ABSTRACT

Title of Dissertation: BOUNDARY-AWARE
NEURAL POINT-TO-POINT SET EVOLUTION
FOR INTERACTIVE TRAJECTORY PREDICTION

Haeyun Seo
Doctor of Philosophy, 2025

Dissertation Directed by: Professor Benjamin Kedem
Department of Mathematics

Trajectory prediction in multi-agent settings requires models that can handle complex interactions under real-world sensing constraints, such as partial observations and temporal irregularities. This thesis presents a unified framework for interactive trajectory prediction that is grounded in deep mathematical theory and realized in a series of practical highly parallelized neural architectures. We establish a theoretical foundation built on universality theorems for set-equivariant functions and the unique decomposition of the spatio-temporal symmetry group, which provide a principled justification for our encoder designs. We translate this theory into a practical system through three signature characteristics: boundary awareness, addressed with a data type classification and loss function; causal-pointwise equivariance, implemented via multi-stage encoders and a JIT-compatible data processing pipeline; and continuous-time dynamics, realized with adaptive causal convolutions and Neural ODE decoders. Our final model, implemented in JAX, is a VAE that demonstrates significant parameter efficiency. Experimental results

on standard benchmarks show that our models are competitive with state-of-the-art methods. Furthermore, we demonstrate that our final architecture successfully handles the more challenging task of multi-horizon prediction from partial trajectories, offering a robust and flexible solution for real-world forecasting.

BOUNDARY-AWARE NEURAL POINT-TO-POINT SET EVOLUTION FOR INTERACTIVE TRAJECTORY PREDICTION

by

Haeyun Seo

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:
Professor Benjamin Kedem, Chair/Advisor
Professor Lizhen Lin
Professor Vince Lyzinski
Professor Deep Ray
Professor P. S. Krishnaprasad

© Copyright by
Haeyun Seo
2025

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Benjamin Kedem, for introducing me to this fascinating research topic and for sharing with me the foundational ideas of state space models, smoothing splines, and data fusion. I am deeply thankful for his unwavering support and invaluable guidance throughout my studies. He ignited my passion for research—and it continues to this day. I truly admire his profound dedication to research: the way his eyes lit up with passion for his work—that was my very first impression of him, and one I will never forget.

I am also grateful to Professor Lizhen Lin and Professor Vince Lyzinski for consistently checking in on me and offering their insightful advice. My thanks extend to Professor Deep Ray and Professor Tamás Darvas, who generously made time to discuss my research with me. Professor Darvas introduced me to the concept of jet bundles at a time when I was struggling to formalize certain ideas—this insight has since become a key stepping stone in my theoretical development. I would also like to thank Professors Lizhen Lin, Vince Lyzinski, Deep Ray, and P. S. Krishnaprasad for kindly agreeing to serve on my dissertation committee.

Looking back, I feel truly fortunate to have met such wonderful friends during my PhD in College Park. Shin guided me through life on campus as a fellow math PhD student. I shared many study sessions and mathematical discussions with Jackson, and I enjoyed spending time with Alex, Josué, and Haoran. In particular, I consider myself especially lucky to have become

close friends with Michael, who continued to reach out to me during a time when I was struggling with self-doubt.

I extend my heartfelt thanks to the staff members in the math department. Ruth has been a great friend and helped me immensely with administrative matters. Cristina consistently offered her support during my periods of uncertainty. Trystan and Jemma not only assisted me with various administrative tasks but also shared thoughtful conversations.

I also thank OpenAI's ChatGPT and Google's Gemini, for helping me debug complex array indexing issues, identify bugs in my code, and refine my written English throughout this process; any remaining mistakes are entirely my own.

Finally, I would like to thank my mom, my relatives, and my friends in Korea. My mom always encouraged me to live my life in pursuit of what I want—not what she or others expect of me—which is quite unusual for a Korean mother. Without her unique philosophy on parenting, I would never have imagined pursuing a PhD.

Table of Contents

Acknowledgements ii

Table of Contents iv

List of Tables ix

List of Figures x

List of Abbreviations xi

Chapter 1: Introduction 1
- 1.1 Context and Motivation . . . 2
- 1.2 Problem Statement and Objectives . . . 4
- 1.3 Contributions . . . 6
- 1.4 Thesis Organization . . . 10

Chapter 2: Related Work in Trajectory Prediction 14
- 2.1 Recurrent Neural Network Models . . . 15
  - 2.1.1 Vanilla RNN . . . 15
  - 2.1.2 Long Short-Term Memory (LSTM) . . . 15
  - 2.1.3 Gated Recurrent Unit (GRU) . . . 16
  - 2.1.4 Application to Trajectory Prediction . . . 17
- 2.2 Temporal Convolutional and Attention-Based Models . . . 17
  - 2.2.1 Temporal Convolutional Networks (TCNs) . . . 17
  - 2.2.2 Attention Mechanism and Transformer . . . 18
  - 2.2.3 Transformers for Trajectory Prediction . . . 19
  - 2.2.4 Trade-Offs and Gaps . . . 20
- 2.3 Spatial Interaction Models . . . 20
  - 2.3.1 Graph-Based and Message-Passing Networks . . . 21
  - 2.3.2 Set-Based Equivariant Architectures . . . 21
  - 2.3.3 Unified Dynamic Set-Evolution . . . 22
- 2.4 Probabilistic and Stochastic Discrete-Time Models . . . 22
  - 2.4.1 Variational Autoencoders (VAEs) and Trajectron++ . . . 23
  - 2.4.2 Generative Adversarial Networks (GANs), WGANs & Social GAN . . . 24
  - 2.4.3 Diffusion Models . . . 25
- 2.5 Continuous-Time Forecasting . . . 26
  - 2.5.1 Neural ODEs and Neural CDEs . . . 26

2.5.2 Neural SDEs . . . 27
2.6 Censoring and Truncation in Statistical Practice . . . 28
2.7 Missing-Data in Trajectory Prediction . . . 30
2.7.1 Overview: Types of Missing Data in Trajectories . . . 30
2.7.2 Imputation-Based Approaches (Mainstream Methods) . . . 30
2.7.3 Limitations of Imputation for Boundary Cases . . . 31
2.7.4 Modeling Boundary Events: An Underexplored Area . . . 31

Chapter 3: Spatial Representation & Permutation Equivariance 33
3.1 Hausdorff and Wasserstein Distances . . . 34
3.2 Unbalanced Optimal-Transport Metrics and the Reservoir-Augmented Wasserstein Distance . . . 37
3.3 Idempotent Measures and Support Functionals . . . 41
3.3.1 Idempotent Integration Framework and the Kolokoltsov–Maslov Representation Theorem . . . 42
3.3.2 Support Functionals and Closed-Set Embedding . . . 48
3.4 Uniform Approximation via the Stone–Weierstrass Theorem . . . 57
3.4.1 The Classical Stone–Weierstrass Theorem . . . 57
3.4.2 Prolla's Vector-Valued Extension . . . 58
3.4.3 Neural Network Universal Approximation . . . 59
3.4.4 Invariant Universal Approximation Theorems on Hyperspaces . . . 63
3.5 Universal Approximation by Set- and Measure-Based Architectures . . . 64
3.5.1 Universal Approximation of Invariant Functions . . . 65
3.5.2 Universal Approximation of Equivariant Functions . . . 70

Chapter 4: Dynamic Modeling & Temporal Equivariance 79
4.1 Recurrence, Markov, and State-Space Models . . . 80
4.1.1 Deterministic k-Step Recurrence and RNNs . . . 80
4.1.2 Linear–Gaussian State-Space Models and the Kalman Filter . . . 82
4.1.3 General State-Space Models and Nonlinear/Non-Gaussian Filters . . . 84
4.1.4 Synthesizing Deterministic and Stochastic Perspectives . . . 86
4.2 Latent Dynamics on Padded Jets and Trajectories . . . 88
4.2.1 It's All Finite Sampling . . . 88
4.2.2 The Jet Bundle Concept . . . 89
4.2.3 Dynamics Lifted to the Jet Bundle . . . 94
4.2.4 Conceptual Ideal: Padded Infinite-Order Jets ($\widetilde{J}_0^\infty$) . . . 95
4.2.5 Infinite History (Function Space Bundles) . . . 98
4.2.6 Padded Partial Trajectories in History Bundles . . . 99
4.2.7 Causal (Time) Equivariance in History Bundles . . . 101
4.3 Retarded Functional Differential Equations . . . 102
4.3.1 Delay Differential Equations . . . 102
4.3.2 RFDEs with Finite Delay Intervals . . . 104
4.3.3 RFDEs with Infinite Delay Intervals . . . 107
4.3.4 The Solution Operator and Semigroup Property for Autonomous RFDEs . . . 109
4.3.5 RFDEs in the Proposed Framework . . . 110

4.4 Universal Approximation of Temporally Equivariant Operators by Convolutional Networks . . . 112
4.4.1 Formalizing Equivariant Operators . . . 113
4.4.2 A Universal Approximation of $\mathbb{R}^n$-Equivariant Operators . . . 114
4.4.3 Learning RFDEs from Universal Approximation . . . 115

Chapter 5: A Theory of Spatio-Temporal Set Evolution 117
5.1 Causal-Pointwise Equivariance . . . 118
5.1.1 The Data Space . . . 118
5.1.2 The $\mathbb{R} \times \mathfrak{S}_n$ Symmetry . . . 121
5.1.3 Causal Convolutions and Set-Pooling for Practical Equivariance . . . 124
5.1.4 Fundamental Decomposition and Architectural Consequences . . . 125
5.1.5 A Conjecture of Universal Approximation Theorems for Causally-Pointwise Equivariant Maps . . . 127
5.2 A Functional Inequality Characterization of Point-to-Point Compact Set Evolution 131
5.2.1 The Extremal-Integral Characterization . . . 135
5.2.2 Necessity of the Conditions . . . 140
5.2.3 Comparison with the Weak Formulation of the Continuity Equation . . . 146
5.3 A Morphological Characterization of Point-to-Point Set Evolution . . . 148
5.3.1 Primitives and Reachable Sets in Single-Valued Evolutions . . . 149
5.3.2 The Autonomous Point-to-Point Set Evolution . . . 157
5.3.3 Comparison with Measure-Theoretic Autonomous Transport . . . 158
5.3.4 RFDEs for Hyperspaces . . . 164

Chapter 6: Signature Characteristics of Trajectory Prediction 179
6.1 The Boundary Challenge . . . 180
6.1.1 A Complete Taxonomy of Trajectory Censoring . . . 180
6.1.2 General Log-Likelihood . . . 183
6.1.3 A Differentiable Surrogate for the Boundary Exit Term . . . 185
6.2 The Encoder-Decoder Architecture and Causal-Pointwise Equivariance . . . 187
6.2.1 The Encoder-Decoder Architecture . . . 187
6.2.2 Implementing Causal-Pointwise Equivariance in the Encoder . . . 189
6.2.3 Implementing Causal-Pointwise Equivariance in the Decoder . . . 195
6.3 The Advantage of Continuous-Time Modeling: Handling Irregular Data . . . 197
6.3.1 Neural Implementation of Continuous-Time Processing . . . 198

Chapter 7: Neural Architecture & Implementation 202
7.1 The Choice of the JAX Ecosystem . . . 202
7.1.1 Key Advantages for Spatio-Temporal Modeling . . . 203
7.1.2 The Library Stack . . . 204
7.2 The Temporal Conjugation Pipeline . . . 205
7.3 Encoder Architectures . . . 207
7.3.1 Canonical Pathways for Benchmark Models . . . 208
7.3.2 The Time→Agent→Time Encoder for Partial Trajectories and Multi-Horizon Prediction . . . 209

7.4 Decoder Architectures: Continuous-Time Trajectory Generation . . . 212
7.4.1 Static-Context Decoder . . . 215
7.4.2 Dynamic-Context Decoder . . . 215
7.4.3 Solver Selection for Real-Time Application . . . 216
7.5 Probabilistic Framework and Training Objectives . . . 216
7.5.1 The VAE Module . . . 217
7.5.2 Loss Functions . . . 218

Chapter 8: Experimental Results & Analysis 221
8.1 Experimental Setup . . . 221
8.1.1 Datasets . . . 222
8.1.2 Evaluation Metrics . . . 222
8.1.3 Implementation Details . . . 223
8.2 Benchmark Performance and Encoder Comparison . . . 224
8.2.1 Quantitative Results . . . 225
8.2.2 Discussion . . . 225
8.3 Performance on Partial Trajectories and Multi-Horizons . . . 226
8.3.1 Bridging Experiment: Relative vs. Absolute Coordinates . . . 226
8.3.2 Multi-Horizon Performance . . . 227
8.3.3 Discussion . . . 227
8.4 Qualitative Analysis . . . 229
8.4.1 Encoder Interactions: Summarizing the Past . . . 230
8.4.2 Decoder Interactions: Planning the Future . . . 232
8.5 Chapter Summary . . . 232

Chapter 9: Conclusion & Future Work 235
9.1 Summary of Contributions . . . 235
9.2 Discussion and Broader Implications . . . 236
9.2.1 Bridging Rigorous Theory and Practical Implementation . . . 237
9.2.2 Moving Beyond Standard Benchmarks . . . 237
9.2.3 Implications for Robotics . . . 238
9.3 Limitations and Critique . . . 238
9.4 Future Work . . . 240

Appendix A: Core Architectural Modules 243
A.1 Causal Convolutions . . . 243
A.2 Interaction Modules . . . 248
A.3 Probabilistic Module (VAE) . . . 252
A.4 Decoder Vector Fields and Solvers . . . 255

Appendix B: Main Encoder Functions and Implementation of Universality Theorems 264
B.1 Benchmark Encoders . . . 264
B.2 Multi-Horizon Encoder . . . 270
B.3 Implementation of the Universality Theorems . . . 274

Appendix C: Loss Function Implementation 276

Bibliography 283

List of Tables

2.1 Summary of missing-data regimes and existing approaches 32

3.1 Dual-norm formulas of various distances and their associated test-space constraints 57

5.1 Comparison of autonomous measure vs. set evolutions 163

6.1 Trajectory data sample 191

8.1 Hyperparameter settings for all models and datasets 224
8.2 ADE / FDE on the ETH/UCY benchmark 225
8.3 Relative vs. Absolute Coordinates on ZARA (8/12 Split) 227
8.4 Multi-Horizon Performance (ADE / FDE) on ZARA01 228
8.5 Multi-Horizon Performance (ADE / FDE) on ZARA02 228
8.6 Multi-Horizon Performance (ADE / FDE) on ZARA03 228
8.7 Most Salient Agents per Channel for the Encoder at `t=5` 231
8.8 Aggregate Encoder Adjacency Matrix at `t=5` 231
8.9 Most Salient Agents per Channel for the Decoder at `t=5` 232
8.10 Aggregate Decoder Adjacency Matrix at `t=5` 233

B.1 Code-level implementation of each theoretical component 274
B.2 Correspondence for the Dynamic-Context Decoder 275

List of Figures

5.1 An example of a point cloud representing a doughnut shape 118

6.1 A complete taxonomy of trajectory censoring within a finite observation window 182

7.1 A complete diagram of the VAEST (Agent → Time) architecture 210
7.2 A complete diagram of the VAETS (Time → Agent) architecture 211
7.3 A complete diagram of the Time→Agent→Time Architecture with a Static-Context Decoder 213
7.4 A complete diagram of the Time→Agent→Time Architecture with a Dynamic-Context Decoder 214

8.1 Agent positions at `t=5` in ZARA01 (agents 0–8). 230

List of Abbreviations

AC	Absolutely Continuous
ADE	Average Displacement Error
API	Application Programming Interface
AR	AutoRegressive
CDE	Controlled Differential Equation
CNN	Convolutional Neural Network
CVAE	Conditional Variational Autoencoder
DDE	Discrete Delay Differential Equation
DDPM	Denoising Diffusion Probabilistic Model
EKF	Extended Kalman Filter
ELBO	Evidence Lower Bound
FDE	Final Displacement Error
FOV	Field of View
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HJB	Hamilton–Jacobi–Bellman
JIT	Just-In-Time
KL	Kullback-Leibler
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
MPNN	Message-Passing Neural Network

NDDE	Neural Delay Differential Equation
ODE	Ordinary Differential Equation
PF	Particle Filter
RFDE	Retarded Functional Differential Equation
RNN	Recurrent Neural Network
SDE	Stochastic Differential Equation
SSM	State-Space Model
TCN	Temporal Convolutional Network
TDNN	Time Delay Neural Network
TPU	Tensor Processing Unit
UAT	Universal Approximation Theorem
VAE	Variational Autoencoder
VAEST	VAE Agent→Time (Variational Autoencoder Space Time)
VAETS	VAE Time→Agent (Variational Autoencoder Time Space)
WGAN	Wasserstein Generative Adversarial Network
XLA	Accelerated Linear Algebra

Chapter 1: Introduction

Forecasting agents' future positions is critical for safe autonomous systems—from self-driving cars anticipating pedestrian crossings to drones navigating crowded plazas. Realistic trajectory predictors must juggle three distinct challenges:

1. **Boundary awareness:** Agents may enter, exit, or be occluded within the sensor's finite spatio-temporal window.

2. **Time-homogeneity with causality and agent anonymity:** The same past-only dynamics apply under any time shift, and agents are treated as anonymous points.

3. **Continuous interactive dynamics:** Each agent evolves smoothly under a shared, peer-driven flow.

Crucially, our architecture treats these as separate yet composable modules—boundary-event modeling, causal and permutation-equivariant encoding, and continuous point-to-point set-evolution ODEs—so that improvements in any one can slot into the overall predictor without reworking the rest. A key design goal is parameter efficiency: on standard benchmarks, our models are comparable to other approaches *while using far fewer parameters*. In this dissertation, we assemble these building blocks into a boundary-aware, continuous-time, point-to-point set-evolution framework—universally expressive yet practical under real-world sensing

constraints—instantiate it as a neural encoder–decoder, and demonstrate on standard benchmarks that its parameter efficiency stems from the tight coupling of theory-driven encoders with a continuous-time decoder.

1.1 Context and Motivation

While trajectory prediction is a cornerstone of modern autonomous systems, consistently achieving high-quality forecasts in complex, real-world environments presents significant challenges. These challenges arise fundamentally from the nature of sensor data and multi-agent dynamics, making it difficult for conventional methods to fully embody the key traits of robust predictors introduced previously [1]. Specifically, autonomous systems must contend with the complexities introduced by spatio-temporal boundaries that limit observability, the strict requirements of time-homogeneity with causality, and the need to maintain interactive agent anonymity when processing influences from multiple, unlabeled peers. This section elaborates on the context and motivation for our work by detailing these critical challenges and highlighting the limitations of existing approaches.

Real deployments confront spatio-temporal boundary problems at every turn. Cameras, LiDARs, and radar only observe a finite window in space and time, so trajectories routinely begin or end mid-sequence: an agent might slip into view half-way through, vanish as it moves out of range, or be occluded by obstacles [2]. A handful of prior works handle such occlusions via data imputation or augmentation, but none have rigorously distinguished spatial truncation from temporal censoring or treated these phenomena within a unified statistical framework. Our thesis is the first to formalize both truncation and censoring in trajectory forecasting and to weave

that distinction into the core modeling and evaluation.

Equally crucial are symmetries in two dimensions. First, the notion of time-homogeneity with causality demands that the mapping from past to future be consistent: what holds at one moment must hold at all, and the model must never exploit information that lies strictly beyond the current timestamp [3]. Second, agent anonymity requires that no individual's identity or index confer special treatment: each agent's forecast must impartially fuse influences from every other agent, regardless of how many are present or how they are labeled [4]. Together, these symmetry principles ensure consistency under time shifts and agent relabelings.

Even though motion is inherently continuous, most predictors operate on discrete video frames or timestep grids [5]. This discretization can obscure rapid maneuvers that occur between samples, struggle with irregular or asynchronous sensor streams, and tie performance to a fixed frame rate. A continuous-time characterization restores fidelity: by modeling trajectories as flows of evolving sets under smooth vector fields, we capture sub-frame interactions, fuse multi-sensor data at arbitrary timestamps, and guarantee that the same underlying dynamics govern every interval regardless of sampling cadence.

Putting these symmetry principles and continuity together leads naturally to a point-to-point set evolution view: at each moment, we treat an agent as a point in an evolving set of peers, and let a shared, causal vector field move that point based on its interactions with the set. In practice, each agent's state combines its own interactive history summary and its current position, and the full set of these states informs how every point advances. This perspective elegantly captures all inter-agent effects under a unified, continuous-time framework.

1.2 Problem Statement and Objectives

In this section we pinpoint the formal challenges introduced above and lay out our goals. We first describe precisely how censoring and truncation interrupt trajectory data, then introduce the symmetry constraints of time-homogeneity with causality and agent anonymity, and finally explain our choice of set-based representations over measures for processing historical data and how interactions are encoded within our proposed encoder-decoder framework. Our ultimate objective is a unified and universal mathematical framework and its neural instantiation that addresses the three signature characteristics of interactive trajectory prediction: spatio-temporal boundary challenges, respecting causal time-shift and permutation equivariance under varying n, and continuous-time point-to-point set evolution under infinitesimal dynamics.

Real-world trajectory data is inherently affected by incomplete observations arising from limitations in sensor coverage and finite observation periods. These incompleteness issues can be broadly categorized as temporal censoring and spatial truncation. Temporal censoring occurs when an agent's trajectory is only partially observed because its entry into the environment happened before the start of the observation window (censoring of the start) or its exit will happen after the end of the observation window (censoring of the end). In contrast, spatial truncation occurs when parts of an agent's path are unobserved because they fall outside the sensor's spatial boundary within the observation interval. This includes agents entering or leaving the field of view. Our model, being conditional, never tries to predict unseen entries; instead, it focuses on in-view agents. Our framework achieves boundary awareness not through explicit geometric modeling within the network, but by incorporating boundary-derived cues—for example, the time step at which an agent crosses the sensor's boundary—directly into the training objective.

This encourages the network to learn dynamics that respect agents exiting the field of view by incorporating a survival-analysis–inspired boundary-consistency loss—derived from the observed exit times—rather than relying on imputation.

To handle the symmetry inherent in prediction under uncertainty, we appeal to two core principles: causal time-homogeneity and agent-anonymity. These symmetries are fundamental properties rigorously enforced throughout our framework, within both the encoder that processes historical data and the decoder that generates predictions. Causal time-homogeneity states that the predictive rule should not depend on the absolute timestamp, only on the past relative to the present. Mathematically, this is modeled as $\mathbb{R}$-equivariance with causality—an invariance under time shifts, constrained to preserve the past–future arrow. Agent-anonymity requires that forecasts be unaffected by how agents are labeled; this leads naturally to $\mathfrak{S}_n$-equivariance, where the model's outputs permute consistently with its inputs. A key objective of this thesis is to provide a universal characterization of trajectory prediction that, by design, rigorously satisfies these causal $\mathbb{R}$-equivariance and varying-n $\mathfrak{S}_n$-equivariance—properties we collectively call causal-pointwise equivariance.

To represent the configuration of agents at a given time, we consider two natural modeling abstractions: measures and sets [6, 7]. In a measure-theoretic approach, each agent is assigned a weight, and the system is treated as a finite measure μ on $\mathbb{R}^d$. This is common in density-based or message-passing models, but has drawbacks in dynamic settings: weights must be reassigned as agents come and go, and the total mass becomes ambiguous. Alternatively, we adopt a set-based approach: agents are viewed as a finite subset $K \subset \mathbb{R}^d$, with no weights attached. This aligns naturally with permutation-equivariant pooling and avoids complications from varying total mass—particularly when paired with a fixed "environment" dummy agent that ensures stability in empty

regions.

Interactions between agents are paramount in determining future trajectories. Within our framework, we capture these interactions by considering each agent not in isolation, but as a pointed object [8, 9]. For a specific agent located at state x, its context is defined by the configuration of all other agents. This context can be represented as a measure μ or, in our chosen set-based approach, as a set K of the states of all other agents. The pointed object is then represented as (x, μ) or (x, K), explicitly highlighting the agent of interest and its surrounding interactive environment.

The core of our proposed framework lies in modeling the continuous-time point-to-point set evolution [10] of agents with spatio-temporal boundary constraints incorporated. This evolution is governed by a vector field $v(t, x)$ defined over the state space. We will realize this framework with a neural encoder–decoder that summarizes past histories via causal, permutation-symmetric pooling into per-agent latents which constitutes states along with positions, and generates states via continuous set-aware ODEs from the underlying static-context or dynamic-context vector field. Importantly, we decouple boundary handling from the network: boundary awareness is enforced via our loss function, while causal-pointwise equivariance and continuous interactive dynamics are built directly into the encoder–decoder architecture.

1.3 Contributions

We organize our key contributions around four interlocking pillars, each grounded in rigorous theory and realized in a practical neural architecture.

