Modeling Multivariate Time Series with Spatio-Temporal Simplicial Complexes

MODELING MULTIVARIATE TIME SERIES WITH SPATIO-TEMPORAL SIMPLICIAL COMPLEXES

Thesis

to obtain the degree of Master in Computer Science with Specialization in Artificial Intelligence at Delft University of Technology,
to be publicly defended on Friday, August 29th 2025 at 14:00.

by

Adit Shankar WHORRA
born in New Delhi, India.

Multimedia Computing Group,
Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS),
Delft University of Technology,
Delft, the Netherlands.

Thesis committee:

Chair and Advisor:	Dr. E. Isufi, Faculty EEMCS, TU Delft
Committee Member:	Dr. G. Leus, Faculty EEMCS, TU Delft
Advisor:	M. Sabaqqi, Faculty, EEMCS, TU Delft

Keywords: Spatio-Temporal Modeling · Simplicial Neural Networks · Topological Signal Processing · Multivariate Time Series · Product Spaces

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

Acknowledgments iii

Summary iv

List of Abbreviations v

List of Notations vii

1 Introduction 1
- 1.1 Multivariate Time Series . . . 1
- 1.2 Modeling Paradigms . . . 2
- 1.3 Research Questions . . . 3
- 1.4 Thesis Contributions . . . 4

2 Background 5
- 2.1 Graph Signal Processing . . . 5
- 2.2 Topological Signal Processing . . . 11
- 2.3 Spatio-Temporal Modeling using Product Spaces . . . 21
- 2.4 Conclusion . . . 26

3 Related Work 27
- 3.1 Higher-Order Spatial Modeling . . . 27
- 3.2 Structure-Unaware Frameworks . . . 29
- 3.3 Graph-Aware Frameworks . . . 31
- 3.4 Topology-Aware Frameworks . . . 33
- 3.5 Discussion . . . 33

4 The Simplicial Product Complex Framework 35
- 4.1 From Strong Product Graph to Simplicial Product Complex . . . 35
- 4.2 Simplicial Product Complex . . . 36
- 4.3 Distinct Simplex Types . . . 40
- 4.4 Parameterized Simplicial Product Complex . . . 46
- 4.5 Hodge Laplacians and Simplicial Product Complex Filters . . . 59
- 4.6 Simplicial Product Complex Convolutional Neural Network . . . 60
- 4.7 Discussion . . . 61

5 Empirical Evaluation 65
- 5.1 Experimental Setup . . . 65
- 5.2 Results . . . 71
- 5.3 Discussion . . . 85

6 Conclusion and Future Work **87**
6.1 Answers to Research Questions 87
6.2 Future Work 88

A Additional Experimental Settings **98**
A.1 Encoder–Processor–Decoder Settings 98
A.2 Hyperparameter Tuning 98
A.3 Evaluation Metrics 102

B Additional Details about Datasets **103**
B.1 Delhi Air Pollution Dataset 103
B.2 Solar Energy Dataset 105
B.3 METR-LA Dataset 105

ACKNOWLEDGMENTS

As I close this chapter of my life, I reflect on a journey filled with incredible growth and discovery. While there were some challenging periods as I adapted to life in a new country, these experiences were far outweighed by the joy of forming meaningful friendships, exploring new perspectives, and embracing countless opportunities. This journey has shaped me profoundly, fostering personal, academic, and professional development.

First, I would like to thank my supervisor Elvin for his generous support throughout my academic journey, both within and beyond this thesis work. His invaluable guidance and our countless conversations have fundamentally shaped my reasoning and the arguments presented in this thesis. I am equally grateful to my daily supervisor Mohammad, who served as a sounding board for my ideas, provided clear direction for my work, and offered constructive feedback on multiple drafts that helped bring this thesis to its final form. I also thank my committee member Dr. Leus for generously taking time out from his busy schedule to be part of my thesis committee.

Next, I extend my heartfelt gratitude to all the friends I made in the Netherlands who transformed this foreign land into a home away from home, making my cross-continental journey so enriching. I would particularly like to thank my four closest friends, Shayan, Thanos, Chrysanthos, and Isidoro, for their incredible friendship. I am also deeply grateful to my friend Kaush, my roommate from undergraduate days in India who became my flatmate here, bringing our shared journey full circle. I am forever grateful to my partner, Devika, who patiently listened to countless thesis-related frustrations and remained a constant presence through every challenge and triumph over the past two years. Finally, I want to thank all my friends back home, especially my best friend Aashwin, who has been a constant source of encouragement through our many meaningful conversations.

Above all, I would like to thank my family - my mother Ashima, father Sanjiv, and brother Shiv - whose tremendous support made this entire endeavor possible. They have played an instrumental role in shaping the person I am today, and I would not be here without their love and encouragement.

SUMMARY

Multivariate time series data comprises multiple variables measured simultaneously over time, commonly found in many real-world domains such as traffic monitoring, energy systems, and environmental sensing. A key challenge in effectively modeling such data lies in capturing the complex but often hidden dependencies that exist both within individual time series and across different variables in the system.

Existing modeling approaches often make restrictive assumptions about the nature of these underlying relationships. Graph-based methods typically represent inter-variable dependencies through relational graphs and combine these with temporal modeling components to model multivariate time series data. Some approaches take this further by operating directly on a unified spatio-temporal product graph to jointly learn space-time interactions. While these methods have proven successful, they are fundamentally limited to modeling only pairwise interactions between variables, which may be insufficient for real-world systems where groups of variables interact across space and time. Some recent approaches have attempted to address this limitation by operating on product cell complexes, which extend product graphs by adding higher-dimensional cells to represent group interactions in the product space. However, these methods assume that all higher-order interactions are homogeneous and universally present, limiting their ability to adapt to specific patterns in different datasets.

This thesis addresses these limitations by proposing the Simplicial Product Complex (SPC), which constructs a simplicial complex in the product space to capture heterogeneous higher-order interactions across space and time. The SPC's key innovation lies in its ability to distinguish between different types of spatio-temporal relationships and learn their relative importance from data, rather than assuming that all interactions exist and are equally relevant. By generalizing from graph-based representations to topological structures and enabling data-driven learning, the SPC provides enhanced flexibility and avoids the restrictive assumptions that limit current approaches. We develop the Simplicial Product Complex Convolutional Neural Network (SPCCNN) as a neural architecture that leverages this framework to perform data-adaptive learning over higher-order spatio-temporal structures.

Our experimental evaluation demonstrates that SPCCNN achieves competitive performance with state-of-the-art methods while offering enhanced flexibility through its parameterized structure. We show that the model can adapt to dataset-specific patterns and maintain computational efficiency through sparsity regularization that prunes irrelevant relationships. Additionally, our spectral analysis reveals interesting properties of signal propagation within the SPC structure. These findings support the effectiveness of higher-order simplicial modeling for capturing complex spatio-temporal dynamics in multivariate time series data.

List of Abbreviations

Abbreviation	Full Form
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
CAN	Cell Attention Networks
CNN	Convolutional Neural Network
CPCB	Central Pollution Control Board
DCRNN	Diffusion Convolutional Recurrent Neural Network
FFNN	Fully Connected Feedforward Neural Network
GAT	Graph Attention Network
GCF	Graph Convolutional Filter
GCNN	Graph Convolutional Neural Network
GFT	Graph Fourier Transform
GLU	Gated Linear Unit
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
GSO	Graph Shift Operator
GSP	Graph Signal Processing
GTCNN	Graph Temporal Convolutional Neural Network
GTU	Gated Tanh Unit
GW	Graph WaveNet
HPC	High-Performance Computing
LSTM	Long Short-Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MLP	Multi-Layer Perceptron
MPSN	Message Passing Simplicial Networks
MSE	Mean Squared Error
NP-SPCCNN	Non-Parameterized SPCCNN
P-SPCCNN	Parameterized SPCCNN
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SAN	Simplicial Attention Networks
SAT	Simplicial Attention
SCCNN	Simplicial Complex Convolutional Neural Network

Continued on the next page

Abbreviation	Full Form
SCCONV	Simplicial Complex Convolution
SGAT	Simplicial Graph Attention
SNN	Simplicial Neural Network
SPC	Simplicial Product Complex
SPCCF	Simplicial Product Complex Convolutional Filter
SPCCNN	Simplicial Product Complex Convolutional Neural Network
SPG	Strong Product Graph
STGCN	Spatial-Temporal Graph Convolutional Network
STGNN	Spatio-Temporal Graph Neural Network
SWL	Simplicial Weisfeiler-Lehman
TCN	Temporal Convolutional Network
TNN	Topological Neural Network
TPE	Tree-structured Parzen Estimator
TSP	Topological Signal Processing
UAT	Universal Approximation Theorem

LIST OF NOTATIONS

Notation	Description
	Sets and Structures
$\mathcal{V}$	Set of nodes/vertices
$\mathcal{E}$	Set of edges
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Graph with vertices and edges
$\mathcal{G}_{\boxtimes} = (\mathcal{V}_{\boxtimes}, \mathcal{E}_{\boxtimes}, \mathbf{S}_{\boxtimes})$	Strong product graph with spatio-temporal nodes, edges and GSO
$\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S, \mathbf{S}_S)$	Spatial Graph with spatial nodes, edges and GSO
$\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T, \mathbf{S}_T)$	Temporal Graph with temporal nodes, edges and GSO
$\mathscr{X}$	Simplicial or cell complex
$\mathscr{X}^k$	k-skeleton of complex $\mathscr{X}$
$\mathcal{Z}$	Simplicial Product Complex (SPC)
$\mathcal{Z}^k$	k-simplices in SPC
$\mathcal{Z}^0$	0-skeleton (nodes) of SPC
$\mathcal{Z}^1$	1-skeleton (edges) of SPC
$\mathcal{Z}^2$	2-skeleton (triangles) of SPC
$\mathcal{T}_S$	Set of 3-cliques (triangles) in spatial graph
$\mathscr{D}$	Dataset
$\mathcal{N}(i)$	Set of neighbors of node i
$\mathcal{N}^k_{i,\ell}$	Lower neighborhood of k-simplex i
$\mathcal{N}^k_{i,u}$	Upper neighborhood of k-simplex i
s^k	k-simplex
$\mathcal{Z}^1_{\text{Spatial}}$	Set of Spatial 1-simplices in the SPC
$\mathcal{Z}^1_{\text{Temporal}}$	Set of Temporal 1-simplices in the SPC
$\mathcal{Z}^1_{\text{ST}}$	Set of Spatio-temporal 1-simplices in the SPC
$\mathcal{Z}^2_{\text{Spatial}}$	Set of Spatial 2-simplices in the SPC
$\mathcal{Z}^2_{\text{Type 1}}$	Set of Type 1 spatio-temporal 2-simplices in the SPC
$\mathcal{Z}^2_{\text{Type 2}}$	Set of Type 2 spatio-temporal 2-simplices in the SPC
$\mathcal{Z}^2_{\text{Type 3}}$	Set of Type 3 spatio-temporal 2-simplices in the SPC
$\mathcal{Z}^2_{\text{Type 4}}$	Set of Type 4 spatio-temporal 2-simplices in the SPC
	Cardinalities and Dimensions
$\mathbb{R}^n$	n-dimensional real vector space
N	Number of nodes
T	Observation window

Continued on the next page

Notation	Description
H	Prediction horizon
$\lvert\mathcal{E}_S\rvert$	Number of edges in the spatial graph
$\lvert\mathcal{T}_S\rvert$	Number of 3-cliques (triangles) in the spatial graph
N^k	Number of k-simplices
N^0, N^1, N^2	Number of 0-, 1-, 2-simplices respectively
N^1_{Spatial}	Number of spatial 1-simplices
N^1_{Temporal}	Number of temporal 1-simplices
N^1_{ST}	Number of spatio-temporal 1-simplices
N^2_{Spatial}	Number of spatial 2-simplices
$N^2_{\text{Type 1}}$	Number of Type 1 2-simplices
$N^2_{\text{Type 2}}$	Number of Type 2 2-simplices
$N^2_{\text{Type 3}}$	Number of Type 3 2-simplices
$N^2_{\text{Type 4}}$	Number of Type 4 2-simplices
$N^{2,t}_{\text{Type k}}$	Per-timestep number of Type k 2-simplices
F	Number of features/channels
K	Maximum order of simplicial complex
L	Number of layers in neural network
L_1, L_2	Filter orders for lower and upper Laplacians
	Matrices
$\mathbf{A}$	Adjacency matrix
$\mathbf{B}$	Incidence matrix
$\mathbf{D}$	Degree matrix
$\mathbf{L}$	Graph Laplacian
$\mathbf{S}$	Graph Shift Operator (GSO)
$\mathbf{I}$	Identity matrix
$\mathbf{I}_S$	Identity matrix in the spatial domain $\mathbb{R}^{N\times N}$
$\mathbf{I}_T$	Identity matrix in the temporal domain $\mathbb{R}^{T\times T}$
$\mathbf{B}_{1S}$	Node-to-edge incidence matrix of spatial graph
$\mathbf{B}_{1T}$	Node-to-edge incidence matrix of temporal graph
$\mathbf{B}_{2S}$	Edge-to-triangle incidence matrix of spatial graph
$\mathbf{B}_k$	k-th boundary operator
$\mathbf{B}_{1\diamond}$	Non-parameterized node-to-edge incidence matrix of SPC
$\mathbf{B}_{2\diamond}$	Non-parameterized edge-to-face incidence matrix of SPC
$\tilde{\mathbf{B}}_{1\diamond}$	Parameterized node-to-edge incidence matrix
$\tilde{\mathbf{B}}_{2\diamond}$	Parameterized edge-to-face incidence matrix
$\mathbf{B}^{\text{Inc}}_{1S}$	Incidence matrix isolating only incoming nodes of the spatial graph
$\mathbf{B}^{\text{Out}}_{1S}$	Incidence matrix isolating only outgoing nodes of the spatial graph
$\mathbf{B}^{\text{Spatial}}_{1\diamond}$	Node-to-edge incidence matrix for spatial edges in the SPC
$\mathbf{B}^{\text{Temporal}}_{1\diamond}$	Node-to-edge incidence matrix for temporal edges in the SPC
$\mathbf{B}^{\text{ST}}_{1\diamond}$	Node-to-edge incidence matrix for spatio-temporal edges in the SPC

Continued on the next page

Notation	Description
$\mathbf{B}_{2\diamond}^{\text{Spatial}}$	Edge-to-triangle incidence matrix for spatial triangles in the SPC
$\mathbf{B}_{2\diamond}^{\text{Type 1}}$	Edge-to-triangle incidence matrix for Type 1 in the SPC
$\mathbf{B}_{2\diamond}^{\text{Type 2}}$	Edge-to-triangle incidence matrix for Type 2 in the SPC
$\mathbf{B}_{2\diamond}^{\text{Type 3}}$	Edge-to-triangle incidence matrix for Type 3 in the SPC
$\mathbf{B}_{2\diamond}^{\text{Type 4}}$	Edge-to-triangle incidence matrix for Type 4 in the SPC
$\mathbf{B}_{2\diamond,t}^{\text{Type k}}$	Type k triangles incidence at time t
$\mathbf{X}_t^{\text{Type k}}$	Spatial edges in Type k triangles at time t
$\mathbf{Y}_t^{\text{Type k}}$	Temporal edges in Type k triangles at time t
$\mathbf{Z}_t^{\text{Type k}}$	Spatio-temporal edges in Type k triangles at time t
$\mathbf{L}_k$	k-th order Hodge Laplacian
$\mathbf{L}_{k,d}$	Lower k-Hodge Laplacian
$\mathbf{L}_{k,u}$	Upper k-Hodge Laplacian
$\mathbf{L}_{k\diamond}$	Non-parameterized k-Hodge Laplacian of SPC
$\mathbf{L}_{k\diamond,d}$	Non-parameterized lower k-Hodge Laplacian of SPC
$\mathbf{L}_{k\diamond,u}$	Non-parameterized upper k-Hodge Laplacian of SPC
$\tilde{\mathbf{L}}_{k\diamond}$	Parameterized Hodge Laplacian of the SPC
$\tilde{\mathbf{L}}_{k\diamond,d}$	Parameterized lower k-Hodge Laplacian of SPC
$\tilde{\mathbf{L}}_{k\diamond,u}$	Parameterized upper k-Hodge Laplacian of SPC
$\mathbf{V}$	Matrix of eigenvectors
$\mathbf{U}_G$	Eigenvectors of gradient subspace
$\mathbf{U}_C$	Eigenvectors of curl subspace
$\mathbf{U}_H$	Eigenvectors of harmonic subspace
$\mathbf{\Lambda}$	Diagonal matrix of eigenvalues
Λ_G	Eigenvalues of gradient subspace
Λ_C	Eigenvalues of curl subspace
Λ_H	Eigenvalues of harmonic subspace
	Signals and Data
$\mathbf{x}$	Node signal vector
$\mathbf{x}^k$	Signal on k-simplices
$\mathbf{X}$	Multivariate node signals matrix
$\mathbf{X}_t$	Signal at time t
$\mathbf{X}_{t:t+T}$	Signal sequence from time t to $t+T$
f^k	k-simplicial signal function
$\tilde{\mathbf{x}}$	Signal in spectral domain
$\mathbf{f}_G^k$	Gradient component of signal
$\mathbf{f}_C^k$	Curl component of signal
$\mathbf{f}_H^k$	Harmonic component of signal
$\mathbf{Y}$	Target signal
$\hat{\mathbf{Y}}$	Predicted signal
$\boldsymbol{\mu}$	Mean vector

Continued on the next page

Notation	Description
$\boldsymbol{\sigma}$	Standard deviation vector
	Parameters
h_k	Filter coefficient for k-hop
h_0	Bias/offset filter coefficient
α_{l_1}	Lower Laplacian filter coefficients
β_{l_2}	Upper Laplacian filter coefficients
e_1, e_2, e_3	Edge type parameters
f_0, f_1, f_2, f_3, f_4	Face type parameters
θ	Model parameters
	Functions and Operations
$\sigma(\cdot)$	Activation function
$\phi^{(1)}$	Lifting function for edges
$\phi^{(2)}$	Lifting function for faces
SoftPlus$(\cdot)$	SoftPlus activation function
ReLU$(\cdot)$	Rectified Linear Unit
$\ell(\cdot,\cdot)$	Loss function
$\odot$	Element-wise multiplication
$\otimes$	Kronecker product
$\oplus$	Direct sum of vector spaces
im$(\cdot)$	Image space of matrix
ker$(\cdot)$	Kernel (null space) of matrix
$\mathbf{P}_{r\to k}$	Projection operator from order-r to order-k
$H_k(\cdot)$	Topological filter on k-simplices
$H(S)$	Graph filter using the GSO S
$h(\lambda)$	Frequency filter response
$H_k(\mathbf{L}_{k\diamond,d}, \mathbf{L}_{k\diamond,u})$	Non-parameterized Simplicial Product Complex Convolutional Filter
$\tilde{H}_k(\tilde{\mathbf{L}}_{k\diamond,d}, \tilde{\mathbf{L}}_{k\diamond,u})$	Parameterized Simplicial Product Complex Convolutional Filter

1

INTRODUCTION

1.1. MULTIVARIATE TIME SERIES

Multivariate time series arise naturally in a wide range of real-world domains where measurements are collected simultaneously across multiple sensors or modalities over time. In traffic networks [39, 75], for instance, sensors placed across different road segments continuously record vehicle speed. In energy systems [43, 83], solar farms and power grids monitor consumption metrics across geographically distributed sites at regular intervals. Environmental sensing [20] offers another example, where monitoring stations track pollutants like PM2.5 at various locations over time.

While each individual time series in such multivariate settings reflects the behavior of a particular sensor or variable, these signals rarely evolve in isolation [14, 48]. Instead, their dynamics are shaped by complex interactions, often governed by a latent relational structure that induces dependencies between variables. For example, in traffic networks, the flow of vehicles at a given intersection is not solely determined by its own historical patterns but also by time-lagged congestion and movement at other intersections that are connected to it. These dependencies between space and time give rise to rich spatio-temporal patterns in multivariate time series data. Effectively modeling such data thus requires capturing both the temporal dynamics within individual variables and the inter-variable dependencies that emerge across space and time [47].

However, in most real-world scenarios, the structure underlying these inter-variable dependencies is not directly observable and must be inferred or approximated from data [16, 85]. As a result, various modeling frameworks have emerged, each making distinct assumptions about the nature of these relationships, how they are learned, and how they are integrated with temporal modeling [41]. These assumptions define the inductive bias [49] employed during learning. While the inductive bias provides beneficial structural guidance, it may also constrain the model's representational capacity. For instance, graph-based models typically assume that dependencies are exclusively pairwise, limiting them from capturing more complex, higher-order interactions. As we shall see, this restriction plays a central role in shaping how different modeling paradigms approach

spatio-temporal learning.

1.2. MODELING PARADIGMS

Structure-unaware approaches [13, 50, 73] model each variable in multivariate time series data independently, capturing temporal dynamics without accounting for inter-variable relationships. Although these methods can effectively learn local patterns in individual sequences, they ignore the relational structure altogether, often resulting in spurious correlations, reduced generalization, and high sample complexity [23].

Graph-aware approaches address this limitation by incorporating inter-variable relationships into the modeling framework, using a relational graph as an inductive bias. Collectively known as Spatio-Temporal Graph Neural Networks (STGNNs), these models combine graph-based convolutions [29] with temporal modules to jointly learn spatial and temporal dependencies. This inductive bias allows STGNNs to effectively model multivariate time series data, often improving predictive performance and sample efficiency [23]. A particularly principled subclass of STGNNs is the class of Graph-Time Convolutional Neural Networks (GTCNNs) [59], which construct a product graph by combining the spatial and temporal graphs into a unified representation. Applying graph convolutions in the product graph directly enables GTCNNs to learn spatio-temporal interactions within a coherent spectral framework, offering both theoretical interpretability and strong empirical performance.

While such models benefit from a strong inductive bias that captures spatial and temporal dependencies, this comes at the cost of a restrictive structural assumption: relationships between variables are treated as strictly pairwise. This limits their ability to represent complex interactions among groups of variables, which often arise in real-world spatio-temporal systems. For instance, simultaneous congestion at intersections A and B in a traffic network may jointly influence traffic at intersection C, particularly when C is accessible only through A and B. Capturing such group-level effects requires modeling frameworks capable of explicitly representing higher-order relationships in space and time.

Recent advances in topological signal processing [62] have sought to go beyond pairwise interactions by modeling multi-node dependencies using higher-order structures. In the spatial domain, topological structures such as simplicial complexes [4] and cell complexes [63] provide a natural way to algebraically represent such group-level relationships through higher-dimensional faces. Filters [81] and neural architectures [15, 34, 78] defined on these complexes have shown success in capturing higher-order interactions inherently present in many real-world networks such as citation [65] and social networks [18].

Building on these ideas, product cell complexes [57] introduced *topologically-aware* spatio-temporal modeling, encoding higher-order relationships using a cellular complex in the product space. While this inductive bias incorporates higher-order space-time structure, it assumes that all such spatio-temporal relationships are present and uniform, leaving no room to adapt to the specific dependency patterns observed in a given dataset. Their framework represents interactions as homogeneous cells, without distinguishing between different types of higher-order relationships that can exist in space

and time. Moreover, their contributions remain theoretical and do not include a trainable neural architecture for downstream spatio-temporal modeling.

To address these limitations, this thesis introduces the *Simplicial Product Complex* (SPC), a novel modeling framework that captures higher-order spatio-temporal dependencies using a structured simplicial representation in the product space. Unlike product cell complexes, which encode all interactions as uniform, homogeneous cells, the SPC leverages the simplicial structure to explicitly isolate different types of spatio-temporal relationships. This separation enables flexible parameterization, allowing the model to learn which interaction patterns are present and relevant from the data itself rather than assuming them a priori. In doing so, the SPC avoids imposing fixed relational assumptions and supports higher-order inductive biases that are both more expressive and data-adaptive.

1.3. RESEARCH QUESTIONS

This thesis is thus driven by the following overarching research question:

(RQ) *How can we model multivariate time-series data by leveraging higher-order spatio-temporal connections through a simplicial structure in the product space?*

To address this central question, we propose the Simplicial Product Complex Convolutional Neural Network (SPCCNN), a novel spatio-temporal learning architecture that models higher-order spatio-temporal dynamics by operating on our constructed SPC framework. Since the SPC retains a purely simplicial structure in space and time, topological filters present in the SPCCNN architecture allow for learning complex multi-variable interactions. We explore the main question through the following sub-questions:

(RQ1) *How can we effectively parameterize the simplicial product space to learn higher-order spatio-temporal relationships from data?*

To answer this research question, we identify and categorize the distinct types of higher-order relationships encoded in the simplicial structure of the SPC. We then propose a parameterized variant of the SPC that introduces learnable weights over different types of spatio-temporal simplices. When the SPCCNN operates on this parameterized structure, its filters can modulate the influence of each relationship type based on the training data, enabling more flexible and data-adaptive modeling.

(RQ2) *To what extent does the parameterized SPCCNN improve predictive performance, data-adaptability, and scalability in spatio-temporal learning?*

To explore this research question, we empirically evaluate the parameterized SPCCNN on multivariate forecasting and spatio-temporal imputation tasks. We benchmark its predictive performance against a range of baselines on real-world datasets, including METR-LA [39], Solar Energy [43], and a self-curated Delhi air pollution dataset [17]. To assess data-adaptability, we analyze the flexibility that parameterization provides to adapt to specific spatio-temporal patterns present in different datasets. Finally, we evaluate scalability by applying sparsity constraints that prune unimportant higher-order relationships, potentially reducing computational overhead without sacrificing performance.

(RQ3) *What are the spectral properties of the SPC, and how do they inform our understanding of signal propagation across the spatio-temporal complex?*

To investigate this research question, we conduct a spectral analysis of the 1-Hodge Laplacian of the SPC to study how edge flows are distributed across its Hodge subspaces. This analysis reveals how signals propagate through different types of spatio-temporal interactions, offering insight into the behavior and utility of higher-order structures in the SPC.

1.4. THESIS CONTRIBUTIONS

By addressing these research questions, this thesis makes the following contributions:

(C1) **SPC Framework and SPCCNN Architecture:** We introduce both the SPC framework and the SPCCNN neural architecture that together perform parameterized, simplicial filtering to capture the latent higher-order connections in multivariate time series data.

(C2) **Empirical Evaluation:** We conduct a comprehensive empirical evaluation of SPCCNN against state-of-the-art baselines on real-world datasets to validate our hypothesis that latent higher-order simplicial structures are valuable in modeling multivariate time series data. Our analysis examines downstream predictive performance as well as the flexibility and scalability advantages provided by parameterization.

(C3) **Spectral Analysis:** We provide a spectral analysis of the 1-Hodge Laplacian, revealing how different simplex types influence signal propagation and validating the structural expressivity of the SPC.

Thesis Structure. The remainder of this thesis is organized as follows. Chapter 2 presents the background on graph signal processing, topological learning, and spatio-temporal modeling. Chapter 3 surveys related literature in the multivariate time series modeling paradigm in further detail. Chapter 4 presents our core methodology, describing the SPC framework, our parameterisation mechanism and the SPCCNN architecture. Chapter 5 reports our empirical results across multiple datasets and tasks. Finally, Chapter 6 concludes the thesis and outlines future directions.

2

BACKGROUND

This chapter introduces the background that this work will build upon. We begin with the foundations of Graph Signal Processing (GSP), introducing graph filters and Graph Convolutional Neural Networks (GCNNs) in Section 2.1. We then introduce Topological Signal Processing (TSP), the extension of GSP to topological domains such as simplicial and cell complexes in Section 2.2. This is followed by a discussion of product graphs and, more generally, product spaces and how they are used to model spatio-temporal data in Section 2.3. We conclude this chapter in Section 2.4.

2.1. GRAPH SIGNAL PROCESSING

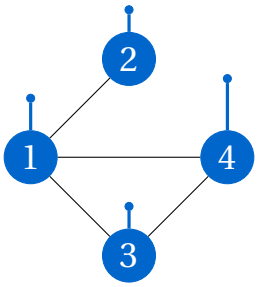
Graphs are mathematical tools that allow us to represent entities and encode relationships between them. They are used to describe both physical and abstract structures. For example, when expressing a road network as a graph, intersections serve as nodes while roads represent the edges between them. In more abstract settings, such as social networks, nodes can represent individuals and edges capture social interactions, which do not correspond to physical connections but to intangible relationships. At their core, graphs provide a framework for modeling such complex systems where relational structure plays a central role.

GSP extends classical signal processing concepts such as filtering and Fourier analysis to graphs, which are irregular and non-Euclidean in nature since the underlying domain lacks a regular grid structure and exhibits varying local connectivity. In this section, we begin by formally defining graphs and introducing the different algebraic representations of the graph structure in Section 2.1.1. We then describe graph signals and discuss the extension of Fourier analysis to the graph domain in Section 2.1.2. This is followed by an introduction to graph filters in Section 2.1.3, and finally, we outline how graph filters are leveraged to build GCNNs in Section 2.1.4.

2.1.1. GRAPHS

Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical structure that models relationships between entities. The entities are represented as a set of N nodes, $\mathcal{V} = \{1, 2, \ldots, N\}$, and the relationships between them are captured by the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The number of nodes and edges in the graph is denoted by $|\mathcal{V}| = N$ and $|\mathcal{E}|$, respectively. If an edge $(i, j) \in \mathcal{E}$, then nodes i and j are said to be connected or adjacent.

Graphs can be classified as either directed or undirected. In undirected graphs, $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$, meaning edges have no orientation. In directed graphs, the edge $(i, j) \in \mathcal{E}$ implies a directed relationship from node i to node j, and (j, i) may or may not be present. The direction induces notions of incoming and outgoing nodes: for a directed edge $(i, j) \in \mathcal{E}$, node i is the outgoing node and node j is the incoming node.



$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -1 & -1 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Figure 2.1: An example undirected graph with its corresponding adjacency (**A**), incidence (**B**, arbitrary directions), degree (**D**) and graph Laplacian (**L**) matrices.

The structure of a graph can be represented using various algebraic representations:

- **Incidence Matrix:** The incidence matrix $\mathbf{B} \in \mathbb{R}^{N \times |\mathcal{E}|}$ encodes the relationship between nodes and edges. For an edge $e_k = (i, j)$, the matrix entry $\mathbf{B}_{ik} = 1$ if node i is the outgoing node, $\mathbf{B}_{jk} = -1$ if node j is the incoming node, and $\mathbf{B}_{lk} = 0$ for all other nodes $l \notin \{i, j\}$. In undirected graphs, which lack inherent edge directionality, orientations are assigned arbitrarily but consistently to ensure a well-defined representation. The rows in incidence matrices represent nodes and the columns represent edges.
- **Adjacency Matrix:** The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ represents pairwise connectivity between nodes. The entry $\mathbf{A}_{ij}$ is non-zero if $(i, j) \in \mathcal{E}$, and zero otherwise. In unweighted graphs, $\mathbf{A}_{ij} \in \{0, 1\}$, whereas in weighted graphs, $\mathbf{A}_{ij}$ reflects the strength

of the connection between nodes i and j. The adjacency matrix is symmetric ($\mathbf{A}^\top = \mathbf{A}$) for undirected graphs, and asymmetric in general for directed graphs.

- **Graph Laplacian:** The direction agnostic graph Laplacian $\mathbf{L} \in \mathbb{R}^{N\times N}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} \in \mathbb{R}^{N\times N}$ is the degree matrix with entries $\mathbf{D}_{ii} = \sum_{j=1}^{N} \mathbf{A}_{ij}$, representing the number of nodes connected to node i. The Laplacian can also be expressed using the incidence matrix as $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$. The graph Laplacian is symmetric and positive semi-definite, and plays a central role in spectral graph theory as we will see in the next section. A normalized variant, the symmetric normalized Laplacian, is given by $\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

The matrices $\mathbf{A}$, $\mathbf{L}$, and $\mathbf{L}_{\text{norm}}$ described above are all special cases of the Graph Shift Operator (GSO), a more general representation denoted by $\mathbf{S} \in \mathbb{R}^{N\times N}$. A GSO is any matrix whose non-zero entries correspond precisely to the edges of the graph: $\mathbf{S}_{ij} \neq 0$ if and only if $(i, j) \in \mathcal{E}$. Figure 2.1 illustrates a simple graph along with its corresponding incidence matrix, adjacency matrix, degree matrix and graph Laplacian.

2.1.2. GRAPH SIGNALS AND FOURIER TRANSFORM

The previous section introduced algebraic representations of the graph structure. In practice, graphs encode real-world systems, with measurable quantities associated with the nodes and edges of the graph, known as graph signals.

Understanding how these signals vary in relation to the graph's structure is crucial for uncovering underlying patterns and designing predictive models for graphs. This is achieved via the spectral graph theory [22], which provides a framework to analyze the frequency content and smoothness of graph signals, revealing how the values change across the structure of the graph.

GRAPH SIGNALS

Formally, a graph signal defined on the nodes is a function $f^v : \mathcal{V} \to \mathbb{R}^N$, which assigns a scalar value to each node, as illustrated by the signal bars defined on the nodes in Figure 2.1. The signal forms a vector space and can be represented as a vector $\mathbf{x}^v \in \mathbb{R}^N$, where $[\mathbf{x}^v]_i$ denotes the value at node i. In the multivariate case, where each node has F features, the signal is stored as a matrix $\mathbf{X}^v \in \mathbb{R}^{N\times F}$, with each column representing a feature across all nodes. A graph may also have signals defined on it's edges, captured by a function $f^e : \mathcal{E} \to \mathbb{R}$, represented as a vector $\mathbf{x}^e \in \mathbb{R}^{|\mathcal{E}|}$. For multivariate edge signals, the data is similarly stored as a matrix $\mathbf{X}^e \in \mathbb{R}^{|\mathcal{E}|\times F}$. For the rest of the section, we will assume that the signal are only present on the nodes, denoted by $\mathbf{x}$.

GRAPH FOURIER TRANSFORM

Graph Fourier Transform (GFT) is the extension of the traditional Fourier Transform in signal processing to signals defined on graphs. GFT is defined based on the eigendecomposition of a diagonalizable GSO $\mathbf{S}$, expressed as

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \tag{2.1}$$

where $\mathbf{V} = [\mathbf{v_1}, \ldots, \mathbf{v_N}]$ contains the eigenvectors of $\mathbf{S}$, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix containing the corresponding eigenvalues. The GFT of the node signal is given by projecting $\mathbf{x}$ onto the eigen-space of $\mathbf{S}$, such that:

$$\tilde{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}, \tag{2.2}$$

where $\tilde{\mathbf{x}}$ represents the graph signal in the spectral domain, indicating the contribution of each eigenvector in the representation of $\mathbf{x}$. The inverse GFT reconstructs the signal as:

$$\mathbf{x} = \mathbf{V}\tilde{\mathbf{x}}. \tag{2.3}$$

The eigenvectors in $\mathbf{V}$ act as the basis functions for the GFT, analogous to the complex exponentials in the classical Fourier Transform. When the GSO is the Graph Laplacian, the eigenvalues $\mathbf{\Lambda} = [\lambda_1, \ldots, \lambda_N]$ represent the graph frequencies and quantify the smoothness of each basis function relative to the graph structure. Specifically, eigenvectors corresponding to smaller eigenvalues vary less across connected nodes, meaning they are smoother over the graph. Conversely, eigenvectors associated with larger eigenvalues exhibit higher variability. This relationship between eigenvalues and smoothness allows the GFT to analyze the signal's variation in alignment with the graph's topology.

2.1.3. GRAPH CONVOLUTIONAL FILTERS

The spectral analysis of graph signals enables their decomposition into components of varying smoothness based on the underlying graph structure. In the context of predictive modeling on graphs, it is often desirable to modulate these components to suppress noise or amplify informative patterns. While one could manually design filters to enhance or suppress specific frequencies, like low pass filters or band pass filters, a more principled approach is to learn such filters directly from graph-structured data [29]. This is typically achieved in the vertex (spatial) domain by learning a weighted combination of shifted signals through a Graph Convolutional Filter (GCF). GCFs also admit an interpretation in the spectral domain.

