Geometric Deep Learning for Electrostatics and Magnetostatics Problems

Aishwarya Ramamurthy

Department of Electrical & Computer Engineering

McGill University

Montréal, Québec, Canada

December 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science (M.Sc) in Electrical Engineering.

Abstract

Conventional numerical methods such as the Finite Element Method (FEM) and Finite Difference Method (FDM) have been employed to solve electrostatics and magnetostatics problems involving Partial Differential Equations (PDEs). With the recent boom in deep learning, alternative ways to solve such PDEs subject to boundary value constraints have been proposed and implemented, in contrast to classical numerical solvers. One such entity is a Graph Neural Network (GNN), which precisely depicts the internodal connectivity in a graph structure comprising nodes and edges. At the structural level, a GNN is analogous to the tessellations (here, triangular simplices) of a finite element mesh. This thesis investigates the prediction quality of a GNN model with spectral graph convolution, to learn and approximate the solution for a time-independent, boundary value problem in the domains of electrostatics and magnetostatics. This network is trained in a supervised manner on datasets generated by a FEM solver with diverse shapes and charge/current non-uniformities. For the given datasets and the GNN framework, experimental results demonstrate that introducing mesh augmentations i.e., structural variations in the finite element meshes, enhance GNN predictions over unseen geometries and inhomogeneities, in comparison to commonly used regularization techniques.

Abrégé

Des méthodes numériques conventionnelles telles que la méthode des éléments finis (FEM) et la méthode des différences finies (FDM) ont été utilisées pour résoudre des problèmes d'électrostatique et de magnétostatique impliquant des équations aux dérivées partielles (PDE). Avec le récent boom de l'apprentissage profond, des moyens alternatifs pour résoudre de telles PDE soumises à des contraintes de valeurs limites ont été proposés et mis en œuvre contrairement aux solveurs numériques classiques. L'une de ces entités est un réseau neuronal graphique (GNN), qui décrit précisément la connectivité internodale dans une structure graphique comprenant des nœuds et des arêtes. Au niveau structurel, un GNN est analogue aux pavages (ici, des simplexes triangulaires) d'un maillage d'éléments finis. Cette thèse étudie la qualité de prédiction d'un modèle GNN avec convolution de graphe spectral, pour apprendre et approximer la solution d'un problème de valeur limite indépendant du temps dans les domaines de l'électrostatique et de la magnétostatique. Ce réseau est formé de manière supervisée sur des ensembles de données, générés par un solveur FEM avec diverses formes et non-uniformités charge/courant. Pour les ensembles de données donnés et le cadre GNN, les résultats expérimentaux démontrent que l'introduction d'augmentations de maillage, c'est-à-dire de variations structurelles dans les maillages d'éléments finis, améliore les prédictions GNN sur des géométries et inhomogénéités invisibles, par rapport aux techniques de régularisation couramment utilisées.

Acknowledgements

I would like to express my sincere and heartfelt gratitude to my supervisor, Dr.Dennis Giannacopoulos, for his constant guidance, encouragement and valuable time.

Further, I extend my thanks to Mr.Winfried Ripken (Merantix Momentum, AI Campus, Berlin, Germany), for his timely help and suggestions in the course of my thesis.

This research was enabled in part by support provided by Calcul Québec (calculquebec.ca) and the Digital Research Alliance of Canada (alliancecan.ca).

Last but not least, I would like to thank my friends, family and teachers for their continued support throughout my academic endeavours.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artifical Intelligence
ANN	Artificial Neural Network
BVP	Boundary Value Problem
CNN	Convolutional Neural Network
DL	Deep Learning
EM	Electromagnetics
FDM	Finite Difference Method
FEM	Finite Element Method
GCNN	Graph Convolutional Neural Network
GDL	Geometric Deep Learning
GEN	Graph Elements Network
GNN	Graph Neural Network
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
IN	Interaction Network
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
PDE	Partial Differential Equation
PINN	Physics Informed Neural Network

Chapter 1

Introduction

1.1 Background and Objective

Several scientific and engineering disciplines effectively use Partial Differential Equations (PDEs) to model various physical phenomena. One such PDE chosen for this thesis work is the Poisson's equation. This well-known PDE arises in the areas of electromagnetics, fluid dynamics, gravitational field problems, heat conduction, elasticity, and several other real-world applications [1]. For decades, classical numerical approximation methods such as the Finite Element Method (FEM) have been used to estimate solutions to such PDEs. As an alternative, several Machine and Deep Learning (ML/DL) algorithms have been developed to learn and predict PDE solutions, based on datasets created using standard FEM solvers. However, the prediction capabilities of DL models such as Convolutional Neural Networks (CNNs) are limited by the input geometry, as they can handle only regular structures.

This paved the way to consider Graph Neural Networks (GNNs) [2], with their proven ability to handle irregular structures. Its variants include mesh-based learning [3, 4, 5], neural operators [6, 7], and physics-informed approaches [8] for different applications. Further, GNN models can be trained using augmentation and dropout techniques, to regularize and enhance the approximation traits of neural networks on newer samples [9].

The objectives of this thesis are enlisted below:

- Train and test the GNN model on a dataset comprising meshes (two-dimensional) with various geometries and charge/current inhomogenities, so as to predict the solutions to static Poisson's equations for electrostatics and magnetostatics problems.
- Evaluate the ability of the GNN model with and without mesh augmentation, to predict electric, magnetic fields and potentials over newer mesh samples varying in geometry and current/charge distributions. Here, mesh augmentation refers to inducing geometrical transformations in the training data, such as varying the mesh resolution or trimming small portions from the original (basic) mesh.
- Analyze and report these prediction quality measures, in comparison with conventional regularization (dropout) techniques.

1.2 Contributions

This work investigates the prediction quality of a Graph Neural Network (GNN) model using mesh augmentations, to learn and approximate the solution to Poisson's equation(s) arising in electrostatics and magnetostatics. The GNN model was trained and tested on a supercomputer GPU cluster, over datasets consisting of diverse mesh geometries and charge/current distributions. The estimation quality of the predicted outputs (electric, magnetic potentials and fields) with respect to the ground truth, were measured in terms of Mean Squared Error (MSE), for mesh augmentation and other regularization techniques. This comprehensive model training and testing process was automated in the GPU environment.

The test results were tabulated and visualized to analyze the impact of augmented meshes on the model's capability to approximate the PDE solution, with minimal prediction (test) error. Experimental outcomes presented in this work demonstrate that, mesh augmented training data improves the prediction quality over meshes with increased charge/current

distributions and new structure, for the specified GNN model trained in a supervised manner. Results obtained here are comparable to those reported in [3]. Initially, an inconsistency in the results presented in [10] was identified and reported to its first author. Subsequent inquiries and discussions led to a revised version [3] of [10].

1.3 Thesis Overview

The structure of this thesis is summarized below:

1.3.1 Chapter 2

Chapter 2 introduces the readers to "Geometric Deep Learning" (GDL), prefaced by pre-requisites on the Finite Element Method (FEM), Boundary Value Problem (BVP) and Maxwell's equations. It discusses the limitations of Convolutional Neural Networks (CNNs), thus motivating the choice of a GNN architecture for this work. Further, it presents a survey on the recent advances in data-driven deep learning (GNN based) approaches, proposed in the literature for PDE-governed physics-based problems, including electromagnetic applications.

1.3.2 Chapter 3

Chapter 3 begins with a background on Poisson's equation and its variational formulation for electrostatics and magnetostatics problems. Subsequently, it provides a detailed description about the datasets (with and without mesh augmentation), the underlying GNN framework and experiments. In addition, it presents a comprehensive set of test results, analyzes the model's prediction quality over various mesh geometries and charge/current inhomogeneities, and compares the behaviour with other regularization techniques.

1.3.3 Chapter 4

This concluding chapter summarizes this work with key findings, and suggests directions for future work.

Chapter 2

Physics, PDEs and Deep Learning

2.1 Introduction

The Finite Element Method (FEM) is a numerical solution paradigm, widely used to solve real-world problems rooted in numerous scientific and engineering disciplines. Having revolutionized the process of engineering design, it also proves to be a robust method in modelling a variety of scientific/physical processes involving Partial Differential Equations (PDEs) [1].

Contrary to the classical numerical approximations, machine learning has paved an alternate way to solve the underlying PDEs – more often expressed as *boundary value problems* (BVPs), skipping the conventional matrix computations. This chapter begins with discussing the fundamentals of FEM and BVPs. Subsequently, we introduce the concept of Geometric Deep Learning (GDL) with graphs, alongside addressing the limitations of Convolutional Neural Networks (CNNs). This is followed by a discussion on Graph Neural Networks (GNNs) and its variants as categorized in the literature. Further, we present an overview of data-driven deep learning methods, proposed and implemented for physics-based problems governed by and described using PDEs. We also survey deep learning techniques for electromagnetics (with brevity) in line with this thesis work.

2.2 Prerequisites

2.2.1 A Boundary Value Problem

In the past decades, several numerical methods have been proposed and employed by numerous scientists and engineers to solve problems involving partial differential equations subject to boundary constraints, widely known as *boundary-value problems* (BVPs). A few examples of BVPs include the Poisson's equation defined for electrostatics and magnetostatics problems, Laplace Equation, heat equation, elasticity (stress/strain displacement), wave equations, etc [11]. The conditions defined and imposed on the boundary $\partial\Omega$ of the domain Ω are referred to as *boundary conditions*. Commonly defined boundary conditions include *Dirichlet*, *Neumann* or a combination of both (*Robin, mixed*).

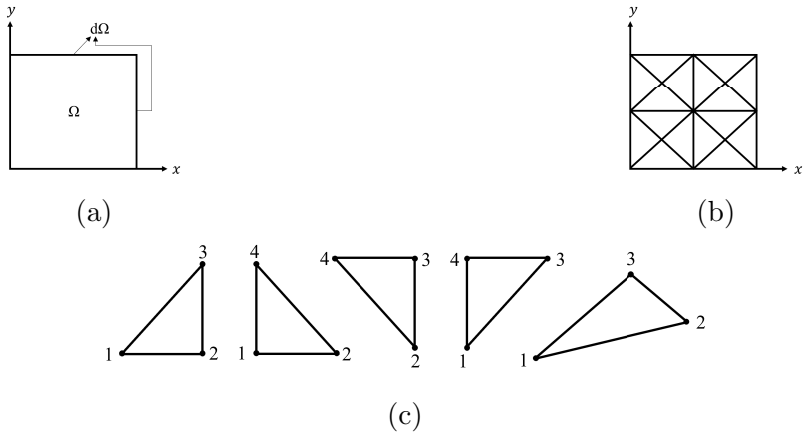


Figure 2.1: Mesh generation; (a) Rectangular solution domain Ω; (b) Discretized Ω (mesh); (c) Triangular finite elements. Image adapted from [12].

For an unknown function $u = u(x, y)$ and a bounded domain Ω[1] (Figure 2.1a), specifying its value on the boundary $\partial\Omega$ refers to the Dirichlet boundary condition. Specifying the normal derivative[2] of u on the boundary (using a unit normal vector $\hat{\mathbf{n}}$) indicates the Neumann boundary condition.

[1] Domain Ω can be arbitrarily-shaped; not restricted to rectangular geometries.
[2] Normal derivative $\partial u/\partial\hat{\mathbf{n}} \perp \Omega$.

A typical definition of a BVP (2D PDE) is as follows:

$$\nabla^2 u + u = 0, \qquad \forall x, y \in \Omega \tag{2.1}$$

$$u(x, y) = f(x, y), \qquad \forall x, y \in \partial\Omega \qquad (Dirichlet) \tag{2.2}$$

$$\frac{\partial u}{\partial \hat{\mathbf{n}}} = g(x, y), \qquad \forall x, y \in \partial\Omega \qquad (Neumann) \tag{2.3}$$

Here, the gradient operator ∇ is a column vector of partial derivatives in two dimensions given by $\big(\partial/\partial x; \partial/\partial y\big)$. The Laplacian operator ∇^2 is defined as $\nabla^2 = \nabla \cdot \nabla$. Let ∇u denote the gradient of scalar u. Then, the Laplcian of u can be redefined as the divergence $(\nabla\cdot)$ of gradient of u:

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \\ \frac{\partial}{\partial y} \end{bmatrix}, \nabla u = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial u}{\partial y} \end{bmatrix}; \nabla^2 u = \begin{bmatrix} \frac{\partial}{\partial x} \\ \\ \frac{\partial}{\partial y} \end{bmatrix}^T \cdot \begin{bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial u}{\partial y} \end{bmatrix} = \frac{\partial^2 u}{\partial x} + \frac{\partial^2 u}{\partial y} \tag{2.4}$$

2.2.2 Maxwell's Equations

A classical electromagnetics problem is governed by a powerful set of partial differential equations, famously known as Maxwell's equations. This mathematical framework imparts an insightful description relating the electric and magnetic fields with charges, currents and their sources. As taken from [13], following are the four equations in their point forms:

$$\nabla \cdot \mathbf{D} = \rho_v \tag{2.5}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.6}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.7}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \tag{2.8}$$

Table 2.1 enlists key electromagnetic quantities and their respective units [13]. Equation

2.5, also known as point form of *Gauss's law*, states that divergence of electric flux density **D**, i.e. the electric flux per unit volume leaving an infinitesimally small volume, equals the enfolded volume charge density ρ_v . Equation 2.6 is the point form of *Gauss's law for magnetism*, signifying that magnetic monopoles do not exist. In other words, divergence of magnetic flux density **B** equals zero. The point form of *Faraday's law* as shown in equation 2.7 implicitly states that a time-varying magnetic field generates an electric field **E**. Further, curl of **E** emphasizes the rotational nature of the field. Equation 2.8, better known as point form of *Ampere's circuital law*, implicitly describes the generation of magnetic field **H** by a time-varying electric field. In other words, the current density **J** and displacement current density $\frac{\partial \mathbf{D}}{\partial t}$ attribute to a rotational magnetic field. Given the conductivity σ of a material, $\mathbf{J} = \sigma\mathbf{E}$. Equations 2.5 - 2.8 in their integral form are stated in [13].