1. **Support-Functional Characterization of Sets** We show that any nonempty closed set

$C \subset \mathbb{R}^d$ is completely determined by its support functional

$$F_C(\eta) = \max_{x \in C} \eta(x)$$

for all test functions η vanishing (as $-\infty$) at infinity. Concretely, a support functional $F : C_0(\mathbb{R}^d; \mathbb{R} \cup \{-\infty\}) \to \mathbb{R}$ is characterized by being sup-additive, translation-equivariant, and positively homogeneous (equivalently, constant-capping preserving). Analogous to the Riesz representation theorem for measures, this result implies that max-pooling alone suffices to recover set geometry. Equivalently, the collection of nonempty closed sets in $\mathbb{R}^d$ can be identified with the dual space of the idempotent semiring with three binary operations, $\big(C_0(\mathbb{R}^d; \mathbb{R} \cup \{-\infty\}, \vee, +, \times)\big)^*$ or $\big(C_0(\mathbb{R}^d; \mathbb{R} \cup \{-\infty\}, \vee, +, \wedge)\big)^*$

2. **Universality Theorems for Equivariance**

The permutation equivariance is naturally captured by treating each agent as a pointed object (x, K), where x is the agent's state and K is the set of all agent states. Any continuous, set-equivariant map $F : \mathrm{Pt}(\mathcal{K}(D)) \to \mathbb{R}^d$ (where $\mathrm{Pt}(\mathcal{K}(D))$ is the space of pointed compact subsets of D) captures exactly those functions on (x, K) that commute with permutations of K as a set. We prove that a set-equivariant map $F : \mathrm{Pt}(\mathcal{K}(D)) \to \mathbb{R}^d$ admits the form

$$F(x, K) \approx \Gamma\big(x, \max_{y \in K} h(y)\big) \text{ or } F(x, K) \approx \gamma\big(\max_{y \in K} \phi(x, y)\big)$$

for suitable functions $h : D \to \mathbb{R}^b$, $\Gamma : D \times \mathbb{R}^b \to \mathbb{R}^d$, $\gamma : \mathbb{R}^b \to \mathbb{R}^d$, and $\phi : D \times D \to \mathbb{R}^b$ for some bottleneck dimension b.

Here $\max_{y \in K} h(y)$ is the global summary of up to b agents (including a dummy), and

$\Gamma(x, \cdot)$ fuses that summary with the individual x. Likewise, $\phi(x, y)$ captures pairwise effects, and $\max_{y \in K} \phi(x, y)$ gives each agent its personalized interaction from the top b neighbors.

3. **Two-Sided Extremal–Integral Inequalities**

Extending the idea of the functional characterization of closed sets via max-pooling, we prove that for any continuous vector field $v(t, x)$ that is uniformly Lipschitz in space, for any pair of timestamps, $t_1 < t_2$, and for any $\eta \in C_c^\infty([t_1, t_2] \times \mathbb{R}^d)$ with $K_0 = K$, the evolving set K_t under the flow generated by v is the unique solution to the two-sided extremal-integral inequalities,

$$\int_{t_1}^{t_2} \min_{x \in K_t} [\partial_t \eta + \nabla \eta \cdot v] \, dt \;\leq\; \max_{x \in K_{t_2}} \eta(t_2, x) - \max_{x \in K_{t_1}} \eta(t_1, x) \;\leq\; \int_{t_1}^{t_2} \max_{x \in K_t} [\partial_t \eta + \nabla \eta \cdot v] \, dt.$$

These extremal-integral bounds are analogous to the continuity PDE $\partial_t \mu_t + \nabla_x \cdot (v(t, x)\mu_t) = 0$ [11, 12] for measures, and they provide a principled justification for using max-pooling to capture causal dynamics in interactive neural models.

4. **Neural Encoder–Decoder with Unique Equivariance Decomposition**

Guided by our theoretical results, we implement an end-to-end model with two main components:

(a) **Equivariant Encoder.** We treat the raw input as a set of partial histories for all agents and encode it in a causal, set-equivariant manner. Because the symmetry group $\mathbb{R} \times \mathfrak{S}_n$ splits uniquely into continuous-time shifts and discrete permutations, we support two canonical architectural pathways:

- Time $\rightarrow$ Agents: First apply causal convolution to each agent's history, then max-pool across agents,
- Agents $\rightarrow$ Time: First max-pool across agents at each timestamp, then apply causal convolution to the resulting sequence.

Furthermore, our conjectured universal approximation theorems for such equivariant operators motivate more expressive, two-stage architectures (i.e., Time $\rightarrow$ Agents $\rightarrow$ Time) that are necessary for full generality. The resulting latent $\ell^{(i)}(t_0)$ for agent i at time t_0 serves as a probabilistic prior condition for the decoder, providing a rich summary of the past. It is concatenated with its current position $x^{(i)}(t_0)$ to form the ODE initial state

$$y^{(i)}(t_0) = \big[\ell^{(i)}(t_0),\, x^{(i)}(t_0)\big].$$

(b) **Continuous-Time Decoder.** We decode future trajectories by integrating a learned neural vector field via a Neural ODE [13]. Two variants respect causal time-homogeneity:

$$\dot{y}(t) = v\big(y(t)\big) \quad \text{(static context)}$$

$$\dot{y}(t) = v\big(y(t),\, K(t)\big) \quad \text{(dynamic context)}$$

where $K(t)$ is the current set of all agent states. The full state $y(t) \in \mathbb{R}^{\ell+d}$ stacks the latent ℓ-vector and the d-dimensional position, so that the predicted positions are simply read off from the last d entries of $y(t)$ without any additional projection layer. This decoder guarantees the same past-only dynamics apply at every time step and remains permutation-equivariant.

A crucial consequence of this end-to-end, continuous-time design is the model's inherent multi-horizon flexibility. Because the adaptive causal kernel in the encoder can produce a valid latent summary $\ell^{(i)}(t_0)$ from a history of any length, and the Neural ODE in the decoder can be integrated forward from any initial time t_0 for any desired duration, the entire architecture is not tied to a single, fixed observation/prediction split. This allows a single, unified model to handle a wide range of forecasting tasks, a significant advantage over traditional architectures designed for a single, fixed benchmark.

1.4 Thesis Organization

The remainder of this thesis is structured to guide the reader from foundational theory through algorithmic design to empirical validation.

- **Chapter 2: Related Work in Trajectory Prediction**

  We survey the foundational and contemporary techniques in trajectory prediction, critically assessing existing approaches to sequential modeling, multi-agent interaction, probabilistic forecasting, continuous-time dynamics, and missing data handling in trajectory prediction. This chapter identifies key limitations and research gaps, particularly regarding principled boundary awareness, causal-pointwise equivariance, and unified continuous interactive dynamics, thereby motivating our integrated but composable framework.

- **Chapter 3: Spatial Representation & Permutation Equivariance**

  We introduce the core mathematical tools for our framework, including Hausdorff and Wasserstein distances, idempotent measures, support functionals of compact/closed sets, and the Stone–Weierstrass/Prolla's generalization. We then use these tools to develop func-

tional characterizations of support functionals and to prove our universality theorems for set- and measure-equivariant functions.

- **Chapter 4: Dynamic Modeling & Temporal Equivariance**

  We build the framework for modeling continuous-time trajectory dynamics. We begin by reviewing classical discrete-time models (recurrence relations, state-space filters) to motivate a shift to continuous-time representations based on history functions. We then formalize these dynamics using the theory of Retarded Functional Differential Equations (RFDEs) and establish that the functional governing autonomous evolution can be universally approximated by a causal convolutional neural network.

- **Chapter 5: A Theory of Spatio-Temporal Set Evolution**

  We synthesize the spatial and temporal frameworks to develop a theory for point-to-point set evolution. We first formalize the crucial combined symmetry of causal-pointwise equivariance. We then establish two equivalent characterizations for the evolution of a compact set under a vector field: a novel formulation based on two-sided extremal-integral inequalities and a second rooted in the established mutational analysis [10]. Building on this, we extend the analysis to systems with memory, culminating in an existence and uniqueness proof for a novel class of coupled functional differential equations on a hyperspace of trajectories. This result provides the rigorous mathematical foundation for the interactive, history-dependent systems our models aim to capture.

- **Chapter 6: Signature Characteristics of Trajectory Prediction**

  We translate our theory into a practical framework by analyzing three signature characteristics of real-world prediction. We formalize the boundary challenge with a complete

taxonomy of censoring and a principled, survival-analysis-inspired loss function . We then detail the architectural principles of causal-pointwise equivariance, including the canonical encoder pathways and our temporal conjugation technique for handling ragged data . Finally, we discuss the advantages of continuous-time dynamics including handling irregular data.

- **Chapter 7: Neural Architecture & Implementation**

  We present the concrete neural implementation of our framework . This chapter details our choice of the JAX [14] ecosystem, the specific implementation of our three encoder architectures (Agent→Time, Time→Agent, and Time→Agent→Time), our continuous-time decoders (static and dynamic context neural ODEs), and the final probabilistic and boundary-aware training objectives.

- **Chapter 8: Experimental Results & Analysis**

  We present a comprehensive empirical validation of our models. We first compare our architectures against standard benchmarks to demonstrate their parameter efficiency. We then evaluate our final model's primary contribution: its robust performance on partial trajectories across a wide range of prediction horizons. The chapter concludes with a qualitative analysis that provides interpretable insights into the model's learned dynamic interaction mechanisms.

- **Chapter 9: Conclusion & Future Work**

  We summarize our contributions, discuss implications, limitations, and critique for interactive forecasting and control, and outline promising directions—probabilistic and stochastic extensions, inverse planning, and Hamilton–Jacobi–Bellman formulations for task comple-

tion in crowd navigation.

- **Appendices**

  We provide complete implementation details in three appendices. Appendix A details the core, reusable architectural modules, including the causal convolutions and decoder vector fields. Appendix B presents the full, end-to-end encoder functions for all models and one-to-one correspondence between the theoretical components of the universality theorems and the specific neural network modules. Finally, Appendix C provides the JAX code for the final, boundary-aware loss functions.

Each chapter builds upon the last, weaving rigorous theory with practical architecture to deliver a unified but modular, boundary-aware, continuous-time framework for interactive trajectory prediction.

Chapter 2: Related Work in Trajectory Prediction

Trajectory prediction sits at the intersection of sequential modeling, multi-agent interaction, and continuous-time dynamics. This chapter surveys the foundational and contemporary techniques that have advanced the field, assessing each against our three guiding principles—boundary awareness, causal-pointwise equivariance, and continuous interactive dynamics. We start in §2.1 with classical recurrent networks (RNNs [15], LSTMs [16], GRUs [17]) that established the basics of sequence learning. In §2.2, we explore convolutional and attention-based approaches (TCNs [18] and Transformers [19]) that boost long-range dependency capture while respecting causality. Section 2.3 examines explicit multi-agent interaction models—graph-based message passing [20] and set-equivariant architectures [21]—that ensure permutation symmetry over agents. We then turn (§2.4) to modern probabilistic and stochastic discrete-time frameworks (VAEs [22], GANs [23], diffusion models [24, 25]) before moving (§2.5) into continuous-time forecasting via Neural ODEs [13], CDEs [26], and SDEs [27, 28]. Finally, §2.6 reviews the statistical treatments [29] of censoring and truncation, thereby laying the theoretical groundwork for §2.7, which surveys practical missing-data and boundary-handling techniques in trajectory prediction and computer vision and identifies gaps our unified, continuous-time point-to-point set-evolution approach will address.

2.1 Recurrent Neural Network Models

Recurrent neural networks (RNNs) form the backbone of many early trajectory-prediction methods by maintaining a hidden state that evolves as new observations arrive. We briefly review three canonical architectures: the vanilla RNN, the Long Short-Term Memory (LSTM) network, and the Gated Recurrent Unit (GRU).

2.1.1 Vanilla RNN

At each timestep t, a vanilla RNN [15] updates its hidden state $h_t \in \mathbb{R}^H$ from the previous state h_{t-1} and the current input $x_t \in \mathbb{R}^D$ via

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b),$$

where σ is an elementwise nonlinearity (e.g. $\tanh$ or ReLU), and $W_h \in \mathbb{R}^{H\times H}$, $W_x \in \mathbb{R}^{H\times D}$, and $b \in \mathbb{R}^H$ are learned parameters. Despite its simplicity, the vanilla RNN suffers from vanishing or exploding gradients over long sequences, limiting its ability to capture extended temporal dependencies.

2.1.2 Long Short-Term Memory (LSTM)

LSTMs [16] mitigate these issues by introducing a memory cell $c_t \in \mathbb{R}^H$ and gating mechanisms that control information flow. Defining concatenated inputs $[h_{t-1}, x_t]$, the LSTM cell

computes

$$i_t = \sigma\big(W_i[h_{t-1},\, x_t] + b_i\big),$$

$$f_t = \sigma\big(W_f[h_{t-1},\, x_t] + b_f\big),$$

$$o_t = \sigma\big(W_o[h_{t-1},\, x_t] + b_o\big),$$

$$\tilde{c}_t = \tanh\big(W_c[h_{t-1},\, x_t] + b_c\big),$$

then updates

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t).$$

Here, i_t, f_t, o_t are the input, forget, and output gates, respectively, and $\odot$ denotes elementwise multiplication. By preserving gradients through the cell state c_t, LSTMs can learn long-range dependencies more effectively.

2.1.3 Gated Recurrent Unit (GRU)

GRUs [17] simplify the LSTM by merging the forget and input gates and eliminating the separate cell state. A GRU defines

$$z_t = \sigma\big(W_z[h_{t-1},\, x_t] + b_z\big), \quad r_t = \sigma\big(W_r[h_{t-1},\, x_t] + b_r\big), \quad \tilde{h}_t = \tanh\big(W_h[\, r_t \odot h_{t-1},\, x_t] + b_h\big).$$

and updates the hidden state via

$$h_t = (1 - z_t) \;\odot\; h_{t-1} \;+\; z_t \;\odot\; \tilde{h}_t.$$

The update gate z_t controls the interpolation between the previous state and the candidate activation $\tilde{h}_t$, while the reset gate r_t modulates how much past information to forget.

2.1.4 Application to Trajectory Prediction

These RNN variants (e.g. Social LSTM [30], SR-LSTM [31]) are often extended with neighbor-pooling or attention to capture inter-agent influences. In effect, each recurrent update acts as a basic discrete-time integrator of continuous dynamics—but it still hinges on a fixed step size (or requires bespoke time-embedding hacks) and must be "hand-tuned" to handle irregular or asynchronous observations. Moreover, while you can graft on permutation-equivariant pooling, vanilla RNNs do not intrinsically enforce causal $\mathbb{R}$-equivariance or pointwise $\mathfrak{S}_n$-equivariance. In contrast, our continuous-time point-to-point set-evolution architecture builds both symmetries into the very definition of its flow, and naturally accommodates arbitrary timestamps without extra modifications.

2.2 Temporal Convolutional and Attention-Based Models

Modern trajectory predictors often build on two complementary paradigms for handling sequential data in discrete time: Temporal Convolutional Networks (TCNs) and attention-based Transformers. Both seek to capture long-range dependencies more effectively than vanilla RNNs, but they differ fundamentally in mechanism and in how they respect causality and permutation symmetry.

2.2.1 Temporal Convolutional Networks (TCNs)

Temporal Convolutional Networks (TCNs) trace back to the causal, dilated-convolution architecture of WaveNet [32], whose effectiveness was thoroughly benchmarked by [18], and later popularized for long-range sequence modeling in action segmentation by [33]. A TCN

replaces recurrence with causal, dilated convolutions: for an input sequence $x_1, \ldots, x_T$, a single convolutional layer computes

$$h_t = \sum_{k=0}^{K-1} W^{(k)}\, x_{t-d\cdot k} + b,$$

where K is the kernel size, d the dilation factor, and the causality constraint $t - d(K-1) \geq 1$ ensures h_t depends only on past inputs. By stacking layers with exponentially increasing dilations $d = 1, 2, 4, \ldots$, TCNs achieve an exponentially large receptive field with only $O(\log T)$ layers. In trajectory forecasting, TCNs have been used to model per-agent motion histories before pooling spatially; they naturally enforce causal time-homogeneity but must rely on separate set- or graph-based pooling to capture agent interactions.

2.2.2 Attention Mechanism and Transformer

The attention mechanism first appeared in machine translation with [34], where an RNN decoder at each step computed a weighted sum of encoder hidden states based on a learned alignment score . This additive attention allowed the model to focus on relevant past tokens dynamically, alleviating the fixed-window limitation of RNNs.

Vaswani et al. (2017) [19] then introduced self-attention within the Transformer architecture, dispensing with recurrence entirely . Given an input token sequence $X \in \mathbb{R}^{T\times d}$, they compute three projections

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

and form

$$\text{Attention}(X) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where $W_Q, W_K \in \mathbb{R}^{d\times d_k}$ and $W_V \in \mathbb{R}^{d\times d_v}$, multi-head attention repeats this in parallel and concatenates the results, enabling the model to attend to different subspaces of the inputs simultaneously.

2.2.3 Transformers for Trajectory Prediction

In the trajectory domain, Transformers have been adapted to handle both temporal and spatial dimensions:

1. **Temporal (causal) attention** treats each agent's history as a token sequence. To inject ordering information, positional encodings p_t (sinusoidal or learned) are added to each token before projection:

   $$X = \big[x_1 + p_1;\ \ldots;\ x_T + p_T\big].$$

   A causal mask then zeros out attention weights for any key index $t' > t$, ensuring no future leakage and preserving time-homogeneity with causality.

2. **Spatial (interaction) attention** views the set of agent embeddings $\{h_i^t\}_{i=1}^n$ at a fixed time t as an unordered set of tokens. Here, no positional encoding is used—so the attention mechanism is naturally permutation-equivariant over agents. Each agent's embedding attends to all others, aggregating peer influences in an order-agnostic way.

Many recent works (e.g. AgentFormer [35], Social Transformer [36]) interleave causal self-attention over time with permutation-equivariant spatial attention at each timestep. These

architectures can be viewed as discrete-time integrators of underlying continuous dynamics; they enforce causality via manually constructed attention masks and handle temporal and spatial relationships using positional encodings, but they operate on fixed sampling grids and depend on explicit discretization choices.

2.2.4 Trade-Offs and Gaps

- TCNs guarantee strict causality by design, with efficient parallel convolution, but their fixed dilations may miss irregular or asynchronous patterns unless heavily tuned.
- Transformers flexibly learn attention patterns over arbitrary ranges, yet require positional encodings and masks to encode causality, and must explicitly omit spatial encodings to maintain permutation symmetry.

Importantly, while discrete-time architectures (e.g. attention and convolutional filters) have proven hugely effective, they lack a principled continuous-time interpretation. No existing work unifies causal, permutation-equivariant attention-style interactions and smooth set-evolution into a single neural ODE framework. Our approach fills this gap by embedding causal temporal filtering and set-equivariant pooling directly into a continuous-time decoder, yielding a truly continuous interactive dynamics model that respects both temporal and spatial symmetries.

2.3 Spatial Interaction Models

In multi-agent forecasting, capturing how each agent influences and is influenced by its peers is crucial. Two complementary paradigms have emerged: explicitly structuring interactions via graphs and, more abstractly, enforcing permutation symmetry over unordered sets.

2.3.1 Graph-Based and Message-Passing Networks

Graph-based methods [20] equip each agent with edges to a selected set of neighbors, then propagate information along these edges via learned message-passing. At time t, a graph $G_t = (V_t, E_t)$ encodes which pairs (j, i) interact. Each node i holds a hidden state h_t^i and its update proceeds in two steps:

$$m_t^i = \mathrm{AGG}\big(\{\phi(h_t^i,\, h_t^j)\}_{\,j:(j,i)\in E_t}\big), \quad h_{t+1}^i = \psi\big(h_t^i,\, m_t^i\big),$$

where ϕ(message function) and ψ(update function) are learned, and AGG is a permutation-invariant aggregator (e.g. sum or max). This Message-Passing Neural Network (MPNN) paradigm [37] underlies many modern multi-agent forecasters.

Social-STGCNN [38] builds a spatio-temporal graph over pedestrian frames, applying spectral graph convolutions in space and causal convolutions in time. Trajectron++ [39] infers a dynamic graph via proximity, aggregates messages with attention weights, and models multimodal futures through a Conditional VAE (CVAE). While powerful, these approaches require constant graph management as agents enter or exit, enforce only local equivariance, and remain tied to fixed, discrete time steps.

2.3.2 Set-Based Equivariant Architectures

Set-based models bypass explicit edges by treating the entire agent collection as an unordered set and using symmetric pooling to ensure permutation invariance:

- Deep Sets [21] embed each element with a shared function and then sum (or max) over the

set.

- PointNet [40] applies a per-point network followed by coordinate-wise max-pooling.
- Set Transformers [41] extend self-attention to sets by omitting positional encodings, using learned attention weights that depend only on pairwise similarities.

These static architectures guarantee that relabeling agents does not change the aggregated representation, laying the foundations for agent anonymity in dynamic settings.

2.3.3 Unified Dynamic Set-Evolution

Our continuous-time point-to-point set-evolution framework not only subsumes fixed-fan-in graph models—by appending a learned dummy agent and using a single max-pool to recover any top-b neighbor aggregation—but also elevates static set architectures like Deep Sets [21] and PointNet [40] into a truly dynamic setting. Rather than computing a one-shot summary of a fixed set, our dynamic context decoder applies the learned interaction function at every instant to the evolving agent set and integrates the resulting vector field via an ODE, ensuring both causal time-homogeneity and smooth, continuous trajectories. This unified approach eliminates discrete edge management, seamlessly accommodates agents entering and exiting, and embeds permutation equivariance directly into the continuous dynamics—yielding a globally equivariant, boundary-aware forecasting model.

2.4 Probabilistic and Stochastic Discrete-Time Models

While §§2.1–2.3 covered deterministic sequence and interaction encoders, real-world systems often demand uncertainty quantification and multimodal forecasts. Historically, this was

addressed by state-space estimators—(extended) Kalman filters and particle filters [42]—which we defer to §4.1 for full treatment. In modern practice, deep generative frameworks—Variational Autoencoders [22], Generative Adversarial Networks [23], and diffusion processes [24, 25]—are wrapped around those same RNN/TCN/Transformer and pooling backbones to produce rich distributions over future trajectories. We now turn to each in turn, starting with VAEs (and Trajectron++ [39]), then GANs (and Social GAN [43]), and finally diffusion-based samplers.

2.4.1 Variational Autoencoders (VAEs) and Trajectron++

VAEs [22] introduce a latent variable z and maximize the Evidence Lower Bound

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi\left(z|X_{1:T},\, X_{T+1:T+F}\right)}\Big[\log p_\theta\big(X_{T+1:T+F} \mid z,\, X_{1:T}\big)\Big] - \text{KL}\Big(q_\phi\big(z \mid X_{1:T},\, X_{T+1:T+F}\big) \,\Big\|\, p(z)\Big).$$

In Trajectron++ [39], a proximity-based graph-RNN encoder processes past positions $X_{1:T}$ to produce Gaussian parameters (μ, σ). Sampling $z \sim \mathcal{N}(\mu, \text{diag}(\sigma^2))$, a decoder RNN then predicts each future step via

$$x_{t+1}^{(i)} \;=\; f_\theta\big(x_t^{(i)},\, z,\, \text{MPNN}(X_t)\big).$$

where MPNN aggregates neighbor messages with gated graph convolutions. Trajectron++ yields diverse, likelihood-trained futures but remains tied to discrete timesteps and explicitly constructs graphs to enforce permutation equivariance.

2.4.2 Generative Adversarial Networks (GANs), WGANs & Social GAN

GANs [23] learn a generator G and discriminator D via the minimax objective

$$\min_G \max_D \mathbb{E}_{X\sim p_{\text{data}}}\big[\log D(X)\big] + \mathbb{E}_{z\sim p(z)}\big[\log\big(1 - D(G(z))\big)\big].$$

However, this formulation can suffer from mode collapse and unstable gradients. Wasserstein GANs (WGANs) [44] address these issues by replacing the Jensen–Shannon divergence with the 1-Wasserstein distance. A WGAN solves

$$\min_G \max_{\substack{D\in\mathcal{D}\\ \|D\|_L\le 1}} \mathbb{E}_{X\sim p_{\text{data}}}\big[D(X)\big] - \mathbb{E}_{z\sim p(z)}\big[D(G(z))\big].$$

where the "critic" D is constrained to be 1-Lipschitz (often via weight clipping or gradient penalties), yielding smoother, more meaningful training signals.

Social GAN [43] adapts these ideas to trajectory forecasting. It first encodes past positions $X_{1:T}$ with an LSTM into per-agent hidden state h_T^i. A noise vector $z \sim \mathcal{N}(0, I)$ is concatenated to each h_T^i, , and the generator $G(h_T^i, z)$ produces the predicted future step. To capture peer interactions in a permutation-invariant way, Social GAN employs a pooling step:

$$m^i = \max_{j\neq i} \text{MLP}\big([\,h_T^i,\, h_T^j\,]\big),$$

which aggregates information from all other agents before decoding. Social GAN trains G and D with the standard cross-entropy GAN loss, but one can readily substitute a WGAN critic in place of D to improve stability.

2.4.3 Diffusion Models

Denoising diffusion probabilistic models (DDPMs) [24, 25] define a forward noising chain

$$q\big(x_t \mid x_{t-1}\big) = \mathcal{N}\Big(x_t \mid \sqrt{1-\beta_t}\, x_{t-1},\ \beta_t I\Big).$$

and learn a score network ϵ_θ to reverse it. Conditional trajectory diffusion pipelines first encode $X_{1:T}$ (often via a Transformer) and then run

$$x_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\Big(x_t \ - \ \beta_t\, \epsilon_\theta\big(x_t,\, t,\, \mathrm{enc}(X_{1:T})\big)\Big).$$

for $t = T+F, \ldots, 1$. Although diffusion excels in sample quality, its tens–hundreds of denoising steps render inference too slow for real-time forecasting; any equivariance or continuous-time consistency must likewise be grafted on separately.

Each of these discrete-time stochastic frameworks advances uncertainty and diversity, but they all operate on fixed sampling grids and treat interaction equivariance as an external plugin. In §2.5 we will survey continuous-time dynamic backbones—Neural ODEs, CDEs, and SDEs—that supply the smooth, causal evolution component, which must then be combined with causally-pointwise-equivariant interaction encoders and probabilistic generation modules to realize a fully uncertainty-aware, interactive, continuous-time forecasting model.

