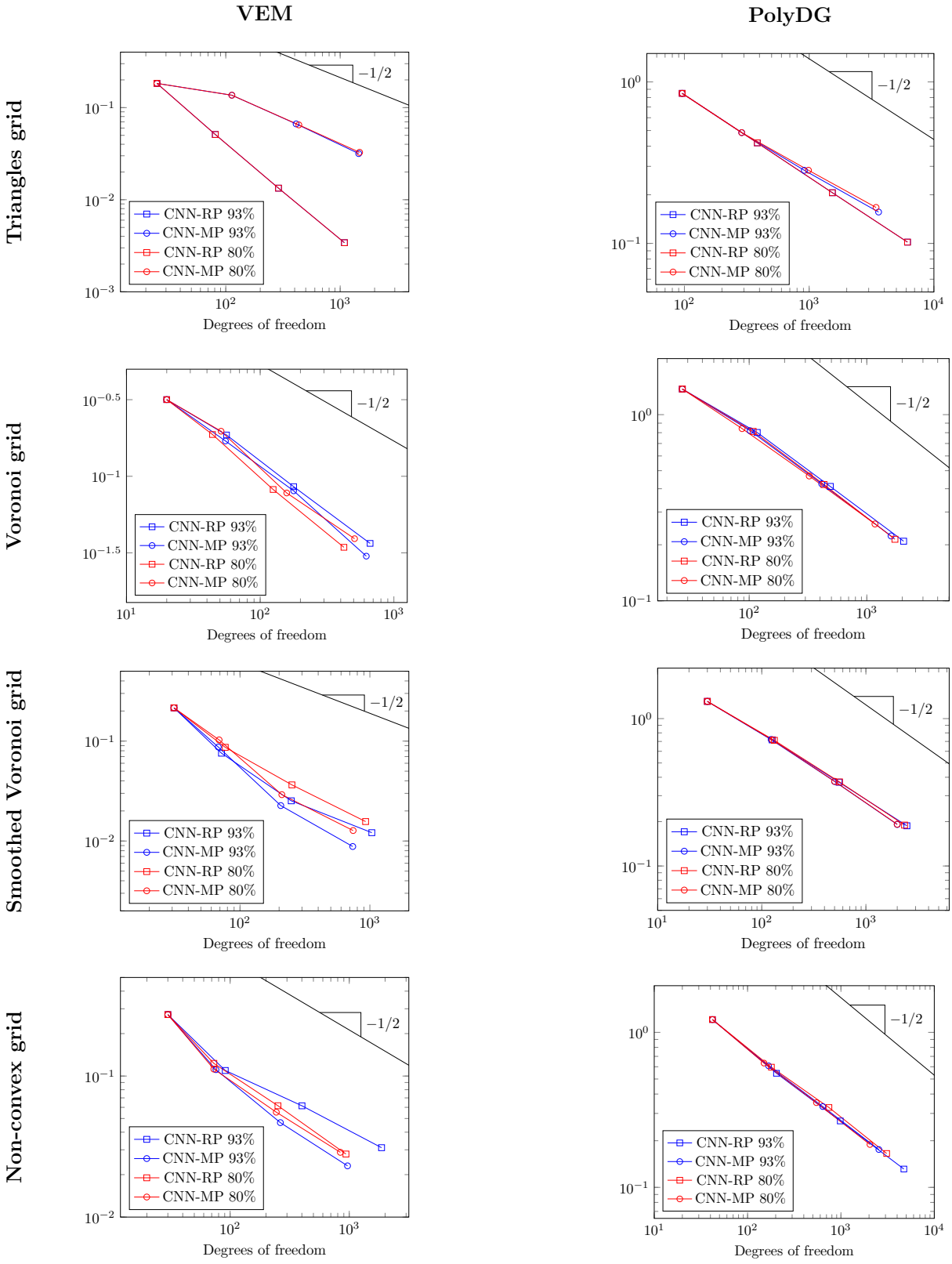

Figure 3.14: Test case of Section 3.6.1. Performance comparison of the currently used CNN with accuracy 93% and of a CNN with accuracy 80%. Computed errors as a function of the number of degrees of freedom. Each row corresponds to the same initial grid (triangles, Voronoi, smoothed Voronoi, non-convex) uniformly refined with the CNN-RP and CNN-MP refinement strategies, respectively. Each column corresponds to a different numerical method (VEM left and PolyDG right).

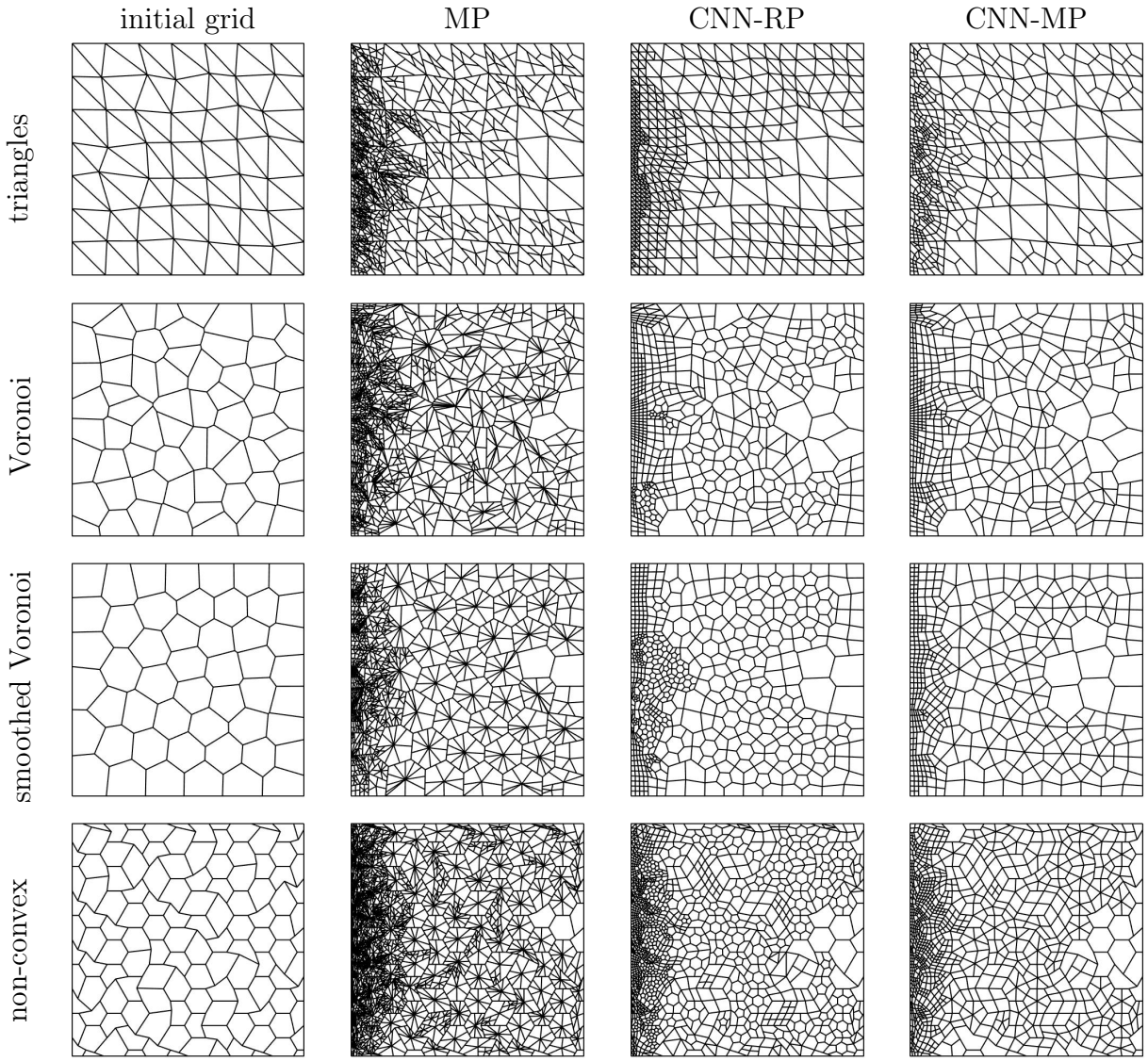

Figure 3.15: Adaptively refined grids for the test case of Section 3.6.2. Each row corresponds to the same initial grid (triangles, Voronoi, smoothed Voronoi, non-convex), while the second-fourth columns correspond to the different refinement strategies (MP, CNN-RP, CNN-MP). Three successively adaptive refinement steps have been performed, with a fixed fraction refinement criterion (refinement fraction r set equal to 30%).

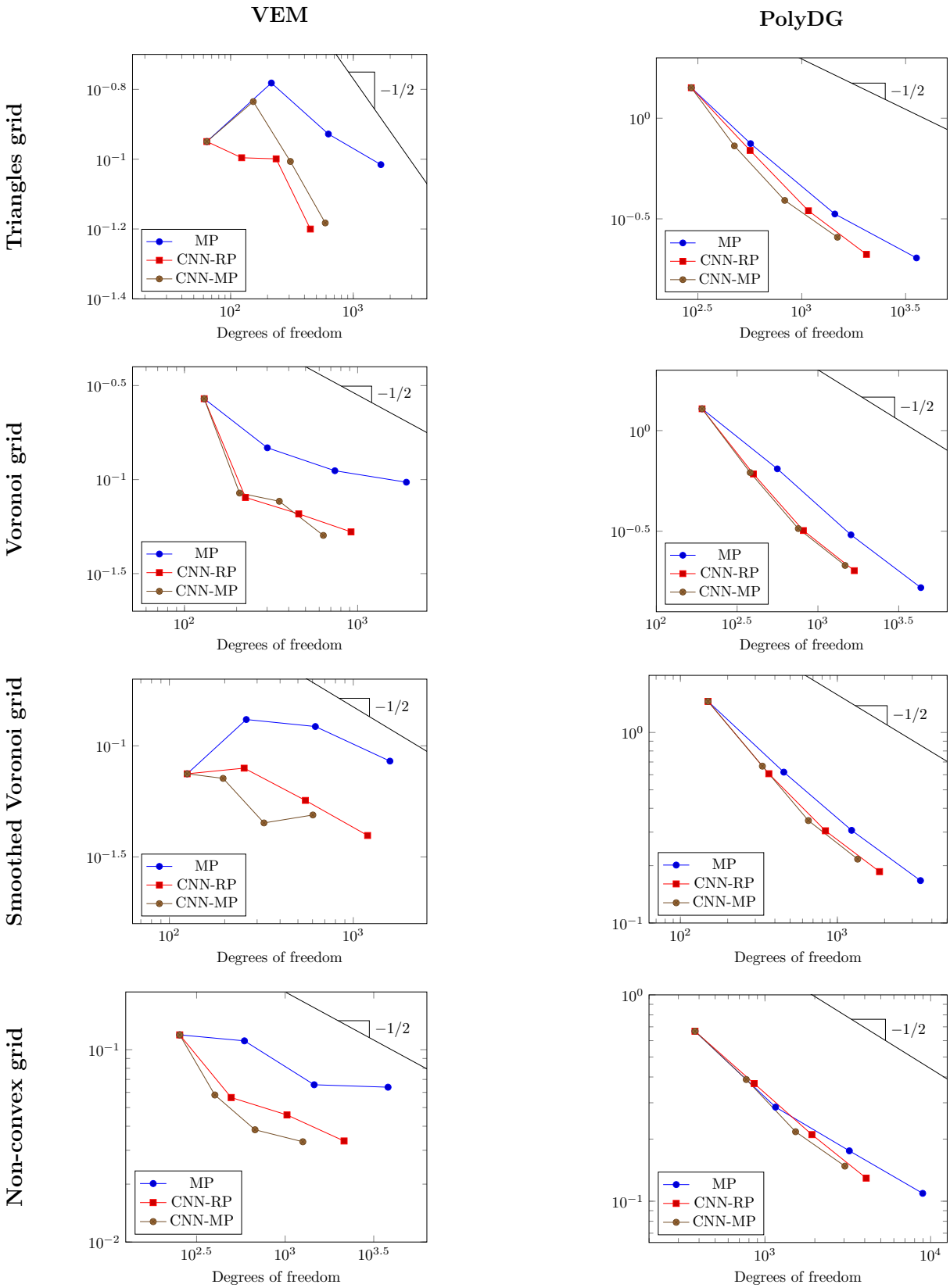

Figure 3.16: Test case of Section 3.6.2. Computed errors as a function of the number of degrees of freedom. Each row corresponds to the same initial grid (triangles, Voronoi, smoothed Voronoi, non-convex) refined adaptively with a fixed fraction refinement criterion (refinement fraction r set equal to 30%) with different strategies (MP, CNN-RP and CNN-MP), while each column corresponds to a different numerical method (VEM left and PolyDG right).

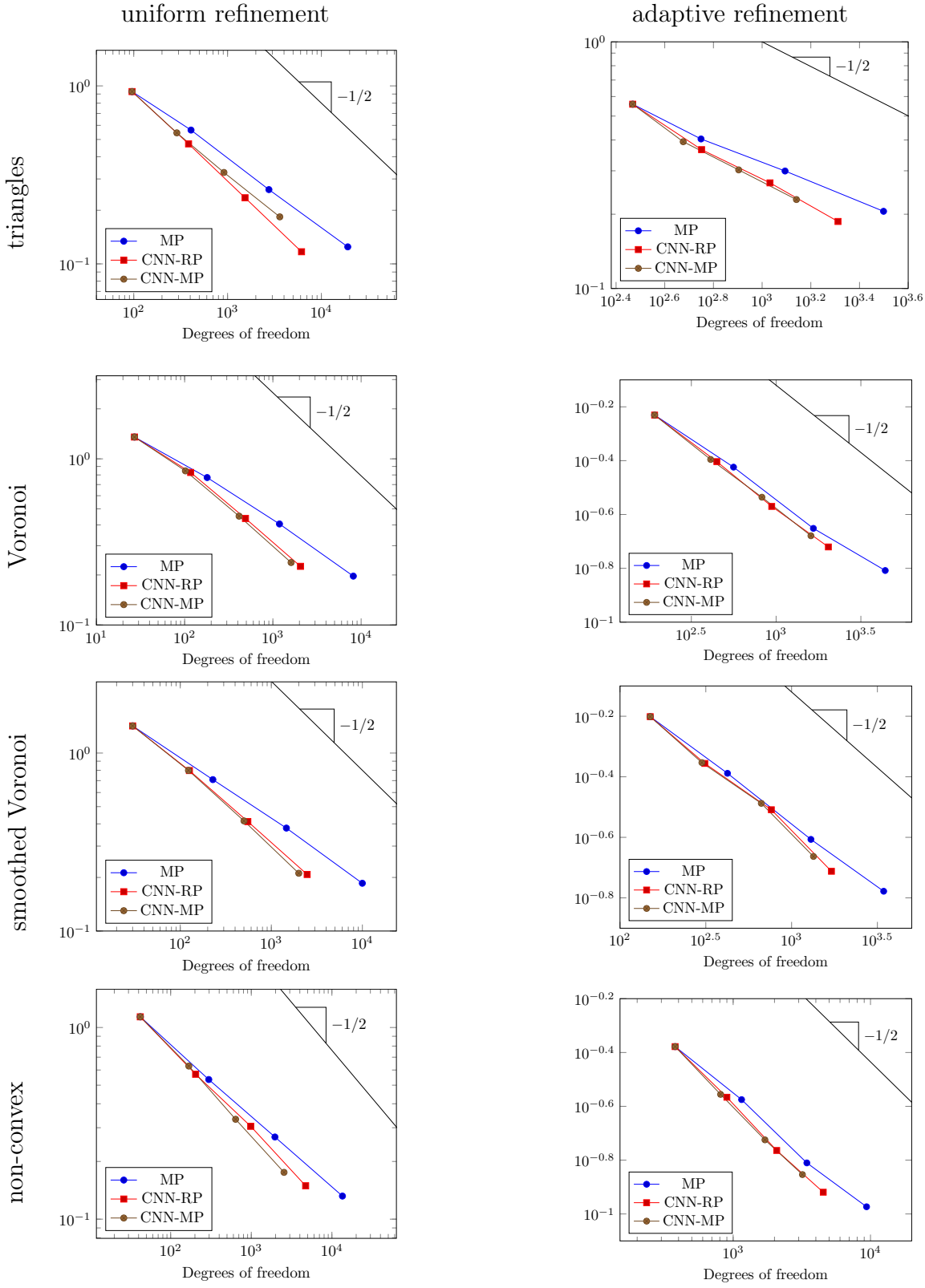

Figure 3.17: Advection-diffusion problem of Section 3.6.3. Computed errors using the DG norm as a function of the number of degrees of freedom. Each row corresponds to the same initial grid (triangles, Voronoi, smoothed Voronoi, non-convex) refined uniformly with the proposed refinement strategies (MP, CNN-RP and CNN-MP). First column: uniform refinement case over the coarse grids in the first column of Figure 3.11. Second column: adaptive refinement case with fixed refinement fraction of 30% of mesh elements over the fine grids in the first column of Figure 3.15.

3.6.4. Application to the Stokes problem

We now consider the following Stokes problem, which describes the motion of an incompressible viscous flow:

$$\begin{cases} \partial_t \mathbf{u} - \Delta \mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = 0 & \text{in } \Omega \\ \mathbf{u} = \mathbf{0} & \text{on } \partial\Omega \end{cases}$$

where $\Omega = (0,1)^2$, $\mathbf{u} : \Omega \to \mathbb{R}^2$ is the velocity, $p : \Omega \to \mathbb{R}$ is the pressure, $\mathbf{f} : \Omega \to \mathbb{R}^2$ is the forcing term. We choose $\mathbf{f}$ in such a way that the exact solution is

$$\boldsymbol{u} = \begin{bmatrix} -\cos(2\pi x)\sin(2\pi y) \\ \sin(2\pi x)\cos(2\pi y) \end{bmatrix}, \quad p = 1 - e^{-x(x-1)(x-0.5)^2 - y(y-1)(y-0.5)^2},$$

and solve it using PolyDG method, both in the uniform refinement (coarse grids in the first column of Figure 3.11), and in the adaptive refinement case with fixed refinement fraction of 40% (fine grids in the first column of Figure 3.15). In Figure 3.18 we show the computed errors in the H^1-broken norm of the velocity and L^2-norm of the pressure as a function of the number of degrees of freedom. As it is clear from the results of Figure 3.18, the CNN-enhanced strategies (both MP and RP ones) perform better than the plain MP rule.

3.7. Concluding remarks

We proposed a new paradigm based on CNNs to enhance both existing refinement criteria and new refinement procedures, withing polygonal finite element discretizations of partial differential equations. In particular, we introduced two refinement strategies for polygonal elements, named "CNN-RP strategy" and "CNN-MP strategy". The former proposes ad-hoc refinement strategies based on reference polygons, while the latter is an improved version of the known MP strategy. These strategies exploit a CNN to suitably classify the "shape" of a polygon in order to later apply an ad-hoc refinement strategy. This approach has the advantage to be made of interchangeable pieces: any algorithm can be employed to classify mesh elements, as well as any refinement strategy can be employed to refine a polygon with a given label.

We have shown that correctly classifying elements shape based on employing CNNs can improve consistently and significantly the quality of the grids and the accuracy of polygonal finite element methods employed for the discretization. Specifically, this has been measured in terms of less elements produced on average at each refinement step, in terms of improved quality of the mesh elements according to different quality metrics, and in

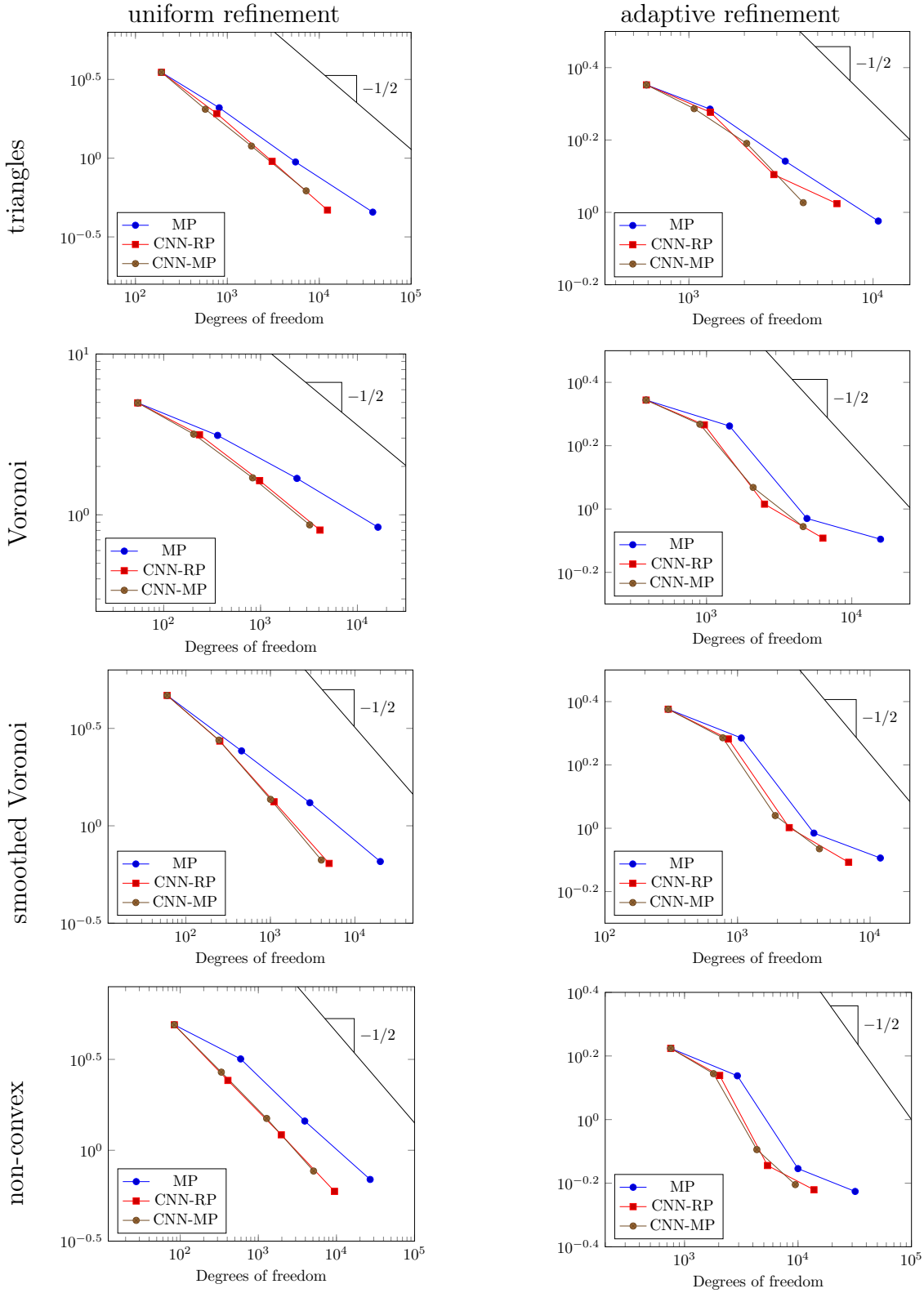

Figure 3.18: Stokes problem of Section 3.6.4. Computed errors in the H^1-broken norm of the velocity and L^2-norm of the pressure as a function of the number of degrees of freedom. Each row corresponds to the same initial grid (triangles, Voronoi, smoothed Voronoi, non-convex) refined uniformly with the proposed refinement strategies (MP, CNN-RP and CNN-MP). First column: uniform refinement case over the coarse grids in the first column of Figure 3.11. Second column: adaptive refinement case with fixed refinement fraction of 40% of mesh elements over the fine grids in the first column of Figure 3.15.

terms of improved accuracy using numerical methods such as PolyDG methods and VEMs. These results show that classifying correctly the shape of a polygonal element plays a key role in which refinement strategy to choose, allowing to extend and to boost existing strategies. Moreover, this classification step has a very limited computational cost when using a pre-trained CNN. The latter can be made off-line once and for all, independently of the differential model under consideration.

As noticed in [78], having skewed meshes can sometimes be beneficial to polytopal methods with hybrid unknowns, if the skewness is compatible with the diffusion anisotropy. We believe that our method may be extended so as to generate distorted elements by including the anisotropy directions into the shape recognition phase. Assume to dispose of a refinement strategy for "squares", which consists in diving the element into other four "squares", and for "rectangles", which consists in diving the element into two "squares". Assume you have to refine square $(0,1)^2$ and that the anisotropy directions are aligned with the axes. Before processing the shape of the element, you may skew the element along the anisotropy directions according to the magnitude of anisotropy, therefore turning the square into a rectangle. In this way the original square will be classified as "rectangle" and divided into two rectangles skewed in the desired direction. When applying the same strategy on the new rectangles, they will be deformed into "squares", and therefore refined into four elements skewed in the desired direction.

4 | Machine learning-based refinement strategies for polyhedral grids

In this chapter[1], we propose different strategies to handle polyhedral grid refinement, that exploit ML techniques to extract information about the "shape" of mesh elements in order to refine them accordingly. Starting from the approach developed in Chapter 3 in two dimensions, here we address the three-dimensional case. In particular, we employ the k-means clustering algorithm [95, 115] to partition the points of the polyhedron to be refined. Learning a clustered representation allows to better handle situations where elements have no particular structure, while preserving the quality of the grid. This strategy is a variation of the well known Centroidal Voronoi Tessellation [82, 117]. We also explore how to enhance existing refinement strategies, including the k-means, using CNNs, powerful function approximators successfully employed in many areas of ML, especially computer vision [2, 114]. We show that CNNs can be employed to identify correctly the "shape" of a polyhedron, without resorting to the explicit calculation of geometric properties which may be expensive in three dimensions. This information can then be exploited to design tailored strategies for different families of polyhedra. According to the classification, the most suitable refinement strategy is then selected among the available ones, therefore enhancing performance. We recall that the proposed approach has several advantages:

- it helps preserving structure and quality of the mesh, since it can be easily tailored for different types of elements;
- it can be combined with suitable user-defined refinement criteria, including the k-means strategy;
- it is independent of the differential model at hand and of the numerical method employed for the discretization;

[1]The results of the thesis are original and are contained in [6].