Table 2.1: Electromagnetic Quantities and their units.

Symbol	Unit	Symbol	Unit
ρ_v	Coulomb/meter^3; C/m^3	σ	Siemens/meter; S/m
μ	Henry/meter; H/m	ε	Farad/meter; F/m
ϕ	Volt; V	A_z	Ampere; A
J	Ampere/meter^2; A/m^2	**A**	Weber/meter; Wb/m
D	Coulomb/meter^2; C/m^2	**E**	Volt/meter; V/m
B	Tesla (Weber/meter^2); T (Wb/m^2)	**H**	Ampere/meter; A/m

2.2.3 Finite Element Method : An Overview

The term "Finite Element Method" was first coined by a structural engineer named Ray W. Clough in his paper at the *Proceedings of 2 nd ASEC Conference on Electronic Computation* in 1960. Alongside his landmark contribution towards FEM, the development

of FEM can be traced back to the early 1940s with seminal works by A.Hrennikoff [14], Douglas McHenry [15], R.Courant [16], John H. Argyris [17], M. J. Turner [18], O. C. Zienkiewicz [19]. FEM, though originally developed for structural analysis arising in civil and aerospace engineering, has had its application extended to several other scientific fields such as electromagnetics, optics, fluid dynamics and so on. In general, a typical finite element analysis of boundary-value problems (described in section 2.2.1) comprise the following steps [12]:

1. Discretization of the PDE defined over a domain Ω. This refers to the division of Ω into subdomains or finite elements (mesh generation). The shape of these finite elements include simplices such as triangles, quadrilaterals in 2D, tetrahedrons (3D) and higher-order simplicial configurations for multi-dimensional domains.

2. Choosing appropriate *basis* or *interpolation* functions, which are later expressed as a weighted linear combination to approximate the solution corresponding to each finite element.

3. Transformation of the defined PDE into a set of linear equations for each element i.e. *element equations*.

4. Assembling the element equations to form a global set or a *system equation*. Such formulations are presented using the Rayleigh-Ritz method, Galerkin weighted residual method and other approaches.

5. Solving the linear *system* equation of the form $\mathbf{Au} = \mathbf{b}$ using efficient solvers such as the Conjugate Gradient method, Cholesky/LU decompositions, etc.

For a rectangular solution domain Ω shown in Figure 2.1a, the process of mesh generation can yield a set of triangular (Figure 2.1b) or rectangular (or any general quadrilateral) elements. Further, the first four simplices in Figure 2.1c illustrate a set of right triangular elements whereas the fifth simplex denotes a general triangular finite mesh element.

2.2.4 Towards Deep Learning

It is important to note that $\mathbf{Au} = \mathbf{b}$ solved using linear algebraic methods (exact or iterative), require large computational resources. Despite them yielding optimal results, operations such as matrix inversion prove to be numerically expensive. Further, tuning certain parameters in the code may mandate multiple reruns - hinting towards an increased resource requirement. To tackle this infeasibility, data-driven methods pave an alternate way to obtain solutions to several such boundary value problems, that are implicitly defined by partial differential equations. This leads us into the world of Artificial Intelligence (famously abbreviated as AI), wherein tasks are accomplished with minimal or no human intervention. Two subdivisions of AI - Machine Learning (ML) and Deep Learning (DL), constitute numerous techniques that are capable of learning patterns and approximating solutions to the "problem under consideration", solely based on the input data samples [20].

Machine learning techniques are broadly classified into *supervised* and *unsupervised*. Further, such predictions are made based on a set of "*features*", and the outcomes are either categorical (*classification*) or quantitative (*regression*). The supervised learning process is guided by a "labelled" *training* dataset, where "labelled" refers to each input sample or observation (tuple) in the training dataset having a corresponding class variable (outcome). On the other hand, unsupervised learning models train and predict on an unlabelled dataset. Examples of ML models include Decision Trees, Naive Bayes classifier, Linear Regression models, Random forests, Support Vector Machine, Artificial Neural Networks (ANNs), etc [21]. Other learning regimes include *Semi-supervised* [22] and *Reinforcement learning* [23]. The former class of algorithms learn and predict on a partly labelled training dataset and the latter deal with "reinforcement agent" that learns through its experiences and takes appropriate actions to maximize the reward signal, unlike learning the hidden mapping/patterns in ML/DL methods.

Although such ML algorithms find vast applications in areas such as image recognition

and text classification, they require the raw input data be transformed into a suitable feature vector representation for training purposes and predictions, making it a strenuous manual process. In this prospect, a new class of techniques known as Deep Learning (DL) emerged to cater to an escalation of day-to-day applications. Unlike the basic ML models, DL algorithms aim to self-learn the key features (auto *feature extraction*) required for detection or classification tasks in various learning settings as described for ML models above. Specifically, artificial neural networks for deep learning are multilayered (has several hidden layers) to deduce a non-linear input-output mapping, justifying the term "deep" in its true sense.

2.3 Geometric Deep learning with Graphs

2.3.1 Shortcomings of CNNs

Training a typical *fully-connected* neural network (Figure 2.2) involves a *feedforward* pass and a *backpropagation* mechanism. In the feedforward step, the training data is propagated through the network model from the input layer to output layer, and the model weights/parameters are computed on passing through consecutive hidden layers. In order to minimize the cost function (e.g. Mean Squared Error (MSE)), the neural network model parameters are backpropagated from the output layer to the input layer, so as to compute the gradients of the cost function with respect to the underlying network weights at each layer. In this manner, the model weights are updated until convergence or sufficient model iterations [24]. Essentially, a machine learning task boils down to an optimization problem.

Convolutional Neural Network (CNN) [25, 26] is one such class of deep neural feedfoward networks that gained popularity with its applications in image and document recognition. Typically, a CNN uses a convolutional operator (kernel) that slides over a structured grid, aggregates information from the underlying nodes and neighbours (e.g. pixels in an image) and yields an updated pixel value. In an effort to use CNN as a PDE solver, Özbay et al.

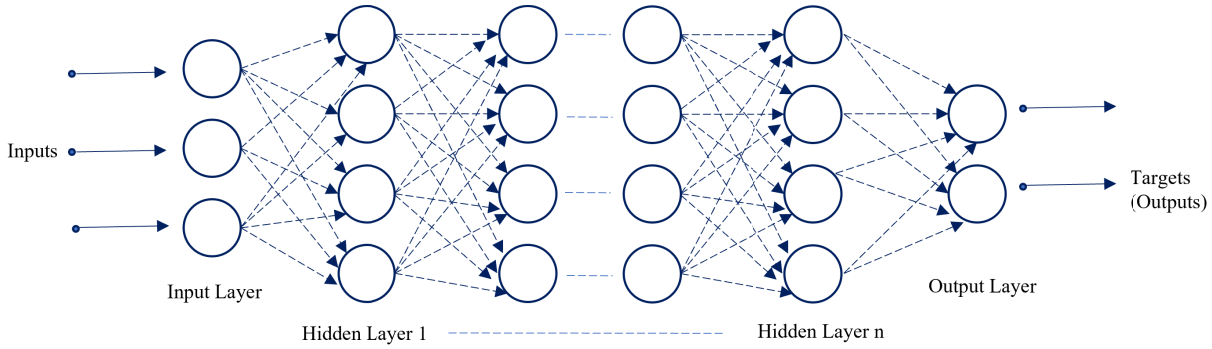


Figure 2.2: A fully-connected neural network.

in [27] propose a CNN to solve the Poisson's equation on a two-dimensional Cartesian grid.

2.3.1.1 Geometric Deep Learning (GDL)

Despite deep learning networks such as CNNs having demonstrated remarkable performances (high accuracies) for various applications, such models are useful only on grid structured data in Euclidean spaces (2D grids/lattices) and text sequences (1D) as opposed to the requirement of processing arbitrary meshes (or) networks. This necessitates a newer deep learning architecture to extend the idea of convolution onto the graph domain and analyze complex patterns in general graph networks. Bronstein et al. in [28, 29] term this process of generalizing deep neural models (such as CNNs) onto non-Euclidean domain geometric data as *Geometric Deep Learning*. They focus on a broad class of neural network architectures proposed to analyze grids, groups, graphs and manifolds, unstructured sets, geodesics and gauges, and demonstrate their unification with regards to first geometric principles of *structural symmetry* and *invariance*. Leveraging such geometric priors aids in encoding stable and multsicale geometric representations and thereby, the quality of information learned by the neural model.

Further, the intuition behind generalization is not only to derive non-Euclidean counterparts for pooling and convolutional layers, but generalize across various domains. The latter is an essential requirement for large scale applications such as computer vision and graphics

[30, 31], physics [32], recommender systems [33], etc. The reader is referred to [28, 29] for an in-depth understanding and analysis of geometric deep learning concepts, applications and relevant approaches with key reference publications.

(a) (b)

Figure 2.3: GNN vs CNN; (a) Unstructured/non-Euclidean mesh (GNN input); (b) Structured/Euclidean grid (CNN input).

2.3.2 Graph Neural Networks (GNNs)

Graph Neural Networks belong to a class of *node embedding* techniques, that aim to encode the graph nodes as low-dimensional vectors onto a hidden space and decode these node embeddings to reconstruct information about its local neighbourhood in the original graph [2]. A sample set of mesh inputs handled by GNN and CNN are shown in Figure 2.3.

Interestingly, the fundamental GNN model was motivated by and proposed in multiple ways, distinguishable on the basis of underlying message passing (update) and aggregating functions. In view of these formulations, authors in [28] with brevity, categorize a section of literature proposed in the context of GNN, into three "flavours" – *convolutional*, *attentional* and *message passing*. Such approaches with diverse parameter sharing strategies across the network emphasizes the *inductive* learning characteristic of GNN, to generalize better over unseen data by leveraging a deep learning architecture satisfying *permutation equivariance*[3] or *permutation invariance*[4] [28, 2].

[3] For a permutation matrix $\mathbf{P}$ and node feature matrix $\mathbf{X}$, function f is permutation equivariant if $f(\mathbf{PX}, \mathbf{PAP}^T) = \mathbf{P}f(\mathbf{X}, \mathbf{A})$.

[4] f is permutation invariant if $f(\mathbf{PX}, \mathbf{PAP}^T) = f(\mathbf{X}, \mathbf{A})$. $\mathbf{PAP}^T$ is the representation of permutation group acting on the graph-associated matrices. f is defined analogous to Equation 2.9.

2.3.2.1 Neural Message Passing Model

In general, a graph is defined by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of *vertices* or *nodes* $\mathcal{V}$ and a set of edges $\mathcal{E}$ connecting the vertices. This nodal interconnectivity is often expressed through an *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where each entry A_{uv} denotes the presence ($A_{uv} = 1$) or absence ($A_{uv} = 0$) of an edge $(u, v) \in \mathcal{E}$ between node $u \in \mathcal{V}$ and $v \in \mathcal{V}$. Given a node feature matrix $\mathbf{X} \in \mathbb{R}^{d\times|\mathcal{V}|}$, each GNN layer is updated in the following manner [2]:

$$\mathbf{h}_u^{(k)} = \text{UPDATE}^{(k)}\big(\mathbf{h}_u^{(k-1)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\})\big) \tag{2.9}$$

where $\mathbf{h}_u$ and $\mathbf{h}_v$ denote the node (hidden) embeddings for nodes u and v respectively.[5] Neighbourhood of node u is denoted by $\mathcal{N}(u)$ with the AGGREGATE ($\bigoplus$ acting on message ψ) and UPDATE (ϕ) functions being arbitrary differentiable (i.e., neural networks). For each iteration (GNN layer) k, the hidden embeddings in the graph neighbourhood of u from the previous iteration – $\mathbf{h}_v^{(k-1)}$ are aggregated to form a message, which is further combined with an embedding of node u from the previous iteration – $\mathbf{h}_u^{(k-1)}$, to generate an updated embedding $\mathbf{h}_u^{(k)}$ [2]. Note that the initial input to the GNN is a set of node features corresponding to all the nodes i.e., $\mathbf{h}_u^{(0)} = \mathbf{x}_u$. After K iterations, the node embedding (output layer) for each node would be $\mathbf{z}_u = \mathbf{h}_u^{(K)} \;\; \forall k \in \{1, ..., K\}$. Figure 2.4, 2.5 and 2.6 are adapted with permission from the original figure creator (also an author) of [28], © Petar Veličković, Google). Alternatively, given the node feature vectors $\mathbf{x}_u, \mathbf{x}_v$, the aggregated message can be expressed as $m_{uv} = \psi(\mathbf{x}_u, \mathbf{x}_v)$. Then,[6,7]

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v\in\mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right) \tag{2.10}$$

[5] Equations 2.9 and 2.10 are taken from [2] and [28] respectively.

[6] Permutation invariant operator $\bigoplus$ can be sum, mean, max, etc.

[7] Examples of $\psi, \phi : \psi(\mathbf{x}) = \mathbf{Wx+b}$, $\phi(\mathbf{x,m}) = \sigma(\mathbf{Wx+Um+b})$, where σ is a non-linear activation function (such as ReLU) and $\{\mathbf{W, U, b}\}$ are trainable parameters.