2.5 Continuous-Time Forecasting

Discrete-time models require rigid timesteps and struggle with irregular sampling or occlusion gaps. Continuous-time frameworks instead learn a smooth, causal evolution of hidden states (and optionally stochasticity) that can be queried at any time. We survey three pillars [45]—Neural ODEs [13], Neural CDEs [26], and Neural SDEs [27, 28].—highlighting how each supplies the smooth evolution component as a component of our model architecture along with permutation-equivariant encoders and boundary-aware losses to form a complete trajectory predictor.

2.5.1 Neural ODEs and Neural CDEs

Neural ODEs [13] posit that the hidden state $h(t) \in \mathbb{R}^H$ follows an ODE:

$$\frac{d}{dt}h(t) \;=\; f_\theta\big(t,\, h(t)\big), \qquad h(t_0) \;=\; h_0.$$

and use a black-box ODE solver to compute at arbitrary times. For trajectory forecasting, one typically encodes past data into $h(t_0)$ (via an RNN or temporal Transformer), then solves forward to predict $x(t)$. Neural ODEs naturally handle irregular timestamps and yield arbitrarily fine-grained predictions.

Neural CDEs [26] generalize this to controlled differential equations driven by a continuous interpolation $X(t)$ of observed data:

$$dh(t) = f_\theta\big(t,\, h(t)\big)\, dX(t), \qquad h(t_0) = h_0.$$

Here, dX replaces the explicit time argument, letting the model flexibly respond to the pathwise increments of X. Neural CDEs excel when inputs arrive at irregular times or with missing data, since one can build $X(t)$ via natural cubic splines or other continuous lifts of the discrete samples.

2.5.2 Neural SDEs

To model intrinsic randomness, Neural SDEs [27, 28] endow the hidden state with both a learned drift and diffusion:

$$dh(t) \;=\; f_\theta\big(t,\, h(t)\big)\, dt \;+\; g_\phi\big(t,\, h(t)\big)\, dW_t,$$

where W_t is Brownian motion. This permits stochastic sample paths and quantifies aleatoric uncertainty directly in continuous time. Kidger et al. (2021) [27] show that training such Neural SDEs with a Wasserstein-GAN loss is equivalent to learning an infinite-dimensional GAN over path spaces, with the SDE flow as the generator and a functional critic distinguishing real from generated trajectories.

A small but growing number of forecasting studies have incorporated these backbones. Liang et al. (2021) [46] embed a Neural ODE within a VAE to denoise irregular GPS traces but lack multi-agent interaction, while [5] propose Social ODE, which augments the ODE vector field with learned, distance-weighted pairwise influences to produce continuous multi-agent trajectories.

2.6 Censoring and Truncation in Statistical Practice

In survival analysis [29] and related fields, incomplete observation of event times is classified into censoring and truncation, each with several subtypes that depend on how and when data become unobservable.

Definition 2.6.1 (Censoring)**.** *An event time T is* censored *when we do not see its exact value, but only that it lies beyond some bound. We distinguish:*

- ***Right-censoring.*** *We only know $T > C$, so we record*

$$\widetilde{T} = \min(T, C), \quad \delta = \mathbf{1}_{\{T \le C\}}.$$

  Equivalently: the event either happens by time C (and we observe T) or later (and we only know it exceeded C).

- ***Left-censoring.*** *We only know $T \le C$, so we record*

$$\widetilde{T} = \max(T, C), \quad \delta = \mathbf{1}_{\{T \ge C\}}.$$

  Equivalently: the event either happens before C (and we only know it occurred prior to C) or beyond C (and we observe T).

- ***Interval-censoring.*** *We only know that $L < T \le U$, with neither exact time observed.*

Statistical estimators accommodate these by modifying the likelihood or risk sets: the Kaplan–Meier estimator [29] handles right-censoring via product-limit formulas, while the Turnbull

estimator [47] generalizes it to arbitrary interval-censoring. Hazard-function methods (e.g. Cox regression [48]) further model how covariates influence the instantaneous event rate under right-censoring.

Truncation refers to selective inclusion of subjects based on their event times relative to a window; truncated observations never appear in the dataset if their event falls outside:

Definition 2.6.2 (Truncation)**.** *An event time T is* truncated *when subjects whose events fall outside a given window are never observed at all. The main cases are:*

- ***Left-truncation.*** *Only those with $T > A$ enter the dataset ("late entrants").*
- ***Right-truncation.*** *Only those with $T \leq B$ appear.*
- ***Double-truncation.*** *Only those with $A < T \leq B$ are included.*

Truncation requires adjusting the risk sets or likelihoods to account for the unobserved segment of the population. Methods like conditional likelihoods and nonparametric maximum likelihood estimators properly weight truncated observations to avoid bias.

In practice, these concepts often co-occur—e.g. in cohort studies where entry times censor early life and study-end censors late events—leading to mixed censoring/truncation patterns. Standard software (e.g. `R`'s `survival` package [49]) supports right-censoring with left-truncation, but fully general interval-censoring plus truncation requires specialized algorithms.

Although the statistical theory of censoring and truncation is mature, its application in interactive trajectory prediction has been minimal. In our work, we draw on these classical ideas to inform a survival-analysis–inspired exit-consistency penalty (right-truncation). By explicitly modeling these effects in the training objective, we ensure that our continuous-time set-evolution framework respects the same rigor as established statistical practice.

2.7 Missing-Data in Trajectory Prediction

Real-world trajectory datasets frequently contain missing observations due to various factors, including occlusions and boundary constraints inherent in fixed-sensor setups. This section reviews existing approaches to handling missing trajectory data, distinguishing clearly between internal gaps (occlusions) and boundary-related gaps (censoring and truncation).

2.7.1 Overview: Types of Missing Data in Trajectories

Missing data in trajectory prediction generally fall into two broad categories:

- Occlusion (internal gaps) [1]: Agents remain within the sensor's field of view (FOV) but are temporarily unobserved due to objects or structures obstructing the line of sight.
- Boundary-related gaps (censoring and truncation): These occur when trajectories begin before or end after the observation period (temporal censoring), or when agents cross spatial boundaries of the sensor's coverage area during observation (spatial truncation).

2.7.2 Imputation-Based Approaches (Mainstream Methods)

Common approaches to address short internal gaps rely primarily on imputation techniques:

- Interpolation methods (linear, cubic splines): These straightforward approaches fill short gaps by estimating missing points based on observed data immediately before and after the gap [50]. While simple, they are only effective for very brief, predictable gaps.

- Learning-based imputation: Recent deep learning methods, such as graph-based variational autoencoders (VAEs) or recurrent neural networks, learn to fill missing segments by leveraging statistical patterns in observed data [39, 51]. These methods have been successful in handling internal, short-duration occlusions.

2.7.3 Limitations of Imputation for Boundary Cases

Imputation techniques assume the missing segments are short, bounded, and predictable from surrounding observations. However, boundary-related gaps, especially due to agents exiting the sensor region, inherently involve uncertainty about future agent behavior beyond sensor boundaries. Thus, traditional imputation is unsuitable for predicting agent trajectories in such cases, as future states cannot reliably be inferred from past data alone. While vision-based occlusion inference uses rich visual cues to reconstruct hidden agent positions, in this survey we concentrate on coordinate-level imputation—the task of filling missing (x, y) points once raw trajectories have been extracted.

2.7.4 Modeling Boundary Events: An Underexplored Area

Surprisingly, systematic approaches to explicitly predict boundary-crossing events—such as exact exit times or spatial locations at boundaries—remain underdeveloped in the trajectory prediction literature. Most current approaches either discard truncated trajectories, pad them with default values, or rely on heuristic penalties. There is a notable absence of principled, survival-analysis–inspired methodologies for modeling and predicting these boundary events, highlighting a clear research gap.

Table 2.1: Summary of missing-data regimes and existing approaches

Missing-data Type	**Common Approaches**	**Limitations**
Occlusion (internal)	Interpolation; latent-variable models	Effective only for short, internal gaps
Boundary (truncation & censoring)	Discarding; padding with heuristic methods	Lack systematic boundary-event modeling

In contrast to imputation-centric approaches, our boundary-aware, continuous-time set-evolution model directly incorporates observed exit times into a survival-analysis-inspired loss. By extracting true exit times from the data and imposing a spatial linger penalty, we provide a principled method for enforcing correct agent exit behavior—without regressing exit times as explicit outputs or relying on imputation.

Chapter 3: Spatial Representation & Permutation Equivariance

This chapter develops the mathematical foundations for learning on sets of points, providing the tools that underpin our main framework for trajectory forecasting. We begin by contrasting two canonical geometries for comparing collections of agents: the Hausdorff metric on compact sets, which tracks worst-case deviations, versus the Wasserstein metric on probability measures, which quantifies average transport cost.

After surveying extensions to unbalanced transport—which handle agents entering or exiting a scene—we use them to motivate our simpler, pure set-based approach. We then build the theoretical justification for this choice by developing the machinery of idempotent measures and support functionals. This framework provides a deep connection between the max-pooling operation common in neural networks and a parallel, idempotent version of the classical Riesz representation theorem.

Finally, we review the foundational Stone–Weierstrass and Prolla approximation theorems. These results guarantee that the architectures we consider are sufficiently expressive, setting the stage for the universal approximation proofs that form the chapter's conclusion.

3.1 Hausdorff and Wasserstein Distances

Let (X, d) be a locally compact Polish space—that is, a separable and complete metric space which is also locally compact. We write $\mathcal{K}(X)$ for the set of all nonempty compact subsets of X.

Definition 3.1.1 (Hausdorff distance)**.** *For $A, B \in \mathcal{K}(X)$, the* Hausdorff distance *between A and B is*

$$d_H(A, B) \;=\; \max\Big\{ \sup_{x\in A} \inf_{y\in B} d(x, y),\ \sup_{y\in B} \inf_{x\in A} d(x, y) \Big\}.$$

Equivalently,

$$d_H(A, B) = \inf\big\{\varepsilon \geq 0 : A \subseteq B_\varepsilon \text{ and } B \subseteq A_\varepsilon\big\},$$

where $A_\varepsilon = \{x \in X : d(x, A) \leq \varepsilon\}$ and $B_\varepsilon = \{x \in X : d(x, B) \leq \varepsilon\}$.

Proposition 3.1.2 (Worst-case transport)**.** *One may equally characterize d_H as the minimal* worst-case *matching cost:*

$$d_H(A, B) = \inf_R \sup_{(x,y)\in R} d(x, y),$$

where the infimum is taken over all closed relations R whose projections onto A and B both cover the entire set.

Proof. See [52], Proposition 2.1. □

Theorem 3.1.3 (Lipschitz-dual for Hausdorff)**.** *One has the equivalent dual formula*

$$d_H(A, B) = \sup\Big\{ \max_{x\in A} f(x) \;-\; \max_{y\in B} f(y) \;:\; f : X \to \mathbb{R} \text{ is 1-Lipschitz} \Big\}.$$

Proof. See [53], Proposition 6.2. □

Proposition 3.1.4 (Topological Properties of $\mathcal{K}(X)$)**.** *Let X be a metric space. The following properties hold:*

1. ***Completeness:*** *If X is a complete metric space, then $\mathcal{K}(X)$ is also a complete metric space.*
2. ***Separability:*** *If X is a separable metric space, then $\mathcal{K}(X)$ is also a separable metric space.*
3. ***Polish Property:*** *If X is Polish then $\mathcal{K}(X)$ is also a Polish space.*
4. ***Local Compactness:*** *If X is locally compact, then $\mathcal{K}(X)$ is also locally compact.*
5. ***Compactness:*** *$\mathcal{K}(X)$ is compact if and only if X is compact.*

Proof. See [54], Appendix D. □

Next, we recall the p–Wasserstein distances on probability measures, which measure the *average* transport cost.

Definition 3.1.5 (Wasserstein distance)**.** *Let $\mathcal{P}(X)$ be the set of Borel probability measures on X. For $1 \leq p < \infty$ and $\mu, \nu \in \mathcal{P}(X)$, the p–Wasserstein distance is*

$$W_p(\mu, \nu) \;=\; \Big(\inf_{\pi \in \Pi(\mu,\nu)} \int_{X \times X} d(x,y)^p \, d\pi(x,y) \Big)^{1/p},$$

where $\Pi(\mu, \nu)$ is the set of all couplings (transport plans) with marginals μ, ν.

The space $\mathcal{P}_p(X)$ is defined to be the subset of probability measures $\mu \in \mathcal{P}(X)$ that have a finite p-th moment. Formally, this is stated as

$$\mathcal{P}_p(X) \;:=\; \Big\{ \mu \in \mathcal{P}(X) \;\Big|\; \exists\, x_* \in X \text{ s.t. } \int_X d(x, x_*)^p \, \mathrm{d}\mu(x) < +\infty \Big\}.$$

Theorem 3.1.6 (Kantorovich duality, $p = 1$)**.** *When $p = 1$, there is the dual representation*

$$W_1(\mu, \nu) = \sup\Big\{ \mathbb{E}_{x\sim\mu}[f(x)] - \mathbb{E}_{y\sim\nu}[f(y)] : f : X \to \mathbb{R},\ \|f\|_{\mathrm{Lip}} \leq 1 \Big\}.$$

Proof. See [12], Theorem 1.14 (Kantorovich–Rubinstein theorem). □

Proposition 3.1.7 (Topological Properties of $\mathcal{P}_p(X)$)**.** *Let X be a metric space. The following properties hold:*

1. ***Completeness:*** *If X is a complete metric space, then $\mathcal{P}_p(X)$ is also a complete metric space.*

2. ***Separability:*** *If X is a separable metric space, then $\mathcal{P}_p(X)$ is also a separable metric space.*

3. ***Polish Property:*** *If X is Polish then $\mathcal{P}_p(X)$ is also a Polish space.*

4. ***Local Compactness:*** *$\mathcal{P}_p(X)$ is locally compact only if X is compact.*

5. ***Compactness:*** *$\mathcal{P}_p(X)$ is compact if and only if X is compact.*

Proof. See [55], §7.1. □

Remark 3.1.8.

- *d_H and W_p induce very different topologies on the space of compact subsets vs. probability measures. In particular, d_H is a* worst-case *(infinite-norm) metric, while W_p is an* average *(finite-norm) metric.*

- *When one interprets a finite set $A \subset X$ as the empirical measure $\frac{1}{n}\sum_{i=1}^{n}\delta_{a_i}$, the Hausdorff and Wasserstein distances coincide only in trivial cases; typically $d_H(A, B) \leq W_\infty(A, B)$ and $W_p(A, B) \leq W_\infty(A, B)$ where $W_\infty(A, B) = \lim_{p\to\infty} W_p(A, B)$ [52].*

3.2 Unbalanced Optimal-Transport Metrics and the Reservoir-Augmented Wasserstein Distance

In many real-world settings—population flows, evolving point clouds, or multi-agent scenes—total mass is not conserved due to confined observational scope: agents enter or exit, point clouds grow or shrink, and overall density can vary. Classical p-Wasserstein distances $W_p(\mu, \nu)$ on probability measures enforce exact mass conservation, requiring ad-hoc renormalization or reweighting whenever total mass changes. To address this, the optimal-transport community has developed a host of unbalanced optimal transport metrics—Kantorovich's original throwing-mass-away formulation [56], dynamic reservoir- or boundary-penalty models [57–59], and entropy-regularized optimal transport [60]—that jointly penalize both transport and mass creation/destruction.

We now introduce the simplest such construction, the reservoir-augmented p-Wasserstein distance W_p^b on $\mathcal{M}_+(X)$, which appends a single reservoir point at fixed per-unit cost b. Even in this elementary form, W_p^b exposes the persistent burden of weight assignment and bookkeeping that any measure-theoretic trajectory model must shoulder; by contrast, our set-based (max-pooling) approach sidesteps these complications entirely.

Definition 3.2.1 (Augmented Space and Reservoir Point)**.** *Let* (X, d) *be a compact metric space with* $\operatorname{diam}(X) < \infty$. *Fix a penalty parameter* $b > \operatorname{diam}(X)^p$. *Define the* augmented space

$$\tilde{X} \;=\; X \sqcup \{\infty\},$$

equipped with the extended metric

$$\tilde{d}(x,y) = \begin{cases} d(x,y), & x,y \in X, \\ b^{1/p}, & \textit{if exactly one of } x,y \textit{ equals } \infty, \\ 0, & x = y = \infty. \end{cases}$$

The point ∞ serves as a reservoir *that can absorb or emit arbitrary mass at constant unit cost $b^{1/p}$.*

Definition 3.2.2 (Reservoir-augmented p-Wasserstein Distance)**.** *For two finite positive Radon measures $\mu, \nu \in \mathcal{M}_+(X)$, choose any extensions*

$$\tilde{\mu} = \mu + r\,\delta_\infty, \quad \tilde{\nu} = \nu + s\,\delta_\infty \quad \textit{on } \tilde{X}$$

so that $\|\tilde{\mu}\| = \|\tilde{\nu}\|$. Their reservoir-augmented *distance is*

$$W_p^b(\mu,\nu) \;=\; W_p\big(\tilde{\mu},\tilde{\nu}\big),$$

the classical p-Wasserstein metric on $\tilde{X}$. One verifies this is independent of the particular $r, s \geq 0$ chosen.

Insisting that the reservoir penalty $b^{1/p}$ exceed the diameter of the domain X has an immediate, intuitive meaning: no matter how you rearrange mass within the scene, it will never be cheaper to send that mass out to—or pull it in from—the reservoir. Thus any appearance or disappearance of an agent carries a strictly higher cost than any smooth, within-scene transport.

(Indeed, exactly the same trick can be applied to the Hausdorff metric—adjoining a reservoir point ∞ at distance b from every location—which underlines that treating entrances and exits as high-cost shocks is a general device, not unique to Wasserstein-style formulations.)

All of these unbalanced or reservoir-augmented constructions—from Kantorovich's early throwing-mass-away idea through modern Fisher–Rao and dynamic formulations—share one unavoidable feature: you must explicitly track each agent's weight as it appears, moves, and disappears. Every time the population changes, you must reassign or re-normalize total mass, update couplings, and ensure the metrics remain well-defined. By contrast, in our pure set-based viewpoint there is no weight to carry: agents simply enter or exit the support, and max-based pooling automatically accommodates such events with zero bookkeeping overhead.

Proposition 3.2.3 (Well-definedness of the reservoir-augmented distance)**.** *Let $\mu, \nu \in \mathcal{M}_+(X)$ be two finite measures on a bounded metric space (X, d), and let $b > 0$ exceed $\operatorname{diam}(X)$. For any choices of $r, s \geq 0$ such that*

$$\mu(X) + r \;=\; \nu(X) + s,$$

define the extended measures

$$\tilde{\mu} \;=\; \mu + r\,\delta_\infty, \qquad \tilde{\nu} \;=\; \nu + s\,\delta_\infty,$$

on the augmented space $\tilde{X} = X \sqcup \{\infty\}$. Then

$$W_p^b(\mu, \nu) \;\equiv\; W_p\big(\tilde{\mu}, \tilde{\nu}\big)$$

is independent of the particular r, s satisfying $\mu(X) + r = \nu(X) + s$.

Proof. Suppose (r, s) and (r', s') are two such pairs with

$$\mu(X) + r \;=\; \nu(X) + s \quad \text{and} \quad \mu(X) + r' \;=\; \nu(X) + s'.$$

Then, without loss of generality,$\Delta = r' - r = s' - s \geq 0$. Writing

$$\tilde{\mu}' = \mu + r'\,\delta_\infty \quad \text{and} \quad \tilde{\nu}' = \nu + s'\,\delta_\infty,$$

we can decompose any coupling $\pi \in \Pi(\tilde{\mu}, \tilde{\nu})$ into two parts:

$$\pi = \pi_{\text{off}} + \Delta\,\delta_{(\infty,\infty)},$$

where π_{off} is a transport plan between $\mu + r\,\delta_\infty$ and $\nu + s\,\delta_\infty$ that moves mass only between X and $\{\infty\}$, and the extra mass Δ is matched at (∞, ∞), which incurs zero cost because $d(\infty, \infty) = 0$.

Since adding or removing this diagonal $\Delta\,\delta_{(\infty,\infty)}$ does not change the p-th moment $\int d(\cdot,\cdot)^p\,d\pi$, the infimum defining $W_p\big(\tilde{\mu}, \tilde{\nu}\big)$ is identical to that for $W_p\big(\tilde{\mu}', \tilde{\nu}'\big)$. Hence $W_p^b(\mu, \nu)$ does not depend on the choice of (r, s). □

Theorem 3.2.4 (Metric and Topology)**.** *1. W_p^b defines a bona fide metric on $\mathcal{M}_+(X)$.*

2. Its induced topology coincides with weak convergence plus convergence of total mass.

Although even the simplest reservoir-augmented p-Wasserstein distance lays bare the relentless weight assignment and mass change bookkeeping that any measure-theoretic approach must endure, our pure set-based (max-pooling) framework sidesteps these hassles entirely. It

precisely motivates our pivot to a pure set-based view—where agents are unweighted points, and all comparisons reduce to simple max-pooling over features, with none of the weight assignment overhead.

3.3 Idempotent Measures and Support Functionals

Just as Radon measures and the Riesz theorem underlie Wasserstein-based models, there is a parallel "idempotent" integration theory in which sums become maxima and integrals become suprema. Concretely, one works over a semiring $(\oplus, \odot)$ (for instance, $(\vee, +)$ or $(\vee, \times)$), and any upper-semicontinuous "weight" function ϕ induces a $\oplus$-linear functional

$$f \longmapsto \int_X^{\oplus} f \, d\mu_\phi = \bigoplus_{x \in X} \big(f(x) \odot \phi(x)\big) = \sup_{x \in X} \big(f(x) \odot \phi(x)\big).$$

We will invoke the Kolokoltsov-Maslov [61,62] analogue of Riesz to show that every continuous $\oplus$-linear map arises this way, and then focus on the special case of support functionals

$$F_C(f) = \max_{x \in C} \; f(x)$$

which embed closed sets C into each idempotent semiring. Later, we give two clean three-axiom characterizations—(i) sup-additivity, (ii) translation-equivariance, and either (iii.a) positive homogeneity or (iii.b) constant capping preservation—that pin down exactly these support functionals, explaining why max-pooling alone suffices to captures set geometry.

3.3.1 Idempotent Integration Framework and the Kolokoltsov–Maslov Representation Theorem

The classical Riesz–Markov–Kakutani representation theorem identifies every continuous linear functional on $C_0(X)$ with integration against a unique Radon measure. In idempotent analysis, an entirely parallel result—the Kolokoltsov–Maslov representation theorem—shows that every continuous $\oplus$-linear functional on the max-plus (max-$\times$ or max-min) module $C_0(X;\mathcal{A})$ arises from taking suprema against a unique upper-semicontinuous weight. We present the idempotent version first, then restate Riesz–Markov–Kakutani immediately afterward for direct comparison.

3.3.1.1 Numerical idempotent semirings

We begin by fixing one of three canonical numerical semirings $(\mathcal{A}, \oplus, \odot)$, each carrying a complete metric compatible with the order topology and the algebraic operations:

1. Tropical (max-plus) semiring

$$\mathcal{A}_1 = \mathbb{R} \cup \{-\infty\}, \quad a \oplus b = a \vee b, \quad a \odot b = a + b, \quad 0_{\mathcal{A}} = -\infty, \quad 1_{\mathcal{A}} = 0.$$

2. Max-times semiring

$$\mathcal{A}_2 = \mathbb{R}_{\geq 0}, \quad a \oplus b = a \vee b, \quad a \odot b = a \times b, \quad 0_{\mathcal{A}} = 0, \quad 1_{\mathcal{A}} = 1.$$

3. Max-min semiring

$$\mathcal{A}_3 = \mathbb{R} \cup \{\pm\infty\}, \quad a \oplus b = a \vee b, \quad a \odot b = a \wedge b, \quad 0_{\mathcal{A}} = -\infty, \quad 1_{\mathcal{A}} = \infty.$$

Here, $a \vee b = \max\{a, b\}$ and $a \wedge b = \min\{a, b\}$. Each $\mathcal{A}_i$ is a commutative semiring in which $\oplus$ is idempotent ($a \oplus a = a$), and admits a natural metric ρ_i making the operations continuous:

$$\rho_1(a, b) = |\exp a - \exp b|, \quad \rho_2(a, b) = |a - b|, \quad \rho_3(a, b) = |\arctan a - \arctan b|.$$

3.3.1.2 Test-function space $C_0(X; \mathcal{A})$

Let X be a locally compact Polish space. We define

$$C_0(X; \mathcal{A}) = \big\{ f \colon X \to \mathcal{A} \ : \ f \text{ is continuous,}$$
$$\text{for each } \epsilon > 0, \text{ there exists a compact } K_\epsilon \subset X$$
$$\text{such that } \rho(f(x), 0_{\mathcal{A}}) < \epsilon \text{ for all } x \notin K_\epsilon \big\}.$$

We equip $C_0(X; \mathcal{A})$ with the uniform metric

$$d_\infty(f, g) = \sup_{x \in X} \rho\big(f(x), g(x)\big),$$

and extend the pointwise semiring operations $f \oplus g$ and $a \odot f$ in the obvious way. Thus $C_0(X; \mathcal{A})$ becomes a semimodule over $\mathcal{A}$, analogous to the classical real-valued $C_0(X)$.

3.3.1.3 Idempotent linear functionals

An idempotent linear functional is a continuous map

$$F : C_0(X; \mathcal{A}) \to \mathcal{A}$$

satisfying, for all $f, g \in C_0(X; \mathcal{A})$ and all scalars $a \in \mathcal{A}$,

$$F(f \oplus g) = F(f) \oplus F(g), \quad F(a \odot f) = a \odot F(f).$$

In words, F preserves the tropical "sum" ($\max$) and commutes with scalar multiplication in $\mathcal{A}$.