GRAPH SIGNAL SHIFT

A graph signal shift propagates information across the graph based on its topology. Given a GSO $\mathbf{S}$ and a signal $\mathbf{x}$, the shifted signal is given by:

$$\mathbf{S}(\mathbf{x}) = \mathbf{S}\mathbf{x}, \tag{2.4}$$

which updates the signal at each node based on the values of its immediate neighbors, making it a localized operation. For instance, when $\mathbf{S} = \mathbf{A}$, the adjacency matrix of an unweighted graph, the shifted signal at node i is:

$$[\mathbf{S}\mathbf{x}]_i = \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j, \tag{2.5}$$

where $\mathcal{N}(i)$ denotes the set of neighbors of node i. While applying the shift operator once updates the node representation using the immediate neighbors, repeated application of $\mathbf{S}$ k times enables signal propagation from a radius of k-neighbors, aggregating

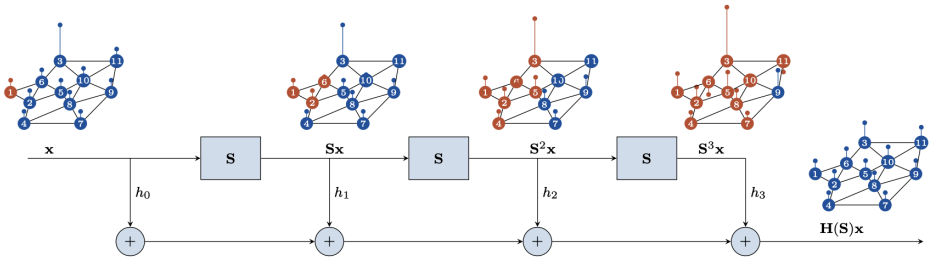


Figure 2.2: The graph convolution and underlying signal shifting operations in the GCF, adapted from [38]

signals from the k-hop neighborhood. Figure 2.2 illustrates the shifting operation over multiple steps. The k-shifted signal $\mathbf{x}^{(k)}$ can be recursively defined as:

$$\mathbf{x}^{(k)} = \mathbf{S}^k \mathbf{x}^{(0)} = \mathbf{S}(\mathbf{S}^{k-1}\mathbf{x}^{(0)}) = \mathbf{S}(\mathbf{x}^{(k-1)}) \tag{2.6}$$

GRAPH CONVOLUTIONAL FILTER

Graph Convolutional Filters (GCFs) extend the classical convolution operation to graphs by performing a weighted combination of multiple shifted signals. A GCF of order K is defined as:

$$\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x}, \tag{2.7}$$

where $\{h_k\}_{k=0}^{K}$ are the filter coefficients, and $H(\mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k$ is the filter matrix. The filter coefficient h_k indicates the importance of the k-hop neighborhood. This formulation ensures that GCFs are linear, shift-invariant, and operate locally within K-hop neighborhoods [38], making them both efficient and scalable. Figure 2.2 illustrates the convolution operations in a GCF.

SPECTRAL DOMAIN INTERPRETATION

In the spectral domain, graph convolution corresponds to point-wise multiplication of frequency components. If $\mathbf{S}$ admits an eigen-decomposition $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$, and $\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}$ is the GFT of the signal $\mathbf{x}$, then the filtered signal in the spectral domain is:

$$\hat{\mathbf{y}} = h(\Lambda)\hat{\mathbf{x}}, \tag{2.8}$$

where $h(\mathbf{\Lambda}) = \text{diag}(h(\lambda_0), h(\lambda_1), \ldots, h(\lambda_{N-1}))$ encodes the filter's frequency response, and

$$h(\lambda) = \sum_{k=0}^{K} h_k \lambda^k \tag{2.9}$$

is the polynomial spectral kernel. This operation modulates each frequency component $\hat{\mathbf{x}}_i$ by scaling it with $h(\lambda_i)$, enabling selective enhancement or attenuation based on the graph's spectral structure.

Hence, GCFs admit dual interpretations: in the spatial domain, they act as a weighted combination of shifted signals and in the spectral domain, they correspond to point-wise multiplication of the signal's frequency components with the filter's spectral response. Importantly, the filter coefficients $\{h_k\}$ can be learned end-to-end from downstream data, allowing the model to automatically determine which graph frequencies to emphasize or suppress.

To enhance the expressiveness of GCFs, they are extended into GCNNs. By stacking multiple GCF layers and introducing non-linearities, GCNNs can extract complex, hierarchical features from graph-structured data [38]. This allows them to learn representations end-to-end for downstream tasks, making them highly effective across a wide range of graph learning applications [74].

2.1.4. GRAPH CONVOLUTIONAL NEURAL NETWORKS

GCNNs are multi-layered architectures where each layer consists of a collection of GCFs followed by point-wise non-linearities.

A single layer transforms a set of input features into a new set of output features. The number of raw input channels is denoted by F_0 and corresponds to the dimensionality of the initial node features. The number of output features at each layer ℓ is a hyperparameter and is denoted as F_ℓ.

Let the input to the ℓ-th layer (which is the output of the $\ell-1$-th layer) be denoted as:

$$\mathbf{X}^{(\ell-1)} = [\mathbf{x}_1^{(\ell-1)}, \mathbf{x}_2^{(\ell-1)}, \ldots, \mathbf{x}_{F_{\ell-1}}^{(\ell-1)}] \in \mathbb{R}^{N \times F_{\ell-1}}, \tag{2.10}$$

where $F_{\ell-1}$ is the number of input feature channels. Each column $\mathbf{x}_g^{(\ell-1)} \in \mathbb{R}^N$ represents the values of the g-th feature across all nodes.

The layer produces F_ℓ output feature channels. For each output feature $f = 1, \ldots, F_\ell$, the intermediate signal before applying the activation function is computed as:

$$\mathbf{z}_f^{(\ell)} = \sum_{g=1}^{F_{\ell-1}} \mathbf{H}_{fg}^{(\ell)} \mathbf{x}_g^{(\ell-1)}, \tag{2.11}$$

where $\mathbf{H}_{fg}^{(\ell)}$ is a learnable GCF that models the contribution of the g-th input feature to the f-th output feature. Following Equation 2.7, each filter is written as:

$$\mathbf{H}_{fg}^{(\ell)} = \sum_{k=0}^{K} h_{fgk}^{(\ell)} \mathbf{S}^k, \tag{2.12}$$

where $h_{fgk}^{(\ell)}$ are learnable coefficients. Substituting Equation 2.12 into Equation 2.11 yields:

$$\mathbf{z}_f^{(\ell)} = \sum_{g=1}^{F_{\ell-1}} \sum_{k=0}^{K} h_{fgk}^{(\ell)} \mathbf{S}^k \mathbf{x}_g^{(\ell-1)}. \tag{2.13}$$

The final output feature $\mathbf{x}_f^{(\ell)}$ is then obtained by applying a non-linear activation function $\sigma(\cdot)$:

$$\mathbf{x}_f^{(\ell)} = \sigma(\mathbf{z}_f^{(\ell)}), \quad f = 1, \ldots, F_\ell, \tag{2.14}$$

and the complete output matrix is:

$$\mathbf{X}^{(\ell)} = [\mathbf{x}_1^{(\ell)}, \mathbf{x}_2^{(\ell)}, \ldots, \mathbf{x}_{F_\ell}^{(\ell)}] \in \mathbb{R}^{N \times F_\ell}. \tag{2.15}$$

This layered structure allows the model to iteratively aggregate and transform information from local neighborhoods, enabling the learning of increasingly abstract node representations in deeper layers.

For a GCNN with L layers, the final output is given by:

$$\mathbf{x}^{(L)} = \sigma\left(\mathscr{H}^{(L)}\left(\mathbf{x}^{(L-1)}\right)\right) \tag{2.16}$$

where $\mathscr{H}^{(L)}$ represents the full set of GCFs applied at layer L. The output $\mathbf{x}^{(L)}$ is typically passed to a readout or classification layer, depending on the downstream task.

Various modifications of GCNNs have been proposed, each altering the filter design, aggregation function, or neighborhood definition to enhance expressiveness, scalability, or domain adaptation. These include spectral methods like ChebNet [25], attention-based mechanisms like GAT [69], and scalable convolutions like GraphSAGE [36], each tailored to specific application contexts. However, despite these innovations, all such models operate within the constraints of graph-based representations and inherit their fundamental limitations.

In particular, GCNNs (and it's variants) struggle to model long-range dependencies due to the phenomenon of over-squashing [31] where information from distant nodes is compressed as it propagates through layers, leading to diminished expressive power. Moreover, conventional GCNNs rely primarily on pairwise interactions encoded in edges, limiting their ability to capture higher-order relationships or group dynamics that are central to many real-world systems [4]. These limitations have motivated the development of more expressive frameworks that extend beyond graphs, leveraging higher-order topologies capable of naturally representing multi-node interactions and richer relational structures.

2.2. TOPOLOGICAL SIGNAL PROCESSING

TSP extends GSP to topological domains. While graphs are limited to modeling pairwise relationships through edges, higher-order topological domains can capture multi-node connections and interactions [62]. We focus specifically on simplicial and cell complexes, which have a rich algebraic representation.

We begin by introducing simplicial and cell complexes and discussing their representations in Section 2.2.1. We then delve into Hodge Theory, highlighting how Fourier transformations are applied within each topological subspace of Hodge Laplacians in Section 2.2.2. This is followed by extensions of GCFs and GCNNs to the topological domain in Sections 2.2.3 and 2.2.4, respectively. Finally, we discuss approaches for translating graph representations into higher-order topologies using feature and topology lifting in Section 2.2.5.

2.2.1. TOPOLOGIES

SIMPLICIAL COMPLEXES

Given a finite vertex set $\mathcal{V}$, a k-simplex s^k is defined as a subset of $\mathcal{V}$ with cardinality $k+1$. The faces of s^k are the $(k-1)$-simplices contained within it, while its cofaces are the $(k+1)$-simplices that include s^k as a face. This requirement that all faces of a simplex must also be included in the complex is known as the face inclusion property of simplicial complexes. Geometrically, simplices follow a strict structure: 0-simplices are vertices, 1-simplices are edges, 2-simplices are triangles, 3-simplices are tetrahedra, and so on.

A simplicial complex $\mathcal{X}$ is a finite collection of simplices closed under the face inclusion property: if $s^k \in \mathcal{X}$, then all its faces $s^{k-1} \subset s^k$ are also in $\mathcal{X}$. All k-simplices in $\mathcal{X}$ form the k-skeleton, denoted $\mathcal{X}^k$, defined as:

$$\mathcal{X}^k = \bigcup_{i=1}^{N_k} s_i^k, \tag{2.17}$$

where s_i^k denotes the i-th k-simplex and $N_k = |\mathcal{X}^k|$ is the number of k-simplices in the complex. The dimension of a simplicial complex is given by the largest order of any simplex it contains.

Orientation: The orientation of a simplex refers to a choice of ordering of its vertices, which determines the direction of traversal of the simplex. For example, a 2-simplex $\{i, j, k\}$ can be oriented as $[i, j, k]$, and reversing the order (e.g., $[j, i, k]$) is considered to change the orientation. For consistency in computations, it is common to assign a fixed orientation to each simplex. A widely used convention is the lexicographic ordering, where the vertices of a simplex are ordered in increasing numerical order. That is, for a k-simplex $s^k = \{v_0, v_1, \ldots, v_k\}$, the orientation is taken to be $[v_0, v_1, \ldots, v_k]$ such that $v_0 < v_1 < \cdots < v_k$.

Adjacencies and Neighborhoods: Unlike graphs, which define the adjacency of nodes through edges, simplicial complexes support richer notions of adjacency for the higher-order topology:

- **Lower Adjacency:** Two k-simplices are lower adjacent if they share a common $(k-1)$-face. For instance, two edges will be lower adjacent, if they share a common node.
- **Upper Adjacency:** Two k-simplices are upper adjacent if they both are faces of a common $(k+1)$-simplex. For instance, two edges will be upper adjacent, if they are a part of the same triangle.

Accordingly, for a k-simplex s_i^k, we define the lower neighborhood $\mathcal{N}_{i,\ell}^k$ as the set of simplices lower adjacent to s_i^k, and the upper neighborhood $\mathcal{N}_{i,u}^k$ as the set of its upper adjacent simplices. The red nodes and triangle in Figure 2.3 highlights the upper (red triangle) and lower (red nodes) neighborhood of a specific 1-simplex (green edge) in a simplicial complex of order 2.

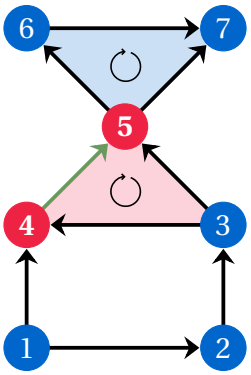
Simplicial Signals: Just as graph signals represent quantifiable measurements associated with the nodes, and possibly, edges of a graph, a k-simplicial signal generalizes this notion by defining a real-valued function on the k-simplices of a simplicial complex.

$$f^k : X^k \to \mathbb{R}^{N_k}. \tag{2.18}$$

It is typically represented as a vector $\mathbf{x}^k = [x_1^k, \ldots, x_{N_k}^k]^\top \in \mathbb{R}^{N_k}$, where x_i^k is the signal value associated with the i-th k-simplex. If the signals are multivariate, they can be represented by matrix $\mathbf{X}^k \in \mathbb{R}^{N_k \times F_k}$, where F_k is the number of channels for the $k-$th simplicial signals.

Since simplicial signals are defined with respect to a chosen reference orientation, the signal value alternates sign when the orientation is flipped. If $s^{k'} = -s^k$ denotes a simplex with reversed orientation, then:

$$f^k(s^{k'}) = -f^k(s^k). \tag{2.19}$$



$$\mathbf{B}_1 = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \mathbf{B}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}$$

Figure 2.3: An example simplicial complex of order 2 with its corresponding node-to-edge ($\mathbf{B_1}$) and edge-to-face ($\mathbf{B_2}$) incidence matrices. The upper and lower neighborhood of a specific edge (green) are marked in red. The signals on nodes, edges and faces are omitted for clarity.

Algebraic Representation: Simplicial complexes admit an algebraic representation via the incidence matrices, also known as boundary operators. The k-th boundary operator

is the incidence matrix:

$$\mathbf{B}_k \in \mathbb{R}^{N_{k-1} \times N_k}, \tag{2.20}$$

which maps each k-simplex to its constituent $(k-1)$-faces. In particular, $\mathbf{B}_1$ is the node-to-edge incidence matrix, and $\mathbf{B}_2$ is the edge-to-triangle incidence matrix and so on. Figure 2.3 illustrates the boundary operators of a rank-2 simplicial complex.

The boundary operators of a simplicial complex satisfy the key topological identity that reflects the fact that the boundary of a boundary is empty:

$$\mathbf{B}_k \mathbf{B}_{k+1} = 0. \tag{2.21}$$

Hodge Laplacians: The Hodge Laplacian is a natural generalization of the graph Laplacian to higher-order topologies. For a simplicial complex of order K, the k-th Hodge Laplacian $L_k \in \mathbb{R}^{N_k \times N_k}$ is defined as:

$$\mathbf{L}_k = \mathbf{B}_k^\top \mathbf{B}_k + \mathbf{B}_{k+1} \mathbf{B}_{k+1}^\top, \tag{2.22}$$

where:

- $\mathbf{L}_{k,d} = \mathbf{B}_k^\top \mathbf{B}_k$ is the lower Laplacian, encoding relationships between k-simplices via shared $(k-1)$-faces (lower adjacency),
- $\mathbf{L}_{k,u} = \mathbf{B}_{k+1} \mathbf{B}_{k+1}^\top$ is the upper Laplacian, encoding relationships between k-simplices via shared $(k+1)$-cofaces (upper adjacency).

The 0-th Hodge Laplacian reduces to the standard graph Laplacian capturing the upper adjacency of nodes through edges:

$$\mathbf{L}_0 = \mathbf{B}_1 \mathbf{B}_1^\top. \tag{2.23}$$

When $k = K$, the order of the simplicial complex, only the lower Laplacian remains as there are no co-faces of K-simplices. Therefore, the K-th Hodge Laplacian reduces to:

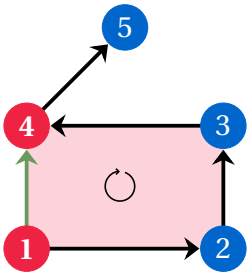
$$\mathbf{L}_K = \mathbf{B}_K^\top \mathbf{B}_K. \tag{2.24}$$

CELL COMPLEXES

While simplicial complexes provide a structured way to represent higher-order relationships, they are restricted by the face inclusion property. This means that they are restricted to structures such as triangles, tetrahedrons, etc. which can be overly rigid for certain applications.

Cell complexes offer a more flexible alternative by allowing general shapes such as polygons as cells, without requiring all lower-dimensional faces to be explicitly included. Each cell represents a basic geometric region (e.g., a node, edge, face) and can be combined with others to form a valid topological space.

In this work, our focus is on regular cell complexes, which satisfy properties like local finiteness, structured intersection, and homeomorphism, ensuring a well-behaved topological structure [62]. Like in simplicial complexes, we can define signals on k-dimensional cells as in Equation 2.18, represent relationships between them using incidence matrices using Equation 2.20, and construct the Hodge Laplacians using Equation 2.22 to analyze signal diffusion across the complex. Figure 2.4 illustrates a rank-2



$$\mathbf{B}_1 = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B}_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 2.4: An example cell complex of order 2 with its corresponding node-to-edge ($\mathbf{B_1}$) and edge-to-face ($\mathbf{B_2}$) incidence matrices. The upper and lower neighborhood of a specific edge (green) are marked in red. The signals on nodes, edges and faces are omitted for clarity.

cell complex with the corresponding boundary operators and the upper and lower adjacencies of a specific (green) edge marked in red. Simplicial complexes are a special case of cell complexes where all cells are simplices, and the face inclusion property holds strictly.

2.2.2. HODGE DECOMPOSITION AND TOPOLOGICAL FOURIER TRANSFORM

Hodge Decomposition: The Hodge decomposition theorem provides a framework to decompose signals defined on higher-order structures, such as simplicial and cell complexes. For ease of explanation, we use the term complex to refer to both simplicial and cell complexes in the notation below. The Hodge decomposition splits the signal space of the k-dimensional complex signals $\mathbb{R}^{N_k}$ as:

$$\mathbb{R}^{N_k} = \text{im}(\mathbf{B}_k^\top) \oplus \text{im}(\mathbf{B}_{k+1}) \oplus \ker(\mathbf{L}_k) \tag{2.25}$$

where $\text{im}(\cdot)$ and $\ker(\cdot)$ denote the image and kernel spaces of a matrix, respectively, and $\oplus$ denotes the direct sum of vector spaces. The three orthogonal subspaces are interpreted as follows:

- $\text{im}(\mathbf{B}_k^\top)$: The gradient space, representing the component of the k-complex signal induced by differences in $(k-1)$-signals of the $(k-1)$-dimensional faces.
- $\text{im}(\mathbf{B}_{k+1})$: The curl space, representing the component of the k-complex signal induced by the $(k+1)$-signals of circulating $(k+1)$-dimensional cofaces.
- $\ker(\mathbf{L}_k)$: The harmonic space, representing the component of the k-complex signal that is neither in the gradient space nor in the curl space (in the kernel space of both).

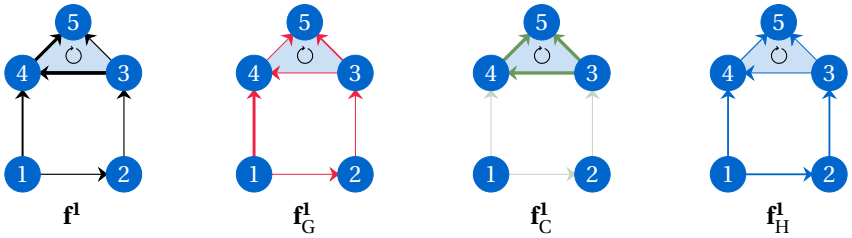


Figure 2.5: Hodge Decomposition of edge flow $\mathbf{f}$ into the gradient ($\mathbf{f}^1_G$), curl ($\mathbf{f}^1_C$) and harmonic ($\mathbf{f}^1_H$) components with $\mathbf{f}^{(1)} = \mathbf{f}^1_G + \mathbf{f}^1_C + \mathbf{f}^1_H$

.

A k-complex signal $\mathbf{f}^k$ can thus be decomposed as:

$$\mathbf{f}^k = \mathbf{f}^k_G + \mathbf{f}^k_C + \mathbf{f}^k_H \tag{2.26}$$

where $\mathbf{f}^k_G \in \text{im}(\mathbf{B}_k^\top)$, $\mathbf{f}^k_C \in \text{im}(\mathbf{B}_{k+1})$, and $\mathbf{f}^k_H \in \ker(\mathbf{L}_k)$ are the gradient, curl, and harmonic components, respectively.

When $k = 1$, the signals $\mathbf{f}^1$ represent edge flows, and the Hodge decomposition provides insights into the decomposition of these edge signals. The divergence operator $\mathbf{B}_1$ maps an edge flow $\mathbf{f}^1$ to a node-level signal $\mathbf{f}^0 = \mathbf{B}_1 f^1$, representing the net inflow or outflow at each node. A flow is considered divergence-free if $\mathbf{B}_1\mathbf{f}^1 = 0$. The gradient operator $\mathbf{B}_1^\top$ induces edge flows by computing differences between node signals, forming the gradient component as $\mathbf{f}^1_G = \mathbf{B}_1^\top \mathbf{f}^0$. Similarly, the curl operator $\mathbf{B}_2^\top$ measures the circulation of edge flows around faces (triangles for simplicial complexes and polygons for cell complexes) and induces face signals as $\mathbf{f}^2 = \mathbf{B}_2^\top \mathbf{f}^1$. A flow is curl-free if $\mathbf{B}_2^\top \mathbf{f}^1 = 0$. The curl component of the edge flow is given by $\mathbf{f}^1_C = \mathbf{B}_2 \mathbf{f}^2$, representing local circulations. The harmonic component $\mathbf{f}_H \in \ker(\mathbf{L}_1)$ satisfies $\mathbf{L}_1\mathbf{f}_H = 0$, indicating it is both divergence- and curl-free. Figure 2.5 illustrates the decomposition of edge signals on a simplicial complex into gradient, curl, and harmonic components.

Topological Fourier Transform: Hodge Laplacians, like graph Laplacians, are positive semi-definite matrices and admit an eigendecomposition, enabling a spectral representation of signals defined on complexes. The k-th Hodge Laplacian $\mathbf{L}_k$ is decomposed as:

$$\mathbf{L}_k = \mathbf{U}_k \mathbf{\Lambda}_k \mathbf{U}_k^\top \tag{2.27}$$

where $\mathbf{U}_k$ is an orthonormal matrix containing the eigenvectors of $\mathbf{L}_k$, and $\mathbf{\Lambda}_k$ is a diagonal matrix of the eigenvalues. The Fourier Transform of a k-complex signal $\mathbf{f}^k$ is defined by projecting it onto the eigenspace of L_k:

$$\tilde{\mathbf{f}}^k = \mathbf{U}_k^\top \mathbf{f}^k \tag{2.28}$$

where $\tilde{\mathbf{f}}^k \in \mathbb{R}^{N_k}$ contains the spectral coefficients. The inverse Fourier Transform reconstructs the signal as:

$$\mathbf{f}^k = \mathbf{U}_k \tilde{\mathbf{f}}^k \tag{2.29}$$

An important property of the Hodge decomposition is it's alignment with the eigenspace structure of the Hodge Laplacian. That is, for $\mathbf{L}_k = \mathbf{U}_k \mathbf{\Lambda}_k \mathbf{U}_k^\top$, we can decompose the eigenspace as $\mathbf{U}_k = [\mathbf{U}_G \ \mathbf{U}_C \ \mathbf{U}_H]$ and $\mathbf{\Lambda}_k = \text{diag}(\mathbf{\Lambda}_G, \mathbf{\Lambda}_C, \mathbf{\Lambda}_H)$, where each group of eigenvectors corresponds to a distinct subspace:

- The gradient space, $\text{im}(\mathbf{B}_k^\top)$, is spanned by eigenvectors $\mathbf{U}_G$ associated with the positive eigenvalues $\mathbf{\Lambda}_G$ of the lower Laplacian $\mathbf{L}_{k,d}$.
- The curl space, $\text{im}(\mathbf{B}_{k+1})$, is spanned by eigenvectors $\mathbf{U}_C$ associated with the positive eigenvalues $\mathbf{\Lambda}_C$ of the upper Laplacian $\mathbf{L}_{k,u}$.
- The harmonic space, $\ker(\mathbf{L}_k)$, is spanned by eigenvectors $\mathbf{U}_H$ corresponding to zero eigenvalues $\mathbf{\Lambda}_H$ of $\mathbf{L}_k$.

Given these components, a k-complex signal $\mathbf{f}^k$ can be mapped into its subspace embeddings as:

$$\tilde{\mathbf{f}^k}_H = \mathbf{U}_H^\top \mathbf{f}^k, \quad \tilde{\mathbf{f}^k}_G = \mathbf{U}_G^\top \mathbf{f}^k, \quad \tilde{\mathbf{f}^k}_C = \mathbf{U}_C^\top \mathbf{f}^k \tag{2.30}$$

These embeddings provide a compact and interpretable representation of the signal in the spectral domain. The Fourier transform after the Hodge decomposition can thus be written as:

$$\tilde{\mathbf{f}^k} = \begin{bmatrix} \tilde{\mathbf{f}^k}_H^\top & \tilde{\mathbf{f}^k}_G^\top & \tilde{\mathbf{f}^k}_C^\top \end{bmatrix}^\top \tag{2.31}$$

2.2.3. TOPOLOGICAL FILTERS

The section above described how Hodge decomposition and the Topological Fourier Transform provide a spectral basis that decomposes the eigen-space of the Hodge Laplacian into gradient, curl, and harmonic components. Topological filters leverage this decomposition to selectively amplify or suppress these components, based on the downstream task.

Analogous to GCFs, topological filters generalize the notion of learning frequency modulation to topological domains. They act on k-complex signals by applying a learnable transformation that modulates the gradient, curl, and harmonic subspaces differently. Given the k-th Hodge Laplacian L_k, topological filters for k-complex signals are defined as:

$$\mathbf{H}_k := \mathbf{H}(\mathbf{L}_{k,d}, \mathbf{L}_{k,u}) = h_0 I + \sum_{l_1=1}^{L_1} \alpha_{l_1} (\mathbf{B}_k^\top \mathbf{B}_k)^{l_1} + \sum_{l_2=1}^{L_2} \beta_{l_2} (\mathbf{B}_{k+1} \mathbf{B}_{k+1}^\top)^{l_2}, \tag{2.32}$$

The parameters include filter coefficients h_0, $\alpha = [\alpha_1, \ldots, \alpha_{L_1}]^\top$, $\beta = [\beta_1, \ldots, \beta_{L_2}]^\top$, and filter orders L_1, L_2. Figure 2.6 illustrates the shift-and-sum lower and upper convolution operations in topological filters described in the equation above for the 1-Hodge Laplacian $\mathbf{L}_1$.

Similar to GCFs, topological filters are a weighted combination of shifted signals. However, topological shifting incorporates signals from both the lower and upper neighborhoods through the lower and upper Hodge Laplacians. By using separate filter coefficients and orders for the lower and upper Laplacians, the topological filter can modulate the gradient and curl spaces independently, increasing it's expressive capacity [81].

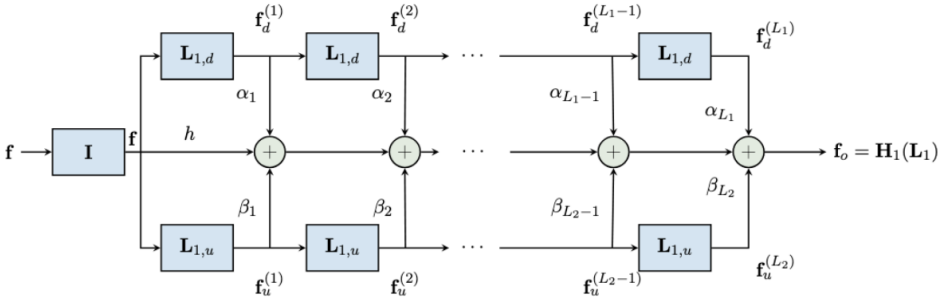


Figure 2.6: The shift-and-sum upper and lower convolution operations in a topological filter for the 1-Hodge Laplacian $\mathbf{L}_1$.

The filter in Equation 2.32 shifts the signal L_1 times over the lower neighborhoods and L_2 times over the upper neighborhoods. The shifted signals are then summed according to the corresponding filter coefficients. Similar to the localized graph signal shifting, described in 2.5, topological shifting is localized in the immediate upper and lower neighborhood. Moreover, topological shifts in filters can be recursively applied, as in Equation 2.6 to obtain filter outputs localized in the L_1-hop lower neighborhood and L_2-hop upper neighborhood. This makes topological filters localized operations that are shift-invariant [62, 81].

Spectral Domain Interpretation: In the spectral domain, topological filters modulate the different frequency components of the Hodge decomposition using pointwise multiplication. The filter frequency response is given by:

$$h(\lambda_i) = \begin{cases} h_0, & \text{for } \lambda_i \in \Lambda_H, \\ h_0 + \sum_{l_1=1}^{L_1} \alpha_{l_1} \lambda_i^{l_1}, & \text{for } \lambda_i \in \Lambda_G, \\ h_0 + \sum_{l_2=1}^{L_2} \beta_{l_2} \lambda_i^{l_2}, & \text{for } \lambda_i \in \Lambda_C, \end{cases} \tag{2.33}$$

This general formulation of topological filters and the spectral interpretation applies to both simplicial complexes and cell complexes.

2.2.4. TOPOLOGICAL NEURAL NETWORKS

Topological Neural Networks (TNNs) are multi-layered architectures where each layer consists of a collection of topological filters followed by point-wise non-linearities. These networks generalize GCNNs from graphs to higher-dimensional topological structures by leveraging Hodge Laplacians and incorporating both lower and upper adjacencies.

A single TNN layer transforms a set of k-complex features into a new set of output features. F_0^k corresponds to the dimensionality of the initial k-complex signal, and the number of output channels at layer ℓ, denoted F_ℓ^k, is a hyperparameter representing the number of topological filter banks.

Let the input to the ℓ-th layer for the k-complex be denoted as:

$$\mathbf{X}^{k,(\ell-1)} = [\mathbf{x}_1^{k,(\ell-1)}, \mathbf{x}_2^{k,(\ell-1)}, \ldots, \mathbf{x}_{F_{\ell-1}^k}^{k,(\ell-1)}] \in \mathbb{R}^{N_k \times F_{\ell-1}^k}, \tag{2.34}$$

Each column $\mathbf{x}_g^{k,(\ell-1)} \in \mathbb{R}^{N_k}$ represents the values of the g-th feature across all k-cells.

The layer produces F_ℓ^k output feature channels. For each output feature $f = 1, \ldots, F_\ell^k$, the intermediate signal before applying the activation is:

$$\mathbf{z}_f^{k,(\ell)} = \sum_{g=1}^{F_{\ell-1}^k} \mathbf{H}_{fg}^{k,(\ell)} \mathbf{x}_g^{k,(\ell-1)}, \tag{2.35}$$

where $\mathbf{H}_{fg}^{k,(\ell)}$ is a learnable topological filter that models the contribution of the g-th input feature to the f-th output feature. Following the definition in Equation 2.32, each filter is written as:

$$\mathbf{H}_{fg}^{k,(\ell)} = h_0 I + \sum_{l_1=1}^{L_1} \alpha_{fg,l_1}^{(\ell)} (\mathbf{B}_k^\top \mathbf{B}_k)^{l_1} + \sum_{l_2=1}^{L_2} \beta_{fg,l_2}^{(\ell)} (\mathbf{B}_{k+1} \mathbf{B}_{k+1}^\top)^{l_2}, \tag{2.36}$$

The output feature is then obtained by applying a non-linear activation function $\sigma(\cdot)$:

$$\mathbf{x}_f^{k,(\ell)} = \sigma(\mathbf{z}_f^{k,(\ell)}), \quad \forall f = 1, \ldots, F_\ell^k, \tag{2.37}$$

and the final output matrix is:

$$\mathbf{X}^{k,(\ell)} = [\mathbf{x}_1^{k,(\ell)}, \mathbf{x}_2^{k,(\ell)}, \ldots, \mathbf{x}_{F_\ell^k}^{k,(\ell)}] \in \mathbb{R}^{N_k \times F_\ell^k}. \tag{2.38}$$

A TNN is formed by stacking multiple such layers for each k-complex. For a TNN with L layers, the final output at level k is:

$$\mathbf{x}^{k,(L)} = \sigma\left(\mathscr{H}^{(L)}(\mathbf{L}_{k,d}, \mathbf{L}_{k,u}) \mathbf{x}^{k,(L-1)}\right), \tag{2.39}$$

where $\mathscr{H}^{(L)}$ denotes the topological filter bank at layer L. This formulation describes the most general form of a TNN, where each k-complex is updated independently at every layer. Variations that allow for coupling between different k-complexes have been proposed to improve expressivity and are discussed in Section 3.1.

2.2.5. CONSTRUCTING TOPOLOGIES FROM GRAPHS

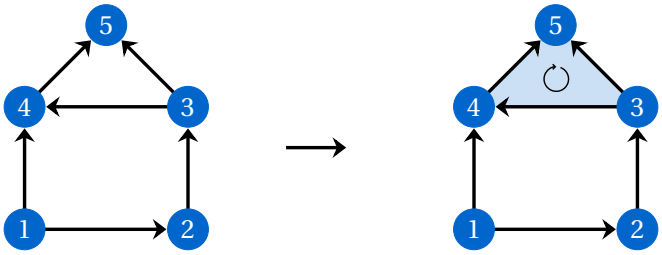
Although many physical systems inherently exhibit higher-order interactions, the complexity of modeling these relationships and the limitations of current experimental methodologies often restrict the availability of higher-order data. As a result, such data is frequently constructed by transforming graph-based representations through a process known as lifting [35]. The lifting procedure consists of two components: structure lifting, which transforms the graph structure into a higher-order topological structure, and feature lifting, which maps graph signals onto signals defined over higher-order simplices or cells.

STRUCTURE LIFTING

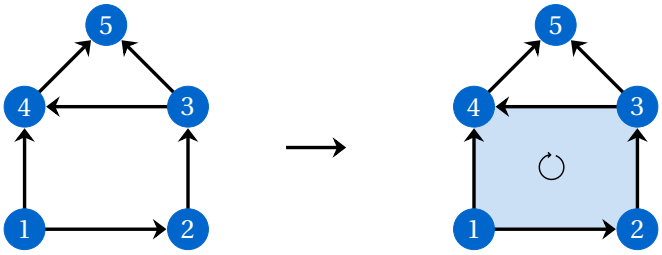
Structure lifting refers to the process of transforming a graph structure into a higher-order topological space such as a simplicial or cell complex. Formally, structure lifting is defined as a map

$$\psi_X : \mathscr{G} \to \mathscr{X}, \tag{2.40}$$

where $\mathscr{G}$ denotes the graph structure, and $\mathscr{X}$ denotes the lifted topological structure (simplices or cells).



(a) Lifting a graph to a simplicial complex using Clique Lifting.



(b) Lifting a graph to a cell complex using Cycle Lifting.

Figure 2.7: Lifting a graph to a simplicial and cell complex using clique lifting and cycle lifting respectively.

Graph to Simplicial Complex: One commonly used approach to lift graphs to simplicial complexes is clique lifting, in which all the cliques in a graph are mapped to simplices. A clique in a graph is a subset of nodes in which every pair of distinct nodes is connected by an edge. That is, a clique forms a complete subgraph. In the lifting procedure, a clique of size $k+1$ gives rise to a k-dimensional simplex in the resulting simplicial complex. This construction yields a clique complex, where nodes correspond to 0-simplices, edges (2-cliques) to 1-simplices, triangles (3-cliques) to 2-simplices, and so on. Although alternative structure lifting strategies exist [35], this work focuses exclusively on clique lifting.

Graph to Cell Complex: Graphs can also be lifted into cell complexes through cycle-based lifting, which provides a more flexible representation of higher-order interactions by incorporating general polygonal structures. In this approach, cycles (closed paths) in the graph are interpreted as boundaries of higher-dimensional cells. Each identified cycle is associated with a 2-cell whose boundary aligns exactly with the cycle, and the nodes and edges of the original graph are retained as the 0- and 1-cells, respectively.

Both clique and cycle-based lifting procedures are skeleton-preserving [10], meaning they retain the original node and edge structure of the graph while augmenting it with higher-order relationships. Figure 2.7 illustrates structure lifting from a graph to a simplicial complex via clique lifting, and to a cell complex via cycle-based lifting.

FEATURE LIFTING

Feature lifting refers to the process of transforming or transferring features defined on a graph $\mathcal{G}$ to the corresponding lifted topological space $\mathcal{X}$, in a way that is consistent with the structural lifting map $\psi_X : \mathcal{G} \to \mathcal{X}$.

Let $F_G : \mathcal{G} \to \mathbb{R}^N$ define the signal space over the nodes or edges of the graph. For the lifted topological space, let $F_X^k : \mathcal{X}^k \to \mathbb{R}^{N_k}$ denote the signal space defined on the k-dimensional cells (simplices or cells) of $\mathcal{X}$, for $k \in \{0, 1, \ldots, K\}$.

Feature lifting is then defined as a transformation:

$$\psi_F^k : \mathcal{G} \times F_G \to F_X^k, \tag{2.41}$$

such that for all $g \in \mathcal{G}$,

$$F_X^k(\psi_X(g)) = \psi_F^k(F_G(g)). \tag{2.42}$$

In other words, for each complex order k, the feature of a k-dimensional simplex or cell is obtained by combining the features of the graph elements that constitute it. For instance, the feature of a 1-simplex (edge) may be constructed from the features of the two nodes it connects, while the feature of a 2-cell (triangle or polygon) may depend on the features of the nodes or edges that form its boundary. This process ensures that the lifted features meaningfully reflect the structure and attributes of the underlying graph components.

In practice, feature lifting can be rule-based, such as averaging or concatenating features of lower-dimensional elements or learned using neural networks that generate higher-order features from constituent components [7].

2.3. SPATIO-TEMPORAL MODELING USING PRODUCT SPACES

The previous sections introduced signal representations and filtering operations on static graphs and topological structures. However, multivariate time series data is dynamic, exhibiting temporal variation in the signals defined over their underlying structure. That is, the signals associated with the graph nodes, edges, or higher-dimensional entities evolve over time. This motivates the need for frameworks like the product space that can capture both spatial dependencies (through the graph or topological structure) and temporal dependencies (through time series dynamics).