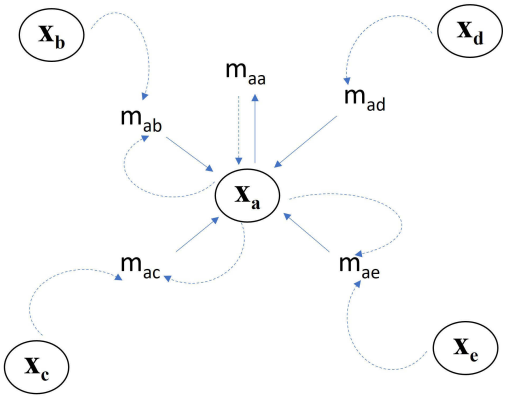


Figure 2.4: A message passing model depicting message aggregation at a single node from its local neighbourhood. Image adapted from [28].

Relevant GNN models derived using the "message passing" mechanism can be found in [34, 35, 36, 37].

2.3.2.2 Convolutional approaches

In accordance with certain convolutional aggregation layers proposed in [38, 39, 40], the node features in the neighbourhood $\mathcal{N}_u$ are aggregated as a weighted linear combination shown below:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{uv}\psi(\mathbf{x}_v)\right) \tag{2.11}$$

where coefficient c_{uv} (dependent on adjacency matrix $\mathbf{A}$ entries) enumerates to node u's

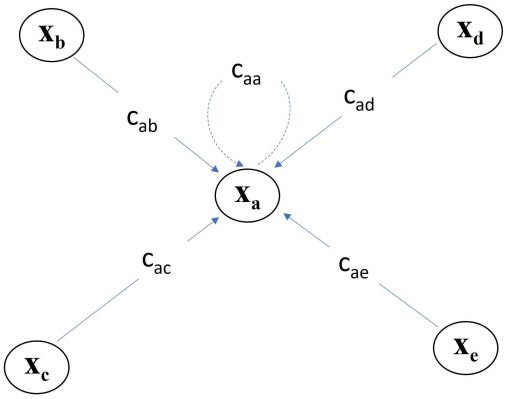


Figure 2.5: Convolutional GNN layer; Image adapted from [28].

representation, the importance of node v. Alongside $\bigoplus$ denoting the summation operation, it can generalize the notion of convolution when considered as a diffusion or a node-wise local filter (linear) operator. Generalizing convolutions onto non-Euclidean data such as graphs involve key principles from graph signal processing theory [41]. Such convolutional deep learning frameworks involving eigenvalues, eigenvectors, graph Fourier transforms are often termed *"spectral-based approaches"*. Convolutions expressed directly on the vertex (graph) domain by aggregating information in the local node neighbourhood are referred to as *"spatial approaches"* [42, 43].

2.3.2.3 Attention-based approaches

Attention-based (Attentional) graph neural network models learn and enable implicit interactions by leveraging a self-attention mechanism [44, 45, 46]. Coefficients $\alpha_{uv} = a(\mathbf{x}_u, \mathbf{x}_v)$ are inherently computed, followed by a *softmax-normalization* across all its neighbours [45]. Mathematically, interactions in each GNN layer are modelled as:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \alpha_{uv} \psi(\mathbf{x}_v)\right) \tag{2.12}$$

While $\bigoplus$ denotes summation, the aggregation of messages is still a weighted linear combination of features in $\mathcal{N}_u$. However, the importance coefficients α_{uv} are feature-dependent.

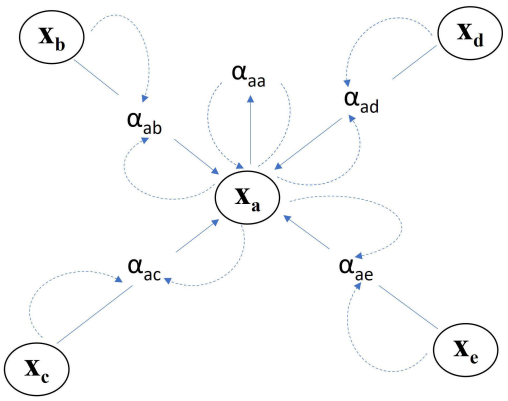


Figure 2.6: Self-attention based GNN layer; Image adapted from [28].

Implications: It is remarkable to note the hierarchical transformation among the three graph approaches in the order : *convolutional* $\subseteq$ *attentional* $\subseteq$ *neural message-passing.* In other words, the self-attention and convolutional GNNs can be termed special cases of the message passing GNN variant. Consequently, their weighted message functions can be transformed as shown below:

$$\psi(\mathbf{x}_u, \mathbf{x}_v) = c_{uv}\psi(\mathbf{x}_v) \quad \rightarrow \quad \text{Convolutional} \Leftrightarrow \text{message-passing} \tag{2.13}$$

$$\psi(\mathbf{x}_u, \mathbf{x}_v) = \alpha_{uv}\psi(\mathbf{x}_v) \quad \rightarrow \quad \text{Attentional} \Leftrightarrow \text{message-passing} \tag{2.14}$$

The aforementioned categorization may seem comprehensive, it however excludes certain GNN models such as k-GNN introduced on the grounds of Weisfeiler-Lehman hierarchy [47], spectral GNNs requiring an explicit computation of graph signal processing entities such as Graph Fourier Transform. For other comprehensive surveys on GNN taxonomy, approaches and relevant literature (in different settings), the reader can refer to the material presented in [42, 43, 48, 49, 50, 51, 52].

2.4 Deep Learning for Physical Systems - A Survey

2.4.1 Motivation and Intuition

A fundamental aspect of perceiving and mastering the intuition behind human intelligence is to model real-world physical systems mathematically described by partial differential equations (PDEs). In order to learn and make predictions, simulating such real-world systems mandates the model to rightly understand and regard the governing principles or laws. Several such physical systems can be modelled as system objects (nodes) with pair-wise interactions. This course of physical system modelling using nodes and pair-wise connections is analogous to a well-known geometric datatype – *graphs* [42].

Early attempts to formulate deep learning on graphs (rather, generalize neural networks

to graphs) were presented by Scarselli, F., Gori, M., et al [34, 35]. This work introduced the eminent Graph Neural Network (GNN) framework, based on recurrent neural networks and random walk models. A surge in the application of GNNs in recent years is attributed to the ubiquitous graph structured data generated across several engineering and science disciplines. A variety of GNN models have been proposed in the literature to efficiently handle arbitrarily sized topographical networks, skip the conventional numerical method approximations of PDEs and train on large datasets with growing computational resources such as Graphics Processing Units (GPUs).

2.4.1.1 Preliminary Neural Network Models

Initial endeavours to learn the physical dynamics explored the potential of primary machine learning algorithms such as feedforward neural networks (ANNs). In their seminal work, Lagaris et al. [53] and Chiaramonte et al. [54] trained ANNs to learn the solutions of ordinary and partial differential equations. This boosted several other researchers to train and test various machine learning frameworks to learn the solutions for PDEs, as discussed in [55, 56, 57, 58, 59, 60]. CNNs, having gained wide popularity in text/image processing applications, was a potent candidate to approximate PDE solutions. This line of work includes PDE-Net [61] proposed to learn evolutions PDEs through convolutional filters (differential operators) and non-linear responses; the model was experimented on convection-diffusion equations. Analogous to the conventional convolutional neural networks, MeshCNN by Hanocka et al.[62] learns for shape classification and segmentation tasks, by integrating specialized pooling and convolutional layers, and leveraging implicit geodesic connectivity in mesh edges (obtained from a highly non-uniform meshed data). In an attempt to solve elliptic PDEs using deep learning, proponents in [63, 64, 65] solve Poisson's equations using CNNs under varied boundary conditions.

However, structured data-based tasks form relatively a small portion in comparison to those dealing with highly unstructured data. Analyzing unstructured meshes using CNNs

seem a non-trivial task due to its constrained requirement of regular shapes. Efforts to solve parametric PDEs using a physics-informed CNN for irregular geometries in an unsupervised learning setting is proposed in [66]. This structural limitation opened doors to analyze commonly found, yet complex topological networks, thus demanding advanced deep learning architectures.

2.4.2 GNN Approaches

Battaglia et al. in [32] introduce the first learnable physics engine in the form of interaction networks (INs) using deep neural network building blocks (MLPs [67], for an in-depth reasoning about relations among objects modelled for challenging physical problems such as rigid-body collision. They learn the root dynamics and infer the system's abstract properties using graph-based data generated by real-time physical system(s) simulations. Addressing the drawbacks of inferior interactions scalability in [32], Hoshen et al. in [68] introduce a vertex attention IN (VAIN) for multi-agent prediction domain tasks.

Motivated by the work done in [32], Chang et al. [69] propose a Neural Physics Engine to learn the simulator dynamics and generalize across different object count and scene configurations. In view of understanding complex rigid deformable bodies, Mrowca et al. [70] present a graph convolutional neural network (hierarchical) to learn the physics predictions, based on a cognitive-science inspired hierarchical graph-based representation of objects. Extending the approach of INs, Sanchez-Gonzalez et al. in [71] introduce a Graph Network model, go on to generalize it in [4] by proposing "Graph Network-based Simulators"(GNS) to learn the complex dynamics from simulated data acquired across a wide range of physical systems (rigid solids, deformable bodies, fluid dynamics). With an introduction of "Encoder-Processor-Decoder" architecture in their work, particle-based simulations are carried out wherein, each particle represents a graph node and simulations are viewed as message-passing on graphs.

2.4.2.1 Mesh-based Deep learning

Besides the aforementioned learnable physics simulator approaches, there has been significant research in the literature in context with physics-based simulations focusing on geometric aspects of input data such as meshes. These frameworks are often termed "mesh-based" processing/learning techniques. This indeed, refers to approaches categorized under the umbrella term "Geometric Deep Learning", coined by Bronstein et al. [28, 29]. A brief introduction to GDL was previously presented in Section 2.3.1.1. Following the line of work discussed in [4], Pfaff et al. [5] introduce a GNN-based framework to learn mesh-based simulations for time-dependent problems called "MeshGraphNets". Their framework is designed to learn resolution-independent dynamics from meshed data (triangular), yielding high accuracies (with 11x-290x simulation speedup) in physical systems such as fabrics (cloth), fluid dynamics and structural mechanics. Scaling the MeshGraphNets to a 3.1 million-node mesh to capture computational fluid dynamics simulations, Bartoldson et al. [72] demonstrate the possibilities to train such GNN-mesh-based physics models on three-dimensional meshes. Alet et al. [73] implement a Graph Element Network (GEN) to learn the solution to a PDE, given its initial states (based on FEM numerical simulations). They demonstrate the optimization of a FEM mesh using GEN, with key focus on the non-linear section of the solution space. A survey on deep-learning techniques for mesh-based data can be found in [74].

A novel neural network architecture – "Simplicial Attention Networks" (SAT) was introduced in [75] that dynamically weighs interactions among neighbouring simplices leveraged by attention-mechanisms, and generalize to unseen simplicial networks. From a scalability perspective, several works investigate multi-level GNNs for physics-simulations [76, 77], wherein such multi-scale graph neural models process and learn on meshes with varied granularity levels such as *fine* and *coarse*. A detailed survey of deep learning algorithms proposed as PDE solvers is discussed in [78, 79, 80].

2.4.2.2 Neural Operators & Physics-Informed Learning

Neural Operators, a novel class of neural networks introduced by Li et al. [6, 7], learns the PDE's solution operator through an input-output mapping among infinite-dimensional spaces in both semi-supervised and supervised settings. This way, the proposed data-driven approach predicts the solution with no prior knowledge about the PDE being solved. Brandstetter et al. [81] extend the work presented in [6] by introducing an end-to-end neural message passing PDE solver. They generalize across various PDEs and implement methods to express classical numerical PDE solvers as versions of autoregressive GNN models.

Recent trends in deep learning speak of *Physics-Informed Neural Networks* (PINNs), proposed as an efficient solution to address the limitation of low, benchmarked training data availabilty and yield state-of-the-art accuracies for several physical phenomena. This novel class of neural networks introduced by Raissi et al. [82, 83, 84], learn the partial differential equation solution and underlying non-linear dynamics, with prior embedded knowledge of governing physical laws (spatio-temporal PDE data). Their two-part treatise of *data-driven solutions* and *data-driven discovery* have paved the way ahead to solve PDEs for varied complex physical systems. As a predecessor to PINNs, Aarts et al. [85] employ a feedforward neural network incorporated with prior information about PDEs (a geohydrological and a physical process) and its initial-boundary conditions. Several such physics-embedded data driven approaches have been summarized by Pateras et al. in [8].

Reflecting on the deep learning approaches discussed above, an alternate categorization of these techniques would be on the basis of input data structure – *regular (structured)* meshes, *irregular (unstructured)* or *mesh-less (mesh-free)*. Mesh-free methods often include particles and point cloud (set of 3D points) data representations, thus eliminating a computationally exhaustive processing of meshes for learning purposes. Relevant works include the Graph Network Simulator proposed in [4], Lagrangian fluid simulations [86], non-linear BVPs [87], 3D solid mechanics [88], diffusion maps based PDE solver [89].

2.4.3 ML/DL for Electromagnetics - A Brief Review

Besides extensive deep learning research done in the fields of computer vision, recommender systems, a key branch of physics — "Electromagnetics" (EM) has gained popularity among a wide community of researchers in the recent years. Preliminary machine learning and deep learning models exist in the literature studying smart metamaterials [90], antenna systems [91] and several other aspects of EM surveyed in [92, 93]. A comprehensive state-of-the-art research in electromagnetics (simulation, optics, radio frequency device modelling, inverse problems) have been reviewed and presented by Campbell et al. in [94].