3.3.1.4 The Kolokoltsov–Maslov representation theorem

The fundamental result of Kolokoltsov and Maslov shows that every such functional F arises by idempotent integration against an upper-semicontinuous weight function $\phi : X \to \mathcal{A}$.

Theorem 3.3.1 (The Kolokoltsov–Maslov representation theorem)**.** *Let X be a locally compact Polish space and $(\mathcal{A}, \oplus, \odot)$ one of the idempotent semirings above. Equip $C_0(X; \mathcal{A})^*$ (the idempotent dual space; continuous $(\oplus, \odot)$-linear functionals) with the operator distance*

$$D_\infty(F_1, F_2) = \sup_{\substack{h \in C_0(X;\mathcal{A}) \\ d_\infty(h, 0_\mathcal{A}) \le 1}} \rho\big(F_1(h), F_2(h)\big).$$

and equip $USC(X; \mathcal{A})$ (the bounded upper-semicontinuous $\mathcal{A}$-valued functions) with the supre-

mum metric

$$d_\infty(\phi, \psi) = \sup_{x \in X} \rho(\phi(x), \psi(x)).$$

Then:

1. *(Representation.) Every continuous $\mathcal{A}$-linear functional*

$$F : C_0(X; \mathcal{A}) \to \mathcal{A}$$

admits a unique weight function $\phi \in USC(X; \mathcal{A})$ so that

$$F(f) = \bigoplus_{x \in X} \big(f(x) \odot \phi(x)\big) = \sup_{x \in X} \big(f(x) \odot \phi(x)\big), \quad \forall f \in C_0(X; \mathcal{A}).$$

2. *(Uniqueness.) The map $F \mapsto \phi$ is bijective between the set of continuous $\mathcal{A}$-linear functionals on $C_0(X; \mathcal{A})$ and $USC(X; \mathcal{A})$.*

3. *(Isometry.) This bijection is in fact an isometry, $C_0(X; \mathcal{A})^* \cong USC(X; \mathcal{A})$: for any two functionals F_1, F_2 with corresponding weights ϕ_1, ϕ_2,*

$$D_\infty(F_1, F_2) = d_\infty(\phi_1, \phi_2).$$

Proof. This theorem is a direct consequence of the isomorphism established in Theorem 1 of [62], particularly points 2, 3, and 4, which describe the properties of the map between functions and functionals, and their correspondence to various continuity properties. □

In particular, when $\mathcal{A} = \mathcal{A}_1$ (max-plus), one has

$$F(f) = \sup_{x \in X} \big(f(x) + \phi(x) \big).$$

3.3.1.5 The Riesz–Markov–Kakutani representation theorem

To highlight the perfect analogy, we now restate the classical duality for real-valued functionals on $C_0(X)$. In its modern form, this is often called the Riesz–Markov–Kakutani theorem.

Theorem 3.3.2 (The Riesz–Markov–Kakutani representation theorem)**.** *Let X be a locally compact Polish space.*

1. *(Representation.) Every continuous linear functional*

$$\Phi : C_0(X) \to \mathbb{R}$$

admits a unique Radon measure μ so that

$$\Phi(f) = \int_X f \, d\mu, \quad \forall f \in C_0(X).$$

2. *(Uniqueness.) The map $\Phi \mapsto \mu$ is bijective between the set of continuous linear functionals on $C_0(X)$ and $\mathcal{M}(X)$.*

3. *(Isometry.) This bijection is in fact an isometry, $C_0(X)^* \cong \mathcal{M}(X)$.*

Proof. See [63], Theorem 7.17 (The Riesz representation theorem). □

Because this theorem identifies every continuous linear functional with integration against

a measure, it provides the exact counterpart to the idempotent result above: there we replaced the sum-integral pairing $\int f \; d\mu$ by the max-supremum pairing $\sup_x \big(f(x) \odot \phi(x)\big)$. In both cases, the dual of a suitably defined continuous function space is captured entirely by a class of weights (Radon measures or upper-semicontinuous kernels).

3.3.1.6 Connection to neural max-pooling

In the idempotent semiring $\mathcal{A}$, the Kolokoltsov–Maslov representation theorem says that any continuous, $\max$-preserving, $\odot$-homogeneous functional on compactly supported functions is nothing but a weighted max-pool. In the special case where the weight ϕ takes only the two values $\{0_{\mathcal{A}}, 1_{\mathcal{A}}\}$, the functional reduces exactly to the support functional of a closed set ($F(f) = \max_{x \in C} f(x)$). Thus, max-pooling layers in set-equivariant neural networks can be viewed as instantiations of idempotent integrals against indicator-type weights.

By contrast, classical integrals against real measures enforce additive linearity and naturally carry the bookkeeping overhead of mass assignment. The idempotent theory sidesteps all of that: one only needs to track the $\odot$-unit($1_{\mathcal{A}}$) contributions of the points that actually achieve the supremum.

This completes our exposition of the idempotent-integration framework and its overarching representation theorem, the direct analogue of Riesz's classical result in the $(\oplus, \odot)$ semirings. In the next subsection, we will specialize further to the case of support functionals and characterize exactly when such an idempotent integral arises from an unweighted closed-set indicator.

3.3.2 Support Functionals and Closed-Set Embedding

In this section we show how every nonempty closed subset $C \subset X$ gives rise to a canonical support functional on $C_0(X; \mathcal{A}_i)$ for $i = 1, 2, 3$. We will see that these support functionals are exactly the continuous, sup-preserving maps on $C_0(X; \mathcal{A}_1)$ which are equivariant under constant shifts and either positively homogeneous or constant-capping. Conversely, any such functional must arise from a unique closed set. When X is compact, one may equivalently work in $C(X)$, avoiding $-\infty$ values altogether. Finally, we explain how the resulting identification $\mathcal{CL}(X) \cong \{F_C\} \subset C_0(X; \mathcal{A}_1)^*$ is algebraically exact (a join-semilattice embedding) but topologically discrete—mirroring the familiar discrepancy between total-variation and Wasserstein topologies.

Definition 3.3.3. *We write*

$$\mathcal{CL}(X) = \{C \subset X : C \neq \varnothing, \quad C \text{ is closed}\}$$

for the hyperspace of nonempty closed subsets of X. We equip $\mathcal{CL}(X)$ with the Vietoris topology*, generated by the subbasis*

$$\{C \in \mathcal{CL}(X) : C \subset U\}, \quad \{C \in \mathcal{CL}(X) : C \cap U \neq \varnothing\}$$

as U ranges over open subsets of X.When X is a locally compact Polish space, the subspace $\mathcal{K}(X)$ of $\mathcal{CL}(X)$ is known to inherit exactly the topology induced by the Hausdorff metric on X [54].

Definition 3.3.4. *For any nonempty closed set $C \subset X$, the* support functional *associated to C is defined to be*

$$F_C(f) : C_0(X; \mathcal{A}) \to \mathcal{A}, \quad F_C(f) = \max_{x \in C} f(x).$$

Here, $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathbb{R}\}$.

One checks at once that each F_C satisfies four key axioms on $C_0(X; \mathcal{A})$, with each axiom understood to apply only when the corresponding semiring operation exists:

- Sup preserving:

$$F_C(f \vee g) = F_C(f) \vee F_C(g), \quad \forall f, g \in C_0(X; \mathcal{A}).$$

- Shift equivariance:

$$F_C(f + c) = F_C(f) + c, \quad \forall f \in C_0(X; \mathcal{A}), \forall c.$$

- Positive homogeneity:

$$F_C(af) = aF_C(f), \quad \forall f \in C_0(X; \mathcal{A}), \forall a \geq 0.$$

- Constant capping:

$$F_C(f \wedge c) = F_C(f) \wedge c, \quad \forall f \in C_0(X; \mathcal{A}), \forall c.$$

Moreover, for each $i = 1, 2, 3$, the support functional F_C admits the unified representation

$$F_C(f) = \sup_{x \in X} \left(f(x) \odot \mathbf{1}_C^{(i)}(x) \right) \quad \text{where} \quad \mathbf{1}_C^{(i)}(x) = \begin{cases} 1_{\mathcal{A}_i}, & x \in C, \\ 0_{\mathcal{A}_i}, & x \notin C. \end{cases}$$

To facilitate the proof of the characterization theorem that follows, we first recall a useful general result from real analysis concerning the interchange of infima and suprema for monotone sequences of functions.

Lemma 3.3.5 (Monotone inf–sup interchange)**.** *Let Y be a Hausdorff compact space, and let $(g_n)_{n \in \mathbb{N}}$ be a pointwise nonincreasing family of upper semicontinuous functions on Y. Then*

$$\inf_n \sup_{y \in Y} g_n(y) \;=\; \sup_{y \in Y} \inf_n g_n(y).$$

Proof. Define, for notational convenience,

$$M_n = \sup_{y \in Y} g_n(y), \quad g(y) = \inf_n g_n(y), \quad M = \sup_{y \in Y} g(y).$$

Since $\{M_n\}_{n \in \mathbb{N}}$ is nonincreasing and bounded below by M, $\inf_n M_n \geq M$.

Suppose, towards a contradiction, that $\inf_n M_n > M$. We may choose $\epsilon > 0$ so that $\inf_n M_n > M + \epsilon$.

Since $\inf_n M_n > M + \epsilon$, for every $n \in \mathbb{N}$, $\sup_{y \in Y} g_n(y) \geq \inf_n M_n > M + \epsilon$.

Note that an upper semicontinuous function on a compact Hausdorff space attains a maximum. Consider a sequence of sets $F_n = \{y \in Y : g_n(y) \geq M + \epsilon\}$.

- F_n is nonempty for all n: Since $\sup_{y \in Y} g_n(y) = \max_{y \in Y} g_n(y) > M + \epsilon$ for each n, there must be at least one y_n such that $g_n(y_n) > M + \epsilon$, so F_n is nonempty.

- F_n is compact for all n: Since g_n is upper semicontinuous, F_n is closed for each n.

- F_n is a nested sequence: This is because $\{g_n\}_{n\in\mathbb{N}}$ is pointwise nonincreasing.

Since F_n is a nested sequence of nonempty compact sets, by the finite intersection property of compact spaces, their intersection is nonempty. So we may pick $y_0 \in \bigcap_{n\in\mathbb{N}} F_n$.

By definition of F_n, $y_0 \in F_n$ for all $n \in \mathbb{N}$, which means $g_n(y_0) \geq M + \epsilon$ for all $n \in \mathbb{N}$. By taking $\inf_n$, we have $g(y_0) \geq M + \epsilon$. However, $g(y) \leq M$ for all $y \in Y$ by definition of M. In particular, $g(y_0) \leq M$. This leads to a contradiction that $M + \epsilon \leq g(y_0) \leq M$ implies $\epsilon \leq 0$.

□

Building upon the definitions of idempotent semirings and support functionals, we now establish a fundamental characterization theorem. This theorem specifies the necessary and sufficient algebraic properties that a continuous functional on $C_0(X; \mathcal{A}_1)$ must satisfy to be a support functional.

Theorem 3.3.6 (Characterization of Support Functionals)**.** *Let X be a locally compact Polish space and $F : C_0(X; \mathcal{A}_1) \to \mathcal{A}_1$ be a continuous functional. The following are equivalent:*

1. ***Support functional.***

   There exists a nonempty closed set $C \subset X$ such that

$$F(f) = \max_{x\in C} f(x) \quad \textit{for all } f \in C_0(X; \mathcal{A}_1)$$

2. ***Shift equivariance and positive homogeneity.***

   The functional F satisfies, for all $f, g \in C_0(X; \mathcal{A}_1)$, all $c \in \mathbb{R}$, and all $a \geq 0$:

(a) $F(f \vee g) = F(f) \vee F(g)$;

(b) $F(f + c) = F(f) + c$;

(c) $F(af) = aF(f)$.

3. ***Shift equivariance and constant capping.***

   The functional F satisfies (a) and (b) above, and in place of (c) the following holds for all $c \in \mathbb{R}$:

$$F(f \wedge c) = F(f) \wedge c.$$

Proof. It suffices to show that statement 2 implies statement 1 and statement 3 implies statement 1.

Since $F : C_0(X; \mathcal{A}_1) \to \mathcal{A}_1$ is a continuous $(\vee, +)$-linear functional, we may choose a unique upper semicontinuous function ϕ so that $F(f) = \sup_{x \in X}(f(x) + \phi(x))$. Fix $x \in X$. We may choose a compact neighborhood K_x of x and we may find a bump function $\eta_x : X \to [0, 1]$ so that $\eta_x(x) = 1$, $\eta_x(y) = 0$ for $y \notin K_x$. Then $\ln \eta_x \in C_0(X; \mathcal{A}_1)$ whose compact support is exactly K_x. Note that the sequence of functionals, $\{n \ln \eta_x\}_{n \in \mathbb{N}}$ is pointwise nonincreasing and converges to $\mathbf{1}^{(1)}_{\{x\}}$.

For 2. $\Rightarrow$ 1., by positive homogeneity, we have for any $a \geq 0$ and each $n \in \mathbb{N}$,

$$\begin{aligned}
& F(an \ln \eta_x) = aF(n \ln \eta_x) \\
\iff & \sup_{y \in X} \big(an \ln \eta_x(y) + \phi(y)\big) = a \sup_{y \in X} \big(n \ln \eta_x(y) + \phi(y)\big) \\
\iff & \max_{y \in K_x} \big(an \ln \eta_x(y) + \phi(y)\big) = a \max_{y \in K_x} \big(n \ln \eta_x(y) + \phi(y)\big).
\end{aligned}$$

For 3. $\Rightarrow$ 1., by constant capping, we have for any $a, b \in \mathbb{R}$ and each $n \in \mathbb{N}$,

$$
\begin{aligned}
& F\big((a + n \ln \eta_x) \wedge b\big) = F(a + n \ln \eta_x) \wedge b \\
\iff & \sup_{y \in X} \big((a + n \ln \eta_x) \wedge b) + \phi(y)\big) = \sup_{y \in X} \big(a + n \ln \eta_x(y) + \phi(y)\big) \wedge b \\
\iff & \max_{y \in K_x} \big((a + n \ln \eta_x) \wedge b) + \phi(y)\big) = \max_{y \in K_x} \big(a + n \ln \eta_x(y) + \phi(y)\big) \wedge b.
\end{aligned}
$$

Lemma 3.3.5 allows us to pass the $\sup$ inside the limit. For 2. $\Rightarrow$ 1.,

$$
\begin{aligned}
(LHS) &= \lim_{n \to \infty} \max_{y \in K_x} \big(an \ln \eta_x(y) + \phi(y)\big) \\
&= \max_{y \in K_x} \lim_{n \to \infty} \big(an \ln \eta_x(y) + \phi(y)\big) \\
&= \max_{y \in K_x} \big(a \mathbf{1}^{(1)}_{\{x\}}(y) + \phi(y)\big) \\
&= a \cdot 0 + \phi(x) = \phi(x)
\end{aligned}
$$

whereas

$$
\begin{aligned}
(RHS) &= \lim_{n \to \infty} a \max_{y \in K_x} \big(n \ln \eta_x(y) + \phi(y)\big) \\
&= \max_{y \in K_x} \lim_{n \to \infty} \big(an \ln \eta_x(y) + a\phi(y)\big) \\
&= \max_{y \in K_x} \big(a \mathbf{1}^{(1)}_{\{x\}}(y) + a\phi(y)\big) \\
&= a \cdot 0 + a\phi(x) = a\phi(x).
\end{aligned}
$$

For 3. $\Rightarrow$ 1.,

$$\begin{aligned}(LHS) &= \lim_{n\to\infty} \max_{y\in K_x} \big((a + n\ln\eta_x) \wedge b) + \phi(y)\big) \\ &= \max_{y\in K_x} \lim_{n\to\infty} \big((a + n\ln\eta_x) \wedge b) + \phi(y)\big) \\ &= \max_{y\in K_x} \big((a + \mathbf{1}^{(1)}_{\{x\}}(y)) \wedge b + \phi(y)\big) \\ &= a \wedge b + \phi(x)\end{aligned}$$

whereas

$$\begin{aligned}(RHS) &= \lim_{n\to\infty} \max_{y\in K_x} \big(a + n\ln\eta_x(y) + \phi(y)\big) \wedge b \\ &= \max_{y\in K_x} \lim_{n\to\infty} \big(a + n\ln\eta_x(y) + \phi(y)\big) \wedge b \\ &= \max_{y\in K_x} \big(a + \mathbf{1}^{(1)}_{\{x\}}(y) + \phi(y)\big) \wedge b \\ &= \big(a + \phi(x)\big) \wedge b.\end{aligned}$$

For 2. $\Rightarrow$ 1., this holds for all $a \geq 0$ and $\phi(x)$ can assume $[-\infty, \infty)$ and $a\phi(x) = \phi(x)$. This implies that $\phi(x) = 0$ or $\phi(x) = -\infty$ for all $x \in X$.

For 3. $\Rightarrow$ 1., this holds for all real numbers a and b and $\phi(x)$ can assume $[-\infty, \infty)$ and $a \wedge b + \phi(x) = (a + \phi(x)) \wedge b$. If $\phi(x) = -\infty$, there is no contradiction, hence, $\phi(x)$ can assume $-\infty$. If $\phi(x)$ is finite, we may choose a large enough so that $a \wedge b = b$ and $(a + \phi(x)) \wedge b = b$, which implies $b + \phi(x) = b \iff \phi(x) = 0$. As $\phi(x) \in \{0, -\infty\}$ for all $x \in X$, $\phi(x) = \mathbf{1}^{(1)}_C$ for some set C. However, C has to be closed as ϕ is an upper semicontinuous function.

□

Remark 3.3.7 (Reduction to the compact domain setting)**.** *When X is compact, one can simplify the support-functional characterization by first working on the familiar Banach space $C(X)$ of real-valued continuous functions. Any $(\vee, +)$-linear map $F : C(X) \to \mathbb{R}$ that is continuous in the sup-norm is automatically uniformly continuous. Moreover, $C(X)$ sits densely inside the tropical semimodule $C_0(X; \mathcal{A}_1)$ via the truncations $f_a = g \vee a$ (with $a \to -\infty$), which converge uniformly to each $g \in C_0(X; \mathcal{A}_1)$. By uniform continuity, F therefore extends in a unique way to a continuous, sup-preserving functional on all of $C_0(X; \mathcal{A}_1)$, and any additional axioms (shift-equivariance and either positive homogeneity or constant-capping) verified on $C(X)$ carry over to the extension. The extended functional then satisfies the hypotheses of our main characterization theorem, forcing it to be the support functional of a unique closed set.*

This two-stage argument keeps the heart of the proof entirely within the compact setting—where no $-\infty$ appears—while seamlessly incorporating the full vanishing-at-infinity behavior through a standard density-and-extension argument.

Assembling everything, we obtain the following unified picture. Recall that a continuous, sup-preserving functional $F \in \big(C_0(X; \mathcal{A}_1)\big)^*$ may—and by our characterization must—also commute either with scalar-multiplication ($\times$) or with constant-capping ($\wedge$). Accordingly, we write

$$\big(C_0(X; \mathcal{A}_1, \vee, +, \times)\big)^* \subset \big(C_0(X; \mathcal{A}_1)\big)^* \text{ and } \big(C_0(X; \mathcal{A}_1, \vee, +, \wedge)\big)^* \subset \big(C_0(X; \mathcal{A}_1)\big)^*$$

or the subspaces of those $\mathcal{A}_1$-linear maps which in addition satisfy

$$F(af) = aF(f) \text{ or } F(f \wedge c) = F(f) \wedge c,$$

respectively. Theorem 3.3.6 then shows that the support-functional embedding $C \longmapsto F_C$ gives a bijection

$$\mathcal{CL}(X) \;\xrightarrow{\cong}\; \big(C_0(X;\, \mathcal{A}_1, \vee, +, \times)\big)^* \;\cong\; \big(C_0(X;\, \mathcal{A}_1, \vee, +, \wedge)\big)^*,$$

and identifies each of these with the join-semilattice $(\mathcal{CL}(X), \cup)$. In other words, whether one demands homogeneity or capping, the only sup-preserving, shift-equivariant, idempotent-linear functionals are exactly the support functionals of closed sets.

Topologically, each of these three-operation duals sits naturally inside the full idempotent dual $\big(C_0(X; \mathcal{A}_1)\big)^*$, but the operator pseudo-norm there makes the image $\{F_C\}$ discrete—despite $\mathcal{CL}(X)$ carrying the (non-discrete) Vietoris topology. This exact same phenomenon occurs in the classical setting: total-variation and Wasserstein metrics induce distinct topologies on $\mathcal{P}(X)$. Indeed, each of these five distances admits a dual-norm-like description of the form

$$\sup_{\substack{\text{test } f \\ \|f\| \le 1}} \Big| \langle \text{test},\, \text{object}_1 \rangle \;-\; \langle \text{test},\, \text{object}_2 \rangle \Big|.$$

with the only difference being whether one uses a Lipschitz bound or a sup-norm bound on the test-functions, and whether the pairing is an integral, a max-over-set, or an idempotent integral. We summarize this duality-based characterization in Table 3.1.

Here, the first two rows use Lipschitz test-functions, while the last three use bounded (sup-norm) test-functions. This unified viewpoint makes clear why the operator-norm topology on support functionals is as coarse (discrete) as total variation, rather than as fine as Wasserstein or Vietoris.

Table 3.1: Dual-norm formulations of various distances and their associated test-space constraints

Distance	**Dual-norm formula**	**Test-space constraint**
$d_H(A, B)$	$\sup_{\|f\|_{\mathrm{Lip}} \leq 1} \left[\max_{x \in A} f(x) - \max_{y \in B} f(y) \right]$	$\|f\|_{\mathrm{Lip}} \leq 1$
$W_1(\mu, \nu)$	$\sup_{\|f\|_{\mathrm{Lip}} \leq 1} [\mathbb{E}_\mu[f] - \mathbb{E}_\nu[f]]$	$\|f\|_{\mathrm{Lip}} \leq 1$
$d_\infty(F_A, F_B)$	$\sup_{\|f\|_\infty \leq 1} \left\| \max_{x \in A} f(x) - \max_{y \in B} f(y) \right\|$	$\|f\|_\infty \leq 1$
$\|\mu - \nu\|_{\mathrm{TV}}$	$\sup_{\|f\|_\infty \leq 1} \left\| \int f \, d\mu - \int f \, d\nu \right\|$	$\|f\|_\infty \leq 1$
$d_\infty^{(2)}(\phi, \psi)$	$\sup_{\|f\|_\infty \leq 1} \left\| \sup_{x \in X} f(x)\phi(x) - \sup_{y \in X} f(y)\psi(y) \right\|$	$\|f\|_\infty \leq 1$

3.4 Uniform Approximation via the Stone–Weierstrass Theorem

This section reviews key classical and modern approximation results based on the Stone–Weierstrass theorem, forming a foundation for our later theoretical developments. We begin by recalling the classical scalar version, move to Prolla's extension for vector-valued functions, and then review recent applications to neural network approximation theorems. We conclude with a recent application in neural approximation of invariant functions on hyperspaces, setting the stage for our equivariant extensions in the next section.

3.4.1 The Classical Stone–Weierstrass Theorem

We start with the classical form of the Stone–Weierstrass Theorem, one of the fundamental approximation theorems in analysis.

Theorem 3.4.1 (Stone–Weierstrass)**.** *Let X be a compact Hausdorff space. Suppose $A \subset C(X)$ is a subalgebra of real-valued continuous functions such that:*

- ***Point separating:*** *For any distinct $x, y \in X$, there is a function $f \in A$ with $f(x) \neq f(y)$.*

- ***Contains constants:*** *A contains the constant functions.*

Then A is dense in $C(X)$ with respect to the uniform norm.

Proof. See [63], Theorem 4.45. or [64], Theorem 7.32. □

This classical result, proved by Marshall Stone [65], generalizes Weierstrass's polynomial approximation theorem and provides a fundamental criterion for approximating continuous functions uniformly.

3.4.2 Prolla's Vector-Valued Extension

Prolla [66] extended the classical theorem to vector-valued functions, an essential result when dealing with approximation problems involving multidimensional outputs. While the classical Stone–Weierstrass theorem assures density of scalar-valued function algebras, many problems in approximation theory, machine learning, and functional analysis naturally involve approximating mappings into normed spaces. Prolla's extension bridges this gap by providing a straightforward characterization of density for submodules of vector-valued continuous functions.

Theorem 3.4.2 (Prolla's Vector-Valued Stone–Weierstrass Theorem)**.** *Let X be a compact Hausdorff space, E a normed vector space, and $C(X; E)$ the space of continuous functions from X to E equipped with the uniform norm. Suppose $A \subset C(X)$ is a subalgebra of real-valued continuous functions that separates points, and $W \subset C(X; E)$ is a vector subspace that is an A-module.*

Then W is dense in $C(X; E)$ if and only if for each fixed $x \in X$, the evaluation set

$$W(x) = \{w(x) : w \in W\}$$

is dense in E.

This result significantly enhances the versatility of the Stone–Weierstrass framework. It enables direct extensions of classical approximation arguments to vector-valued scenarios by simplifying the task of verifying density to pointwise conditions. In practical terms, one only needs to ensure, at each individual point, that the evaluation images of the chosen approximating functions are sufficiently rich.

An intuitive yet powerful consequence of this result is its direct applicability to neural network approximation theory. Specifically, it clarifies precisely when a family of neural network-like architectures, viewed as vector-valued functions parameterized by weights, biases, and activations, forms a dense subset within a certain function space. By checking the pointwise density criterion, one can verify universal approximation capabilities of various architectures.

3.4.3 Neural Network Universal Approximation

The Stone–Weierstrass theorem and its vector-valued extensions provide powerful tools for establishing the universal approximation capabilities of neural networks. These theorems demonstrate that networks with sufficient complexity can approximate a wide range of functions if their activation functions and architectures satisfy certain algebraic properties. Seminal works in the late 1980s laid the groundwork for these results, showing that even relatively simple feedforward networks could act as universal approximators.