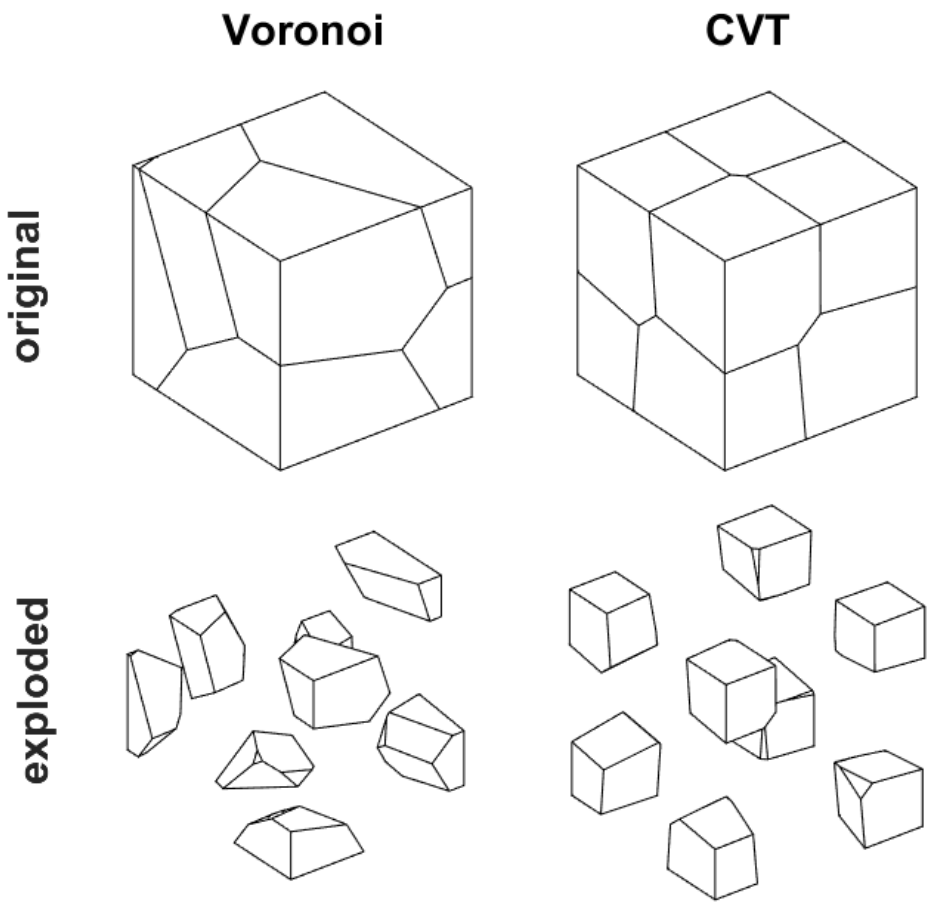

Figure 4.1: A Voronoi tessellation (top left) and a CVT (top right) of a cube and their exploded versions (bottom). “Small" edges and faces are generated.

- the overall computational cost is kept low, since CNNs can be trained offline once and for all.

To investigate the capabilities of the proposed approaches, we consider a second-order model problem discretized by the VEM and the PolyDG method. We measure effectiveness through an analysis of quality metrics and accuracy of the discretization error.

4.1. Refinement strategies

A polyhedral mesh can be generated, e.g., by standard triangulation algorithms or by hexahedral elements in the structured case. Another possibility is to use a Voronoi Tessellation [100]: suitable points, called seeds, are chosen inside the polyhedron and each element of the new partition is the set of points which are closer to a specific seed (see Figure 4.1 left column). A Centroidal Voronoi Tessellation (CVT) [82, 117] is a Voronoi tessellation whose seeds are the centroids (centers of mass) of the corresponding new elements (see

Figure 4.1 right column). This strategy produces elements which are shape-regular and of similar size (these properties will be more formally addressed in Section 4.3). However, it has a considerable computational cost and it may produce elements with many small edges and faces, which implies higher costs in terms of memory storage and algorithm complexity. Moreover, although small edges and faces do not necessarily deteriorate the accuracy of numerical methods, they are in general not beneficial [35, 44, 79, 121]. As we will see in the following, these algorithms for grids generation can be adapted to perform mesh refinement.

In order to refine a three dimensional polyhedron a possible strategy is to subdivide the polyhedron along a chosen preferential direction [37]. It has a low computational cost and allows to adjust the cutting plane at each iteration. For instance it is possible to choose such plane so that we avoid small edges and consequently yield simpler mesh elements. In particular, we propose the general refinement strategy described in Algorithm 4.1, where the "size" or "diameter" of a polyhedron $P \subset \mathbb{R}^3$ is defined, as usual, as

$$\text{diam}(P) = \sup\{||x - y||,\ x, y \in P\},$$

where $||\cdot||$ is the standard Euclidean distance. We also recall that given a polyhedral mesh, i.e., a set of non-overlapping polyhedra $\{P_i\}_{i=1}^{N_P}$, $N_P \geq 1$ that covers a domain Ω, the mesh size is defined as

$$h = \max_{i=1,\dots,N_P} \text{diam}(P_i).$$

Possible ways of choosing the cutting plane in step 1 will be discussed in Section 4.1.1.

Algorithm 4.1 General refinement strategy of a polyhedron P

Input: polyhedral element P to refine, plane tolerance `tol`, maximum number of elements `nmax`, target size $\bar{h}$

Output: partition of P into polyhedral sub-elements

1: Choose a plane along which to slice the element.
2: If there are vertices closer than the user-defined tolerance `tol` to the plane, adjust the plane so that it passes thought at least the three closest ones.
3: If the plane passes the "validity check", slice the element into two sub-elements, otherwise use the "emergency strategy". Until there are less than `nmax` elements, repeat the above steps for each element with a size above $\bar{h}$.

The "validity check" in step 3, to see if it is possible to slice the element along the chosen plane, may encompass several requirements:

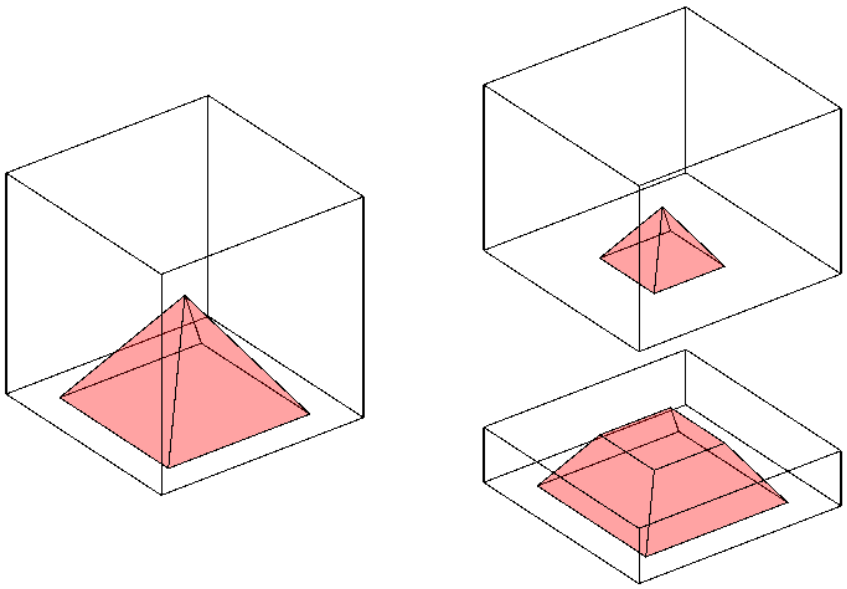

Figure 4.2: Left: a non-convex polyhedron is obtained from a cube (transparent) by removing a pyramid (red). Right: the polyhedron is sliced along a plane generating two polyhedra, one of which has a hole.

Geometrical. Avoid generating elements with holes and other degenerations, which may occur if the polyhedron is non-convex (see Figure 4.2).

Numerical. Avoid generating small edges and faces according to a prescribed threshold, which may still occur because in step 2 a plane can only be adjusted to pass from 3 points.

If the chosen plane does not pass the "validity check" a possible "emergency strategy" is the following: consider small perturbations of position and orientation of the original plane, until a valid configuration is found.

4.1.1. Choosing the cutting direction

Several strategies can be designed by making different choices of the cutting direction in step 1 of Algorithm 4.1. In Figure 4.3 the proposed strategies are applied on a tetrahedron. Notice that which strategy is the most effective may depend on several factors, such as the initial geometry of the mesh, the quality of the refined elements, the computational cost, the stability of the numerical scheme, or the regularity of the solution.

Diameter strategy. The diameter of a polyhedron is identified by the two farthest apart vertices of the element, say v_1 and v_2. The cutting plane has origin $x_0 = (v_1 + v_2)/2$ and is orthogonal to the direction of the diameter, i.e., it has normal $n = (v_2 - v_1)/||v_2 -$

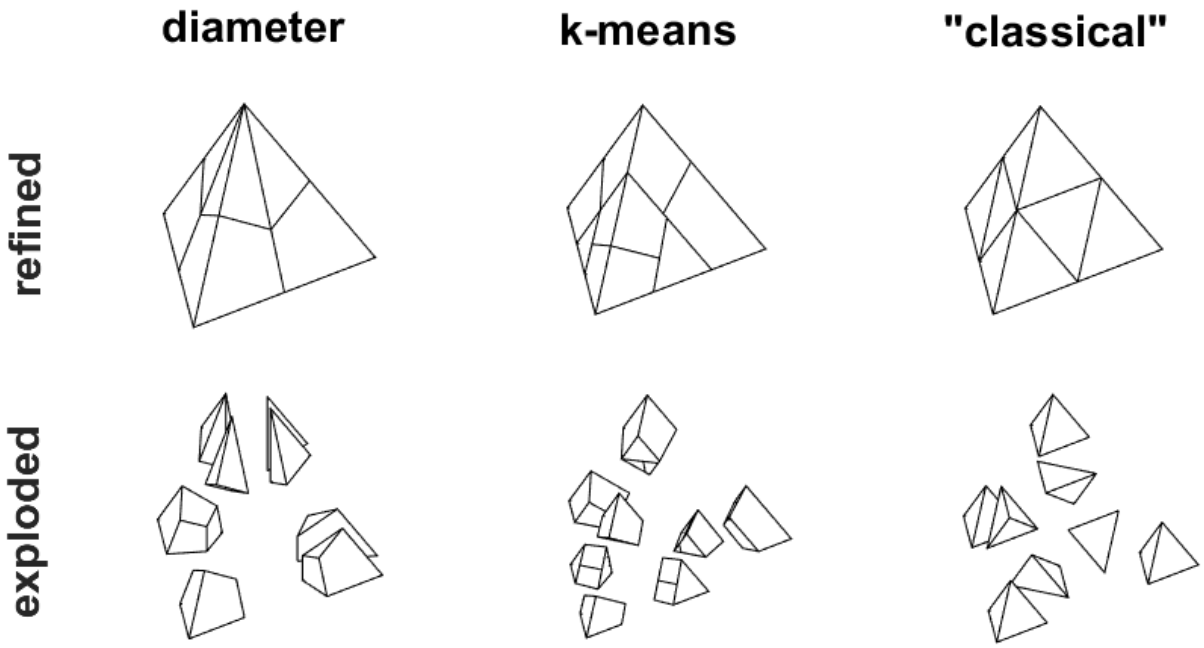

Figure 4.3: A tetrahedron is refined using diameter, k-means and “classical" strategies (top row) and their exploded versions (bottom row).

$v_1||$. This is a greedy algorithm to halve the size of the element. It has the advantage of easily computing the cutting plane, but it may produce skewed elements because the overall shape of the polyhedron is not taken into account.

K-means strategy. Points belonging to the polyhedron are grouped into two clusters using the k-means algorithm [95, 115] and are then separated by the cutting plane, as described in Algorithm 4.2. This is equivalent to a CVT with only two seeds. This strategy produces rounded elements of similar size, but it has a higher computational cost. In principle, different clustering algorithms may be used to learn different representations of the element, such as hierarchical clustering [104, 123] or self-organizing maps [105, 134].

Algorithm 4.2 K-means cutting plane

Input: polyhedron P to refine, number of grid points n

Output: cutting plane of P

1: Construct a grid of n of points inside the smallest domain of the form $[x_1, x_2] \times [y_1, y_2] \times [z_1, z_2]$ which contains P.
2: Remove the points of the grid outside P [112], which can be done efficiently when P is convex.
3: Randomly sample two points and label them centroids c_1 and c_2.
4: Assign each point to the cluster with the closest centroid in L^2 norm.
5: Compute the average of points in each cluster to obtain two new centroid c_1 and c_2.
6: Repeat steps 4 and 5 until cluster assignments do not change, or the maximum number of iterations is reached.
7: The cutting plane has normal $n = (c_2 - c_1)/||c_2 - c_1||$ and origin $x_0 = (c_1 + c_2)/2$.

"Classical" strategies. If the shape of the polyhedron is known (up to rotation, scaling, stretching or reflection) it is possible to design tailored refinement strategies using multiple cutting planes. This is the case of tetrahedra, cubes and prisms, which are employed in classical FEMs. These refinement strategies, shown in Figure 4.4, produce regular elements at low computational cost. Moreover, since the new elements have the same shape as the original one, the algorithm can be applied recursively. Strategies for other polyhedral element can be designed. The main drawback is that these strategies cannot be applied if the shape of the polyhedron is unknown.

4.2. Enhancing refinement strategies using Convolutional Neural Networks

Consider elements at the top row of Figure 4.5 from left to right:

1. A tetrahedron where a corner has been cut. This situation may arise when slicing a background mesh, e.g., in order to model a fracture or a soil discontinuity.
2. A prism where one of its faces is divided into two triangles, due to the presence of non-matching grids. This situation can arise each time a neighbouring mesh element is refined, because this may require to partition a face which is shared by two elements.
3. An element of a CVT. These tessellations are employed for meshing irregular domains with few regular elements.

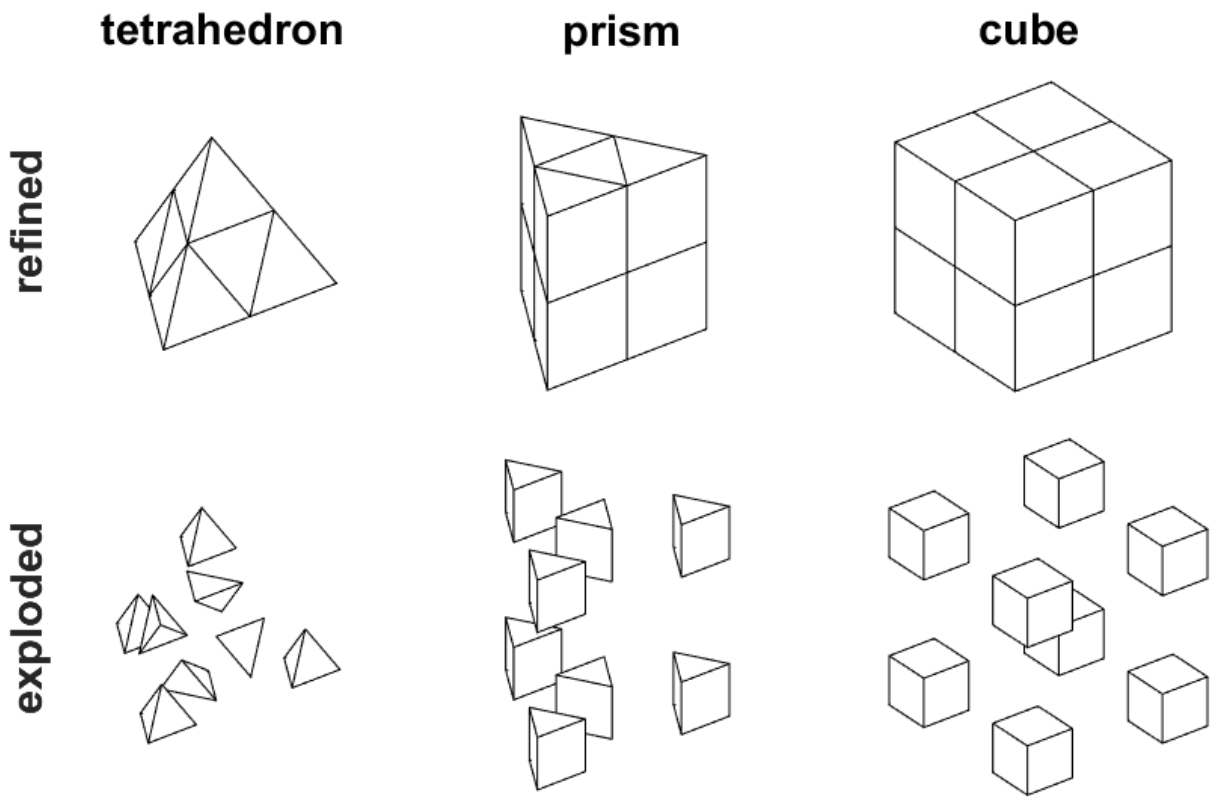

Figure 4.4: “Classical" refinement strategies for a tetrahdron, a prism and a cube (top row) and their exploded version (bottom row), employed in classical FEMs.

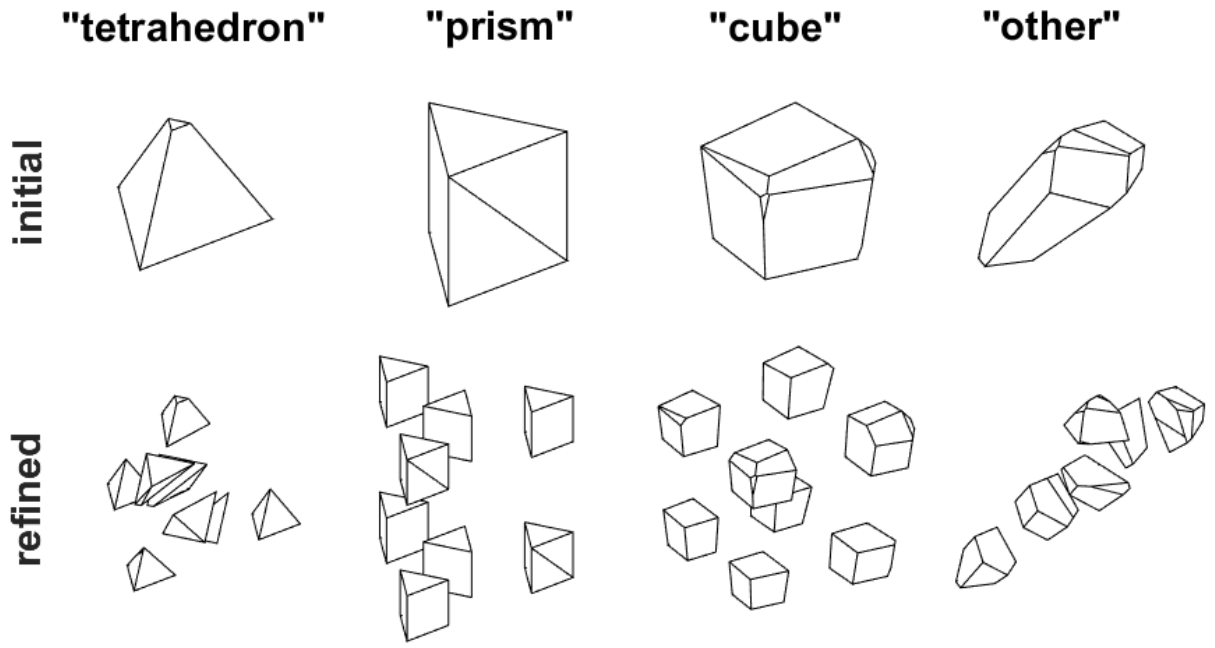

Figure 4.5: Top row: four polyhedra are classified as "tetrahedron", "prism", "cube" and "other", according to their overall shape and not to their geometrical description. Bottom row: the polyhedra are refined with the corresponding "classical" strategies according to their labels (columns 1-3) and the k-means strategy (column 4).

4. A skewed element with no particular structure. These elements can be found in standard Voronoi tessellations, or may appear even after refining a regular element.

Despite the fact that the geometrical representations of elements 1-3 in Figure 4.5 do not match any classical polyhedron, their overall shapes are similar to a tetrahedron, a prism and a cube, respectively. Therefore refining them with the corresponding shape strategies would produce regular elements at low computational cost. This can be done by computing cutting directions on a "reference shape" which can be constructed using Algorithm 4.3.

Algorithm 4.3 Computing the reference shape

Input: polyhedron P, shape type S (tetrahedron, cube, prism, ...)

Output: reference shape of P

1: Find n vertices of the polyhedron far apart from each other and from the centroid of the element, using the farthest first traversal algorithm [113], where n is the number of vertices of S.

2: Apply the convex hull algorithm [27] to the n vertices to obtain a triangulated surface.

3: Merge the m couples of neighbouring triangles of the surface whose angle is closest to $180°$, where m is the number of quadrilateral faces of S.

The last element does not have a particular shape, and therefore we apply the k-means strategy, which is more robust and flexible even though more expensive. The refined elements are shown at the bottom row of Figure 4.5. However, in order to apply the correct strategy for each element, we need an algorithm to access whether the shape of a polyhedron is more similar to a tetrahedron, a prism, a cube or none of these. We will use labels "tetrahedron", "prism", "cube" and "other", respectively, and we will refer to such labels as "equivalence classes". More in general, given a set of polyhedra to refine, $\mathcal{P} = \{P_1, P_2, ...\}$, and a set of possible refinement strategies, $\mathcal{R} = \{R_1, R_2, ..., R_n\}$, we want to find a classifier $F : \mathcal{P} \to \mathcal{R}$ that assigns to every polyhedron the most effective refinement strategy in terms of quality of the refined elements and computational cost. The effectiveness will be measured with suitable metrics later in Section 4.3.

When neighbouring elements need to be refined, e.g. when refining uniformly a mesh, a possible issue is how to split their interfaces in a compatible manner, in order to avoid small faces which are not in general beneficial for numerical methods [35, 44, 79, 121]. In order to solve this problem, ad-hoc criteria that consider the shape of multiple elements simultaneously could certainly be designed. However, taking into account potentially many elements would certainly raise considerably the online computational cost of the

approach, and would also require effort in terms of design complexity of the algorithm. Therefore, in such cases our approach consists in designing a suitable "validity check" that avoids small faces in step 3 of Algorithm 4.1. In such a way, more iterations may be required to find a valid plane but the online cost would remain contained, because we are still considering one element at a time.

4.2.1. Polyhedra classification using conovolutional neural networks

In principle, any classifier may be used for polyhedra. However, considering geometrical properties of the element alone, such as connectivity and area of the faces, may be too complex to operate a suitable classification: for example, the element labeled as “tetrahedron” in Figure 4.5 is actually a deformed prism. Approaches like Shape Analysis [80], which relies on applying transformations the polyhedron in order to match a reference shape, are too expensive in three-dimensions and seem to lack of the required flexibility: for example there is no reference shape for class “other”. Instead, providing samples of polyhedra together with the desired label can be easily done: for classes "tetrahedron", "prism" and "cube" we start from the corresponding regular polyhedra and then apply small deformations and rotations, while for class "other" we consider polyhedra from several Voronoi tessellations. This database can be used to “train” our function, i.e., to tune its parameters in order to obtain the desired classifier. This framework, which exploits labeled data, is known as “supervised learning” and CNNs are powerful function approximators in the context of image classification. Indeed, the most intuitive way to asses the shape of an object is by visual inspection and polyhedra can be easily converted into binary images: each pixel assumes value 1 inside the polyhedron and 0 outside [112] (see Figure 4.6), which can be done efficiently in the convex case. Mathematically, while the geometrical representation of a polyhedron is given by its boundary, the information about the overall shape is given by the distribution of its volume, i.e., the location of its pixels. Notice that a three-dimensional image contains the same information (up to a scaling and a translation) required by the k-means algorithm, which is a form of “unsupervised” learning because only unlabeled data are provided. The proposed CNN-enhanced refinement strategy is described in Algorithm 4.4.

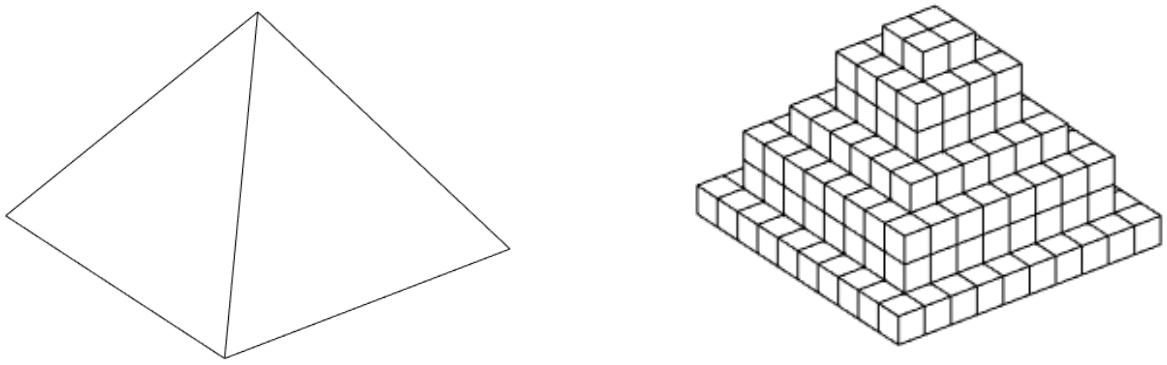

Figure 4.6: A pyramid (left) and its three-dimensional image (right). Pixels have value 1 inside the polyhedron (represented with cubes) and 0 outside (not represented).

Algorithm 4.4 CNN-enhanced refinement strategy

Input:polyhedron P to refine

Output: partition of P into polyhedral sub-elements

1: Construct a binary image of P. Classify the image using a CNN and obtain the label $\mathcal{L}$.
2: **if** $\mathcal{L}$ = "tetrahedron", "prism" or "cube" **then**
3: Apply Algorithm 4.3 to compute the corresponding reference shape.
4: Compute the cutting directions on the reference shape using the corresponding "classical" strategy.
5: Refine P by applying the cutting directions using Algorithm 4.1.
6: **end if**
7: **if** $\mathcal{L}$ = "other" **then**
8: Refine P using the k-means strategy, i.e., computing the cutting directions using Algorithm 4.2 and then using Algorithm 4.1.
9: **end if**

In general, the use of CNNs should be seen as tool to enhance existing refinement strategies, not as a refinement strategy on its own in competition with the others. In the case of "classical" strategies, the CNN is extending their domain of applicability to general polyhedra, which would not be possible otherwise.

4.2.2. Convolutional neural network training

We adopt the framework of supervised learning for image classification using CNNs described in 2.1. In particular, we consider three dimensional binary images of the form

$\mathbf{B} \in \{0,1\}^{n\times n\times n}$, $n \geq 1$,.
We have used a database of 22500 binary images of size $16 \times 16 \times 16$ pixels, with labels "tetrahedron", "prism", "cube" and "other". The data were divided into training, validation and test set with ratios 60%-20%-20% respectively. They were generated as follows:

- for classes "tetrahedron", "prism" and "cube" we generated 6000 images each, starting from the corresponding regular polyhedra, and then randomly applying perturbations, such as rotation, scaling, stretching and reflection;
- for the class "other" we generated 4500 images from elements of several Voronoi tessellations, where seeds have been randomly located.