Regardless of the system complexity, solving the Poisson's equation (which can be obtained from the powerful set of Maxwell's equations), form an integral part of electromagnetic analysis. Conventionally, solutions to these PDEs are approximated using classical numerical solvers such as the Finite Element Method. In recent years, successful attempts in adopting deep learning methodologies for such boundary value problems have been demonstrated using a finite-element neural network [95], a CNN-based architecture [96, 97], a spectral graph convolutional layer in a GNN model [10, 3] and an attention-based Operator Transformer [98].

Early ideas to use physics-informed neural networks as a potential EM solver via transfer learning is presented by Khan et al. [99]. They investigate the behaviour of PINNs (with and without labelled training data) and achieve accurate steady-state solutions for well-posed electrostatics and magnetostatics problems. Physics-embedded approaches for other EM problems such as magnetic/magnetostatic field estimation are reviewed in [100],[101],[102]. Further, key efforts to solve inverse problems arising in electromagnetics include the hypernetwork proposed by Lowther et al. [103] for an optimal coil design, a 3D meshless DNN and a contrastive learning-based subspace optimization and semantic segmentation-assisted reconstruction (CLSO-SSR) model for EM inverse scattering in [104] and [105] respectively.

Such ground-breaking scientific research has certainly brought to our attention various ways to generate highly accurate and reliable outcomes. However, it is equally important for these data-driven approaches to be considered not a replacement to the classical numerical methods (FDM, FEM), but only an alternative set of methodologies to approximate solutions to PDEs.

Chapter 3

Solving Poisson's Equation using Graph Neural Networks

3.1 Introduction

Previously, we were introduced to various deep learning methodologies adopted to solve a partial differential equation arising in scientific and engineering problems. Specifically, Graph Neural Networks seem to be the parent model heading a broad class of message-passing frameworks, serving as an alternative to conventional numerical PDE approximation methods.

This chapter focuses on one such GNN model implemented to learn on meshes and solve the Poisson's equation for electrostatics and magnetostatics problems. A comprehensive description of the GNN model, its GPU implementation and results, supported by the variational formulation of Poisson's equation boundary value problem (2D) are presented in the following sections. Experiments presented in the following sections are based on the material proposed by Lötzsch et al. in [3, 106].

3.2 The Poisson's Equation BVP

The steady-state time-independent Poisson's equation with a homogenous Dirichlet boundary condition in the spatial domain $\Omega \in \mathbb{R}^2$ is given by:

$$-\nabla^2 u(\mathbf{g}) = f \quad \forall\, \mathbf{g} \in \Omega \tag{3.1a}$$

$$u(\mathbf{g}) = 0 \quad \forall\, \mathbf{g} \in \partial\Omega \tag{3.1b}$$

where $\mathbf{g}$ denotes the two-dimensional spatial coordinate vector i.e. $\mathbf{g} = (x, y) : g_i \in \mathbb{R}$ in an open planar domain Ω bounded by $\partial\Omega$. Further, $\nabla^2 u(\mathbf{g}) = \frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y}$, where the *trial function* $u(\mathbf{g}) = u(x, y)$ denotes the solution to the aforementioned BVP (Equation 3.1a and 3.1b). Classical solution to the *strong formulation* of the Poisson BVP described above will be u, a twice differentiable function in the continuous solution space $C^2(\Omega)$ satisying the Dirichlet boundary condition at all points $\mathbf{g} \in \partial\Omega$. However, there may be practical scenarios limiting the differentiability of u (discontinuous or non-existent derivatives). Thus, we opt for a "*weak formulation*" of the boundary value problem defined over an appropriate solution space, as discussed in section 3.2.1.

3.2.1 Variational Formulation

As a first step in transforming a strong form BVP into a "weak" or "variational" problem, we multiply the second-order PDE by a *test function* v and integrate over Ω [107]:

$$-\int_\Omega (\nabla^2 u) v \; d\mathbf{g} = \int_\Omega f v \; d\mathbf{g} \tag{3.2}$$

Equation 3.2 can be expanded as : $-\int_\Omega (\nabla^2 u) v \; dxdy = \int_\Omega f v \; dxdy$, where $d\mathbf{g} = dxdy$ represents a differentiable element integrable over Ω. Integrating equation 3.2 by parts, its

left-hand side can be rewritten as:

$$\int_{\Omega} \nabla u \cdot \nabla v \; d\mathbf{g} = \int_{\partial\Omega} v \frac{\partial u}{\partial n} \; ds - \int_{\Omega} (\nabla^2 u) v \; d\mathbf{g} \tag{3.3}$$

$$\int_{\Omega} \nabla u \cdot \nabla v \; d\mathbf{g} = \int_{\Omega} f v \; d\mathbf{g} \tag{3.4}$$

where ds is a differentiable element integrable on $\partial\Omega$ and $\frac{\partial u}{\partial n}$ denotes the normal derivative (outwardly orthogonal to $\partial\Omega$). Further, Equation 3.3 refers to the "Green's identity" or "Green's formula". Requiring the test function to vanish on parts of $\partial\Omega$ where u is known, i.e., $v = 0$ on $\partial\Omega$, Equation 3.3 reduces to a dot product of gradients $(\nabla u, \nabla v)$ integrated over Ω as shown in Equation 3.4 [108]. Given the homogenous Dirichlet boundary condition, a suitable trial and test function space for Eq. 3.4 is a Hilbertian Sobolev space $H_0^1(\Omega)$. This space consists of (continuous) functions and their weak first order derivatives that vanish on the boundary and are square-integrable [109]. Mathematically, $\int_{\Omega} v^2 d\Omega \; < \infty$, $\int_{\Omega} |\nabla v^2| d\Omega \; < \infty$ and $v|_{\partial\Omega} \; = \; 0$. Consider the source function $f \in L^2(\Omega)$, where $L^2(\Omega)$ denotes the Lebesgue space containing a set of square-integrable functions (continuous, unbounded and discontinuous). For convenience, let the trial and test function spaces be denoted by V and $\hat{V}$ respectively[1]. Subsequently, we define our variational problem as to find a solution $u \in V$ such that:

$$\int_{\Omega} \nabla u \cdot \nabla v \; d\mathbf{g} = \int_{\Omega} f v \; d\mathbf{g} \quad \forall v \; \in \; \hat{V} \tag{3.5a}$$

$$a(u, v) = b(v) \; \forall v \; \in \hat{V} \tag{3.5b}$$

where $u, v \in H_0^1(\Omega)$ implicitly satisfy the homogenous Dirchlet boundary condition. However, for a non-homogenous Dirichlet boundary problem, the trial function space changes to an *affine space* [109]. Note that the weak formulation in Equation 3.5a is still a continuous

[1] Trial space $V \subset H_0^1(\Omega)$, Test space $\hat{V} \subset H_0^1(\Omega)$.

problem with solution u in an infinite-dimensional trial space V. Equation 3.5b represents the abstract variational formulation[2] of the boundary value problem.

Revisiting the first step of FEM - discretization of the spatial domain, triangulation of Ω would result in K triangular (non-overlapping) finite elements whose solution u is approximated in a finite-dimensional subspace $V_h \subset V(trial)$ and $\hat{V}_h \subset \hat{V}(test)$. This leads to a discrete version of the variational problem stated to find $u_h \in V_h$ such that:

$$\int_\Omega \nabla u_h \cdot \nabla v_h \; d\mathbf{g} = \int_\Omega f v_h \; d\mathbf{g} \quad \forall v_h \in \hat{V}_h \tag{3.6a}$$

$$a(u_h, v_h) = b(v_h) \quad \forall v_h \in \hat{V}_h \tag{3.6b}$$

Each triangular element $K_e, e \in 1 : |K|$ has a diameter $h_{K_e} = \max_{x,y \in K_e} |x - y|$, pointing to the longest side of element K_e. Thus, the subscript 'h' refers to the largest diameter in this set, i.e., $h = \max_e h_{K_e}$. Intuitively, each node N_j in K_e can be associated with a *basis* function[3] ϕ_i spanning the subspace V_h (equivalent to a Kronecker delta function δ_{ij}) as:

$$\phi_i(x, y) = \phi_i(N_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{3.7}$$

Let u_i and v_j denote the values of u_h and v_h at node N_j respectively. Then, u_h and v_h can be expressed as a (weighted) linear combination of basis function(s) defined above in Equation 3.7:

$$u_h = \sum_{i=1}^{3} u_i \phi_i(x, y) \; ; \; v_h = \sum_{j=1}^{3} v_j \phi_j(x, y) \tag{3.8}$$

where $i, j = 1, 2, 3$ represent the three nodes/vertices of a triangle. Implictly, the domain of

[2] In Equation 3.5b, $a(\cdot, \cdot)$ is a continuous bilinear functional on $V \times V \to \mathbb{R}$ and $l(\cdot)$ is a continuous linear functional on $V \to \mathbb{R}$.

[3] Here, $\phi_i(x, y)$ is a first-order Lagrangian interpolation function : $ax + by + c$.

integration changes from Ω to area over K_e. Combining Equations 3.6a, 3.6b and 3.8 results in an elemental equation corresponding to element K_e:

$$\int_{K_e} \left[\nabla\Big(\sum_{i=1}^{3} u_i \phi_i(x,y) \Big) \cdot \nabla\Big(\sum_{j=1}^{3} v_i \phi_j(x,y) \Big) \right] \, dxdy = \int_{K_e} f \sum_{j=1}^{3} v_j \phi_j(x,y) \, dxdy \tag{3.9}$$

which holds good for all $v_h \in \hat{V}_h$ and can be chosen arbitrarily such that:

$$\sum_{i=1}^{3} u_i \underbrace{\int_{K_e} \nabla\phi_i \nabla\phi_j \, dxdy}_{A_{i,j}^{K_e}} = \underbrace{\int_{K_e} f\phi_j \, dxdy}_{b_j^{K_e}} \tag{3.10}$$

Here, $A_{i,j}^{K_e}$ and $b_j^{K_e}$ denote the entries of the local stiffness matrix and load vector corresponding to finite element K_e. Augmenting these entries for all triangular elements in Ω yields the global stiffness matrix $\mathbf{A}$ and load vector $\mathbf{b}$ expressed as a linear system equation:

$$\mathbf{Au} = \mathbf{b} \; ; \; \mathbf{u} = (u_i)^T, \; i = 1, 2, ..n. \tag{3.11}$$

Discussing the mathematics behind the aforementioned solution spaces is beyond the scope of this thesis. Thus, the reader is referred to [107, 109, 110] for supplementary information.

3.3 Application : Electrostatics and Magnetostatics

Electrostatics deals with static electric fields and stationary charges, whereas magnetostatics explores constant magnetic fields with steady currents. In this context, the Maxwell's equations in the absence of time-varying electric and magnetic fields can be restated as [13]:

(Refer to Table 2.1 for units of the electric and magnetic parameters discussed in this section.)

$$\nabla \cdot \mathbf{D} = \rho_v \tag{3.12}$$

$$\nabla \times \mathbf{E} = 0 \tag{3.13}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{3.14}$$

$$\nabla \times \mathbf{H} = \mathbf{J} \tag{3.15}$$

In other words, $\frac{\partial \mathbf{B}}{\partial t}$ $(\mu \frac{\partial \mathbf{H}}{\partial t})$ in Equation 2.7 and $\frac{\partial \mathbf{D}}{\partial t}$ $(\varepsilon \frac{\partial \mathbf{E}}{\partial t})$ in Equation 2.8 equals zero[4]. Using $\mathbf{D} = \varepsilon \mathbf{E}$ in Equation 3.12 results in $\nabla \cdot \mathbf{E} = \rho_v / \varepsilon$. Further, the electric field $\mathbf{E}$ is expressed as the negative gradient of electrostatic potential ϕ. Combining these mathematical expressions results in the Poisson's equation for Electrostatics:

$$\mathbf{E} \;=\; -\nabla\phi \;;\; -\nabla^2\phi \;=\; \frac{\rho_v}{\varepsilon} \tag{3.16}$$

Analogous to electrostatics, the magnetic vector potential $\mathbf{A}$ associated with the magnetic field $\mathbf{H} = \mathbf{B}/\mu$ is given by $\mathbf{B} \;=\; \nabla \times \mathbf{A} \;\implies\; \mu\mathbf{H} \;=\; \nabla \times \mathbf{A}$. Substituting this expression in Equation 3.15 yields the Poisson's equation for magnetostatics[5]:

$$\nabla \times (\nabla \times \mathbf{A}) \;=\; \mu\mathbf{J} \tag{3.17}$$

$$\nabla \times (\nabla \times \mathbf{A}) \;= \nabla(\nabla \cdot \mathbf{A}) \;-\; \nabla^2\mathbf{A} \tag{3.18}$$

$$\nabla^2\mathbf{A} \;=\; -\mu\mathbf{J} \quad (\because \nabla \cdot \mathbf{A} = 0) \tag{3.19}$$

For experimental purposes, we consider $J_x = J_y = 0$ and currents directed only along the z-axis (perpendicular to the 2D mesh). This implies the existence of A_z component alone, resulting in a magnetic scalar potential. Further, computing the curl of $\mathbf{A}$ will simplify into

[4] Propagation medium/material properties – Permittivity ε, Permeability μ.

[5] $\nabla \cdot \mathbf{A} = 0$ in Equation 3.19 refers to the *Coulomb gauge* [111].

curl of A_z, yielding magnetic field components along the x (H_x) and y (H_y) axes:

$$\mathbf{H} = \frac{1}{\mu}\Big(\nabla \times \big[0, 0, A_z\big]\Big) \implies (H_x, H_y) = \frac{1}{\mu}\Big(\frac{\partial A_z}{\partial y}, -\frac{\partial A_z}{\partial x}\Big) \tag{3.20}$$

In fact, $H_z = 0$ can be justified with the reason of symmetry, as this problem can be visualized as a set of infinitely long wires whose cross-section is independent of the z-axis (z-coordinate) [3]. Besides such static (linear) problems, one can also address and analyze the non-linearity arising in media such as ferromagnetic materials, whose properties are principally quantified by μ, σ and ε (future work).