One of the cornerstone results in this area is by Cybenko [67], who showed that single hidden layer feedforward networks with continuous sigmoidal activation functions are capable of universal approximation. Specifically, sums of the form

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j \cdot x + \theta_j)$$

are dense in the space of continuous functions on a compact set (i.e., $C(I_n)$ where I_n is the n-dimensional unit cube), where σ is any continuous "squashing" (sigmoidal) function. While the primary proof of his main theorem for sigmoidal functions utilized tools from functional analysis such as the Hahn–Banach theorem and Riesz Representation theorem, Cybenko explicitly acknowledged the relevance of Stone–Weierstrass arguments for the broader landscape of universal approximation. This insight underscores the versatility of the Stone–Weierstrass framework in the context of neural network approximation for various types of nonlinearities.

Contemporaneously, Hornik, Stinchcombe, and White [68] significantly broadened the understanding of universal approximation. They used the Stone–Weierstrass theorem to show that a single hidden layer of sufficiently many units—and any continuous, nonconstant "squashing" activation—can approximate any continuous function on a compact domain. In particular, they verify that the algebra of finite linear combinations of $\sigma(w_j \cdot x + \theta_j)$ both separates points and contains constants, so Stone–Weierstrass implies density in $C(K)$. This allows the conclusion that these network architectures are universal approximators for continuous functions on compact sets. Funahashi [69] also independently provided an early proof of universal approximation capabilities around the same time, though his approach was different and did not directly invoke the Stone–Weierstrass theorem.

While these foundational papers established that various "squashing" functions were sufficient for universal approximation, the precise necessary and sufficient condition on the activation function was established shortly after. A definitive characterization was provided by Leshno et al. [70], who proved that a single-hidden-layer network is a universal approximator if and only if its activation function is not an algebraic polynomial. This result solidified the understanding of exactly which functions grant neural networks their expressive power, a finding that we formalize in the following theorem.

Theorem 3.4.3 (Characterization of Activation Functions for Universal Approximation)**.** *Let K be a compact subset of $\mathbb{R}^m$, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous function. Let $\Sigma_{m,n}$ denote the family of single-hidden-layer neural networks, which are functions $G : \mathbb{R}^m \to \mathbb{R}^n$ of the form*

$$G(x) = V\,\sigma(Wx + b),$$

where for some $N \in \mathbb{N}$, W is an $N \times m$ matrix, $b \in \mathbb{R}^N$, V is an $n \times N$ matrix, and the activation function σ is applied element-wise to the vector $Wx + b$.

*Then the family $\Sigma_{m,n}$ is dense in the space of continuous functions $C(K, \mathbb{R}^n)$ with respect to the uniform norm **if and only if** the activation function σ is not an algebraic polynomial.*

Proof. The proof follows by extending the scalar-valued result ($n = 1$), established in [70], to the vector-valued case on a component-wise basis.

A continuous function $F : K \to \mathbb{R}^n$ can be expressed by its scalar component functions, $F(x) = [f_1(x), \ldots, f_n(x)]^T$. By the original theorem, each component $f_i : K \to \mathbb{R}$ can be uniformly approximated by a network with a scalar output, $G_i(x) = v_i \cdot \sigma(Wx + b)$, where v_i is a row vector of output weights.

Stacking these n individual output vectors v_i as rows forms the $n \times N$ output matrix V. The resulting network $G(x) = V\sigma(Wx + b)$ then uniformly approximates the target function $F(x)$, proving the family is dense in $C(K, \mathbb{R}^n)$. For the complete proof of the necessary and sufficient condition on σ, see [70]. □

The principle of universal approximation, significantly illuminated by the Stone–Weierstrass theorem for function approximation, was extended to more complex mappings by later research. Notably, Chen & Chen (1995) [71] demonstrated that neural networks could also serve as universal approximators for continuous operators between function spaces. This established that networks can learn to approximate transformations from one entire function $\phi \in C(K)$ to another function $\mathcal{N}[\phi] \in C(K')$. Their work showed such an operator approximation can take the form:

$$\big(\mathcal{N}[\phi]\big)(x') = \sum_{k=1}^{N} \left(\sum_{i=1}^{M} c_i^k g \left(\sum_{j=1}^{m} \xi_{ij}^k \phi(\xi_j) + \theta_i^k \right) \right) g(\omega_k \cdot x' + \zeta_k)$$

In this structure, $\phi(\xi_j)$ represent samples of the input function ϕ at m discrete sensor points.

A crucial element enabling this capability, as with earlier universal approximation results for functions, is the nature of the activation function g. They made a significant contribution here by identifying a very broad class of suitable continuous activation functions. They established that essentially any continuous function that is not a polynomial can be used for universal approximation purposes (what they term a Tauber-Wiener function). This generalization concerning activation functions reinforces the robustness of neural networks as universal approximators—a concept for which the Stone–Weierstrass theorem provides a fundamental algebraic understanding at the level of function approximation.

While their work on operators involved specific network architectures and analytical tech-

niques tailored for function spaces, its success builds upon the foundational capacity of neural networks to approximate continuous functions. It is in this foundational layer of function approximation that the Stone–Weierstrass theorem offers key insights into why such approximation is possible. The demonstration by Chen and Chen that neural networks can effectively approximate operators mapping between infinite-dimensional spaces was a significant theoretical advance, foreshadowing later developments (e.g. DeepONet [72], Fourier neural operators [73]) in specialized areas like neural operators.

Collectively, these landmark results demonstrate the profound connection between classical approximation theory, particularly the Stone–Weierstrass theorem, and the expressive power of neural networks. By verifying that the set of functions realizable by a network architecture forms an algebra satisfying the theorem's conditions, researchers have been able to provide strong theoretical guarantees for their universal approximation capabilities.

3.4.4 Invariant Universal Approximation Theorems on Hyperspaces

The Stone–Weierstrass theorem's principles continue to underpin modern universal approximation results, particularly for neural networks processing permutation-invariant set data on hyperspaces. This is exemplified by Bueno & Hylton [6], extending approximation theory to set pooling networks.

Their approach hinges on a Topological-UAT, which itself relies on Stone–Weierstrass principles for its proof. This theorem states that for a compact Hausdorff space X and a non-polynomial activation σ, if a family of functions $S \subset C(X)$ separates points and contains a non-zero constant, then $\mathrm{span}(\sigma \circ \mathrm{span}\ S)$ is dense in $C(X)$. These conditions—separating points and

containing constants—are the hallmarks of the Stone–Weierstrass theorem.

To apply this Topological-UAT to specific set-pooling architectures, [6] identify:

- The relevant compact Hausdorff hyperspaces, such as $\mathcal{K}(\Omega)$ for max-pooling networks (like PointNet [40]) or $\mathcal{P}(\Omega)$ for average-pooling networks (like normalized-DeepSets [21]).
- Appropriate initial function families S. For max-pooling, this is $S_{Max} = \{K \mapsto \max_{x \in K} f(x) : f \in \mathcal{N}^\sigma\}$, and for average-pooling $S_{Ave} = \{\mu \mapsto \int_\Omega f \, d\mu : f \in \mathcal{N}^\sigma\}$ where $\mathcal{N}^\sigma$ represents functions from a single-hidden layer neural network.

A crucial step involves verifying that these families, S_{Max} and S_{Ave}, satisfy the Stone–Weierstrass-like conditions required by their Topological-UAT on $\mathcal{K}(\Omega)$ and $\mathcal{P}(\Omega)$ respectively. Having established these conditions, the invocation of their Topological-UAT yields universal approximation theorems for these invariant set pooling networks. For instance, PointNet-like architectures are shown to be dense in the space of functions uniformly continuous with respect to the Hausdorff metric (d_H) on $\mathcal{K}(\Omega)$.

Thus, the Stone–Weierstrass theorem, via this tailored Topological-UAT, provides the theoretical foundation for understanding the expressive power of these modern networks designed for invariant processing of set data on hyperspaces.

3.5 Universal Approximation by Set- and Measure-Based Architectures

This section provides a unified framework for proving the universal approximation capabilities of architectures for set and measure data. While the expressive power of models like PointNet [40] and (normalized) Deep Sets [21] is well known, we present a single, powerful proof strategy based on Prolla's vector-valued Stone–Weierstrass theorem. Since the hyperspaces

$\mathcal{K}(X)$ and $\mathcal{P}(X)$ are themselves compact when the base space X is compact, Prolla's theorem provides the ideal tool for this task. We use this approach to re-establish the foundational results for invariant functions and extend the analysis to their equivariant counterparts on both sets and probability measures.

3.5.1 Universal Approximation of Invariant Functions

An invariant function on a hyperspace is one whose output depends on the entire collection of points but not on their ordering. Following the line of reasoning for hyperspace learning in [6], this subsection provides simplified proofs for the universal approximation capabilities of PointNet [40] and Deep Sets [21] using a direct application of Prolla's vector-valued Stone–Weierstrass theorem.

Theorem 3.5.1. *Let X be a compact subspace of a Euclidean space $\mathbb{R}^m$. Any continuous function $F : \mathcal{K}(X) \to \mathbb{R}^n$ can be uniformly approximated by a function of the form $K \mapsto \gamma_{MLP}\big(\max_{x\in K} h_{MLP}(x)\big)$, where h_{MLP} and γ_{MLP} are Multi-Layer Perceptrons (MLPs) with a Lipschitz continuous non-polynomial activation function.*

Proof. Let $W = \{K \mapsto \gamma(\max_{x\in K} h(x)) : h \in C(X, \mathbb{R}^b), \gamma \in C(\mathbb{R}^b, \mathbb{R}^n)$ for some $b > 0\}$. We will show that W is dense in $C(\mathcal{K}(X), \mathbb{R}^n)$. According to Theorem 3.4.2, W must satisfy two conditions.

1. **Pointwise Density:** For any $K_0 \in \mathcal{K}(X)$, the evaluation set $W(K_0) = \{g(K_0) : g \in W\}$ must be dense in $\mathbb{R}^n$. This is satisfied because for any target vector $v \in \mathbb{R}^n$, we can choose any continuous function h and define $\gamma : \mathbb{R}^b \to \mathbb{R}^n$ as the constant function $\gamma(y) = v$. The

resulting function $g(K) = \gamma(\max_{x \in K} h(x)) = v$ is in W. Thus, $W(K_0)$ contains every vector $v \in \mathbb{R}^n$ and is therefore dense.

2. **Separating Multiplier:** For any two distinct sets $S_1, S_2 \in \mathcal{K}(X)$, we must construct a multiplier φ such that $\varphi(S_1) \neq \varphi(S_2)$ and for any $g_1, g_2 \in W$, the function $g = \varphi g_1 + (1 - \varphi) g_2$ is also in W.

   Assume $S_1 \neq S_2$ and, without loss of generality, let $x_* \in S_1 \setminus S_2$. Define the continuous function $\varphi : \mathcal{K}(X) \to [0, 1]$ by:

$$\varphi(S) = \frac{\min_{x \in S} d(x, x_*)}{1 + \min_{x \in S} d(x, x_*)}$$

   This function separates the sets, as $\varphi(S_1) = 0$ and $\varphi(S_2) > 0$. To show it is a multiplier, let $g_1(S) = \gamma_1(\max_{x \in S} h_1(x))$ and $g_2(S) = \gamma_2(\max_{x \in S} h_2(x))$. We construct a new function $g(S) = \gamma\big(\max_{x \in S} h(x)\big)$ by defining:

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ -d(x, x_*) \end{bmatrix} \quad \text{and} \quad \gamma\left(\begin{bmatrix} y_1 \\ y_2 \\ a \end{bmatrix}\right) = \frac{-a}{1-a}\gamma_1(y_1) + \left(1 - \frac{-a}{1-a}\right)\gamma_2(y_2)$$

   Since $\max_{x \in S}(-d(x, x_*)) = -\min_{x \in S} d(x, x_*)$, a direct substitution shows that $g(S) = \varphi(S) g_1(S) + (1 - \varphi(S)) g_2(S)$. Thus, φ is a valid separating multiplier.

Since both conditions of Theorem 3.4.2 are met, the family of ideal functions W is dense in $C(\mathcal{K}(X), \mathbb{R}^n)$.

Let $F \in C(\mathcal{K}(X), \mathbb{R}^n)$ and $\epsilon > 0$ be given. We need to find MLPs h_{MLP} and γ_{MLP} such that the uniform error $\|F(K) - \gamma_{\text{MLP}}(\max_{x \in K} h_{\text{MLP}}(x))\|$ is less than ϵ.

By the above approximation, we know there exist continuous functions h and γ such that:

$$\|F(K) - \gamma\big(\max_{x \in K} h(x)\big)\| < \frac{\epsilon}{3} \quad \text{for all } K \in \mathcal{K}(X)$$

Using the triangle inequality, the total error is bounded by:

$$\begin{aligned}\|F(K) - \gamma_{\text{MLP}}(\max_{x \in K} h_{\text{MLP}}(x))\| \leq \|F(K) - \gamma(\max_{x \in K} h(x))\| \\ + \|\gamma(\max_{x \in K} h(x)) - \gamma_{\text{MLP}}(\max_{x \in K} h(x))\| \\ + \|\gamma_{\text{MLP}}(\max_{x \in K} h(x)) - \gamma_{\text{MLP}}(\max_{x \in K} h_{\text{MLP}}(x))\|\end{aligned}$$

We bound each term:

1. Term 1: $\|F(K) - \gamma(\max_{x \in K} h(x))\| < \epsilon/3$ by the above approximation.

2. Term 2: The domain for γ is the set $C = \{\max_{x \in K} h(x) \mid K \in \mathcal{K}(X)\}$. Since X is compact and h is continuous, $h(X)$ is compact, and therefore C is a compact subset of $\mathbb{R}^b$. By the Universal Approximation Theorem for MLPs, we can choose an MLP γ_{MLP} such that

   $$\|\gamma(y) - \gamma_{\text{MLP}}(y)\| < \epsilon/3 \quad \text{for all } y \in C.$$

   This bounds the second term.

3. Term 3: Since γ_{MLP} has a Lipschitz continuous activation function, it is Lipschitz continuous on the compact set C with some constant L_γ. Furthermore, the $\max$ operator is

1-Lipschitz. Therefore:

$$\begin{aligned}&\|\gamma_{\mathrm{MLP}}(\max_{x\in K} h) - \gamma_{\mathrm{MLP}}(\max_{x\in K} h_{\mathrm{MLP}})\| \\ &\leq L_\gamma \|\max_{x\in K} h - \max_{x\in K} h_{\mathrm{MLP}}\| \leq L_\gamma \max_{x\in X} \|h(x) - h_{\mathrm{MLP}}(x)\|\end{aligned}$$

By Theorem 3.4.3 on the compact set X, we can choose an MLP h_{MLP} such that its uniform distance from h is

$$\sup_{x\in X} \|h(x) - h_{\mathrm{MLP}}(x)\| < \frac{\epsilon}{3L_\gamma}.$$

This ensures the third term is less than $\epsilon/3$.

Combining these bounds, the total error is less than

$$\epsilon/3 + \epsilon/3 + \epsilon/3 = \epsilon.$$

This holds uniformly for all $K \in \mathcal{K}(X)$, completing the proof. □

Theorem 3.5.2. *Let X be a compact subspace of a Euclidean space $\mathbb{R}^m$. Any continuous function $F : \mathcal{P}(X) \to \mathbb{R}^n$ can be uniformly approximated by a function of the form $P \mapsto \gamma_{\mathrm{MLP}}\left(\int h_{\mathrm{MLP}}(x)\, dP(x)\right)$, where h_{MLP} and γ_{MLP} are MLPs with a Lipschitz continuous non-polynomial activation function.*

Proof. The proof follows the same two-stage structure as the set-invariant case.

First, we show that the family of ideal functions

$$W = \left\{ P \mapsto \gamma\left(\int h(x)\, dP(x)\right) : h \in C(X, \mathbb{R}^b),\ \gamma \in C(\mathbb{R}^b, \mathbb{R}^n) \right\}$$

is dense in $C(\mathcal{P}(X), \mathbb{R}^n)$ by applying Theorem 3.4.2.

1. **Pointwise Density:** This condition is satisfied for the same reason as in the set-invariant case; for any target vector $v \in \mathbb{R}^n$, one can choose γ to be the constant function $\gamma(y) = v$.

2. **Separating Multiplier:** To separate two distinct probability measures $P_1, P_2 \in \mathcal{P}(X)$, we invoke the Riesz–Markov–Kakutani representation theorem (Theorem 3.3.2). It guarantees the existence of a continuous function $g \in C(X)$ such that

$$\int g \, dP_1 \neq \int g \, dP_2.$$

We can assume g is non-negative. We define the continuous separator $\varphi : \mathcal{P}(X) \to [0, 1]$ by:

$$\varphi(P) = \frac{\int g(x) \, dP(x)}{1 + \int g(x) \, dP(x)}$$

This function separates P_1 and P_2. The proof that φ is a valid multiplier follows the same construction as in the set-invariant case, this time embedding the separating function $g(x)$ into the feature map $h(x)$.

Since both conditions are met, the family of ideal functions W is dense in $C(\mathcal{P}(X), \mathbb{R}^n)$.

Second, the extension to MLP-based architectures follows an identical uniform approximation argument using the triangle inequality. This argument relies on the Universal Approximation Theorem for MLPs (Theorem 3.4.3) and the fact that the integral operator $P \mapsto \int f \, dP$ is 1-Lipschitz (with respect to the p-Wasserstein distance for $p \geq 1$ and a Lipschitz function f). This ensures that small errors in approximating the feature extractor h with h_{MLP} are not amplified, allowing the total error to be made arbitrarily small. This completes the proof. □

3.5.2 Universal Approximation of Equivariant Functions

An equivariant function on a hyperspace operates on a pointed space, taking an element and its surrounding context as input to produce an output corresponding to that specific element. This structure is fundamental to attention mechanisms and any architecture that produces per-point features. Before presenting the approximation theorems, we formally define the domains for these functions and establish their starkly different topological properties.

Definition 3.5.3. *Let X be a compact metric space. The* space of pointed compact sets*, denoted $\mathrm{Pt}(\mathcal{K}(X))$, is the subspace of $X \times \mathcal{K}(X)$ defined by*

$$\mathrm{Pt}(\mathcal{K}(X)) \triangleq \big\{(x, K) \in X \times \mathcal{K}(X) \colon x \in K\big\}.$$

Definition 3.5.4. *Let X be a compact metric space. The* space of pointed probability measures*, denoted $\mathrm{Pt}(\mathcal{P}(X))$, is the subspace of $X \times \mathcal{P}(X)$ defined by*

$$\mathrm{Pt}(\mathcal{P}(X)) \triangleq \big\{(x, P) \in X \times \mathcal{P}(X) \colon x \in \operatorname{supp} P\big\}.$$

These two domains, while structurally similar, are topologically very different. The set-based domain is a hard-boundary closed set, while the measure-based domain is a soft-boundary dense set, a crucial distinction for approximation theory.

Theorem 3.5.5. *Let X be a compact metric space with at least two distinct points.*

1. *The space of pointed compact sets $\mathrm{Pt}(\mathcal{K}(X))$ is a closed and compact proper subset of $X \times \mathcal{K}(X)$.*

2. *The space of pointed probability measures* $\mathrm{Pt}(\mathcal{P}(X))$ *is a dense and non-compact proper subset of* $X \times \mathcal{P}(X)$.

Proof. To prove that $\mathrm{Pt}(\mathcal{K}(X))$ is closed, we show its complement is open. Let $(x, K) \notin \mathrm{Pt}(\mathcal{K}(X))$, meaning $x \notin K$. Since K is closed, $d(x, K) > 0$. Set $\epsilon = d(x, K)/2 > 0$. Then

$$U \;=\; \big\{y \in X : d(x, y) < \epsilon\big\} \;\times\; \big\{S \in \mathcal{K}(X) : d_H(K, S) < \epsilon\big\}$$

is an open neighborhood of (x, K) in $X \times \mathcal{K}(X)$. For any $(y, S) \in U$,

$$d(y, x) < \epsilon, \quad d_H(S, K) < \epsilon \quad \Longrightarrow \quad d_H(\{y\}, K) \geq d(x, K) - d(x, y) > 2\epsilon - \epsilon = \epsilon,$$

so $y \notin S$ and $(y, S) \notin \mathrm{Pt}(\mathcal{K}(X))$. Thus U lies in the complement, showing it is open. Hence $\mathrm{Pt}(\mathcal{K}(X))$ is closed. Since $X \times \mathcal{K}(X)$ is compact, any closed subset—including $\mathrm{Pt}(\mathcal{K}(X))$—is compact. It is proper because X has at least two points, so there exist (x, K) with $x \notin K$.

To prove that $\mathrm{Pt}(\mathcal{P}(X))$ is dense, let $(x, P) \in X \times \mathcal{P}(X)$ be arbitrary. For any $\epsilon > 0$, choose $0 < \alpha < 1$ and define

$$P' = (1 - \alpha)P + \alpha\, \delta_x.$$

Then $x \in \mathrm{supp}(P')$, so $(x, P') \in \mathrm{Pt}(\mathcal{P}(X))$. Moreover, by properties of the Wasserstein metric W_p,

$$W_p(P, P') = W_p\big(P, (1 - \alpha)P + \alpha\delta_x\big) \leq \alpha\, W_p(P, \delta_x) \to 0 \quad (\alpha \to 0),$$

so we can make $W_p(P, P') < \epsilon$. Hence (x, P) can be approximated arbitrarily well by pointed measures, proving density.

Since X has at least two points, pick $x_1 \neq x_2$ and let $P = \delta_{x_1}$. Then $x_2 \notin \operatorname{supp}(P)$, so $(x_2, P) \notin \operatorname{Pt}(\mathcal{P}(X))$, showing the subset is proper. Finally, a dense proper subset of a compact space is never closed, so $\operatorname{Pt}(\mathcal{P}(X))$ is not compact. □

An equivariant function on a hyperspace operates on a pointed space, taking an element and its surrounding context as input to produce an output corresponding to that specific element. In the context of multi-agent systems and interactive trajectory prediction, the two primary forms of equivariant functions represent fundamentally different modeling philosophies.

The first form,

$$(x, S) \;\mapsto\; \gamma\big(x,\ \text{aggregate}_y h(y)\big),$$

can be interpreted as a *global summary* model. The term $\text{aggregate}_y h(y)$ computes a single, fixed-size context vector that summarizes the entire "scene" (i.e., all agents). The specific agent x then processes its own state alongside this global summary to determine its behavior.

The second form,

$$(x, S) \;\mapsto\; \gamma\big(\text{aggregate}_y \phi(x, y)\big),$$

represents an *individualized interaction* model. Here, the function $\phi(x, y)$ explicitly models the unique influence or relationship of each agent y on the specific agent x. The aggregator then pools these personalized influences to form a custom context vector for x. This allows each agent to react differently to the same set of peers based on its own state and perspective, which is crucial for modeling complex, heterogeneous interactions.

Understanding this distinction between a global context and individualized interactions is key. The following theorems establish that both architectural styles are universal approximators

for any continuous equivariant function.

We now state the universal approximation results for continuous functions on the compact domain $\mathrm{Pt}(\mathcal{K}(X))$.

Theorem 3.5.6. *Let X be a compact subspace of $\mathbb{R}^m$. Any continuous function*

$$F : \mathrm{Pt}(\mathcal{K}(X)) \;\to\; \mathbb{R}^n$$

can be uniformly approximated by MLP-based architectures of the form:

(1) $(x, K) \mapsto \gamma_{\mathrm{MLP}}\big(x,\ \max_{y\in K} h_{\mathrm{MLP}}(y)\big)$ *(Global summary model)*

(2) $(x, K) \mapsto \gamma_{\mathrm{MLP}}\big(\max_{y\in K} \phi_{\mathrm{MLP}}(x, y)\big)$ *(Individualized interaction model)*

Proof. Both proofs follow the two-stage structure: (i) apply Prolla's theorem to the ideal function families, then (ii) extend to MLPs via the Universal Approximation Theorem and the triangle inequality. The key is to construct a separating multiplier φ on $\mathrm{Pt}(\mathcal{K}(X))$.

Let (x_1, S_1) and (x_2, S_2) be distinct points in $\mathrm{Pt}(\mathcal{K}(X))$. Choose $x_* \in S_1 \setminus S_2$. Define

$$\varphi(x, S) \;=\; \frac{1}{2}\Big(\frac{d_X(x, x_1)}{d_X(x, x_1) + d_X(x, x_2)} + \frac{\min_{z\in S} d_X(z, x_*)}{1 + \min_{z\in S} d_X(z, x_*)}\Big).$$

Then $\varphi(x_1, S_1) = 0$ while $\varphi(x_2, S_2) > 0$ (avoiding edge cases by generic choice).