This section begins by formally introducing time-varying signals in Section 2.3.1 as a framework for representing multivariate time series data where variables exhibit a relational structure. We then define the spatio-temporal modeling problem and highlight the need to jointly capture spatial and temporal dependencies in Section 2.3.2. This is followed by a description of the product graph framework, a widely used approach for modeling spatio-temporal data by constructing a graph in the product space in Section 2.3.3. Finally, we discuss the extension of product graphs to the higher-order product spaces through the product cell complex framework in Section 2.3.4.

2.3.1. TIME-VARYING GRAPH SIGNALS

Multivariate time series data can be expressed as time-varying graph signals that extend static graph signals by associating each node (or edge) with a sequence of values over

time. Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with $N = |\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges. A time-varying signal on the nodes is a function $f_t^v : \mathcal{V} \times \mathbb{Z} \to \mathbb{R}$, which assigns a scalar value to each node at each discrete time step. The signal at time t is denoted by the vector $\mathbf{x}_t^v \in \mathbb{R}^N$, where $[\mathbf{x}_t^v]_i$ represents the value at node i and time t.

Similarly, a time-varying signal on the edges is defined as a function $f_t^e : \mathcal{E} \times \mathbb{Z} \to \mathbb{R}$, assigning a scalar value to each edge at each time step. The edge signal at time t is denoted by $\mathbf{x}_t^e \in \mathbb{R}^{|\mathcal{E}|}$, where $[\mathbf{x}_t^e]_j$ denotes the value at edge j and time t. Although each node or edge can have multiple features at each time step, we assume scalar time-varying signals for both nodes and edges for notational simplicity in the remainder of this section.

The underlying graph structure itself may be either static or dynamic. In the dynamic case, we have a sequence of graphs $\{\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)\}$, where the edge set $\mathcal{E}_t$ and corresponding adjacency matrix A_t may change over time. In this work, however, we focus on the setting where the graph structure $\mathcal{G}$ remains static, and only the signals vary through time.

This framework naturally generalizes to time-varying signals on higher-order topological structures, such as simplicial and cell complexes, where signals may be defined on nodes, edges, faces, or higher-dimensional cells over time.

2.3.2. SPATIO-TEMPORAL MODELING

Spatio-temporal modeling of multivariate time series data takes advantage of the spatial, temporal and spatio-temporal relationships present in the data. Figure 2.8 shows these three types of dependencies in multivariate time series data.

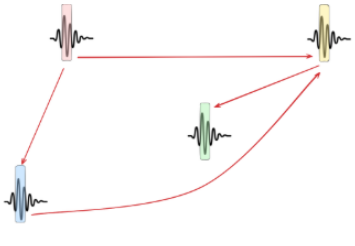
Capturing temporal dependencies allows the model to learn patterns over time, such as trends, periodicities, or delays. Spatial dependencies, on the other hand, capture how different nodes influence each other through the graph structure through diffusion or propagation dynamics. Since these two forms of correlation are often interdependent, effective spatio-temporal modeling requires joint reasoning over both domains. This gives rise to spatial, temporal and spatio-temporal dependencies in multivariate data

While a variety of methods for modeling spatio-temporal data are discussed in Section 3.3, the following two sections focus on approaches that model joint spatio-temporal dependencies by operating on the spatio-temporal product space, first through product graphs and then through product cell complexes.

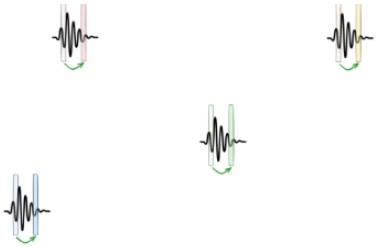
2.3.3. PRODUCT GRAPHS

The product graph framework provides a unified representation for spatio-temporal data by combining spatial and temporal graphs through Kronecker products [46]. Let the spatial graph be defined as $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$, where $\mathcal{V}_S$ denotes the set of $N = |\mathcal{V}_S|$ spatial nodes and $\mathbf{S}_S \in \mathbb{R}^{N \times N}$ is the associated GSO encoding spatial connectivity between spatial edges $\mathcal{E}_S$. At each discrete time step $t \in \{1, \ldots, T\}$, a node-level graph signal $\mathbf{x}_t^v \in \mathbb{R}^N$ is defined over $\mathcal{V}_S$, and the sequence $\{\mathbf{x}_t^v\}_{t=1}^T$ forms a time-varying graph signal.

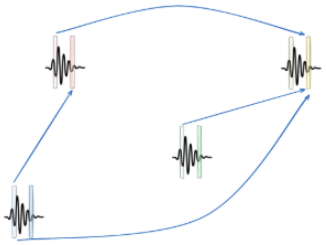
Temporal dependencies are captured by a temporal graph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$, where $\mathcal{V}_T = \{1, \ldots, T\}$ indexes the time steps and $\mathcal{E}_T \subseteq \mathcal{V}_T \times \mathcal{V}_T$ defines edges between them. An edge $(t, t') \in \mathcal{E}_T$ represents a dependency between time steps t and t', and the structure is encoded by the temporal shift operator $\mathbf{S}_T \in \mathbb{R}^{T \times T}$. In this work, we model $\mathbf{S}_T$ as a line



(a) Spatial dependencies between variables at a fixed time slice.



(b) Temporal dependencies within variables across time slices.



(c) Spatio-temporal dependencies across variables and time slices.

Figure 2.8: Spatial, temporal, and spatio-temporal dependencies in multivariate time series data.

graph, where each time step is connected to its immediate predecessor to capture first-order temporal dependencies.

The resulting product graph is defined as $\mathcal{G}_\circ = \mathcal{G}_T \circ \mathcal{G}_S = (\mathcal{V}_\circ, \mathcal{E}_\circ, \mathbf{S}_\circ)$, where $\mathcal{V}_\circ = \mathcal{V}_S \times \mathcal{V}_T$ denotes the set of spatio-temporal nodes and $|\mathcal{V}_\circ| = NT$. Each node $(i,t) \in \mathcal{V}_\circ$ represents spatial node i at time t. Edges in $\mathcal{E}_\circ$ are induced by both spatial and temporal adjacencies, and the combined structure is encoded by the spatio-temporal shift operator $\mathbf{S}_\circ \in \mathbb{R}^{NT \times NT}$. Intuitively, product graphs represent the space-time relational structure of multivariate time series data.

TYPES OF PRODUCT GRAPHS

Different formulations of $\mathbf{S}_\circ$ correspond to different types of product graphs [61], visualized in Figure 2.9:

- **Kronecker Product:** Connects nodes in a spatio-temporal manner only if both their spatial and temporal components are adjacent. The GSO is given by Kronecker product $\mathbf{S}_\otimes = \mathbf{S}_T \otimes \mathbf{S}_S$ and the product graph is represented as $\mathcal{G}_\otimes = \mathcal{G}_T \otimes \mathcal{G}_S = (\mathcal{V}_\otimes, \mathcal{E}_\otimes, \mathbf{S}_\otimes)$
- **Cartesian Product:** Captures both the spatial and temporal adjacencies. The GSO is given by Kronecker product $\mathbf{S}_\times = \mathbf{S}_T \otimes \mathbf{I}_N + \mathbf{I}_T \otimes \mathbf{S}_S$ and the product graph is represented as $\mathcal{G}_\times = \mathcal{G}_T \times \mathcal{G}_S = (\mathcal{V}_\times, \mathcal{E}_\times, \mathbf{S}_\times)$

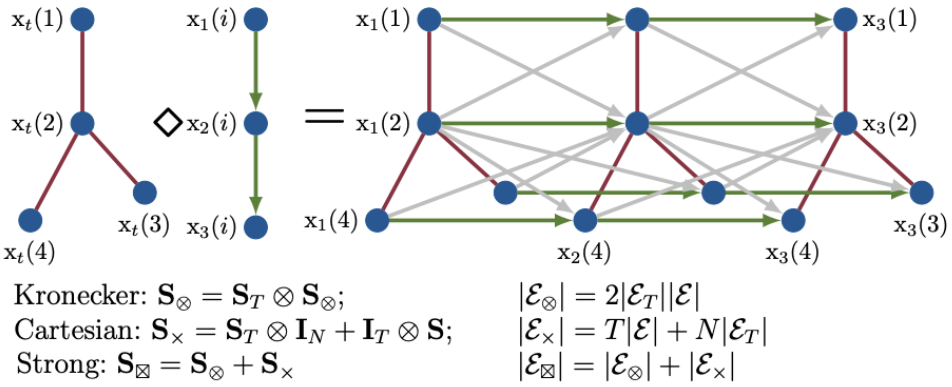


Figure 2.9: The different types of product graph for spatio-temporal modeling, adapted from [59]

- **Strong Product:** Combines both Kronecker and Cartesian interactions. The GSO is the sum $\mathbf{S}_{\boxtimes} = \mathbf{S}_{\times} + \mathbf{S}_{\otimes}$ and the product graph is represented as $\mathcal{G}_{\boxtimes} = \mathcal{G}_T \boxtimes \mathcal{G}_S = (\mathcal{V}_{\boxtimes}, \mathcal{E}_{\boxtimes}, \mathbf{S}_{\boxtimes})$.
- **Parametric Product:** Generalizes all the types above with learnable scalars $\{s_{ij}\}$ controlling the contributions of spatial and temporal interactions:

$$\mathbf{S}_{\diamondsuit} = \sum_{i=0}^{N} \sum_{j=0}^{T} s_{ij} (\mathbf{S}_T^i \otimes \mathbf{S}_S^j), \tag{2.43}$$

  where setting all $s_{ij} = 1$ recovers the strong product graph. When s_{ij} are learnable parameters, the spatio-temporal coupling can be optimized end-to-end with the downstream task. The allows parametric product graphs to learn the importance of different types of pairwise relationships in multivariate time series data.

FILTERING ON PRODUCT GRAPHS

Once the product graph $G_{\circ} = G_T \circ G_S$ is defined, one can directly apply graph convolutional filtering on this unified structure. Since these filters filter across space and time, they are called Graph-Time Convolutional Filters (GTCF) [59] and are defined as:

$$\mathbf{H}_{\circ} = \sum_{k=0}^{K} h_k \mathbf{S}_{\circ}^k, \tag{2.44}$$

where the shift-and-sum operation is performed over the spatio-temporal graph shift operator $\mathbf{S}_{\circ}$, and $\{h_k\}$ are learnable filter coefficients.

Analogous to GCNNs, which stack multiple GCFs with nonlinearities, GTCNNs [59] build spatio-temporal convolutional architectures by stacking GTCFs. These models learn expressive representations of space-time dynamics in time-varying graph signals

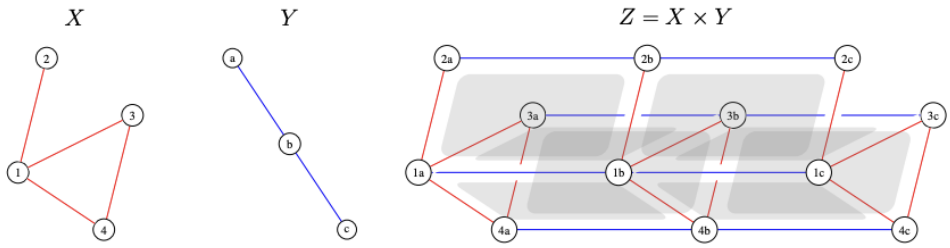


Figure 2.10: The product cell complex **Z** formed as a Cartesian product of spatial and temporal simplicial complexes **X** and **Y** , respectively. Adapted from [59]

by applying localized, shared filters over the product graph domain. Furthermore, by employing parametric product graphs as defined in Equation 2.43, GTCNNs can learn the relative importance of different edge types, thereby enabling the model to learn an appropriate inductive bias directly from the data, rather than relying on predefined structural assumptions.

2.3.4. PRODUCT CELL COMPLEX

While product graphs effectively model time-varying signals on the nodes of a graph, they only model time-varying signals on nodes and cannot represent temporal dynamics on edges or higher-order structures. As a result, they cannot model higher-order relationships in space and time. To address this limitation, [57] introduced a higher-order topological space represented by a cell complex that jointly encodes spatial and temporal structures by taking the product of spatial and temporal simplicial complexes.

Let **X** and **Y** be spatial and temporal simplicial complexes, respectively. The product cell complex is defined as:

$$\mathbf{Z} = \mathbf{X} \boxtimes \mathbf{Y},$$

where each k-cell in **Z** arises from the product of a spatial k_s-simplex and a temporal k_t-simplex, satisfying $k = k_s + k_t$.

For example, the product of two vertices yields another vertex (product of 0-simplex and 0-simplex yields a 0-cell), and the product of a vertex and an edge yields an edge (product of 0-simplex and 1-simplex yields a 1-cell). However, the product of two edges does not produce a 2-simplex, but rather a 2-cell shaped like a filled rectangle (product of 1-simplex and 1-simplex yields a 2-cell), as shown in Figure 2.10.

To perform signal processing on this product complex, the authors of [57] derive its boundary operators by combining the spatial and temporal boundary matrices via Kronecker sums and products. This construction leads directly to Hodge Laplacians that capture interactions across both space and time dimensions for the constructed cell complex. However, they propose this as a theoretical framework and do not construct a TNN on the product cell complex.

It is important to note that $\mathscr{X}_P$ forms a regular cell complex in the product space rather than a simplicial complex. Retaining a simplicial structure requires subdividing

the constructed cells into triangles, which is not done in [57]. In contrast, our work ensures that the constructed product complex remains simplicial and construct a TNN on the resulting simplicial product complex. We argue that constructing a simplicial product complex rather than a cellular product complex has various benefits discussed in 4.7 that benefit higher order spatio-temporal modeling.

2.4. CONCLUSION

This chapter began with a review of classical graph signal processing, covering graph filters, their spectral properties, and GCNNs. We then motivated the move to higher-order topological structures such as simplicial and cell complexes that capture multi-variable relationships that graphs cannot. Within this setting, we introduced Hodge decomposition and topological Fourier analysis as tools for decomposing signals on topological domains, and defined both topological filters and TNNs, extending graph filters and GCNNs to the topological domain.

Next, we examined time-varying graph signals and showed how product graphs model temporal dynamics for node-level data. To overcome their limitations in representing higher-order interactions, we introduced product cell complexes, which enable spatio-temporal modeling on higher-order topological structures. Finally, we emphasized a key difference between product cell complexes and our proposed approach. While the product cell complexes yields regular cell structures in the product space, our method explicitly enforces a simplicial structure, a design choice we hypothesize offers a richer and more flexible representation of higher-order space-time relations.

3

Related Work

This chapter reviews existing work relevant to our proposed framework that captures higher-order relationships in multivariate time series data. We begin in Section 3.1 by reviewing existing works that use TNNs to model higher-order structure using simplicial or cell complexes in the spatial domain. We then delve deeper into the various multivariate time series modeling paradigms discussed in Chapter 1. Section 3.2 surveys *structure-unaware* frameworks that do not incorporate any relational structure while modeling multivariate time series data, reviewing both classical and deep learning-based methods. In Section 3.3, we review *graph-aware* spatio-temporal models that leverage relational priors to jointly capture spatial and temporal dependencies. Finally, in Section 3.4, we examine recent efforts that extend topological modeling to spatio-temporal domains via product spaces, enabling *topologically-aware* spatio-temporal learning. We conclude the chapter with a discussion in Section 3.5. Figure 3.1 provides a taxonomy of the literature we survey in this chapter.

3.1. Higher-Order Spatial Modeling

This section reviews prior work on developing signal processing foundations and neural network architectures for higher-order spatial modeling with simplicial and cell complexes.

Simplicial Complexes

Simplicial complexes were initially studied in the context of signal processing in [5, 9], paving the way for the development of the spectral theory of simplicial signals [4] and simplicial filtering techniques [80] using the Hodge Laplacian. Early applications included edge flow denoising [64], interpolation [40], and topology inference [4]. Building upon this foundation, HodgeNet [56] introduced the first integration of Hodge Theory with deep learning by leveraging the lower Laplacian $L_{1,\ell}$ for learning convolutions on edge features.

Following this, several works extended simplicial processing to higher-dimensional

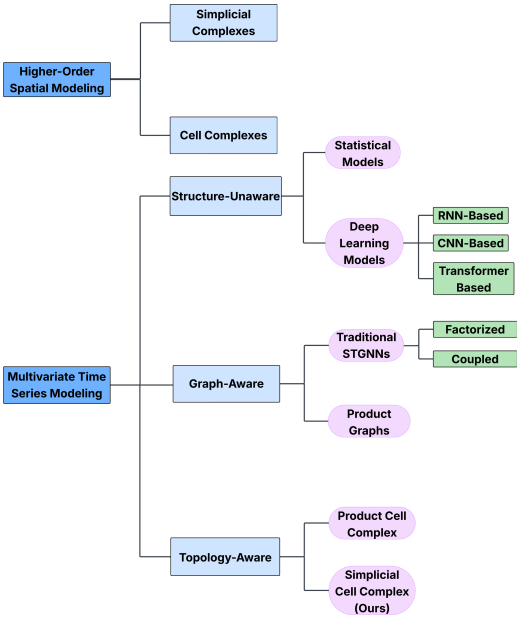


Figure 3.1: A taxonomy of our literature review covering higher-order spatial modeling and the three paradigms for multivariate time series modeling: structure-unaware, graph-aware and topology-aware frameworks.

structures. Simplicial Neural Networks (SNN) [26] and SCCONV [15] generalized the convolution framework of [56] to support signals on simplices of higher ranks, incorporating both upper and lower Laplacians. Compared to SNNs, SCCONVs incorporated features from simplices of adjacent ranks by using boundary and co-boundary operators. [11] proposed Message Passing Simplicial Networks (MPSNs) that adapted the message passing framework to simplicial complexes by aggregating upper and lower neighborhoods as well as neighborhoods of adjacent ranks. They also introduced a Simplicial Weisfeiler-Lehman (SWL) test to differentiate non-isomorphic graphs and demonstrated that MPSNs are provably more powerful and expressive than GNNs. The MPSN framework also unified and generalized previous approaches such as [15] and [54] with specific aggregate and update functions in the message passing framework.

Building on these developments, SCNNs [79] introduced a framework for multi-hop message passing, enabling separate processing of upper and lower neighborhoods to enhance model flexibility. This multi-hop convolution framework was further extended in SCCNNs [78] through inter-simplicial coupling, which facilitates interactions by propagating signals from lower and upper neighborhoods in each layer.

Recent works have also explored self-attention mechanisms for simplicial complexes, allowing models to adaptively learn the importance of different neighborhoods. Various works, such as SAN [8], SAT [33], and SGAT [45], extend traditional message passing by incorporating attention-based weighting schemes that selectively amplify or suppress information flow across simplicial structures. While SGAT applies attention mechanisms

using the upper Laplacian, SAT expands this approach by considering interactions between simplices of different orders. SAN further generalizes these ideas by introducing a flexible attention framework that dynamically adjusts weights across multiple simplicial neighborhoods.

CELL COMPLEXES

Similar to simplicial complexes, foundational work in signal processing has also been established for cell complexes. Early research introduced linear filtering techniques for cell complexes [55], leveraging the Hodge Laplacian as the shift operator and outlining how these filters could be extended into neural network architectures. Building on this, [63] proposed topological filters tailored for cell complexes, enabling independent processing of the different Hodge subspaces, similar to previous work in simplicial signal processing [80]. A more theoretical perspective on neural network design for cell complexes was introduced in [34], establishing a general framework for learning on these higher-order structures. The first practical implementation of such models was demonstrated in [10], where the authors showed that TNNs on cell complexes exhibit superior expressivity and classification performance compared to traditional graph-based models. More recently, [32] introduced Cell Attention Networks (CAN), extending the attention mechanisms originally developed for simplicial complexes [33] to the more general cell complex setting, enabling flexible and adaptive weighting of multi-cell interactions.

While the works discussed above leverage higher-order interactions, they do so exclusively in the spatial domain. Some recent works have explored the use of topologies in the product space to model time-varying data defined on higher-order structures. We discuss these works later in Section 3.4. We now discuss the different modeling paradigms for multivariate time series data. Figure 3.2 provides a visual illustration of the three paradigms: structure-unaware, graph-aware and topology-aware frameworks.

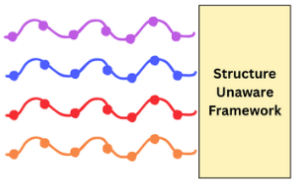
3.2. STRUCTURE-UNAWARE FRAMEWORKS

Structure-unaware approaches [13, 37, 84, 87] model each variable independently, capturing temporal dynamics without accounting for inter-variable relationships. Although these methods can effectively learn local patterns in individual sequences, they ignore the relational structure altogether. We explore both traditional statistical models and more recent deep learning models.

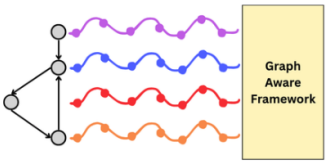
STATISTICAL MODELS

Traditional statistical models have been widely used for time series modeling and include methods like Autoregressive (AR) [84] and Moving Average (MA) [70], which form the foundation for the combined Autoregressive Moving Average (ARMA) [58]. These models are designed for univariate stationary time series, where AR models capture dependencies on past values, and MA models focus on past random errors. They are usually applied independently for each variable in multivariate data. However, these models assume stationarity in the data.

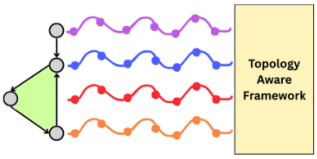
To address non-stationarity, Autoregressive Integrated Moving Average (ARIMA) [13] introduces a differencing step to handle trends and seasonal components. Despite their



(a) Structure-unaware frameworks that ignore relational dependencies altogether.



(b) Graph-aware frameworks that model pairwise relational dependencies using graphs.



(c) Topology-aware frameworks that model higher-order relational dependencies using topologies.

Figure 3.2: Multivariate time series modeling paradigms.

simplicity and interpretability, these models are limited in capturing complex, long-term temporal dependencies and non-linear relationships. These limitations have motivated the development of deep learning methods, which provide greater flexibility and scalability for modeling complex temporal patterns in time series data.

DEEP LEARNING MODELS

Neural networks, supported by the Universal Approximation Theorem [66], are powerful tools for modeling complex temporal data as they are capable of approximating any arbitrary continuous function. Popular architectures such as Recurrent Neural Networks (RNNs) [27], Convolutional Neural Networks (CNNs) [44], and Transformers [68] have been extensively applied to time series modeling, each offering unique strengths for capturing temporal dependencies.

RNNs were specifically designed to handle sequential data, such as time series and natural language, by leveraging recurrent connections which pass information across time steps. The hidden state acts as a memory unit, enabling the network to retain information from previous inputs and capture temporal dependencies. However, traditional RNNs struggle with modeling long-term dependencies due to vanishing and exploding gradients [52], which hinder their ability to propagate information over extended sequences. To address these limitations, gated architectures like Long Short-Term Memory (LSTM) networks [37] and Gated Recurrent Units (GRUs) [21] were developed. These architectures introduce mechanisms to selectively retain or forget information, effectively capturing both short- and long-term dependencies. GRUs, in particular, offer a computationally efficient alternative to LSTMs by using fewer parameters while maintaining comparable performance. Further advancements, such as Dilated RNNs [19], improve

the modeling of long-range dependencies by introducing dilated connections that expand the receptive field. Temporal attention mechanisms [28] further enhance this capability by dynamically focusing on relevant time steps, while sequence-to-sequence frameworks [72] provide flexible input-output mappings for tasks like multi-step forecasting, making these architectures versatile tools for time series prediction.

CNNs, originally designed for grid-like data structures like images, have been adapted to time series tasks due to their efficient parallel computations and smaller number of trainable parameters compared to RNNs. By employing causal convolutions [50], CNN-based models ensure that temporal predictions are based solely on past observations, making them suitable for autoregressive tasks. Temporal Convolutional Networks (TCNs) [3] extend this approach with dilated convolutions, exponentially expanding the receptive field as layers are stacked. This enables the model to capture dependencies across both short and long time scales effectively. Additionally, the WaveNet-CNN architecture [12], inspired by the WaveNet model [50], combines dilated convolutions with parameterized skip connections to efficiently model temporal and inter-variable relationships, making it a highly effective choice for time series modeling.

More recently, Transformer architectures [68] have revolutionized temporal data modeling by replacing recurrence with self-attention mechanisms. Unlike RNNs, Transformers process entire sequences simultaneously, enabling them to capture both local and global dependencies in a highly parallelized manner. This capability makes Transformers particularly effective for handling long-term temporal dependencies, which traditional architectures often struggle with. Several variants have been proposed to enhance performance. For instance, the Informer [87] introduces a sparse self-attention mechanism that reduces the quadratic complexity of traditional Transformers, making them scalable to long sequences. The Autoformer [73] incorporates a decomposition block that separates time series data into trend and seasonal components, improving interpretability and forecasting accuracy.

Despite the success of these models in capturing temporal dependencies, they do not explicitly account for the inter-variable spatial structure as an inductive bias which is often inherent in multivariate time series data. Incorporating the relational prior can prune spurious correlations, improve generalization, and enhance sample efficiency [23], thus motivating the use of graph-aware temporal modeling frameworks.

3.3. GRAPH-AWARE FRAMEWORKS

Graph-aware temporal models, known collectively as Spatio-Temporal Graph Neural Networks (STGNNs), are designed to capture spatial and temporal dependencies in multivariate data through distinctive spatial and temporal modules. The spatial module captures inter-variable relationships using graph convolutions, which may operate in the node domain [47, 61, 76], the spectral domain [16, 82], or through hybrid approaches that combines both [85]. The temporal module captures temporal relationships using RNNs [47, 51], CNNs [76, 82, 85], attention mechanisms [86], or hybrid models either in the time [47, 61, 82] or in the frequency domain [16]. The interaction between spatial and temporal modules determines whether a model is factorized [51, 76, 82] or coupled [47, 61, 86]. Factorized models treat spatial and temporal dependencies as separate processes, learning and processing them independently and then combining the outputs of

the two modules at a later fusion step. Coupled models, on the other hand, integrate spatial and temporal dependencies into a unified process, enabling the two modules to be learned jointly. This integrated approach often results in a more cohesive and expressive representation of spatio-temporal interactions, allowing the model to better capture complex patterns across time and space in multivariate data. Figure 3.3 illustrates the general composition of graph-aware frameworks, depicting the spatial and temporal modules and the interactions between them.

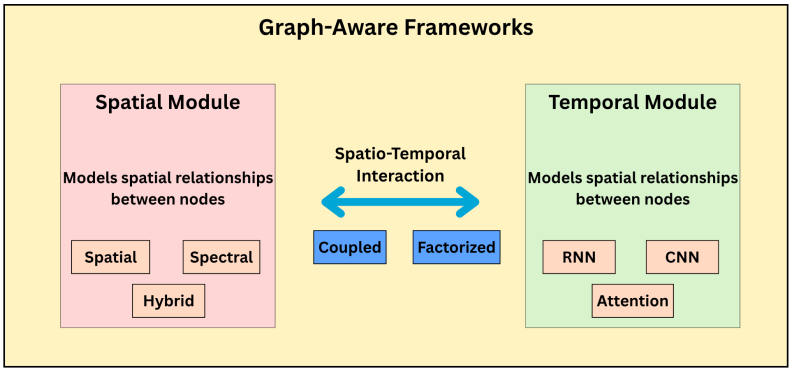


Figure 3.3: An illustration of graph-aware frameworks, composed of distinct spatial and temporal modules with their interaction giving rise to coupled and factorized models.

Building on this general structure, various previous works propose distinct designs for spatial and temporal modules and explore different strategies for their integration. STGCN [82] employs a modular design by stacking temporal convolutions followed by Gated Linear Units (GLUs) [24] and spatial ChebConv layers [25]. These ChebConv layers operate in the spectral domain to capture inter-variable relationships, and the alternating arrangement of spatial and temporal modules enables the model to separately learn temporal and spatial dependencies. Similarly, StemGNN [16] operates in the spectral domain for spatial modeling, but its temporal module takes a different approach by applying convolutions in the frequency domain. StemGNN first transforms the graph into the spectral domain and then projects the time series data into the frequency domain, enabling it to capture periodic patterns before applying temporal convolutions.

In contrast, DCRNN [47] and Graph WaveNet [76] operate in the node domain for spatial modeling. DCRNN incorporates graph diffusion convolutions into Gated Recurrent Units (GRUs) to jointly learn spatial and temporal dependencies in a coupled framework. This integration allows the spatial and temporal modules to influence each other directly during the learning process. Graph WaveNet, on the other hand, separates these dependencies into distinct modules, employing diffusion convolutions for spatial relationships and dilated causal convolutions for temporal dependencies. Graph Wavenet also learns an adaptive adjacency matrix as part of training to learn hidden relationships not captured by the given spatial graph.

GTCNNs [59], as discussed in Section 2.3.3, take a unique approach by constructing a spatio-temporal product graph and performing graph convolutions directly in the spatio-temporal space. Unlike other STGNNs that have distinct spatial and temporal

modules, GTCNNs inherently capture spatio-temporal dynamics through graph convolutions, resulting in well-defined spectral properties and provable stability to perturbations in the spatial graph. These advantages make product graph-based approaches particularly appealing for spatio-temporal modeling. However, all graph-aware models are inherently limited to capturing pairwise space-time relationships. We thus explore topologically-aware frameworks that model higher-order relationships in space and time.

3.4. TOPOLOGY-AWARE FRAMEWORKS

The concept of product spaces was introduced in [57], which presented a signal processing framework over product cell complexes, topological domains constructed through the Cartesian product of spatial and temporal simplicial complexes, as described in Section 2.3.4. This formulation enabled the modeling of multi-way spatio-temporal interactions within a higher-order topological space and, importantly, introduced the representation of time-varying signals on higher-order structures such as edges.

This line of work was extended in [53], which developed signal recovery methods for product cell complexes. By leveraging the factorized structure, the study demonstrated how edge and node signals on the product complex can be inferred from sub-sampled observations on the individual factor complexes.

These works represent an important step toward extending topological modeling to the spatio-temporal domain, as product spaces provide a principled framework for integrating higher-order structure with time-evolving data. However, existing approaches operate within the product cell complex and do not maintain a simplicial structure in the product space. Furthermore, they do not construct a TNN directly on the product complex, nor do they train it end-to-end for the downstream task. In contrast, our work ensures that the constructed product complex remains simplicial and construct a TNN on the resulting simplicial product complex. We argue that this leads to a more expressive and flexible framework for capturing higher-order relationships in the product space.

3.5. DISCUSSION

This chapter surveyed four broad areas of research relevant to our work. We began with studies that model purely spatial higher-order interactions. These lay the foundation for the topology-aware modeling paradigm. We then discussed the different paradigms for multivariate time series modeling starting with structure-unaware temporal models that capture dependencies in time series data without incorporating relational structure. While statistical structure-unaware methods offer interpretability and are effective for small-scale problems, they struggle with non-linear and long-range dependencies. Deep learning models address these limitations through more complex architectures that introduce non-linearities, but treat multivariate time series as flat inputs and do not leverage inter-variable relationships as an inductive bias.

To address this, graph-aware temporal models incorporate spatial priors, enabling improved generalization and more structured representations. However, they are fundamentally limited to pairwise interactions encoded by the graph structure, and thus cannot capture higher-order dependencies among entities.

To model time-varying signals on higher-order structures, recent works have proposed the use of product spaces, particularly the product cell complex, which enables the representation higher-order structures in the spatial and temporal domains. This construction offer a principled foundation for topologically-aware spatio-temporal modeling.

Building on these foundations, we now propose our framework that retains a simplicial structure in the product space. Our central hypothesis is that the simplicial structure in our framework offers a richer topological domain than the product cell complex by supporting a broader class of parameterized higher-order interactions. The next chapter formalizes this construction and introduces our proposed framework and neural network architecture.

4

THE SIMPLICIAL PRODUCT COMPLEX FRAMEWORK

This chapter presents a unified framework for spatio-temporal modeling of time-varying signals using the Simplicial Product Complex (SPC). We begin by introducing the SPC as a higher-order extension of the Strong Product Graph (SPG) in Section 4.1, demonstrating how cliques in the SPG naturally lift to 2-simplices. We then formally define the SPC, its algebraic structure, and associated signal spaces in Section 4.2. To enable data-adaptive learning of higher-order relationships, we develop a parameterized variant of the SPC by first cataloging the different simplex types that arise in Section 4.3, then constructing the parameterized SPC through separate incidence matrices for each simplex type in Section 4.4. We subsequently derive the corresponding parameterized Hodge Laplacians and define the convolutional filters used by our proposed Simplicial Product Complex Convolutional Neural Network (SPCCNN) architecture in Sections 4.5-4.6. The SPCCNN leverages these parameterized Hodge Laplacians to learn the importance of different higher-order relationships in multivariate time series data in a data-adaptive manner. Finally, we address scalability challenges and discuss the advantages of the SPC over the product cell complex in Section 4.7.

4.1. FROM STRONG PRODUCT GRAPH TO SIMPLICIAL PRODUCT COMPLEX

Product graphs offer a principled framework for spatio-temporal modeling by enabling convolutions over space-time graphs. Recall that the SPG contains spatial, temporal, and Kronecker edges, and is represented as $\mathcal{G}_{\boxtimes} = \mathcal{G}_T \boxtimes \mathcal{G}_S = (\mathcal{V}_{\boxtimes}, \mathcal{E}_{\boxtimes}, \mathbf{S}_{\boxtimes})$, where $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S, \mathbf{S}_S)$ and $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T, \mathbf{S}_T)$ are the spatial and temporal graphs, respectively, as defined in Section 2.3.3. Let $\mathbf{B}_{1S}$ and $\mathbf{B}_{1T}$ denote the node-to-edge incidence matrices of $\mathcal{G}_S$ and $\mathcal{G}_T$ respectively. Figure 4.1 illustrates an example SPG, highlighting spatial (red), temporal (green), and spatio-temporal (gray) edges.

While the SPG encodes rich pairwise connectivity, it is inherently limited to binary re-

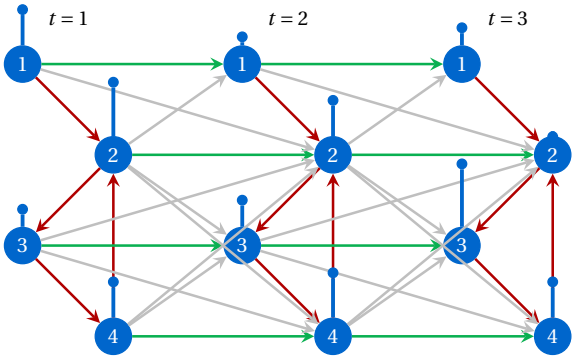


Figure 4.1: An example SPG with the different edge types highlighted and the signals defined on the nodes.

lationships, capturing only spatial, temporal, or spatio-temporal edges between nodes. As discussed earlier, many real-world systems involve coordinated inter-temporal interactions among multiple nodes that cannot be represented through pairwise links alone. Consider a traffic monitoring system where sensors at intersections record vehicle flow over time. Traffic dynamics often involve joint influences over time: for example, simultaneous congestion at intersections A and B may collectively affect future flow at intersection C, especially if C is only reachable via A and B. To explicitly capture such multi-node dependencies, we lift the SPG into the SPC, which introduces higher-order structures by forming triangles in the SPG. Intuitively, the SPC is a higher-order extension of the SPG: in addition to vertices representing space-time locations and edges encoding spatial and temporal connections, it incorporates filled triangles to form a spatio-temporal simplicial complex. As illustrated in Figure 4.2, this results in simplicial complex in space and time, enabling TNNs operating on the SPC to model spatio-temporal interactions that go beyond pairwise relationships. Moreover, the simplicial structure in the product space decomposes the homogeneous cells present in the product cell complex into distinguishable, heterogeneous higher-order relationships, as we will demonstrate later in this chapter.

4.2. SIMPLICIAL PRODUCT COMPLEX

In this section, we formally define the SPC by detailing the construction of each skeleton. Let $\mathcal{G}_S$, $\mathcal{G}_T$ and $\mathcal{G}_{\boxtimes}$ be the spatial, temporal and strong product graphs as defined above. We construct the SPC $\mathcal{X}$ as the clique complex over $\mathcal{G}_{\boxtimes}$. That is, the SPC $\mathcal{X}$ of dimension K contains all cliques in SPG $\mathcal{G}_{\boxtimes}$ of size at most $(K+1)$, closed under the face inclusion property. Below, we describe its structure by explicitly defining its k-skeletons for $k \geq 0$ in terms of the $\mathcal{V}_{\boxtimes}$ and $\mathcal{E}_{\boxtimes}$, the node and edge sets of SPG $\mathcal{G}_{\boxtimes}$.