3.4 GNN Framework

3.4.1 Encoder-Processor-Decoder model

As previously discussed, structured grids are majorly defined in the Euclidean domain. However, the experiments in this thesis deal with non-square geometries such as a hollow disk, L-shaped and U-shaped meshes. From the literature, it is evident that CNNs are incapable of handling such irregular structures. Inherently, as we are progressing towards irregular geometric surfaces with triangular tessellations (meshes), we require a much more complex deep learning architecture such as GNNs. In the context of Geometric Deep Learning, proponents in [28, 29] classify these structured, undirected graphs under a combination of graphs and manifolds (2D).

For simulations, we adopt the GNN model in [3], employing an Encoder-Processor-Decoder architecture for learning mesh-based simulations, originally proposed by Pfaff et al. in [5].

- **Encoder:** The encoder consists of a multilayer perceptron (MLP), resembling a fully-connected ANN depicted in Figure 2.2. The encoder MLP has two linear layers inter-

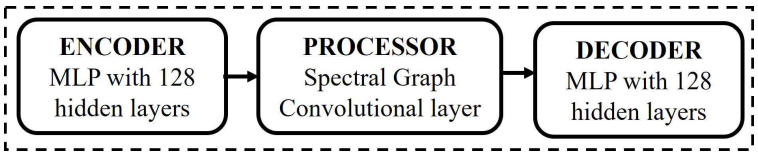


Figure 3.1: A block diagram depicting the Encoder-Processor-Decoder framework.

leaved with non-linear activation functions - Rectified Linear Unit[6] (ReLU) [112] and 128 hidden layers.

- **Processor:** The processor block encompasses a spectral graph convolutional layer proposed by Defferrard et al. [38], more often cited as a predecessor to the Graph Convolution Network (GCN) introduced in [39]. They generalize CNNs onto the graph domain by leveraging spectral graph theory and relevant graph signal processing tools [41] to design graph convolutional filters (fast localized). Similar to the convolutional kernel/filter defined on a CNN, the spectral graph convolution kernel is given by [38]:

$$g_\theta(\Lambda) \;=\; \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \tag{3.21}$$

where $\theta \in \mathbb{R}^K$ represents the Chebyshev polynomial coefficients vector and Λ denotes the diagonal matrix of [7] eigenvalues. $T_k(\tilde{\Lambda}) \in \mathbb{R}^{n\times n}$ is the k^{th} order Chebyshev polynomial, expressed as function of the diagonal matrix of scaled eigenvalues $\left\{\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_n \in [-1, 1]\right\}$. Further, K represents the maximum number of hops in the neighbourhood for convolution on graphs, indexed by variable k. Note that, the term "spectral" associates itself with the eigenvalues used to define the kernel (Equation 3.21).

- **Decoder:** Analogous to the encoder, the decoder is designed as a two-layer MLP with

[6] ReLU activation: $f(x) = \max\{0, x\}$.

[7] Here, the Laplacian defined on a graph with n nodes, degree matrix D and weighted adjacency matrix W is $L = D - W$ [38].

128 hidden layers. The ReLU non-linearity is applied to all its hidden layers except the decoder's output layer.

3.5 Experiments

3.5.1 Datasets

For electrostatics and magnetostatics problems discussed in Section 3.3, we use the dataset provided in [106][8]. The dataset consists of 5 different meshes with square, disk, hollow disk, inverted L-shaped and U-shaped geometries, created using an open-source FEM solver - *FEniCS (FEniCSx)* library [113, 114, 115, 116]. Each mesh and its configuration data (such as resolution, boundary conditions) is appended with the associated PDE solution (potential and field values obtained using the FEM solver), and saved as a single instance in the generated dataset.

3.5.1.1 Mesh Geometries

Following are the five different mesh geometries encompassing the entire dataset with normalized coordinates in the range [0,1] (no units expressed). The outer layer of all the 5 meshes and the inner circle in the hollow disk geometry, represent the boundary nodes.

1. **U-mesh**: Figure 3.2 depicts a regular U-mesh obtained by deleting a U-shaped portion from a unit square. This 68-node mesh is an example of an instance in the test dataset (Table 3.1), and is untouched for mesh augmentation. (**Note:** Meshes shown in Figures 3.2 - 3.7 have been adapted with permission from the first author of [3], © Winfried Ripken (Winfried Lötzsch), Merantix Momentum AI).

[8] A copy of an email from the author of [3], granting copyrights permission for content re-use can be found in the "Copyright" chapter, towards the end of this document.

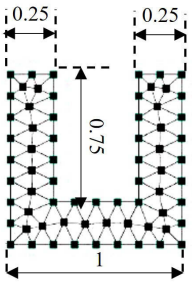


Figure 3.2: U-mesh with 68 vertices. Image adapted from [3].

2. **L-mesh**: Trimming the area of an unit square mesh by 25% (from any corner) yields a L-mesh with 80 vertices as shown in Figure 3.3. Augmenting the L-mesh involves varying the location, its breadth and height of the interior rectangular portion in the range [0.2, 0.8].

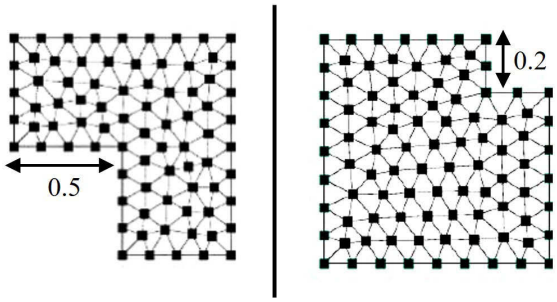


Figure 3.3: A simple triangulated L-mesh (left) and its augmented version (right). Image adapted from [3].

3. **Square**: A 16×16 unit square mesh yields 256 nodes. Its augmented versions (altering the mesh resolution) includes regular grids with nodes in the range 64 - 441. An augmented mesh with the 16×16 mesh is shown in Figure 3.4.

4. **Disk**: A circular mesh, also a disk with 252 nodes is shown in Figure 3.5. Similar to the square mesh, inducing variability in this structure involves varying the node density 'n' (number of nodes spanning the geometry) in the range [63, 411].

5. **Hollow Disk**: The 82-node hollow disk mesh visualized in Figure 3.6, is constructed using a disk (similar to Figure 3.5) with a 0.5-unit radius, centered at (0.5, 0.5). A

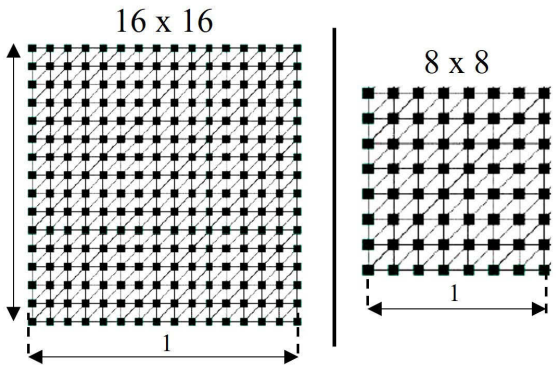


Figure 3.4: A mesh with square geometry (left); Augmented square mesh (right). Image adapted from [3].

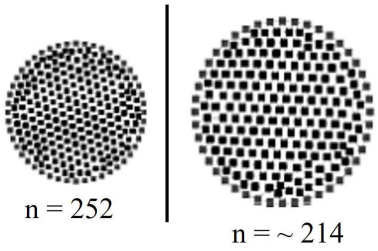


Figure 3.5: Circular (Disk) mesh (left); Augmented disk mesh with varied node density 'n' (right). Image adapted from [3].

hollow circular portion with 0.24-unit diameter, centered at (0.5, 0.5) is cut out from the disk to generate a hollow disk mesh. The spatial coordinates (x, y) and the hole diameter (hole_dia) of the hollow circular cutout are varied in the range [0.35, 0.65] and [0.1, 0.5] respectively.

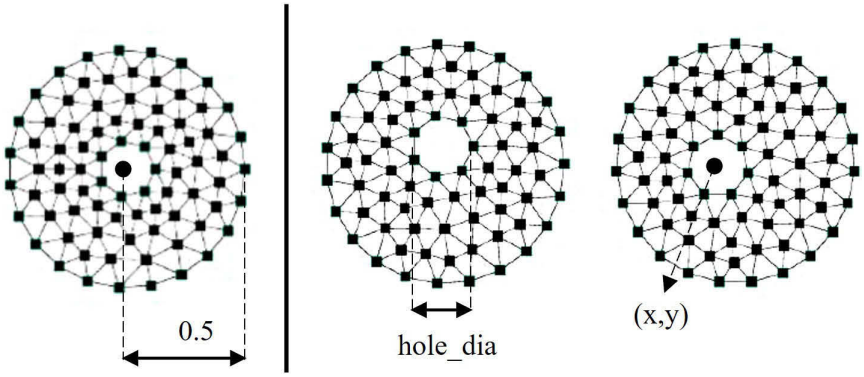


Figure 3.6: Hollow Disk mesh (left); Augmented disk mesh with varying cutout locations (right). Image adapted from [3].

In the context of electrostatics problems, the charge distribution ranges from 1-3 circular, randomly distributed positive charges[9] (with constant permittivity ε), generated for each instance in the dataset. In the case of magnetosatics, the dataset consists of meshes with current distribution, ranging from 1-3 randomly distributed electric currents[10] (with constant permeability μ). Both these datasets were generated with (Group 1) and without *mesh augmentation(s)* (Group 2). "Mesh augmentation" (mesh_aug) essentially refers to the technique of introducing geometrical variations in the meshes for training purposes. The dataset also contains meshes with higher charge (current) inhomogeneities such as 4 or 5 electric charges/electric currents for electrostatics(magnetostatics) problems without mesh augmentation (Group 3).

Alongside learning the PDE solution(s) on 2D meshes, we aim to evaluate the capabilities of the GNN model through the following tasks [3]:

- *Task 1*—To train and test across diverse mesh shapes and predict the physical quantities on unobserved geometries during the training phase such as a U-shaped mesh (task_1);
- *Task 2*—To learn and approximate PDE solution(s) for varied levels of inhomogeneities such as higher current/charge distributions introduced in the meshes (task_2).

In order to perform these tasks effectively, the generated dataset is divided into three subsets namely - training, validation and test datasets. There exists 2500 samples corresponding to each mesh implying, $2500 * 5 = 12500$ samples per group described above. Thus, the total number of dataset instances across all the three groups sums up to $12500 * 3 = 37500$. Details pertinent to this dataset partitioning is summarized in Table 3.1.

[9] Stationary electric charges.
[10] Steady electric currents.

Table 3.1: Training, Validation and Test datasets.

Dataset Type	Characteristics	Mesh Geometries
Training	80% samples from Group 1: 8000 samples (mesh_aug); 80% samples from Group 2: 8000 samples (without mesh_aug)	Disk, Hollow disk, Square and L-mesh.
Validation	20% samples from Group 1: 2000 samples (mesh_aug); 20% samples from Group 2: 2000 samples (without mesh_aug)	Disk, Hollow disk, Square and L-mesh.
Test	task_1: 20% samples (=2500) from Group 2 (only 1 shape); task_2: 25% samples (=2500) from Group 3 (excluding U-mesh)	task_1: U-mesh; task_2: Disk, Hollow disk, Square and L-mesh.

3.5.2 Training & Testing GNN models

Based on the structural analogy between a mesh and the graph, we can easily recast a triangulated mesh onto a graph comprising a finite set of nodes and edges. As a data preprocessing step, the Delaunay triangulated mesh (generated using FEniCS) is characterized by a set of input and output mesh features $\mathbf{x}$. Further, $\mathbf{x}$ is made up of node attributes $\mathbf{n}$ and edge attributes $\boldsymbol{\mathcal{E}}$, whose parameters vary in accordance with the PDE domain chosen. For electrostatics problems, the node attributes consist of a boundary condition value $bc_i \in 0, 1$ denoting the position of node n_i (either inside the mesh or at the boundary), distance from n_i to the nearest boundary node (dx, dy), and a charge inhomogeneity (essentially, the charge density ρ_v). On the other hand, $\mathcal{E}_{ij}$ denotes the relative distance(s) between nodes i, j, constituting the edge attributes. The associated output features (obtained through a FEM solver) refers to the electric potential (U) and the time-independent electric field $\mathbf{E} = (E_x, E_y)$. Similarly, magnetostatics problems require as node attributes:

$\mathcal{E}_{ij}, bc_i$, current inhomogeneity (current density $\mathbf{J}$, essentially J_z) and distances (dx, dy). Its output feature set includes the magnetic potential A_z and the magnetic field $\mathbf{H} = (H_x, H_y)$. For convenience, the training data is normalized to values in the range $[-1, 1]$, yielding unitless quantities.