We verify φ is a valid multiplier for each ideal family:

Global summary model.

$$(x, K) \;\mapsto\; (1 - \varphi)\, g_1(x, \max_{y\in K} h(y)) \;+\; \varphi\, g_2(x, \max_{y\in K} h(y))$$

can be written as

$$(x, K) \;\mapsto\; \gamma\big(x, \max_{y\in K}\, h_{\text{new}}(y)\big),$$

where

$$h_{\text{new}}(y) = \begin{bmatrix} h_1(y) \\ h_2(y) \\ -d_X(y, x_*) \end{bmatrix}, \qquad \gamma\big(x, \begin{bmatrix} u \\ v \\ a \end{bmatrix}\big) = \varphi_{\text{new}}(x, a)\, \gamma_1(x, u) + (1 - \varphi_{\text{new}}(x, a))\, \gamma_2(x, v),$$

and

$$\varphi_{\text{new}}(x, a) = \frac{1}{2}\Big(\frac{d_X(x, x_1)}{d_X(x, x_1) + d_X(x, x_2)} + \frac{-a}{1 - a}\Big).$$

Individualized interaction model.

$$(x, K) \;\mapsto\; (1 - \varphi)\, g_1\big(\max_{y\in K} \phi(x, y)\big) \;+\; \varphi\, g_2\big(\max_{y\in K} \phi(x, y)\big)$$

can be written as

$$(x, K) \;\mapsto\; \gamma\big(\max_{y\in K} \phi_{\text{new}}(x, y)\big),$$

where

$$\phi_{\text{new}}(x, y) = \begin{bmatrix} \phi_1(x, y) \\ \phi_2(x, y) \\ d_X(x, x_1) \\ d_X(x, x_2) \\ -d_X(y, x_*) \end{bmatrix}, \quad \gamma\big(\begin{bmatrix} u \\ v \\ \alpha \\ \beta \\ a \end{bmatrix}\big) = \varphi_{\text{new}}(\alpha, \beta, a)\, \gamma_1(v) + (1 - \varphi_{\text{new}}(\alpha, \beta, a))\, \gamma_2(u),$$

and

$$\varphi_{\text{new}}(\alpha, \beta, a) = \tfrac{1}{2}\Big(\tfrac{\alpha}{\alpha+\beta} + \tfrac{-a}{1-a}\Big).$$

Since Prolla's conditions hold, the ideal families are dense. Extending each to MLPs via the usual approximation arguments completes the proof sketch. □

We now address the universal approximation of equivariant functions on the space of probability measures. The natural domain is the complete space $X \times \mathcal{P}(X)$, which is the closure of the pointed measures $\mathrm{Pt}(\mathcal{P}(X))$.

Theorem 3.5.7. *Let X be a compact subspace of $\mathbb{R}^m$. Any continuous function*

$$F : X \times \mathcal{P}(X) \;\to\; \mathbb{R}^n$$

can be uniformly approximated by MLP-based architectures of the form:

1. $(x, P) \;\mapsto\; \gamma_{\text{MLP}}\big(x,\; \int h_{\text{MLP}}(y)\,\mathrm{d}P(y)\big)$ *(Global summary model)*
2. $(x, P) \;\mapsto\; \gamma_{\text{MLP}}\Big(\int \phi_{\text{MLP}}(x, y)\,\mathrm{d}P(y)\Big)$ *(Individualized interaction model)*

Proof. Both proofs follow the two-stage structure: (i) prove density of the ideal function families via Prolla's theorem, then (ii) extend to MLPs by the Universal Approximation Theorem and the triangle inequality. The key is constructing a separating multiplier $\varphi\colon X \times \mathcal{P}(X) \to [0, 1]$.

By the Riesz representation theorem, given two distinct $(x_1, P_1) \neq (x_2, P_2)$, there exists $g \in C(X)$ with $\int g\,\mathrm{d}P_1 \neq \int g\,\mathrm{d}P_2$. Choose $\alpha, \beta \geq 0$ and combine with a point-separator to define

$$\varphi(x, P) \;=\; \frac{1}{2}\Big(\frac{d_X(x, x_1)}{d_X(x, x_1) + d_X(x, x_2)} \;+\; \frac{\int g(y)\,\mathrm{d}P(y)}{1 + \int g(y)\,\mathrm{d}P(y)}\Big).$$

Then $\varphi(x_1, P_1) = 0$ while $\varphi(x_2, P_2) > 0$.

Global summary model. The convex combination

$$(1-\varphi)\, g_1\big(x, \int h(y)\,\mathrm{d}P(y)\big) \;+\; \varphi\, g_2\big(x, \int h(y)\,\mathrm{d}P(y)\big)$$

can be written as

$$(x, P) \;\mapsto\; \gamma\Big(x,\; \int h_{\mathrm{new}}(y)\,\mathrm{d}P(y)\Big),$$

where

$$h_{\mathrm{new}}(y) = \begin{bmatrix} h_1(y) \\ h_2(y) \\ g(y) \end{bmatrix}, \quad \gamma\Big(x, \begin{bmatrix} u \\ v \\ a \end{bmatrix}\Big) = \varphi_{\mathrm{new}}(x,a)\,\gamma_1(x,u) \;+\; (1-\varphi_{\mathrm{new}}(x,a))\,\gamma_2(x,v),$$

and

$$\varphi_{\mathrm{new}}(x,a) = \tfrac{1}{2}\Big(\frac{d_X(x,x_1)}{d_X(x,x_1)+d_X(x,x_2)} \;+\; \frac{a}{1+a}\Big).$$

Individualized interaction model. Similarly,

$$(1-\varphi)\, g_1\Big(\int \phi(x,y)\,\mathrm{d}P(y)\Big) \;+\; \varphi\, g_2\Big(\int \phi(x,y)\,\mathrm{d}P(y)\Big)$$

becomes

$$(x, P) \;\mapsto\; \gamma\Big(\int \phi_{\mathrm{new}}(x,y)\,\mathrm{d}P(y)\Big),$$

where

$$\phi_{\text{new}}(x,y) = \begin{bmatrix} \phi_1(x,y) \\ \phi_2(x,y) \\ d_X(x,x_1) \\ d_X(x,x_2) \\ g(y) \end{bmatrix}, \quad \gamma\Bigg(\begin{bmatrix} u \\ v \\ \alpha \\ \beta \\ a \end{bmatrix} \Bigg) = \varphi_{\text{new}}(\alpha,\beta,a)\,\gamma_1(v) \;+\; (1-\varphi_{\text{new}}(\alpha,\beta,a))\,\gamma_2(u),$$

and

$$\varphi_{\text{new}}(\alpha,\beta,a) = \tfrac{1}{2}\Big(\tfrac{\alpha}{\alpha+\beta} + \tfrac{a}{1+a}\Big).$$

Since Prolla's conditions hold, the ideal families are dense; extension to MLPs via uniform approximation completes the proof. □

Remark 3.5.8 (On the Domain of Equivariant Functions on Measures)**.** *The preceding universal approximation theorem for measure-equivariant functions is stated for a target function F in the space $C(X \times \mathcal{P}(X), \mathbb{R}^n)$. This choice of domain warrants a technical clarification.*

Let $A = X \times \mathcal{P}(X)$ and $D = \mathrm{Pt}(\mathcal{P}(X))$. We have established that A is a compact metric space and D is a dense subset of A. A standard theorem in analysis states that a continuous function $f : D \to Y$ from a dense subset D of a metric space A into a complete metric space Y (such as $\mathbb{R}^n$) has a unique continuous extension $\tilde{f} : A \to Y$ if and only if f is uniformly continuous on D.

Therefore, for the universal approximation theorem on the full space A to be meaningful, we are implicitly considering the class of functions for which such a continuous extension from D exists. Any continuous function F defined on the compact domain A is, by the Heine–Cantor

theorem, automatically uniformly continuous. Thus, its restriction to the dense subset D is also uniformly continuous, guaranteeing the condition is met.

In summary, stating the theorem on the complete space $X \times \mathcal{P}(X)$ is equivalent to considering the approximation of uniformly continuous functions on the pointed space $\mathrm{Pt}(\mathcal{P}(X))$.

Chapter 4: Dynamic Modeling & Temporal Equivariance

This chapter develops a comprehensive framework for modeling temporal dynamics, bridging classical discrete-time methods with a modern, continuous-time perspective suitable for deep learning. We begin by reviewing the foundational concepts of recurrence relations and state-space models, such as the Kalman filter, highlighting the inherent link between deterministic and stochastic viewpoints but also their limitations in handling continuous, irregular phenomena.

Motivated by these limitations, we transition to a richer, continuous-time representation where the state of a system is not merely its current position, but its entire past trajectory, or history function. We formalize the dynamics on this space using the theory of Retarded Functional Differential Equations (RFDEs), which provides a principled way to model systems with memory.

Finally, we connect this theoretical framework to practical implementation by applying a universal approximation theorem, which establishes that the functional governing these dynamics can be learned by a causal convolutional neural network. This result provides the theoretical justification for using a principled neural architecture to learn complex, time-equivariant dynamics from data.

4.1 Recurrence, Markov, and State-Space Models

In many forecasting tasks—pedestrian trajectories, vehicle tracks, etc.—we only ever see discrete samples of a moving object. One common strategy is to assume a deterministic k-step update (or RNN in a learned latent) that maps the last k positions to the next one. An alternative is a stochastic state-space view (e.g. Kalman or particle filters) in which each new state is drawn from a noisy transition. In this section, we explain how these two perspectives are formally related—deterministic kth-order recurrence relations versus first-order hidden Markov models—and survey the classical filters that arise when Gaussian or nonlinear noise is introduced.

4.1.1 Deterministic k-Step Recurrence and RNNs

In many classical time-series and forecasting settings, one assumes that the next state $x_{t+1} \in \mathbb{R}^d$ is a deterministic function of a fixed finite history of the last k observations. Concretely, a kth-order recurrence relation [74] takes the form

$$x_{t+1} = F(x_t, x_{t-1}, \ldots, x_{t-k+1}), \quad F : (\mathbb{R}^d)^k \to \mathbb{R}^d.$$

Once the vector $(x_t, x_{t-1}, \ldots, x_{t-k+1})$ is known, there is no remaining randomness: the next position x_{t+1} is fixed. In probabilistic language, this is exactly a degenerate (Dirac–delta) kth-order Markov chain, since

$$\Pr(x_{t+1} \mid x_t, x_{t-1}, \ldots) = \delta\left(x_{t+1} - F(x_t, \ldots, x_{t-k+1})\right).$$

A well-known special case is the (linear) $AR(k)$ model, where

$$x_{t+1} = A_1 x_t + A_2 x_{t-1} + \cdots + A_k x_{t-k+1}, \quad A_i \in \mathbb{R}^{d \times d}.$$

A closely related viewpoint arises in recurrent neural networks (RNNs). A standard (deterministic) RNN maintains a hidden state $h_t \in \mathbb{R}^m$ and updates it via

$$h_{t+1} = f_\theta(h_t, x_t), \quad \hat{x}_{t+1} = g_\phi(h_{t+1})$$

where $f_\theta : \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^m$ is any parametrized recurrent-cell update (e.g. vanilla RNN, GRU, LSTM) and $g_\phi : \mathbb{R}^m \to \mathbb{R}^d$ is the readout map (typically affine). Because h_{t+1} depends only on h_t (and the current input x_t), the sequence $\{h_t\}$ forms a first-order deterministic Markov chain in $\mathbb{R}^m$. If m is large enough, the vector h_t can encode the last k observations, so an RNN may implicitly implement a kth-order recurrence relation in the original x-space.

In summary, deterministic kth-order recurrence relation and RNNs both produce a single best-guess trajectory, without modeling any noise: once the required past values (or hidden state) are given, the next state is fully determined. In applications where one only needs a point estimate (no uncertainty quantification), such deterministic models are simple and effective. Their limitation, however, is that they cannot capture or propagate any inherent randomness in the system—every run yields exactly one output given the same inputs.

4.1.2 Linear–Gaussian State-Space Models and the Kalman Filter

A classical approach to trajectory estimation posits that the true state $\beta_t \in \mathbb{R}^n$ evolves linearly in time with additive Gaussian process noise, and that each observation $y_t \in \mathbb{R}^p$ is a linear, noisy measurement of β_t. Concretely, one assumes

$$\beta_t = F\beta_{t-1} + w_t, \qquad w_t \sim \mathcal{N}(0, Q),$$
$$y_t = H\beta_t + v_t, \qquad v_t \sim \mathcal{N}(0, R),$$

where $F \in \mathbb{R}^{n\times n}$ is the state-transition matrix, $H \in \mathbb{R}^{p\times n}$ the observation matrix, and Q, R are the process- and measurement-noise covariances. An initial prior $\beta_0 \sim \mathcal{N}(\hat{\beta}_{0|0}, P_{0|0})$ is also assumed.

Under these assumptions, the Kalman filter [42] provides the exact minimum-mean-squared-error estimates of β_t given the observations $y_{1:t}$. Denote by $\hat{\beta}_{t|t}$ and $P_{t|t}$ the posterior mean and covariance of $\beta_t \mid y_{1:t}$, and by $\hat{\beta}_{t|t-1}$ and $P_{t|t-1}$ the one-step-ahead prediction of β_t given $y_{1:t-1}$. The recursion splits into two steps:

- **Predict (Time Update):**

$$\hat{\beta}_{t|t-1} = F\hat{\beta}_{t-1|t-1},$$
$$P_{t|t-1} = FP_{t-1|t-1}F^T + Q.$$

- **Correct (Measurement Update):**

$$K_t = P_{t|t-1}H^T \left(HP_{t|t-1}H^T + R\right)^{-1},$$
$$\hat{\beta}_{t|t} = \hat{\beta}_{t|t-1} + K_t \left(y_t - H\hat{\beta}_{t|t-1}\right),$$
$$P_{t|t} = (I - K_t H)\, P_{t|t-1}.$$

Here, K_t is the Kalman gain matrix, which balances the predicted covariance against measurement noise. Once $\hat{\beta}_{t|t}$ is available, one may form a one-step forecast of the next observation as

$$\hat{y}_{t+1|t} = H\hat{\beta}_{t+1|t}, \quad \hat{\beta}_{t+1|t} = F\hat{\beta}_{t|t}.$$

This fundamental capability for one-step-ahead prediction can be extended for longer forecast horizons. For instance, predicting k steps into the future, given data up to time t, can be achieved by recursively applying the state transition matrix: $\hat{\beta}_{t+k|t} = F^k\hat{\beta}_{t|t}$. The covariance of this multi-step prediction also evolves, accumulating process noise over the prediction interval.

This foundational Kalman filter framework has been notably extended to handle more complex scenarios. For instance, [75] addressed the tracking and prediction of maneuvering targets by modeling target acceleration as a time-correlated process within a Kalman filter structure. This approach allowed for explicit k-step ahead trajectory predictions, $\hat{\beta}_{t+k|t}$, and an analysis of how maneuver characteristics (via a correlation coefficient α and maneuver variance σ_m^2) impact prediction error covariance, providing crucial insights for systems dealing with agile targets.

Similarly, [76] tackled the challenge of tracking in cluttered environments where measurement origins are uncertain. They introduced the Probabilistic Data Association Filter (PDAF). This approach calculates the probability of each validated return being correct and uses a weighted

combination of these returns to inform the state update. This update process is integrated with the predictive and corrective steps of a Kalman filter, which is adapted to handle this probabilistic input. This significantly improved tracking robustness when false alarms or multiple returns are present, adapting the estimate's confidence based on the current measurement scenario.

4.1.3 General State-Space Models and Nonlinear/Non-Gaussian Filters

In the most general (discrete-time) state-space formulation [42], one assumes an unobserved (latent) state process $\{\beta_t\}$ and an observed measurement process $\{Y_t\}$ satisfying:

$$\begin{aligned} \text{(General observation equation)} : &\quad Y_t \mid \beta_t \sim p(y_t \mid \beta_t), \\ \text{(General system equation)} : &\quad \beta_t \mid \beta_{t-1} \sim p(\beta_t \mid \beta_{t-1}), \\ \text{(Initial information)} : &\quad \beta_0 \sim p(\beta_0). \end{aligned}$$

In words, at each time step t, the new state β_t is sampled from a (potentially nonlinear, non-Gaussian) transition density $p(\beta_t|\beta_{t-1})$, and the new measurement Y_t is drawn from an observation density $p(y_t|\beta_t)$. In robotics and computer-vision systems, one typically needs to perform two complementary tasks [77] at each time step t:

- Filtering(tracking): compute the posterior $p(\beta_t|Y_{1:t})$ after seeing $Y_1, \ldots, Y_t$.
- Prediction (forecasting): generate $p(\beta_{t+k}|Y_{1:t})$ or its moments for $k \geq 1$.

Exact, closed-form computation of $p(\beta_t|Y_{1:t})$ or multi-step forecasts $p(\beta_{t+k}|Y_{1:t})$ is intractable whenever the transition density $p(\beta_t|\beta_{t-1})$ or the observation density $p(y_t|\beta_t)$ is highly nonlinear or non-Gaussian. For this reason, most real-time systems rely on approximate filtering algorithms. Two of the most widely used are :

- Extended Kalman Filter (EKF). [78]

  When both the state-transition function and the measurement function can be well approximated by local first-order (linear) expansions, and when process and measurement noises are (approximately) Gaussian, the EKF provides a lightweight, analytic-approximation filter. By linearizing around the current mean estimate, EKF propagates its Gaussian belief $\mathcal{N}\big(\hat{\beta}_{t|t}, P_{t|t}\big)$ forward and then corrects it using each new observation. EKF (and its variants) remains one of the most popular choices for real-time pose-tracking and short-term forecasting in both robotics (e.g. mobile-robot localization, UAV attitude estimation) and computer vision (e.g. camera-based object tracking, inertial-vision fusion) [77].

- Particle Filters (PFs). [79]

  In scenarios where the transition or observation models are strongly nonlinear and/or noise distributions are multimodal or heavy-tailed, EKF's Gaussian approximation can break down. Particle Filters offer a flexible sampling-based alternative. By maintaining a set of weighted particles $\big\{\beta_t^{(i)}, w_t^{(i)}\big\}_{i=1}^{N}$, PF can approximate arbitrary posterior shapes $p(\beta_t|Y_{1:t})$. Although PF incurs a sampling cost that grows with the number of particles N, it has become the standard approach for nonlinear, non-Gaussian tracking tasks—such as monocular SLAM using approaches, localization in scenarios with highly multimodal sensor data, and hybrid three-dimensional camera pose estimation [77].

In short, EKF and PF exemplify how the very general state-space paradigm has been made practically useful in robotics, computer vision, and trajectory-prediction/tracking. By summarizing these two canonical filters, we highlight the gap between ideal inference in arbitrary $p(\cdot \mid \cdot)$ models and the real-time approximations that practitioners must use.

4.1.4 Synthesizing Deterministic and Stochastic Perspectives

We have now seen three related threads:

1. A deterministic kth-order recurrence relation

$$x_{t+1} = F(x_t, x_{t-1}, \ldots, x_{t-k+1})$$

is precisely the zero-noise (Dirac-delta) special case of a kth-order Markov model

$$x_{t+1} \sim p(x_{t+1}|x_t, x_{t-1}, \ldots, x_{t-k+1}).$$

In other words, adding stochasticity to a kth-order recurrence transforms it into a kth-order Markov model.

2. A standard RNN

$$h_{t+1} = f_\theta(h_t, x_t), \quad \hat{x}_{t+1} = g_\phi(h_{t+1})$$

is exactly the deterministic, computer-science instantiation of a first-order state-space model

$$\beta_{t+1} \sim p_\theta(\beta_{t+1} \mid \beta_t, x_t), \quad Y_{t+1} \sim p_\phi(y_{t+1} \mid \beta_{t+1}),$$

with the special choice that each transition and observation is a point-mass (no noise). In other words, an RNN's hidden-state update is the zero-noise limit of a general (possibly nonlinear, non-Gaussian) state-space model.

Putting these two observations side by side yields a clear analogy:

$$\underbrace{k\text{th-order recurrence relation}}_{\text{zero-noise}} \;:\; \underbrace{k\text{th-order Markov chain}}_{\text{stochastic}}$$

$$= \underbrace{\text{RNN}}_{\text{zero-noise SSM}} \;:\; \underbrace{\text{General state-space model.}}_{\text{stochastic SSM}}$$

Thus:

- kth-order recurrence relation is to kth-order Markov model as
- RNN is to general state-space model.

In each pair, the left side is the purely deterministic, zero-noise version of the right side's stochastic model. This parallelism clarifies why RNNs can learn to implement deterministic kth-order dependencies, and why adding noise to either an AR(k)-type update or an RNN update yields the classical filters (EKF, PF, etc.) used for tracking and prediction.

In closing, these observations highlight two takeaways:

1. Deterministic recurrence and RNNs serve as simple, point-estimate predictors when uncertainty is not modeled or needed.
2. Adding stochasticity converts each into a full Bayesian filtering framework—kth-order Markov models for explicit finite-history dependence, and general (first-order) state-space models for latent-variable formulations—each of which then calls for approximate filters (EKF, PF) in practice.

In the next section, we will shift from these discrete-time, sample-based viewpoints to a

continuous-time, delay-aware framework, showing how a retarded functional model can generalize both recurrence-based and state-space-based approaches.

4.2 Latent Dynamics on Padded Jets and Trajectories

This section initiates the shift towards a continuous-time, delay- and boundary-aware paradigm. We begin by exploring how finite historical information—of varying lengths—can define a state richer than mere position, conceptualized through padded jet bundles. Although these padded jets capture more context than position by itself, they have inherent limitations. Recognizing these limitations motivates infinite-history functional representations—history bundles that explicitly pad missing intervals—and naturally leads to Retarded Functional Differential Equations (RFDEs), which are central to the set-evolution framework of this thesis. Throughout, the augmented state always includes the observable $x(t) \in \mathbb{R}^d$ as a direct component, so no separate projection layer is needed to recover position.

4.2.1 It's All Finite Sampling

While agents move continuously in $\mathbb{R}^d$, our sensors only record discrete samples $x(t_i)$ at times $t_0 < t_1 < \cdots < t_N$. This gap between continuous reality and discrete observation is fundamental to any practical trajectory model:

1. *Discrete snapshots of continuous motion.* No matter how high-frequency the sensor, we only ever see a finite list of points $x(t_i)$. Inferring the unobserved path between samples requires modeling assumptions.

2. *Interpolation vs. missing segments.* If two consecutive samples satisfy $\Delta t = t_{i+1} - t_i <$

τ_0—where τ_0 is a small, problem-dependent threshold—one can reasonably interpolate between $x(t_i)$ and $x(t_{i+1})$. But if $\Delta t \geq \tau_0$, the intervening path is too uncertain, so we treat that interval as missing. Thus τ_0 encodes both agent-dynamics and sensor fidelity.

3. *Thresholds vary by context.* There is no universal τ_0. For example, τ_0 might be milliseconds for pedestrian cameras but seconds for aircraft radar. Defining τ_0 and handling data gaps heuristically highlights the need for models that natively manage continuous dynamics and irregular or incomplete observations.

These considerations motivate everything from simple finite-lag schemes to full infinite-history models. We now formalize the finite-lag notion via jets.

4.2.2 The Jet Bundle Concept

To move beyond a purely instantaneous view of an agent's state $x(t) \in \mathbb{R}^d$, we can enrich it by incorporating information about its local time evolution, such as its derivatives. This leads to the concept of a jet, which is naturally framed within the mathematical language of fiber bundles, where the state itself includes not just the position but also its temporal derivatives.

The foundational idea is that a finite history, typically comprising $N + 1$ discrete samples such as $x(t_0), x(t_0 - h), \ldots, x(t_0 - Nh)$, contains sufficient information to uniquely specify an $(N + 1)$-component local state representation at the current time t_0. This local state is conceptualized as a **jet of order** N (or simply N-jet), denoted $\hat{\mathbf{j}}(t_0) = \big(x(t_0), \hat{x}'(t_0), \ldots, \hat{x}^{(N)}(t_0)\big)^T$. Formally, such an N-jet is an element of the **jet bundle** $J^N(\mathbb{R}, \mathbb{R}^d)$, the space of N-jets of curves from $\mathbb{R}$ (time) to $\mathbb{R}^d$ (configuration space). In this framework, the current position $x(t_0) = \hat{x}(t_0)$ serves as the base point of the jet in $\mathbb{R}^d$. The higher-order derivative estimates $\big(\hat{x}'(t_0), \ldots, \hat{x}^{(N)}(t_0)\big)^T$

constitute the **fiber** over that base point, capturing local geometric information about the trajectory. For mathematical rigor and precision, the definitions below are adapted from the foundational exposition in [80].

Definition 4.2.1 (r-jet of a map from $\mathbb{R}$ to $\mathbb{R}^d$)**.** *Let $\gamma : \mathbb{R} \to \mathbb{R}^d$ be a smooth map (a curve). Two such maps γ_1, γ_2 are said to have the same r-**jet** at $t \in \mathbb{R}$ if they have the same value at t and all their derivatives up to order r agree at t. That is, $\gamma_1(t) = \gamma_2(t)$, $\gamma_1'(t) = \gamma_2'(t)$, ..., $\gamma_1^{(r)}(t) = \gamma_2^{(r)}(t)$. The equivalence class of γ under this relation is called the r-jet of γ at t, denoted $j_t^r\gamma$. It can be identified with the tuple $(t, \gamma(t), \gamma'(t), \ldots, \gamma^{(r)}(t))$ or, if t is implicit, as $(\gamma(t), \gamma'(t), \ldots, \gamma^{(r)}(t))$.*

Definition 4.2.2 (Jet Bundle $J^r(\mathbb{R}, \mathbb{R}^d)$)**.** *The set of all r-jets of smooth maps from $\mathbb{R}$ to $\mathbb{R}^d$ is denoted by $J^r(\mathbb{R}, \mathbb{R}^d)$. This set has a natural manifold structure and is called the **r-jet bundle of maps from $\mathbb{R}$ to $\mathbb{R}^d$**. An element of $J^r(\mathbb{R}, \mathbb{R}^d)$ is typically of the form $j_t^r\gamma$, representing the r-jet of some curve γ at some time t. The projection map $\pi : J^r(\mathbb{R}, \mathbb{R}^d) \to \mathbb{R}$ given by $\pi(j_t^r\gamma) = t$ defines it as a fiber bundle over $\mathbb{R}$ (the source space). Alternatively, it can be viewed as a bundle over $\mathbb{R} \times \mathbb{R}^d$ with projection $\pi_0(j_t^r\gamma) = (t, \gamma(t))$.*

Definition 4.2.3 (Jet Fiber $J_t^r(\mathbb{R}, \mathbb{R}^d)$)**.** *For a fixed $t \in \mathbb{R}$, the set of all r-jets of smooth maps from $\mathbb{R}$ to $\mathbb{R}^d$ at this specific point t is denoted by $J_t^r(\mathbb{R}, \mathbb{R}^d)$. This is the fiber of the jet bundle $J^r(\mathbb{R}, \mathbb{R}^d)$ over the point $t \in \mathbb{R}$ (if considering the source projection) or over $(t, x) \in \mathbb{R} \times \mathbb{R}^d$ where $x = \gamma(t)$ (if considering the source-target projection). An element $j_t^r\gamma \in J_t^r(\mathbb{R}, \mathbb{R}^d)$ can be represented by the tuple $(\gamma(t), \gamma'(t), \ldots, \gamma^{(r)}(t))$, and thus $J_t^r(\mathbb{R}, \mathbb{R}^d)$ is isomorphic to $\mathbb{R}^d \times (\mathbb{R}^d)^r \cong (\mathbb{R}^d)^{r+1}$.*

Definition 4.2.4 (r-th Jet Prolongation)**.** *Given a smooth map (curve) $\gamma : \mathbb{R} \to \mathbb{R}^d$, its **r-th jet prolongation** is a map $j^r\gamma : \mathbb{R} \to J^r(\mathbb{R}, \mathbb{R}^d)$ defined by $t \mapsto j_t^r\gamma$. This map assigns to each $t \in \mathbb{R}$*

the r-jet of γ at t. The r-th jet prolongation $j^r\gamma$ is a section of the jet bundle $J^r(\mathbb{R}, \mathbb{R}^d)$.