4.2.1. SIMPLICIAL STRUCTURE

0-skeleton: The 0-skeleton of $\mathcal{X}$ consists of all nodes (1-node cliques) in the strong product graph $\mathcal{G}_{\boxtimes}$ defined as:

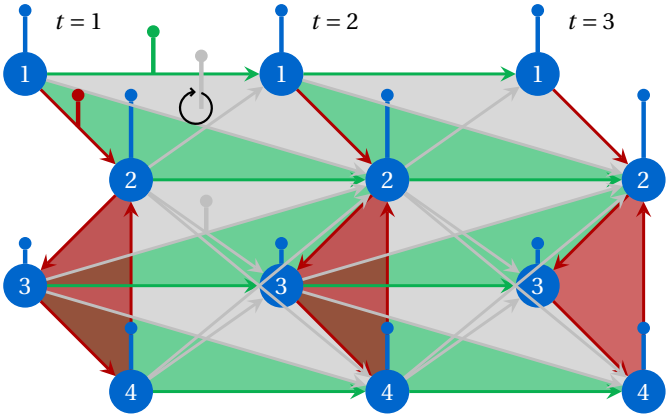


Figure 4.2: The lifted SPC for the given SPG. For visual clarity, we mark the orientation of only one triangle and show signals on a selected subset of edges and triangles.

$$\mathcal{Z}^0 = \left\{ v \;\middle|\; v \in \mathcal{V}_{\boxtimes} \right\}, \tag{4.1}$$

All nodes in the SPG form 0-simplices in the SPC. Each 0-simplex $v \in \mathcal{Z}^0$ can be written as a tuple (v_T, v_S), where $v_T \in \mathcal{V}_T$ is a temporal node and $v_S \in \mathcal{V}_S$ is a spatial node. That is, a 0-simplex (v_T, v_S) denotes the space-time location of node v_S at time-step v_T. The total number of 0-simplices, represented by $N^{0\,1}$, is given by:

$$N^0 = |\mathcal{V}_{\boxtimes}| = NT. \tag{4.2}$$

1-skeleton: The 1-skeleton of $\mathcal{Z}$ consists of all edges (2-node cliques) in the strong product graph $\mathcal{G}_{\boxtimes}$ defined as:

$$\mathcal{Z}^1 = \left\{ \{v, v'\} \subset \mathcal{Z}^0 \;\middle|\; (v, v') \in \mathcal{E}_{\boxtimes} \right\}. \tag{4.3}$$

All edges in the SPG form 1-simplices in the SPC. The total number of 1-simplices, represented by $N^{1\,1}$, is the same as the total number of edges in the SPG and is given by:

$$N^1 = |\mathcal{E}_{\boxtimes}| = T \cdot |\mathcal{E}_S| + N \cdot (T-1) + 2|\mathcal{E}_S| \cdot (T-1), \tag{4.4}$$

2-skeleton: The 2-simplices of $\mathcal{Z}$ are all triangles formed by 3-node cliques in $\mathcal{G}_{\boxtimes}$. That is, for any distinct nodes $x_i, x_j, x_k \in Z^0$, the triplet $\{x_i, x_j, x_k\}$ forms a 2-simplex if edges exist among each pair of 0-simplices:

$$\mathcal{Z}^2 = \left\{ \{x_i, x_j, x_k\} \subset \mathcal{Z}^0 \;\middle|\; \{x_i, x_j\}, \{x_j, x_k\}, \{x_i, x_k\} \in \mathcal{Z}^1 \right\}. \tag{4.5}$$

Each such triangle represents a higher-order spatio-temporal interaction between the three involved nodes. The total number of 2-simplices, represented by $N^{2\,1}$, is derived later in Equation 4.24.

[1] The superscript on N denotes the order of the simplex and not the exponentiation of N.

***k*-skeleton:** More generally, the k-skeleton of the SPC consists of all $(k+1)$-cliques in $\mathcal{G}_{\boxtimes}$, each lifted as a k-simplex. A set of 0-simplices $\sigma = \{x_0, x_1, \ldots, x_k\} \subset \mathcal{Z}^0$ forms a k-simplex if every pair of 0-simpices in the set forms a 1-simplex in $\mathcal{Z}^1$, i.e., it forms a complete subgraph in $G_{\boxtimes}$. The k-skeleton is defined as:

$$\mathcal{Z}^k = \left\{\sigma \subset \mathcal{Z}^0 \;\middle|\; |\sigma| = k+1,\ \forall\, x_i, x_j \in \sigma,\ i \neq j \Rightarrow \{x_i, x_j\} \in \mathcal{Z}^1\right\}. \tag{4.6}$$

This construction ensures that every k-simplex corresponds to a $(k+1)$-clique in $\mathcal{G}_{\boxtimes}$, and the resulting complex satisfies the face-inclusion property.

Full complex: The full Simplicial Product Complex of rank K is the union of all k-skeletons up to dimension K:

$$\mathcal{Z} = \bigcup_{k=0}^{K} \mathcal{Z}^k. \tag{4.7}$$

This defines $\mathcal{Z}$ as a clique complex over $\mathcal{G}_{\boxtimes}$, where higher-order simplices are constructed by lifting cliques in the product graph and are guaranteed to include all of their lower-dimensional faces.

In this work, we focus on a rank-2 SPC, i.e., we restrict to spatio-temporal simplices of dimension at most 2. This means that we lift only up to 3-node cliques from $\mathcal{G}_{\boxtimes}$, resulting in a simplicial complex composed of vertices (0-simplices), edges (1-simplices), and triangles (2-simplices).

4.2.2. SIGNAL SPACE

Above, we described how the SPG is topologically lifted to an SPC. We now describe the signal space of the SPC from the signals defined on the SPG. We assume that time-varying signals are only available on the nodes, as in most real-world applications. Since the temporal edges, spatio-temporal edges and all triangles are abstract structures that don't exist in the original graph, signals for these structures have to be lifted. The signal space of the SPC is thus extended from the nodes of the SPG to higher-order 1- and 2-simplices. To facilitate this, we define lifting functions ϕ^1 and ϕ^2 that map the signals present on a set of 0-simplices to real-valued signal features defined on the 1-simplices and 2-simplices of the SPC, respectively. We define the signal space of each order as follows:

0-simplices: The 0-simplicial signal space of the SPC directly inherits the node-level signals from the SPG. Let $\mathbf{X} \in \mathbb{R}^{N \times T}$ denote the input signal defined on the nodes of the SPG, where each node is indexed by a tuple $(v_T, v_S) \in \mathcal{V}_{\boxtimes}$. The corresponding simplicial signal on 0-simplices is defined as:

$$f^0 : \mathcal{Z}^0 \to \mathbb{R}, \quad f^0(v_T, v_S) = \mathbf{X}[v_S, v_T]. \tag{4.8}$$

Thus, each 0-simplex $(v_T, v_S) \in \mathcal{Z}^0$ is associated with the same signal as the corresponding node in the SPG by accessing the signal at the corresponding space-time location.

1-simplices: The 1-simplicial signal space extends the node signals to the edges of the SPC. Given a 1-simplex $\sigma = \{v^i, v^j\} \in \mathcal{X}^1$ where $v^i = (v_S^i, v_T^i)$ and $v^j = (v_S^j, v_T^j)$, the signal on the edge is derived by applying a lifting function $\phi^{(1)} : \mathbb{R}^2 \to \mathbb{R}$ to the node-level signals at its endpoints:

$$f^1 : \mathcal{X}^1 \to \mathbb{R}, \quad f^1(\{v^i, v^j\}) = \phi^{(1)}\left(f^0(v^i), f^0(v^j)\right). \tag{4.9}$$

2-simplices: For triangles in the SPC, the signal is computed by aggregating the node-level signals at the triangle's three vertices. Given a 2-simplex $\tau = \{v^i, v^j, v^k\} \in \mathcal{X}^2$ where $v^i = (v_S^i, v_T^i)$, $v^j = (v_S^j, v_T^j)$ and $v^k = (v_S^k, v_T^k)$, we define:

$$f^2 : \mathcal{X}^2 \to \mathbb{R}, \quad f^2(\{v^i, v^j, v^k\}) = \phi^{(2)}\left(f^0(v^i), f^0(v^j), f^0(v^k)\right). \tag{4.10}$$

The lifting functions $\phi^{(1)}$ and $\phi^{(2)}$ follow the properties defined in 2.2.5. Note that even though we assume edge and triangle level time-varying signals are not available, they can be incorporated in the signal space simply by adding them as inputs to the lifting functions whenever they are available.

4.2.3. ALGEBRAIC REPRESENTATION

Like any simplicial complex, the algebraic structure of the SPC is also captured by its boundary operators, represented by incidence matrices. Since we restrict the SPC to rank-2, the SPC is fully described by its node-to-edge and edge-to-triangle space-time incidence matrices, denoted by:

$$\mathbf{B}_{1\diamond} \in \mathbb{R}^{N^0 \times N^1} \quad \text{and} \quad \mathbf{B}_{2\diamond} \in \mathbb{R}^{N^1 \times N^2}. \tag{4.11}$$

$\mathbf{B}_{1\diamond}$ encodes the incidence relationship between 0-simplices and 1-simplices, capturing the incidence between all the space-time nodes and the spatial, temporal and spatio-temporal edges in the SPC. Each column corresponds to a 1-simplex and contains $+1$ and -1 entries indicating the orientation of the 1-simplex with respect to its two endpoint 0-simplices. $\mathbf{B}_{2\diamond}$ encodes the incidence relationship between 1-simplices and 2-simplices, capturing the incidence between the edges and all the higher-order space-time triangles formed in the SPC. Each column corresponds to a 2-simplex and contains entries $+1$, -1, or 0, depending on the relative orientation of the participating 1-simplices with respect to each 2-simplex.

It is important to understand that the $\mathbf{B}_{1\diamond}$ and $\mathbf{B}_{2\diamond}$ matrices aggregate all node-to-edge and edge-to-triangle relationships respectively, without distinguishing between the different types of pairwise and three-way relationships that arise in the SPC. When constructing the Hodge Laplacian using Equation 2.24, this aggregation prevents us from assigning distinct learnable parameters to each simplex type. To enable parameterization, we must construct separate incidence matrices for each simplex type and associate learnable parameters with each type of higher-order relationship. To accomplish this, we first catalog all the different higher-order structures in the rank-2 SPC in the following section, then move on to building the parameterized incidence matrices that allow for selective modulation of each higher-order relationship through learnable weights.

4.3. DISTINCT SIMPLEX TYPES

To construct these separate incidence matrices, we characterize the distinct types of 1- and 2-simplices that arise in the SPC, describe how they are induced from the structure of the SPG, and provide an intuition for the spatio-temporal relationships they encode.

4.3.1. 1-SIMPLICES

The 0- and 1-simplices of the SPC coincide with the node set $\mathcal{V}_{\boxtimes}$ and edge set $\mathcal{E}_{\boxtimes}$ of the SPG. The nodes in the SPG are treated the same way in SPC and we differentiate between the edges the same way as in SPG into three categoties: spatial 1-simplices, temporal 1-simplices and spatio-temporal 1-simplices, corresponding to the different edge types in SPG.

SPATIAL 1-SIMPLICES

A spatial 1-simplex e^{Spatial} is formed between two spatially adjacent nodes at the same time step. Formally:

$$e^{\text{Spatial}} = \{(t, v_i), (t, v_j)\} \quad \text{such that} \quad (v_i, v_j) \in \mathcal{E}_S,\ t \in \mathcal{V}_T.$$

The full set of such 1-simplices is denoted by:

$$\mathcal{Z}^1_{\text{Spatial}} = \left\{e_n^{\text{Spatial}}\right\}_{n=1}^{N^1_{\text{Spatial}}} \quad ; \quad N^1_{\text{Spatial}} = T \cdot |\mathcal{E}_S|, \tag{4.12}$$

where e_n^{Spatial} denotes the n^{th} spatial 1-simplex in a fixed index set.

TEMPORAL 1-SIMPLICES

A temporal 1-simplex e^{Temporal} connects the same spatial node across two consecutive time steps. Formally:

$$e^{\text{Temporal}} = \{(t, v_i), (t+1, v_i)\} \quad \text{such that} \quad (t, t+1) \in \mathcal{E}_T,\ v_i \in \mathcal{V}_S.$$

The full set of temporal 2-simplices is given by:

$$\mathcal{Z}^1_{\text{Temporal}} = \left\{e_n^{\text{Temporal}}\right\}_{n=1}^{N^1_{\text{Temporal}}} \quad ; \quad N^1_{\text{Temporal}} = N \cdot (T-1). \tag{4.13}$$

where e_n^{Temporal} denotes the n^{th} temporal 1-simplex in a fixed index set.

SPATIO-TEMPORAL 1-SIMPLICES

A spatio-temporal 1-simplex e^{ST} connects a node to a spatial neighbor in the next time step via a spatio-temporal edge induced by the spatial graph. Formally:

$$e^{\text{ST}} = \{(t, v_i), (t+1, v_j)\} \quad \text{such that} \quad (v_i, v_j) \in \mathcal{E}_S,\ (t, t+1) \in \mathcal{E}_T.$$

We denote the full set of such edges as:

$$\mathcal{Z}^1_{\text{ST}} = \left\{e_n^{\text{ST}}\right\}_{n=1}^{N^1_{\text{ST}}} \quad ; \quad N^1_{\text{ST}} = 2 \cdot |\mathcal{E}_S| \cdot (T-1). \tag{4.14}$$

Type	Notation	Cardinality
Spatial	$\mathcal{Z}^1_{\text{Spatial}}$	$N^1_{\text{Spatial}} = T \cdot \lvert\mathcal{E}_S\rvert$
Temporal	$\mathcal{Z}^1_{\text{Temporal}}$	$N^1_{\text{Temporal}} = N \cdot (T-1)$
Spatio-temporal	$\mathcal{Z}^1_{\text{ST}}$	$N^1_{\text{ST}} = 2 \cdot \lvert\mathcal{E}_S\rvert \cdot (T-1)$

Table 4.1: The cardinalities of 1-Simplices in the SPC.

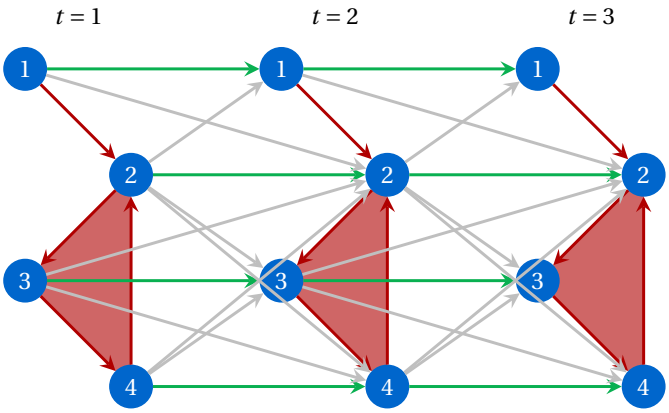


Figure 4.3: The SPC with the Spatial 2-simplices highlighted.

where e^{ST}_n denotes the n^{th} spatio-temporal 1-simplex in a fixed index set. It is trivial to note that:

$$N^1 = N^1_{\text{Spatial}} + N^1_{\text{Temporal}} + N^1_{\text{ST}} = |\mathcal{E}_{\boxtimes}| = T \cdot |\mathcal{E}_S| + N \cdot (T-1)| + 2|\mathcal{E}_S| \cdot (T-1) \tag{4.15}$$

The total number for 1-simplices of each type are summarized in Table 4.1. These three types of 1-simplices partition the set of all 1-simplices in the SPC (and the set of all edges in the SPG). They are visualized as red (spatial), green (temporal), and gray (spatio-temporal) 1-simplices of the SPC in Figure 4.2.

4.3.2. 2-SIMPLICES

We now turn our attention to the 2-simplices in the SPC, which correspond to filled triangles formed from 3-node cliques in the SPG. Each 2-simplex encodes a distinct higher-order interaction involving combinations of the different types of 1-simplices. In the remainder of this section, we catalog all distinct 2-simplex types and describe the specific relational structure they capture.

SPATIAL 2-SIMPLICES

A spatial 2-simplex $\triangle^{\text{Spatial}}$ is a triangle composed entirely of spatial 1-simplices within a single time step. It captures purely spatial, time-invariant higher-order interactions in the SPC. Figure 4.3 illustrates an example of spatial 2-simplices in the SPC from before.

Formally, for any time step $t \in \mathcal{V}_T$, if three spatial nodes $v_i, v_j, v_k \in \mathcal{V}_S$ form a 3-clique in the spatial graph $\mathcal{G}_S$, then their corresponding time-stamped copies $(t, v_i), (t, v_j), (t, v_k)$ define the vertices of a spatial 2-simplex in the SPC.

Let $\mathcal{X}^2_{\text{Spatial}}$ denote the set of all such simplices, with cardinality N^2_{Spatial}, indexed as:

$$\mathcal{X}^2_{\text{Spatial}} = \left\{ \triangle_n^{\text{Spatial}} \right\}_{n=1}^{N^2_{\text{Spatial}}} \tag{4.16}$$

where $\triangle_n^{\text{Spatial}}$ denotes the n-th spatial 2-simplex in a fixed ordering.

If $\mathcal{T}_S$ denotes the set of all 3-cliques in $\mathcal{G}_S$, then each such clique induces one spatial 2-simplex per time step, yielding a total count of:

$$N^2_{\text{Spatial}} = T \cdot |\mathcal{T}_S| \tag{4.17}$$

SPATIO-TEMPORAL 2-SIMPLICES

Unlike purely spatial simplices defined above, a spatio-temporal 2-simplex captures interactions between nodes that span multiple time steps and involve both spatial and temporal dependencies. These triangles exist in the spaces between consecutive time steps, analogous to the cells formed in the product cell complex. They have two nodes in one time step and one node in the neighboring time step (past or future). In order to intuitively understand the relationship they capture, we assume that the multi-node triangles aggregate information at the time step where the single node is present. We now describe the distinct configurations of spatio-temporal 2-simplices. Each configuration corresponds to a specific multi-node space-time interaction pattern in the SPC.

Type 1. Figure 4.4 illustrates examples of Type 1 2-simplices in the SPC. As shown in the figure, Type 1 triangles $\triangle^{\text{Type 1}}$ formed between two consecutive time steps capture how the spatial influence between two neighboring nodes affects the future state of one of them. Intuitively, these triangles represent how spatial interactions at time t propagate forward through time in a causal fashion. Structurally, each Type 1 triangle consists of one spatial edge, one temporal edge, and one spatio-temporal edge, where the spatio-temporal edge connects the spatial edge to the isolated node in the next time step.

We denote the set of all Type 1 triangles by $\mathcal{X}^2_{\text{Type 1}}$, with cardinality $N^2_{\text{Type 1}}$, and index it as:

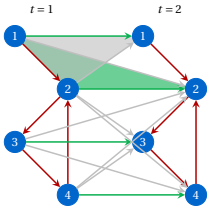
$$\mathcal{X}^2_{\text{Type 1}} = \left\{ \triangle_n^{\text{Type 1}} \right\}_{n=1}^{N^2_{\text{Type 1}}} \tag{4.18}$$

where $\triangle_n^{\text{Type 1}}$ refers to the n-th Type 1 2-simplex in a fixed ordering.

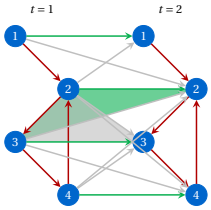
To compute the total number of Type 1 simplices, observe in Figure 4.4 that each spatial edge induces two Type 1 triangles between consecutive time steps $(t, t+1)$: one where the temporal edge originates from one endpoint of the spatial edge (green triangle), and another where it originates from the other endpoint (gray triangle). Since this occurs for every spatial edge and across each of the $T-1$ spaces between consecutive time steps, the total number of Type 1 triangles is given by:

$$N^2_{\text{Type 1}} = 2 \cdot |\mathcal{E}_S| \cdot (T-1) \tag{4.19}$$

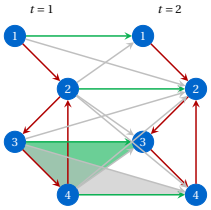
signifying that two Type 1 triangles are created for each spatial edge across all the $T-1$ consecutive time step pairs.



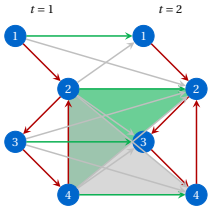
(a) The two Type 1 2-simplices created by spatial edge **1** → **2**



(b) The two Type 1 2-simplices created by spatial edge **2** → **3**



(c) The two Type 1 2-simplices created by spatial edge **3** → **4**



(d) The two Type 1 2-simplices created by spatial edge **4** → **2**

Figure 4.4: Type 1 2-simplices created in the SPC formed using one spatial edge, one temporal edge and one spatio-temporal edge with the spatial edge in the **previous** time step.

Type 2. A Type 2 2-simplex $\triangle^{\text{Type 2}}$ is a spatio-temporal triangle formed between two consecutive time steps and is structurally similar to a Type 1 triangle but with the isolated node shifted to the previous time step. Intuitively, it is a non-causal version of the spatio-temporal interaction in Type 1, where spatial structure at time $t+1$ influences one of the nodes from the past. Figure 4.5 illustrates examples of a Type 2 triangles in the SPC above.

We denote the set of all Type 2 triangles by $\mathcal{X}^2_{\text{Type 2}}$, with cardinality $N^2_{\text{Type 2}}$, and index its elements as:

$$\mathcal{X}^2_{\text{Type 2}} = \left\{ \triangle_n^{\text{Type 2}} \right\}_{n=1}^{N^2_{\text{Type 2}}} \tag{4.20}$$

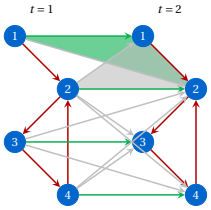
where $\triangle_n^{\text{Type 2}}$ denotes the n-th Type 2 2-simplex in a fixed ordering.

As shown in Figure 4.5, each spatial edge induces two Type 2 triangles between every consecutive time step pair, just like in the Type 1 case. Therefore, the total number of Type 2 triangles is also given by:

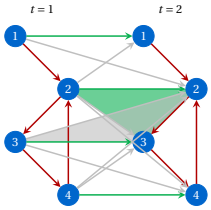
$$N^2_{\text{Type 2}} = 2 \cdot |\mathcal{E}_S| \cdot (T-1) \tag{4.21}$$

It can be seen visually that the Type 1 and Type 2 triangles together capture the relationship encoded by the spatio-temporal cell in the product cell complex. We discuss this in detail in Section 4.7.2 where we compare the SPC to the product cell complex.

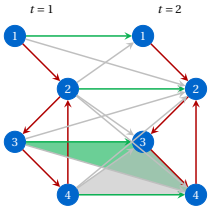
Type 3. Just as Type 1 and Type 2 2-simplices represent causal and non-causal versions of the same relationship pattern, Type 3 and Type 4 2-simplices are also structurally simi-



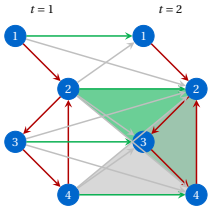
(a) The two Type 2 2-simplices created by spatial edge **1** → **2**



(b) The two Type 2 2-simplices created by spatial edge **2** → **3**



(c) The two Type 2 2-simplices created by spatial edge **3** → **4**



(d) The two Type 2 2-simplices created by spatial edge **4** → **2**

Figure 4.5: Type 2 2-simplices created in the SPC formed using one spatial edge, one temporal edge and one spatio-temporal edge with the spatial edge in the **next** time step.

lar, forming causal and non-causal variants of another distinct relationship pattern. Figure 4.6 illustrates examples of Type 3 triangles in the SPC. A Type 3 triangle $\triangle^{\text{Type 3}}$ forms between two consecutive time steps and captures how two spatial neighbors jointly influence a common third node in the future through causal propagation. Unlike Type 1 and Type 2 triangles, which capture the influence on one of the two spatially connected nodes, these triangles capture the joint effect of spatial neighbors on a third node. Structurally, each Type 3 triangle consists of one spatial edge and two spatio-temporal edges, where both spatio-temporal edges originate from the endpoints of the spatial edge and connect to the shared neighbor at the next time step.

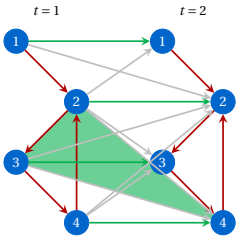
We denote the set of all such 2-simplices by $\mathcal{Z}^2_{\text{Type 3}}$, with cardinality $N^2_{\text{Type 3}}$, and index them as:

$$\mathcal{Z}^2_{\text{Type 3}} = \left\{ \triangle_n^{\text{Type 3}} \right\}_{n=1}^{N^2_{\text{Type 3}}} \tag{4.22}$$

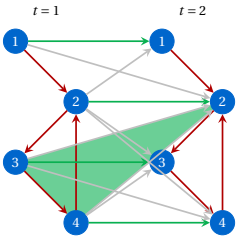
where $\triangle_n^{\text{Type 3}}$ is the n-th Type 3 triangle in a fixed index ordering.

To compute the total number of such triangles, note that Type 3 simplices are only formed from spatial 3-cliques, as all three nodes must be pairwise connected for the necessary spatial edge and the spatio-temporal edges to exist. Within each spatial 3-clique, each of the three edges gives rise to one Type 3 triangle per time step, as illustrated in Figure 4.6. Since this structure repeats across $T-1$ temporal intervals, the total number of Type 3 triangles is given by:

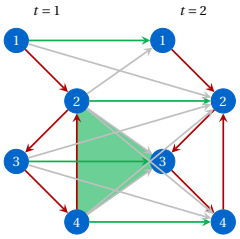
$$N^2_{\text{Type 3}} = 3 \cdot |\mathcal{T}_S| \cdot (T-1) \tag{4.23}$$



(a) The Type 3 2-simplex created by edge **2** → **3** of spatial triangle {**2**,**3**,**4**}



(b) The Type 3 2-simplex created by edge **3** → **4** of spatial triangle {**2**,**3**,**4**}



(c) The Type 3 2-simplex created by edge **4** → **2** of spatial triangle {**2**,**3**,**4**}

Figure 4.6: Type 3 2-simplices created in the SPC formed using one spatial edge and two spatio-temporal edges with the spatial edge in the **previous** time step.

signifying that a Type 3 triangle is created for each of the three edges for every spatial 3-node clique across the $T-1$ consecutive time step pairs.

Type 4. A Type 4 2-simplex $\triangle^{\text{Type 4}}$ is a spatio-temporal triangle formed between two consecutive time steps that is structurally analogous to a Type 3 triangle, but with the isolated node shifted to the previous time step. Type 4 triangles capture a non-causal interaction where two spatial neighbors jointly influence a common third node but in the past instead of the future. Due to the shift in the spatial edge to the next time step, the two spatio-temporal edges now terminate at the endpoints of the spatial edge, rather than originating from them as they did in Type 3 triangles. Figure 4.7 illustrates examples of such a triangle in the example SPC and demonstrates the structure described above.

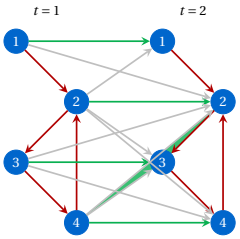
The set of all such simplices is denoted $\mathcal{Z}^2_{\text{Type 4}}$, with cardinality $N^2_{\text{Type 4}}$, and is indexed as:

$$\mathcal{Z}^2_{\text{Type 4}} = \left\{ \triangle_n^{\text{Type 4}} \right\}_{n=1}^{N^2_{\text{Type 4}}}$$

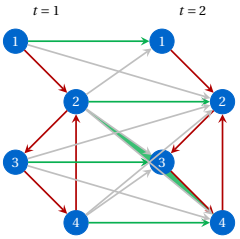
where $\triangle_n^{\text{Type 4}}$ denotes the n-th Type 4 spatio-temporal 2-simplex.

As with Type 3 triangles, each edge in a spatial 3-clique gives rise to one Type 4 triangle per time step as shown in 4.7. Since this structure repeats over $T-1$ temporal intervals, the total number of Type 4 triangles is given by:

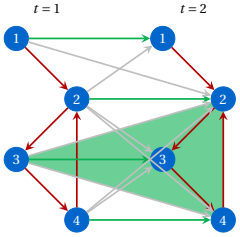
$$N^2_{\text{Type 4}} = 3 \cdot |\mathcal{T}_S| \cdot (T-1)$$



(a) The Type 4 2-simplex created by edge **2** → **3** of spatial triangle {**2**,**3**,**4**}



(b) The Type 4 2-simplex created by edge **3** → **4** of spatial triangle {**2**,**3**,**4**}



(c) The Type 4 2-simplex created by edge **4** → **2** of spatial triangle {**2**,**3**,**4**}

Figure 4.7: Type 4 2-simplices created in the SPC formed using one spatial edge and two spatio-temporal edges with the spatial edge in the **next** time step.

Together, these five types of 2-simplices form a complete, mutually exclusive set of filled triangles in the SPC. The cardinalities of each type of 2-simplex are summarised in Table 4.2 The total number of 2 simplices is therefore given as:

$$N^2 = \underbrace{T \cdot |\mathcal{T}_S|}_{\text{Spatial}} + \underbrace{2 \cdot |\mathcal{E}_S| \cdot (T-1)}_{\text{Type 1}} + \underbrace{2 \cdot |\mathcal{E}_S| \cdot (T-1)}_{\text{Type 2}} + \underbrace{3 \cdot |\mathcal{T}_S| \cdot (T-1)}_{\text{Type 3}} + \underbrace{3 \cdot |\mathcal{T}_S| \cdot (T-1)}_{\text{Type 4}} \quad (4.24)$$

4.4. PARAMETERIZED SIMPLICIAL PRODUCT COMPLEX

Having identified the distinct simplex types, we now construct a representation of the SPC that can parameterize these relationships. We follow an approach similar to the parameterized product graph described in Section 2.3.3: we construct separate incidence matrices for each higher-order relationship type and associate learnable parameters with each to form the parameterized incidence matrix. This approach isolates each of the higher-order relationships discussed above, enabling us to create a parameterized Hodge Laplacian matrix in Section 4.5. We argue that parameterization provides the following benefits:

- **Flexibility:** Rather than assuming uniform presence of all higher-order relationships across datasets, parameterization allows the model to adapt to dataset-specific

Type	Notation	Cardinality
Spatial	$\mathcal{Z}^2_{\text{Spatial}}$	$N^2_{\text{Spatial}} = T \cdot \lvert\mathcal{T}_S\rvert$
Type 1	$\mathcal{Z}^2_{\text{Type 1}}$	$N^2_{\text{Type 1}} = 2 \cdot \lvert\mathcal{E}_S\rvert \cdot (T-1)$
Type 2	$\mathcal{Z}^2_{\text{Type 2}}$	$N^2_{\text{Type 2}} = 2 \cdot \lvert\mathcal{E}_S\rvert \cdot (T-1)$
Type 3	$\mathcal{Z}^2_{\text{Type 3}}$	$N^2_{\text{Type 3}} = 3 \cdot \lvert\mathcal{T}_S\rvert \cdot (T-1)$
Type 4	$\mathcal{Z}^2_{\text{Type 4}}$	$N^2_{\text{Type 4}} = 3 \cdot \lvert\mathcal{T}_S\rvert \cdot (T-1)$

Table 4.2: The cardinalities of 2-Simplices in the SPC.

structures by learning the relevance of each type of higher-order interaction from the data.

- **Improving downstream performance:** Learning the spatio-temporal coupling would allow finer control over the contribution of different higher-order relationships in the SPC and is hypothesized to improve task-specific performance.
- **Scalability:** Incorporating regularization to induce sparsity in the learned parameters can allow the model to disregard unimportant relationship types entirely, reducing computational overhead and improving scalability.

The rest of this section now provides a systematic procedure to construct the incidence matrices for each simplex type: $\mathbf{B}^{\text{Spatial}}_{1\diamond}$, $\mathbf{B}^{\text{Temporal}}_{1\diamond}$, and $\mathbf{B}^{\text{ST}}_{1\diamond}$ for 1-simplices, and $\mathbf{B}^{\text{Spatial}}_{2\diamond}$, $\mathbf{B}^{\text{Type1}}_{2\diamond}$, $\mathbf{B}^{\text{Type2}}_{2\diamond}$, $\mathbf{B}^{\text{Type3}}_{2\diamond}$, and $\mathbf{B}^{\text{Type4}}_{2\diamond}$ for 2-simplices. We then define $\tilde{\mathbf{B}}_{1\diamond}$ and $\tilde{\mathbf{B}}_{2\diamond}$ as the parameterized combinations of these matrices, forming the full parameterized representation of the SPC. This formulation enables the filter introduced in Section 4.5 to learn the relative influence of each simplex type during training.

4.4.1. PARAMETERIZING 1-SIMPLICES

Let $\mathbf{I}_S \in \mathbb{R}^{N\times N}$ and $\mathbf{I}_T \in \mathbb{R}^{T\times T}$ be the identity matrices over the spatial and temporal domains, respectively. We define the incidence matrices of the different 1-simplex types introduced in Section 4.3, following the construction principles outlined in [60], as follows:

CONSTRUCTING $\mathbf{B}^{\text{SPATIAL}}_{1\diamond}$

The matrix $\mathbf{B}^{\text{Spatial}}_{1\diamond} \in \mathbb{R}^{N_0\times N^{\text{Spatial}}_1}$ encodes the incidence between 0-simplices and spatial 1-simplices in the SPC. To construct it, we repeat the spatial incidence matrix $\mathbf{B}_{1S}$ across all time steps using the Kronecker product:

$$\mathbf{B}^{\text{Spatial}}_{1\diamond} = \mathbf{I}_T \otimes B_{1S} \tag{4.25}$$

The block matrix constructed by this Kronecker product is shown in Figure 4.8 where each B_{1S} block has shape $\mathbb{R}^{|\mathcal{V}_S|\times|\mathcal{E}_S|}$. By construction, the rows of $\mathbf{B}^{\text{Spatial}}_{1\diamond}$ (corresponding to 0-simplices) are grouped by time: the first $|\mathcal{V}_S|$ rows correspond to time $t = 1$, the next

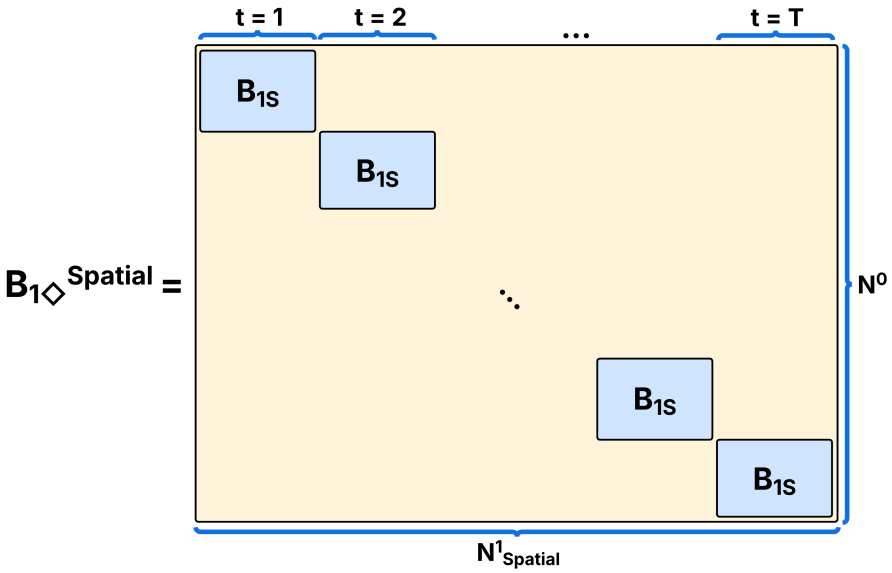


Figure 4.8: The block structure of $\mathbf{B}_{1\diamond}^{\text{Spatial}}$ constructed as a Kronecker product of I_T and B_{1S}.

$|\mathcal{V}_S|$ to $t = 2$, and so forth. Similarly, the columns of $\mathbf{B}_{2\diamond}^{\text{Spatial}}$ (corresponding to spatial 1-simplices) are also grouped by time: the first $|\mathcal{E}_S|$ columns correspond to time $t = 1$, the next $|\mathcal{E}_S|$ to $t = 2$, and so on. Each diagonal block B_{1S} therefore captures the spatial graph at a specific time slice. Within each time slice, the ordering of simplices follows the ordering of nodes and edges in the spatial incidence matrix B_{1S}.

CONSTRUCTING $\mathbf{B}_{1\diamond}^{\text{TEMPORAL}}$

The matrix $\mathbf{B}_{1\diamond}^{\text{Temporal}} \in \mathbb{R}^{N_0 \times N_1^{\text{Temporal}}}$ encodes the incidence between 0-simplices and temporal 1-simplices in the SPC. It is constructed by repeating the temporal incidence structure for each spatial node using the Kronecker product:

$$\mathbf{B}_{1\diamond}^{\text{Temporal}} = \mathbf{B}_{1T} \otimes \mathbf{I}_S \tag{4.26}$$

The block structure is shown in Figure 4.9. It can be observed that each pair of I_S and $-I_S$ blocks in a column defines directed edges from each node to itself in the next time step, representing the temporal edges for a specific time slice. The ordering of the rows of $\mathbf{B}_1^{\text{Temporal}}$ remains the same as $\mathbf{B}_1^{\text{Spatial}}$ and the columns are similarly grouped by time: the first N temporal 1-simplices correspond to time $t = 1$, the next N to time $t = 2$, and so on, up to $t = T - 1$, Within each time step, the columns are ordered according to the spatial node index in $\mathcal{V}_S$.