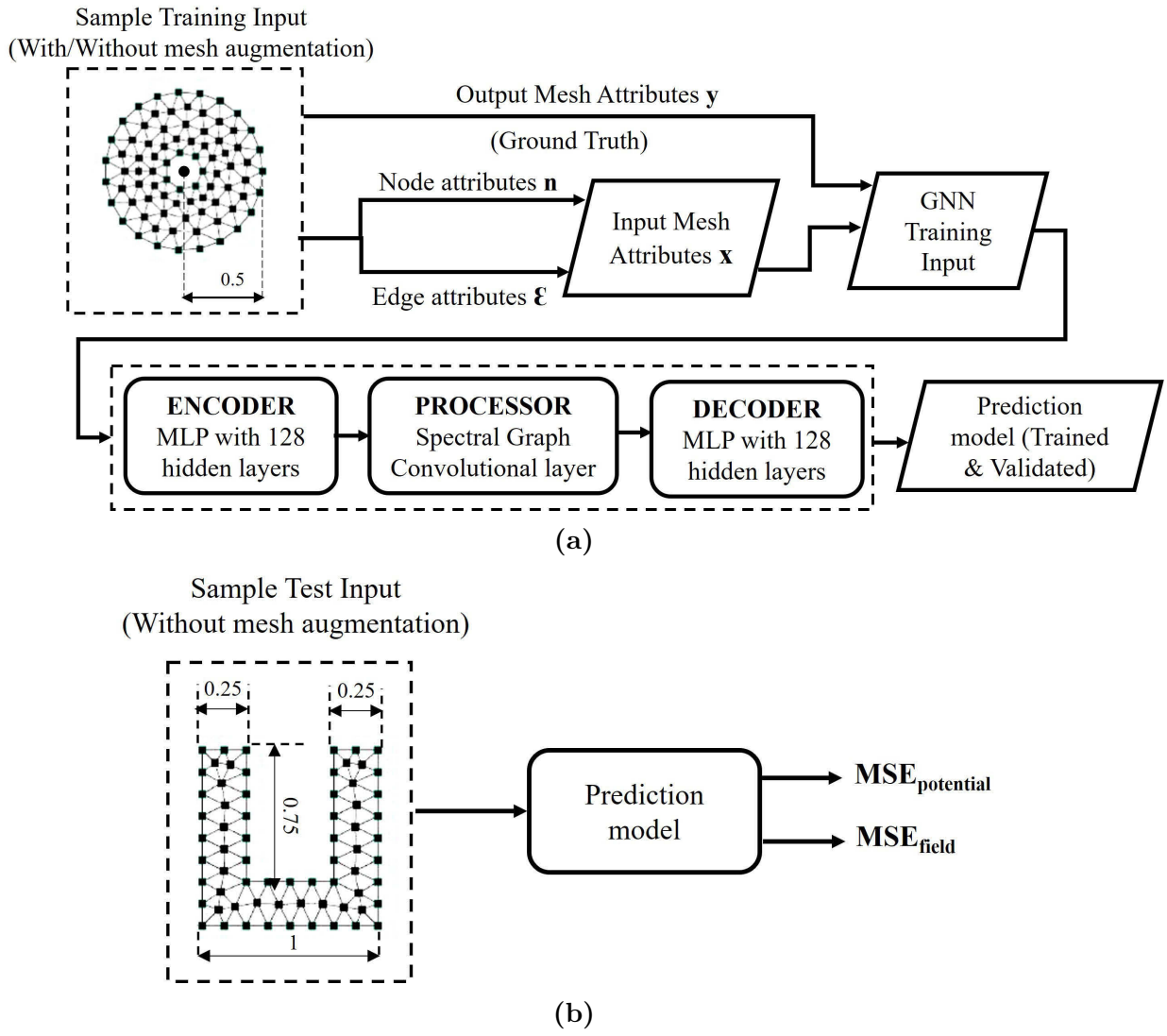


Figure 3.7: An overview of the (a) GNN model (Section 3.4) training and (b) testing process. Mesh images adapted from [3].

These input and output attributes are fed to the GNN model (described in Section 3.4.1) as (training/validation/test) inputs. Following the 128-hidden layered encoder MLP, the processor layer with 128-dimensional hidden attributes is configured with $K = 5$ graph

convolutional hops. The model employs 3 graph convolutions consecutively, interleaved with a nonlinear ReLU activation after every step. Its parametric outputs are passed onto a decoder MLP, after which we obtain the prediction model. The entire GNN model, developed using PyTorch based frameworks - PyTorch Lightning [117] and PyG (PyTorch Geometric) [118], learns nearly 0.28 million parameters with stepsize $\alpha = 0.001$ and exponential decay rates of the rolling average (moment estimates) $\beta_1 = 0.9, \beta_2 = 0.999$. The learning rate α and the betas are hyperparameters offered by the Adaptive Moment Estimates (Adam) optimization algorithm [119], used here to optimize the network parameters. Each PDE task (Table 3.4 was trained for 150 epochs[11] with batch size=32, followed by a validation after every epoch. Following the training phase, the prediction model with the lowest validation loss was chosen for testing purposes.

3.5.2.1 GPU Training

In view of accelerating the deep learning processes, the GNN models were trained for both electrostatics and magnetostatics problems on a heterogenous supercomputer cluster[12], using a 4× NVIDIA P100 Pascal with a 16GB high bandwidth random-access memory. A virtual environment was manually setup with relevant Python-based library installations for parallelization on GPU [120].

Training, Testing and Logging: Each task was submitted as a "SLURM job" (using a job script) on the GPU cluster. The job script enlisted parameters such as the GPU-CPU configuration, memory requirements and was submitted using the `sbatch` command. Further, Weights & Biases (WandB) [121] , an AI cloud platform was used to log both training and test runs (offline/online mode), track experiments and analyze their results. Each training task generated multiple versions of prediction models (WandB artifact) with a unique

[11] Epoch refers to the number of times the model iterates through the entire dataset.
[12] Digital Research Alliance of Canada: Cedar cluster.

identifier. Each artifact marked as "best_k" (lowest validation loss) on the WandB cloud server had to be manually passed as an argument to the `srun` test command in the SLURM job script (for testing purposes).

Reproducibility: For better representation and enhanced reproducibility of the results, each experiment was conducted and averaged over 5 random seeds. Performance comparison of mesh augmentation with other regularization techniques for each PDE task yields $5 * 4 = 20$ model versions per seed, and 100 versions in total for 5 random seeds (Table 3.4). Thus, each seed with 20 models were submitted as a "job array". Bash scripts were developed to automate such manual preprocessing steps, additionally involving WandB artifact(s) download (from the cloud server) corresponding to each of the 20 models, preparing a text file with task names (input for job arrays) and WandB logging (training and test runs) through a bash command line interface in a Visual Studio Integrated Development Environment (IDE).

3.5.3 Results and Discussion

Results presented in this work are unitless and use the mean squared error metric over all the predicted potential and field quantities (with respect to the ground truth), across all learning tasks. As indicated by the results reported in Table 3.2 and 3.3, mesh augmentation, in comparison to the baseline model (without augmented data), relatively enhances the prediction quality over unseen data (U-mesh for task_1), and higher current/charge inhomogeneities (task_2). Specifically, we observe a substantial improvement upto a factor of 10 (order of magnitude) for potentials (ϕ, A_z).

Further, this analysis is extended with a comparison over conventional regularization (dropout) approaches. Referring to Table 3.4, it enlists a comprehensive set of results generated by training and testing the same GNN framework (discussed in Section 3.4) for all PDE tasks, with/without mesh augmentation (mesh_aug/mesh_no_aug), dropout of nodes

Table 3.2: Test results for Electrostatics problems with/without mesh augmentation (Mesh_Aug); (Values averaged over 5 random seeds).

PDE_task	**Mesh_Aug**	**MSE$_{\text{elec_potential}}$**	**MSE$_{\text{elec_field}}$**
task_1	**Yes**	**1.4707E-05**	**2.2011E-03**
task_1	No	1.3317E-04	2.5311E-03
task_2	**Yes**	**1.6095E-04**	**4.3788E-04**
task_2	No	2.7048E-04	7.8656E-04

Table 3.3: Test results for Magnetostatics problems with/without mesh augmentation (Mesh_Aug); (Values averaged over 5 random seeds).

PDE_task	**Mesh_Aug**	**MSE$_{\text{mag_potential}}$**	**MSE$_{\text{mag_field}}$**
task_1	**Yes**	**1.3629E-05**	**1.772E-03**
task_1	No	1.2402E-04	2.4251E-03
task_2	**Yes**	**1.7529E-04**	**4.6735E-04**
task_2	No	2.3381E-04	6.0364E-04

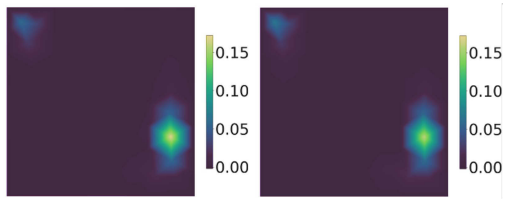
($p = 0.1$), edges ($p = 0.2$) and embeddings (node features, $p = 0.2$) respectively. Here, p refers to the dropout probability. The "drop_nodes" and "drop_edges" approaches do augment the input graph topology by randomly dropping their nodes or edges, thereby learning an effective input-output mapping and enhancing the model's approximation quality on unseen samples. The "embedding_dropout" method emerges with the highest mean squared error over all PDE tasks. With the given dataset and the adapted GNN implementation, mesh augmentation seem to considerably boost the model's prediction quality with respect to the potential and field values, in both electrostatics and magnetostatics problems. Besides the enhanced solution learning ability through geometric transformations in the training

Table 3.4: Mesh Augmentation vs conventional regularization techniques (test results averaged over 5 random seeds).

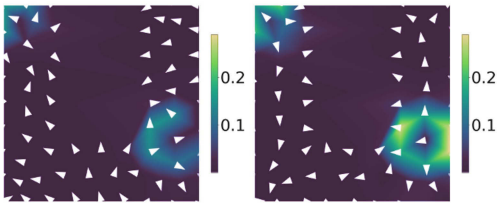
PDE_task	Regularizer	$\mathbf{MSE_{potential}}$	$\mathbf{MSE_{field}}$
Electrostatics task_1	**mesh_aug**	**1.4708E-05**	**2.2012E-03**
	mesh_no_aug	1.3317E-04	2.5311E-03
	drop_nodes	5.9198E-05	2.6705E-03
	drop_edges	1.0155E-04	3.1018E-03
	embedding_dropout	1.1272E-03	3.8800E-03
Electrostatics task_2	**mesh_aug**	**1.6095E-04**	**4.3788E-04**
	mesh_no_aug	2.7048E-04	7.8656E-04
	drop_nodes	1.2031E-03	9.5001E-04
	drop_edges	5.6847E-04	6.8198E-04
	embedding_dropout	1.5451E-03	1.5555E-03
Magnetostatics task_1	**mesh_aug**	**1.3629E-05**	**1.7720E-03**
	mesh_no_aug	1.2402E-04	2.4251E-03
	drop_nodes	6.1227E-05	2.0600E-03
	drop_edges	6.9059E-05	2.5726E-03
	embedding_dropout	1.4316E-03	3.6265E-03
Magnetostatics task_2	**mesh_aug**	**1.7529E-04**	**4.6735E-04**
	mesh_no_aug	2.3381E-04	6.0364E-04
	drop_nodes	9.1714E-04	6.6237E-04
	drop_edges	5.1015E-04	5.2734E-04
	embedding_dropout	1.7525E-03	1.2243E-03

data, the prediction behaviour can also be attributed to the underlying spectral graph convolutional processor layer (Section 3.4.1). Its graph pooling strategy with fast, K-localized, polynomial-based convolutional kernels (filters) enable node-wise feature learning, with edge weights passed on to consecutive layers. In addition, it is interesting to note that, the aggregation behaviour demonstrated by these kernels, further enable the model to predict the resulting potential and field, in the presence of multiple charges/currents on various geome-

tries. Results organized in Table 3.4 are comparable to the prediction error estimates (MSEs) reported in Table 8 of [3]. Sample visualizations for both task_1 and task_2 are shown in Figures 3.8 - 3.11, wherein the left column denotes the set of predicted values (magnitude of potentials and field; arrows denote the orientation of the electric(magnetic field)), and the right column denotes the ground truth.

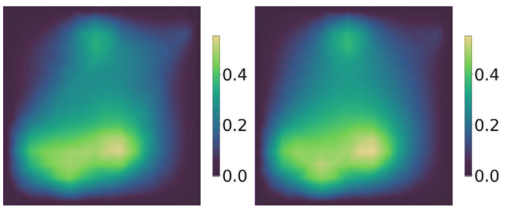


(a)

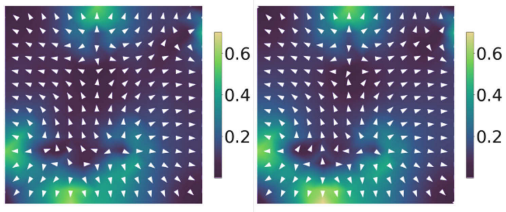


(b)

Figure 3.8: Sample test predictions (vs ground truth) for Electrostatics task_1;(a) Electric potential; (b) Electric field.



(a)



(b)

Figure 3.9: Sample test predictions (vs ground truth) for Electrostatics task_2;(a) Electric potential; (b) Electric field.

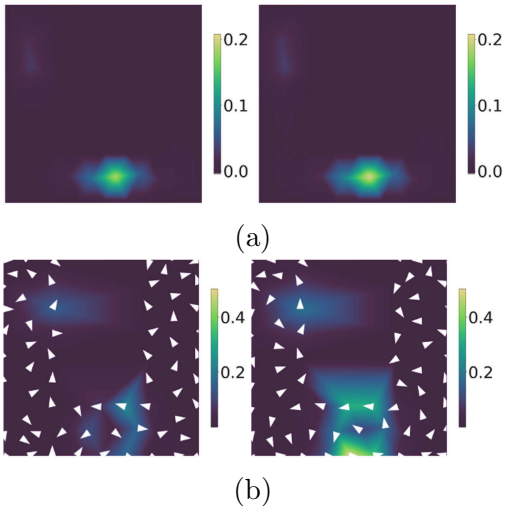


Figure 3.10: Sample test predictions (vs ground truth) for Magnetostatics task_1;(a) Magnetic potential; (b) Magnetic field.