We generated less images for class "other", because among all of the randomly generated Voronoi elements, some of them will be actually deformed tetrahedra, prisms and cubes. In these doubtful situations, the CNN will be more likely to classify polyhedra as having a particular shape rather assigning them label "other". Indeed, the CNN is capable to learn the whole data distribution, including the relative frequency of each class. In this way, we resort to the k-means only when really needed.
Generating data which are representative of the application of interest is the most critical part of the process: the CNN can easily misclassify new samples if they are too different from the training set, even if the accuracy on the test set is extremely high. In three dimensions, the variability of polyhedra is high and generating representative sample is much more complex than in the two-dimensional case [4]. It is important to have a class "other" to which unknown polyhedra will be assigned, in order to treat them with a robust backup strategy, such as the k-means. This should not discourage from using "classical" strategies, because when employed they are likely to produce regular elements.
Adding other shapes to the refinement process, such as "pyramid" or "icosahedron", is certainly possible. This would ultimately improve performance, as the refinement algorithm should be capable to handle in a tailored manner a wider range of configurations. This implies generating a larger database of images and training a network with more parameters. However, in the two dimensional version of this problem [4] it was observed that a large amount of shapes does not dramatically improve performance. In this work, we have selected only the shapes "tetrahedron", "cube" and "prism" because when refined they produce sub-elements with the same shape of the original one. This facilitates successive refinements and helps preserving the mesh structure.

The CNN architecture we used is $\mathrm{CNN} : \{0,1\}^{16\times16\times16} \to (0,1)^4$, where

$$\begin{aligned}\mathrm{CNN} = {} & \textsc{Conv}(\bar{f}=8, m=8) \to \textsc{ReLU} \to \textsc{Pool}(\bar{f}=2, s=2) \to \\ & \textsc{Conv}(\bar{f}=4, m=8) \to \textsc{ReLU} \to \textsc{Pool}(\bar{f}=2, s=2) \to \\ & \textsc{Conv}(\bar{f}=2, m=8) \to \textsc{ReLU} \to \textsc{Pool}(\bar{f}=2, s=2) \to \\ & \textsc{Linear} \to \textsc{Softmax},\end{aligned}$$

where $\textsc{Conv}(\bar{f}, s)$ is convolutional layer with filter size $\bar{f}$ (i.e. a window $\bar{f} \times \bar{f} \times \bar{f}$) and m feature maps, $\textsc{Pool}(\bar{f}, s)$ is an avergage pooling layer with filter size $\bar{f}$ and stride s, ReLU and Linear were defined the previous Section 2.1.1, and for any two compatible functions f_1 and f_2 the arrow notation is defined as $f_1 \to f_2 = f_2 \circ f_1$. We employed average pooling layers, rather than max pooling layers, because they have the effect to blurry and coarse-grain image features. This has been done in order to focus the CNN on the overall shape of the polyhedron, and not on small scale details and corners. Three sequences of $\textsc{Conv} - \textsc{ReLU} - \textsc{Pool}$ layers seem to be sufficient to enforce invariance with respect to rotations of the input polyhedron. The size of the images is $16 \times 16 \times 16$ pixels, i.e., a resolution large enough to apply 3 pooling layers, whose effect is to halve the size of the image across all the dimensions. Such a small resolution allows to quickly generate and classify the image of a polyhedron, which is important in an online setting. Moreover, a smaller CNN is more likely to be robust with respect to samples very different from the training data. It was empirically observed that a larger resolution was not needed for the considered dataset. This is motivated by the fact that the approximation properties of neural networks generally depend on the dimension of the data manifold, not on the dimension of the input space [126]. A larger resolution may be needed to classify polyhedra characterized by smaller scale features, e.g., to correctly distinguish between a dodecahedron and an icosahedron. As shown in Figure 4.7, the class “other" is reasonably the most misclassified, because it includes polyhedra with very different shapes, some of which are actually deformed tetrahedra, prisms and cubes. We trained the CNN using the Adam optimizer [109], with mini-batch size 128, data shuffling every epoch, L^2 regularization coefficient 0.1 and learn rate 10^{-3}. 50 epochs of training take approximately 15 minutes with MATLAB2020b on a Windows OS 10 Pro 64-bit, CPU NVidia GeForce GTX 1050 Ti and 8244Mb Video RAM. Further training does not seem to improve performance.

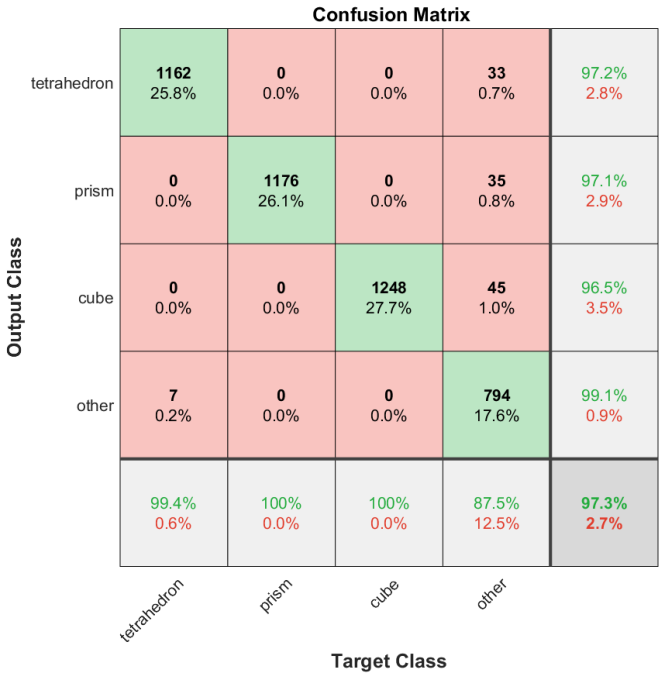

Figure 4.7: Confusion matrix for polyhedra classification using a CNN. Class “other" is reasonably the most misclassified, because it includes polyhedra with very different shapes, some of which are actually deformed tetrahedra, prisms and cubes.

4.3. Validation on a set of polyhedral grids

In this section we compare the performance of the proposed algorithms. To evaluate the quality of the refined grids, we employ the following quality metrics introduced in [23]:

- *Uniformity Factor* (UF): ratio between the diameter of an element P and the mesh size
$$\mathrm{UF}(P) = \frac{\mathrm{diam}(P)}{h}.$$
This metric takes values in $[0, 1]$. The higher its value is the more mesh elements have comparable sizes.

- Ball *Ratio* (BR): ratio between the radius of the inscribed ball and the radius of the circumscribed ball of a polyhedron P
$$\mathrm{BR}(P) = \frac{\max_{\{B(r)\subset P\}} r}{\min_{\{P\subset B(r)\}} r},$$
where $B(r)$ is a ball of radius r. For the practical purpose of measuring the roundness of an element the radius of the circumscribed ball has been approximated with $\mathrm{diam}(P)/2$. This metric takes values in $[0, 1]$. The higher its value is the more

rounded mesh elements are.

Moreover, to measure the computational complexity of the different refinement strategies, for the refined grids we will consider the number of vertices, of edges, of faces, of elements, the total refinement time and the average refinement time per element.

In the following, we apply Algorithm 4.1 to a polyhedron P using a tolerance $\texttt{tol} = \text{diam}(P) \cdot 10^{-3}$, a maximum number of elements $\texttt{nmax} = 8$ (needed to apply the "classical" refinement strategy for cubes), and a target size $\bar{h} = \text{diam}(P)/2$. Concerning the k-means strategy, we apply Algorithm 4.2 using a grid of $n = 20^3$ points, which is comparable to the size of 16^3 pixels required by the CNN. This value of n may seem high, but it is needed in order compute the cutting plane with a satisfactory accuracy. Indeed, many grid points lie outside the polyhedron and therefore they are not included in the computation of the clusters, which is also the most expensive part of the process. We consider five different coarse grids of domain $(0, 1)^3$: a grid of tetrahedra, a grid of cubes, a grid of prisms, a Voronoi grid with random seeds location, and a CVT. In Figure 4.8 these grids have been refined uniformly, i.e., each mesh element has been refined, for three times using the diameter strategy (diameter), the k-means strategy (k-means) and the CNN-enhanced strategy (CNN). As we can see, the initial mesh geometry seems to be disrupted by the diameter strategy, approximately preserved by the k-means strategy, and well preserved by the CNN-enhanced strategy.

In Figure 4.9 we show the computed quality metrics for the uniformly refined grids. As we can see, in general quality metrics are lower for the diameter strategy. The k-means strategy produces elements with higher BR, while the CNN-enhanced strategy with higher of UF, although in many cases both of them are comparable.

In Figure 4.10 we show, for all the considered grids, the number of vertices, edges, faces, elements, the total refinement time and the average refinement time per element. These results have been obtained by using MATLAB2020b on a Windows OS 10 Pro 64-bit, Intel(R) Core(TM) i7-8750H CPU (2.20GHz / 2.21GHz) and 16 GB RAM memory. As we can see, in general the diameter strategy has the highest computational complexity, which is comparable to the one of the k-means strategy. The diameter strategy is supposed to be the fastest in computing the cutting direction, but, because it produces low quality elements with lost of vertices, the refinement time becomes comparable with that of the k-means strategy. The CNN-enhanced strategy outperforms both of them in terms of number of geometrical entities and computational time. It is worth noticing that in the CVT the CNN-enhanced strategy achieves higher or comparable quality with respect to the k-means strategy (Figure 4.9 bottom) in much less time. This is because the CNN is classifying CVT elements as "cubes", therefore reducing the computational cost. This

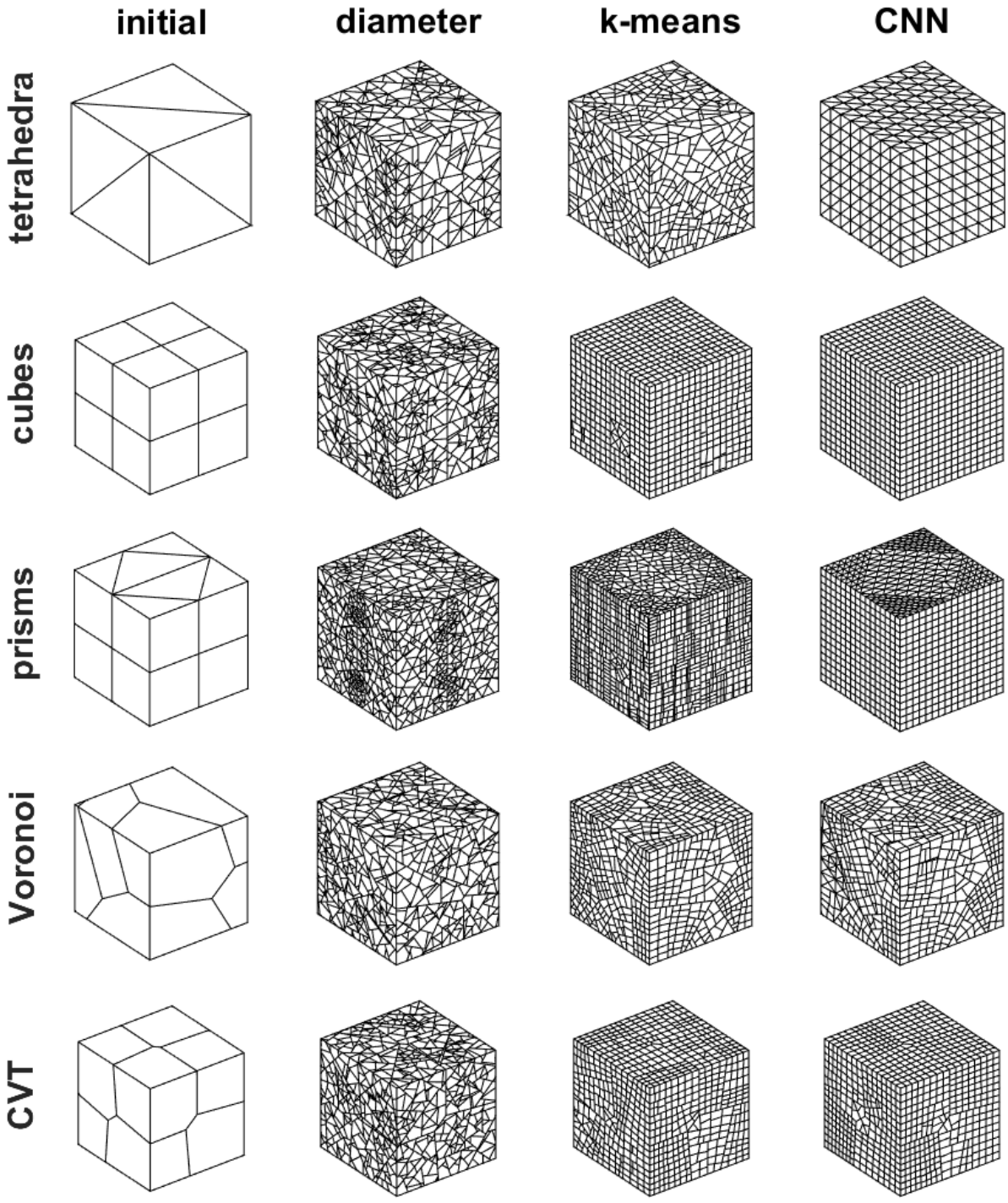

Figure 4.8: In the first column, coarse grids of domain $(0,1)^3$: a grid of tetrahedra, a grid of cubes, a grid of prisms, a Voronoi grid with random seeds location, and a CVT. Second to fourth columns: refined grids obtained after three steps of uniform refinement based on employing the k-means strategy (k-means) and the CNN-enhanced strategy (CNN). Each row corresponds to the same initial grid, while each column corresponds to the same refinement strategy.

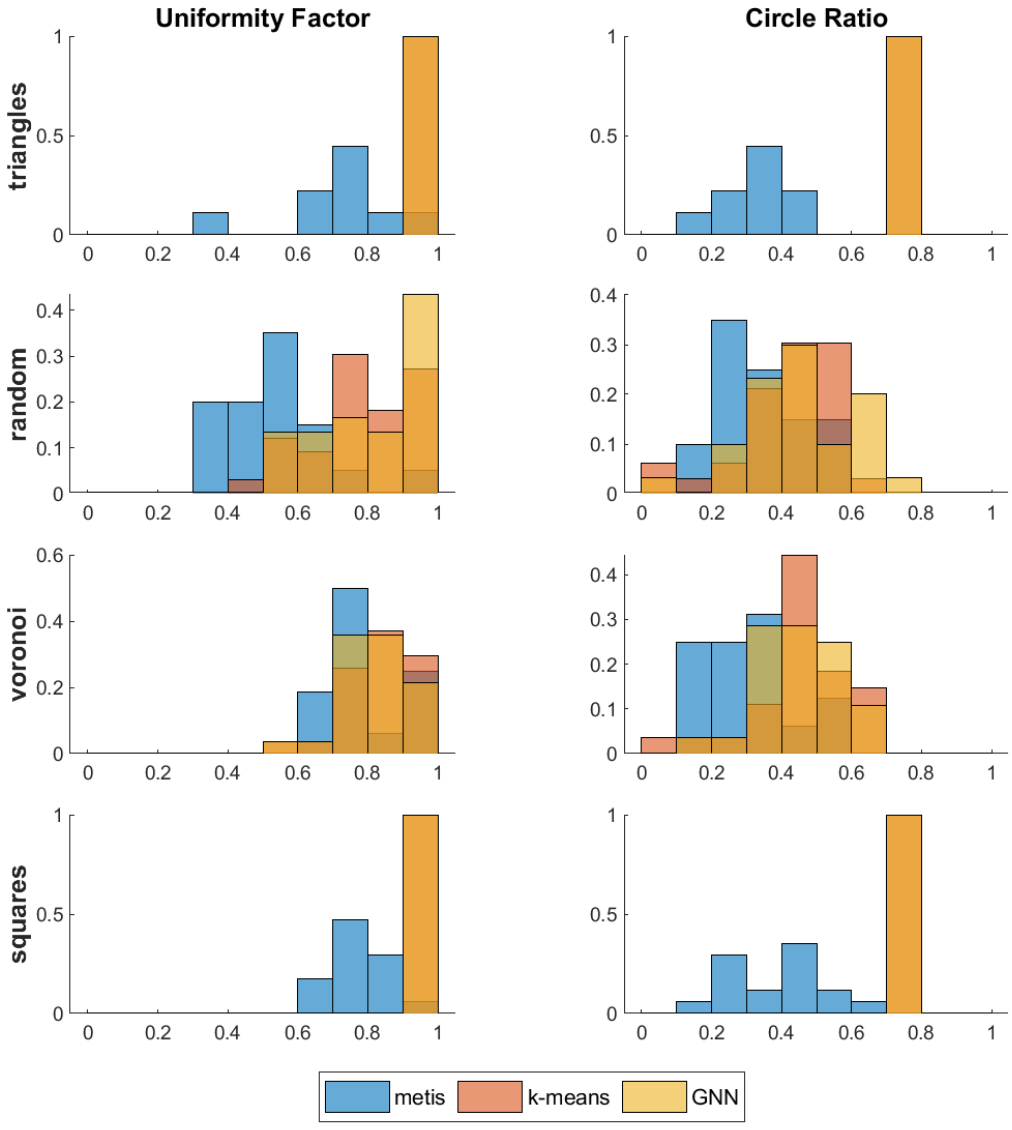

Figure 4.9: Histograms of the computed quality metrics (UF and BR on the x-axis and percentage of grid elements on the y-axis) for the refined grids reported in Figure 4.8 (second to fourth column), obtained based on employing different refinement strategies (diameter, k-means, CNN).

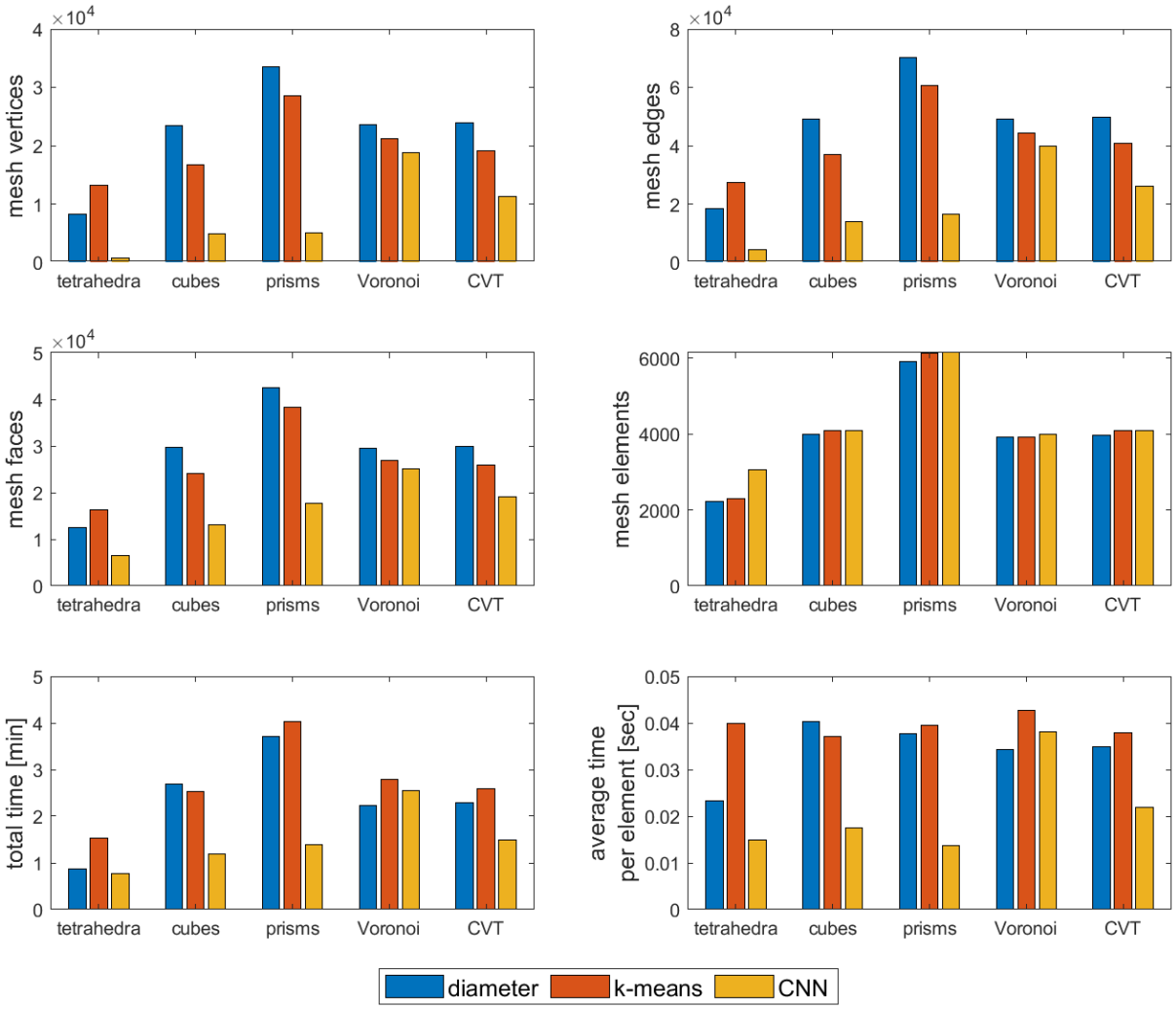

Figure 4.10: Statistics of computational complexity (number of vertices, edges, faces, elements, total refinement time and average refinement time per element) for the refined grids reported in Figure 4.8 (second to fourth column), obtained based on employing different refinement strategies (diameter, k-means, CNN). Lower values mean lower complexity in terms of memory storage and computational cost of the algorithms.

means that the CNN is generalizing properly on new samples, because CVT elements were not included in the training set.

Summary. The diameter strategy has the simplest implementation but it produces low quality elements, raising the computational cost. The k-means strategy produces the most rounded elements and performs better in unstructured grids. It is robust but has a considerable computational cost. The CNN-enhanced strategy well preservers the mesh structure, producing simple elements of uniform size at a low computational cost. However, it is less effective on unstructured grids.

4.3.1. Application to agglomerated meshes

In this section we consider the mesh of a hinge, shown in Figure 4.11, obtained by agglomerating a grid of tetrahedra using PARMETIS [107]. Such a mesh contains holes and small features. Moreover, elements maybe stretched and non-convex, because they have been obtained through a process of agglomeration. In Figure 4.12 we show an example of a non-convex element from the mesh (left) obtained during the agglomeration process and its corresponding refinement based on employing the diameter, the k-means and the CNN strategies (of Figure 4.12, from left to right). As we can see the diameter strategy, which does not take into account the non-convexity of the element, produces elements with pronounced non-convex spikes, contrary to the k-means and the CNN strategies. In this case, we decided not to re-train the CNN including non-convex elements in the database, in order to test its generalization capabilities in a more “extreme" setting.
In Figure 4.13 we show the computed quality metrics after one step of uniform refinement of the considered mesh. The k-means strategy attains the highest BR and comparable performance with the diameter strategy. The CNN and diameter strategies provide comparable results. We highlight the following comments:

- The considered mesh contains agglomerated elements which are not deliberately present in the training database of the CNN. On the contrary, the k-means strategy is expected to perform well even in unknown scenarios, because it does not require prior knowledge on the shape of mesh elements. Obviously, it always possible to tune the behaviour of the CNN strategy by including agglomerated elements in the training database.
- The CNN strategy classifies 52% of the elements as "tetrahedron", 47% as "other" and 1% as "prism". Such a distribution of labels is probably due to the fact that the considered mesh was obtained starting from a mesh tetrahedra, from which many

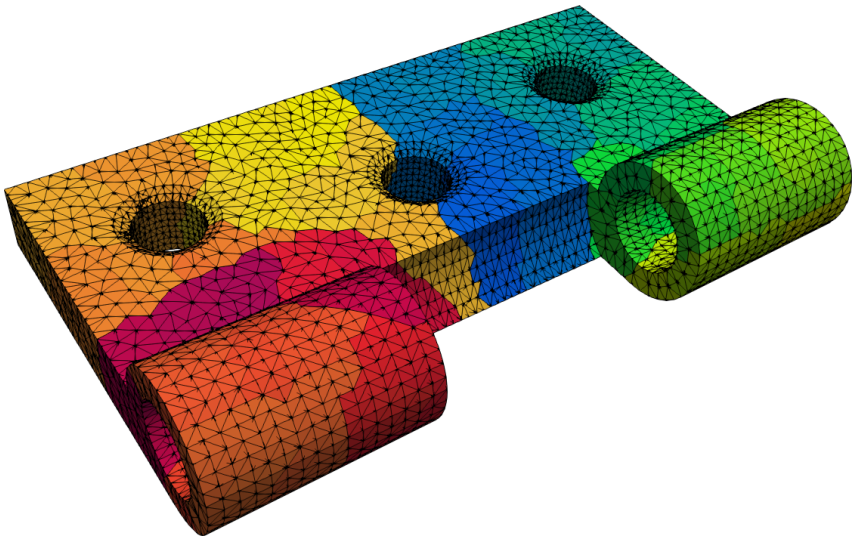

Figure 4.11: Mesh of the hinge geometry taken into account. Each color corresponds to one of the 9361 agglomerated sets of tetrahedra.

features may have been preserved during the process of agglomeration. Therefore, on the one hand the CNN may be trying to enforce a tetrahedral structure, while on the other hand trying to enforce more roundness using the k-means strategy for those elements with no particular structure. The presence of multiple structures may therefore have a stronger impact the UF metric and a lighter impact on the BR metric. Indeed, the chosen quality metrics may not take into account some important aspects of the mesh, such as the presence of multiple structures or of small spikes such as the ones shown in Figure 4.12.

In Figure 4.14 we show the computed statistics of computational complexity. As we can see, the ML strategies perform slightly better than the diameter strategy in terms of mesh complexity. Moreover, the computational cost of the CNN strategy is lower than the cost of the diameter strategy, while the one of the k-means strategy is comparable or higher. The advantage in using ML strategies is not as evident as in the Figures 4.9 and 4.10 because in that case three refinement steps were performed, while in this case only one. Indeed, refining a mesh element while preserving its quality greatly eases in case of sequential refinements, increasing exponentially the benefits of using ML strategies.
This example shows that, in new scenarios, generalization capabilities of CNNs should be assessed.

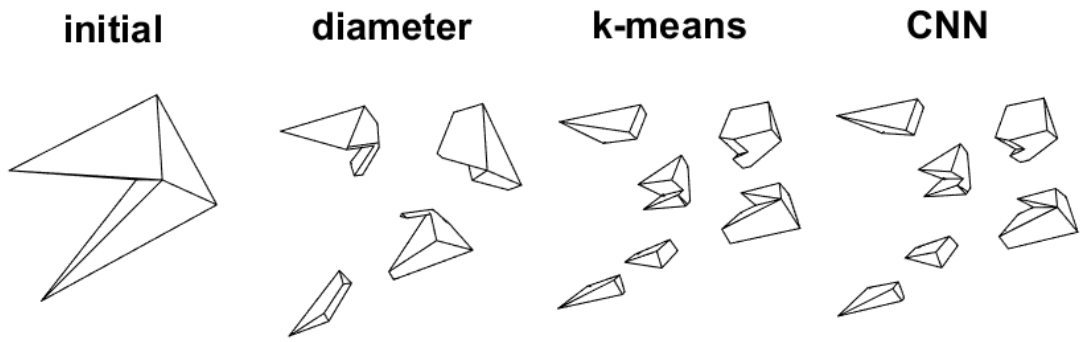

Figure 4.12: A non-convex element from the mesh of a hinge is refined (left) is refined using the diameter, the k-means and the CNN strategy (second to fourth figures). The k-means and the CNN strategy produce the same result because the CNN has classified the element as "other", hence also applying the k-means strategy. The diameter strategy, which does not take into account the non-convexity of the element, produces elements with pronounced non-convex spikes, contrary to the other two strategies.