CONSTRUCTING $\mathbf{B}_{1\diamond}^{\text{ST}}$

The matrix $\mathbf{B}_{1\diamond}^{\text{ST}} \in \mathbb{R}^{N_0 \times N_1^{\text{ST}}}$ encodes the incidence between 0-simplices and spatio-temporal 1-simplices in the SPC. We know that each directed spatial edge $(v_i, v_j) \in$

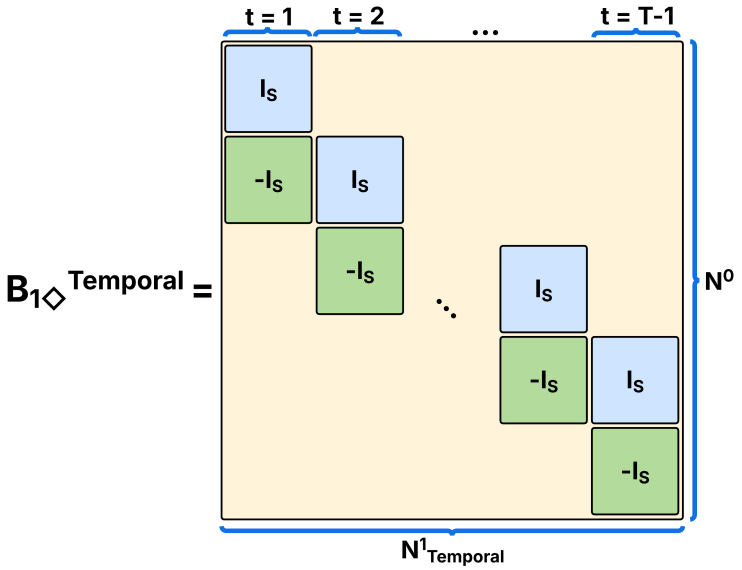


Figure 4.9: The block structure of $\mathbf{B}_{1\diamond}^{\text{Temporal}}$ constructed as a Kronecker product of B_{1T} and I_S.

$\mathscr{E}_S$ induces two spatio-temporal edges between consecutive time steps: $((t,v_i),(t+1,v_j))$ and $((t,v_j),(t+1,v_i)) \in \mathcal{I}_{\text{ST}}^1$. These are categorized as:

- *Concordant:* Spatio-temporal edges where the direction is preserved across time, i.e., from (t,v_i) to $(t+1,v_j)$, aligned with the original spatial edge direction from v_i to v_j. Concordant spatio-temporal edges essentially go from the outgoing endpoint in t to the incoming endpoint in $t+1$, preserving the original direction of the edge across space-time.
- *Discordant:* Spatio-temporal edges where the direction is reversed across time, i.e., from (t,v_j) to $(t+1,v_i)$, opposite to the spatial edge direction. Discordant spatio-temporal edges go from the incoming endpoint in t to the outgoing endpoint in $t+1$, reversing the original direction of the edge across space-time.

To construct this incidence structure, we therefore isolate the incoming and outgoing nodes of each spatial edge as follows:

$$\mathbf{B}_{1S}^{\text{Inc}} = \text{ReLU}(\mathbf{B}_{1S}), \quad \mathbf{B}_{1S}^{\text{Out}} = \text{ReLU}(-\mathbf{B}_{1S}), \tag{4.27}$$

The ReLu operator removes the negatively signed outgoing nodes from each column in $\mathbf{B}_{1S}$ to create $\mathbf{B}_{1S}^{\text{Inc}}$. Conversely, negating $\mathbf{B}_{1S}$ and applying the ReLu operator retains only the outgoing endpoints by suppressing the originally positive incoming entries.

We construct $\mathbf{B}_{1\diamond}^{\text{ST}}$ by stacking these block matrices, creating edges from outgoing nodes in time t to incoming nodes in time $t+1$ for concordant spatio-temporal edges

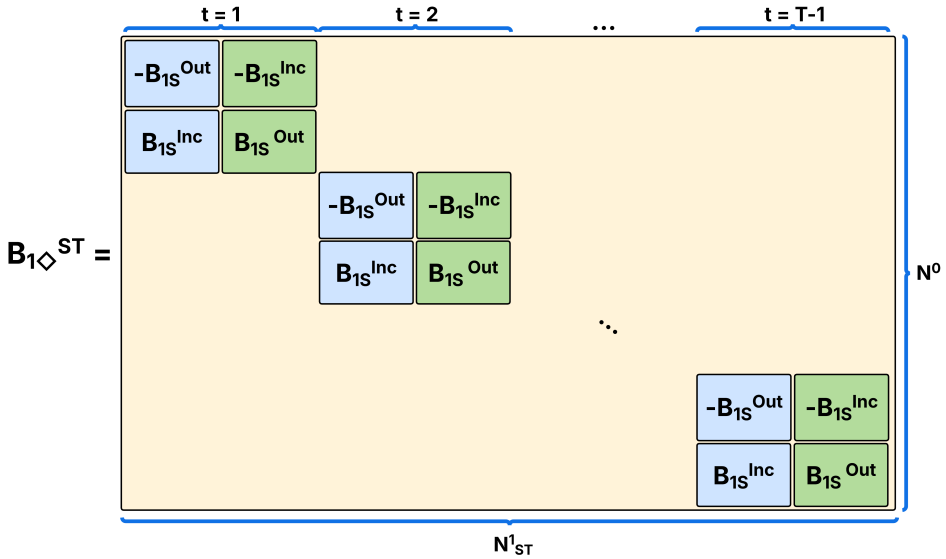


Figure 4.10: The block structure of $\mathbf{B}_{1\diamond}^{\mathrm{ST}}$ constructed by stacking $\mathbf{B}_{1S}^{\mathrm{Inc}}$ and $\mathbf{B}_{1S}^{\mathrm{Out}}$.

(blue blocks) and vice versa for discordant spatio-temporal edges (green blocks), as illustrated in Figure 4.10.

Similar to the matrices we constructed above, the columns of $\mathbf{B}_{1\diamond}^{\mathrm{ST}}$ are ordered by time: the first $2|\mathscr{E}_S|$ columns correspond to spatio-temporal edges from $t = 1$ ($|\mathscr{E}_S|$ concordant and $|\mathscr{E}_S|$ discordant), the next to $t = 2$, and so on, up to $t = T - 1$. The row ordering is consistent with the time-indexed row ordering used in $\mathbf{B}_{1\diamond}^{\mathrm{Spatial}}$ and $\mathbf{B}_{1\diamond}^{\mathrm{Temporal}}$.

CONSTRUCTING $\mathscr{B}_{1\diamond}$

Having constructed the incidence matrices for each type of 1-simplex, we now define the full parameterized node-to-edge incidence matrix of the SPC as:

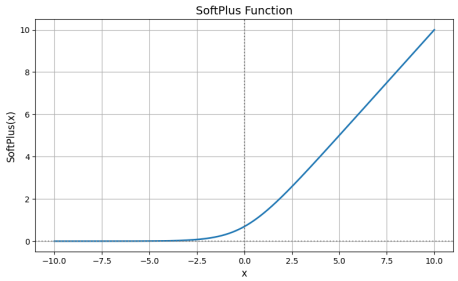
$$\tilde{\mathbf{B}}_{1\diamond} = \left[\begin{array}{c|c|c} \sqrt{e_1} \cdot \mathbf{B}_{1\diamond}^{\mathrm{Spatial}} & \sqrt{e_2} \cdot \mathbf{B}_{1\diamond}^{\mathrm{Temporal}} & \sqrt{e_3} \cdot \mathbf{B}_{1\diamond}^{\mathrm{ST}} \end{array} \right] \tag{4.28}$$

Here, $|$ represents a column-wise concatenation and e_1, e_2, and e_3 are non-negative scalars obtained by applying the SoftPlus function to the corresponding learnable parameters to enforce non-negativity. The SoftPlus function is defined as:

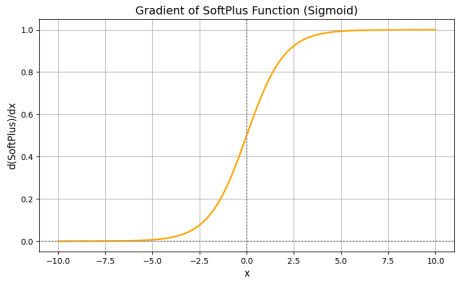
$$\mathrm{SoftPlus}(x) = \log(1 + e^x) \tag{4.29}$$

The gradient of the SoftPlus function is identical to the sigmoid function and is given by:

$$\frac{d}{dx}\mathrm{SoftPlus}(x) = \frac{1}{1 + e^{-x}} \tag{4.30}$$



(a) The SoftPlus function given by Equation 4.29



(b) The gradient of the SoftPlus function given by Equation 4.30

Figure 4.11b shows the graph and the gradient of the SoftPlus function. We can see that this function ensures that the learned weights remain strictly positive and is smooth and differentiable, making it suitable for gradient-based optimization. The square roots of the post-softmax value of e_1, e_2, and e_3 are then used to scale the respective incidence matrices, modulating the contribution of each 1-simplex type in the construction of the parameterized Hodge Laplacians (Section 4.5).

The order of concatenation in Equation (4.28) implies that the first $T \cdot |\mathcal{E}_S|$ columns correspond to spatial 1-simplices, the next $N \cdot (T-1)$ columns to temporal 1-simplices, and the final $2 \cdot |\mathcal{E}_S| \cdot (T-1)$ columns to spatio-temporal 1-simplices. The resulting matrix $\tilde{\mathbf{B}}_{1\diamond} \in \mathbb{R}^{N_0 \times N_1}$ thus encodes the complete node-to-edge structure of the SPC and assigns a learnable non-negative importance weight to each type of 1-simplex through its parameterization.

4.4.2. PARAMETERIZING 2-SIMPLICES

We now turn our attention to constructing the individual edge-to-triangle incidence matrices for each 2-simplex type: $\mathbf{B}_{2\diamond}^{\text{Spatial}}$, $\mathbf{B}_{2\diamond}^{\text{Type 1}}$, $\mathbf{B}_{2\diamond}^{\text{Type 2}}$, $\mathbf{B}_{2\diamond}^{\text{Type 3}}$, and $\mathbf{B}_{2\diamond}^{\text{Type 4}}$. Let $\mathbf{B}_{2S} \in \mathbb{R}^{|\mathcal{E}_S| \times |\mathcal{T}_S|}$ denote the edge-to-triangle incidence matrix for the spatial graph G_S.

To ensure compatibility across constructions, we fix the row ordering of all $\mathbf{B}_{2\diamond}$ matrices to match the column ordering of the corresponding $\tilde{\mathbf{B}}_{1\diamond}$ matrices. This consistency in the ordering of 1-simplices is essential for the validity of the Hodge Laplacians defined in Section 4.5.

Initially, we construct each incidence matrix by encoding the presence of an edge within a triangle without considering orientation, by taking the absolute values of the incidence entries. We omit the absolute values in the figures and equations below for ease of notation. The orientation is then assigned separately in a subsequent step, as we discuss later in this section. The remainder of this section details the construction of the edge-to-triangle incidence matrices for each 2-simplex type.

CONSTRUCTING $\mathbf{B}_{2\diamond}^{\text{SPATIAL}}$

The matrix $\mathbf{B}_{2\diamond}^{\text{Spatial}} \in \mathbb{R}^{N_1 \times N_2^{\text{Spatial}}}$ encodes the incidence between 1-simplices and spatial 2-simplices in the SPC. Since spatial triangles are formed solely from spatial edges within the same time step, only the first $T \cdot |\mathcal{E}_S|$ rows, corresponding to the spatial 1-simplices

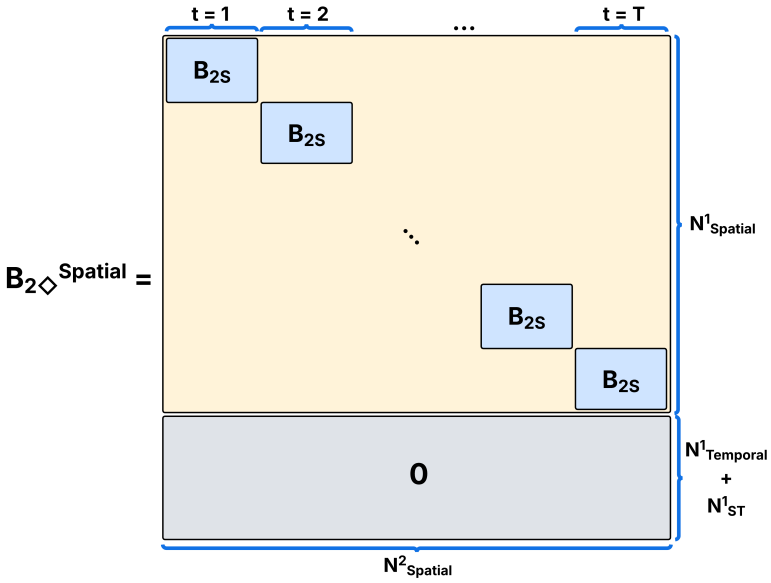


Figure 4.12: The block structure of the $\mathbf{B}_{2\diamond}^{\text{Spatial}}$ matrix.

participate in their construction. All other rows associated with temporal and spatio-temporal 1-simplices are filled with zeros.

As shown in Figure 4.12, we construct this matrix by repeating the spatial triangle incidence matrix $\mathbf{B}_{2S} \in \mathbb{R}^{|\mathcal{E}_S| \times |\mathcal{T}_S|}$ across all time steps and appending zeros below to match the total number of 1-simplices. Formally, we define it as:

$$\mathbf{B}_{2\diamond}^{\text{Spatial}} = \begin{bmatrix} \mathbf{I}_T \otimes \mathbf{B}_{2S} \\ \mathbf{0}_{(N_1^{\text{Temporal}} + N_1^{\text{ST}}) \times N_2^{\text{Spatial}}} \end{bmatrix} \tag{4.31}$$

Each diagonal block $\mathbf{B}_{2S}$ captures the spatial triangular relationships at a specific time slice between the edges and the triangles at that time. Within each time slice, the ordering of simplices follows the ordering of edges and cliques in the spatial edge-to-clique incidence matrix $\mathbf{B}_{2S}$.

GENERAL FRAMEWORK TO CONSTRUCT $\mathbf{B}_{2\diamond}^{\text{TYPE K}}$

We now introduce a general framework we follow to construct the parameterized incidence matrix $\mathbf{B}_{2\diamond}^{\text{Type } k}$ for each spatio-temporal triangle type $k \in \{1,2,3,4\}$. For each type, we first define the per-time-step incidence matrix $\mathbf{B}_{2\diamond,t}^{\text{Type } k} \in \mathbb{R}^{N^1 \times N_{\text{Type } k}^{2,t}}$, where $N_{\text{Type } k}^{2,t}$ denotes the number of Type k triangles at time step t. Each such matrix encodes the incidence between 1-simplices and 2-simplices of Type k formed at time t. We then construct the complete incidence matrix by concatenating the per-time-step incidence matrices column-wise.

Moreover, to facilitate the construction of the $\mathbf{B}_{2\diamond,t}^{\text{Type } k}$ matrix, we further decompose it into three blocks:

Type	Cardinality
Type 1	$N^{2,t}_{\text{Type 1}} = 2 \cdot \lvert\mathscr{E}_S\rvert$
Type 2	$N^{2,t}_{\text{Type 2}} = 2 \cdot \lvert\mathscr{E}_S\rvert$
Type 3	$N^{2,t}_{\text{Type 3}} = 3 \cdot \lvert\mathscr{T}_S\rvert$
Type 4	$N^{2,t}_{\text{Type 4}} = 3 \cdot \lvert\mathscr{T}_S\rvert$

Table 4.3: Per time step cardinalities of spatio-temporal 2-Simplices in the SPC.

- $\mathbf{X}_t^{\text{Type } k} \in \mathbb{R}^{N^1_{\text{Spatial}} \times N^{2,t}_{\text{Type } k}}$: isolates just the spatial edges (first $T.|\mathscr{E}_S|$ rows) and captures their incidence with Type k triangles at time t,
- $\mathbf{Y}_t^{\text{Type } k} \in \mathbb{R}^{N^1_{\text{Temporal}} \times N^{2,t}_{\text{Type } k}}$: isolates just the temporal edges (second $N(T-1)$ rows) and captures their incidence with Type k triangles at time t,
- $\mathbf{Z}_t^{\text{Type } k} \in \mathbb{R}^{N^1_{\text{ST}} \times N^{2,t}_{\text{Type } k}}$: isolates just the spatio-temporal edges (last $2|\mathscr{E}_S| \cdot (T-1)$ rows) and captures their incidence with Type k triangles at time t.

These blocks are then vertically stacked to form the full incidence matrix for time step t:

$$\mathbf{B}_{2,t}^{\text{Type } k} = \begin{bmatrix} \mathbf{X}_t^{\text{Type } k} \\ \mathbf{Y}_t^{\text{Type } k} \\ \mathbf{Z}_t^{\text{Type } k} \end{bmatrix} \in \mathbb{R}^{N^1 \times N^{2,t}_{\text{Type } k}}, \tag{4.32}$$

and the complete incidence matrix $\mathbf{B}_{2\diamond}^{\text{Type } k} \in \mathbb{R}^{N_1 \times N_2^{\text{Type } k}}$, is constructed by concatenating the per-time-step matrices column-wise where $N_2^{\text{Type } k} = \sum_{t=1}^{T-1} N^{2,t}_{\text{Type } k}$:

$$\mathbf{B}_{2\diamond}^{\text{Type } k} = \left[\mathbf{B}_{2\diamond,1}^{\text{Type } k} \;\middle|\; \mathbf{B}_{2\diamond,2}^{\text{Type } k} \;\middle|\; \cdots \;\middle|\; \mathbf{B}_{2\diamond,T-1}^{\text{Type } k} \right]. \tag{4.33}$$

Figure 4.13 illustrates the structure of $\mathbf{B}_{2\diamond,t}^{\text{Type } k}$ using its three constituent blocks and Table 4.3 summarizes the per-time-step cardinality of each spatio-temporal 2-simplex. In the remaining section, we follow this general framework to construct the spatio-temporal edge-to-triangle incidence matrices for each type.

CONSTRUCTING $\mathbf{B}_2^{\text{TYPE 1}}$

Each of the blocks $\mathbf{X}_t^{\text{Type 1}}$, $\mathbf{Y}_t^{\text{Type 1}}$, and $\mathbf{Z}_t^{\text{Type 1}}$ has $2|\mathscr{E}_S|$ columns, corresponding to the per-time-step cardinality of Type 1 triangles.

$\mathbf{X}_t^{\text{Type 1}}$: Figure 4.14a illustrates how the block matrix $\mathbf{X}_t^{\text{Type 1}}$ is constructed. In order to account for the participation of the spatial edges in Type 1 triangles, we horizontally stack two identity matrices of size $I_{E_S} \in \mathbb{R}^{|\mathscr{E}_S| \times |\mathscr{E}_S|}$, resulting in a matrix of shape $\mathbb{R}^{|\mathscr{E}_S| \times 2|\mathscr{E}_S|}$. This captures the fact that each spatial edge contributes to two distinct Type 1 triangles

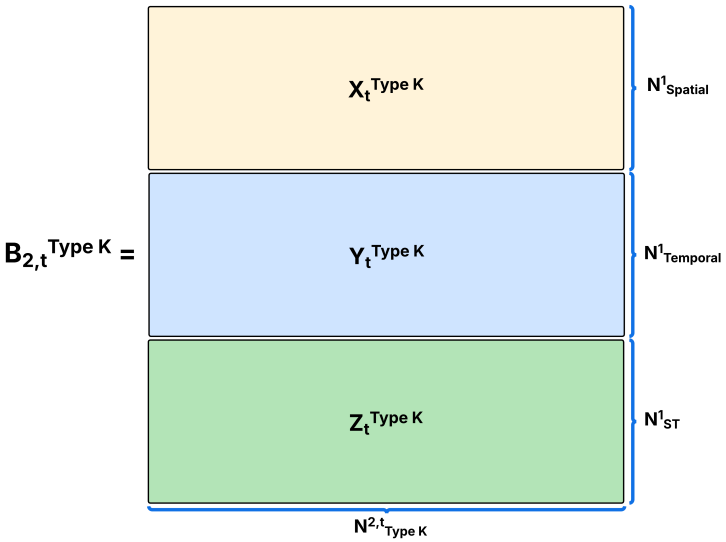
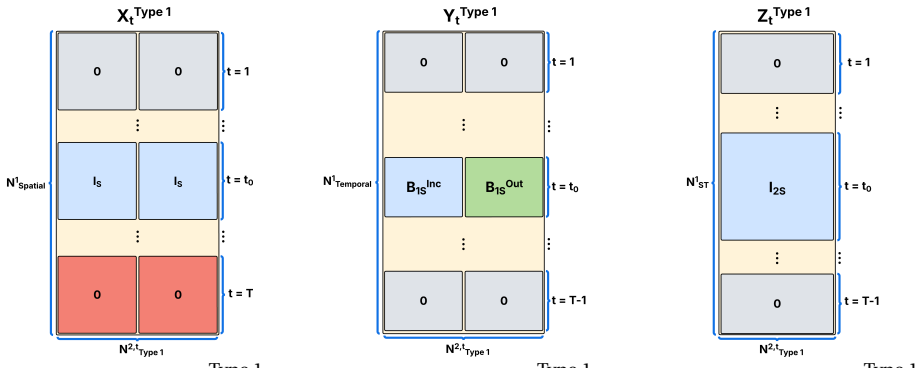


Figure 4.13: The general structure for $\mathbf{B}_{2,t}^{\text{Type } k}$ as a concatenation of $\mathbf{X}_t^{\text{Type } k}$, $\mathbf{Y}_t^{\text{Type } k}$ and $\mathbf{Z}_t^{\text{Type } k}$.

at time t as shown in Figure 4.4. For all other time steps, rows are filled with zeros. Additionally, note that there are no spatial edges contributing to Type 1 triangles at the final time step $t = T$ (shown as the red blocks in the figure), as these spatio-temporal triangles don't originate in the last time step.

$\mathbf{Y}_t^{\text{Type } 1}$: As showin in 4.4, a temporal edge originates from each of the two endpoints of the spatial edge in the two Type 1 triangles created for the spatial edge. The incoming and the outgoing endpoints can be isolated by $\mathbf{B}_{1S}^{\text{Inc}}$ and $\mathbf{B}_{1S}^{\text{Out}}$ respectively. We horizontally stack these two matrices, where the first block corresponds to the temporal edges originating at the incoming nodes, and the second block corresponds to the temporal edges originating at the outgoing nodes. For all other time steps, the rows are filled with zeros. The block matrix for $\mathbf{Y}_t^{\text{Type } 1}$ is shown in 4.14b.

$\mathbf{Z}_t^{\text{Type } 1}$: Figure 4.4 illustrates how the concordant spatio-temporal edges connect the outgoing endpoint of a spatial edge to the temporal edge originating at its incoming node for every spatial edge. Conversely, the discordant edges connect the incoming endpoint to the temporal edge originating at the outgoing node. To encode this structure, we construct an identity matrix $I_{2|\mathscr{E}_S|} \in \mathbb{R}^{2|\mathscr{E}_S| \times 2|\mathscr{E}_S|}$, where the first $|\mathscr{E}_S|$ columns correspond to the contribution of concordant spatio-temporal edges (used in the first triangle), and the next $|\mathscr{E}_S|$ columns correspond to discordant edges (used in the second triangle). Note that using the identity matrix works here since the ordering of the spatio-temporal edges is the same as that of the spatial edges within the concordant and discordant groups. This identity structure is valid since the spatio-temporal edges retain the same ordering as the spatial edges within each group. The constructed block matrix is shown in 4.14c.



(a) Block structure of $\mathbf{X}_t^{\text{Type 1}}$. **(b)** Block structure of $\mathbf{Y}_t^{\text{Type 1}}$. **(c)** Block structure of $\mathbf{Z}_t^{\text{Type 1}}$. Not drawn to scale.

Figure 4.14: Construction of the $\mathbf{B}_{2\diamond}^{\text{Type 1}}$ matrix from its spatial, temporal, and spatio-temporal blocks.

We obtain the full incidence matrix $\mathbf{B}_{2\diamond,t}^{\text{Type 1}}$ by vertically stacking these blocks as described in Equation 4.32 and the final matrix $\mathbf{B}_{2\diamond}^{\text{Type 1}} \in \mathbb{R}^{N^1 \times N^2_{\text{Type 1}}}$ is constructed using Equation 4.33 by stacking each $\mathbf{B}_{2\diamond,t}^{\text{Type 1}}$ column-wise.

CONSTRUCTING $\mathbf{B}_{2\diamond}^{\text{TYPE 2}}$

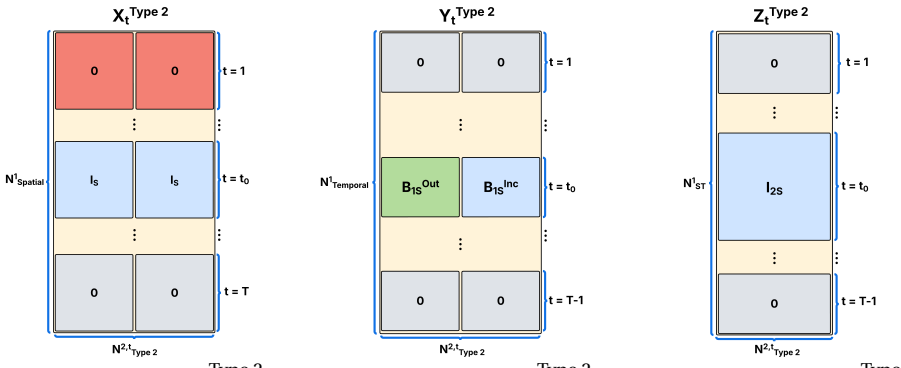
Figure 4.15 illustrates how the block matrices $\mathbf{X}_t^{\text{Type 2}}$, $\mathbf{Y}_t^{\text{Type 2}}$, and $\mathbf{Z}_t^{\text{Type 2}}$ are constructed. Type 2 triangles follow a similar configuration to Type 1 triangles but with the spatial edge shifted to the next time step. Each block again has $2|\mathcal{E}_S|$ columns, corresponding to the per-time-step cardinality of Type 2 triangles.

$\mathbf{X}_t^{\text{Type 2}}$: In contrast to Type 1, the spatial edge in a Type 2 triangle lies at time step $t+1$. To reflect this, we use the same $I_{E_S} \in \mathbb{R}^{|\mathcal{E}_S| \times |\mathcal{E}_S|}$ but in time $t+1$ instead of t, leaving the corresponding entries for all other time steps. Also note that the rows for $t = 1$ are will always be 0 for Type 2 triangles (unlike Type 1 where $t = T$ was always 0), shown as the red blocks in Figure 4.15a.

$\mathbf{Y}_t^{\text{Type 2}}$: As in Type 1, each of the two endpoints of the spatial edge has a temporal edge from time t to $t+1$. However, the stacking order of B_{1S}^{Inc} and B_{1S}^{Out} is reversed. This is because the temporal edge originating at the incoming node is now paired with the discordant spatio-temporal edge instead of the concordant one as illustrated in Figure 4.5. Switching the order of the stacking allows us to keep $\mathbf{Z}_t^{\text{Type 2}}$ the same as $\mathbf{Z}_t^{\text{Type 1}}$.

$\mathbf{Z}_t^{\text{Type 2}}$: The spatio-temporal edge configuration remains identical to Type 1.

Putting these blocks together, we construct the full Type 2 incidence matrix at time as described in Equation 4.32. Finally, the full matrix $\mathbf{B}_{2\diamond}^{\text{Type 2}} \in \mathbb{R}^{N^1 \times N^2_{\text{Type 2}}}$ is obtained using Equation 4.33.



(a) Block structure of $\mathbf{X}_t^{\text{Type 2}}$ **(b)** Block structure of $\mathbf{Y}_t^{\text{Type 2}}$ **(c)** Block structure of $\mathbf{Z}_t^{\text{Type 2}}$. Not drawn to scale.

Figure 4.15: Construction of the $\mathbf{B}_{2\diamond}^{\text{Type 2}}$ matrix from its spatial, temporal, and spatio-temporal blocks.

CONSTRUCTING $\mathbf{B}_2^{\text{TYPE 3}}$

Type 3 triangles consist of one spatial edge and two spatio-temporal edges. Since temporal edges do not participate in this configuration, the temporal block is zero:

$$\mathbf{Y}_t^{\text{Type 3}} = \mathbf{0}_{N^1_{\text{Temporal}} \times N^{2,t}_{\text{Type 3}}}$$

The spatial and spatio-temporal blocks, $\mathbf{X}_t^{\text{Type 3}}$ and $\mathbf{Z}_t^{\text{Type 3}}$, cannot be constructed using simple block structures as in previous cases, and must instead be generated algorithmically.

The pseudocode in Algorithm 1 constructs the spatial block $\mathbf{X}_t^{\text{Type 3}}$. Since each edge of a spatial triangle creates a Type 3 2-simplex as illustrated in Figure 4.6, we iterate over all the spatial triangles present at time step t and add a binary column for each edge, marking its presence in the corresponding triangle. These columns are collected and stacked to form the full spatial block for time t. We assign this block to the rows representing the spatial edges for time step t, leaving all the other edges as 0.

Algorithm 1: Constructing $\mathbf{X}_t^{\text{Type 3}}$ for Type 3 Triangles

Input: Time step t, number of spatial edges per time step $|\mathscr{E}_S|$, number of spatial triangles per time step $|\mathscr{T}_S|$, spatial edge-to-triangle incidence matrix $\mathbf{B}_{2S} \in \mathbb{R}^{|\mathscr{E}_S| \times |\mathscr{T}_S|}$

Output: Spatial block $\mathbf{X}_t^{\text{Type 3}} \in \mathbb{R}^{N^1_{\text{Spatial}} \times N^{2,t}_{\text{Type 3}}}$

$\mathbf{X}_t^{\text{Type 3}} \leftarrow \mathbf{0}_{N^1_{\text{Spatial}} \times N^{2,t}_{\text{Type 3}}}$; // Zero matrix to hold final output

$r_{\text{start}} \leftarrow |\mathscr{E}_S| \cdot t$, $r_{\text{end}} \leftarrow r_{\text{start}} + N^{1,t}_{\text{Spatial}}$; // Row range for spatial edges at time t

all_columns = []

for $j = 1$ **to** $\mathscr{T}_S$ **do**

  $\mathbf{v} \leftarrow \mathbf{B}_{2S}[:, j]$; // Get edges incident on triangle j

  $\mathscr{E}_j \leftarrow \{i \mid \mathbf{v}[i] \neq 0\}$; // Indices for the 3 edges

  foreach $e \in \mathscr{E}_j$ **do**

    $\mathbf{c} \leftarrow \mathbf{0}_{N^{1,t}_{\text{Spatial}} \times 1}$; // Column for triangle j with edge e

    $\mathbf{c}[e] \leftarrow 1$

    all_columns.append($\mathbf{c}$)

// Each of the $N^{2,t}_{\text{Type 3}}$ columns marks the contribution of one spatial edge in each spatial triangle

$\mathbf{X}_t^{\text{Type 3}}[r_{\text{start}} : r_{\text{end}}, :] \leftarrow$ HorizontalStack(all_columns)

return $\mathbf{X}_t^{\text{Type 3}}$

The pseudocode in Algorithm 2 outlines the construction of the spatio-temporal block $\mathbf{Z}_t^{\text{Type 3}}$. As visualized in Figure 4.6, the two spatio-temporal edges in a Type 3 triangle originate from the endpoints of the spatial edge. Therefore, for each such triangle, we identify the spatio-temporal edges that originate at the endpoints of the corresponding spatial edge and mark their contribution as 1 in the incidence column. These columns are collected and stacked to form the complete block, which is then inserted into the rows corresponding to the spatio-temporal edges at time step t, while all other entries are left as zero.

$\mathbf{B}_{2\diamond,t}^{\text{Type 3}}$ can now be constructed as in Equation 4.32 and the full matrix with all time steps $\mathbf{B}_{2\diamond}^{\text{Type 3}} \in \mathbb{R}^{N^1 \times N^2_{\text{Type 3}}}$ is obtained using Equation 4.33.

CONSTRUCTING $\mathbf{B}_{2\diamond}^{\text{TYPE 4}}$

Type 4 triangles share the same structural components as Type 3 triangles consisting of one spatial edge and two spatio-temporal edges. However, their configuration introduces two key differences:

- The spatial edge appears at time step $t+1$ rather than t, as the spatial component of the triangle is shifted forward. To account for this, we modify the row index range in the spatial block construction (Algorithm 1). Specifically, we replace

$$r_{\text{start}} \leftarrow |\mathscr{E}_S| \cdot t \quad \text{with} \quad r_{\text{start}} \leftarrow |\mathscr{E}_S| \cdot (t+1).$$

With this change, the spatial block $\mathbf{X}_t^{\text{Type 4}}$ is constructed using the same procedure as for Type 3 but the spatial edges get shifted one step ahead.

Algorithm 2: Constructing $\mathbf{Z}_t^{\text{Type 3}}$ for Type 3 Triangles

Input: Time step t, number of spatial edges per time step $|\mathcal{E}_S|$, number of spatial triangles per time step $|\mathcal{T}_S|$, spatial edge-to-triangle incidence matrix $\mathbf{B}_{2S} \in \mathbb{R}^{|\mathcal{E}_S| \times |\mathcal{T}_S|}$

Output: Spatio-temporal block $\mathbf{Z}_t^{\text{Type 3}} \in \mathbb{R}^{N^1_{\text{Spatial}} \times N^{2,t}_{\text{Type 3}}}$

$\mathbf{Z}_t \leftarrow \mathbf{0}_{N_1^{\text{ST}} \times 3|T_S|}$; // Zero matrix to hold final output

$r_{\text{start}} \leftarrow |\mathcal{E}_S| \cdot t$, $r_{\text{end}} \leftarrow r_{\text{start}} + N^{1,t}_{\text{ST}}$; // Spatio-temporal edge range for time t

all_columns = [] ; // Will store triangle columns

for $j = 1$ **to** $\mathcal{T}_S$ **do**

  $\mathbf{v} \leftarrow \mathbf{B}_{2S}[:, j]$; // Get edges in triangle j

  $\mathcal{E}_j \leftarrow \{i \mid \mathbf{v}[i] \neq 0\}$; // Get indices of 3 edges in the triangle

  foreach $e_s \in \mathcal{E}_j$ **do**

    $\mathcal{E}_{\text{other}} \leftarrow \mathcal{E}_j \setminus \{e_s\}$; // Edges other than e_s

    $\mathcal{E}_{\text{ST}} \leftarrow$ GETSPATIOTEMPORALEDGES($\mathcal{E}_{\text{other}}$) ; // ST edges induced by $\mathcal{E}_{\text{other}}$

    $\mathbf{c} \leftarrow \mathbf{0}_{N^{1,t}_{\text{ST}} \times 1}$; // Column for triangle j with spatial edge e_s

    foreach $e_{st} \in \mathcal{E}_{ST}$ **do**

      if ISORIGINATINGFROMENDPOINT; // If e_{st} originates from nodes of e_s (e_{st}, e_s) **then**

        $\mathbf{c}[e_{\text{st}}] \leftarrow 1$

    all_columns.append($\mathbf{c}$)

// Each column marks ST edges that pair with spatial edge to form a Type 3 triangle

$\mathbf{Z}_t[r_{\text{start}} : r_{\text{end}}, :] \leftarrow$ HorizontalStack(all_columns)

return $\mathbf{Z}_t$

- In the spatio-temporal block (Algorithm 2), we reverse the role of the matching function. Figure 4.7 demonstrates that the spatio-temporal edges in Type 4 triangles terminate at the endpoints of the spatial edge (that is shifted one step ahead). Therefore, instead of checking whether the spatio-temporal edge *originates from* a node in the spatial edge, we now check whether it *terminates at* one. This is done by replacing the call to ISORIGINATINGFROMENDPOINT with ISTERMINATINGATENDPOINT. With this change, the spatio-temporal block $\mathbf{Z}_t^{\text{Type 4}}$ is generated analogously using Algorithm 2.

As with Type 3, the temporal block is zero:

$$\mathbf{Y}_t^{\text{Type 4}} = \mathbf{0}_{N^1_{\text{Temporal}} \times N^{2,t}_{\text{Type 4}}}$$

Therefore, Algorithms 1 and 2 can be reused for Type 4 triangles with these two modifications. As before, once $\mathbf{X}_t^{\text{Type 4}}$, $\mathbf{Y}_t^{\text{Type 4}}$ and $\mathbf{Z}_t^{\text{Type 4}}$ are constructed, we use 4.32 to construct $\mathbf{B}_{2\diamond,t}^{\text{Type 4}}$ and use 4.33 to construct $\mathbf{B}_{2\diamond}^{\text{Type 4}} \in \mathbb{R}^{N^1 \times N^2_{\text{Type 4}}}$

ASSIGNING ORIENTATIONS

To ensure consistent orientation across simplices, we define the reference orientation of the triangles using the lexicographic ordering of vertices, represented by the ascending orientation set $[i, j, k]$ with $i < j < k$. We then assign signs to the entries in each independently $\mathbf{B}_2$ matrix as follows: for each non-zero row in a given column, representing a participating edge in a triangle, we assign a $+1$ if the local orientation of the edge within the triangle is consistent with its global direction, and a -1 otherwise.