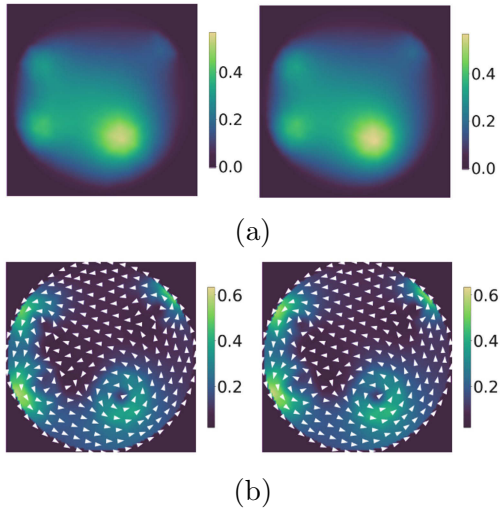


Figure 3.11: Sample test predictions (vs ground truth) for Magnetostatics task_2;(a) Magnetic potential; (b) Magnetic field.

3.5.3.1 Limitations and Recommendations

In comparison to the predictions obtained for task_2 (Table 3.4), we find the mean squared errors on field values, majorly in the order of 10^{-4}. The magnitude of MSE for predicted electric and magnetic fields on U-shaped meshes (task_1) are in the order of 10^{-3}. The gap in the performance possibly arises due to random initialization of weights, method of sampling instances for dataset split, the dependency of model performance on random

seeds for reproducibility. With regards to the model, the number of hops set to $K = 5$, may have limited the node-wise feature learning in a relatively wider neighbourhood, which implicitly refers to detecting new mesh structural boundaries and making predictions.

However, such errors affecting the model's prediction performance can possibly be accommodated for or minimized by considering the following approaches:

- Previously, it was observed that incorporating mesh augmentation introduces diversity in the training data and brings along a certain level of regularization in the deep neural network model, thereby enhancing prediction quality (electric/magnetic potential and field values) on newer samples. As an extension, one may use mesh augmentation in conjunction with commonly used dropout techniques such as dropping of nodes or edges with a dropout probability p, during the training phase. This will introduce an additional effect of regularization in the neural network, and may prove beneficial as the random dropping of graph entities (nodes/edges) minimizes their interdependency across the network layers (co-adaptation), and aid in learning more useful features [122]. Exploring this combination of training data diversity and efficient feature learning brought in by mesh augmentation and dropout techniques respectively, may yield better predictions over unobserved samples (newer geometry or higher charge/current distributions).

- In view of learning effective node-wise features in a wider graph neighbourhood, one may adopt a self-attention based network (discussed in Section 2.3.2.3) for the processor layer, replacing the spectral graph convolutional processor layer in the GNN model. Considering nodes in the same neighbourhood but not connected by an edge, may provide insights on key features in the input mesh, during the learning process. This can be achieved using a Multi-hop Attention Graph Neural Network (MAGNA) proposed in [123]. By assigning and propagating the attention scores through the deep neural network, it widens the receptive field (graph node neighbourhood), thus efficiently

accounting for all possible paths between a pair of nodes with no edges/connections. Thus, for a pair of nodes that are many hops away from each other, such a wide receptive field may enable better potential and field predictions on a given mesh geometry.

Chapter 4

Conclusions

4.1 Remarks & Key Findings

This thesis presented a comprehensive survey of GNN approaches proposed in the literature to solve PDEs for electromagnetics and other physics-based problems. A GNN model with spectral graph convolution was implemented to predict the electric, magnetic potential and field values closer to the ground truth, for time-independent Poisson's equations in electrostatics and magnetostatics. The model was trained and tested on datasets comprising 2D meshes with multiple geometries and current/charge distributions. The prediction quality of the GNN model with and without mesh augmentation were analyzed and compared with conventional regularization methods.

The results show that mesh augmentation incorporated in the training phase, improves GNN predictions over previously unseen samples, in comparison to the model trained with other regularizers and unaugmented meshes. It is observed that the mean squared errors in the case of predicted potentials are comparatively lower than that of fields, for U-shaped meshes. This performance gap can be addressed with adopting a combination of regularizers such as mesh augmentation with dropout of node/edges during the training phase, or a self-attention based network processor layer in the GNN model.

4.2 Future work

Directions for future work include:

- Training a different GNN model, such as a Graph Nets based architecture – MeshGraphNets [5] using the datasets described in Section 3.5.1;
- Implementing the GNN model discussed in Section 3.4 to estimate solutions for Poisson's equations defined under different boundary conditions (Neumann, mixed);
- Incorporating adaptive mesh refinement in the existing GNN implementation;
- Using physics-informed neural networks to estimate electric, magnetic fields and potentials for static PDEs.

Bibliography

[1] Wing Liu, Shaofan Li, and Harold Park. "Correction to: Eighty Years of the Finite Element Method: Birth, Evolution, and Future". In: *Archives of Computational Methods in Engineering* 30 (June 2022), pp. 1–1. DOI: 10.1007/s11831-022-09784-x.

[2] William L. Hamilton. "Graph Representation Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.3 (), pp. 1–159.

[3] Winfried Lötzsch, Simon Ohler, and Johannes S. Otterbach. *Learning the Solution Operator of Boundary Value Problems using Graph Neural Networks*. 2023. arXiv: 2206.14092 [cs.LG].

[4] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. "Learning to Simulate Complex Physics with Graph Networks". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8459–8468.

[5] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. *Learning Mesh-Based Simulation with Graph Networks*. 2021. arXiv: 2010.03409 [cs.LG].

[6] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. "Neural operator: Graph ker-

nel network for partial differential equations". In: *arXiv preprint arXiv:2003.03485* (2020).

[7] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. "Neural operator: Learning maps between function spaces". In: *arXiv preprint arXiv:2108.08481* (2021).

[8] Joseph Pateras, Pratip Rana, and Preetam Ghosh. "A Taxonomic Survey of Physics-Informed Machine Learning". In: *Applied Sciences* 13.12 (2023). ISSN: 2076-3417. DOI: 10.3390/app13126892.

[9] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. *Regularization for Deep Learning: A Taxonomy*. 2017. arXiv: 1710.10686 [cs.LG].

[10] Winfried Lötzsch, Simon Ohler, and Johannes S Otterbach. "Learning the solution operator of boundary value problems using graph neural networks". In: *arXiv preprint arXiv:2206.14092* (2022).

[11] R. Haberman. *Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems*. Featured Titles for Partial Differential Equations. Pearson, 2013. ISBN: 9780321797056.

[12] J.D. Hoffman and S. Frankel. *Numerical Methods for Engineers and Scientists*. CRC Press, 2018. ISBN: 9781482270600.

[13] J.A. Buck and W.H. Hayt. *Engineering Electromagnetics*. McGraw-Hill Education, 2011. ISBN: 9780073380667.

[14] A. Hrennikoff. "Solution of Problems of Elasticity by the Framework Method". In: *Journal of Applied Mechanics* 8.4 (Dec. 1941), A169–A175. DOI: 10.1115/1.4009129.

[15] Douglas McHenry. "A LATTICE ANALOGY FOR THE SOLUTION OF STRESS PROBLEMS." In: *Journal of the Institution of Civil Engineers* 21.2 (1943), pp. 59–82.

[16] R Courant. "Variational methods for the solution of problems of equilibrium and vibrations". In: *Bulletin of the American Mathematical Society* 49.1 (1943), pp. 1–23.

[17] John H Argyris. "Energy Theorems and Structural Analysis: A Generalized Discourse with Applications on Energy Principles of Structural Analysis Including the Effects of Temperature and Non-Linear Stress-Strain Relations". In: *Aircraft Engineering and Aerospace Technology* 26.10 (1954), pp. 347–356.

[18] M Jon Turner, Ray W Clough, Harold C Martin, and LJ Topp. "Stiffness and deflection analysis of complex structures". In: *Journal of the Aeronautical Sciences* 23.9 (1956), pp. 805–823.

[19] Olek C Zienkiewicz and Robert L Taylor. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 2005.

[20] Wikipedia contributors. *Artificial intelligence — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Artificial_intelligence&oldid=1181810241`. 2023.

[21] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 9780387848846.

[22] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. *Semi-Supervised Learning with Deep Generative Models*. 2014. arXiv: `1406.5298 [cs.LG]`.

[23] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN: 9780262352703.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[25] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[27] Ali Girayhan Özbay, Arash Hamzehloo, Sylvain Laizet, Panagiotis Tzirakis, Georgios Rizos, and Björn Schuller. "Poisson CNN: Convolutional neural networks for the solution of the Poisson equation on a Cartesian mesh". In: *Data-Centric Engineering* 2 (2021), e6.

[28] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478 [cs.LG].

[29] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418.

[30] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010, pp. 253–256. DOI: 10.1109/ISCAS.2010.5537907.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[32] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. *Interaction Networks for Learning about Objects, Relations and Physics*. 2016. arXiv: 1612.00222 [cs.AI].

[33] Federico Monti, Michael M. Bronstein, and Xavier Bresson. *Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks*. 2017. arXiv: 1704.06803 [cs.LG].

[34] M. Gori, G. Monfardini, and F. Scarselli. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. 2005, 729–734 vol. 2. DOI: 10.1109/IJCNN.2005.1555942.

[35] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[36] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG].

[37] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew M. Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[38] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. 2017. arXiv: 1606.09375 [cs.LG].

[39] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG].

[40] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr. au2, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. *Simplifying Graph Convolutional Networks*. 2019. arXiv: 1902.07153 [cs.LG].

[41] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[42] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications". In: *AI open* 1 (2020), pp. 57–81.

[43] Deyu Bo, Xiao Wang, Yang Liu, Yuan Fang, Yawen Li, and Chuan Shi. *A Survey on Spectral Graph Neural Networks*. 2023. arXiv: 2302.05631 [cs.LG].

[44] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. *GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs*. 2018. arXiv: 1803.07294 [cs.LG].

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML].

[46] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. *Geometric deep learning on graphs and manifolds using mixture model CNNs*. 2016. arXiv: 1611.08402 [cs.CV].

[47] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks*. 2021. arXiv: 1810.02244 [cs.LG].

[48] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. *Computing Graph Neural Networks: A Survey from Algorithms to Accelerators*. 2021. arXiv: 2010.00130 [cs.LG].

[49] Ryoma Sato. *A Survey on The Expressive Power of Graph Neural Networks*. 2020. arXiv: 2003.04078 [cs.LG].

[50] Lilapati Waikhom and Ripon Patgiri. "A survey of graph neural networks in various learning paradigms: methods, applications, and challenges". In: *Artificial Intelligence Review* 56.7 (2023), pp. 6295–6364.

[51] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[52] Ziwei Zhang, Peng Cui, and Wenwu Zhu. "Deep Learning on Graphs: A Survey". In: *IEEE Trans. on Knowl. and Data Eng.* 34.1 (2022), 249–270. ISSN: 1041-4347. DOI: 10.1109/TKDE.2020.2981333.

[53] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations". In: *IEEE transactions on neural networks* 9.5 (1998), pp. 987–1000.

[54] M Chiaramonte, M Kiener, et al. "Solving differential equations using neural networks". In: *Machine Learning Project* 1 (2013).

[55] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. "Solving parametric PDE problems with artificial neural networks". In: *European Journal of Applied Mathematics* 32.3 (2021), pp. 421–435.

[56] Zhiqiang Cai, Jingshuang Chen, Min Liu, and Xinyu Liu. "Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs". In: *Journal of Computational Physics* 420 (2020), p. 109707.

[57] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. "NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models". In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 1998, 9–20. ISBN: 0897919998. DOI: 10.1145/280814.280816.

[58] Lubor Ladicky, Sohyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus H. Gross. "Data-driven fluid simulations using regression forests". In: *ACM Transactions on Graphics (TOG)* 34 (2015), pp. 1 –9.

[59] Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. *Newtonian Image Understanding: Unfolding the Dynamics of Objects in Static Images*. 2015. arXiv: 1511.04048 [cs.CV].

[60] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. "Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[61] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. *PDE-Net: Learning PDEs from Data*. 2018. arXiv: 1710.09668 [math.NA].

[62] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. "MeshCNN". In: *ACM Transactions on Graphics* 38.4 (2019), pp. 1–12. DOI: 10.1145/3306346.3322959.

[63] Riya Aggarwal and Hassan Ugail. "On the Solution of Poisson's Equation using Deep Learning". In: *2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. 2019, pp. 1–8. DOI: 10.1109/SKIMA47702.2019.8982518.

[64] Lionel Cheng, Ekhi Ajuria Illarramendi, Guillaume Bogopolsky, Michael Bauerheim, and Benedicte Cuenot. *Using neural networks to solve the 2D Poisson equation for electric field computation in plasma fluid simulations*. 2021. arXiv: 2109.13076 [cs.LG].

[65] Zhongyang Zhang, Ling Zhang, Ze Sun, Nicholas Erickson, Ryan From, and Jun Fan. *Solving Poisson's Equation using Deep Learning in Particle Simulation of PN Junction*. 2018. arXiv: 1810.10192 [physics.comp-ph].

[66] Han Gao, Luning Sun, and Jian-Xun Wang. "PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain". In: *Journal of Computational Physics* 428 (2021), p. 110079.

[67] Frank Rosenblatt and Cornell Aeronautical Lab Inc Buffalo NY. "PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS". In: (1961).

[68] Yedid Hoshen. "Vain: Attentional multi-agent predictive modeling". In: *Advances in neural information processing systems* 30 (2017).

[69] Michael B. Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. *A Compositional Object-Based Approach to Learning Physical Dynamics*. 2017. arXiv: `1612.00341 [cs.AI]`.