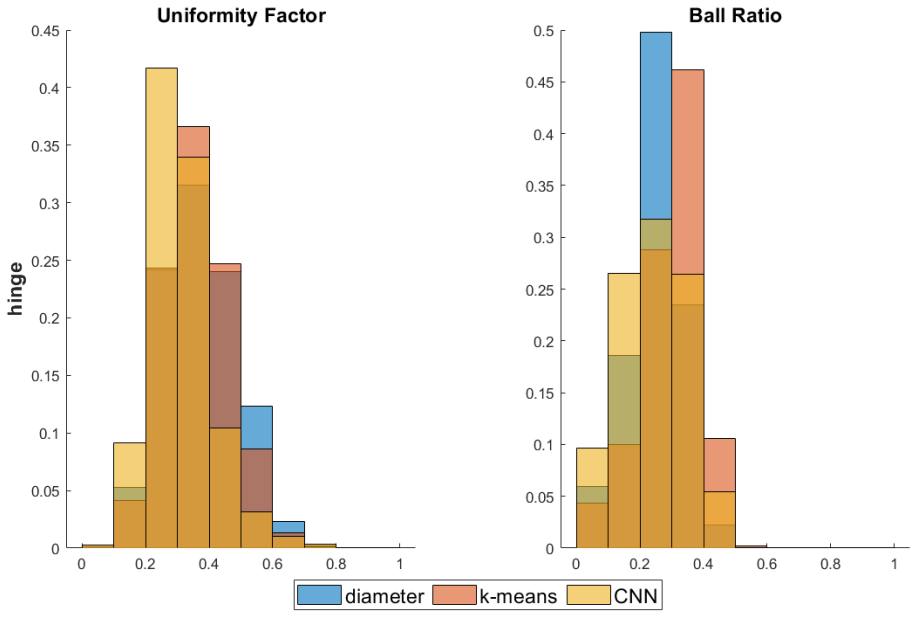

Figure 4.13: Histograms of the computed quality metrics (UF and BR on the x-axis and percentage of grid elements on the y-axis) for the refined grid of the hinge, obtained based on employing different refinement strategies (diameter, k-means, CNN).

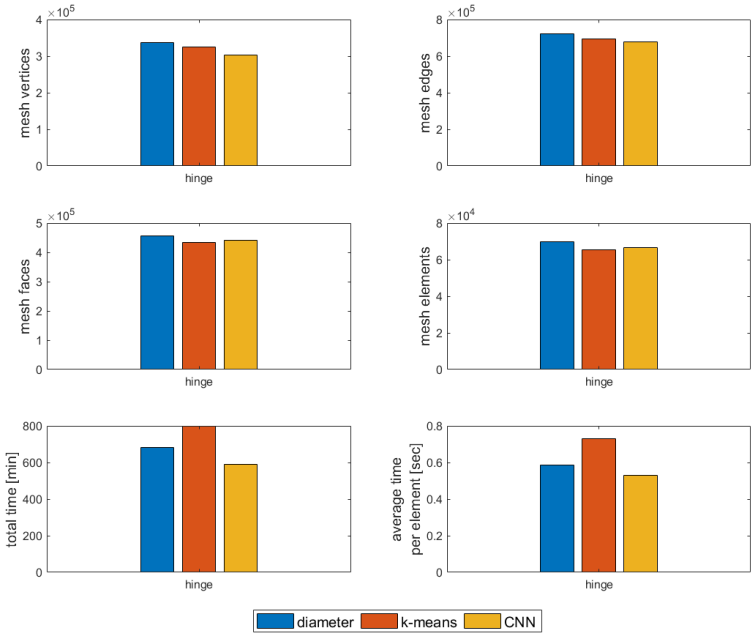

Figure 4.14: Statistics of computational complexity (number of vertices, edges, faces, elements, total refinement time and average refinement time per element) for the refined grid of the hinge, obtained based on employing different refinement strategies (diameter, k-means, CNN). Lower values mean lower complexity in terms of memory storage and computational cost of the algorithms.

4.4. Applications to VEM and PolyDG method

In this section we test the effectiveness of the proposed refinement strategies within polyhedral finite element discretizations. We consider the Virtual Element Method (VEM) [30, 31, 33, 34, 66, 68] and the Polygonal Discontinuous Galerkin (PolyDG) method [10, 13, 28, 50, 53, 99] to solve a standard Laplacian problem in 3D: find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v \quad \forall v \in H_0^1(\Omega), \tag{4.1}$$

with $f \in L^2(\Omega)$ a given forcing term. We take $\Omega = (0,1)^3$ and consider a uniform refinement process in Section 4.4.1, and an adaptive a priori mesh adaptation procedure in Section 4.4.2. In both cases, given a grid of the domain Ω we compute numerically the solution of problem (4.1) using either the VEM or the PolyDG method. Then we compute the error: in the VEM case in the discrete H^1 seminorm

$$|v|_{H^1}^2 := \sum_P |v - \Pi_k^\nabla v|_{H^1}^2 , \tag{4.2}$$

where Π_k^∇ is the standard VEM projection operator defined via the H^1 [31], while in the PolyDG case we take the DG norm

$$\|v\|_{\mathrm{DG}}^2 = \sum_P \|\nabla v\|_{L^2(P)}^2 + \sum_F \|\gamma^{1/2}[\![v]\!]\|_{L^2(F)}^2, \tag{4.3}$$

where P is a polyhedral mesh element and F is a polygonal element face [22, 63]. Here $\gamma = \alpha p^2/C_{el}$ is the stabilization function, where α is the penalty parameter, p is the polynomial degree used for the approximation, and C_{el} is a function of the shape of the element and is chosen as in [50]. The jump operator $[\![\cdot]\!]$, which measures the discontinuity of v across elements, is defined as in [22].

4.4.1. Uniform refinement

In this Section we consider a uniform refinement process, i.e., at each refinement step each mesh element is refined. The forcing term f in (4.1) is selected in such a way that the exact solution is given by

$$u(x, y) = \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

The grids obtained after three steps of uniform refinement are those already reported in Figure 4.8 (second to fourth column).
We now consider the PolyDG method applied to the grid of cubes for different values of the penalty parameter α, using polynomials of degree 1. Larger values of α penalize more discontinuities of the solution, rather than its gradient. In Figure 4.15 we show the computed relative errors as a function of the number of degrees of freedom.

As we can see, the CNN-enhance strategy is more effective for $\alpha = 2$, while the diameter strategy for $\alpha = 10$. In order to guarantee convergence, α needs to be larger than a threshold that depends on the shape of the elements. For example, $\alpha = 2$ works for the CNN-enhanced strategy, because it preserves the cube structure of the grid, but not for the other strategies. Since computing the threshold is in general too complex, in the following we will use $\alpha = 5$.
In Figure 4.16 we show the computed relative errors as a function of the number of degrees of freedom for both VEM and PolyDG method with the lowest-order approximation degree. More specifically, a VEM scheme with a polynomial consistency equal to 1 and a PolyDG approach that considers polynomials of degree 1 on edges, faces and volumes. Although this relative error could seem too high, the convergence rate are already the expected one and we limit ourselves to these three uniform refinement steps.

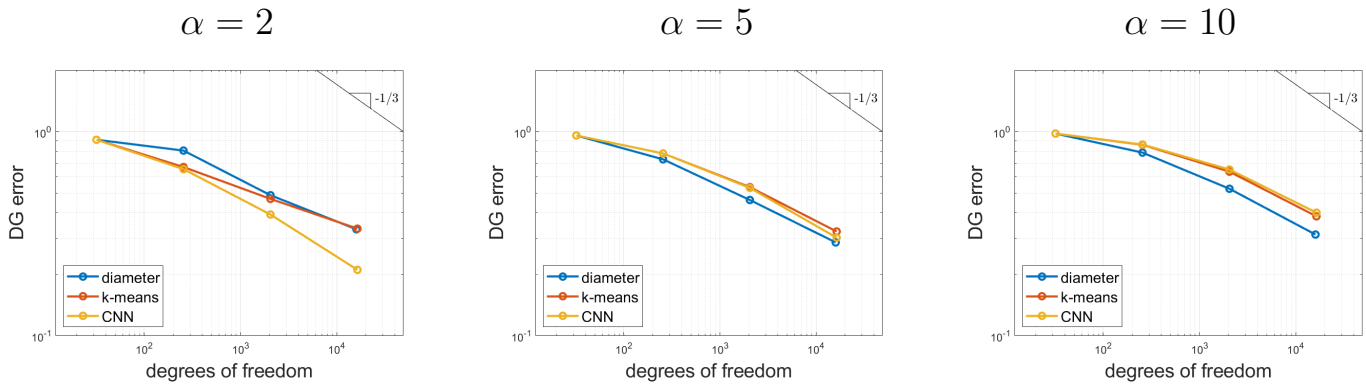

Figure 4.15: Uniform refinement test case of Section 4.4.1. Computed relative errors as a function of the number of degrees of freedom. The PolyDG method is applied to the grid of cubes for different values of the penalty parameter $\alpha = 2, 5, 10$, using polynomials of degree 1. The choice of the most effective refinement strategy (diameter, k-means, CNN) depends on the value of α.

When using the PoyDG method, the CNN-enhanced strategy is the most effective in the tetrahedral grid, while the diameter strategy in all the other cases. To maximize the performance, one may design a CNN-enhanced strategy that classifies polyhedra into two classes only, namely “tetrahedron” and “other”, in order to apply the “classical" strategy in the former case and the diameter strategy in the latter. However, considering that the performance of the different strategies are overall comparable and also that these results may vary depending on the choice of α, further analysis would be required in order to effectively exploit the use of ML techniques when employing the PolyDG method. On the contrary, in the VEM case, the CNN-enhanced strategy significantly outperforms the others, followed by the k-means strategy. At each step the error obtained with the CNN-enhanced refinement is associated with a lower number of degrees of freedom. Indeed, the error lines are shifted to the left. This sensitivity to mesh distortions may be due to the fact that the VEM is a hybrid method, with unknowns on the elements boundary. On the contrary, the robustness of the PolyDG method to mesh distortions may be due to the fact that the degrees of freedom are interior to the elements. In Figure 4.17 we show that the same results hold in the VEM case also for approximation orders 2 and 3.

4.4.2. Adaptive refinement

In this section we consider an adaptive refinement process. In particular, we proceed as follows:

1. Given a grid of the domain Ω we compute numerically the solution of problem (4.1) using either the VEM method or the PolyDG.

2. We compute the error with respect to the exact solution considering the H^1 semi-

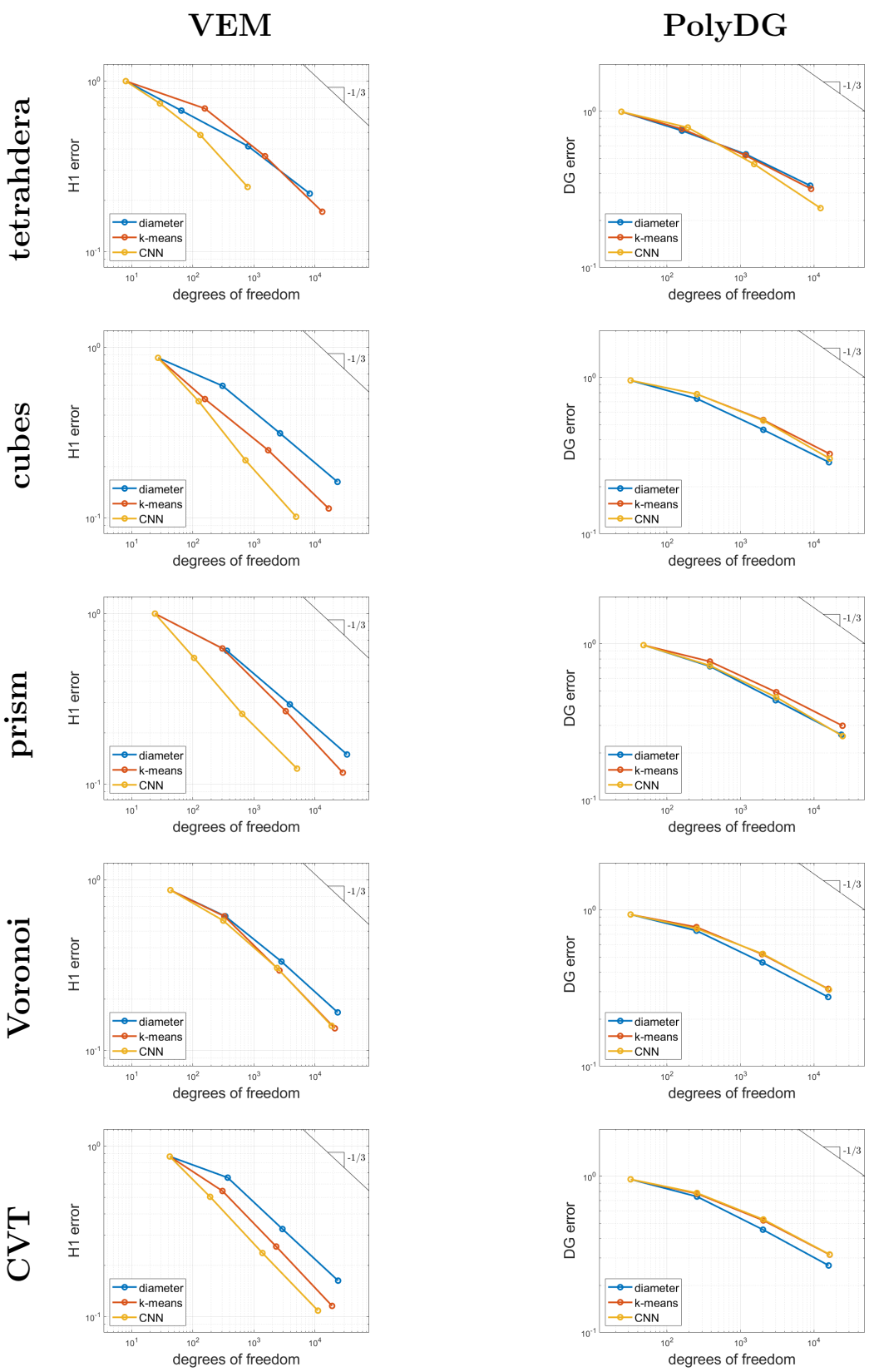

Figure 4.16: Uniform refinement test case of Section 4.4.1. Computed relative errors as a function of the number of degrees of freedom. Each row corresponds to the same initial grid refined uniformly with the proposed refinement strategies (diameter, k-means, CNN), while each column corresponds to a different numerical method (VEM left and PolyDG right).

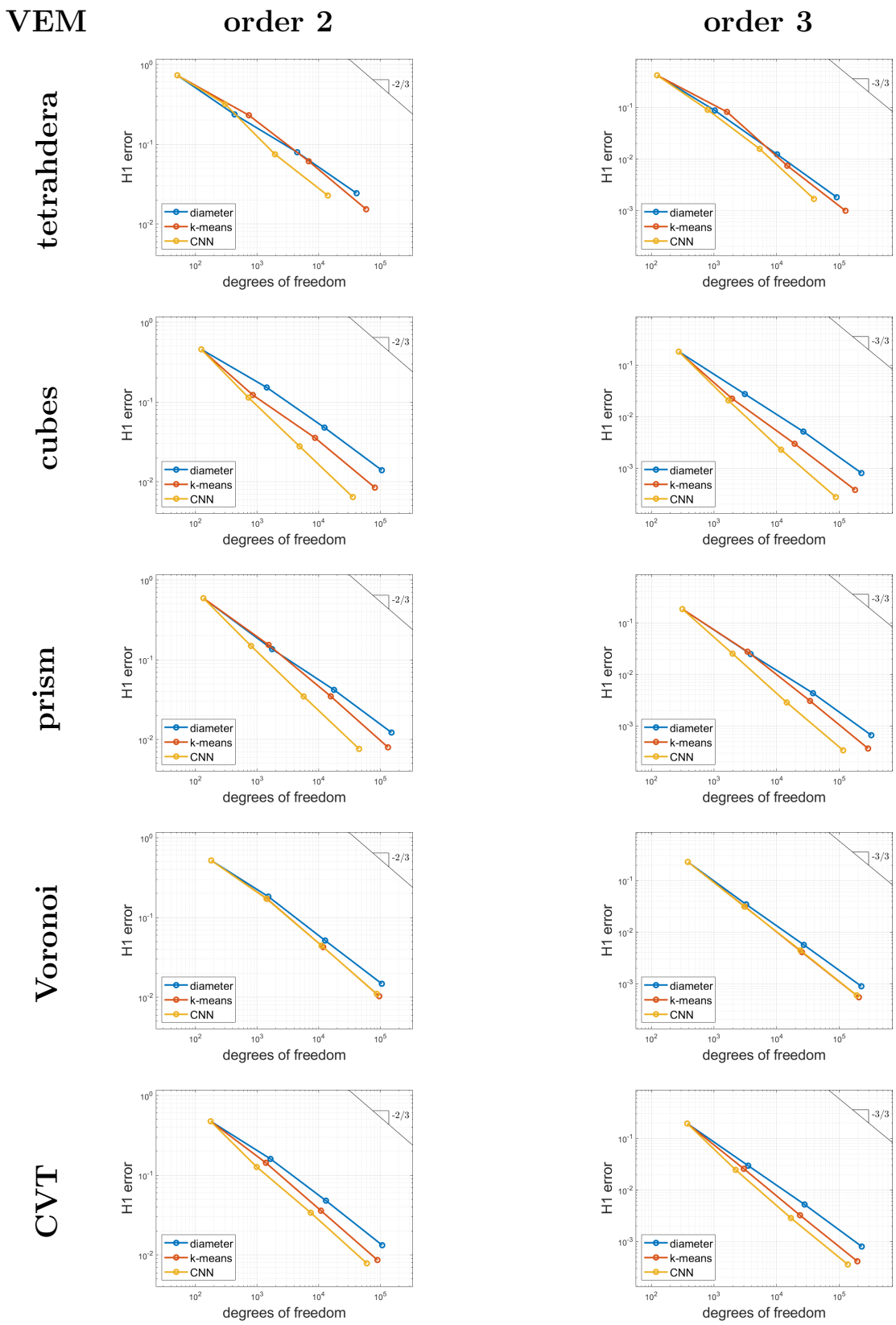

Figure 4.17: Uniform refinement test case of Section 4.4.1. Computed relative errors as a function of the number of degrees of freedom. Each row corresponds to the same initial grid refined uniformly with the proposed refinement strategies (diameter, k-means, CNN), while column corresponds to employing the VEM of order 2 and 3.

norm error, Equation (4.2), and the DG norm, Equation (4.3) for VEM and PolyDG case, respectively. Notice that we did not employ any a posteriori estimator of the error, since we would like to investigate the effect of the proposed refinement strategies.

3. We refine a fixed fraction r of elements with the highest error. To refine the marked elements we employ one of the proposed strategies (diameter, k-means, CNN).

The forcing term f in (4.1) is selected in such a way that the exact solution is given by

$$u(x, y) = (1 - e^{-10x})(x - 1)\sin(\pi y)\sin(\pi z),$$

that exhibits a boundary layer along $x = 0$. We take as initial grids the first ones of the previous example and we repeat Steps 1-3 for four times using refinement ratio $r = 0.4$. The adapted meshes employing the PolyDG method are shown in Figure 4.18, similar grids are obtained with VEM. In Figures 4.19 and 4.20 we show the computed relative errors for the PolyDG method of degree 1 and the VEM of orders 1, 2 and 3. Results are analogous to the uniform refinement case: in the VEM case there is a clear advantage in using the CNN-enhanced refinement strategy. Also in this case at each step the error obtained with the CNN-enhanced refinement is associated with a lower number of degrees of freedom: the error lines are shifted to the left. In the PolyDG case all strategies have comparable performance.

4.5. Concluding remarks

We proposed the extension to three dimensions of a paradigm based on ML techniques to enhance existing polygonal grid refinement strategies [4], within polyhedral finite element discretizations of partial differential equations. In particular, the k-means algorithm is used to learn a clustered representation of the element that is used to perform the partition. This strategy is a variation of the well known Centroidal Voronoi Tessellation. It produces shape-regular elements and it is robust when applied on unstructured grids.
Another approach consists in using a CNN as a classifier for polyhedra, that associates to each element the most suitable refinement criterion, including the k-means. This approach has the advantage of being modular: any refinement strategy can be employed, as well as any numerical method can be used to solve the differential problem. Moreover, the CNN has low computational cost once trained, which makes it appealing especially in three dimensions where processing mesh elements is much more expensive than in two dimensions. The use of CNNs allows to exploit strategies explicitly based on the "shape"

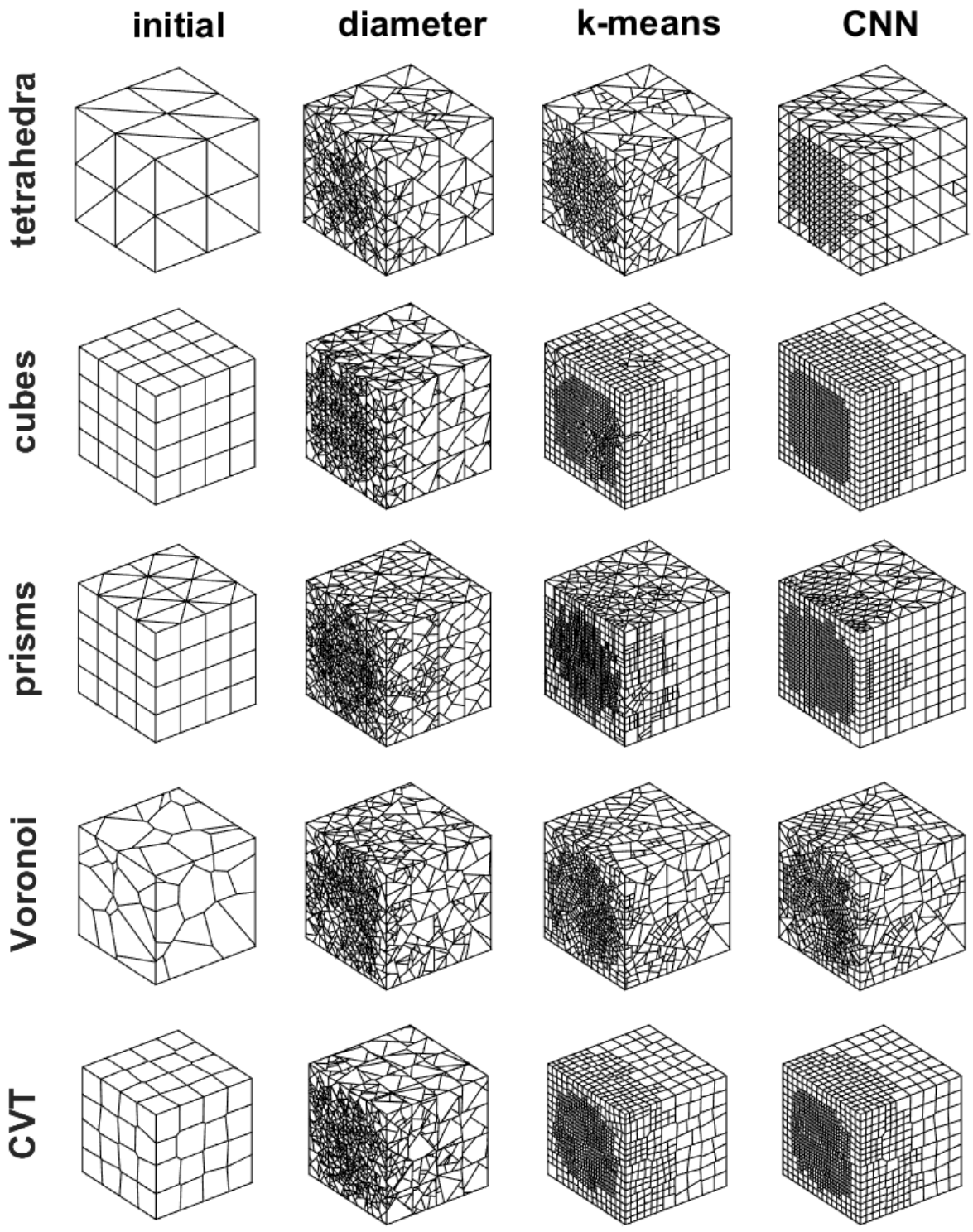

Figure 4.18: Adaptive refinement test case of Section 4.4.2. Each row corresponds to the same initial grid (tetrahedra, cubes, prisms, Voronoi, CVT), while each column corresponds to the same refinement strategy (diameter, k-means, CNN). Three steps of adaptive refinement have been performed, with a fixed fraction refinement criterion of $r = 0.4$.

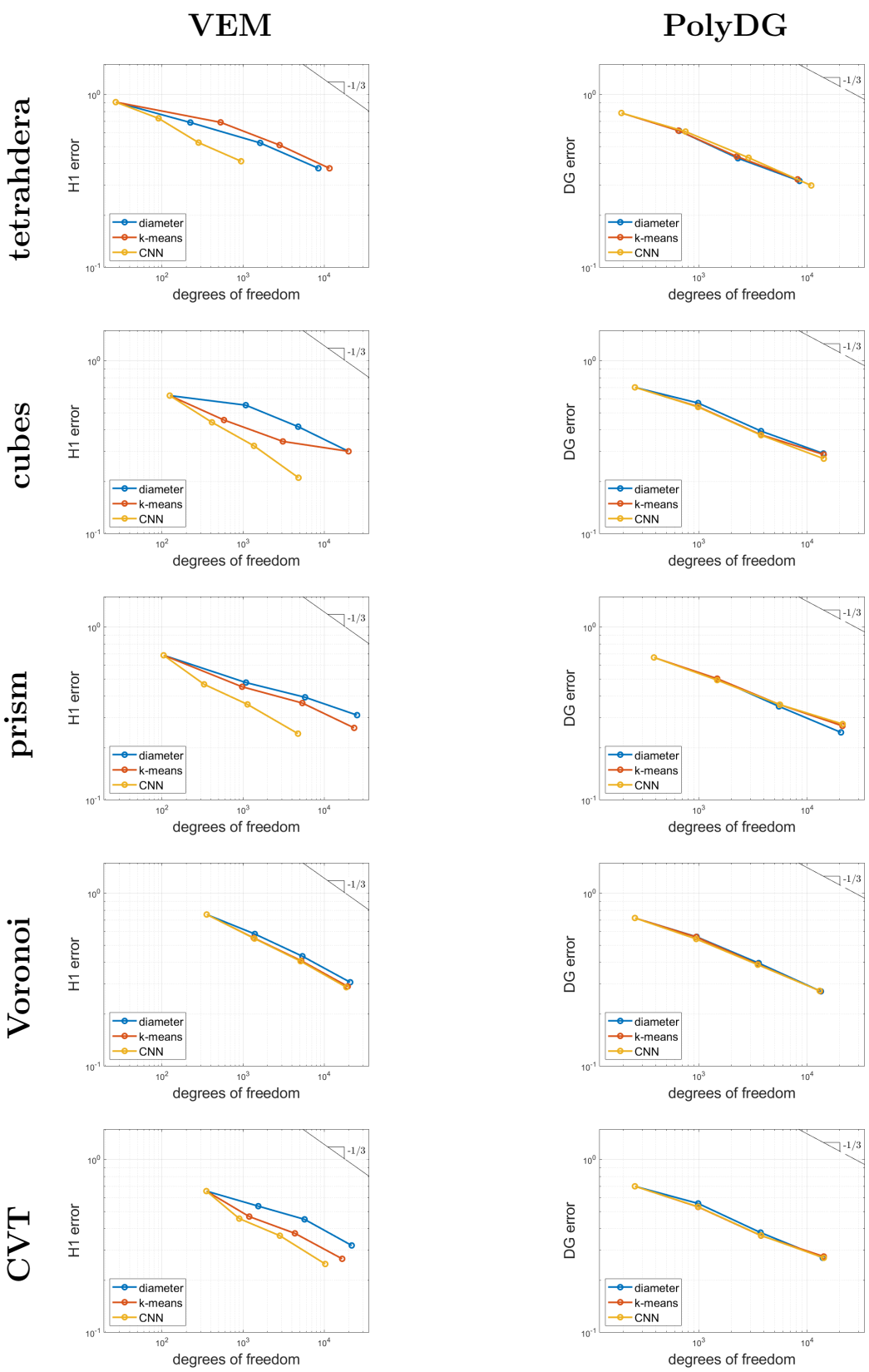

Figure 4.19: Adaptive refinement test case of Section 4.4.1. Computed relative errors as a function of the number of degrees of freedom. Each row corresponds to the same initial grid refined uniformly with the proposed refinement strategies (diameter, k-means, CNN), while each column corresponds to a different numerical method (VEM left and PolyDG right).