CONSTRUCTING $\mathscr{B}_{2\diamond}$

Having constructed the incidence matrices for each type of 2-simplex, we now define the full parameterized edge-to-triangle incidence matrix of the SPC as:

$$\tilde{\mathbf{B}}_{2\diamond} = \left[\sqrt{f_0} \cdot \mathbf{B}_{2\diamond}^{\text{Spatial}} \,\middle|\, \sqrt{f_1} \cdot \mathbf{B}_{2\diamond}^{\text{Type 1}} \,\middle|\, \sqrt{f_2} \cdot \mathbf{B}_{2\diamond}^{\text{Type 2}} \,\middle|\, \sqrt{f_3} \cdot \mathbf{B}_{2\diamond}^{\text{Type 3}} \,\middle|\, \sqrt{f_4} \cdot \mathbf{B}_{2\diamond}^{\text{Type 4}} \right] \tag{4.34}$$

Here, $\{f_i\}_{i=0}^4$ are learnable non-negative scalar parameters associated with each of the 2-simplices. As with $\tilde{\mathbf{B}}_{1\diamond}$, we make these parameters non-negative by applying the SoftPlus function (Equation 4.29) and the square roots ensure appropriate scaling when computing the parameterized Hodge Laplacians introduced in the next section.

The resulting matrix $\tilde{\mathbf{B}}_{2\diamond} \in \mathbb{R}^{N_1 \times N_2}$ encodes the complete edge-to-triangle structure of the SPC and enables the model to learn the relative importance of each type of 2-simplex during training.

4.5. HODGE LAPLACIANS AND SIMPLICIAL PRODUCT COMPLEX FILTERS

In Section 4.2, we introduced the non-parameterized incidence matrices $\mathbf{B}_{1\diamond}$ and $\mathbf{B}_{2\diamond}$, and in the previous section we introduced their parameterized counterparts $\tilde{\mathbf{B}}_{1\diamond}$ and $\tilde{\mathbf{B}}_{2\diamond}$. We now use these matrices to define the non-parameterized and parameterized Hodge Laplacians of the SPC and the corresponding non-parameterized and parameterized Simplicial Product Complex Convolutional Filters (SPCCF)

4.5.1. HODGE LAPLACIANS OF THE SPC

NON-PARAMETERIZED

The non-parameterized Hodge Laplacians for the SPC capture spatio-temporal interactions across the upper and lower simplex orders while treating all simplex types uniformly, without incorporating any type-specific weighting. We can define the 0^{th}-, 1^{st}- and 2^{nd}- order non-parameterized Hodge Laplacians of the rank-2 SPC, represented as $\mathbf{L}_{0\diamond}$, $\mathbf{L}_{1\diamond}$, and $\mathbf{L}_{2\diamond}$ respectively, by using the non-parameterized incidence matrices $\mathbf{B}_{1\diamond}$ and $\mathbf{B}_{2\diamond}$ in Equation 2.22. In the context of the SPC, these Laplacians support the modeling of structured spatio-temporal dynamics by enabling signal propagation using higher-order spatio-temporal interactions.

PARAMETERIZED

The parameterized forms of the Hodge Laplacians of the SPC allow learning the importance of each simplex type during training as they utilize the parameterized incidence matrices $\tilde{\mathbf{B}}_{1\diamond}$ and $\tilde{\mathbf{B}}_{2\diamond}$. We define the 0^{th}-, 1^{st}- and 2^{nd}- order parameterized Hodge Laplacians of the rank-2 SPC, represented as $\tilde{\mathbf{L}}_{0\diamond}$, $\tilde{\mathbf{L}}_{1\diamond}$, and $\tilde{\mathbf{L}}_{2\diamond}$ respectively, by using the parameterized incidence matrices $\tilde{\mathbf{B}}_{1\diamond}$ and $\tilde{\mathbf{B}}_{2\diamond}$ in Equation 2.22.

4.5.2. SIMPLICIAL PRODUCT COMPLEX CONVOLUTIONAL FILTER

Given the k^{th} order Hodge Laplacian of the SPC $\mathbf{L}_{k\diamond}$, we define the non-parametric version of the SPCCF, represented by $H_k(\mathbf{L}_{k\diamond,d}, \mathbf{L}_{k\diamond,u})$ using the topological filter defined in Equation 2.32. That is, the filter is the same as the topological filter but it operates on the Hodge Laplacians of a spatio-temporal SPC instead of the Hodge Laplacian of a regular spatial simplicial complex. Therefore, the locality, shift-invariance and permutation-invariance properties of topological filters defined on purely spatial topologies also extend to the spatio-temporal domain in the SPC framework.

We can similarly define the parameterized version of the filter that weighs each simplex type by replacing $\mathbf{L}_{k\diamond,d}$ and $\mathbf{L}_{k\diamond,u}$ with $\tilde{\mathbf{L}}_{k\diamond,d}$ and $\tilde{\mathbf{L}}_{k\diamond,u}$ in Equation 2.32 respectively. We represent the parameterized filter on the SPC as $\tilde{H}_k(\tilde{\mathbf{L}}_{k\diamond,d}, \tilde{\mathbf{L}}_{k\diamond,u})$. Since we restrict the SPC to rank 2, we define both versions of the filters up to $k = 2$.

4.6. SIMPLICIAL PRODUCT COMPLEX CONVOLUTIONAL NEURAL NETWORK

We introduce the Simplicial Product Complex Convolutional Neural Network (SPCCNN), a multi-layered architecture where each layer comprises of an SPCCF followed by a non-linearity and an inter-simplicial propagation mechanism for each simplex order. Formally, consider an L-layer SPCCNN and let $H_k^l(\mathbf{L}_{k\diamond,d}, \mathbf{L}_{k\diamond,u})$ denote the SPCCF for the k^{th}-order simplicial signals ($k \in \{0,1,2\}$) at layer $l \in [1,\dots,L]$. Let σ be a pointwise non-linearity.

The intermediate output (prior to cross-order propagation) for each k is computed as:

$$\tilde{\mathbf{s}}_l^k = \sigma\left(H_k^l(\mathbf{L}_{k\diamond,d}, \mathbf{L}_{k\diamond,u})\mathbf{s}_{l-1}^k\right). \tag{4.35}$$

We then incorporate inter-simplicial information sharing by propagating features from higher-dimensional simplices to lower-dimensional ones. Specifically, for each simplex order $k \in \{0,1\}$, the final updated signal at layer l is computed as:

$$\mathbf{s}_l^k = \tilde{\mathbf{s}}_l^k + \sum_{r>k} \mathbf{P}_{r\to k}\tilde{\mathbf{s}}_l^r, \tag{4.36}$$

where $\mathbf{P}_{r\to k}$ is a learnable projection operator that aggregates higher-order contextual information from order-r to order-k simplices. For example, node-level features $\mathbf{s}_l^0$ are updated using edge and face signals, and edge-level features $\mathbf{s}_l^1$ incorporate signals from faces. Since our SPC is restricted to rank-2, face-level features $\mathbf{s}_l^2$ are not updated from higher-order simplices. In this work, these operators are implemented by projecting signals via incidence matrices followed by a learnable linear layer.

To increase expressivity, we extend this to a filter bank setting where each layer processes multi-channel inputs and outputs per k, adapting Equations 2.34 to 2.38 by modifying Equation 2.37 to incorporate cross-order propagation. By recursively applying these filters across layers, the final outputs $\mathbf{S}_L^k$ for each $k \in \{0,1,2\}$ are obtained for downstream tasks.

In order to obtain the parameterized version of the SPCCNN, the convolutional filters H_k^l can optionally be replaced by the parameterized versions $\tilde{H}_k^l$, to enable learning the importance of different simplex types. This results in two variants of the architecture: the parameterized SPCCNN (P-SPCCNN) and the non-parameterized SPCCNN (NP-SPCCNN).

Together, this framework provides a general and extensible approach for constructing spatio-temporal simplicial neural networks on the SPC. By leveraging higher-order space-time Hodge Laplacians introduced in Section 4.5, existing simplicial neural networks [45, 78, 79] can also be extended to handle time-varying higher-order signals while learning type-specific structure from data in an end-to-end fashion.

4.7. DISCUSSION

In this chapter, we introduced a framework to construct the SPC, which retains a strictly simplicial structure in the product space. We then examined the various types of simplices formed in the SPC and developed parameterized Hodge Laplacians and filters, enabling the design of SPCCNNs that can learn the relative importance of different higher-order spatio-temporal relationships in a data-driven manner.

Before we evaluate the performance of our architecture in the next chapter, we discuss the scalability and complexity limitations of our model and also present a detailed comparison between our approach and the product cell complex.

4.7.1. COMPLEXITY ANALYSIS AND SCALABILITY CHALLENGES

Parameter Complexity. The standard topological filter (Equation 2.32) defined on the SPC contains $(1+L_1+L_2)$ learnable parameters per feature map, where L_1 and L_2 denote the filter orders over the lower and upper convolutions, respectively. For an SPCCNN model with P layers and F input and output features per layer, the number of parameters per simplex order (nodes, edges, or faces) is $\mathcal{O}(PF^2(L_1+L_2))$. Assuming L_1, L_2, and F are consistent across all simplicial orders, and denoting the rank of the SPC by R, the total number of parameters in the SPCCNN is $\mathcal{O}(RPF^2(L_1+L_2))$.

In the case of the parameterized P-SPCCNN model, additional learnable parameters are introduced through the simplex-type-specific coefficients in the parameterized Hodge Laplacians:

- Each node filter incurs 3 additional parameters for the three edge types in the parameterized $\tilde{\mathbf{L}}_{0\diamond}$.
- Each edge filter incurs 8 additional parameters: 3 from the parameterized lower Laplacian $\tilde{\mathbf{L}}_{1\diamond,d}$ (edge types), and 5 from the parameterized upper Laplacian $\tilde{\mathbf{L}}_{1\diamond,u}$ (face types).

- Each face filter incurs 5 additional parameters, corresponding to the five face types in $\tilde{\mathbf{L}}_{2\diamond}$.

The number of model parameters of the SPCCNN does not scale with the number of nodes, edges, or triangles in the SPC. This makes the SPCCNN architecture parameter-efficient with larger graphs.

Computational Complexity. If D denotes the maximum number of neighboring simplices in both upper and lower adjacencies, then the computational complexity of filtering a single scalar-valued simplex on the SPC is $\mathcal{O}((L_1 + L_2)D)$.

For an SPCCNN model with F input and output features and P layers, the per-simplex filtering complexity scales as $\mathcal{O}((L_1 + L_2)DF^2P)$. Considering the number of simplices in the SPC, we analyze the complexity for each simplex order:

- **Nodes:** Since the number of nodes in the SPC grows by a factor of NT, where N is the number of spatial nodes and T is the temporal window, the overall complexity for node-level filtering in the SPCCNN becomes $\mathcal{O}((L_1 + L_2)DF^2PNT)$.
- **Edges:** As shown in Table 4.1, the number of edges in the SPC grows with $\mathcal{O}(T(N + |\mathcal{E}_S|))$, where $|\mathcal{E}_S|$ is the number of edges in the spatial graph. Therefore, the edge-level filtering complexity in the SPCCNN is $\mathcal{O}((L_1 + L_2)DF^2PT(N + |\mathcal{E}_S|))$.
- **Faces:** As shows in Table 4.2, the number of faces scales in the SPC with $\mathcal{O}(T(|\mathcal{E}_S| + |\mathcal{T}_S|))$, where $|\mathcal{T}_S|$ is the number of triangles in the spatial graph. Hence, the filtering complexity for faces in the SPCCNN becomes $\mathcal{O}((L_1 + L_2)DF^2PT(|\mathcal{E}_S| + |\mathcal{T}_S|))$

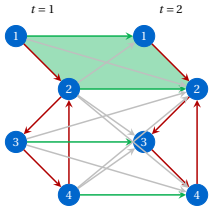
Scalability Challenges. The computational complexity of SPCCNN scales linearly with the number of spatial nodes, edges, and triangles, as well as the temporal observation window T. While theoretically manageable, this becomes prohibitive in practice for large graphs or long time horizons, as the size of the SPC and, thus the number of simplices, increases rapidly.

In the parameterized variant, edge and face weights are embedded in the Hodge Laplacians, making the shift operators parameter-dependent. This prevents precomputing shifted signals and introduces significant overhead during training and inference.

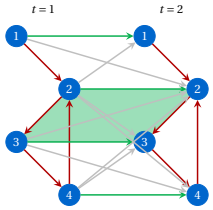
To mitigate similar issues, GTCNN [59] exploits Kronecker product structure to decouple spatial and temporal shifts and shift parameterization to the filter coefficients, enabling efficient recursive computation. However, this strategy is not applicable to SPCCNN, as the higher-order simplicial structure of the SPC cannot be expressed via Kronecker products. Consequently, we cannot decouple the parameters or derive a recursive formulation. Due to these computational constraints, we restrict our experiments to tractable SPC sizes and discuss potential scalability solutions as part of future work in Chapter 6.

4.7.2. COMPARISON WITH THE PRODUCT CELL COMPLEX FRAMEWORK

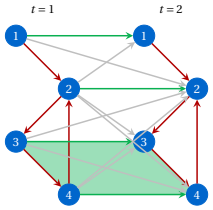
As discussed in Section 2.3.4, the product cell complex framework constructs a cell complex in the product space by defining spatio-temporal cells between consecutive time



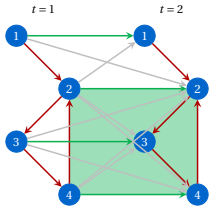
(a) The cell created by spatial edge **1** → **2**



(b) The cell created by spatial edge **2** → **3**



(c) The cell created by spatial edge **3** → **4**



(d) The cell created by spatial edge **4** → **2**

Figure 4.16: Cells created in the product cell complex which are subsumed by Type 1 (Figure 4.4) and Type 2 (Figure 4.5) triangles in the SPC.

steps. In contrast, our work maintains a strictly simplicial structure by constructing and operating on the SPC. We argue that our framework provides a richer topological representation and more flexibility through parameterization.

Richer Topological Structure. Consider cells created in the product cell complex for each edge shown in Figure 4.16. Each spatial edge induces a single spatio-temporal cell between adjacent time steps. In the SPC, however, this interaction is captured through Type 1 and two Type 2 triangles, each visualized in Figures 4.4 and 4.5. These triangles effectively subdivide the corresponding spatio-temporal cell into finer simplicial components. Thus, the structural information encoded by a product cell is fully subsumed by these simplices in the SPC.

Beyond this equivalence, the SPC additionally captures multi-node spatio-temporal interactions through Type 3 and Type 4 triangles. These encode the joint influence of two spatial nodes on a third node across time, patterns that cannot be directly represented within the cell complex framework. While the product cell complex may indirectly approximate such interactions through multi-hop paths between cells, the SPC captures them natively and explicitly through its simplicial structure.

Since the interactions captured by spatio-temporal cells are fully represented by Type 1 and Type 2 triangles in the SPC, and the SPC further incorporates more complex relationships through Type 3 and Type 4 triangles, we argue that the SPC constitutes a more expressive topological space for modeling higher-order spatio-temporal dynamics.

Parameterization. As discussed in previous sections, retaining a simplicial structure also enables parameterization. Unlike the product cell complex, where each cell encodes a uniform type of spatio-temporal interaction, the SPC yields a diverse and well-

defined set of simplex types, each associated with a specific higher-order relationship across space and time. This structural differentiation makes it possible to assign type-specific parameters within the Hodge Laplacians, as done in Section 4.5, allowing the model to learn the relative importance of distinct spatio-temporal patterns directly from the data. The SPC therefore provides the flexibility to adapt to higher-order inductive biases without assuming them in advance.

In summary, the SPC serves as a more expressive and adaptable framework compared to the product cell complex framework. Moreover, SPCCNNs defined on the SPC capture a wide range of higher-order spatio-temporal interactions while supporting parameterized learning over structurally distinct simplex types.

5

EMPIRICAL EVALUATION

In this chapter, we present the numerical experiments conducted to evaluate the proposed SPCCNN model on multivariate time series modeling. We begin by outlining the experimental setup and the design of our evaluation procedure in Section 5.1. We then provide the results of our framework on downstream tasks across various datasets in Section 5.2. We conclude with a discussion of the results and key observations in Section 5.3.

5.1. EXPERIMENTAL SETUP

This section outlines the experimental setup for the empirical evaluation of our framework. We first introduce the downstream node-level forecasting task in Section 5.1.1. We then describe the datasets used in our experiments and the baseline models we benchmark against in Sections 5.1.2 and 5.1.3 respectively. Finally, we discuss the training and evaluation settings for our experiments in Section 5.1.4.

5.1.1. NODE TIME SERIES FORECASTING

Forecasting is one of the foundational tasks in multivariate data modeling and involves predicting future values based on historical observations. In our setup, a multivariate time series consists of N correlated and regularly sampled time sequences, each associated with a distinct spatial node. Each individual time series is a time-varying signal $[\mathbf{x}_t^v]_i \in \mathbb{R}^F$ on a node, where $i \in \{1,\dots,N\}$ denotes the node index, t represents the time index and F is the number of raw features. The collection of signals at time t can be represented as a matrix $\mathbf{X}_t \in \mathbb{R}^{N\times F}$, and the sequence of signals over a window $[t, t+T]$ is denoted as $\mathbf{X}_{t:t+T} \in \mathbb{R}^{T\times N\times F}$.

The forecasting task aims to predict future values over a given prediction horizon $H \geq 1$. Specifically, at each time step t, the objective is to predict the future values $\mathbf{X}_{t+h}$ for all $h \in \{1,\dots,H\}$, based on the past T observations $\mathbf{X}_{t-T:t}$. When the forecasting horizon $H > 1$, the predictions are generated in an autoregressive manner, where each subsequent forecast is appended to the input sequence and used recursively to predict further into the future. In addition, we assume access to a static spatial graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$,

which encodes the relational structure among the nodes.

Our objective is then to train a predictive model $F(\cdot;\theta)$, parameterized by θ, that estimates the future signal at time $t+H$ based on historical inputs from time $t-T$ to t and the graph structure $\mathcal{G}$. Since the predictions are autoregressive, we can formalize the predictive model as:

$$\hat{\mathbf{Y}}_{t+h} = F(G, [\mathbf{X}_{t-T+h:t}, \hat{\mathbf{Y}}_{t:t+h}]; \theta), \quad \forall h \in \{1, \dots, H\}. \tag{5.1}$$

We organize our dataset $\mathcal{D} = \{(X^{(i)}, Y^{(i)})\}_{i=1}^{|\mathcal{D}|}$ as a collection of input-output pairs, where $X^{(i)} \in \mathbb{R}^{T\times N\times F}$ represents the input signals in the observation window and $Y^{(i)} \in \mathbb{R}^{H\times N\times F}$ contains the corresponding future ground-truth values. For each sample $(X^{(i)}, Y^{(i)})$, the model generates a sequence of predictions $\hat{Y}^{(i)} \in \mathbb{R}^{H\times N\times F}$ representing predictions for time steps $[t+1, \cdots, t+H]$. We compute the loss using the prediction and ground truth at the final time step $t+H$, denoted as $\hat{Y}_H^{(i)}$ and $Y_H^{(i)}$ respectively. The model parameters are therefore optimized by minimizing the average loss over all samples in our dataset as:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \ell\left(\hat{Y}_H^{(i)}, Y_H^{(i)}\right), \tag{5.2}$$

While the proposed SPC framework is capable of modeling time-varying signals on higher-order simplices, we focus our evaluation on node-level forecasting due to the availability of publicly accessible datasets for time-varying node-level signals. Furthermore, many baseline methods considered in our comparison are limited to processing node-level data, making this setting the most appropriate for fair and comprehensive evaluation.

5.1.2. DATASETS

We evaluate our framework on the forecasting task on datasets from two domains: energy demand and traffic monitoring. Both datasets provide different variable and sample sizes.

SOLAR ENERGY DATASET

The Solar Energy dataset [43], collated by the National Renewable Energy Laboratory, records the solar power output (in megawatts) for 137 photovoltaic stations in Alabama, USA. The data is sampled every 5 minutes throughout 2006, resulting in a total of 105,120 measurements per station.

METR-LA DATASET

The METR-LA dataset [39] consists of traffic speed readings from 207 sensors installed on highways in the Los Angeles County. The dataset spans a period of four months, with readings sampled every 5 minutes, resulting in a total of 34,272 measurements per sensor. Each reading represents the average vehicle speed (in miles per hour) at a sensor location. We use the processed data provided by [47].

DATA PREPARATION

Splitting and normalization: We split the datasets into training, validation, and test sets using an 80/10/10 chronological split, ensuring that there is no data leakage between the splits. Following the approach in [16], we apply Z-score normalization to both datasets. Specifically, we use our training split of the dataset to compute the node and feature-wise mean and standard deviation vectors $\boldsymbol{\mu} \in \mathbb{R}^{N \times F}$ and $\boldsymbol{\sigma} \in \mathbb{R}^{N \times F}$ respectively, and normalize each split as

$$\tilde{\mathbf{X}} = \frac{\mathbf{X} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \quad \tilde{\mathbf{Y}} = \frac{\mathbf{Y} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}. \tag{5.3}$$

Note that the division here is element-wise. The loss is computed on the normalized data for all models. During evaluation, we rescale the predictions back to the original scale and calculate the evaluation metrics described in Section 5.1.4 on the rescaled data. The rescaling is done as follows:

$$\hat{\mathbf{Y}} = \hat{\tilde{\mathbf{Y}}} \cdot \boldsymbol{\sigma} + \boldsymbol{\mu}. \tag{5.4}$$

where $\hat{\mathbf{Y}}$ is the final re-scaled prediction and $\hat{\tilde{\mathbf{Y}}}$ is the normalized prediction.

Constructing the spatial graph: We construct the spatial graph for each dataset using a k-nearest neighbor (k-NN) approach based on geographic distance, computed from the latitude and longitude coordinates of the nodes. We set $k = 3$ across both datasets to keep the number of higher-order simplices in the SPC computationally tractable, given the scalability constraints discussed in Section 4.7.1. Notably, the 3-nearest neighbor graph is not fully connected for SOLAR and METR-LA.

The temporal observation window T is fixed at 3 for both datasets. Table 5.1 summarizes the number of spatial nodes, edges and triangles, as well as the number of nodes, edges, and triangles in the corresponding SPC. Visualizations of the constructed spatial graphs for both datasets are provided in Appendix B.

5.1.3. BASELINES

We consider a diverse set of baselines for comparison, spanning both classical and deep learning-based approaches. These include simple statistical forecasting models, feed-forward models, and several variants of spatio-temporal models. The baselines are described below:

STRUCTURE-UNAWARE STATISTICAL BASELINE

- **ARIMA [13]** The Auto-Regressive Integrated Moving Average model implemented using a Kalman filter. Each node's time series is modeled independently, without incorporating spatial structure.

NON-SPATIO-TEMPORAL DEEP LEARNING BASELINES

- **FFNN:** A fully connected feedforward neural network that concatenates raw features from all timesteps in the observation window to form a flat input vector. This model lacks both spatial and temporal inductive biases.

Statistic	SOLAR	METR-LA
Spatial Graph		
Nodes	137	207
Edges	258	384
Triangles	132	227
SPC		
0-Simplices	411	621
1-Simplices		
Spatial	774	1152
Temporal	274	414
ST	1032	1563
Total	*2080*	*3129*
2-Simplices		
Spatial	396	681
Type 1	1032	1536
Type 2	1032	1536
Type 3	792	1362
Type 4	792	1362
Total	*4044*	*6477*

Table 5.1: Spatial graph and SPC statistics for the SOLAR and METR-LA datasets.

- **LSTM** [67] A fully connected LSTM that models each node's time series independently. While LSTM parameters are shared across nodes, there is no interaction between nodes during training. We use a sequence-to-sequence encoder-decoder architecture with peephole LSTM cells. This baseline captures only temporal dependencies.
- **GCNN** [42] A graph convolutional neural network that models spatial relationships using graph convolutions. For each node, input features across the observation window are concatenated and passed through multiple GCN layers. This baseline captures only spatial dependencies.

STRUCTURE-AWARE SPATIO-TEMPORAL BASELINES

- **GNN-RNN:** A spatio-temporal model that processes spatial and temporal dependencies in a decoupled manner. First, a GCN layer models spatial relationships, followed by an LSTM that captures temporal dynamics. This baseline incorporates both inductive biases, but disjointly.
- **GTCNN** [59] A graph temporal convolutional neural network based on parametric product graphs as described in Section 3.3. It models both spatial and temporal dependencies jointly through convolutional filters on the product graph.
- **STGCN** [82] The Spatial-Temporal Graph Convolutional Network as described in Section 3.3 that integrates graph convolutions with gated temporal convolutions in a unified architecture.

- **GW (Graph WaveNet) [76]** A dynamic spatio-temporal model as described in Section 3.3 that combines graph convolutions with adaptive adjacency matrices and dilated causal convolutions for temporal modeling.

TOPOLOGY-AWARE SPATIO-TEMPORAL BASELINE

- **CELL[57]** We construct the product cell complex of rank-2. To make it comparable to our work, we use the same TNN as our framework but using the edge-to-face incidence matrix of the cell complex rather than our edge-to-triangle incidences of the SPC. We use the non-parameterized Hodge Laplacians in order to keep it similar to the original product cell complex which did not propose any parameterization.

PROPOSED BASELINE

- **SNN-RNN:** We introduce a novel baseline architecture to model spatio-temporal data. SNN-RNN lifts the spatial graph to a rank-2 Simplicial Complex using clique lifting and then applies the SCCNN [78] on the lifted spatial topology to capture spatial dependencies, followed by an LSTM for temporal modeling. Like GNN-RNN, spatial and temporal processing are decoupled, but this baseline additionally encodes higher-order topological information.

Further details about the architectures used for each baseline are provided in Appendix A.2.

5.1.4. TRAINING AND EVALUATION SETTINGS

We now describe the training and evaluation settings used for our experiments. We begin by discussing the encoder-process-readout architectural design used to train all models. We then describe our hyperparameter tuning and training strategy.

ENCODER-PROCESS-READOUT ARCHITECTURE

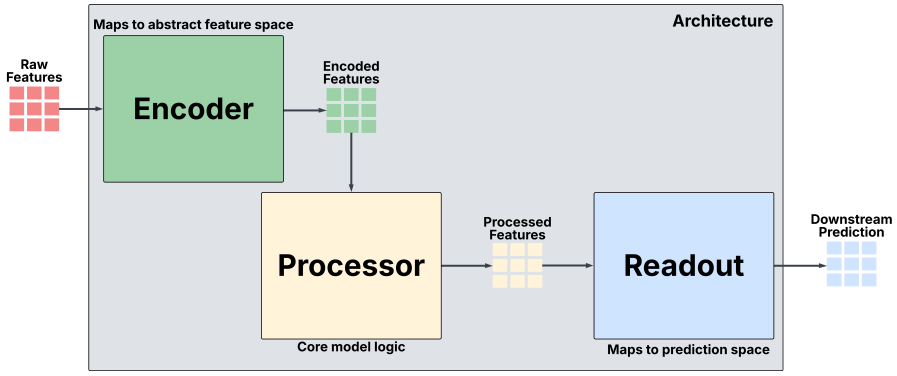


Figure 5.1: The encoder-processor-readout architecture used to train all the models.

We adopt an encoder-processor-readout architecture [6] to train all the deep-learning models, illustrated in Figure 5.1. The Encoder block takes the raw input signals and maps

them to a higher-dimensional abstract feature space. The Processor block implements the core model logic and serves as the backbone of the architecture. Each model receives the encoded feature representations and processes them according to its spatio-temporal inductive biases. Finally, the Readout block transforms the processed features back to the prediction space for the downstream forecasting task. Further details about the encoder-processor-readout configurations used for each model are provided in Appendix A.1.

HYPERPARAMETER TUNING AND TRAINING STRATEGY

We use the Optuna framework [2] to perform hyperparameter tuning for each model's processor block. For forecasting, we optimize for 1-step validation loss during hyperparameter tuning and the best-performing configuration from the 1-step forecasting task is used across all prediction horizons. For each model–dataset combination, we run 100 trials using the Tree-structured Parzen Estimator (TPE) [71] provided by Optuna and apply median-based pruning to prune off under-performing trials early. Details of the tuning procedure and the search space for all baselines are provided in Appendix A.2.

All models are trained using the Mean Squared Error (MSE) loss on normalized signals with early stopping and a learning rate scheduler. We add a weight decay (L_2 regularization) on all the model parameters as a regularizer (including on the edge and face parameters in the parameterized SPCCNN). We use the Adam optimizer and Xavier weight initialization for all experiments. We train all our models on the DelftBlue [1] High-Performance Computing (HPC) cluster at TU Delft, utilizing NVIDIA A100 and V100 GPUs with 5 GB RAM. All models are trained on 10 random seeds and we report the average across the seeds in the results below. Further details about the training settings are provided in the appendix in Table A.2.

The SPCCNN architecture uses averaging for the node-to-edge lifting where the edge features are created by averaging the node features at the endpoints for every edge. We use a rank-1 SPC for both datasets (signals are not lifted to faces) in order to reduce memory and computational load. The final SPCCNN configuration and its search space are summarized in Appendix A.2.

EVALUATION METRICS

We evaluate our framework using two widely used regression metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). MAE measures the average magnitude of the prediction errors without considering their direction, making it easy to interpret and robust to outliers. RMSE, on the other hand, squares the errors before averaging, thereby penalizing larger errors more heavily, making it more sensitive to outliers than MAE.

Each metric is computed on the rescaled predictions $\hat{Y}_H$ and the corresponding ground-truth values $Y_H^{(i)}$, as defined in Equation 5.4. All metrics are computed for the final step H of the prediction horizon, averaged over all samples in the evaluation set. More details on the metrics are available in Appendix A.3.

5.2. RESULTS

We structure this section as follows. We begin by benchmarking the proposed SPCCNN architecture against all baseline models on the downstream forecasting task in Section 5.2.1. This is followed by experiments that measure the impact of parameterization on the performance and flexibility of the SPCCNN. We then investigate the effect of $\mathbf{L_1}$ regularization on the edge and face parameters and determine its impact on the scalability of the SPCCNN model in Section 5.2.2. Following the empirical evaluation of the parameterized SPCCNN in terms of performance, flexibility, and scalability, we conduct a spectral analysis of the SPC framework to understand the spectral properties of $\mathbf{L}_{1\diamond}$, the 1-Hodge Laplacian of the SPC, in Section 5.2.3. Finally, we present a brief case study applying our framework to a spatio-temporal imputation task on a self-curated air pollution monitoring dataset in Section 5.2.4.

5.2.1. FORECASTING RESULTS

We organize the discussion around three key aspects: forecasting performance, data efficiency, and parameter efficiency.

PERFORMANCE ANALYSIS

Table 5.2 reports the forecasting performance of all models for 1-step, 2-step, and 3-step horizons on the METR-LA and Solar datasets. The results represent the average performance across 10 random seeds on the test set, with standard deviation shown on the right. As expected, model performance generally degrades with increasing prediction horizon. This degradation is particularly pronounced among non-spatio-temporal baselines such as FFNN, LSTM, and GCNN, which lack mechanisms to jointly model spatial and temporal dependencies. Their consistently poor performance across both datasets underscores the importance of explicitly incorporating spatio-temporal structure in time series forecasting. ARIMA performs comparably to the non-spatio-temporal deep learning baselines. This is likely because ARIMA is trained independently for each node, allowing it to better capture localized temporal patterns. In contrast, the deep learning baselines are global models that process all nodes jointly without explicitly modeling the spatio-temporal relationships among them.

Among the spatio-temporal baselines, Graph WaveNet (GW) achieves the best overall performance across both datasets and all forecasting horizons. Its ability to learn the graph structure adaptively, combined with dilated causal convolutions for modeling long-range dependencies, provides a strong advantage in forecasting tasks. The STGCN model also performs well and consistently ranks as the second-best model on the Solar dataset, likely due to its use of spatial graph convolutions combined with gated temporal modules, which are well suited for the strong periodicity present in solar energy signals.

The proposed SPCCNN model consistently ranks as the second best model on the METR-LA dataset, outperforming STGCN, GTCNN and CELL. Compared to GTCNN, which operates on a strong product graph, SPCCNN benefits from modeling higher-order spatio-temporal relationships through its simplicial representation. Additionally, the improved performance over CELL supports our claims about the benefits of maintaining a simplicial structure in the product space, as discussed in 4.7.2.

On the Solar dataset, however, SPCCNN performs comparably to GTCNN and STGCN

Data	Models	1-Step		2-Step		3-Step	
		MAE	RMSE	MAE	RMSE	MAE	RMSE
METR-LA	ARIMA	5.21	9.21	6.27	11.70	6.58	14.15
	FFNN	4.59 ±0.19	7.90 ±0.20	5.23 ±0.20	9.05 ±0.15	5.62 ±0.23	9.81 ±0.18
	LSTM	5.18 ±0.17	8.62 ±0.09	5.82 ±0.16	9.75 ±0.14	6.01 ±0.21	10.29 ±0.25
	GCNN	6.41 ±0.14	10.42 ±0.27	6.73 ±0.09	10.87 ±0.16	7.00 ±0.11	11.25 ±0.19
	GNN-RNN	4.51 ±0.09	9.02 ±0.20	4.79 ±0.16	9.30 ±0.16	5.22 ±0.17	9.93 ±0.23
	GTCNN	3.01 ±0.11	5.73 ±0.05	3.53 ±0.20	6.52* ±0.10	3.77 ±0.17	7.06 ±0.09
	STGCN	3.12 ±0.21	5.77 ±0.17	3.68 ±0.24	6.60 ±0.18	4.11 ±0.18	7.34 ±0.12
	GW	**2.77** ±0.14	**5.41** ±0.09	**3.27** ±0.10	**6.21** ±0.06	**3.69** ±0.08	**6.89** ±0.08
	CELL	2.99* ±0.09	5.65* ±0.05	3.41* ±0.08	6.46 ±0.02	3.87 ±0.13	7.16 ±0.08
	SNN-RNN (ours)	3.05 ±0.11	5.69 ±0.08	3.52 ±0.08	6.54 ±0.08	3.92 ±0.12	7.21 ±0.05
	P-SPCCNN (ours)	2.96 ±0.08	5.64 ±0.04	3.39 ±0.08	6.46 ±0.04	3.86* ±0.13	7.16 ±0.08
SOLAR	ARIMA	0.48	2.04	0.63	2.82	0.85	3.30
	FFNN	0.55 ±0.05	0.91 ±0.05	0.79 ±0.06	1.38 ±0.06	1.07 ±0.04	1.74 ±0.05
	LSTM	0.68 ±0.02	1.07 ±0.02	0.88 ±0.03	1.47 ±0.03	1.06 ±0.03	1.71 ±0.03
	GCNN	1.41 ±0.00	2.23 ±0.01	1.51 ±0.01	2.53 ±0.01	1.56 ±0.01	2.60 ±0.01
	GNN-RNN	1.35 ±0.03	2.12 ±0.05	1.45 ±0.02	2.41 ±0.05	1.50 ±0.03	2.50 ±0.06
	GTCNN	0.38 ±0.01	0.79 ±0.00	0.56 ±0.01	1.23 ±0.01	0.70 ±0.01	1.46 ±0.01
	STGCN	0.37 ±0.01	0.75 ±0.01	0.52 ±0.01	1.14 ±0.01	0.63 ±0.01	1.33 ±0.01
	GW	**0.35** ±0.01	**0.74** ±0.01	**0.50** ±0.01	**1.12** ±0.02	**0.61** ±0.01	**1.30** ±0.02
	CELL	0.37 ±0.00	0.78 ±0.00	0.56 ±0.01	1.21 ±0.01	0.70 ±0.01	1.44 ±0.01
	SNN-RNN (ours)	0.37 ±0.01	0.77* ±0.01	0.54* ±0.01	1.17* ±0.01	0.66* ±0.01	1.36* ±0.02
	P-SPCCNN (ours)	0.38 ±0.00	0.78 ±0.00	0.55 ±0.01	1.20 ±0.01	0.70 ±0.01	1.44 ±0.02

Table 5.2: Downstream node-level forecasting performance for all models on the Solar and METR-LA datasets. Best values are **bolded**, second-best underlined, and third-best marked with *.

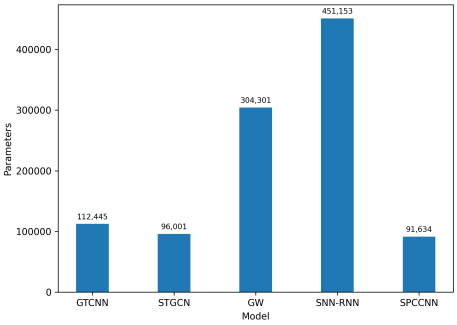
and does not consistently outperform them. This suggests that higher-order simplicial structures may be less relevant for this dataset. We provide evidence supporting this observation below by analyzing the learned parameter values for both datasets.

Finally, our proposed SNN-RNN model also performs competitively, often surpassing GTCNN and STGCN despite its simpler architecture. It also consistently outperforms GNN-RNN, another disjoint spatio-temporal model that does not capture higher-order spatial dependencies. These results further demonstrates the value of incorporating higher-order structures in spatio-temporal modeling, even within simple architectural designs.