[70] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B. Tenenbaum, and Daniel L. K. Yamins. *Flexible Neural Representation for Physics Prediction*. 2018. arXiv: `1806.08047 [cs.AI]`.

[71] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. *Graph networks as learnable physics engines for inference and control*. 2018. arXiv: `1806.01242 [cs.LG]`.

[72] Brian R. Bartoldson, Yeping Hu, Amar Saini, Jose Cadena, Yucheng Fu, Jie Bao, Zhijie Xu, Brenda Ng, and Phan Nguyen. *Scientific Computing Algorithms to Learn Enhanced Scalable Surrogates for Mesh Physics*. 2023. arXiv: `2304.00338 [cs.LG]`.

[73] Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. "Graph element networks: adaptive, structured computation and memory". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 212–222.

[74] He Wang and Juyong Zhang. "A survey of deep learning-based mesh processing". In: *Communications in Mathematics and Statistics* 10.1 (2022), pp. 163–194.

[75] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Liò. *Simplicial Attention Networks*. 2022. arXiv: 2204.09455 [cs.LG].

[76] Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. *MultiScale MeshGraphNets*. 2022. arXiv: 2210.00612 [cs.LG].

[77] Wenzhuo Liu, Mouadh Yagoubi, and Marc Schoenauer. "Multi-resolution graph neural networks for pde approximation". In: *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part III 30*. Springer. 2021, pp. 151–163.

[78] Christian Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benno Kuckuck. "An overview on deep learning-based approximation methods for partial differential equations". In: *arXiv preprint arXiv:2012.12348* (2020).

[79] Harender Kumar and Neha Yadav. "Deep learning algorithms for solving differential equations: a survey". In: *Journal of Experimental & Theoretical Artificial Intelligence* (2023), pp. 1–40.

[80] Derick Nganyu Tanyu, Jianfeng Ning, Tom Freudenberg, Nick Heilenkötter, Andreas Rademacher, Uwe Iben, and Peter Maass. "Deep learning methods for partial differential equations and related parameter identification problems". In: *Inverse Problems* 39.10 (2023), p. 103001.

[81] Johannes Brandstetter, Daniel Worrall, and Max Welling. "Message passing neural PDE solvers". In: *arXiv preprint arXiv:2202.03376* (2022).

[82] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. *Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations*. 2017. arXiv: 1711.10561 [cs.AI].

[83] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. *Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations*. 2017. arXiv: 1711.10566 [cs.AI].

[84] Maziar Raissi. *Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations*. 2018. arXiv: 1801.06637 [stat.ML].

[85] Lucie P Aarts and Peter Van Der Veer. "Neural network method for solving partial differential equations". In: *Neural Processing Letters* 14 (2001), pp. 261–271.

[86] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. "Lagrangian fluid simulation with continuous convolutions". In: *International Conference on Learning Representations*. 2019.

[87] Riya Aggarwal, Hassan Ugail, and Ravi Kumar Jha. "A deep artificial neural network architecture for mesh free solutions of nonlinear boundary value problems". In: *Applied Intelligence* 52.1 (2022), pp. 916–926.

[88] Diab W Abueidda, Qiyue Lu, and Seid Koric. "Meshless physics-informed deep learning method for three-dimensional solid mechanics". In: *International Journal for Numerical Methods in Engineering* 122.23 (2021), pp. 7182–7201.

[89] Senwei Liang, Shixiao W. Jiang, John Harlim, and Haizhao Yang. *Solving PDEs on Unknown Manifolds with Machine Learning*. 2022. arXiv: 2106.06682 [math.NA].

[90] Yongju Zheng, Huajie Dai, Junyi Wu, Chuanping Zhou, Zhiwen Wang, Rougang Zhou, and Wenxin Li. "Research progress and development trend of smart metamaterials". In: *Frontiers in Physics* 10 (2022), p. 1191.

[91] Andrea Massa, Davide Marcantonio, Xudong Chen, Maokun Li, and Marco Salucci. "DNNs as Applied to Electromagnetics, Antennas, and Propagation—A Review". In: *IEEE Antennas and Wireless Propagation Letters* 18.11 (2019), pp. 2225–2229. DOI: 10.1109/LAWP.2019.2916369.

[92] Rajendran Ramasamy and Maria Anto Bennet. "An Efficient Antenna Parameters Estimation Using Machine Learning Algorithms". In: *Progress In Electromagnetics Research C* 130 (2023), pp. 169–181.

[93] Danilo Erricolo, Pai-Yen Chen, Anastasiia Rozhkova, Elahehsadat Torabi, Hakan Bagci, Atif Shamim, and Xianglian Zhang. "Machine Learning in Electromagnetics: A Review and Some Perspectives for Future Research". In: *2019 International Conference on Electromagnetics in Advanced Applications (ICEAA)*. 2019, pp. 1377–1380. DOI: 10.1109/ICEAA.2019.8879110.

[94] S.D. Campbell and D.H. Werner. *Advances in Electromagnetics Empowered by Artificial Intelligence and Deep Learning*. IEEE Press Series on Electromagnetic Wave Theory. Wiley, 2023. ISBN: 9781119853893.

[95] P. Ramuhalli, L. Udpa, and S.S. Udpa. "Finite-element neural networks for solving differential equations". In: *IEEE Transactions on Neural Networks* 16.6 (2005), pp. 1381–1392. DOI: 10.1109/TNN.2005.857945.

[96] Arbaaz Khan, Vahid Ghorbanian, and David Lowther. "Deep Learning for Magnetic Field Estimation". In: *IEEE Transactions on Magnetics* 55.6 (2019), pp. 1–4. DOI: 10.1109/TMAG.2019.2899304.

[97] Wei Tang, Tao Shan, Xunwang Dang, Maokun Li, Fan Yang, Shenheng Xu, and Ji Wu. "Study on a Poisson's equation solver based on deep learning technique". In: *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*. 2017, pp. 1–3. DOI: 10.1109/EDAPS.2017.8277017.

[98] Zijie Li, Kazem Meidani, and Amir Barati Farimani. *Transformer for Partial Differential Equations' Operator Learning*. 2023. arXiv: 2205.13671 [cs.LG].

[99] Arbaaz Khan and David A. Lowther. "Physics Informed Neural Networks for Electromagnetic Analysis". In: *IEEE Transactions on Magnetics* 58.9 (2022), pp. 1–4. DOI: 10.1109/TMAG.2022.3161814.

[100] Mohammad Mushfiqur Rahman, Arbaaz Khan, David Lowther, and Dennis Giannacopoulos. "Evaluating magnetic fields using deep learning". In: *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering* (2023).

[101] Zhi Gong, Yang Chu, and Shiyou Yang. "Physics-Informed Neural Networks for Solving Two-Dimensional Magnetostatic Fields". In: *IEEE Transactions on Magnetics* (2023), pp. 1–1. DOI: 10.1109/TMAG.2023.3281863.

[102] Andrés Beltrán-Pulido, Ilias Bilionis, and Dionysios Aliprantis. "Physics-Informed Neural Networks for Solving Parametric Magnetostatic Problems". In: *IEEE Transactions on Energy Conversion* 37.4 (2022), pp. 2678–2689. DOI: 10.1109/TEC.2022.3180295.

[103] Marco Baldan, Paolo Di Barba, and David A. Lowther. "Physics-Informed Neural Networks for Inverse Electromagnetic Problems". In: *IEEE Transactions on Magnetics* 59.5 (2023), pp. 1–5. DOI: 10.1109/TMAG.2023.3247023.

[104] Yanjin Chen, Hongrui Zhang, Tie Jun Cui, Fernando L. Teixeira, and Lianlin Li. "A Mesh-Free 3-D Deep Learning Electromagnetic Inversion Method Based on Point Clouds". In: *IEEE Transactions on Microwave Theory and Techniques* 71.8 (2023), pp. 3530–3539. DOI: 10.1109/TMTT.2023.3248174.

[105] Zeyang Wu, Yuexing Peng, Peng Wang, Wenbo Wang, and Wei Xiang. "A Physics-Induced Deep Learning Scheme for Electromagnetic Inverse Scattering". In: *IEEE Transactions on Microwave Theory and Techniques* (2023), pp. 1–21. DOI: 10.1109/TMTT.2023.3300185.

[106] Winfried Lötzsch, Simon Ohler, and Johannes S Otterbach. *gnn-bvp-solver*. GitHub Source Code Repository. 2022. URL: https://github.com/merantix-momentum/gnn-bvp-solver.

[107] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Dover Books on Mathematics Series. Dover Publications, Incorporated, 2012. ISBN: 9780486131597.

[108] Hans Petter Langtangen and Kent-Andre Mardal. *Introduction to numerical methods for variational problems*. Vol. 21. Springer Nature, 2019.

[109] Masayuki Yano. *AER1418: Variational Methods for PDEs Lecture Notes*. 2018-2022.

[110] Benyam Mebrate, Purnachandra Rao Koya, et al. "Numerical solution of a two dimensional poisson equation with dirichlet boundary conditions". In: *American Journal of Applied Mathematics* 3.6 (2015), pp. 297–304.

[111] J.D. Jackson. *Classical Electrodynamics*. Wiley, 1962. ISBN: 9780471431312.

[112] Kunihiko Fukushima. "Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements". In: *IEEE Transactions on Systems Science and Cybernetics* 5.4 (1969), pp. 322–333. DOI: 10.1109/TSSC.1969.300225.

[113] M. W. Scroggs, J. S. Dokken, C. N. Richardson, and G. N. Wells. "Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes". In: *ACM Transactions on Mathematical Software* 48.2 (2022), 18:1–18:23. DOI: 10.1145/3524456. URL: https://fenicsproject.org/.

[114] M. S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. "The FEniCS Project Version 1.5". In: *Archive of Numerical Software* 3 (2015). DOI: 10.11588/ans.2015.100.20553.

[115] A. Logg, K.-A. Mardal, G. N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. DOI: 10.1007/978-3-642-23099-8.

[116] A. Logg and G. N. Wells. "DOLFIN: Automated Finite Element Computing". In: *ACM Transactions on Mathematical Software* 37 (2010). DOI: 10.1145/1731022.1731030.

[117] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. DOI: 10.5281/zenodo.3828935. URL: https://github.com/Lightning-AI/lightning.

[118] Matthias Fey and Jan Eric Lenssen. *Fast Graph Representation Learning with PyTorch Geometric*. 2019. arXiv: 1903.02428 [cs.LG].

[119] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[120] Compute Canada. *Python*. 2023. URL: https://docs.alliancecan.ca/wiki/Python.

[121] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[122] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[123] Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. *Multi-hop Attention Graph Neural Network*. 2021. arXiv: 2009.14332 [cs.LG].

Copyright

Following are the copies of relevant copyright permissions acquired from original author(s) through email:

Re-use of copyrighted images

Winfried Ripken <winfried.ripken@gmail.com>
Wed 10/11/2023 1:59 AM

To: Aishwarya Ramamurthy <aishwarya.ramamurthy@mail.mcgill.ca>

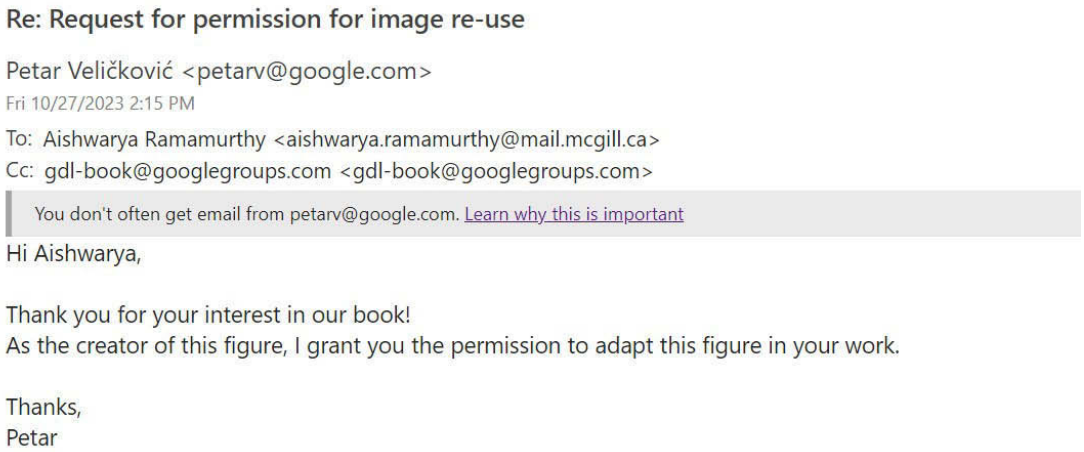
You don't often get email from winfried.ripken@gmail.com. Learn why this is important

I am the author of the paper "learning the solution operator of boundary value problems using Graph neural networks". Published at the 2nd ai for science workshop at ICML 2022.

I hereby grant Aishwarya the permission to re-use images and other content of the paper and the associated talk, as long as each instance of re-use is properly labeled as such and a reference to the paper is added directly next to it.

Best
Winfried

Figure C.1: Copyright permission for adapting figures, dataset and GNN model from [3],[106].



Re: Request for permission for image re-use

Petar Veličković <petarv@google.com>
Fri 10/27/2023 2:15 PM

To: Aishwarya Ramamurthy <aishwarya.ramamurthy@mail.mcgill.ca>
Cc: gdl-book@googlegroups.com <gdl-book@googlegroups.com>

You don't often get email from petarv@google.com. Learn why this is important

Hi Aishwarya,

Thank you for your interest in our book!
As the creator of this figure, I grant you the permission to adapt this figure in your work.

Thanks,
Petar

Figure C.2: Copyright permission to adapt Figure 17 from [28].