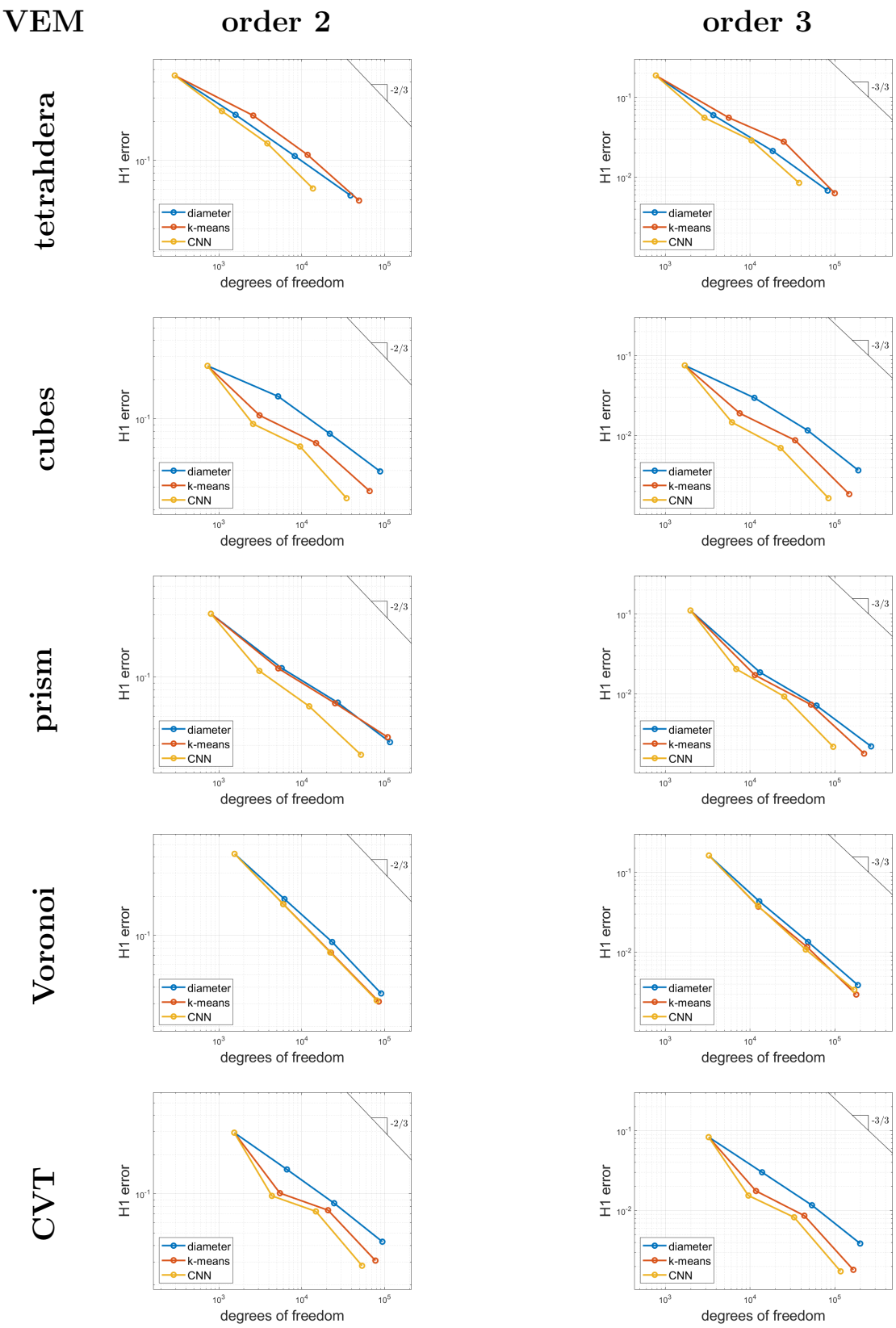

Figure 4.20: Adaptive refinement test case of Section 4.4.1. Computed relative errors as a function of the number of degrees of freedom. Each row corresponds to the same initial grid refined uniformly with the proposed refinement strategies (diameter, k-means, CNN), while column corresponds to employing the VEM of order 2 and 3.

of the polyhedron, which would not be possible otherwise unless explicit and costly geometrical checks are performed. As a result, the initial structure and quality of the grid is preserved, significantly lowering the computational cost and the number of vertices, edges and faces required to refine the mesh. These ML-enhanced refinement strategies are particularly beneficial for the VEM, which is sensitive to mesh distortions and/or vertex proliferation, as the error is decreasing faster using the same number of degrees of freedom. On the other hand, as expected the PolyDG method is less sensitive as the discretization space is not associated with any geometrical entity.
In terms of future research, the use of a posteriori error estimators to drive the refinement process is under investigation. Possible applications include, e.g., fluid flow in heterogeneous porous media [40, 41].

5 | Agglomeration of polygonal grids using graph neural networks

In this chapter[1], we propose the use of Graph Neural Networks (GNNs) [85, 86, 94, 114, 143], deep learning architectures specifically meant to work with graph-structured data, to efficiently handle polygonal mesh agglomeration. Mesh agglomeration, i.e. the process of merging neighbouring elements to obtain a coarse grid, can be naturally performed when dealing with polygonal and polyhedral grids, due to the flexibility in the definition of mesh elements. This operation has multiple applications in the numerical solution of partial differential equations, for example:

- it can be used, with adaptive procedures, to reduce the number of degrees of freedom where not needed because in certain parts of the domain the error is already under control;
- it can be used to generate a hierarchy of (nested) coarser grids starting from a fine mesh of a complex physical domain of interest, in order to employ them in multigrid solvers [5, 12, 15, 28, 29, 55, 144] to accelerate the converge of iterative algebraic;
- it can be employed together with domain decomposition techniques [7, 11, 84, 141] to obtain a meaningful decomposition of the domain, starting from a fine discretization.

The problem of mesh agglomeration can be re-framed as a graph partitioning problem, by exploiting the connectivity structure of the mesh. By exploiting such a representation, GNNs can be applied to solve a node classification problem, where each element is assigned to a cluster, which corresponds an element of the agglomerated mesh. GNNs can process naturally and simultaneously both the graph structure of mesh and the geometrical information that can be attached to the nodes, such the elements areas or their barycentric coordinates. This is not the case of other approaches such as METIS [106],

[1]The results of the thesis are original and are contained in [9]

a standard solver for graph partitioning which can process only the graph information, or k-means, which can process only the geometrical information. The proposed GNN-based algorithms exploit an unsupervised training procedure, where parameters are set to minimize the expected value of the normalized cut, over a database of polygonal grids. To investigate the capabilities of the proposed approaches, we consider a second-order model problem discretized by the PolyDG method in a multigrid framework. We measure effectiveness through an analysis of quality metrics and number of iterations of the the algebraic iterative solver. We also consider the generalization capabilities over a human brain MRI scan section.

5.1. Mesh agglomeration strategies

We recall that the problem of mesh agglomeration can be re-framed as a graph partitioning problem, by exploiting the connectivity structure of the mesh. In particular, the graph representation of the mesh is obtained by assigning a node to each element of the mesh, and connecting with an edge the pair of nodes which are relative to adjacent elements in the original mesh, i.e. polygons that share at least one edge. Moreover, features can be assigned to each node, storing geometrical information such as the area of the element or its barycentric coordinates. Finally, partitioning the nodes of the mesh into proper clusters using a suitable algorithm allows to obtain an agglomerated representation of the original mesh.

5.1.1. METIS

A standard algorithm for partitioning large graphs or meshes and computing fill-reducing orderings of sparse matrices is METIS [106]. As depicted in Figure 5.1, METIS graph partitioning algorithm is based on a multi-level graph bisection procedure, consisting of the following main steps:

1. *Graph coarsening phase*: from the input graph successively smaller graphs are derived by collapsing adjacent pairs of vertices, until the size of the graph is sufficiently small.

2. *Initial partitioning phase*: a partition of the coarsest graph is computed my minimizing the edge cut.

3. *Uncoarsening phase*: the partitioning of the smallest graph is projected back to the successively larger graphs, by assigning the pairs of vertices that were collapsed together to the same partition.

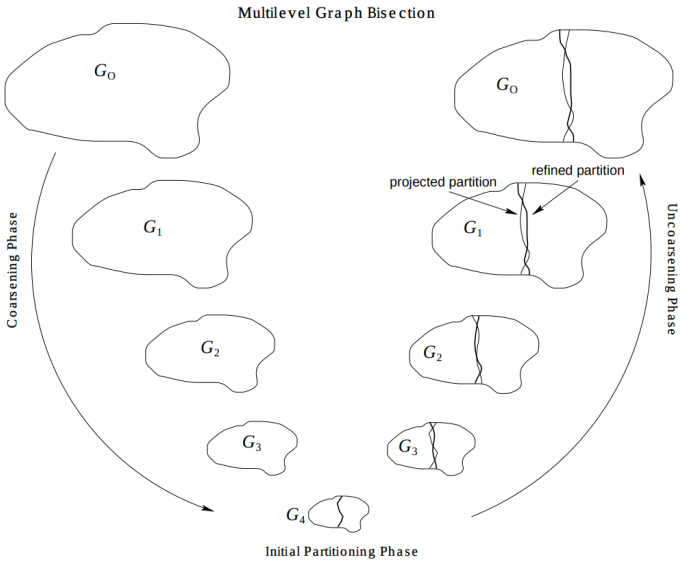

Figure 5.1: Scheme of the METIS graph bisection algorithm, taken from [106].

4. *Refinement phase*: after each uncoarsening step the partition is refined, adjusting nodes close to the interface of the two sets.

Once the graph has been divided into two sets, METIS applies the same algorithm recursively on the new sub-graphs, until the desired number of sets is reached.

5.1.2. Machine learning-based graph partitioning

We propose to employ ML-based bisection models of the form $\mathcal{M}(G, X) = Y$, that take as input a graph G together with a set of features X attached to each node, such as the barycentric coordinates or the area of the mesh elements, and output the vector of probabilities Y of each node belonging to cluster 1 or cluster 2. In order to apply such models for mesh agglomeration, we can use Algorithm 5.1, that recursively bisect the connectivity graph of the input mesh until the agglomerated elements have the desired size. We recall that the diameter of a domain $\mathcal{D}$ is defined, as usual, as $\text{diam}(\mathcal{D}) := \sup\{|x - y|,\ x, y \in \mathcal{D}\}$. Given a polygonal mesh, i.e. a set of non-overlapping polygonal regions $\mathcal{T}_h = \{P_i\}_{i=1}^{N_P}$, $N_P \geq 1$, that covers a domain Ω, we can define the mesh size $h = \max_{i=1:N_P} \text{diam}(P_i)$. Algorithm 5.1 automatically generates a hierarchy of nested grids with different sizes, to be employed e.g. within multigrid solvers. If the model $\mathcal{M}$ does not return a valid partition Y, e.g., because within a set the graph is not connected, a suitable fixing procedure is required. This can be done, for example, by considering the largest suitable connected components as new clusters. Possible choices for the bisection model $\mathcal{M}$ are, but not limited to, the k-means clustering algorithm and GNNs, as we will see in the following.

Algorithm 5.1 General mesh agglomeration strategy

Input: mesh $\mathcal{T}_h$, target mesh size h^*, bisection model $\mathcal{M}$.
Output: agglomerated mesh $\mathcal{T}_{h^*}$.

Function AGGLOMERATE ($\mathcal{T}_h,\ h^*$)

if $\text{diam}(\mathcal{T}_h) \leq h^*$ **then**
  return $\mathcal{T}_h$
else
  Extract the connectivity graph G and features X from $\mathcal{T}_h$
  $Y \leftarrow \mathcal{M}(G, X)$
  Refine partition Y, by minimizing the length of the boundary of the agglomerated polygons.
  Partition $\mathcal{T}_h$ into sub-meshes $\mathcal{T}_h^{(1)}, \mathcal{T}_h^{(2)}$ according to Y.
  $\mathcal{T}_{h^*}^{(1)} \leftarrow \text{AGGLOMERATE}(\mathcal{T}_h^{(1)}, h^*)$
  $\mathcal{T}_{h^*}^{(2)} \leftarrow \text{AGGLOMERATE}(\mathcal{T}_h^{(2)}, h^*)$
  $\mathcal{T}_{h^*} \leftarrow$ merge $\mathcal{T}_{h^*}^{(1)}$, $\mathcal{T}_{h^*}^{(2)}$
end if

5.1.3. The k-means clustering algorithm

n order to apply the k-means algorithm for graph bisection within Algorithm 5.1, we simply use Algorithm 2.1 employing only two clusters and using as features the barycentric coordinates of the mesh elements. This strategy tends to produces "rounded" (star-shaped) agglomerated elements of similar size. Notice that no information on the connectivity graph is taken into account directly. However, for a regular mesh with elements of similar size, barycentric coordinates are usually strongly correlated with elements being adjacent.

5.2. Graph neural networks-based agglomeration

In order to perform grid agglomeration by exploiting effectively both the graph representation and the geometrical features of meshes, we employ GNNs as bisection model in Algorithm 5.1. Examples of GNNs-based models directly applied to the original grid are [85, 86], which employ additional processing based on the graph spectrum to perform a preliminary embedding step, in order to extract features that can later be fed to a GNN-based partitioning module. In our case, the graph extraction of geometrical features such the elements area or their barycentric coordinates can be leveraged to perform a classification task, therefore avoiding the need for an additional spectral embedding module. The graph-bisection model performs a classification task, by taking as input the graph $G = (V, E)$ and the features related to its nodes $X \in \mathbb{R}^{N \times F}$, and outputting a probability

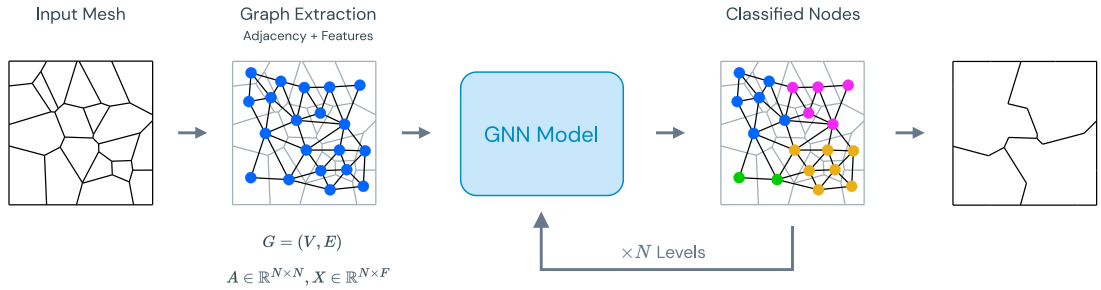

Figure 5.2: General framework for mesh agglomeration via GNNs.

tensor $Y \in \mathbb{R}^{N \times 2}$, where Y_{ij} represents the probability that the node $v_i \in V$ belongs to the partition S_j, with $j = 1, 2$. This approach can be obviously generalized to an arbitrary number of partitions. However, this would require a specific GNN model for each fixed number of classes, while the 2-classes case can be easily extended to a multi-class case by recursively calling the bisection model on the graphs of each partition. The general framework for mesh agglomeration via GNNs is shown in Figure 5.2.

5.2.1. Graph neural network training

The input features matrix is $X \in \mathbb{R}^{N \times 3}$, where the first columns contains the area of each mesh element followed by the two coordinates of its barycenter. The GNN architecture we employed first applies a INorm layer, then four SAGEConv layer with features dimensions 64, then three Linear layer with progressively decreasing output dimensions (32, 8 and 2) and finally a Softmax layer. Each SAGEConv layer is followed by a tanh activation function to keep the features inside the interval $[-1, 1]$, so that the geometrical information regarding the re-scaled coordinates will be kept in the same domain, as information flows through the layers. To further simplify the classification process, the INorm layer also rotates of 90 degrees the barycentric coordinates if the input mesh is more stretched along the y axis rather than the x axis. The resulting model consists of approximately 28k parameters, where roughly 25k resulting from the SAGEConv layers and the remaining from the Linear layers. A scheme of the described GNN model is shown in Figure 5.3.

The training of the models has been performed by employing an unsupervised approach by minimizing the expected normalized cut, as described in Section 2.2. To perform the training a train and a validation datasets were generated. The training dataset consists of meshes of the following type: grids of regular squares, grids of regular triangles, grids of triangles with random perturbation of the vertices, grids of Voronoi with random location of the seeds. The database is composed of 800 meshes, with 200 meshes per type, while

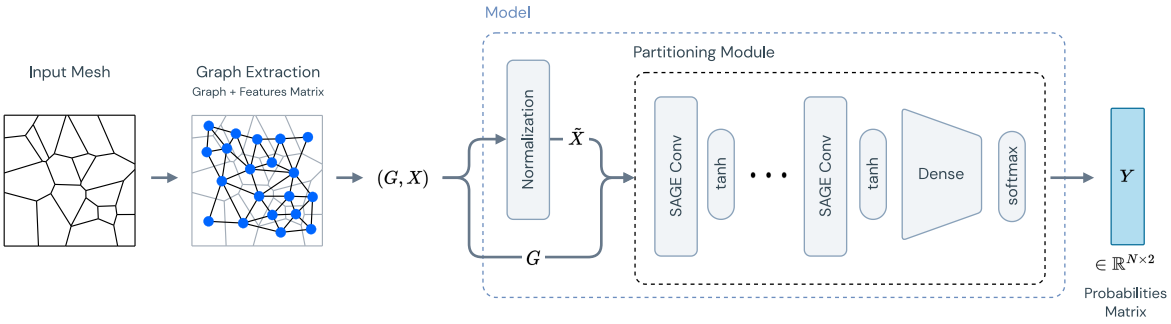

Figure 5.3: GNN model structure, consisting of a normalization module, responsible of rescaling the features extracted from the input mesh, and of a partitioning module made of a stack of graph convolutions and a dense classifier. The output is a matrix of probabilities $Y \in \mathbb{R}^{N\times 2}$, containing the probabilities of belonging either to one of the two classes for each node.

the validation dataset consists of 200 meshes, 50 per type. The cardinality of the datasets has been chosen to keep a 80-20 split ratio between train and validation respectively. Training has been performed using the Adam optimizer [109] with a learning rate 1e-5, L^2 regularization coefficient 1e-5 and mini-batch size 4. Training was performed for 300 epochs in approximately 35 minutes on Google Colab cloud platform, using a 2.20GHz Intel Xeon processor, with 12GB of RAM memory and NVIDIA Tesla T4 GPU with 16GB of GDDR6 memory.

5.3. Validation on a set of polyhedral grids

In this section we compare the performance of the proposed algorithms. To evaluate the quality of the refined grids, we employ the Uniformity Factor (UF) and Circle Ratio (CR) quality metrics[23] introduced in Section 3.3. We consider four different grids of domain $(0, 1)^2$: a grid of regular triangles, a grid of triangles with random location of the vertices, a Voronoi grid with random location of the seeds, and a grid of regular squares. In Figure 5.4 these grids have been agglomerated using METIS, k-means and GNNs strategies. For METIS the target number of mesh elements is $N_0/16$, where N_0 is the initial number of elements, while for k-means and GNN the target mesh size in Algorithm 5.1 is $h_0/4$, where h_0 is the initial mesh size. As we can see, the k-means and the GNN algorithms are capable to recover a regular grid of squares when starting from regular meshes (triangles and squares), while this is not the case for METIS. In Figure 5.5 we show the box plots of the computed quality metrics for the selected grids. In general, quality metrics are lower for the METIS algorithm, while k-means and GNNs have comparable performance. This is further confirmed by Tables 5.1 and 5.2, where we report the average values of UF and CR. In particular, the performance difference is much more evident for regular grids. In

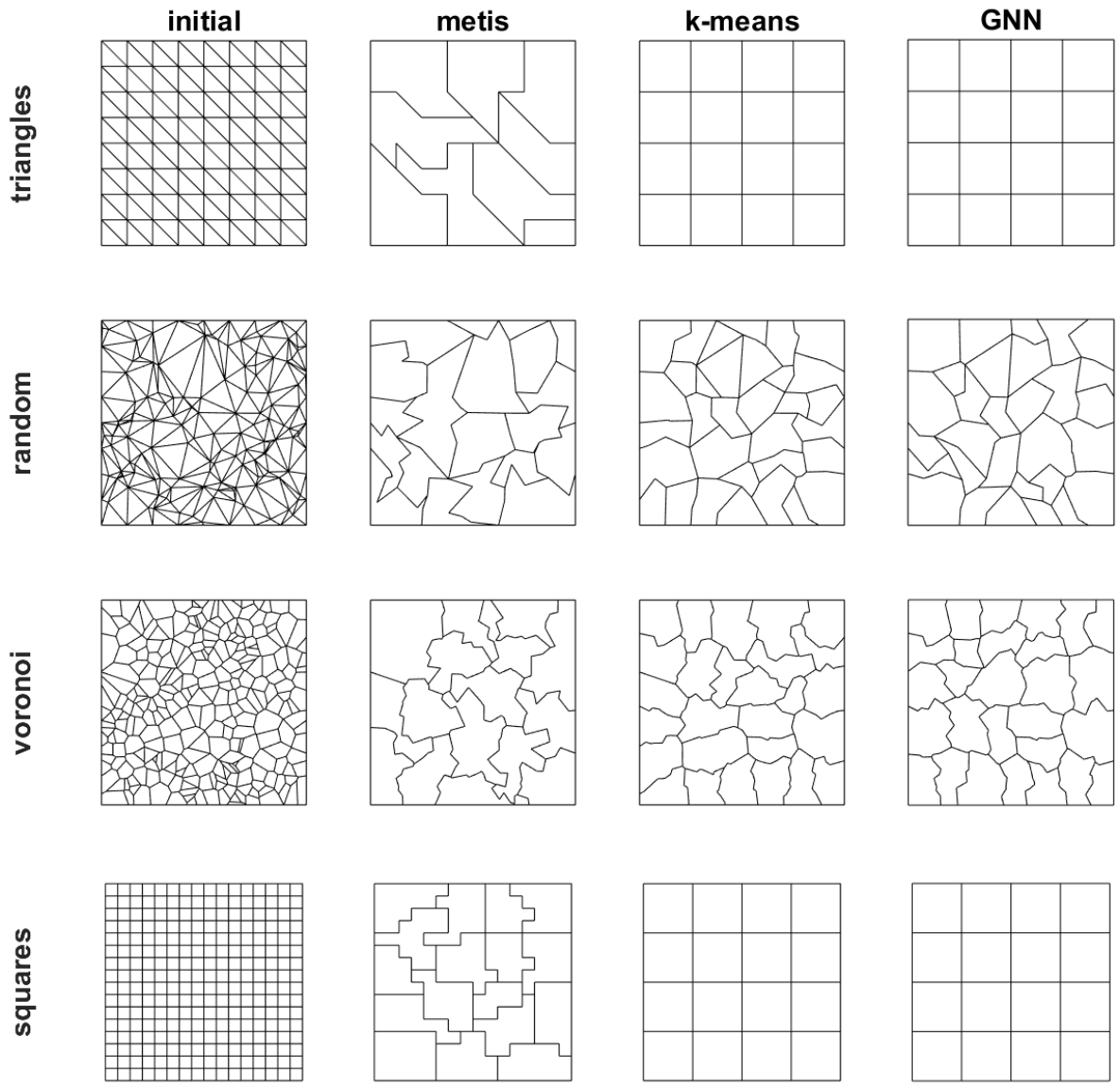

Figure 5.4: Initial grids (first column) and corresponding agglomerated versions (second to fourth column) obtained based on employing different strategies. Each row corresponds to the same initial grid (from top to bottom: regular triangles, random triangles, Voronoi, squares) while each column corresponds to the same agglomeration strategy (from left to right: initial grids, METIS, k-means, GNN).

UF	metis	k-means	GNN
triangles	0.7648	1.0000	1.0000
random	0.6115	0.7003	0.7418
Voronoi	0.7605	0.8105	0.8225
squares	0.7725	1.0000	1.0000

Table 5.1: Average values of the UF for the agglomerated grids reported in Figure 5.4, obtained based on employing different agglomeration strategies (METIS, k-means, GNN).