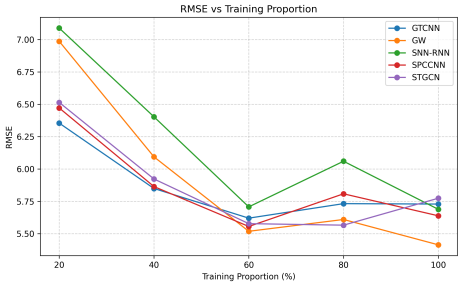
PARAMETER AND DATA EFFICIENCY

While Graph WaveNet achieves the best performance in our experiments, it does so with a significantly larger number of parameters than other models. As shown in Figure 5.2a, GW has the highest parameter count, followed by GTCNN and STGCN. SPCCNN has the lowest number of parameters across all top-performing models, suggesting that SPCCNN provides a favorable balance between performance and parameter efficiency.

Figure 5.2b explores data efficiency by plotting RMSE against the fraction of training data used for METR-LA 1-step prediction. In low-data regimes, SPCCNN consistently outperforms GW and is consistently among the best performing models, indicating that it is more data-efficient. This makes SPCCNN a strong candidate for real-world applications where data may be limited.



(a) Parameter counts for METR.



(b) RMSE vs. training data proportion on METR-LA 1-step forecasting.

Figure 5.2: Model complexity (parameter counts) and performance (RMSE) in data-scarce settings.

EFFECT OF PARAMETERIZATION

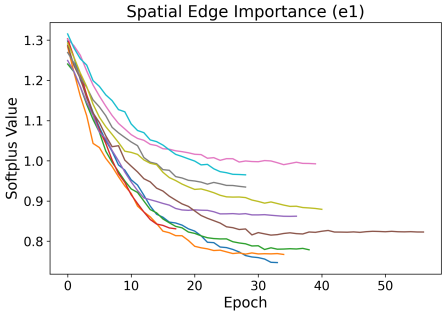
We now discuss the impact of parameterization on the performance and flexibility of the SPCCNN model. Table 5.3 presents the test set performance of the parameterized (P-SPCCNN) and non-parameterized (NP-SPCCNN) variants across 1-step, 2-step, and 3-step forecasting tasks on the METR-LA and Solar datasets. P-SPCCNN consistently outperforms NP-SPCCNN in most settings, indicating that learning to modulate the contributions of different edge and face types improves predictive accuracy. Notably, this improvement is achieved with only a few additional learnable parameters (as discussed in Section 4.7.1), suggesting that even lightweight parameterization of the Hodge Laplacians can substantially improve performance by allowing the model to better adapt to the structural characteristics of each dataset.

To understand the impact of parameterization on flexibility and data-adaptability, we analyze the scalar weights that the model learns for each type of edge and face for the two datasets. We begin by studying the training evolution of these parameters to assess whether they stably converge. Recall that we apply the softplus function (see Equation 4.29) to the raw parameter values in order to ensure non-negativity. Figure 5.3 shows the convergence behavior of the post-softplus values of the spatial edge parameter (e_1) and

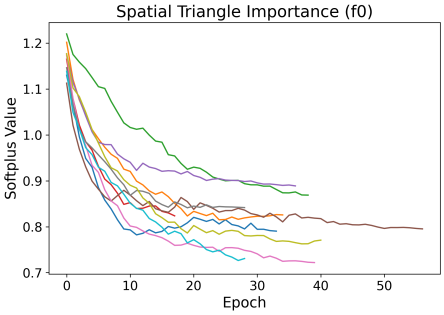
Data	Variant	1-Step		2-Step		3-Step	
		MAE	RMSE	MAE	RMSE	MAE	RMSE
METR-LA	P-SPCCNN	**2.96** ±0.08	**5.64** ±0.04	**3.39** ±0.08	**6.46** ±0.04	3.86 ±0.13	7.16 ±0.08
	NP-SPCCNN	3.04 ±0.11	5.69 ±0.05	3.41 ±0.08	**6.46** ±0.06	**3.79** ±0.07	**7.13** ±0.07
SOLAR	P-SPCCNN	**0.38** ±0.00	**0.78** ±0.00	**0.55** ±0.01	**1.20** ±0.01	**0.70** ±0.01	**1.44** ±0.02
	NP-SPCCNN	0.43 ±0.04	0.82 ±0.03	0.70 ±0.01	1.33 ±0.02	0.84 ±0.03	1.56 ±0.03

Table 5.3: Comparison of parametric (P-SPCCNN) and non-parametric (NP-SPCCNN) variants across forecasting steps on the METR-LA and Solar datasets. Best values are **bolded**.

the spatial triangle parameter (f_0) across multiple random seeds on the METR dataset. Both parameters drift steadily from their initial values and converge to consistent values, demonstrating stable and reliable learning behavior across all seeds. Similar convergence patterns are observed for the remaining edge and face parameters (e_2, e_3, f_1, f_2, f_3, f_4) across both datasets.



(a) Edge parameter (e_1) from multiple random seeds converging stably for METR 1-step forecasting.



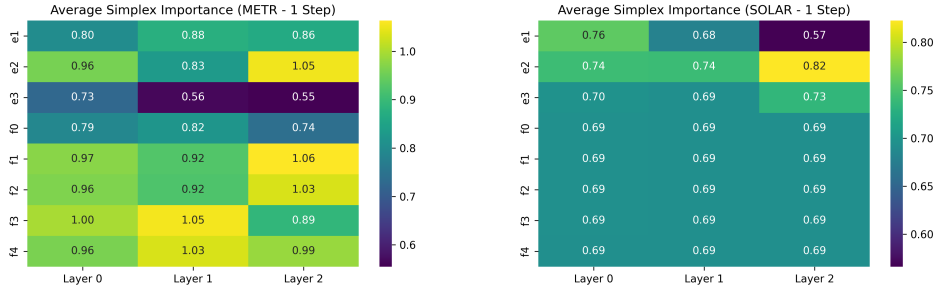
(b) Face parameter (f_0) from multiple random seeds converging stably for METR 1-step forecasting.

Figure 5.3: Training evolution of edge and face parameters on METR.

Next, we compare the final learned parameter values across datasets. Figure 5.4 plots the post-softplus values of edge and face parameters for all SPCCNN layers, averaged across random seeds, for the 1-step forecasting task. The reported values are from the epoch with the lowest validation loss.

For the METR dataset, SPCCNN learns distinct parameter values for different edge and face types, indicating that it can differentiate the relevance of various higher-order relationships. Most face parameters cluster around a post-softplus value of approximately 1, suggesting that all higher-order structures are actively utilized by the model.

In contrast, on the Solar dataset, all face-level parameters converge to the same post-softplus value of approximately 0.69, corresponding to a raw value of approximately zero (see Figure 4.11a). This lack of variation across face types, along with their uniformly low values, indicates that the model finds little benefit in modeling higher-order structures



(a) Learned parameters on METR 1-step forecasting.

(b) Learned parameters on Solar 1-step forecasting.

Figure 5.4: Final learned SPCCNN parameters across all layers.

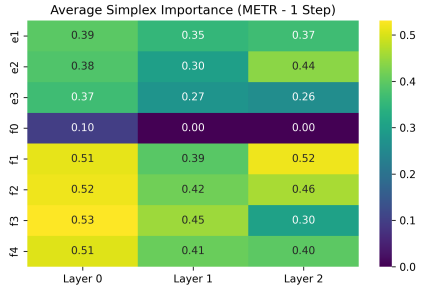
for this dataset. These findings help explain why SPCCNN performs well on METR but not on Solar and highlight the flexibility of the parameterized variant in aligning with the structural properties of a given dataset.

Moreover, since the gradient of the softplus function diminishes near zero (see Figure 4.11b), parameters in this range experience weaker gradient signals and thus have limited incentive to decrease further without explicit regularization. We hypothesize that these suppressed face parameters on Solar could likely be pruned entirely if sparsity was enforced, a hypothesis we explore in the next section.

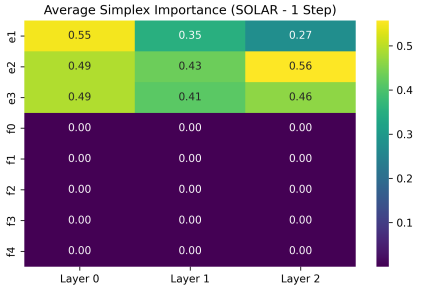
5.2.2. L_1 REGULARIZATION OF EDGE AND FACE PARAMETERS

In our previous experiments, we applied L_2 regularization to all model parameters, including the edge and face parameters in SPCCNN, as described in Section 5.1.4. Although this encourages smaller parameter magnitudes, it does not explicitly promote sparsity. We hypothesize that incorporating an L_1 regularization term for the edge and face parameters will induce sparsity, effectively pruning unnecessary higher-order components from the model and improving the scalability of the SPCCNN model. In particular, based on the findings from the previous section, we expect the face parameters for the Solar dataset to be pruned entirely. Pruning irrelevant types of simplices (parameters that converged to extremely low values) will reduce the number of computations in the convolution operation and hence improve the computational runtime of the SPCCNN during inference.

Figure 5.5 plots the post-softplus values of the edge and face parameters for the SPCCNN model trained with explicit L_1 regularization applied to these parameters. For the Solar dataset, we observe that all face parameters are driven to zero, effectively pruning the corresponding higher-order connections during training. This confirms our hypothesis that higher-order relationships offer little to no utility for this dataset and explains why SPCCNN's performance is comparable to GTCNN for the Solar dataset. In contrast, the METR dataset retains nearly all higher-order relationships, with only the f_0 parameter, associated with purely spatial faces, being pruned. This indicates that higher-order structures add value in modeling the spatio-temporal relationships present in the METR dataset despite explicit regularization.



(a) Learned parameters on METR 1-step forecasting with explicit L_1 regularization.



(b) Learned parameters on Solar 1-step forecasting with explicit L_1 regularization.

Figure 5.5: Final learned SPCCNN parameters across all layers with explicit L_1 regularization on edge and face parameters.

Dataset	Model	MAE	RMSE	Time (ms)
METR-LA	NR	**2.96** ±0.08	**5.64** ±0.04	6.00 ±1.16
	R	3.00 ±0.09	5.66 ±0.05	**4.34** ±1.58
Solar	NR	**0.38** ±0.01	**0.78** ±0.00	6.8 ±1.38
	R	0.37 ±0.00	**0.78** ±0.00	**3.9** ±1.13

Table 5.4: Comparison of non-regularized (NR) and regularized (R) SPCCNN variants for 1-step forecasting on the METR-LA and Solar datasets. 'Time' reports average clock time (in milliseconds) for a forward pass during testing. Best values are **bolded**.

Table 5.4 compares the 1-step forecasting performance and average computation time for a single forward pass on the test set, averaged across all random seeds, for both the regularized (R) and non-regularized (NR) versions of SPCCNN. We observe that pruning has a negligible impact on forecasting accuracy, as the performance remains comparable to that of the non-regularized model. However, the computation time is nearly halved across both datasets. Pruning off unnecessary higher-order connections increases the sparsity of the Hodge Laplacians and thereby reduces the number of computations in the higher-order filtering layers in the SPCCNN. This demonstrates that explicit L_1 regularization can significantly enhance the scalability of SPCCNN by eliminating unnecessary higher-order connections during inference, resulting in notable computational savings without compromising predictive performance.

5.2.3. SPECTRAL ANALYSIS OF HODGE LAPLACIANS OF THE SPC

Having presented a detailed empirical evaluation of the SPCCNN model on the downstream forecasting task, we now examine the spectral properties of the 1-dimensional Hodge Laplacian $\mathbf{L}_{1\diamond}$ of the SPC for both the datasets. We first obtain the Hodge decomposition of the SPC by computing the eigendecomposition of $\mathbf{L}_{1\diamond}$ and its lower and upper components. The gradient subspace of the SPC is spanned by the eigenvectors of

the lower Laplacian $\mathbf{L}_{1\diamond,d}$ associated with positive eigenvalues. Similarly, the curl subspace of the SPC is spanned by eigenvectors of the upper Laplacian $\mathbf{L}_{1\diamond,u}$ associated with positive eigenvalues. The harmonic subspace of the SPC corresponds to the null space of $\mathbf{L}_{1\diamond}$. We denote the eigenvectors of the gradient, curl, and harmonic subspaces by $\mathbf{U}_G$, $\mathbf{U}_C$, and $\mathbf{U}_H$, with corresponding eigenvalues Λ_G, Λ_C, and Λ_H.

We begin by examining the spectral properties and eigenvalue distributions of each Hodge subspace. We then analyze the smoothest and most oscillatory gradient and curl basis vectors to understand the structural characteristics they capture. Next, we investigate spectral behavior by edge type, highlighting differences across spatial, temporal, and spatio-temporal edge flows. Finally, we project edge signals onto each Hodge subspace and quantify the proportion of total energy captured by each subspace.

SPECTRAL PROPERTIES

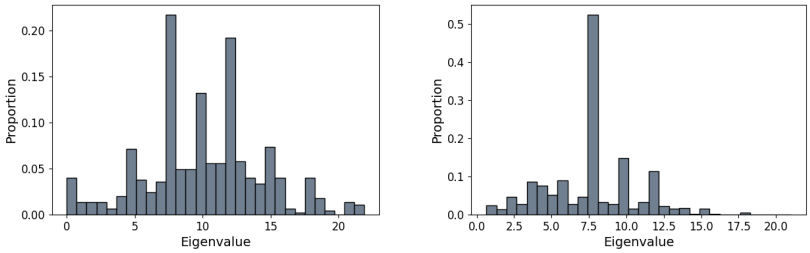
Analyzing the spectral properties provides a high-level overview of the behavior of edge signals in the spectrum. Table 5.5 summarizes the dimension, spectral entropy, and spectral gap of each Hodge subspace of $\mathbf{L}_{1\diamond}$. The spectral entropy quantifies the energy dispersion across the spectrum of a subspace, where higher values indicate larger spread in the distribution of eigenvalues, while lower values suggest concentration in a fewer modes. The spectral gap refers to the value of the smallest non-zero eigenvalue, capturing how well-separated the subspace is from the null space. These metrics help us assess the richness and stability of the topological signals in each Hodge component.

Metric	**Gradient**	**Curl**	**Harmonic**
Solar			
Dimension	404	1670	13
Spectral Entropy	5.90	7.31	–
Spectral Gap	0.0597	0.5858	–
METR			
Dimension	613	2488	1
Spectral Entropy	6.31	7.73	–
Spectral Gap	0.0078	0.6342	–

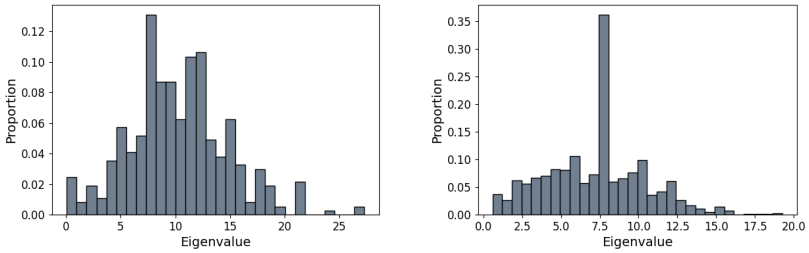
Table 5.5: Spectral properties of the $L_{1\diamond}$ Hodge Laplacian for METR and Solar datasets.

The $\mathbf{L}_{1\diamond}$ Laplacian for Solar and METR exhibit significantly larger curl subspaces (1670 and 2488 dimensions respectively) compared to their gradient subspaces (404 and 613 dimensions respectively). Since each triangle introduces a local circulation that could potentially be captured by its own eigenmode, the abundance of triangles in the SPC naturally results in a higher-dimensional curl subspace.

The Curl subspaces also exhibit higher spectral entropy and larger spectral gaps, indicating more diverse and localized circulation patterns. In contrast, the gradient subspaces have low spectral gaps, suggesting the presence of smooth, slowly-varying global



(a) Eigenvalue distribution of the gradient (left) and curl (right) subspaces for the METR dataset.



(b) Eigenvalue distribution of the gradient (left) and curl (right) subspaces for the Solar dataset.

Figure 5.6: Histogram of eigenvalues of the $L_{1\diamond}$ Hodge Laplacian.

flows. The full eigenvalue distribution, shown in Figure 5.6, visually confirms that curl modes are more widely spread than gradient modes.

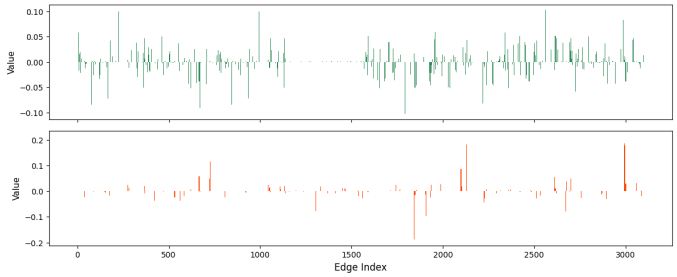
Finally, the harmonic component is negligible for both datasets due to the lack of topological holes in the SPC. Topological holes correspond to cycles that are not bounded by filled triangles and most cycles in the SPC are filled in by 2-simplices. Since harmonic modes are negligible, we omit their analysis in the remaining section.

ANALYZING EXTREME MODES OF HODGE SUBSPACES

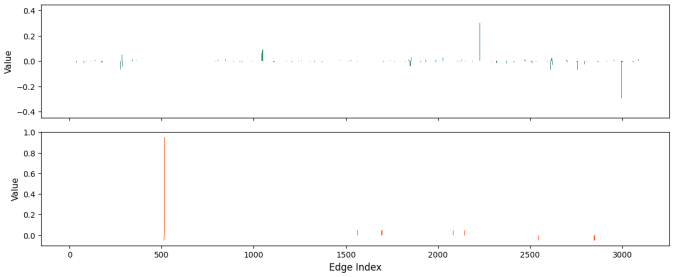
We now analyze the extreme modes of the gradient and curl subspaces - the eigenvectors associated with the lowest and highest eigenvalues. This will provide insights into the smoothest and most oscillatory patterns captured by the $\mathbf{L}_{1\diamond}$ Hodge Laplacian. Figures 5.7 and 5.8 visualizes the eigenvectors corresponding to the lowest and highest eigenvalues in the gradient and curl subspaces for the METR and Solar datasets.

As seen in the top rows (green) of Figures of both the gradient and curl subspaces for both datasets, the low-eigenvalue eigenvectors exhibit globally smooth structure, with values distributed across many edges. In contrast, the high-eigenvalue modes in the bottom rows (orange) are sharp and localized, with energy concentrated on a small subset of edges, appearing as spiky patterns. This behavior is consistent with the expected characteristics of low- and high-eigenvalue spectral components.

Across both datasets, the curl eigenvectors appear consistently sparser and exhibit larger magnitudes than their gradient counterparts, reflecting the inherently localized nature of curl flows which are often associated with localized circulations. In contrast, gradient modes display smoother variations with less pronounced localization, even at

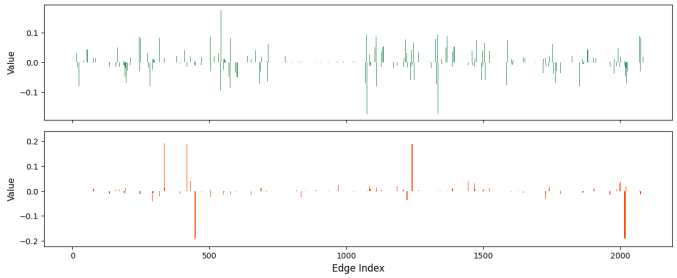


(a) Lowest (top) and highest (bottom) gradient modes for METR.

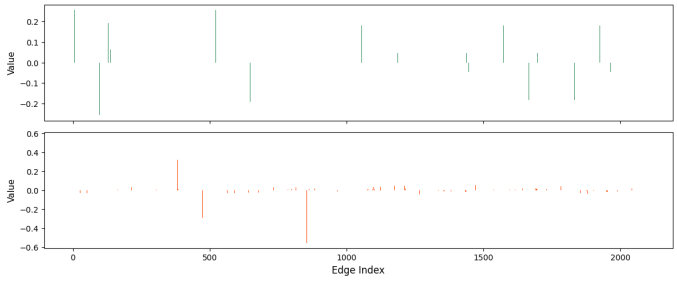


(b) Lowest (top) and highest (bottom) curl modes for METR.

Figure 5.7: Lowest and highest eigenvalue gradient and curl modes for the METR dataset.



(a) Lowest (top) and highest (bottom) gradient modes for Solar.



(b) Lowest (top) and highest (bottom) curl modes for Solar.

Figure 5.8: Lowest and highest eigenvalue gradient and curl modes for the Solar dataset.

the higher eigenvalue. These visual patterns align with earlier spectral statistics, where the curl subspace showed higher entropy and a wider spectral spread.

Notably, we observe a large gap in the low-eigenvalue (green) portions of both gradient and curl spectra (between Edge Index 1200 and 1600 in the top rows of 5.7 and Edge Index 750 to 1100 in the top rows of 5.8). These edge indices actually correspond to the temporal edges and seem to exhibit near-zero coefficients in the low eigenvalue modes. This suggests that temporal edges are weakly represented in the smoothest modes of both subspaces.

EDGE-TYPE SPECIFIC ANALYSIS

The analysis in the previous section suggested that temporal edges contribute minimally to the smoothest frequency modes of both the gradient and curl subspaces. To investigate this further, we visualize the energy distribution across individual eigenmodes, ordered from low to high frequency, for each edge type within both subspaces. Since the eigenvectors have dimensionality equal to the number of edges, we can extract specific components of the eigenvectors that correspond to the different edge types using their edge indices and compute their energy using the ℓ_2 norm. The plots are presented in Figure 5.9 for the METR-LA dataset and the Solar dataset is omitted due to exhibiting similar trends.

In both gradient and curl subspaces, temporal edges contribute negligible energy to the lowest-frequency modes. Since these modes typically encode smooth, global patterns, this indicates that temporal edges are misaligned with global flows and more representative of abrupt transitions. This reflects their structural role in the SPC: unlike spatial and spatio-temporal edges that link distinct nodes and encode geometric relationships, temporal edges connect a node to itself across time, acting as stitching links that preserve temporal continuity rather than directional flow.

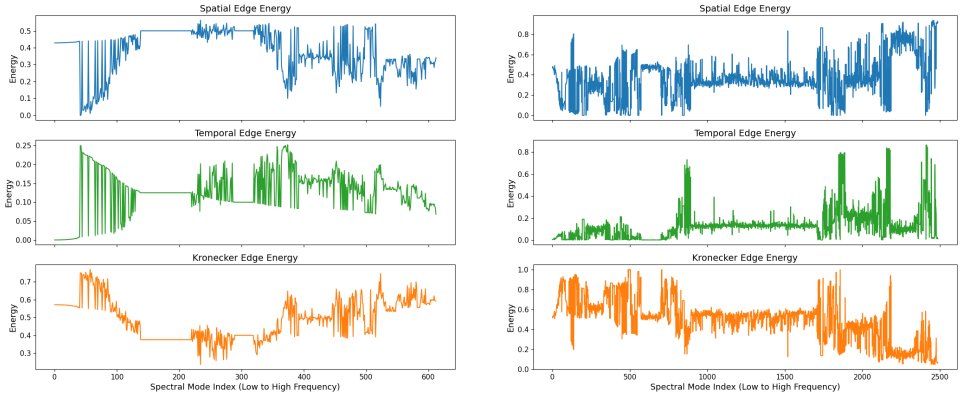
Across the full gradient spectrum, temporal edges contribute substantially less energy compared to other edge types. However, in the curl subspace, their energy is more evenly distributed and relatively higher. This suggests that while temporal edges are weakly aligned with smooth gradients, they participate more actively in localized, triangle-induced circulations. Thus, in the SPC, temporal edges primarily serve as anchors for forming spatio-temporal triangles rather than as carriers of smooth signal propagation.

Overall, the analysis suggests that temporal edges are structurally less aligned with smooth, global signal propagation and play a greater role in supporting localized, circulatory flows. In contrast, spatial and spatio-temporal edges contribute more prominently to the smoothest modes, highlighting their structural alignment with coherent, global interactions across the SPC.

ENERGY-BASED SIGNAL ANALYSIS

In the final section, we now shift from a purely structural analysis, based on the properties of the Hodge Laplacian and its eigenspaces, to a functional perspective by analyzing the energy of edge signals projected onto the gradient, curl, and harmonic subspaces.

Recall that our datasets do not inherently contain edge or higher-order signals, and edge features are instead constructed by lifting node-level signals. In this analysis, we focus on the encoded edge signals produced by a trained SPCCNN's edge encoder that take the averaged (lifted) node signals and pass them through the encoder to produce a



(a) Gradient energy by edge type

(b) Curl energy by edge type

Figure 5.9: Energy distribution across edge types within the eigenbasis of the gradient and curl subspaces.

multi-channel representation (16 channels in our experiments) for each edge. We compute the mean across the channel dimension to obtain a single scalar value per edge.

To ensure consistency with the model inputs, we use normalized raw signals provided to the encoder, applying the exact normalization procedure used during training (see Section 5.1.2). For statistical robustness, we randomly sample 1000 signal snapshots from the dataset. Each snapshot corresponds to an edge-level signal on the SPC. We then project these signals onto the gradient, curl, and harmonic subspaces of the $L_{1\diamond}$ Hodge Laplacian and calculate the average projection energy across these samples for each Hodge subspace.

Formally, let $\mathbf{X} \in \mathbb{R}^{N^1 \times M}$ represent the matrix of sampled (and encoded) edge signals where $M = 1000$ and N^1 is the number of edges in the SPC. The projected components are then given by:

$$\mathbf{X}_G = \mathbf{U}_G \mathbf{U}_G^\top \mathbf{X}, \quad \mathbf{X}_C = \mathbf{U}_C \mathbf{U}_C^\top \mathbf{X}, \quad \mathbf{X}_H = \mathbf{U}_H \mathbf{U}_H^\top \mathbf{X}$$

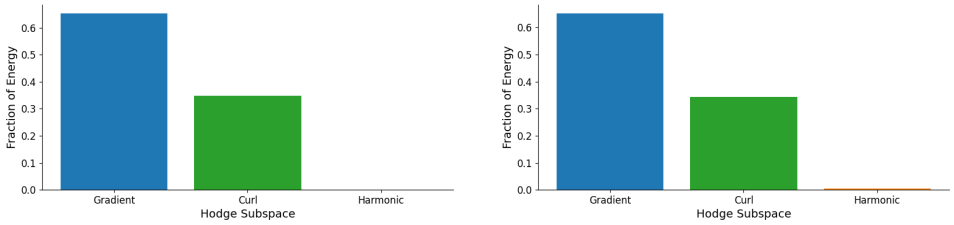
and the corresponding average energies are given by

$$E_G = \frac{1}{M} \|\mathbf{X}_G\|_F^2, \quad E_C = \frac{1}{M} \|\mathbf{X}_C\|_F^2, \quad E_H = \frac{1}{M} \|\mathbf{X}_H\|_F^2$$

Figure 5.10 shows the average distribution of signal energy across the gradient, curl, and harmonic subspaces for both datasets. The majority of energy in both METR and Solar is concentrated in the gradient subspace, indicating that the encoded edge signals primarily follow node-induced flows. This is expected, as the edge features are derived from node-level signals and thus naturally inherit gradient-like structure.

Notably, approximately one-third of the energy resides in the curl subspace in both datasets. Although raw lifted edge signals would lie entirely in the gradient subspace, the encoded signals used here are produced by a learnable edge encoder. This encoder,

being a multi-layer MLP, can project the initially smooth edge signals into more complex representations that include local circulatory patterns captured by the curl space. Moreover, the use of normalization prior to encoding may also redistribute energy across the spectral domain, further explaining this shift.



(a) METR **(b)** Solar

Figure 5.10: Fraction of total encoded signal energy projected onto each Hodge subspace.

Finally, the harmonic component holds negligible energy in both datasets due to its small dimensionality and the absence of strong topological holes in the underlying structures.

In summary, through this spectral analysis, we conclude that the SPC induces a rich spectral structure with a dominant curl subspace arising from abundant triangle-based interactions. Temporal edges contribute minimally to smooth modes yet play a greater role in localized curl patterns, and encoded signals align strongly with gradient modes but also carry notable energy in the curl subspace. Overall, the spectral analysis demonstrates that the SPC provides a mathematically principled and expressive framework for modeling complex spatio-temporal dynamics.

5.2.4. CASE STUDY: SPATIO-TEMPORAL IMPUTATION OF THE NEW DELHI AIR POLLUTION MONITORING NETWORK

Urban air pollution poses significant environmental and health challenges in densely populated cities like New Delhi, where vehicle emissions and industrial activities contribute to elevated levels of harmful pollutants such as Particulate Matter (PM2.5). Effective air quality monitoring systems are thus crucial for public health protection, yet they face significant challenges due to sensor failures and incomplete measurements.

Real-world air quality data often exhibits substantial missing values which makes reliable pollution assessment extremely difficult [77]. For instance, the raw dataset used in this study, as detailed later in this section, contains over 60% missing data. To address this challenge, robust imputation strategies are essential for recovering missing observations and maintaining the integrity of environmental monitoring systems. In this final section, we present a case study applying our proposed SPCCNN model to spatio-temporal imputation of air pollution data. We first describe the imputation task and problem formulation, then detail our dataset construction methodology, and finally discuss the experimental results of this case study.

SPATIO-TEMPORAL IMPUTATION

Spatio-temporal imputation involves recovering missing values in multivariate time series by leveraging both temporal dynamics and spatial correlations. Unlike traditional time series imputation that treats each variable independently, spatio-temporal imputation exploits the underlying relational structure among variables to improve reconstruction accuracy. The spatial correlations help infer missing values based on measurements from neighboring sensors, while temporal dynamics can guide prediction of missing observations using the past observations of a particular variable.

Similar to the forecasting setting, we construct a dataset $\mathscr{D} = \{(X^{(i)}, M^{(i)})\}_{i=1}^{|\mathscr{D}|}$, where each $X^{(i)} \in \mathbb{R}^{T\times N\times F}$ is a sample extracted from the full time series using a sliding window of length T. Additionally, we create a binary mask $M^{(i)} \in \{0,1\}^{T\times N\times F}$ that randomly hides a fixed percentage of entries in $X^{(i)}$. To create the model input, masked entries are replaced with their last observed values using forward-fill imputation, resulting in $\tilde{X}^{(i)}$, which serves as the input to the model. The objective is to reconstruct the complete original observations $X^{(i)}$ from this partially corrupted data $\tilde{X}^{(i)}$. The model prediction is denoted by $\hat{X}^{(i)} \in \mathbb{R}^{T\times N\times F}$, and the loss is computed only over the originally missing entries:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{|\mathscr{D}|} \sum_{i=1}^{|\mathscr{D}|} \ell\left(\hat{X}^{(i)} \odot (1 - M^{(i)}),\ X^{(i)} \odot (1 - M^{(i)})\right), \tag{5.5}$$

where $\odot$ denotes element-wise multiplication.

DELHI AIR POLLUTION DATASET

We construct the air pollution imputation dataset for this case study using the raw data provided by the Central Pollution Control Board (CPCB) of India [17]. The original dataset comprises hourly measurements of 34 air quality related metrics recorded at sensors across 453 cities in India between 2010 and 2023. For our experiments, we focus exclusively on the PM2.5 variable, measured in $\mu g/m^3$, due to its critical relevance to air quality monitoring. Moreover, we use the sensor data for the 38 monitoring stations located within New Delhi and further discard two stations that exhibit inconsistent PM2.5 measurements, resulting in a final set of 36 sensors. The locations of these sensors are detailed in the appendix in Table B.1. The spatial graph is built using a 3-nearest neighbors (3-NN), which results in a fully-connected graph, illustrated in Figure B.1 in the appendix.

We filter the time series data for the selected sensors to keep measurements from January 2021 to March 2023, yielding 37,224 hourly PM2.5 measurements per station. As discussed above, the raw dataset contains substantial missing values that are irregularly distributed across time and sensors, with some periods having much more missing data than others. To create a controlled imputation dataset, we create segments of data using a sliding window approach with windows of size 5, and only keep complete windows that have no missing values. In each of the temporal windows, we then create artificial missing data by randomly removing exactly 20% of the PM2.5 readings.

We use this approach for two important reasons. First, we need ground truth values to train and evaluate our model and we cannot use the original missing values as

the imputed observations because we do not have access to the ground truth for values. Second, the original missing data patterns are highly irregular, which would make it difficult to train models consistently. Through our approach, we create a standardized dataset where we know the true values for all observations, allowing us to properly train and evaluate our imputation method while simulating a hypothetical missing data scenario.

Once we create the final imputed dataset using the process described above, we preprocess it using the same procedure used for the METR and Solar datasets, as discussed in Section 5.1.2.

RESULTS

For the experiments in the case study, we follow a similar training and evaluation setting as our forecasting experiments, as described in Section 5.1.4. Table 5.6 presents the imputation results for the SPCCNN model and selected spatio-temporal baselines. The final architectures after hyperparameter tuning for each of the models are provided in A.2.1.

Model	MAE	RMSE
STGCN	**14.5** ±1.45	22.05 ±2.21
GW	14.59 ±1.69	**21.87** ±2.33
GTCNN	16.67 ±4.66	24.61 ±6.0
SNN-RNN (ours)	17.48 ±1.12	25.31 ±1.27
SPCCNN (ours)	19.23 ±1.42	27.88 ±2.21

Table 5.6: Imputation results on the Delhi air pollution dataset. Best values are **bolded**.

We observe that Graph WaveNet (GW) and STGCN achieve the best performance with very similar results, while GTCNN shows moderate performance but with much higher variance across runs. Notably, both proposed models that operate on higher-dimensional topological spaces, SPCCNN and SNN-RNN, perform significantly worse than all baseline methods, ranking as the two worst-performing approaches on the imputation task.

We hypothesize that the poor performance of our proposed models may be attributed to fundamental challenges that likely arise when applying signal lifting to incomplete data. When node signals contain forward-filled values that may be stale or inaccurate, the lifting process could compound these errors by deriving edge signals from already corrupted inputs. This noise may then propagate through the higher-order structures during simplicial convolutions, potentially amplifying uncertainty rather than extracting useful topological relationships. Moreover, the multi-node dependencies that SPCCNN relies on could become unreliable when multiple nodes within the same triangular relationship contain imputed values, possibly further degrading the quality of higher-order representations and hindering effective reconstruction of missing observations.

Therefore, applying SPCCNN for imputation tasks reveals a critical limitation of our framework: its effectiveness is fundamentally constrained when operating on incom-

plete data. While the SPCCNN architecture excelled at capturing complex multi-node relationships in the forecasting task where data was complete, its dependency on signal lifting makes it inherently fragile to the noisy or missing data encountered in imputation settings. We hypothesize that SPCCNN's accuracy would improve on imputation tasks if the dataset contained genuine edge flows that didn't require lifting from potentially corrupted node signals.

5.3. DISCUSSION

Effectively modeling multivariate time series data requires capturing not only individual temporal trends but also the complex, often latent, interactions between variables across space and time. These relationships frequently extend beyond simple pairwise dependencies, and often involve higher-order spatio-temporal relationships among groups of variables. In this work, we hypothesized that such multi-way interactions could be modeled effectively using a simplicial structure in the product space. To this end, we proposed the parameterized SPCCNN architecture that learns to weigh different topological relationships, making the model more adaptive and flexible to the underlying data.

Our evaluation of SPCCNN on multivariate node-level forecasting across two real-world datasets demonstrated that the proposed architecture performed comparably to state-of-the-art baselines and outperformed approaches that lacked relational structure. Notably, SPCCNN exhibited greater parameter and data efficiency, making it promising for limited-data settings. The architecture performed particularly well on the METR-LA dataset but showed weaker performance on the Solar dataset.

To understand this performance discrepancy, we conducted an analysis of the learned parameter values. Parameterization consistently improved downstream results across both datasets and all forecasting horizons, indicating the SPCCNN model's successful adaptation to the relationships present in each dataset. Interestingly, the Solar dataset exhibited convergence to low, nearly uniform parameter values, while the METR-LA dataset showed higher and more diverse parameter magnitudes. This observation suggested that topological structures were less relevant for the Solar dataset, leading us to hypothesize that sparsity constraints would prune these connections.

To test this hypothesis, we applied L1-regularization to the edge and face parameters. The results confirmed our assumption: all face parameters were eliminated for the Solar dataset, while only spatial face parameters were removed for the METR-LA dataset. Importantly, predictive accuracy remained largely unaffected, and inference speed improved due to increased sparsity. This finding highlighted how explicit regularization promoted scalability by pruning redundant topological structures.

Following the empirical evaluation, we conducted a spectral analysis of edge flow signals on the SPC. Our findings revealed that the SPC induced a rich spectral structure dominated by curl subspaces. Temporal edges contributed minimally to smooth eigenmodes yet participated more actively in localized circulation patterns. Additionally, encoded signals aligned primarily with gradient subspaces while retaining significant energy in curl components. This analysis confirmed that the SPC provided an expressive framework for capturing complex temporal dynamics.

Finally, we extended our evaluation to a case study applying the SPCCNN model for spatio-temporal imputation on the New Delhi air pollution monitoring network. This

case study revealed a fundamental limitation of our framework: the challenge of lifting signals from noisy or corrupted data in imputation tasks. Our findings indicated that SPCCNN is unsuitable for imputation settings where input data quality is compromised.