Crucially, an invertible linear transformation—a one-to-one correspondence—generally exists between the $(N+1)$-sample history vector and such an $(N+1)$-component N-jet vector. This implies that these two representations are, in principle, informationally equivalent, merely expressing the same local trajectory information in different bases. The specific method chosen to establish this mapping influences the precise properties of the resulting jet.

4.2.2.1 A Concrete Realization: Jet Fiber Construction via Polynomial Interpolation

To make the concept of the jet tangible, particularly the construction of its fiber (the derivative components) from discrete samples, we briefly outline one standard and mathematically rigorous method. This approach uses polynomial interpolation and inherently involves Vandermonde matrices. It serves as an important illustrative example, even if it's not the method ultimately employed in a specific model, because it grounds the abstract notion of a jet.

Given $\mathbf{x}_{\text{samples}} = \big(x(t_0), x(t_1), \ldots, x(t_{-N})\big)^T$, where $t_{-i} = t_0 - ih$, a unique polynomial $P(\tau) = a_0 + a_1\tau + \cdots + a_N\tau^N$ (with $\tau = t - t_0$) of degree N is fitted to these $N+1$ points. The determination of the polynomial coefficients $\mathbf{a} = (a_0, \ldots, a_N)^T$ from $\mathbf{x}_{\text{samples}}$ relies on the invertibility of the $(N+1) \times (N+1)$ Vandermonde matrix W (where $W_{ij} = (-ih)^j$), such that $\mathbf{a} = W^{-1}\mathbf{x}_{\text{samples}}$.

The derivative estimates are then $\hat{x}^{(j)}(t_0) = P^{(j)}(0) = j!a_j$, forming the N-jet $\hat{\mathbf{j}} = \big(\hat{x}^{(0)}, \ldots, \hat{x}^{(N)}\big)^T$. The overall transformation from samples to this specific jet can be written as $\hat{\mathbf{j}} = (D_{\text{poly}}W^{-1})\,\mathbf{x}_{\text{samples}}$, where D_{poly} is a diagonal matrix with entries $j!$. For example, con-

structing a 1-jet (position and velocity) from 2 samples $\big(x(t_0), x(t_0 - h)\big)$ yields

$$\hat{x}^{(1)}(t_0) = \frac{x(t_0) - x(t_0 - h)}{h} + \mathcal{O}(h).$$

Constructing a 2-jet from 3 samples yields formulas such as

$$\hat{x}^{(2)}(t_0) = \frac{1}{h^2}\left(x(t_0) - 2x(t_0 - h) + x(t_0 - 2h)\right) + \mathcal{O}(h).$$

The inverse transformation, $\mathbf{x}_{\text{samples}} = \left(W D_{\text{poly}}^{-1}\right)\hat{\mathbf{j}} + \mathcal{O}(h^{N+1})$, demonstrates how the sample values can be reconstructed from the jet using truncated Taylor series. The matrix $M_{\text{Taylor}} = W D_{\text{poly}}^{-1}$ has entries

$$(M_{\text{Taylor}})_{im} = \frac{(-ih)^m}{m!}$$

(for the i-th sample $x(t_0 - ih)$ and m-th jet component $\hat{x}^{(m)}(t_0)$ with $m \leq N$). This relationship underscores the role of the jet in capturing information sufficient for local trajectory reconstruction.

The vector $\hat{\mathbf{j}}(t_0)$, defined as an N-jet (order N, with $N + 1$ components), serves as an element in $J^N(\mathbb{R}, \mathbb{R}^d)$. It provides a local state representation where the position $x(t_0)$ is directly available as its zeroth-order component, $\hat{x}^{(0)}(t_0)$.

While the principle of an invertible map between $N+1$ samples and an $(N+1)$-component local state (like an N-jet) is general, the specific values, accuracy (e.g., error terms $\mathcal{O}(h^p)$ relative to true derivatives), and stability of the derivative estimates heavily depend on the chosen numerical implementation. The polynomial interpolation method offers one consistent and prin-

cipled way to define these properties. It is important to note that this method, in its simplest form using fixed-stencil formulas, assumes regular sampling; irregular sampling makes the reliable estimation of a full, high-order jet more complex, often requiring adaptive methods or limiting the reliably estimable derivative order.

The concept of jets, particularly as structured representations of derivatives, is relevant to certain continuous-time models such as augmented [81] or higher-order [82] Neural ODEs, where the evolving state explicitly includes such derivative-like quantities. However, as a general means of summarizing history for trajectory prediction, any fixed-order jet bundle representation faces fundamental conceptual limitations:

- **Fixed Order (N)**: The choice of N (the highest derivative order) is often heuristic and may not adequately match the true complexity or memory depth of the system being modeled.
- **Finite Window ($(N+1)h$)**: Information is drawn only from a finite, recent segment of the trajectory, potentially discarding older but still relevant historical information. This limitation is exacerbated by irregular sampling, where defining a consistent "window" or ensuring sufficient data for high-order derivatives becomes difficult.
- **Expressivity for Complex Dynamics**: While offering a richer state than position alone, a finite N-jet may struggle to adequately represent trajectories with highly non-polynomial local behavior or systems governed by long-range temporal dependencies.

These conceptual limitations primarily motivate the subsequent shift towards more flexible and encompassing infinite-history representations.

4.2.3 Dynamics Lifted to the Jet Bundle

When an N-jet $\hat{\mathbf{j}}(t) = \big(\hat{x}^{(0)}(t), \ldots, \hat{x}^{(N)}(t)\big)^T$ is used as the state, the system's dynamics are lifted from the base configuration space $\mathbb{R}^d$ as an N-th jet prolongation to evolve on the jet bundle $J^N(\mathbb{R}, \mathbb{R}^d)$ (or, more precisely, a state space isomorphic to the fibers $J^N_t \cong (\mathbb{R}^d)^{N+1}$).

The evolution of the entire jet state can be described by a system of ordinary differential equations:

$$\frac{d}{dt}\hat{\mathbf{j}}(t) = \mathbf{F}\big(t, \hat{\mathbf{j}}(t)\big).$$

This system inherently respects the structure of the jet:

$$\frac{d}{dt}\hat{x}^{(i)}(t) = \hat{x}^{(i+1)}(t) \quad \text{for } 0 \leq i < N.$$

The dynamics of the highest-order component, $\frac{d}{dt}\hat{x}^{(N)}(t)$, are then given by

$$F_{\text{model}}\big(t, \hat{x}^{(0)}(t), \ldots, \hat{x}^{(N)}(t)\big),$$

where F_{model} is derived from underlying physical assumptions or, more commonly in machine learning contexts, is a learned function (e.g., a neural network). This structure is characteristic of how higher-order ODEs are expressed as first-order systems on an augmented state space, a technique also employed in higher-order Neural ODEs.

A key architectural feature of this lifted dynamic is that the observable $\mathbb{R}^d$ position, $x(t) = \hat{x}^{(0)}(t)$, is an integral component of the state vector $\hat{\mathbf{j}}(t)$ that the ODE evolves. Thus, the current position can be directly read off from the state at any time, without requiring a separate projection

or readout mapping from a purely latent or hidden space to the observation space $\mathbb{R}^d$.

If the dynamics are time-homogeneous (i.e., $\mathbf{F}$ does not explicitly depend on t, so $\frac{d}{dt}\hat{\mathbf{j}}(t) = \mathbf{F}(\hat{\mathbf{j}}(t))$), then the rules governing the evolution of the jet components depend only on the current values of these components themselves. In such cases, while the full jet prolongation $t \mapsto j_t^N x$ is a section of the bundle $J^N(\mathbb{R}, \mathbb{R}^d)$ fibered over time, the vector field defining the dynamics on the components $(\hat{x}^{(0)}, \dots, \hat{x}^{(N)})$ is fixed. Thus, for analysis of these rules, one can consider the dynamics as occurring in a fixed state space $\mathcal{J}$ isomorphic to any fiber $J_t^N(\mathbb{R}, \mathbb{R}^d)$ (e.g., canonically, $J_0^N(\mathbb{R}, \mathbb{R}^d)$ or simply $(\mathbb{R}^d)^{N+1}$).

Despite providing a richer state than position alone, the finite-order (N) nature of the jet bundle means it still offers only a localized approximation of the system's true state, which might be influenced by a history far more extensive than what can be encoded in N derivatives. This motivates moving towards truly infinite-history representations.

4.2.4 Conceptual Ideal: Padded Infinite-Order Jets ($\widetilde{J}_0^\infty$)

We can conceive of a padded infinite-order jet at a point t_0 as an element of a space we denote $\widetilde{J}_0^\infty(\mathbb{R}, \mathbb{R}^d)$. Formally, this space can be understood as the *categorical direct limit* of the sequence of padded finite-order jet spaces $\left(\widetilde{J}_0^N(\mathbb{R}, \mathbb{R}^d)\right)_{N \geq 0}$ under natural inclusion maps

$$\iota_{NM} : \widetilde{J}_0^N \hookrightarrow \widetilde{J}_0^M \quad \text{for } N \leq M.$$

These inclusions map an N-jet $(\xi_0, \dots, \xi_N)$ to an M-jet $(\xi_0, \dots, \xi_N, \perp, \dots, \perp)$ by padding the higher-order derivatives with a "missing" symbol $\perp$. An element of this direct limit $\widetilde{J}_0^\infty(\mathbb{R}, \mathbb{R}^d)$ is thus an infinite sequence $(\xi_0, \xi_1, \xi_2, \dots)$, where each ξ_j is either an $\mathbb{R}^d$ vector or $\perp$. Crucially,

due to the direct limit construction, for any specific element in this space, only a finite number of initial components $\xi_0, \ldots, \xi_{N_{\text{eff}}}$ can be actual $\mathbb{R}^d$ values; all subsequent components ξ_j for $j > N_{\text{eff}}$ must be $\perp$. There are no "holes" of $\perp$ values within the defined prefix $\xi_0, \ldots, \xi_{N_{\text{eff}}}$.

The primary utility of $\widetilde{J}_0^\infty(\mathbb{R}, \mathbb{R}^d)$ is as a *universal conceptual space for local pointwise information*. It provides a common theoretical ground to embed and compare the local state descriptions of different trajectories, even if they are derived from histories of vastly different effective depths (i.e., different N_{eff} values). This is especially valuable when comparing datasets with varying amounts of data, as it enables all available local derivative information for any agent to be expressed within a common maximal structure—differing only in the extent of the initial non-$\perp$ segment.

Given the impracticality of directly working with infinite sequences, we typically operate with padded N-jets as practical, finite-dimensional representations. The space for such jets, $\widetilde{J}_0^N(\mathbb{R}, \mathbb{R}^d)$, contains elements $(\xi_0, \xi_1, \ldots, \xi_N)$. Each element is characterized by an effective order $m \leq N$, such that components $\xi_0, \ldots, \xi_m$ are $\mathbb{R}^d$-valued (representing $\hat{x}^{(0)}(t_0), \ldots, \hat{x}^{(m)}(t_0)$), while the remaining components $\xi_{m+1}, \ldots, \xi_N$ are $\perp$. Here, N is a chosen maximal derivative order considered relevant or practically estimable.

This structure directly addresses varying data availability. If an agent's observed history only permits the reliable estimation of derivatives up to order $m < N$, then the components $\xi_{m+1}, \ldots, \xi_N$ are $\perp$. A newly observed agent might only have a defined $\xi_0 = x(t_0)$, with all $\xi_j = \perp$ for $j \in \{1, \ldots, N\}$.

Even the conceptually complete $\widetilde{J}_0^\infty(\mathbb{R}, \mathbb{R}^d)$, along with its practical truncation $\widetilde{J}_0^N(\mathbb{R}, \mathbb{R}^d)$, is fundamentally a *pointwise* descriptor of local information at t_0. This inherent characteristic leads to significant limitations:

1. **Incomputability of Infinite Non-$\perp$ Terms**: While $\widetilde{J}_0^\infty(\mathbb{R}, \mathbb{R}^d)$ allows infinite sequences, deriving an infinite number of *meaningful*, non-$\perp$ derivatives from finite data is impossible. The direct limit construction already ensures elements are effectively finite in their non-$\perp$ extent.

2. **Non-Analyticity of Trajectories**: Most real-world trajectories are not analytic. Thus, even a perfectly known infinite sequence of derivatives at t_0 (an element of the classical unpadded J_0^∞ as an inverse limit) would only define the function in a potentially very small neighborhood via its Taylor series and would fail for global behavior or non-analytic features.

3. **Unreliability from Data Gaps/Irregularity**: Even for derivatives of a theoretically computable order $j \leq N_{\text{eff}}$ (the effective order of an element in $\widetilde{J}_0^\infty(\mathbb{R}, \mathbb{R}^d)$), significant gaps between timestamps or severe irregularities in the underlying sampled data can render numerical approximations highly unreliable. This practical unreliability would necessitate treating these derivative estimates as $\perp$, further reducing the number of useful non-$\perp$ components in any jet derived from such imperfect data.

These collective limitations underscore that while padded jets offer a structured way to handle incomplete local derivative information, a more robust and data-faithful approach for systems with memory is needed. This motivates the transition to representations based on history *functions* over an interval, which can directly utilize raw data segments.

Remark 4.2.5 (Contrast with Classical Infinite Jets.)**.** *In contrast to our padded-jet construction—defined as the direct limit of finite–order jet spaces—the classical infinite jet space $J^\infty(\mathbb{R}, \mathbb{R}^d)$ is typically introduced via the inverse limit of the sequence $\{J_0^N(\mathbb{R}, \mathbb{R}^d)\}_{N\geq 0}$. An element of J^∞*

corresponds to a full Taylor series at a single timepoint and thus presupposes the existence of all derivatives of the trajectory. While this inverse-limit formulation is elegant in a smooth, analytic setting, it is ill-suited for most real-world trajectory data, which frequently lack sufficient smoothness or complete derivative information. Our padded construction $\widetilde{J}_0^\infty$, by contrast, accommodates exactly that kind of incompleteness: no finite or infinite set of derivatives is required, and each missing component is simply marked by $\perp$. Consequently, $\widetilde{J}_0^\infty$ remains well-defined and meaningful even when only partial, discrete trajectory observations are available.

4.2.5 Infinite History (Function Space Bundles)

Finite-order jet bundles, and even the theoretical constructs like $\widetilde{J}_0^\infty(\mathbb{R}, \mathbb{R}^d)$, ultimately focus on information localized at a single point in time t_0. To robustly capture long-range memory effects, complex temporal dependencies, and the influence of past events over extended durations—all crucial for the continuous interactive dynamics—we transition to considering the state as the history function itself. This function can be viewed as an element of an infinite-history bundle or a state space constructed over $\mathbb{R}^d$.

The state of the system at time t is represented by the entire relevant past trajectory segment, $x_t : (-\infty, 0] \to \mathbb{R}^d$, where $x_t(\tau) = x(t + \tau)$ for $\tau \leq 0$. This function x_t encapsulates the path leading up to the present. The space of such history functions ending at time t, say $\mathcal{H}_t$, is isomorphic for all t to a canonical history space, such as functions defined on $(-\infty, 0]$. We denote this canonical space by $\mathcal{H}_0 \equiv \widetilde{\mathcal{C}}((-\infty, 0], \mathbb{R}^d)$, which will be defined more formally in the next subsection.

Within this history bundle paradigm, the current $\mathbb{R}^d$ position $x(t)$ is directly obtained as

$x_t(0)$—this is the base point of the history function in the bundle. The entire function $x_t(\tau)$ for $\tau < 0$ constitutes the fiber, carrying all available information about the past evolution.

Dynamics governed by such history functions are described by Retarded Functional Differential Equations (RFDEs). Given the isomorphism of history spaces and assuming the underlying physical process is *time-homogeneous*, the RFDE can be written in an *autonomous form*:

$$\dot{x}(t) = F(x_t).$$

Here, F is a functional that maps from the canonical history space $\mathcal{H}_0$ to $\mathbb{R}^d$, and it does not explicitly depend on t. This formulation directly supports the modeling of systems whose future evolution depends on an extended segment of their past, aligning with the principle of time-homogeneity with causality.

4.2.6 Padded Partial Trajectories in History Bundles

To practically implement the infinite-history concept using real-world data—which is often incomplete due to irregular sampling, occlusions, or boundary events (leading to spatial truncation or temporal censoring)—we define a space of padded partial trajectories. This space, $\widetilde{\mathcal{C}}((-\infty, 0], \mathbb{R}^d)$, serves as our canonical history space $\mathcal{H}_0$.

The history function x_t may not be continuously known or observed over the entire semi-infinite interval $(-\infty, 0]$. We require a representation that explicitly handles these discontinuities and unobserved segments, a direct consequence of the sensing limitations.

The space $\mathcal{H}_0 \equiv \widetilde{\mathcal{C}}((-\infty, 0], \mathbb{R}^d)$ is introduced as the set of functions $\gamma : (-\infty, 0] \to \mathbb{R}^d \cup \{\perp\}$. An element γ in this space is $\mathbb{R}^d$-valued and continuous only on a finite union of

disjoint closed subintervals $\bigcup_{j=1}^{M}[a_j, b_j]$ within $(-\infty, 0]$ (where M is the number of observed segments). For all s outside these observed segments, $\gamma(s) = \perp$. The bundle-like structure $\mathcal{H}_0 \to \mathbb{R} \cup \{\perp\}$ is equipped with a projection, $\gamma \mapsto \gamma(0)$, assigning to each history its current value. Note that $\gamma(0)$—the current value—may be missing (i.e., equal to $\perp$). Restricting to the subset of the base space where $\gamma(0) \neq \perp$, the structure becomes a genuine fiber bundle. An element $x_t \in \mathcal{H}_0$ *is* the state in this padded history bundle.

The current position $x(t)$ is given by $x_t(0)$. The past history $x_t(s)_{s<0}$ explicitly marks unobserved segments with $\perp$. This formalism directly encodes where trajectory data is reliable and where it is missing due to occlusions, boundary effects, or other sensing limitations. Notably, allowing $\perp$ in the base means that a history ending in $\gamma(0) = \perp$ still retain informative past segments with $\gamma(s) \neq \perp$ for $s < 0$. Even if an agent is absent at the prediction start, its partial trajectory—recorded over the observed subintervals—remains part of the state space and may still influence the evolution of other agents' dynamics. By contrast, in jet-based representations, any jet at time t would vanish for $\gamma(0) = \perp$, erasing all memory of that agent's prior trajectory. In $\mathcal{H}_0$, however, the padded history up to $s < 0$ survives, so the functional $F(x_t)$ can legitimately incorporate—even after departure—whatever influence that agent's earlier behavior has on the present and future trajectories of the agents still in view. Crucially, this representation operates directly on raw (or minimally processed) samples over intervals, rather than relying on derived quantities such as pointwise derivatives, making it well-suited for modeling potentially non-analytic, real-world trajectory data.

This framework is important for developing models that are robust to the diverse imperfections of real-world observational data, allowing the dynamic functional $F(x_t)$ to potentially learn to operate over, or account for, these explicitly marked missing segments.

4.2.7 Causal (Time) Equivariance in History Bundles

A fundamental principle for predictive models operating on temporal data is time-homogeneity with causality, ensuring that predictions are derived consistently across different time origins and strictly utilize only past information. This forms a key component of the causal-pointwise equivariance principle.

Let $f \in \mathcal{H}_0$ represent a history function (element of the canonical history space $\mathcal{H}_0$, corresponding to some x_t). A temporal shift operator τ_a (for $a > 0$) acting on the underlying timeline data can be thought of as producing a new history element x_{t-a} which, when mapped to $\mathcal{H}_0$, represents the history of the process as it would have appeared a time units prior.

A predictor $\Phi : \mathcal{H}_0 \to \mathbb{R}^d$ (where F from the RFDE is a component of Φ) is considered time-equivariant if its output is consistent with such time shifts. That is, the rules applied to x_t are the same as those applied to x_{t-a} (when both are viewed as elements in $\mathcal{H}_0$). This ensures that the model learns underlying dynamic patterns that are invariant to absolute time, a form of $\mathbb{R}$-equivariance under causal time shifts. The autonomous form of the RFDE,

$$\dot{x}(t) = F(x_t),$$

inherently respects this: $\tau_a\big(\Phi\big(x(\cdot)\big)\big) = \Phi\big(\tau_a\big(x(\cdot)\big)\big)$.

This property is essential for models aiming to learn generalizable temporal patterns from diverse trajectory segments observed at different points in time.

4.3 Retarded Functional Differential Equations

Previous sections of this document have explored discrete-time models, some of which inherently possess memory, whether implicitly or explicitly (e.g., kth order recurrence relation in §4.1, or the hidden states within RNNs). Subsequently, §4.2 established the motivation for continuous-time representations capable of capturing a richer notion of history than what can be described by instantaneous states or their finite-order derivatives (jets). This section introduces a class of continuous-time models known as Retarded Functional Differential Equations (RFDEs). In these systems, the instantaneous rate of change of a state, $\dot{x}(t)$, is not merely a function of the current state $x(t)$ but depends on a segment of its past states. This formulation allows for the modeling of systems with delayed dynamics, where the past history continuously influences the future trajectory, moving beyond purely instantaneous cause-and-effect relationships.

4.3.1 Delay Differential Equations

The journey from Ordinary Differential Equations (ODEs) to a functional dependence on history segments often begins with simpler extensions.

- **Discrete Delay Differential Equations (DDEs):**

  As a first step beyond ODEs, classical DDEs [83] introduce delays by allowing the dynamics to depend on states at specific, fixed past times. Neural Delay Differential Equations (NDDEs) [84] directly model systems where the derivative of a variable at time t is affected not only by the current state but also by states at previous time instants. Their general form

can be written as:

$$\dot{x}(t) = f\big(t, x(t), x(t-\tau_1), \ldots, x(t-\tau_k)\big)$$

where $\tau_1, \ldots, \tau_k$ are positive constants representing fixed delays. DDEs are useful in contexts where it is known that specific past moments have a dominant influence. For example, in population dynamics, the growth rate of a species can depend on the population size at a previous time, reflecting a maturation period or the time required for resource regeneration. However, DDEs are limited if the entire history over a continuous interval, rather than just a few discrete past points, is influential in shaping the dynamics.

- **Motivation for Functional Dependence on History Segments:**

  In many complex systems—such as population models where the cumulative resource history affects growth rates, or gesture-recognition tasks where an entire past segment informs ongoing movement—the present state cannot be captured by a single point in $\mathbb{R}^d$. A connection is presented in [85] between time delay systems and time delay neural networks (TDNNs) from a continuous-time perspective, where TDNNs are used to learn the nonlinear dynamics of time delay systems from trajectory data. This perspective naturally leads to a formulation where the state itself, at any given time, is not just a point in, but a function representing a segment of the trajectory's recent past. This shift is important because, as highlighted by [81], traditional Neural ODEs have limitations in representing some functions, particularly those where trajectories in lower-dimensional phase space might intersect—a characteristic that DDEs can model.

4.3.2 RFDEs with Finite Delay Intervals

When the influence of the past is considered to be confined to a finite, bounded duration, we enter the realm of **Retarded Functional Differential Equations (RFDEs)** with finite delay intervals. This class of equations provides a precise mathematical framework for systems where memory has a limited depth.

In this context, the state of the system at any time t is no longer just its instantaneous configuration in $\mathbb{R}^d$, but is instead a function that encapsulates its recent past. This history function, denoted as x_t, is defined as a segment of the path traced by x. Specifically, $x_t : [-r, 0] \to \mathbb{R}^d$, where $x_t(s) = x(t + s)$ for all $s \in [-r, 0]$. Here, $r > 0$ is a positive constant representing the finite maximum delay, indicating that only the behavior of the system over the interval $[t - r, t]$ directly influences its future evolution. An important aspect of this representation is that the instantaneous value $x(t)$ is explicitly included as part of the state x_t, since $x(t) = x_t(0)$. The space of such history functions is typically taken as $C([-r, 0], \mathbb{R}^d)$, the space of continuous functions mapping $[-r, 0]$ into $\mathbb{R}^d$.