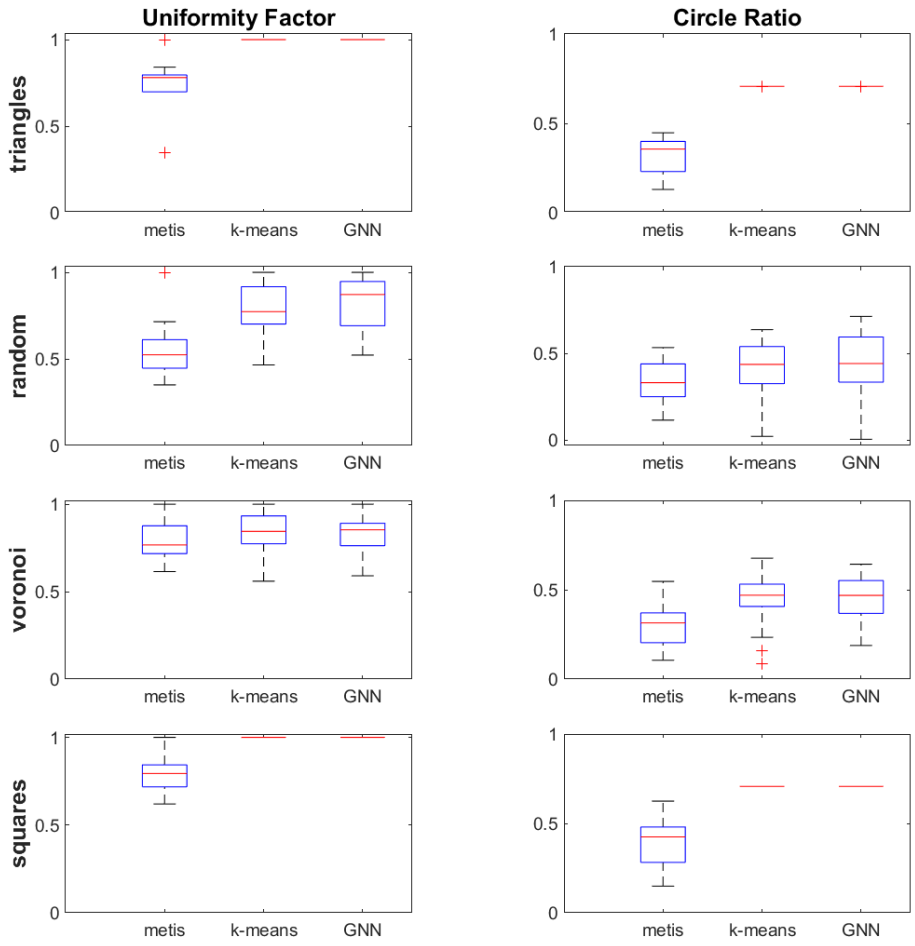

Figure 5.5: Box plots of the computed quality metrics (UF and CR) for the agglomerated grids reported in Figure 5.4 (second to fourth column), obtained based on employing different agglomeration strategies (METIS, k-means, GNN). Some box plots collapse into a single line because the corresponding metric has the same value for all mesh elements.

CR	metis	k-means	GNN
triangles	0.3447	0.7071	0.7071
random	0.3399	0.4190	0.3928
Voronoi	0.3056	0.4509	0.4845
squares	0.3733	0.7071	0.7071

Table 5.2: Average values of the CR for the agglomerated grids reported in Figure 5.4, obtained based on employing different agglomeration strategies (METIS, k-means, GNN).

UF relative	k-means/metis	GNN/metis
triangles	1.3076	1.3076
random	1.1452	1.2130
Voronoi	1.0658	1.0815
squares	1.2945	1.2945

Table 5.3: Relative UF: performance improvement with respect to METIS, i.e., ratio between the average UF of the ML-strategies (k-means and GNN) and METIS, for the agglomerated grids reported in Figure 5.4.

CR relative	k-means/metis	GNN/metis
triangles	2.0512	2.0512
random	1.2327	1.1554
Voronoi	1.4755	1.5853
squares	1.8940	1.8940

Table 5.4: Relative CR: performance improvement with respect to METIS, i.e., ratio between the average CR of the ML-strategies (k-means and GNN) and METIS, for the agglomerated grids reported in Figure 5.4.

Tables 5.3 and 5.4 we also report the relative performance with respect to METIS, i.e., the ratio between the average UF and CR metrics of the ML-strategies (k-means and GNN) and METIS. This further highlights the gain is using ML-based strategies. In general, ML-based strategies (either the k-means and the GNN algorithms), seem to preserve the initial geometry and quality of the grids, because the geometric information attached to the nodes is taken into account, making them suitable for adaptive mesh coarsening. This is not the case for the METIS algorithm, because it processes only the information coming from the graph topology of the mesh.

5.3.1. Application to a computational mesh stemming from a human brain MRI-scan

In order to further test the generalization capabilities of our models, we apply them on a much more complex domain with respect to the meshes considered so far. In particular, we consider the mesh of a section of a human brain coming from an MRI-scan, consisting of 14372 triangular elements. The domain is highly non-convex and presents many constrictions and narrowed sections. We agglomerated such a grid using METIS, k-means and GNN algorithms: the result is reported in Figure 5.6. For k-means and GNN the target mesh size in Algorithm 5.1 is $0.2D$, where D is the maximum distance between any two vertices of the initial mesh. For METIS the target number of mesh elements is 50, which corresponds approximately to the same mesh size. In Figure 5.7 we show also

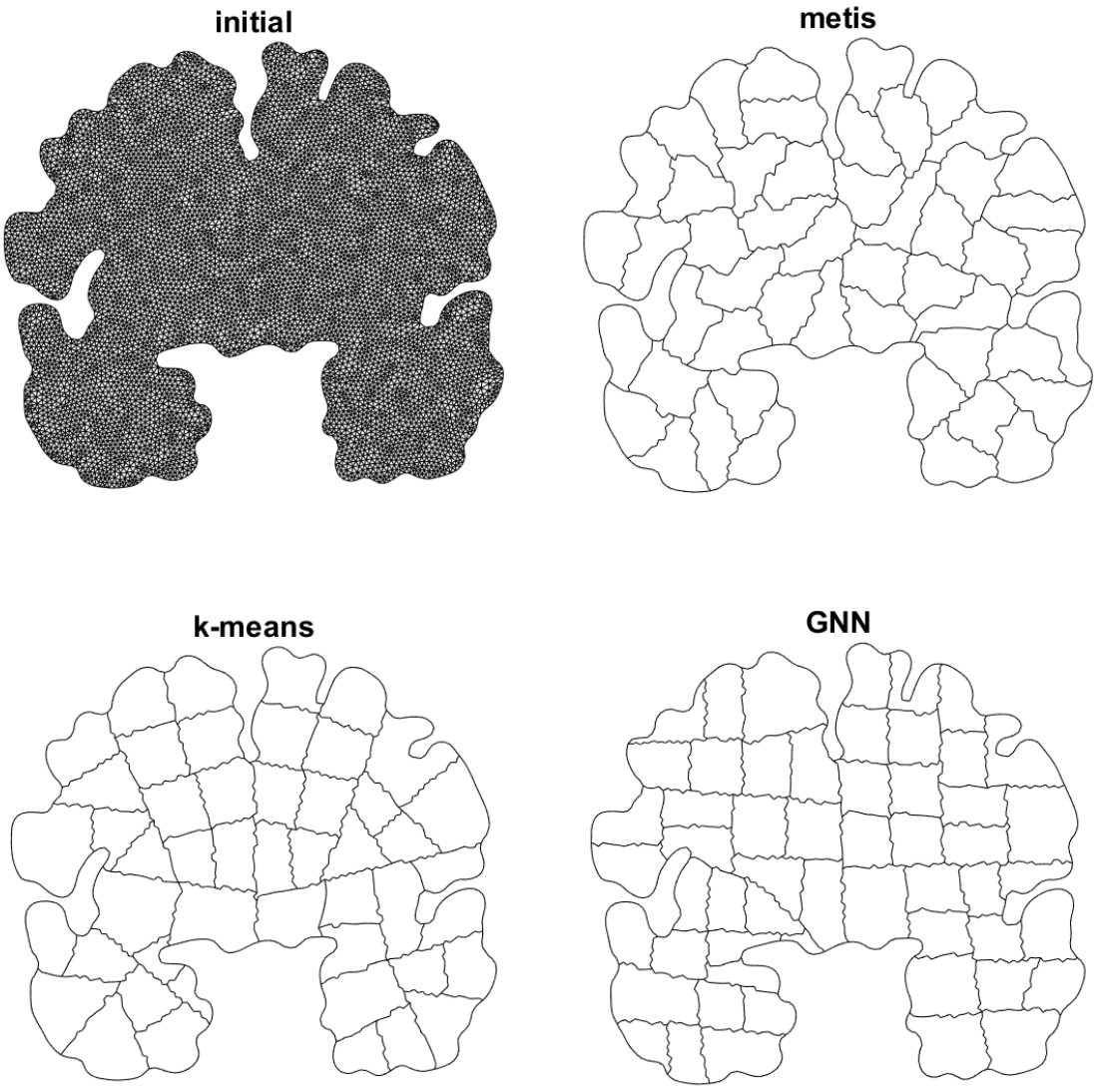

Figure 5.6: Agglomerated meshes using different strategies (METIS, k-means, GNN), starting from an initial grid of a human brain MRI-scan, consisting of 14372 triangular elements as shown in the top-left figure.

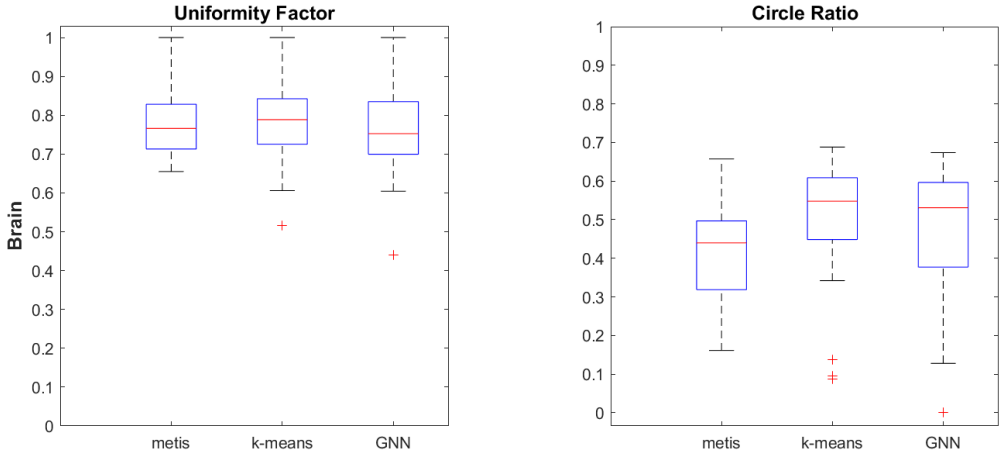

Figure 5.7: Box plots of the computed quality metrics (UF and CR) for the agglomerated MRI brain scan meshes reported in Figure 5.6, obtained based on employing different agglomeration strategies (METIS, k-means, GNN).

metric	metis	k-means	GNN
UF	0.7808	0.7875	0.7672
CR	0.4134	0.5146	0.4841
UF relative	1	1.0085	0.9826
CR relative	1	1.2449	1.1712

Table 5.5: Average and relative values (with respect to METIS) of the UF and the CR for different agglomeration methods (METIS, k-means, GNN), applied to the section of the MRI brain scan, for the agglomerated MRI brain scan meshes reported in Figure 5.6.

the corresponding box plots of the quality metrics and in Table 5.5 we report the average values. As we can see, performance are comparable in terms of UF, while the ML-based strategies achieve on average a higher "roundness", measured in terms of CR. This indicates good generalization capabilities of the GNN algorithm, as the considered mesh was very different from the ones included in the training set, both in terms of shape, dimensions and number of elements. The k-means reasonably performs slightly better than the GNN, because it does not require prior information coming from a database at the cost of having a higher online computational cost, as we will see in the next section.

5.3.2. Runtime performance

The main advantage in using Neural Network-based solutions is the low computational cost for online inference, with respect to k-means or METIS. We measured the computational cost of the different graph partitioning algorithms on 21 random Voronoi meshes with an increasing number of elements from 25 to 5000. Since the runtimes are not de-

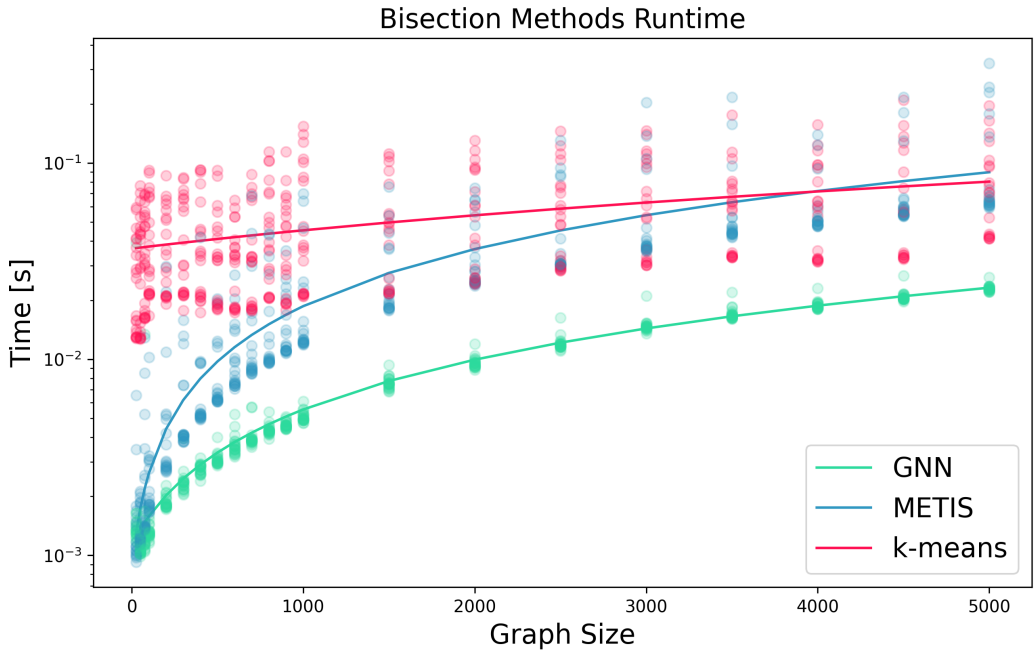

Figure 5.8: Runtime performance for different graph bisection models (METIS, k-means, GNN) as a function of the number of nodes in the connectivity graph of Voronoi meshes. The y-axis is in logarithmic scale.

terministic, we performed a sampling of 20 runtimes for each mesh. Results are shown in Figure 5.8. As we can see, the GNN algorithm outperforms the METIS and the k-means ones. For a mesh with 5000 elements we have that GNN is 4.4836 times faster than Metis and 3.5670 times faster than k-means. Such a performance improvement is expected to grow asymptotically, due to the fact that the computational complexity of the Neural Network mainly scales with the dimension of the data manifold [126], while METIS and k-means mainly scale with the dimension of the graph. Being able to compute the solution in fast manner, even with a slight loss of accuracy, can be particularly beneficial when using multigrid scheme as preconditioners.

5.4. Applications to agglomeration-based multigrid methods

In this section we test the effectiveness of the proposed agglomeration strategies, to be used in combination with polygonal finite element discretizations within a MultiGrid (MG) framework [5, 12, 15, 28, 29, 55, 144]. We consider the following model problem:

grids	ℓ	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	2	21	9	9	114
	3	21	9	9	
	4	21	9	9	
random	2	46	41	32	655
	3	46	41	32	
	4	46	41	32	
Voronoi	2	19	16	15	348
	3	19	16	15	
	4	19	16	15	
squares	2	20	17	17	109
	3	20	17	17	
	4	20	17	17	

Table 5.6: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 2, 3, 4$, $p = 1$, $m = 3$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

find $u \in V = \mathrm{H}^2(\Omega) \cap \mathrm{H}_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v \quad \forall v \in V, \tag{5.1}$$

with Ω and $f \in L^2(\Omega)$ the forcing term, selected in such a way that the exact solution is given by $u(x, y) = \sin(\pi x)\sin(\pi y)$. We consider the V-cycle MG algorithm with additive Schwarz smoothing within a PolyDG discretization framework [5], described in Section 1.4. We employ the initial grids shown in the first column of Figure 5.4 (triangles, random, Voronoi, squares), agglomerated using the proposed strategies (METIS, k-means, GNNs). In particular, for each initial mesh we construct three agglomerated grids of increasing size: for METIS the target numbers of mesh elements are $N_0/64$, $N_0/16$, $N_0/4$, where N_0 is the initial number of elements, while for k-means and GNN the target mesh sizes in Algorithm 5.1 are $h_0/8$, $h_0/4$, $h_0/2$, where h_0 is the initial mesh size. These correspond to the different levels of the V-cycle algorithm, where level 1 corresponds to the initial grid (level 3 grids are the ones reported in Figure 5.4). As performance metric, we consider the Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} in solving the algebraic formulation of problem (5.1). We vary the following parameters: number of levels employed ℓ, polynomial degree p, number of smoothing steps m (with $m = m_1 = m_2$ in Algorithm 1.3). In Table 5.6 we report the iteration count when varying the number of levels $\ell = 2, 3, 4$ with $m = 3$ and $p = 1$.

grids	ℓ	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	3	230	67	67	114
	4	227	66	66	
random	3	706	551	405	655
	4	708	577	407	
Voronoi	3	154	143	129	348
	4	154	143	131	
squares	3	178	162	162	109
	4	180	162	162	

Table 5.7: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 3, 4$, $p = 1$, $m = 1$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

We can observe the following:

- The iteration count of the MG methods are significantly lower than the ones the of the classical Conjugate Gradient (CG) method, meaning the considered MG implementation is indeed effective.
- The iteration count of the k-means and GNN algorithms are lower than the ones of METIS, meaning the higher grid quality provided by the ML-based agglomeration strategies can help accelerating the convergence of the numerical method, with GNN having the best performance.
- The iteration count required when varying the number of levels ℓ is constant, meaning it is independent of the granularity of the underlying grid and therefore scalable in terms of mesh size.

In Table 5.7 we also consider the case for $m = 1$. Despite using such a low value, the MG method still converges in all cases. As expected, the iteration count increases but the iteration count of the ML strategies is still lower with respect to the ones of the CG, while this is not the case for the METIS algorithm.
In Table 5.8 we report the iteration count when varying the number of smoothing steps $m = 3, 5$ with $\ell = 3$ and $p = 1$. As we can see, when the number of steps increases the iteration count decreases, as well as the performance difference between the different methods. In Table 5.9 we report the iteration count when varying the polynomial dgree $p = 1, 2, 3$ with $\ell = 3$ and $m = 3$. As expected, since m is fixed, when p increases also the iteration count increases, because more degrees of freedom are involved in the formulation of the problem [5].

grids	m	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	3	21	9	9	114
	5	10	5	5	
random	3	46	41	32	655
	5	20	18	15	
Voronoi	3	19	16	15	348
	5	10	8	8	
squares	3	20	17	17	109
	5	9	8	8	

Table 5.8: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 3$, $p = 1$, $m = 3, 5$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

grids	p	Agglomeration-based MG			CG
		metis	k-means	GNN	
triangles	1	21	9	9	114
	2	49	10	10	355
	3	63	17	17	1101
random	1	46	41	32	655
	2	89	76	53	4425
	3	115	102	64	9893
Voronoi	1	19	16	15	348
	2	36	25	25	987
	3	39	33	27	3235
squares	1	20	17	17	109
	2	45	34	34	365
	3	46	52	52	703

Table 5.9: Iteration count of the MG algorithm to reduce the (relative) residual below 10^{-6} employing different initial grids (triangles, random, Voronoi, squares) agglomerated with different strategies (METIS, k-means, GNN) with $\ell = 3$, $p = 1, 2, 3$, $m = 3$. As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

In general, the considered experiments highlights that when employing ML-based agglomeration strategies a lower iteration count is required by the MG solver with respect to METIS, thanks to the higher quality of the produced grids, with GNN having the best performance.

5.5. Concluding remarks

In this chapter, we presented a new ML-based framework to efficiently handle the open problem of mesh agglomeration for polygonal grids. We first re-framed the problem of mesh agglomeration as a graph partitioning problem, by exploiting the graph representation of the connectivity structure of mesh elements. We then developed in this context a framework to employ ML-based strategies, such as k-means and GNNs which can exploit the geometrical information of the grid. We trained GNNs to perform graph partitioning over a suitably constructed database of meshes. We compared the performance of the ML-based strategies with METIS, a classical algorithm for graph partitioning, over a set of different grids, featuring also complex real domains such a brain MRI-scan. Results show that ML-based strategies are more robust and can better preserve mesh quality, making them suitable for adaptive mesh coarsening. We also employed the proposed algorithms in the context MG methods in PolyDG framework, showing a lower iteration count for ML-based strategies, with GNN having the best performance. Indeed, GNNs have the advantage to process naturally and simultaneously both the graph structure of mesh and the geometrical information, such the elements areas or their barycentric coordinates. This is not the case of METIS [106], which is meant to process only the graph information, or k-means, which can process only the geometrical information. Moreover, GNNs have a significantly lower online computational cost. Being able to compute the solution in fast manner, even with a slight loss of accuracy, can be particularly beneficial when using multigrid scheme as preconditioners.

6 | Future developments: variational physics-informed neural networks for the solution of the Helmholtz impedance problem

In this chapter[1], we explore the use of Variational Physics-Informed Neural Networks (VPINNs) to solve the one-dimensional Helmholtz impedance problem. In particular, neural networks are used to minimize the residual of the weak formulation of the problem. The main reason to use artificial neural networks to tackle the Helmholtz problem resides in their powerful approximation properties for highly oscillatory functions, which are of particular interest for the chosen application. Indeed, the required number of parameters of the network, in particular the number of its layers, scales logarithmically with respect to the considered frequency.

This work is still in a preliminary phase, and many numerical experiments still need to be performed in order to actually access the feasibility of the approach.

6.1. Variational formulation

The Helmholtz equation has the general form

$$-\nabla^2 u(\underline{x}) - k^2 u(\underline{x}) = f(\underline{x}), \quad \underline{x} \in \Omega,$$

where $k \in \mathbb{R}^+$ is called wavenumber. Larger values of the wavenumber result in more oscillatory solutions. This equation often arises in various physical problems such as seismology [3] and acoustics [138].

[1]The results of the thesis are original and are contained in [97].

In this work, we will consider the one-dimensional Helmholtz equation with impedance boundary conditions

$$\begin{cases} -u''(x) - k^2 u(x) = f(x), & x \in \Omega = (a,b), \\ -u'(a) - \imath k u(a) = g_a, \\ +u'(b) - \imath k u(b) = g_b, \end{cases} \tag{6.1}$$

with $g_a,\ g_b \in \mathbb{C}$. The Helmholtz equation with impedance boundary conditions is well-posed, and has a unique complex solution. The classical variational formulation of (6.1) reads: find $u \in H^1(\Omega)$ such that

$$\mathrm{a}(u,v) = l(v) \quad \forall v \in H^1(\Omega),$$

where $\mathrm{a} :\ H^1(\Omega) \times H^1(\Omega) \to \mathbb{C}$ and $l : H^1(\Omega) \to \mathbb{C}$ are defined as

$$\begin{aligned} \mathrm{a}(u,v) &= \int_a^b u'v'dx - k^2 \int_a^b uvdx - \imath k(u(a)v(a) + u(b)v(b)) \\ l(v) &= \int_a^b fvdx + g_a v(a) + g_b v(b) \end{aligned}$$

However, when the wavenumber is large the above formulation is sign-indefinite, in the sense that the bilinear forms cannot be bounded below by a positive multiple of the appropriate norm squared. This indefiniteness has implications for both the analysis and the practical implementation of finite element methods. To overcome this limitation, an alternative variational formulation is proposed in [119]. In particular, let us consider the Hilbert space $V := H^2(\Omega)$ with the inner product $(.,.)_V : V \times V \to \mathbb{C}$ defined as

$$\begin{aligned} (u,v)_V :=& k^2 (u,v)_{L^2(\Omega)} + (u',v')_{L^2(\Omega)} + k^{-2} (u'',v'')_{L^2(\Omega)} \\ &+ 2k^2 (u(a)\overline{v(a)} + u(b)\overline{v(b)}) \\ &+ k^2 \left(u'(a)\overline{v'(a)} + u'(b)\overline{v'(b)} \right), \quad u,v \in V \end{aligned}$$

and the induced norm $\|\cdot\|_V := \sqrt{(.,.)_V}$, where $(.,.)_{L^2(\Omega)}$ corresponds to the $L^2(\Omega)$-inner product in Ω. For $k, \beta \in \mathbb{R}^+$, we define the sesquilinear form $a : V \times V \to \mathbb{C}$ as

$$\begin{aligned} \mathrm{a}(u,v) := & \int_\Omega u'\overline{v'}dx + k^2 \int_\Omega u\bar{v}dx + \int_\Omega \left(\mathcal{M}u + \frac{A}{k^2}\mathcal{L}u \right) \overline{\mathcal{L}v}dx \\ & - \imath k(\overline{\mathcal{M}v(b)}u(b) + \overline{\mathcal{M}v(a)}u(a)) - \overline{v'}\mathcal{M}u\big|_{\partial\Omega} - xk^2 u\bar{v}\big|_{\partial\Omega} + xu'\overline{v'}\big|_{\partial\Omega} \end{aligned} \tag{6.2}$$

and the linear form $l : V \to \mathbb{C}$ as

$$l(v) := \int_\Omega \left(\overline{\mathcal{M}v} - \frac{A}{k^2}\overline{\mathcal{L}v} \right) f dx + \overline{\mathcal{M}v(b)} g_b + \overline{\mathcal{M}v(a)} g_a \tag{6.3}$$

where $A \in \mathbb{R}_+$and the operators $\mathcal{M}$ and $\mathcal{L}$ are defined as

$$\mathcal{M}v := xv' - \imath k\beta v$$
$$\mathcal{L}v := v'' + k^2 v$$

As before, the new vartiaional problem reads: find $u \in V$ such that $\mathrm{a}(u, v) = l(v)$ for all $v \in V$. It can be proved that setting $\beta = 1$ and $A = 1/3$ makes the sesquilinear form $a : V \times V \to \mathbb{C}$ V-elliptic (or coercive) according to

$$\mathfrak{R}\{\mathrm{a}(u, v)\} \geq \gamma \|v\|_V^2 \quad \forall v \in V,$$

where $\gamma = \frac{1}{6}$ and $\mathfrak{R}\{\cdot\}$ indicates the real part. Notice that the ellipiticity constant is independent of the wavenumber, making this formulation stable for high frequencies.

6.2. Variational physics-informed neural networks

Physics-Informed Neural Networks (PINNs) [128] make use of prior knowledge that stems from the physics of the system as a regularization method to narrow the range of the admissible solutions. The approach consists in using a neural network to directly approximate the solution of the differential problem, by optimizing its parameters based on a suitable loss function that takes into account the residual in strong form the of PDE, the initial conditions and the boundary conditions. The equation is enforce strictly in quadrature points inside the domain.

Variational Physics-Informed Neural Networks (VPINNs) [108] are a variation of the PINNs scheme where the residual is taken in variational form, instead of using it in strong form. This Petrov-Galerkin formulation of the PINNs scheme makes use of the space of the neural networks as trial space, while as a test space different choices can be made, such as trigonometric functions, Legendre polynomials, Lagrangian basis functions or also neural networks.

Let $\mathcal{N}$ be the space of neural networks and $V_K = \mathrm{span}\{v_1, v_2, \cdots, v_K\}$ be the space of the test functions. The VPINNs formulation of the Helmholtz impedance problem reads:

find $u_N \in \mathcal{N}$ that minimizes

$$\mathcal{L}\left(u_N\right) = \frac{1}{K}\sum_{k=1}^{K}\left|a\left(u_N, v_k\right) - l\left(v_k\right)\right|^2,$$

where $a(\cdot,\cdot)$ and $l(\cdot)$ are defined respectively in equations (6.2) and (6.3). Notice that no additional artificial loss terms are needed to satisfy the impedance boundary conditions, contrary to the case of Dirichlet boundary conditions, which would require to introduce additional parameters to weight the correspondent contribution.