In conclusion, our empirical evaluation demonstrated that the SPCCNN framework provided an effective approach for modeling multivariate time series by capturing multi-node dependencies in the product space, particularly for tasks where data is complete and not noisy. The parameterization mechanism improved predictive accuracy while enhancing model flexibility and scalability through data-adaptive learning. Our analysis showed that our core assumption underlying SPCCNN, that multivarite time series data exhibits simplicial structure in product spaces, served as a powerful inductive bias enabling more effective spatio-temporal modeling.

6

Conclusion and Future Work

In this chapter, we conclude this thesis and discuss directions for future research. We begin by revisiting the research questions discussed in Chapter 1 and providing answers to them based on the results we obtained. We then discuss limitations of this work and propose future research directions that address these limitations.

6.1. Answers to Research Questions

(RQ) *How can we model multivariate time-series data by leveraging higher-order spatio-temporal connections through a simplicial structure in the product space?*

We addressed this central research question by proposing the SPC as a general framework for modeling multivariate time-series data using a simplicial structure in the product space. We showed that this approach enabled the modeling of higher-order spatio-temporal relationships beyond pairwise interactions, while retaining a well-defined simplicial structure. The SPCCNN architecture, built on this framework, enabled flexible adaptation to different datasets and effectively captured complex, multi-node spatio-temporal dependencies in a data-driven manner.

(RQ1) *How can we effectively parameterize the simplicial product space to learn higher-order spatio-temporal relationships from data?*

This question was addressed in Chapter 4 through our construction of the parameterized SPC. Our analysis revealed three distinct edge types (spatial, temporal, and spatio-temporal) and five face types (one spatial and four spatio-temporal), each representing unique higher-order relationships in the product space. We demonstrated a systematic procedure for building the SPC with parameterized incidence matrices, assigning a dedicated learnable weight to each simplex type. This parameterization enabled the SPCCNN to flexibly and explicitly modulate the influence of different higher-order spatio-temporal interactions during learning, confirming the viability and effectiveness of our parameterization approach.

(RQ2) *To what extent does the parameterized SPCCNN improve predictive performance, data-adaptability, and scalability in spatio-temporal learning?*

We addressed this question in Section 5.2 through a comprehensive empirical evaluation. Our results demonstrated that the parameterized SPCCNN achieved comparable predictive performance to state-of-the-art spatio-temporal models in forecasting tasks, while showing limited effectiveness in the imputation setting where the data is incomplete. For forecasting, SPCCNN consistently outperformed other product space models such as GTCNN and CELL. The parameterized variant significantly outperformed its non-parameterized counterpart, confirming that parameterization improved predictive performance. Furthermore, analyses on the Solar and METR-LA datasets revealed that parameterization enabled the model to adaptively tune the influence of different higher-order relationships based on dataset-specific characteristics, demonstrating data-adaptability. Finally, inducing sparsity on the parameters allowed the model to prune irrelevant connections, leading to faster inference while maintaining accuracy, thus demonstrating improved scalability. We therefore concluded that parameterization enhanced the performance, flexibility, and scalability of the SPCCNN, particularly when the input data is complete and not noisy.

(RQ3) *What are the spectral properties of the SPC, and how do they inform our understanding of signal propagation across the spatio-temporal complex?*

We explored this question in Section 5.2.3 through a spectral analysis of edge flows on the 1-Hodge Laplacian $\mathbf{L}_{1\diamond}$ of the SPC. The spectral properties revealed that temporal edges functioned primarily as structural anchors rather than signal pathways, evidenced by their sparse representation in smooth frequency modes and under-representation in the gradient subspace. Despite the SPC's large curl subspace arising from its triangle-rich structure, signal energy concentrated predominantly in the gradient subspace. This indicated that while the SPC contained abundant triangular structures creating curl components, actual signal propagation occurred primarily through gradient flows, with temporal edges serving as structural backbone for spatio-temporal connectivity. Our spectral analysis thus provided valuable insights into the fundamental signal propagation mechanisms within the spatio-temporal complex.

6.2. FUTURE WORK

To conclude this thesis, we identify the limitations of our work and outline several promising directions for future research that address these limitations.

EXPERIMENTING ON EDGE FLOW DATA

Our current approach does not leverage genuine edge flow data, despite simplicial complexes being naturally designed to operate on higher-order structures such as edges and faces. To address this limitation, future work could focus on introducing genuine edge flow signals. Since our spectral analysis showed that the curl subspace structurally dominates the SPC, genuine edge flow signals, which can contain curl components that node-lifted signals cannot directly capture, could potentially better utilize this dominant subspace and improve model performance. Furthermore, considering the limitations of lifting noisy node features onto edges discussed in Section 5.2.4, genuine edge flows might

mitigate noise propagation from corrupted data, thereby improving SPCCNN's imputation performance. One promising domain is water distribution networks, where real edge flow time-series data is available [30]. It is worth noting, however, that while such datasets provide genuine edge flow signals for spatial edges, signals on temporal and spatio-temporal edges would still need to be lifted.

VECTOR-BASED PARAMETERIZATION

Our current parameterization scheme is overly restrictive, assigning only a single scalar weight to each simplex type rather than allowing individual control over specific simplices within each type. A more expressive alternative would be to assign a vector of learnable parameters to each simplex type, where each element in the vector modulates the contribution of an individual higher-order simplex within that type. This would allow the model to differentially weigh specific spatial or spatio-temporal interactions, rather than treating all simplices of a given type uniformly. The advantage of maintaining separate vectors for each simplex type is that it enables distinct regularization schemes for different types of higher-order relationships while still allowing individual control within each type. This could provide greater modeling flexibility and facilitate the integration of domain-informed structural priors.

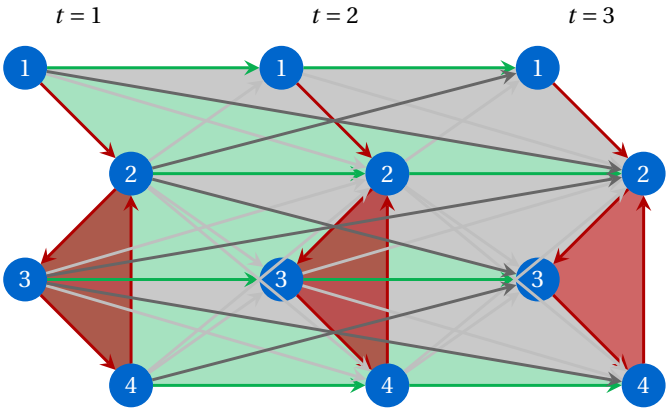


Figure 6.1: An example of a time-lagged SPC where spatio-temporal triangles (in green and gray) are formed with a time-lag of two time steps instead of consecutive steps. Red triangles represent spatial triangles as before. Signals and orientations are omitted for clarity, and not all possible triangles are shown.

SCALABLE TRAINING STRATEGIES

The current training procedure does not scale effectively to large graphs due to substantial memory and computational overhead from constructing the full set of simplices and the inability to precompute parameterized Hodge Laplacians. To mitigate this, future work could explore progressive construction strategies, wherein the SPC is built incrementally during training by selectively including only the most informative simplex types based on intermediate model feedback or gradient signals. The large size of the $\mathbf{B}_{2\diamond}$ matrix, particularly for large graphs or long temporal windows, could be reduced

through such selective construction approaches. Another promising direction is to decouple the simplex-type parameter update frequency from that of the rest of the network. For instance, updating the Hodge Laplacian parameters once per epoch or every few epochs would allow pre-computing the shifted Laplacians for a fixed period during training, significantly reducing the overhead of repeated simplicial shifting computations and improving training efficiency.

TIME-LAGGED TRIANGLE FORMATION IN THE SPC

The current SPC framework does not scale effectively to longer temporal windows due to the rapid growth in the number of triangles formed between consecutive time steps. A promising solution is the construction of a time-lagged SPC where triangles span multiple time steps rather than consecutive ones, as illustrated in Figure 6.1. While temporal and spatio-temporal edges can still be maintained between adjacent time steps to preserve local relationships, triangles would be formed between time steps that are two or more steps apart. This approach maintains the same number of triangles while distributing them across a longer temporal window, effectively enabling extended time horizons without the computational burden of exponentially increasing triangle counts. Such time-lagged connections would capture longer-range higher-order dependencies while keeping structural complexity manageable.

BIBLIOGRAPHY

[1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2024.

[2] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. 2018. arXiv: 1803.01271 [cs.LG]. URL: https://arxiv.org/abs/1803.01271.

[4] Sergio Barbarossa and Stefania Sardellitti. "Topological Signal Processing Over Simplicial Complexes". In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 2992–3007. ISSN: 1941-0476. DOI: 10.1109/tsp.2020.2981920. URL: http://dx.doi.org/10.1109/TSP.2020.2981920.

[5] Sergio Barbarossa, Stefania Sardellitti, and Elena Ceci. "LEARNING FROM SIGNALS DEFINED OVER SIMPLICIAL COMPLEXES". In: *2018 IEEE Data Science Workshop (DSW)*. 2018, pp. 51–55. DOI: 10.1109/DSW.2018.8439885.

[6] Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[7] Claudio Battiloro et al. *From Latent Graph to Latent Topology Inference: Differentiable Cell Complex Module*. 2023. arXiv: 2305.16174 [cs.LG]. URL: https://arxiv.org/abs/2305.16174.

[8] Claudio Battiloro et al. *Generalized Simplicial Attention Neural Networks*. 2024. arXiv: 2309.02138 [cs.LG]. URL: https://arxiv.org/abs/2309.02138.

[9] Federico Battiston et al. "Networks beyond pairwise interactions: Structure and dynamics". In: *Physics Reports* 874 (Aug. 2020), pp. 1–92. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2020.05.004. URL: http://dx.doi.org/10.1016/j.physrep.2020.05.004.

[10] Cristian Bodnar et al. *Weisfeiler and Lehman Go Cellular: CW Networks*. 2022. arXiv: 2106.12575 [cs.LG]. URL: https://arxiv.org/abs/2106.12575.

[11] Cristian Bodnar et al. *Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks*. 2021. arXiv: 2103.03212 [cs.LG]. URL: https://arxiv.org/abs/2103.03212.

[12] Anastasia Borovykh, Sander Bohte, and Cornelis W. Oosterlee. *Conditional Time Series Forecasting with Convolutional Neural Networks*. 2018. arXiv: 1703.04691 [stat.ML]. URL: https://arxiv.org/abs/1703.04691.

[13] G. E. Box et al. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[14] George E. P. Box et al. "Time Series Analysis: Forecasting and Control". In: *John Wiley & Sons* (2015).

[15] Eric Bunch et al. *Simplicial 2-Complex Convolutional Neural Nets*. 2020. arXiv: 2012.06010 [math.AT]. URL: https://arxiv.org/abs/2012.06010.

[16] Defu Cao et al. *Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting*. 2021. arXiv: 2103.07719 [cs.LG]. URL: https://arxiv.org/abs/2103.07719.

[17] Central Pollution Control Board. *Air Quality Monitoring Data*. https://cpcb.nic.in. Accessed: 2025-07-29. 2023.

[18] Ines Chami et al. "Hyperbolic Graph Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 32. 2019, pp. 4868–4879.

[19] Shiyu Chang et al. "Dilated recurrent neural networks". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*. 2017, pp. 76–86.

[20] Yao-Yuan Chang et al. *A Memory-Network Based Solution for Multivariate Time-Series Forecasting*. 2018. arXiv: 1809.02105 [cs.LG]. URL: https://arxiv.org/abs/1809.02105.

[21] Kyunghyun Cho et al. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* (2014).

[22] Fan R. K. Chung. *Spectral Graph Theory*. Vol. 92. CBMS Regional Conference Series in Mathematics. Providence, RI: American Mathematical Society, 1997.

[23] Andrea Cini et al. "Graph Deep Learning for Time Series Forecasting". In: *ACM Computing Surveys* 57.12 (July 2025), pp. 1–34. ISSN: 1557-7341. DOI: 10.1145/3742784. URL: http://dx.doi.org/10.1145/3742784.

[24] Yann N. Dauphin et al. *Language Modeling with Gated Convolutional Networks*. 2017. arXiv: 1612.08083 [cs.CL]. URL: https://arxiv.org/abs/1612.08083.

[25] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. 2017. arXiv: 1606.09375 [cs.LG]. URL: https://arxiv.org/abs/1606.09375.

[26] Stefania Ebli, Michaël Defferrard, and Gard Spreemann. *Simplicial Neural Networks*. 2020. arXiv: 2010.03633 [cs.LG]. URL: https://arxiv.org/abs/2010.03633.

[27] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179–211.

[28] Chengzhong Fan et al. "Multi-horizon time series forecasting with temporal attention learning". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2019, pp. 2527–2535.

[29] Fernando Gama et al. “Graphs, Convolutions, and Neural Networks: From Graph Filters to Graph Neural Networks”. In: *IEEE Signal Processing Magazine* 37.6 (Nov. 2020), pp. 128–138. ISSN: 1558-0792. DOI: 10.1109/msp.2020.3016143. URL: http://dx.doi.org/10.1109/MSP.2020.3016143.

[30] Alexander Garzón et al. “Transferable and data efficient metamodeling of storm water system nodal depths using auto-regressive graph neural networks”. In: *Water Research* 266 (2024), p. 122396. ISSN: 0043-1354. DOI: https://doi.org/10.1016/j.watres.2024.122396. URL: https://www.sciencedirect.com/science/article/pii/S0043135424012958.

[31] Lorenzo Giusti. *Topological Neural Networks: Mitigating the Bottlenecks of Graph Neural Networks via Higher-Order Interactions*. 2024. arXiv: 2402.06908 [cs.LG]. URL: https://arxiv.org/abs/2402.06908.

[32] Lorenzo Giusti et al. *Cell Attention Networks*. 2022. arXiv: 2209.08179 [cs.LG]. URL: https://arxiv.org/abs/2209.08179.

[33] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Liò. *Simplicial Attention Networks*. 2022. arXiv: 2204.09455 [cs.LG]. URL: https://arxiv.org/abs/2204.09455.

[34] Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi. *Cell Complex Neural Networks*. 2021. arXiv: 2010.00743 [cs.LG]. URL: https://arxiv.org/abs/2010.00743.

[35] Mustafa Hajij et al. *Topological Deep Learning: Going Beyond Graph Data*. 2023. arXiv: 2206.00606 [cs.LG]. URL: https://arxiv.org/abs/2206.00606.

[36] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI]. URL: https://arxiv.org/abs/1706.02216.

[37] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[38] Elvin Isufi et al. *Graph Filters for Signal Processing and Machine Learning on Graphs*. 2024. arXiv: 2211.08854 [eess.SP]. URL: https://arxiv.org/abs/2211.08854.

[39] HV Jagadish et al. “Big data and its technical challenges”. In: *Communications of the ACM* 57.7 (2014), pp. 86–94.

[40] Junteng Jia et al. “Graph-based Semi-Supervised and Active Learning for Edge Flows”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’19. ACM, July 2019, pp. 761–771. DOI: 10.1145/3292500.3330872. URL: http://dx.doi.org/10.1145/3292500.3330872.

[41] Ming Jin et al. *A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection*. 2024. arXiv: 2307.03759 [cs.LG]. URL: https://arxiv.org/abs/2307.03759.

[42] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *International Conference on Learning Representations*. 2017.

[43] Guokun Lai et al. *Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks*. 2018. arXiv: 1703.07015 [cs.LG]. URL: https://arxiv.org/abs/1703.07015.

[44] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[45] See Hian Lee, Feng Ji, and Wee Peng Tay. *SGAT: Simplicial Graph Attention Network*. 2022. arXiv: 2207.11761 [cs.LG]. URL: https://arxiv.org/abs/2207.11761.

[46] Jure Leskovec et al. *Kronecker Graphs: An Approach to Modeling Networks*. 2009. arXiv: 0812.4905 [stat.ML]. URL: https://arxiv.org/abs/0812.4905.

[47] Yaguang Li et al. *Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting*. 2018. arXiv: 1707.01926 [cs.LG]. URL: https://arxiv.org/abs/1707.01926.

[48] H. Lütkepohl. "Forecasting with VARMA models". In: *Handbook of Economic Forecasting*. Vol. 1. 2006, pp. 287–325.

[49] Tom Mitchell. "The Need for Biases in Learning Generalizations". In: (Oct. 2002).

[50] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: 1609.03499 [cs.SD]. URL: https://arxiv.org/abs/1609.03499.

[51] Zheyi Pan et al. "Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 1720–1730. ISBN: 9781450362016. DOI: 10.1145/3292500.3330884. URL: https://doi.org/10.1145/3292500.3330884.

[52] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "Understanding the exploding gradient problem". In: *CoRR* abs/1211.5063 (2012), p. 417.

[53] Thummaluru Siddartha Reddy and Sundeep Prabhakar Chepuri. "Sampling and Recovery of Signals Over Product Cell Structures". In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2024, pp. 13191–13195. DOI: 10.1109/ICASSP48485.2024.10445983.

[54] T. Mitchell Roddenberry, Nicholas Glaze, and Santiago Segarra. *Principled Simplicial Neural Networks for Trajectory Prediction*. 2021. arXiv: 2102.10058 [cs.LG]. URL: https://arxiv.org/abs/2102.10058.

[55] T. Mitchell Roddenberry, Michael T. Schaub, and Mustafa Hajij. "Signal Processing On Cell Complexes". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2022, pp. 8852–8856. DOI: 10.1109/icassp43922.2022.9747233. URL: http://dx.doi.org/10.1109/ICASSP43922.2022.9747233.

[56] T. Mitchell Roddenberry and Santiago Segarra. *HodgeNet: Graph Neural Networks for Edge Data*. 2019. arXiv: 1912.02354 [eess.SP]. URL: https://arxiv.org/abs/1912.02354.

[57] T. Mitchell Roddenberry et al. “Signal Processing On Product Spaces”. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095735.

[58] I. Rojas et al. “Soft-computing techniques and ARMA model for time series prediction”. In: *Neurocomputing* 71.4–6 (2008), pp. 519–537.

[59] Mohammad Sabbaqi and Elvin Isufi. *Graph-Time Convolutional Neural Networks: Architecture and Theoretical Analysis*. 2022. arXiv: 2206.15174 [cs.LG]. URL: https://arxiv.org/abs/2206.15174.

[60] Mohammad Sabbaqi and Elvin Isufi. “Graph-Time Trend Filtering and Unrolling Network”. In: *2023 31st European Signal Processing Conference (EUSIPCO)*. 2023, pp. 1230–1234. DOI: 10.23919/EUSIPCO58844.2023.10289885.

[61] Aliaksei Sandryhaila and Jose M.F. Moura. “Big Data Analysis with Signal Processing on Graphs: Representation and processing of massive data sets with irregular structure”. In: *IEEE Signal Processing Magazine* 31.5 (2014), pp. 80–90. DOI: 10.1109/MSP.2014.2329213.

[62] Stefania Sardellitti and Sergio Barbarossa. *Topological Signal Processing over Generalized Cell Complexes*. 2023. arXiv: 2201.08993 [eess.SP]. URL: https://arxiv.org/abs/2201.08993.

[63] Stefania Sardellitti and Sergio Barbarossa. *Topological Signal Processing over Generalized Cell Complexes*. 2023. arXiv: 2201.08993 [eess.SP]. URL: https://arxiv.org/abs/2201.08993.

[64] Michael T. Schaub and Santiago Segarra. “FLOW SMOOTHING AND DENOISING: GRAPH SIGNAL PROCESSING IN THE EDGE-SPACE”. In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, Nov. 2018, pp. 735–739. DOI: 10.1109/globalsip.2018.8646701. URL: http://dx.doi.org/10.1109/GlobalSIP.2018.8646701.

[65] Prithviraj Sen et al. “Collective classification in network data”. In: *AI magazine* 29.3 (2008), pp. 93–93.

[66] Sho Sonoda and Noboru Murata. “Neural network with unbounded activation functions is universal approximator”. In: *Applied and Computational Harmonic Analysis* 43.2 (Sept. 2017), pp. 233–268. ISSN: 1063-5203. DOI: 10.1016/j.acha.2015.12.005. URL: http://dx.doi.org/10.1016/j.acha.2015.12.005.

[67] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 [cs.CL]. URL: https://arxiv.org/abs/1409.3215.

[68] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[69] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. URL: https://arxiv.org/abs/1710.10903.

[70] G. T. Walker. “On periodicity in series of related terms”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 131.818 (1931), pp. 518–532.

[71] Shuhei Watanabe. *Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance*. 2023. arXiv: 2304.11127 [cs.LG]. URL: https://arxiv.org/abs/2304.11127.

[72] Ruofeng Wen et al. "A multi-horizon quantile recurrent forecaster". In: *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*. Long Beach, CA, USA, 2017, pp. 1–9.

[73] Haixu Wu et al. "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting". In: *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*. Vol. 34. 2021, pp. 22419–22430.

[74] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.2978386. URL: http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[75] Zonghan Wu et al. "Connecting the dots: Multivariate time series forecasting with graph neural networks". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2020, pp. 753–763. DOI: 10.1145/3394486.3403128.

[76] Zonghan Wu et al. *Graph WaveNet for Deep Spatial-Temporal Graph Modeling*. 2019. arXiv: 1906.00121 [cs.LG]. URL: https://arxiv.org/abs/1906.00121.

[77] Sen Yan et al. *Comparative Analysis of Machine Learning-Based Imputation Techniques for Air Quality Datasets with High Missing Data Rates*. 2024. arXiv: 2412.13966 [cs.LG]. URL: https://arxiv.org/abs/2412.13966.

[78] Maosheng Yang and Elvin Isufi. *Convolutional Learning on Simplicial Complexes*. 2023. arXiv: 2301.11163 [cs.LG]. URL: https://arxiv.org/abs/2301.11163.

[79] Maosheng Yang, Elvin Isufi, and Geert Leus. *Simplicial Convolutional Neural Networks*. 2021. arXiv: 2110.02585 [cs.LG]. URL: https://arxiv.org/abs/2110.02585.

[80] Maosheng Yang et al. "Finite Impulse Response Filters for Simplicial Complexes". In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, Aug. 2021. DOI: 10.23919/eusipco54536.2021.9616185. URL: http://dx.doi.org/10.23919/EUSIPCO54536.2021.9616185.

[81] Maosheng Yang et al. "Simplicial Convolutional Filters". In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 4633–4648. ISSN: 1941-0476. DOI: 10.1109/tsp.2022.3207045. URL: http://dx.doi.org/10.1109/TSP.2022.3207045.

[82] Bing Yu, Haoteng Yin, and Zhanxing Zhu. "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI-2018. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 3634–3640. DOI: 10.24963/ijcai.2018/505. URL: http://dx.doi.org/10.24963/ijcai.2018/505.

[83] Zhenheng Yue et al. “TS2Vec: Towards Universal Representation of Time Series”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 8. 2022, pp. 8980–8987. DOI: 10.1609/aaai.v36i8.20883.

[84] G. U. Yule. “On a method of investigating periodicities in disturbed series, with special reference to Wolfer’s sunspot numbers”. In: *Statistical Papers George Udny Yule* 226 (1971), pp. 389–420.

[85] Qinghai Zhang et al. “Spatiotemporal graph structure learning for traffic forecasting”. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2020, pp. 1177–1185.

[86] Chuanpan Zheng et al. *GMAN: A Graph Multi-Attention Network for Traffic Prediction*. 2019. arXiv: 1911.08415 [eess.SP]. URL: https://arxiv.org/abs/1911.08415.

[87] Haoyi Zhou et al. “Informer: Beyond efficient transformer for long sequence time-series forecasting”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 12. 2021, pp. 11106–11115.

A

Additional Experimental Settings

A.1. Encoder–Processor–Decoder Settings

Table A.1 summarizes the encoder, processor, and decoder components used across all deep learning models. For models operating on higher-order topologies (such as SNN-RNN, SPCCNN, and CELL), we employ an independent `MLPEncoder` for each simplex type (nodes, edges, and faces). In contrast, for external baselines like STGCN and GraphWaveNet, we use an `IdentityEncoder`, which passes the raw input features directly to the processor, preserving the original architecture. The `Processor` block implements the core model logic and serves as the backbone of each architecture. As with the encoders, external baselines use an `IdentityReadout` to maintain architectural integrity. For all other models, we apply an `MLPReadout` to transform the output. In higher-order models, this readout operates only on the processed node-level features, as inter-simplicial interactions are handled within the processor.

The `MLPEncoder` and `MLPReadout` components are implemented as multi-layer perceptrons, with multiple hidden layers, each followed by an activation function and a normalization layer. The architectural hyperparameters for these components are listed in Table A.2.

A.2. Hyperparameter Tuning

In order to limit the search space, we conduct hyperparameter tuning only for the hyperparameters of the processor block for each model. The hyperparameters fixed across all models are shown in Table A.2.

The Optuna framework [2] is used for efficient hyperparameter optimization via Bayesian methods. It builds surrogate models of the objective function to guide the search toward promising regions while discarding suboptimal configurations early. We use the Tree-structured Parzen Estimator (TPE) sampler [71], a sequential model-based approach

Model	Encoder	Processor	Readout
FFNN	IdentityEncoder	FFNN	IdentityReadout
FC-LSTM	IdentityEncoder	FC-LSTM	IdentityReadout
GCNN	MLPncoder	GCNN	MLPReadout
GNN-LSTM	MLPEncoder	GNN-LSTM	MLPReadout
GTCNN	MLPEncoder	GTCNN	MLPReadout
STGCN	IdentityEncoder	STGCN	IdentityReadout
GW	IdentityEncoder	GW	IdentityReadout
CELL	MLPEncoder (independent)	CELL	MLPReadout (nodes)
SNN-LSTM	MLPEncoder (independent)	SNN-LSTM	MLPReadout (nodes)
SPCCNN	MLPEncoder (independent)	SPCCNN	MLPReadout (nodes)

Table A.1: Encoder-process-readout components for each deep learning model.

Architecture	
Component	**Hyperparameters**
`MLPEncoder`	Activation Function: ReLU Normalization: LayerNorm Hidden Dims: [8, 16, 16]
`MLPReadout`	Activation Function: ReLU Normalization: LayerNorm Hidden Dims: [16, 8]
Training Settings	
Parameter	**Value**
Batch Size	32
Epochs	200
Optimizer	Adam
Weight Initialization	Xavier
Early Stopping Patience	10
Early Stopping Delta	1e-4
LR Scheduler Patience	5
LR Scheduler Factor	0.5
LR Scheduler Minimum	1e-5

Table A.2: Model training settings that are fixed across all models. These are not in the search space for hyperparameter tuning.

that selects new trials likely to outperform existing ones. To accelerate tuning, underperforming trials are pruned using the `MedianPruner`, which stops trials whose intermediate performance falls below the median of completed trials at the same step. Each trial is allowed to run for at least 10 epochs before being eligible for pruning, and the first 10 trials are run without pruning to gather sufficient data for robust sampling and pruning decisions.

A.2.1. HYPERPARAMETER CONFIGURATIONS FOR ALL MODELS AND DATASETS

Hyperparameter	Range	METR-LA	Solar	Air Pollution
	SPCCNN			
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$1e^{-2}$	$3e^{-4}$	$1e^{-3}$
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$3e^{-5}$	$1e^{-5}$	$1e^{-4}$
Number of Layers (L)	$\{2, \ldots, 5\}$	3	3	4
Output Channels (F)	$\{16, 32, 64, 128\}$	64	64	128
Lower Order (L_1)	$\{2, \ldots, 5\}$	2	5	3
Upper Order (L_2)	$\{2, \ldots, 5\}$	2	2	3
	SNN-LSTM			
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$3e^{-4}$	$1e^{-3}$	$1e^{-3}$
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$1e^{-5}$	$1e^{-5}$	$1e^{-4}$
SNN Layers	$\{2, \ldots, 4\}$	3	2	2
SNN Hidden Dim	$\{16, 32, 64\}$	64	64	64
SNN Lower Order	$\{2, \cdots, 4\}$	3	3	4
SNN Upper Order	$\{2, \cdots, 4\}$	2	2	2
LSTM Layers	$\{2, \ldots, 4\}$	4	2	3
LSTM Hidden Dim	$\{32, 64, 128\}$	128	128	128
	STGCN			
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$1e^{-3}$	$1e^{-3}$	$3e^{-3}$
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$1e^{-5}$	$1e^{-5}$	$1e^{-3}$
Activation Function	{GLU, GTU, ReLU}	ReLU	GTU	GLU
Graph Conv Type	{Spectral, Spatial}	Spectral	Spectral	Spatial
Temporal Features	$\{16, 32, 64, 128, 256\}$	128	64	32
Spatial Features	$\{16, 32, 64, 128\}$	32	64	64
Output Hidden Dim	$\{16, 32, 64, 128, 256\}$	32	64	256
	Graph WaveNet			
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$3e^{-4}$	$3e^{-4}$	$1e^{-2}$
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$1e^{-5}$	$1e^{-5}$	$1e^{-3}$
Adaptive Adjacency	{ True, False }	True	True	False
Residual Channels	$\{16, 32, 64\}$	64	64	16
Dilation Channels	$\{16, 32, 64\}$	32	32	64

Continued on the next page

Hyperparameter	Range	METR-LA	Solar	Air Pollution
Skip Channels	$\{64, 128, 256\}$	256	128	256
End Channels	$\{128, 256, 512\}$	256	128	512
Kernel Size	$\{2, \cdots, 4\}$	2	4	3
Number of Blocks	$\{2, \cdots, 4\}$	4	4	3
Layers Per Block	$\{2, 3\}$	2	3	2
GTCNN				
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$3e^{-3}$	$1e^{-3}$	$3e^{-3}$
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$3e^{-4}$	$1e^{-5}$	$1e^{-5}$
Number of Layers	$\{3, \cdots, 6\}$	3	4	4
Output Channels	$\{32, 64, 128\}$	128	128	128
Filter Taps	$\{3, \cdots, 6\}$	3	6	6
FFNN				
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$3e^{-4}$	$3e^{-4}$	–
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$1e^{-5}$	$3e^{-5}$	–
Number of Layers	$\{3, \ldots, 6\}$	3	3	–
Hidden Dimension	$\{32, 64, 128, 256\}$	256	256	–
LSTM				
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$3e^{-4}$	$3e^{-4}$	–
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$3e^{-5}$	$1e^{-5}$	–
Number of Layers	$\{2, \ldots, 5\}$	2	2	–
Hidden Dimension	$\{32, 64, 128, 256\}$	128	128	–
GNN-LSTM				
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$1e^{-3}$	$1e^{-3}$	–
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$1e^{-5}$	$1e^{-5}$	–
GNN Layers	$\{3, \ldots, 5\}$	3	3	–
GNN Hidden Dim	$\{32, 64, 128\}$	64	64	–
LSTM Layers	$\{2, \ldots, 4\}$	2	4	–
LSTM Hidden Dim	$\{32, 64, 128\}$	128	64	–
GCNN				
Learning Rate	$\{1e^{-4}, 3e^{-4}, 1e^{-3}, 3e^{-3}, 1e^{-2}\}$	$3e^{-4}$	$1e^{-5}$	–
Weight Decay	$\{1e^{-5}, 3e^{-5}, 1e^{-4}, 3e^{-4}, 1e^{-3}\}$	$1e^{-5}$	$1e^{-5}$	–
Layers	$\{3, \ldots, 6\}$	4	3	–
Hidden Dimension	$\{64, 128, 256\}$	128	128	–

Table A.3: Hyperparameter search space and selected configurations for all models across datasets. METR-LA and Solar configurations are for the forecasting task while Air Pollution configurations are for the imputation task.

A.3. EVALUATION METRICS

- **MAE** measures the average magnitude of the absolute errors:

$$\text{MAE} = \frac{1}{|\mathscr{D}|} \sum_{i=1}^{|\mathscr{D}|} \left\| \hat{\mathbf{Y}}_H^{(i)} - \mathbf{Y}_H^{(i)} \right\|_1 . \tag{A.1}$$

- **RMSE** provides an interpretable error in the same units as the original data:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathscr{D}|} \sum_{i=1}^{|\mathscr{D}|} \left\| \hat{\mathbf{Y}}_H^{(i)} - \mathbf{Y}_H^{(i)} \right\|_2^2}. \tag{A.2}$$

Lower values indicate better predictive performance across all metrics. The next section now provides additional experimental settings for training and hyperparameter tuning.

B

Additional Details about Datasets

In this section, we provide additional details and visualisations for the datasets used in our evaluation framework.

B.1. Delhi Air Pollution Dataset

We construct an air pollution forecasting dataset for New Delhi using raw data from the Central Pollution Control Board (CPCB) of India [17]. The selected sensor locations are listed in Table B.1, and the spatial graph constructed over these sensors is shown in Figure B.1.

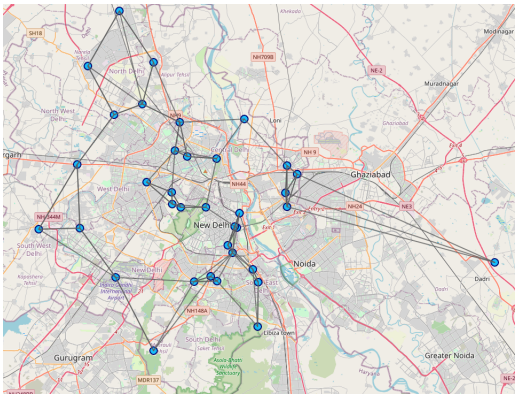


Figure B.1: Spatial graph created using 3-nearest neighbor for the Delhi Air Quality dataset.

Location	Latitude	Longitude
ITO	28.627541	77.243793
Shadipur	28.651027	77.156220
Sirifort	28.550583	77.214799
NSIT Dwarka	28.610273	77.037882
IHBAS Dilshad Garden	28.681169	77.304712
DTU	28.749987	77.118314
North Campus DU	28.688944	77.214125
Pusa	28.634055	77.167847
CRRI Mathura Road	28.611254	77.240116
Aya Nagar	28.472044	77.132942
Mandir Marg	28.634175	77.200475
R K Puram	28.550350	77.185149
Punjabi Bagh	28.661975	77.124156
Anand Vihar	28.650218	77.302706
IGI Airport (T3)	28.555084	77.084401
Lodhi Road	28.591063	77.228079
Okhla Phase 2	28.549291	77.267814
Wazirpur	28.697544	77.160440
Najafgarh	28.609013	76.985453
Narela	28.854882	77.089215
Sonia Vihar	28.733247	77.249589
Rohini	28.738268	77.082215
Nehru Nagar	28.563867	77.260810
Vivek Vihar	28.671246	77.317654
Patparganj	28.634731	77.304571
Dr. Karni Singh Shooting Range	28.499727	77.267095
Major Dhyan Chand Stadium	28.612547	77.237335
Dwarka Sector 8	28.572038	77.572038
Ashok Vihar	28.690979	77.176524
Jawaharlal Nehru Stadium	28.582846	77.234366
Jahangirpuri	28.729617	77.166631
Sri Aurobindo Marg	28.556310	77.206338
Bawana	28.793229	77.048335
Mundka	28.682314	77.034937
Pusa (DPCC)	28.637672	77.157144
Alipur	28.797226	77.133136

Table B.1: Sensor locations and coordinates in the Delhi Air Quality dataset.

B.2. SOLAR ENERGY DATASET

The Solar Energy dataset [43], collated by the National Renewable Energy Laboratory, records the solar power output (in megawatts) for 137 photovoltaic stations in Alabama, USA. The spatial graph, created using 3-nearest neighbor, remains disconnected as seen in Figure B.2.

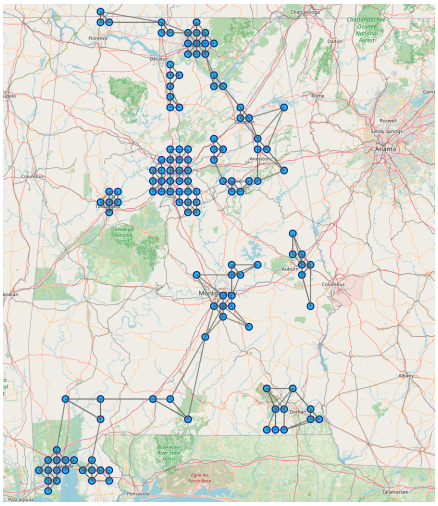


Figure B.2: Spatial graph created using 3-nearest neighbor for the Solar Energy dataset.

B.3. METR-LA DATASET

The METR-LA dataset [39] consists of traffic speed readings from 207 sensors installed on highways in the Los Angeles County. The spatial graph is created using 3-nearest neighbor and is provided in Figure B.3.

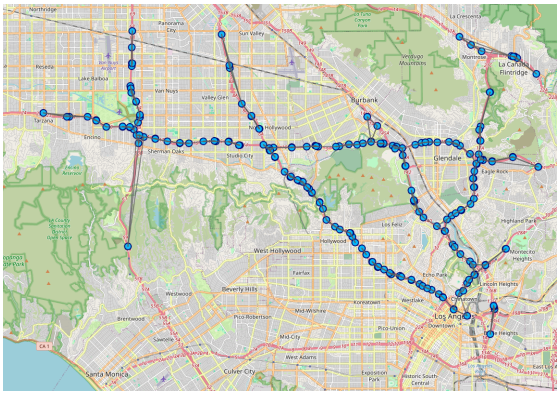


Figure B.3: Spatial graph created using 3-nearest neighbor for the METR-LA dataset.