The integrals appearing in this formulation can be discretized using different quadrature rules, such as Legendre-Gauss-Lobatto, of the form

$$\int_a^b f(x)dx \simeq \sum_{i=1}^{N_q} w_i f\left(x_i\right),$$

where $\{w_i\}_{i=1}^{N_q}$ and $\{x_i\}_{i=1}^{N_q}$ are respectively the quadrature weights and nodes. When using test functions with local support, such as Lagrangian basis functions, the quadrature points should be chosen within the support, in order to improve accuracy.

Compared to classical PINNs, this approach has the following advantages:

- by using test functions with local support, the learning of the solution can be more easily controlled in a local and stable manner;
- it less likely to fall in local minima, because the differential equation is not evaluate point-wise but globally;
- it is more natural from the point of view of the theory of PDEs, allowing to exploit formulations that can be written only in variational form, which is the case of the formulation of the Helmholtz problem described in Section 6.1;
- ultimately, it can reduce the training time of the neural network and it can increase the accuracy of the solution.

6.2.1. Approximation using deep neural networks

Let $\mathcal{N}$ be the space of a fully-connected multilayer perceptron (MLP) with 1 node at the input layer, N nodes on each of D hidden layers and 2 nodes at the output layer, as depicted in Figure 6.1. The two-dimensional output represents the real and the imaginary parts of the solution: $u_N(x) = u_{N,1}(x) + \imath u_{N,2}(x)$. For this generic architecture of the MLP,

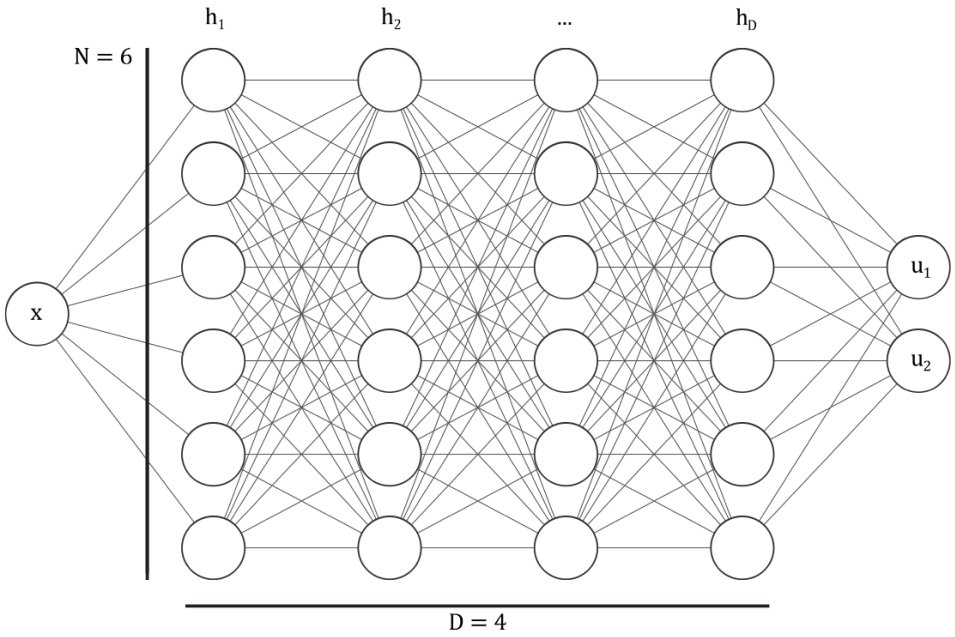

Figure 6.1: MLP with 4 hidden layer and $N = 6$ for the solution of the Helmholtz problem.

each element of the the two-dimensional solution could be expressed as

$$u_{N,i} = c_0^i + \sum_{j=1}^{N} c_j^i \sigma\left(h^D \circ h^{D-1} \circ \cdots \circ h^1(x)\right), \quad i \in \{1, 2\}$$

where $\sigma(x)$ is a non-linear activation function and the output of the hidden layers $\{h^d\}_{d=1}^D$ are defined as

$$\begin{aligned} h_i^1 &= \sigma\left(w_i^1 x + b_i^1\right), \\ h_i^d &= \sigma\left(\sum_{n=1}^{N} w_{i,n}^d h_n^{d-1} + b_i^d\right), \quad d = 2, \cdots, D. \end{aligned}$$

Because the considered weak formulation of the Helmholtz problems involves computing the second order derivatives of the solution, we employ $\sigma(x) = \tanh(x)$ in order to have u_N smooth.

Definition 6.1. *For $D \in \mathbb{R}^+$, let the set $\mathcal{S}_D \subseteq C^\infty([-D, D], \mathbb{R})$ be given by*

$$\mathcal{S}_D = \left\{ f \in C^\infty([-D, D], \mathbb{R}) : \left\| f^{(n)}(x) \right\|_{L^\infty[-D,D]} \leq n!, \text{ for all } n \in \mathbb{N}_0 \right\}$$

Consider now the following oscillatory texture, of particular interest as potential solution

of the Helmholtz problem, of the form

$$u(x) = \cos(kg(x))h(x), \tag{6.4}$$

with $k \in \mathbb{R}^+$, $g, h \in \mathcal{S}_D$. Thanks to the approximation properties of neural networks, in order to approximate (6.4) up to a desired tolerance, the number of layers of constant width required scales logarithmically with respect to k, as shown in [69, 125] (see also [92, 93, 146]). This is not the case for the FEM applied to weak formulation of the Helmholtz impedance problem descibed in Section 6.1, as analyzed in [76]. Indeed, in this case in order to start observing a convergent behaviour for such a scheme, the number of basis functions needs to be greater than a threshold which scales linearly with the wave number.

6.3. Training strategies

Despite the powerful theoretical approximation properties of neural networks, in practice training remains the main limitation to the applicability of this approach, due to the presence of multiple local minima in the loss function and of the expensive optimization processes. In order to alleviate this problem, we propose the following strategies, to be possibly used in combination.

6.3.1. Hybrid training

Usually gradient based algorithms, such as Adam [109], are applied to train instinctively all of the parameters of the network, because of their non-linear dependence from the loss function. However, the parameters of the output layer of the network may be found, deepening on the considered loss function, by using a least-squares approach. In particular, the weights of the last layer are computed as the real and imaginary part of the least-square solution of the system of equations

$$Ac = f$$

where $A \in \mathbb{C}^{K\times(N+1)}$ and $f \in \mathbb{C}^{N+1}$ are define as

$$\begin{aligned} &A_{i,j} = a\left(\sigma(h_j^D), v_i\right) \quad j \leq N, \\ &A_{N+1,j} = a(1, v_j) \\ &f_i = l\left(v_i\right) \end{aligned}$$

for $i = 1, 2, \ldots, K$. By alternating the training of these "linear" parameters and of the remaining "non-linear" ones, it is possible to speed up the training process. This approach is described in [65] and it is analogous with the concept of adaptive basis for classical FEMs. The basis functions are represented by $\{\sigma(h^d)\}_{d=1}^{D}$ in the neural network architecture.

6.3.2. Physics-based initialization

An important factor for training a neural network is the initialization of the parameters [116]. Consider a shallow neural network, i.e. with one hidden layer, with N nodes. We initialize the weights of such layer as $w_i^1 = k$, where k is the wave number of the equation. The biases are initialized as $b_i^1 = -w_i^1 x_i^*$, where x_i^* for $i = 1, 2, \ldots, N$ is uniformly distributed on the interval Ω. With this initialization, the output of the shallow network could be expressed as

$$u_{N,i} = c_0^i + \sum_{j=1}^{N} c_j^i h_j^1 = c_0^i + \sum_{j=1}^{N} c_j^i \sigma\left(k\left(x - x_j^*\right)\right).$$

Finally, the linear parameters of the layers can be initialized using the least squares approach described in Section 6.3.1. This kind of initialization is particularly well suited for oscillatory solutions with leading frequency k.

6.3.3. Adaptive mesh refinement

This approach is based on the following analogies with classical FEMs:

- deep neural networks with ReLU activation function are piece-wise linear, corresponding to linear finite elements [96];
- the partition of the physical domain induced by the neural network, determined by where the function changes its trend, corresponds to the mesh;
- the loss function, computed locally for the different mesh elements induced by the neural network, corresponds to an estimator of the error.

Inspired by classical FEMs [49], it is possible introduce a scheme of the following form:

1. solve: train the neural network to solve the desired problem;
2. estimate: estimate the error by computing the loss function for each mesh element;
3. refine: insert new neurons, by initializing its weights in such a way to partition the

desired fraction α of elements with the highest estimated error.

The idea is that by refining the mesh induced by the neural network, we obtain an initialization with a richer space of basis functions where needed.

The dual of this approach consists in performing adaptive mesh refinement to enlarge the test space, using again the loss function as error estimator. The refinement can be performed in the classical sense if the test space is defined through a mesh, as for standard Lagrangian basis functions.

6.3.4. Incremental frequency approach

Consider functions

$$f : (x, y)^T \to (\cos(x), \sin(x))^T,$$
$$g : (x, y)^T \to (2xy, x^2 - y^2)^T.$$

By exploiting trigonometric identities we can double the initial frequency by composition

$$\sin(2x) = 2\sin(x)\cos(x),$$
$$\cos(2x) = \cos^2(x) - \sin^2(x),$$
$$g \circ f : (x, y)^T \to (\sin(2x), \cos(2x))^T.$$

The process can be repeated by iterative composition with g. With two shallow neural networks it is possible to well approximates both f and g, and their composition is a neural network with two hidden layers. This means that by adding a single layer to a pretrained neural network it is possible to double its frequency content. Therefore, the incremental frequency approach to solve the Helmholtz problem for frequency k consists in using as initialization the neural network solution of the same problem for frequency $k/2$, with an additional final layer of neurons. This process can be applied iteratively and the initial problem can be solve using a shallow neural network. The first shallow network can be initialized using the physics-based approach of Section 6.3.2, while the additional layers of the following networks can be initialized using the adaptive refinement approach of Section 6.3.3.

6.3.5. Generative adversarial networks

This approach is based on the idea of learning both the trial and the test space simultaneously. This allows to select effectively few test functions, hence simplifying the optimization problem, improving the accuracy and reducing the computational cost. To accomplish this we employ the Orthogonal Greedy Algorithm [135], by repeating the

following steps until convergence is reached:

1. Given a test space, we train a neural network for the trial space, that minimizes the loss function to approximate the solution. For a fixed value of the basis functions, i.e. of the non-linear network parameters, it is sufficient to apply the least squares approach of Section 6.3.1.

2. Given the candidate solution of step 1, we train a neural network for the test space, that maximizes the loss function to find the direction along which the error is largest. This direction should be normalized and it is supposed to be orthogonal with respect to the space of basis functions currently employed.

3. The trained test networks is used both to enrich the space of test functions and as additional basis function to initialize the next trial network.

This process allows to incrementally the build a space of basis functions which is optimal for the considered problem. The concept of neural networks competing to minimize and maximize the loss is analogous to the one of Generative Adversarial Networks (GANs) [64, 90, 145], where a generative network tries to learn the data distribution in order to generate new samples, while a discriminative network tries to distinguish the candidates produced by the generator from the true data.

6.4. Numerical experiments

As a first exploratory test case, in the following numerical experiments we considered the approximation problem

$$\min_{u_N} \frac{1}{K} \sum_{k=1}^{K} \left| \int_\Omega (u_N(x) - f(x)) v_k(x) \right|^2 . \tag{6.5}$$

6.4.1. Adaptive mesh refinement

We consider the following solution of (6.5)

$$f(x) = \sqrt{x}, \quad x \in [0, 1]. \tag{6.6}$$

We want employ a shallow neural network and Lagrangian test functions, together with the following adaptive refinement strategy:

1. Train a neural network to minimize (6.5).

2. Compute the loss element-wise for the test space mesh.
3. Refine 20% of the element with the highest error, therefore increasing the number of test functions.
4. Add one neuron to the trial neural network for each new element.
5. Repeat steps 1-4 until convergence.

We employed an initial number of $N = 4$ neurons and $K = 10$ hat functions. After 4 iteration of the adaptive procedure the relative L^2 error is 2.0204e-02 and results are shown in Figure 6.2. As we can see, mesh nodes in test space are clustering close to zero, where $f(x)$ is singular. The same experiment has been repeated in the deep case, i.e. by adding one hidden layer each time in the refinement process, keeping the width of the network constant. Results are completely analogous.

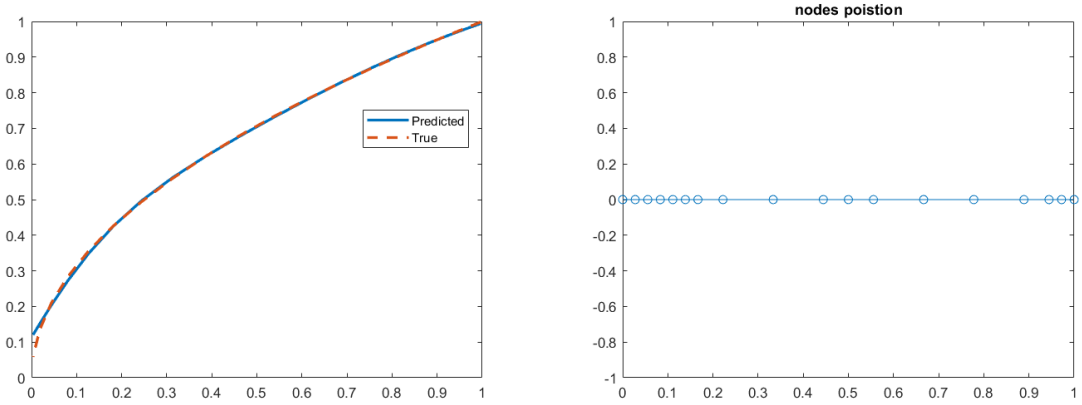

Figure 6.2: Left: shallow neural network solution of (6.5) and (6.6), after 4 iterations of the adaptive refinement procedure. Right: mesh nodes of the refined test space.

6.4.2. Physics-based initialization and hybrid training

We consider the class of solutions of (6.5)

$$f(x) = x^2 \sin(a \cos(x)^2), \quad x \in [0, \pi], \tag{6.7}$$

for different values of the leading frequency $a \in \mathbb{R}^+$, with u_N a shallow neural network and $\{v_k\}_{k=1}^K$ equally spaced Lagrangian basis functions. We employ both a physics-based initialization and hybrid training, described in Sections 6.3.2 and 6.3.1. As we can see in Figure 6.3, employing a physics based initialization already yields a very good approximation of the solution, with a relive L^2 error of order 10^{-1}. Then, by employing a hybrid training strategy it is possible to improve the accuracy of one order of magnitude.

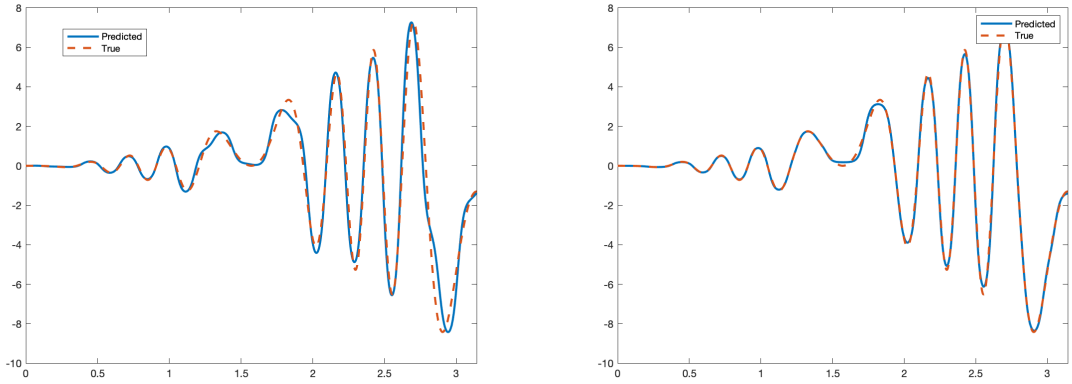

Figure 6.3: Solution of (6.5) and (6.7) for frequency $a = 25$, $N = 25$ neurons, $K = 3a = 75$ basis functions. Left: solution after the physics-based initialization, with relative L^2 error 2.0444e-01. Right: solution after 1000 iterations of hybrid training, with relative L^2 error 4.4297e-02.

neurons\frequency	25	50	100
10	6.9004e-01	8.0182e-01	8.0304e-01
50	1.7119e-02	4.9621e-02	6.3034e-01
100	6.7817e-04	2.0629e-02	5.2678e-02
150	7.2992e-04	1.0146e-03	2.4594e-02
200	7.9429e-04	5.6830e-04	1.5395e-02

Table 6.1: Relative L^2 errors for the solution of (6.5) and (6.7), for different values of the frequency a and number of neurons N. In all cases the number of basis functions is $K = 3a$.

This approach seems robust even for high frequencies, e.g. $a = 300$ shown in Figure 6.4. This is further confirmed from results in Table 6.1. As expected, as the frequency grows more neurons are required to reach the same accuracy. At the same time however, using more neurons means solving a more complex optimization problem, which is why for frequency $a = 25$ the error even increases a bit when using 200 neurons instead of 150.

6.4.3. Generative adversarial networks with incremental frequency

We consider the class of solutions of (6.5)

$$f(x) = \sin(ax), \quad x \in [0, \pi], \tag{6.8}$$

for different values of frequency $a \in \mathbb{R}^+$, with equally spaced Lagrangian basis functions. We employed a GANs approach to incrementally build the optimal space of basis functions. For frequency $a = 1$, the convergence plots are repented in Figure 6.5, in agreement with the theoretical estimates of classical FEMs, while the constructed space of basis functions is represented in Figure 6.6.

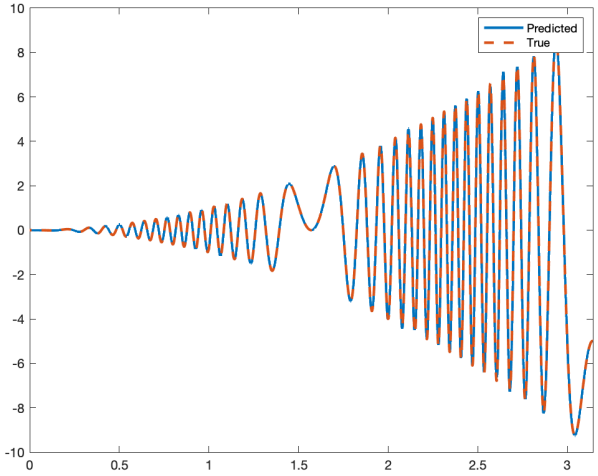

Figure 6.4: Solution of problem (6.5) and (6.7) for frequency $a = 100$, $N = 200$ neurons, $K = 3a = 300$ basis functions. Physics-based initialization and hybrid training have been employed. Relative L^2 error: 1.5395e-02.

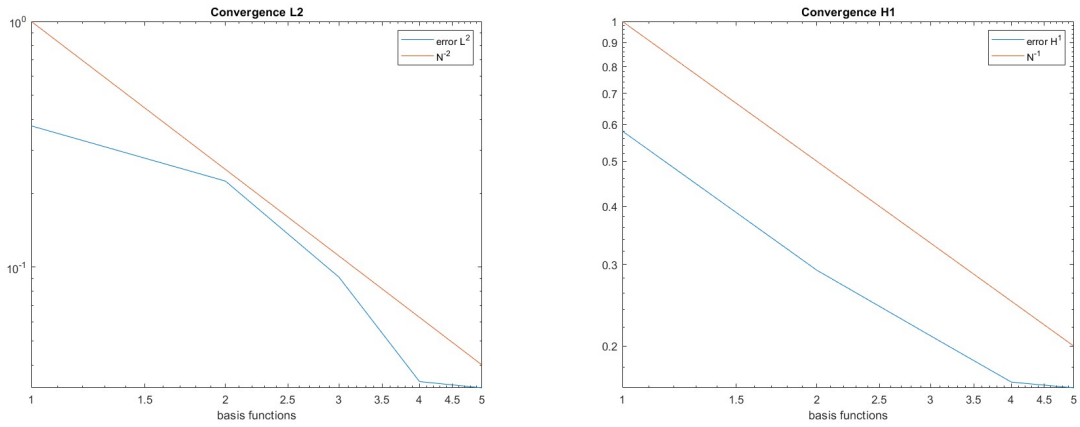

Figure 6.5: Relative L^2 error (left) and H^1 error (right) vs the number of basis functions, using the GANs approach to solve (6.5) and (6.8) for frequency $a = 1$.

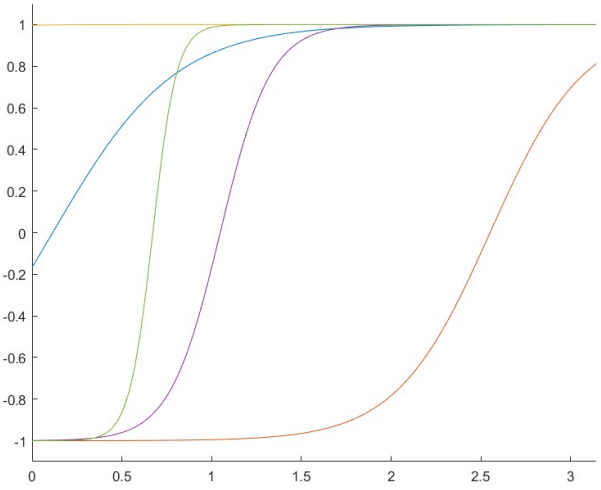

Figure 6.6: Test space constructed using the GANs approach to solve (6.5) and (6.8) for frequency $a = 1$.

basis functions\frequency	1	2	4
1	3.76e-01	4.72e-01	1.06e+00
2	2.24e-01	4.92e-01	1.14e+00
3	9.11e-02	1.85e-01	1.11e+00
4	3.42e-02	8.93e-02	1.08e+00
5	3.23e-02	8.24e-02	1.07e+00

Table 6.2: Relative L^2 error as function of the number of basis functions and the frequency, using GANs with an incremental frequency approach to solve (6.5) and (6.8).

The GANs strategy has been combined with an incremental frequency approach, resolving in order frequencies 1, 2 and 4. As shown in Tables 6.2 and 6.3, while convergence is observed moving from frequency 1 to frequency 2, the method fails moving to frequency 4. Indeed, when applying this approach, tuning of training paramters must be done with care and it is still subject of ongoing research.

6.5. Concluding remarks

The VPINNs strategy proposed to solve the Helmholtz impedance problem for high frequency is appealing from a theoretical point of view, in terms of approximation power of the numerical solution. Training still remains the major issue for the applicability of this methods. The strategies presented to simplify the optimization problem draw interesting parallelism between neural networks and finite elements, from which both worlds could greatly benefit in terms of computational efficiency and interpretability. For example, the

basis functions\frequency	1	2	4
1	5.81e-01	7.30e-01	9.06e-01
2	2.91e-01	5.78e-01	7.80e-01
3	2.11e-01	3.27e-01	7.76e-01
4	1.67e-01	2.58e-01	7.81e-01
5	1.63e-01	2.43e-01	8.11e-01

Table 6.3: Relative H^1 error as function of the number of basis functions and the frequency, using GANs with an incremental frequency approach to solve (6.5) and (6.8).

concept of adaptive mesh refinement from FEMs has been applied to VPINNs in order to improve convergence. Vice-versa, the concept of GANs has been an applied to build a problem-informed FEM basis, in order to reduce the computational cost. All of the considered training strategies seem promising tools to solve the VPINNs formulation of the Helmholtz impedance problem, to be carefully combined and studied in terms of robustness. This work is still in a preliminary phase, and many numerical experiments still need to be performed in order to actually access the feasibility of the approach.

7 | Conclusions and future developments

In this thesis we presented a ML-based framework to efficiently handle the open problems of mesh refinement and agglomeration within polytopal finite element discretizations of differential problems. It is based on the concept of learning the geometrical information contained in the shape of mesh elements in an automatic and cost effective manner, in order to tailor the approach a wide range of possible situations which would not be possible using classical strategies.

The novelty of the proposed methods consist in boosting and generalising classical methods for mesh refinement and agglomeration using Deep Neural Network architectures and the k-means clustering algorithm. Advantages include computational efficiency, handling a wider range of data variability, independence from the differential model and the numerical method at hand, flexibility in exploiting additional geometrical information, exploitation of predefined classical algorithms, memory saving and higher grid quality.

At first, we developed in two dimensions a new paradigm based on CNNs to enhance both existing refinement criteria and new refinement procedures, withing polygonal finite element discretizations of Partial Differential Equations. We introduced two CNN-enhanced refinement strategies, respectively based on boosting the classical Midpoint strategy and on generalizing the refinement strategies for regular polygons to any element shape. These strategies exploit a CNN to suitably classify the "shape" of a polygon in order to later apply an ad-hoc refinement strategy, by choosing among the available ones. This approach has the advantage to be made of interchangeable pieces: any algorithm can be employed to classify mesh elements, as well as any refinement strategy can be employed to refine a polygon with a given label. As a result, the quality of the grids consistently and significantly improves, together with the accuracy of polygonal finite element methods (DG and VEM) employed for the discretization. Moreover, this classification step has a very limited computational cost when using a pre-trained CNN. The latter can be made offline once and for all, independently of the differential model under consideration.

We developed the non-trivial extension to three dimensions of the above ML-based paradigm for polygonal grid refinement, to exploit it within polyhedral finite element discretizations of Partial Differential Equations. By doing so we inherit the benefits of the two dimensional approach. In particular, the computational efficiency of CNNs plays even a more important role in three dimensions: it allows to exploit strategies explicitly based on the shape of the polyhedron, which would not be possible otherwise unless explicit and costly geometrical checks are performed. Due to the very high variability of the shapes of polyhedra in three dimensions, an approach based solely on the supervised learning using CNNs does not seem enough to effectively tackle the problem, because of the impossibility to generate an exhaustive database of "shapes" for the training. This calls for the need of unsupervised learning approaches, that do not require prior labelled data, at the cost of performing more computations online. In particular, the k-means algorithm is used to learn a clustered representation of the element that is used to perform the partition. It produces shape-regular elements and it is robust when applied on unstructured grids. The flexibility of CNNs allows to combine them with other strategies, including the k-means algorithm and making the overall approach robust. As a result, the initial structure and quality of the grid is preserved, significantly lowering the computational cost and the number of vertices, edges and faces required to refine the mesh. These ML-enhanced refinement strategies are particularly beneficial for the VEM and for the PolyDG method.

Then, we developed a GNN-based approach to tackle the problem of polygonal grids agglomeration. This is achieved by exploiting the graph representation of the connectivity structure of mesh elements. GNNs have the advantage to process naturally and simultaneously both the graph structure of mesh and the geometrical information, such the elements areas or their barycentric coordinates, contrary to METIS [106] or k-means. Performance in terms of quality metrics is enhanced, with GNNs featuring a lower computational cost online. Such models also show a high degree of generalization when applied to more complex geometries and capability of preserving the grids quality. The effectiveness of these strategies is measured also when applied to multigrid solvers in a PolyDG framework.

Finally, we introduced the use of VPINNs to solve the one-dimensional Helmholtz impedance problem. Such approach is appealing for high frequency from a theoretical point of view, thanks to the approximation power of neural networks. We presented different training strategies to simplify the optimization problem, exploiting parallelism between neural networks and finite elements, and performed some preliminary numerical experiments. This work is still in a preliminary phase, and many numerical experiments still need to be performed in order to actually access the feasibility of the approach.

The results of the thesis are original and are contained in [4, 6, 9, 97].

Future developments that arise from the work carried out in this thesis are the following.

- Concerning mesh refinement, it would be of great interest to tackle also the anisotropic case. Indeed, as noticed in [78], having skewed meshes can sometimes be beneficial to polytopal methods with hybrid unknowns, if the skewness is compatible with the diffusion anisotropy. We believe that our method may be extended so as to generate distorted elements by including the anisotropy directions into the shape recognition phase. This extension would certainly require to process information coming from the physical problem at hand, therefore loosing a bit in generality of the approach. However, this additional information can be naturally and automatically processed by neural networks, without having to perform a consistent change in the refinement framework.

- Concerning mesh agglomeration, further improvements certainly include fostering the generalization capabilities of the GNNs, by generating a larger database of grids or by considering different geometrical features, such as quality metrics or error estimators. Another possibility is to reframe the approach in a reinforcement learning framework, as proposed by [86]. The extension to three dimensional agglomeration strategies should certainly be explored. It is worth noticing that all of the proposed agglomeration strategies can be applied to the three dimensional case with little or no modification, as they mainly rely on the connectivity graph structure of the mesh, which is a dimensionless entity, or on geometrical quantities with natural extensions such the area/volume of polytopes or their barycentric coordinates. The described approach could also be employed within domain decomposition techniques, and MG schemes as preconditioner.

- From a point of view of the applications, the effect of using a posteriori error estimators to drive the refinement and agglomeration processes should be investigated. Indeed, in the adaptive refinement case we did not employ any a posteriori estimator of the error but the actual error of the numerical solution, since we wanted to investigate the direct effect of the proposed refinement strategies. This simplifying assumption cannot hold in practice where the true solution is unknown.

- Other discretization frameworks other than the PolyDG and VEM could also be automatically considered, such as the Hybrid High-Order method, the mimetic finite difference method, or the hybridizable discontinuous Galerkin method.

- This work explored the capabilities of a ML based framework for handling polytopal

grids mainly considering idealized test cases. More complex and real applications should certainly be considered in order to effectively test the capabilities of this approach, for example geophysical numerical simulations, including fluid-structure interaction with complex and moving geometries. Preliminary results of agglomeration of the brain obtained from medical images are encouraging.

- Finally, the work on VPINNs to solve the Helmholtz impedance problem is still in a preliminary phase. The effectiveness and the robustness of the proposed training strategies still needs to be studied and validated through extensive numerical experiments. Moreover, an efficient implementation of this approach is required, to practically explore its computational limitations for very high frequencies.

Bibliography

[1] B. Ahmad, A. Alsaedi, F. Brezzi, L. D. Marini, and A. Russo. Equivalent projectors for virtual element methods. *Computers & Mathematics with Applications*, 66(3): 376–391, 2013.

[2] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[3] T. Alkhalifah, C. Song, U. bin Waheed, and Q. Hao. Wavefield solutions from machine learned functions constrained by the helmholtz equation. *Artificial Intelligence in Geosciences*, 2:11–19, 2021.

[4] P. Antonietti and E. Manuzzi. Refinement of polygonal grids using convolutional neural networks with applications to polygonal discontinuous galerkin and virtual element methods. *Journal of Computational Physics*, 452:110900, 2022. ISSN 0021-9991.

[5] P. Antonietti and G. Pennesi. V-cycle multigrid algorithms for discontinuous galerkinmethods on non-nested polytopic meshes. *Journal of Scientific Computing*, 78: 625–652, 2019.

[6] P. Antonietti, F. Dassi, and E. Manuzzi. Machine learning based refinement strategies for polyhedral grids with applications to virtual element and polyhedral discontinuous galerkin methods. *Journal of Computational Physics*, page 111531, 2022.

[7] P. F. Antonietti and B. Ayuso. Schwarz domain decomposition preconditioners for discontinuous galerkin approximations of elliptic problems: non-overlapping case. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(1):21–54, 2007.

[8] P. F. Antonietti and G. Pennesi. V-cycle multigrid algorithms for discontinuous galerkin methods on non-nested polytopic meshes. *Journal of Scientific Computing*, 78(1):625–652, 2019.

[9] P. F. Antonietti, N. Farenga, E. Manuzzi, G. Martinelli, and L. Saverio. Agglomer-

ation of polygonal grids using graph neural networks with applications to multigrid solvers. *In preparation*.

[10] P. F. Antonietti, S. Giani, and P. Houston. hp-version composite discontinuous galerkin methods for elliptic problems on complicated domains. *SIAM Journal on Scientific Computing*, 35(3):A1417–A1439, 2013.

[11] P. F. Antonietti, S. Giani, and P. Houston. Domain decomposition preconditioners for discontinuous galerkin methods for elliptic problems on complicated domains. *Journal of Scientific Computing*, 60(1):203–227, 2014.

[12] P. F. Antonietti, M. Sarti, and M. Verani. Multigrid algorithms for hp-discontinuous galerkin discretizations of elliptic problems. *SIAM Journal on Numerical Analysis*, 53(1):598–618, 2015.

[13] P. F. Antonietti, A. Cangiani, J. Collis, Z. Dong, E. H. Georgoulis, S. Giani, and P. Houston. Review of discontinuous galerkin finite element methods for partial differential equations on complicated domains. In *Building bridges: connections and challenges in modern approaches to numerical partial differential equations*, pages 281–310. Springer, 2016.

[14] P. F. Antonietti, L. B. Da Veiga, S. Scacchi, and M. Verani. A cˆ1 virtual element method for the cahn–hilliard equation with polygonal meshes. *SIAM Journal on Numerical Analysis*, 54(1):34–56, 2016.

[15] P. F. Antonietti, P. Houston, X. Hu, and M. Sarti, M.and Verani. Multigrid algorithms for hp-version interior penalty discontinuous galerkin methods on polygonal and polyhedral meshes. *Calcolo*, 54(4):1169–1198, 2017.

[16] P. F. Antonietti, P. Houston, G. Pennesi, and E. Süli. An agglomeration-based massively parallel non-overlapping additive schwarz preconditioner for high-order discontinuous galerkin methods on polytopic grids. *Mathematics of Computation*, 2020.

[17] P. F. Antonietti, M. Caldana, and L. Dede. Accelerating algebraic multigrid methods via artificial neural networks. *arXiv preprint arXiv:2111.01629*, 2021.

[18] P. F. Antonietti, C. Facciolà, P. Houston, I. Mazzieri, G. Pennesi, and M. Verani. High–order discontinuous galerkin methods on polyhedral grids for geophysical applications: seismic wave propagation and fractured reservoir simulations. *Polyhedral Methods in Geosciences*, pages 159–225, 2021.

[19] P. F. Antonietti, L. Beirão da Veiga, and G. Manzini. *The Virtual Element Method and its Applications*. Springer International Publishing, 2022.

[20] P. F. Antonietti, L. Mascotto, M. Verani, and S. Zonca. Stability analysis of polytopic discontinuous galerkin approximations of the stokes problem with applications to fluid–structure interaction problems. *Journal of Scientific Computing*, 90(1):1–31, 2022.

[21] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM journal on numerical analysis*, 19(4):742–760, 1982.

[22] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[23] M. Attene, S. Biasotti, S. Bertoluzza, D. Cabiddu, M. Livesu, G. Patanè, M. Pennacchio, D. Prada, and M. Spagnuolo. Benchmark of polygon quality metrics for polytopal element methods. *arXiv preprint arXiv:1906.01627*, 2019.

[24] I. Babuška and M. Suri. The *hp* version of the finite element method with quasiuniform meshes. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 21(2):199–238, 1987.

[25] L. Babuška and M. Suri. The optimal convergence rate of the p-version of the finite element method. *SIAM Journal on Numerical Analysis*, 24(4):750–776, 1987.

[26] G. A. Baker. Finite element methods for elliptic equations using nonconforming elements. *Mathematics of Computation*, 31(137):45–59, 1977.

[27] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[28] F. Bassi, L. Botti, A. Colombo, D. A. Di Pietro, and P. Tesini. On the flexibility of agglomeration based physical space discontinuous galerkin discretizations. *Journal of Computational Physics*, 231(1):45–65, 2012.

[29] F. Bassi, L. Botti, A. Colombo, and S. Rebay. Agglomeration based discontinuous galerkin discretization of the euler and navier–stokes equations. *Computers & fluids*, 61:77–85, 2012.

[30] L. Beirão da Veiga, F. Brezzi, A. Cangiani, L. D. Manzini, GianM.and Marini, and A. Russo. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences*, 23(01):199–214, 2013.

[31] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. The hitchhiker's guide to the virtual element method. *Mathematical models and methods in applied sciences*, 24(08):1541–1573, 2014.

[32] L. Beirao da Veiga, K. Lipnikov, and G. Manzini. *The mimetic finite difference method for elliptic problems*, volume 11. Springer, 2014.

[33] L. Beirão da Veiga, F. Brezzi, L. Marini, and A. Russo. Virtual element method for general second-order elliptic problems on polygonal meshes. *Mathematical Models and Methods in Applied Sciences*, 26(04):729–750, 2016.

[34] L. Beirao da Veiga, F. Brezzi, L. D. Marini, and A. Russo. Mixed virtual element methods for general second order elliptic problems on polygonal meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 50(3):727–747, 2016.

[35] L. Beirão da Veiga, C. Lovadina, and A. Russo. Stability analysis for the virtual element method. *Mathematical Models and Methods in Applied Sciences*, 27(13): 2557–2594, 2017.

[36] L. Beirao da Veiga, N. Bellomo, F. Brezzi, and L. Marini. Recent results and perspectives for virtual element methods. *Mathematical Models and Methods in Applied Sciences*, pages 1–6, 2021.

[37] S. Berrone, A. Borio, and A. D'Auria. Refinement strategies for polygonal meshes applied to adaptive vem discretization. *Finite Elements in Analysis and Design*, 186:103502, 2021.

[38] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[39] D. Boffi, F. Brezzi, M. Fortin, et al. *Mixed finite element methods and applications*, volume 44. Springer, 2013.

[40] F. Bonizzoni and F. Nobile. Regularity and sparse approximation of the recursive first moment equations for the lognormal darcy problem. *Computers & Mathematics with Applications*, 80(12):2925–2947, 2020.

[41] F. Bonizzoni, F. Nobile, and D. Kressner. Tensor train approximation of moment equations for elliptic equations with lognormal coefficient. *Computer Methods in Applied Mechanics and Engineering*, 308:349–376, 2016.

[42] J. H. Bramble, J. E. Pasciak, and J. Xu. The analysis of multigrid algorithms with nonnested spaces or noninherited quadratic forms. *Mathematics of Computation*, 56(193):1–34, 1991.

[43] S. C. Brenner and L. Owens. A w-cycle algorithm for a weakly over-penalized interior penalty method. *Computer methods in applied mechanics and engineering*, 196(37-40):3823–3832, 2007.

[44] S. C. Brenner and L.-Y. Sung. Virtual element methods on meshes with small edges or faces. *Mathematical Models and Methods in Applied Sciences*, 28(07):1291–1336, 2018.

[45] S. C. Brenner, J. Cui, T. Gudi, and L.-Y. Sung. Multigrid algorithms for symmetric discontinuous galerkin methods on graded meshes. *Numerische Mathematik*, 119 (1):21–47, 2011.

[46] F. Brezzi and L. D. Marini. Virtual element methods for plate bending problems. *Computer Methods in Applied Mechanics and Engineering*, 253:455–462, 2013.

[47] F. Brezzi, K. Lipnikov, and M. Shashkov. Convergence of the mimetic finite difference method for diffusion problems on polyhedral meshes. *SIAM Journal on Numerical Analysis*, 43(5):1872–1896, 2005.

[48] F. Brezzi, K. Lipnikov, and V. Simoncini. A family of mimetic finite difference methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 15(10):1533–1551, 2005.

[49] Z. Cai, J. Chen, and M. Liu. Self-adaptive deep neural network: numerical approximation to functions and pdes. *Journal of Computational Physics*, 455:111021, 2022.

[50] A. Cangiani, E. H. Georgoulis, and P. Houston. hp-version discontinuous galerkin methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 24(10):2009–2041, 2014.

[51] A. Cangiani, Z. Dong, E. H. Georgoulis, and P. Houston. hp-version discontinuous galerkin methods for advection-diffusion-reaction problems on polytopic meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 50(3):699–725, 2016.

[52] A. Cangiani, Z. Dong, and E. H. Georgoulis. hp-version space-time discontinuous galerkin methods for parabolic problems on prismatic meshes. *SIAM Journal on Scientific Computing*, 39(4):A1251–A1279, 2017.

[53] A. Cangiani, Z. Dong, E. H. Georgoulis, and P. Houston. *hp-Version discontinuous Galerkin methods on polygonal and polyhedral meshes*. Springer, 2017.

[54] A. Cangiani, E. H. Georgoulis, T. Pryer, and O. J. Sutton. A posteriori error

estimates for the virtual element method. *Numerische mathematik*, 137(4):857–893, 2017.

[55] T. F. Chan, S. Go, and L. Zikatanov. Multilevel elliptic solvers on unstructured grids. In *Computational Fluid Dynamics Review 1998: (In 2 Volumes)*, pages 488–511. World Scientific, 1998.

[56] T. F. Chan, J. Xu, and L. Zikatanov. An agglomeration multigrid method for unstructured grids. *Contemporary Mathematics*, 218:67–81, 1998.

[57] A. Chernov. Optimal convergence estimates for the trace of the polynomial l^2-projection operator on a simplex. *Mathematics of Computation*, 81(278):765–787, 2012.

[58] P. Ciarlet. The finite element method for elliptic problems, vol. 4 of stud. *Math. Appl. North-Holland Publishing Co., Amsterdam*, 1978.

[59] B. Cockburn, B. Dong, and J. Guzmán. A superconvergent ldg-hybridizable galerkin method for second-order elliptic problems. *Mathematics of Computation*, 77(264): 1887–1916, 2008.

[60] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[61] B. Cockburn, J. Guzmán, and H. Wang. Superconvergent discontinuous galerkin methods for second-order elliptic problems. *Mathematics of Computation*, 78(265): 1–24, 2009.

[62] B. Cockburn, J. Gopalakrishnan, and F.-J. Sayas. A projection-based error analysis of hdg methods. *Mathematics of Computation*, 79(271):1351–1367, 2010.

[63] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. *Discontinuous Galerkin methods: theory, computation and applications*, volume 11. Springer Science & Business Media, 2012.

[64] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[65] E. C. Cyr, M. A. Gulian, R. G. Patel, M. Perego, and N. A. Trask. Robust training and initialization of deep neural networks: An adaptive basis viewpoint. In *Mathematical and Scientific Machine Learning*, pages 512–536. PMLR, 2020.

[66] L. B. Da Veiga, F. Dassi, and A. Russo. High-order virtual element method on polyhedral meshes. *Computers & Mathematics with Applications*, 74(5):1110–1122, 2017.

[67] L. B. da Veiga, C. Lovadina, and G. Vacca. Divergence free virtual elements for the stokes problem on polygonal meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 51(2):509–535, 2017.

[68] F. Dassi and L. Mascotto. Exploring high-order three dimensional virtual elements: bases and stabilizations. *Computers & Mathematics with Applications*, 75(9):3379–3401, 2018.

[69] T. De Ryck, S. Lanthaler, and S. Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021.

[70] D. A. Di Pietro and J. Droniou. *The Hybrid High-Order method for polytopal meshes*, volume 19. Springer, 2019.

[71] D. A. Di Pietro and A. Ern. A hybrid high-order locking-free method for linear elasticity on general meshes. *Computer Methods in Applied Mechanics and Engineering*, 283:1–21, 2015.

[72] D. A. Di Pietro and A. Ern. Hybrid high-order methods for variable-diffusion problems on general meshes. *Comptes Rendus Mathématique*, 353(1):31–34, 2015.

[73] D. A. Di Pietro, A. Ern, and S. Lemaire. An arbitrary-order and compact-stencil discretization of diffusion on general meshes based on local reconstruction operators. *Computational Methods in Applied Mathematics*, 14(4):461–472, 2014.

[74] D. A. Di Pietro, A. Ern, and S. Lemaire. A review of hybrid high-order methods: formulations, computational aspects, comparison with other methods. In *Building bridges: connections and challenges in modern approaches to numerical partial differential equations*, pages 205–236. Springer, 2016.

[75] D. A. Di Pietro, L. Formaggia, R. Masson, et al. Polyhedral methods in geosciences. 2021.

[76] G. C. Diwan, A. Moiola, and E. A. Spence. Can coercive formulations lead to fast and accurate solution of the helmholtz equation? *Journal of Computational and Applied Mathematics*, 352:110–131, 2019.

[77] V. A. Dobrev, R. D. Lazarov, P. S. Vassilevski, and L. T. Zikatanov. Two-level

preconditioning of discontinuous galerkin approximations of second-order elliptic equations. *Numerical Linear Algebra with Applications*, 13(9):753–770, 2006.

[78] J. Droniou. Interplay between diffusion anisotropy and mesh skewness in hybrid high-order schemes. In *International Conference on Finite Volumes for Complex Applications*, pages 3–23. Springer, 2020.

[79] J. Droniou and L. Yemm. Robust hybrid high-order method on polytopal meshes with small faces. *arXiv preprint arXiv:2102.06414*, 2021.

[80] I. L. Dryden and K. V. Mardia. *Statistical shape analysis: with applications in R*, volume 995. John Wiley & Sons, 2016.

[81] M. Dryja, J. Galvis, and M. Sarkis. Bddc methods for discontinuous galerkin discretization of elliptic problems. *Journal of Complexity*, 23(4-6):715–739, 2007.

[82] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[83] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

[84] X. Feng and O. A. Karakashian. Two-level additive schwarz methods for a discontinuous galerkin approximation of second order elliptic problems. *SIAM Journal on Numerical Analysis*, 39(4):1343–1365, 2001.

[85] A. Gatti, Z. Hu, T. Smidt, E. G. Ng, and P. Ghysels. Deep learning and spectral embedding for graph partitioning, 2021.

[86] A. Gatti, Z. Hu, T. Smidt, E. G. Ng, and P. Ghysels. Graph partitioning and sparse matrix ordering using reinforcement learning and graph neural networks, 2021.

[87] E. H. Georgoulis and A. Lasis. A note on the design of hp-version interior penalty discontinuous galerkin finite element methods for degenerate problems. *IMA journal of numerical analysis*, 26(2):381–390, 2006.

[88] S. Giani and P. Houston. Domain decomposition preconditioners for discontinuous galerkin discretizations of compressible fluid flows. *Numerical Mathematics: Theory, Methods and Applications*, 7(2):123–148, 2014.

[89] J. R. Gilbert, G. L. Miller, and S.-H. Teng. Geometric mesh partitioning: Implementation and experiments. *SIAM Journal on Scientific Computing*, 19(6):2091–2110, 1998.

[90] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[91] J. Gopalakrishnan and G. Kanschat. A multilevel discontinuous galerkin method. *Numerische Mathematik*, 95(3):527–550, 2003.

[92] I. Gühring and M. Raslan. Approximation rates for neural networks with encodable weights in smoothness spaces. *Neural Networks*, 134:107–130, 2021.

[93] I. Gühring, G. Kutyniok, and P. Petersen. Error bounds for approximations with deep relu neural networks in w s, p norms. *Analysis and Applications*, 18(05): 803–859, 2020.

[94] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs, 2017.

[95] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[96] J. He, L. Li, J. Xu, and C. Zheng. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.

[97] F. Henrìquez, J. S. Hesthaven, and E. Manuzzi. Variational physics-informed neural networks for the solution of the helmholtz impedance problem. *In preparation.*

[98] J. S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[99] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[100] T. Y. Hoshina, I. F. Menezes, and A. Pereira. A simple adaptive mesh refinement scheme for topology optimization using polygonal meshes. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 40(7):348, 2018.

[101] P. Houston and E. Süli. Stabilised hp-finite element approximation of partial differential equations with nonnegative characteristic form. *Computing*, 66(2):99–119, 2001.

[102] J. Hyman, M. Shashkov, and S. Steinberg. The numerical solution of diffusion prob-

lems in strongly heterogeneous non-isotropic materials. *Journal of Computational Physics*, 132(1):130–148, 1997.

[103] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[104] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[105] J. Kangas, T. Kohonen, and J. Laaksonen. Variants of self-organizing maps. *IEEE transactions on neural networks*, 1(1):93–99, 1990.

[106] G. Karypis and V. Kumar. Kumar, v.: A fast and high quality multilevel scheme for partitioning irregular graphs. siam journal on scientific computing 20(1), 359-392. *Siam Journal on Scientific Computing*, 20, 01 1999. doi: 10.1137/S1064827595287997.

[107] G. Karypis, K. Schloegel, and V. Kumar. Parmetis. *Parallel graph partitioning and sparse matrix ordering library. Version*, 2, 2003.

[108] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

[109] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[110] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[111] M.-J. Lai and G. Slavov. On recursive refinement of convex polygons. *Computer Aided Geometric Design*, 45:83–90, 2016.

[112] J. Lane, B. Magedson, and M. Rarick. An efficient point in polyhedron algorithm. *Computer vision, graphics, and image processing*, 26(1):118–125, 1984.

[113] N. Le Hoang, T. K. Dang, et al. A farthest first traversal based sampling algorithm for k-clustering. In *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–6. IEEE, 2020.

[114] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[115] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[116] X. Liu, X. Zhang, W. Peng, W. Zhou, and W. Yao. A novel meta-learning initialization method for physics-informed neural networks. *Neural Computing and Applications*, pages 1–24, 2022.

[117] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics (ToG)*, 28(4):1–17, 2009.

[118] J. Macqueen. Some methods for classification and analysis of multivariate observations, 1967.

[119] A. Moiola and E. A. Spence. Is the helmholtz equation really sign-indefinite? *Siam Review*, 56(2):274–312, 2014.

[120] D. Mora, G. Rivera, and R. Rodríguez. A virtual element method for the steklov eigenvalue problem. *Mathematical Models and Methods in Applied Sciences*, 25(08): 1421–1445, 2015.

[121] L. Mu, X. Wang, and Y. Wang. Shape regularity conditions for polygonal/polyhedral meshes, exemplified in a discontinuous galerkin discretization. *Numerical Methods for Partial Differential Equations*, 31(1):308–325, 2015.

[122] R. Muñoz-Sola. Polynomial liftings on a tetrahedron and applications to the hp version of the finite element method in three dimensions. *SIAM journal on numerical analysis*, 34(1):282–314, 1997.

[123] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[124] Y. Pan and P.-O. Persson. Agglomeration-based geometric multigrid solvers for compact discontinuous galerkin discretizations on unstructured meshes. *Journal of Computational Physics*, 449:110775, 2022.

[125] D. Perekrestenko, P. Grohs, D. Elbrächter, and H. Bölcskei. The universal approximation power of finite-width deep relu networks. *arXiv preprint arXiv:1806.01528*, 2018.

[126] P. C. Petersen. Neural network theory. *University of Vienna*, 2020.

[127] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357: 125–141, 2018.

[128] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[129] D. Ray and J. S. Hesthaven. An artificial neural network as a troubled-cell indicator. *Journal of Computational Physics*, 367:166–191, 2018.

[130] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[131] F. Regazzoni, L. Dedè, and A. Quarteroni. Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics*, 397:108852, 2019.

[132] F. Regazzoni, L. Dedè, and A. Quarteroni. Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics. *Computer Methods in Applied Mechanics and Engineering*, 370:113268, 2020.

[133] F. Regazzoni, M. Salvador, L. Dedè, and A. Quarteroni. A machine learning method for real-time numerical simulations of cardiac electromechanics. *arXiv preprint arXiv:2110.13212*, 2021.

[134] H. Ritter, T. Martinetz, K. Schulten, et al. *Neural computation and self-organizing maps: an introduction*. Addison-Wesley Reading, MA, 1992.

[135] J. W. Siegel and J. Xu. Improved convergence rates for the orthogonal greedy algorithm. *arXiv preprint arXiv:2106.15000*, 2021.

[136] T. Sorgente, S. Biasotti, G. Manzini, and M. Spagnuolo. The role of mesh quality and mesh quality indicators in the virtual element method. *arXiv preprint arXiv:2102.04138*, 2021.

[137] E. M. Stein. *Singular integrals and differentiability properties of functions*, volume 2. Princeton university press, 1970.

[138] Q. Sun, E. Klaseboer, B.-C. Khoo, and D. Y. Chan. Boundary regularized integral equation formulation of the helmholtz equation in acoustics. *Royal Society open science*, 2(1):140520, 2015.

[139] C. Talischi, G. H. Paulino, A. Pereira, and I. F. Menezes. Polymesher: a general-purpose mesh generator for polygonal elements written in matlab. *Structural and Multidisciplinary Optimization*, 45(3):309–328, 2012.

[140] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *Advances in neural information processing systems*, 27, 2014.

[141] A. Toselli and O. Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media, 2004.

[142] M. F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM Journal on Numerical Analysis*, 15(1):152–161, 1978.

[143] H. Xu, Z. Duan, Y. Wang, J. Feng, R. Chen, Q. Zhang, and Z. Xu. Graph partitioning and graph neural network based hierarchical graph matching for graph similarity computation. *Neurocomputing*, 439:348–362, 2021.

[144] J. Xu and L. Zikatanov. Algebraic multigrid methods. *Acta Numerica*, 26:591–721, 2017.

[145] L. Yang, D. Zhang, and G. E. Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal on Scientific Computing*, 42(1):A292–A317, 2020.

[146] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

List of Figures

List of Tables

List of Abbreviations

Area-Perimeter Ratio (APR), Ball Ratio (BR), Centrodial Voronoi Tesselation (CVT), Circle Ratio (CR), Convolutional Neural Networks (CNNs), Convolutional Neural Network-enhanced Reference Polygon strategy (CNN-RP), Edge Ratio (ER), Finite Element Methods (FEMs), Generative Adversarial Networks (GANs), Graph Neural Networks (GNNs), Machine Learning (ML), Mid-Point (MP), Minmum Angle (MA), MultiGrid (MG), Normalized Point Distance (NPD), Physics-Informed Neural Networks (PINNs), Polytopal Discontinuous Galerkin (PolyDG), Uniformity Factor (UF), Variational Physics-Informed Neural Networks (VPINNs), Virtual Element Method (VEM).