A Retarded Functional Differential Equation with a finite delay is then an equation where the derivative of $x(t)$ depends on the history function x_t. It is formally expressed as:

$$\dot{x}(t) = f(t, x_t)$$

where f is a functional that maps the current time t and the history segment $x_t \in C([-r, 0], \mathbb{R}^d)$ to a vector in $\mathbb{R}^d$, which represents the instantaneous velocity of the system.

For the purpose of learning dynamical laws that are invariant over time, a time-homogeneous

(autonomous) formulation is generally preferred:

$$\dot{x}(t) = f(x_t)$$

In this autonomous case, the functional f maps from the function space $C([-r, 0], \mathbb{R}^d)$ to $\mathbb{R}^d$.

The theoretical foundation for RFDEs with finite delay addresses their well-posedness: whether solutions exist, are unique for given initial conditions, and how they depend on these conditions. An initial condition for such an RFDE at an initial time σ is specified by an initial function $\phi \in C([-r, 0], \mathbb{R}^d)$, defining $x_\sigma = \phi$.

The fundamental results that establish these properties under appropriate conditions on the function f are central to the theory of RFDEs.

Theorem 4.3.1 (Existence of Solutions for RFDEs)**.** *Let Ω be an open subset of $\mathbb{R} \times C([-r, 0], \mathbb{R}^n)$, and consider the Retarded Functional Differential Equation (RFDE) $\dot{x}(t) = f(t, x_t)$, where the function $f : \Omega \to \mathbb{R}^n$ is continuous.*

1. *If (σ, ϕ) is any point in Ω, then there exists a solution x of the RFDE passing through (σ, ϕ), defined on some interval $[\sigma, \sigma + \alpha)$ for some $\alpha > 0$.*

2. *More generally, if W is any compact subset of Ω, there exists a time $T > 0$ (depending on W) such that for every initial condition $(\sigma, \phi) \in W$, there is a solution x of the RFDE passing through (σ, ϕ) defined on the interval $[\sigma, \sigma + T]$.*

Proof. See [86], Theorem 2.1. in Chapter 2. □

Theorem 4.3.2 (Uniqueness of Solutions for RFDEs)**.** *Let Ω be an open subset of $\mathbb{R} \times C([-r, 0], \mathbb{R}^n)$, and consider the RFDE $\dot{x}(t) = f(t, x_t)$. Assume that the function $f : \Omega \to \mathbb{R}^n$ is continuous.*

Furthermore, assume that $f(t, \phi)$ is locally Lipschitz continuous with respect to ϕ in Ω; that is, for every compact set $K \subset \Omega$, there exists a constant L_K such that

$$\|f(t, \psi) - f(t, \chi)\|_{\mathbb{R}^n} \leq L_K \|\psi - \chi\|_C \quad \textit{for all } (t, \psi), (t, \chi) \in K.$$

If (σ, ϕ) is any point in Ω, then there exists a unique solution x of the RFDE passing through (σ, ϕ) on its interval of existence.

Proof. See [86], Theorem 2.3. in Chapter 2. □

Theorem 4.3.3 (Continuous Dependence of Solutions for RFDEs)**.** *Let Ω be an open subset of $\mathbb{R} \times C([-r, 0], \mathbb{R}^n)$. Consider a sequence of functions $\{f_k\}_{k=0}^{\infty}$, where each $f_k : \Omega \to \mathbb{R}^n$ is continuous. Assume that the sequence of functions f_k converges to f_0 as $k \to \infty$, uniformly on compact subsets of Ω.*

Let $\{(\sigma_k, \phi_k)\}_{k=0}^{\infty}$ be a sequence of initial conditions within Ω such that $(\sigma_k, \phi_k) \to (\sigma_0, \phi_0) \in \Omega$ as $k \to \infty$.

For each $k \geq 1$, let x_k be a solution of the RFDE $\dot{x}(t) = f_k(t, x_{k,t})$ passing through (σ_k, ϕ_k), defined on some time interval $[\sigma_k, b_k)$.

Suppose that the set of solution values $\{x_k(t) : t \in [\sigma_k, b_k) \text{ for all } k\}$ forms a precompact subset of $\mathbb{R}^n$, and similarly, the set of solution history segments $\{x_{k,t} : t \in [\sigma_k, b_k) \text{ for all } k\}$ forms a precompact subset of $C([-r, 0], \mathbb{R}^n)$.

Under these conditions, there exists a subsequence of solutions $\{x_{k_j}\}_{j=1}^{\infty}$ and a solution x_0 of the limiting RFDE $\dot{x}(t) = f_0(t, x_{0,t})$ passing through (σ_0, ϕ_0), such that the subsequence x_{k_j} converges to x_0. This convergence is uniform on compact subsets of the maximal interval of existence of x_0.

Proof. See [86], Theorem 2.2. in Chapter 2. □

These theorems ensure that, under conditions of continuity and (for uniqueness and continuous dependence) local Lipschitz continuity of f, solutions to RFDEs are well-behaved, making them suitable for modeling and analysis. The theory also extends to the continuation of solutions, addressing the maximal interval of their existence.

RFDEs with finite delay are particularly relevant for modeling systems where there is a clear, identifiable maximum duration for memory effects. This could include systems with explicit signal transmission delays, fixed processing times, biological processes with known maturation periods where influences beyond this period are negligible, or various engineering control systems.

4.3.3 RFDEs with Infinite Delay Intervals

In some systems, the influence of past states may extend arbitrarily far back in time, necessitating a generalization to infinite delay intervals. Generalizing the delay interval to $(-\infty, 0]$ allows for the modeling of systems with arbitrarily long-range temporal dependencies, as foreshadowed in §4.2.5. This is crucial for systems where memory effects do not have a fixed cutoff or where the extent of relevant history is unknown or varies.

The general RFDE formulation remains:

$$\dot{x}(t) = F(x_t)$$

(preferring the autonomous form). However, the state x_t is now a function representing the entire past history, $x_t : (-\infty, 0] \to \mathbb{R}^d$, where $x_t(s) = x(t + s)$ for $s \in (-\infty, 0]$. The interpretation is

that the system's instantaneous rate of change can, in principle, depend on its behavior over its entire past.

The primary challenge with infinite delays is defining an appropriate function space (phase space) $\mathcal{B}$ for the history x_t such that the problem is well-posed. Standard spaces like $\mathcal{C}((-\infty, 0], \mathbb{R}^d)$ are often too large or do not lead to desirable properties for the solution operator.

An axiomatic approach, such as that outlined by [86], provides a framework for defining suitable phase spaces. These axioms typically ensure that if the initial history x_σ belongs to $\mathcal{B}$, then x_t remains in $\mathcal{B}$ for $t > \sigma$, that x_t is continuous in t with respect to the norm of $\mathcal{B}$, and that the norm itself has certain desirable properties (e.g., relating the norm of x_t to the norm of $x(t)$ and past segments).

Examples of spaces that can satisfy such axioms include "fading memory" spaces, like certain weighted L^p spaces or spaces of continuous functions $\mathcal{C}_g$ where the influence of past states diminishes over time (e.g., $\mathcal{C}_\gamma$ for exponential decay).

Within such an axiomatic framework, if the phase space $\mathcal{B}$ is appropriately chosen and the functional $F : \mathcal{B} \to \mathbb{R}^d$ satisfies suitable conditions (e.g., continuity and local Lipschitzness), then fundamental theorems regarding the local existence, uniqueness, and continuous dependence of solutions on initial data can be established.

Even when the history x_t is defined on an infinite interval, it is often natural or necessary for the functional F to exhibit a "fading memory" property. This means that the influence of past states $x(t + s)$ on $\dot{x}(t)$ diminishes as $s \to -\infty$. This property can be either built into the choice of phase space or learned by the functional F itself.

4.3.4 The Solution Operator and Semigroup Property for Autonomous RFDEs

The existence and uniqueness of solutions to autonomous Retarded Functional Differential Equations (RFDEs), whether with finite or infinite delays (within an appropriate phase space framework for the latter), allow for the definition of a solution operator that describes the system's evolution.

For an autonomous RFDE of the form $\dot{x}(t) = f(x_t)$, where f does not explicitly depend on time and satisfies conditions ensuring the existence and uniqueness of solutions (e.g., being Lipschitz continuous on its domain), we can define the solution operator $T(t)$ that maps an initial history function ϕ at time $\sigma = 0$ to the history function x_t at a subsequent time $t \geq 0$. That is,

$$T(t)\phi = x_t(0, \phi).$$

This family of operators $\{T(t) : t \geq 0\}$ possesses fundamental structural properties. Specifically, on its phase space $\mathcal{B}$ (which could be $C([-r, 0], \mathbb{R}^n)$ for finite delays, or a suitable axiomatic phase space for infinite delays), $T(t)$ forms a C_0-*semigroup* for $t \geq 0$; see [86, p. 68].

This means the solution operator satisfies the following key properties:

1. **Identity at Initial Time:** $T(0) = I$, where I is the identity operator on the phase space $\mathcal{B}$.

   This signifies that at the initial moment, the state is simply the initial condition.

2. **Semigroup Property:** $T(t + s) = T(t)T(s)$ for all $t, s \geq 0$.

   This property expresses that evolving the system for a duration s and then for a further duration t is equivalent to evolving it directly for the total duration $t + s$.

3. **Strong Continuity:** For each $\phi \in \mathcal{B}$, the map $t \mapsto T(t)\phi$ is continuous from $[0, \infty)$ into $\mathcal{B}$.

   This ensures that the state evolves smoothly in time within the chosen phase space.

The semigroup property, $T(t + s) = T(t)T(s)$ for $t, s \geq 0$, is particularly significant. It is a direct mathematical consequence of the system being both *causal* (the future depends only on the present state, which in this context is the history x_t) and *autonomous* (the rules governing the evolution, represented by f, do not change over time). This inherent consistency means that the way the system evolves from a given history state for a certain duration is independent of when that evolution begins. This characteristic can be interpreted as a form of **causal temporal equivariance** of the solution map for forward evolution in autonomous RFDE systems. It underscores that the underlying dynamical law $f(x_t)$ is time-invariant, a crucial aspect for learning and understanding fundamental system behaviors.

While this semigroup property robustly describes forward evolution, extending $T(t)$ to a full group (allowing for unique backward evolution for all $t \in \mathbb{R}$ on the entire phase space) is generally not possible for RFDEs due to the smoothing nature of the solution operator for positive time and potential issues with backward uniqueness.

4.3.5 RFDEs in the Proposed Framework

The theory of Retarded Functional Differential Equations (RFDEs) offers a robust mathematical structure for understanding and modeling systems where memory plays a crucial role. The implications of adopting this continuous-time, history-dependent perspective are significant for our proposed framework.

- **Advantages of the RFDE Framework:**

  - RFDEs provide a principled way to represent systems whose current rate of change depends on a continuous segment of their past, offering a richer modeling capacity than systems based purely on instantaneous states or discrete delays.

  - The continuous-time nature of RFDEs makes them inherently suitable for handling data that may be sampled irregularly. Furthermore, once the functional $f(x_t)$ is determined (or learned), predictions can be made at arbitrarily fine-grained future time points by integrating the equation.

  - As will be detailed in §4.4, the functional state x_t can be adapted to handle real-world data imperfections such as missing observations or truncated histories through the use of padded trajectories, making the overall approach more robust.

- **The Nature of the Functional $f(x_t)$ to Be Learned:**

  - In our framework, the central challenge and objective is to learn the functional f in the autonomous RFDE $\dot{x}(t) = f(x_t)$ directly from observed trajectory data. This f encapsulates the underlying, time-invariant dynamical laws of the system.

  - The learned functional f must respect causality, meaning the output $\dot{x}(t)$ can only depend on the history x_s for $s \leq t$. This is naturally enforced by the definition of x_t as the history segment leading up to time t.

  - The autonomous formulation $f(x_t)$ ensures that the learned dynamics are time-homogeneous, a desirable property for discovering fundamental, invariant rules governing the system.

- **Forward Pointer to Learning Architecture and Universal Approximation:**
  - Having established the mathematical form of RFDEs and the nature of their state representation (as history functions x_t), the subsequent section (§4.4) will address the practical aspects of learning f. This will involve discussing a canonical representation for histories that accommodates real-world data (padded partial trajectories in $\mathcal{H}_0$) and exploring the theoretical basis that allows neural network architectures to approximate such functionals. Specifically, we will touch upon universal approximation theorems in the context of learning operators or functionals that map from a function space (like $\mathcal{H}_0$) to $\mathbb{R}^d$. This sets the stage for introducing the specific learnable models.
  - The mechanisms by which these learned functionals can incorporate more complex structures, such as multi-agent interactions, will be deferred to Chapter 5.

4.4 Universal Approximation of Temporally Equivariant Operators by Convolutional Networks

A core principle underlying many dynamical systems is that their governing laws are constant over time or space. As established in §4.3.5, the solution operator of an autonomous RFDE exhibits this property as time-translation equivariance. This concept is fundamental not only in time-series analysis but also in a wide range of other domains, such as image processing, where the features of an object are independent of its location.

Building a learning architecture that inherently respects this equivariance provides a powerful inductive bias. It constrains the model to a class of functions relevant to the problem, improving sample efficiency and generalization. This section formally presents the universal ap-

proximation theorem that justifies a specific class of equivariant models—namely, Convolutional Neural Networks (CNNs)—as a principled choice for this task.

4.4.1 Formalizing Equivariant Operators

While our primary application to RFDEs concerns causal operators on time-series data (the $n = 1$ case), the underlying principle of translation equivariance is more general. For the sake of theoretical completeness and to highlight the broad applicability of this framework to other domains like computer vision, we will state and discuss the universal approximation theorem in the general setting of $\mathbb{R}^n$.

To formalize the central property, we first define the group action of translation. Let T_v be the translation operator for any vector $v \in \mathbb{R}^n$, defined by its action on a function f as:

$$(T_v f)(x) = f(x - v)$$

An operator Φ that maps between function spaces defined on $\mathbb{R}^n$ is said to be $\mathbb{R}^n$**-equivariant** if it commutes with this translation operator for all valid inputs. Mathematically, this property is expressed as:

$$\Phi(T_v f) = T_v(\Phi(f)) \quad \text{for all } f \text{ in the domain of } \Phi \text{ and all } v \in \mathbb{R}^n$$

This property ensures that if an input is shifted, the output is also identically shifted, without any other change to its structure. This is the foundational symmetry that convolutional architectures are designed to preserve.

4.4.2 A Universal Approximation of $\mathbb{R}^n$-Equivariant Operators

The central theoretical result is that any well-behaved equivariant operator can be approximated by a two-layer convolutional network.

Theorem 4.4.1. *Let $\Phi : C_0(\mathbb{R}^n, \mathbb{R}^{d_{in}}) \to C_0(\mathbb{R}^n, \mathbb{R}^{d_{out}})$ be an $\mathbb{R}^n$-equivariant and Lipschitz continuous map.*

Then for any compact set of input functions $E \subset C_0(\mathbb{R}^n, \mathbb{R}^{d_{in}})$ and any $\epsilon > 0$, there exist:

1. *An integer bottleneck dimension $b > 0$.*
2. *Continuous matrix-valued convolution kernels $A_1 : \mathbb{R}^n \to \mathbb{R}^{b \times d_{in}}$ and $A_2 : \mathbb{R}^n \to \mathbb{R}^{d_{out} \times b}$.*
3. *Bias vectors $\mathbf{b}_1 \in \mathbb{R}^b$ and $\mathbf{b}_2 \in \mathbb{R}^{d_{out}}$.*
4. *A non-polynomial Lipschitz continuous activation function $\sigma : \mathbb{R} \to \mathbb{R}$ (applied element-wise).*

such that the two-layer, translation-equivariant convolutional map

$$\Psi(f)(t) = \mathbf{b}_2 + \int_{\mathbb{R}^n} A_2(t-s)\, \sigma\left(\mathbf{b}_1 + \int_{\mathbb{R}^n} A_1(s-\varsigma) f(\varsigma)\, d\varsigma\right) ds$$

approximates Φ uniformly on the extended set

$$E + \mathbb{R}^n = \{ f(\cdot - h) \mid f \in E,\, h \in \mathbb{R}^n \} \subset C_0(\mathbb{R}^n, \mathbb{R}^{d_{in}})$$

in the sense that

$$\sup_{g \in E + \mathbb{R}^n} \|\Phi(g) - \Psi(g)\|_\infty < \epsilon$$

Proof. This theorem is a direct corollary of Theorem 14 in [87], by specializing to the case where the group $G = \mathbb{R}^n$, the source and target index sets $\mathcal{S} = \mathcal{T} = \mathbb{R}^n$, and the invariant measure is the standard Lebesgue measure. □

4.4.3 Learning RFDEs from Universal Approximation

This theorem is significant because it guarantees that a two-layer CNN architecture is, in principle, sufficiently expressive to model any continuous, translation-equivariant nonlinear operator. This has profound implications for learning RFDEs.

First, for the specific application to RFDEs (where $n = 1$ for time), the operator must also be **causal**. This requires specializing the theorem by replacing the standard, two-sided convolutions ($*$) with **causal convolutions**, where the integration is performed only over the past:

$$(K *_{\text{causal}} h)(t) = \int_0^\infty K(s)h(t-s)\,ds = \int_{s\le t} K(t-s)h(s)\,ds$$

Second, this theorem illuminates two potential paths for learning RFDE dynamics:

1. **The Neural RFDE Paradigm:** The most direct application would be to model the functional F in $\dot{x}(t) = F(x_t)$ with a neural network, F_θ, which embodies the causal convolutional architecture justified by the universal approximation theorem, Theorem 4.4.1. The set of all trainable parameters of this network is represented by θ. Specifically, θ comprises the parameters defining the convolutional kernels A_1 and A_2 and the bias vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. We would define the output of this Neural RFDE model as:

$$F_\theta(x_t) = \mathbf{b}_2 + \int_{s\le t} A_2(t-s)\,\sigma\left(\mathbf{b}_1 + \int_{\varsigma\le s} A_1(s-\varsigma)x(\varsigma)\,d\varsigma\right)ds.$$

In principle, one could then use a numerical RFDE solver [88] to integrate the system $\dot{x}(t) = F_\theta(x_t)$ and train the model end-to-end. However, this approach is computationally challenging for real-time applications due to the repeated, expensive evaluation of the history-based functional F_θ within the solver.

2. **The History Encoder Paradigm:** An alternative, computationally efficient strategy is to use the universal approximation theorem to justify the architecture of an *encoder*. This is the approach taken in this work. The theorem provides a principled reason to use a causal convolutional network to process an observed history segment and extract its relevant features into a finite-dimensional representation. This insight forms the foundation for a practical, hybrid architecture suitable for real-time prediction, which will be explored in detail in Chapter 6.

Chapter 5: A Theory of Spatio-Temporal Set Evolution

The accurate prediction of complex, interacting multi-agent systems requires models that are not only empirically effective but also mathematically well-founded. This chapter develops a rigorous theory for the spatio-temporal evolution of sets of trajectories, the natural data representation for such systems. We begin by analyzing the fundamental symmetries inherent to this problem—namely, time-homogeneity and agent permutability, captured by the group $\mathbb{R} \times \mathfrak{S}_n$—and demonstrate how these symmetries impose strict constraints on any principled learning architecture.

With the architectural desiderata established, the chapter then builds the analytical framework necessary to prove that the dynamics these architectures seek to learn are well-posed. We first develop two equivalent characterizations for the evolution of a single compact set: a novel formulation based on a two-sided functional inequality and a second rooted in the established theory of morphological equations and mutational analysis. Equipped with these powerful tools, we then extend the analysis to systems with memory, proving existence and uniqueness theorems for delayed morphological and transport equations. This work culminates in a proof of the existence and uniqueness for a novel class of coupled functional differential equations on a hyperspace of trajectories, providing a rigorous mathematical foundation for the very systems our predictive models aim to capture.

5.1 Causal-Pointwise Equivariance

5.1.1 The Data Space

To formalize the symmetries of interactive dynamics, we must first precisely define the mathematical object representing the system's state. The choice of representation is dictated by the nature of the problem, and a crucial distinction must be made between modeling interactive trajectory prediction and other tasks involving evolving point clouds, such as shape animation. This distinction begins at the level of abstraction. In a physical point cloud, a point often has a direct, low-level correspondence to a physical location on an object. In contrast, for interactive trajectory prediction, a point is a high-level abstraction of a complex agent (e.g., a person or vehicle) with its own internal state and intent.

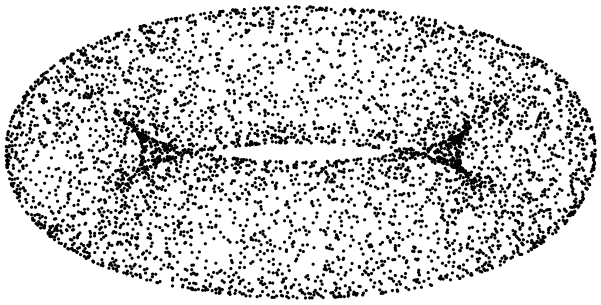


Figure 5.1: An example of a point cloud representing a doughnut shape

This high level of abstraction dictates the modeling goal: because each point represents an individual agent, the goal is necessarily to forecast the future path of each specific agent. This requires a data representation that explicitly preserves individual identities over time. We must therefore distinguish between two fundamentally different ways of encoding dynamic, multi-agent data:

1. **A Set of Trajectories:** The data is represented as a collection of individual history func-

tions, $\{\gamma_i\}_{i\in\Lambda}$, where each function $\gamma_i : (-\infty, 0] \to \mathbb{R}^d \cup \{\perp\}$ is the partial history of a specific agent i. This is the required domain for our work, as it preserves individual identities. Conceptually, the state of the entire system is a single point in the power set of the function space, $2^{\widetilde{C}((-\infty,0],\mathbb{R}^d)}$.

2. **A Trajectory of Sets:** The data is represented as a single function of time, $E : I \to \mathcal{K}(\mathbb{R}^d)$, which returns the spatial configuration of the group at each instant. This formulation, which underpins classical morphological equations, captures the collective dynamics of the crowd but does not preserve the identity of individual agents between time steps. The evolution is driven by a set-valued vector field acting pointwise on the shape at each moment. Yet, as this vector field maps the set as a whole, the identity of individual points is discarded as soon as the field is applied, and no agent-level trajectories are retained in the description.

Theoretically, if one had a continuous-time observation of a finite set of agents, it would be possible to recover the individual trajectories from the trajectory of sets. However, in practice, we only have access to discrete-time snapshots $E(t_1)$, $E(t_2), \ldots$. This discrete sampling creates a fundamental *data association problem* [89]: given the set of positions at one frame and the set at the next, it is impossible to know which point corresponds to which without additional information.

This ambiguity is resolved in real-world systems by using additional features. An upstream perception and tracking system takes the raw sensor data (like video or LiDAR frames) and uses features such as visual appearance to solve the data association problem. Alternatively, data from sensors with persistent identities (like GPS) provides this information natively [90]. Our framework thus operates on this higher-level, pre-processed representation, assuming individual

agent trajectories have already been established.

To understand the topological complexity of the set of trajectories space, we must introduce the conditions that govern the compactness of a set of functions. These are pointwise boundedness and equicontinuity, which are formalized in the Arzelà-Ascoli theorem.

Definition 5.1.1. *Let X be a topological space and let Y be a metric space with metric d. A set of functions $\mathcal{F} \subset C(X, Y)$ is said to be* pointwise bounded *if, for each $x \in X$, the set of evaluations*

$$\{\, f(x) \colon f \in \mathcal{F}\}$$

is a bounded subset of Y.

A set of functions $\mathcal{F} \subset C(X, Y)$ is called equicontinuous *if for every $x \in X$ and every $\epsilon > 0$, there exists a neighborhood U of x such that*

$$\forall y \in U,\ \forall f \in \mathcal{F}, \quad d\big(f(y), f(x)\big) < \epsilon.$$

Theorem 5.1.2 (Arzelà–Ascoli)**.** *Let X be a compact Hausdorff space and let Y be a complete metric space. A subset $\mathcal{F} \subset C(X, Y)$ is relatively compact (i.e., its closure in $C(X, Y)$ is compact) if and only if $\mathcal{F}$ is equicontinuous and pointwise bounded.*

Proof. See [63], Theorem 4.43 (Arzelà–Ascoli theorem I). □

In the context of trajectory prediction, these formal definitions have direct physical interpretations. Pointwise boundedness means that at any given instant the entire swarm of agents is confined to a finite region of space, while equicontinuity enforces a uniform speed limit—no agent can move arbitrarily fast. In practice, the empirical trajectory sets we work with are finite

and therefore trivially compact; the Arzelà–Ascoli framework provides conditions under which their limits (or any compact set approximated by such data) remain well-behaved in $\widetilde{C}(I, \mathbb{R}^d)$.

It is precisely these two physical constraints that ensure a set of real-world trajectories is topologically well-behaved (i.e., relatively compact). This connection is crucial, as it allows us to ground the abstract requirement of compactness—a prerequisite for the universal approximation theorems developed in this thesis—in reasonable assumptions about the physical systems being modeled. Therefore, the natural mathematical domain for our study is the hyperspace of all non-empty compact subsets of the function space $\widetilde{C}(I, \mathbb{R}^d)$, which we denote by $\mathcal{K}\big(\widetilde{C}(I, \mathbb{R}^d)\big)$

5.1.2 The $\mathbb{R} \times \mathfrak{S}_n$ Symmetry

Having established the data space as a collection of individual agent trajectories, we now formalize the symmetries that a robust predictive model must respect when operating on this space. These principles of time-homogeneity and agent anonymity are captured by $\mathbb{R} \times \mathfrak{S}_n$.

The principle of time-homogeneity requires that the underlying laws of motion are constant over time. A model satisfies this principle if its predictions are consistent under arbitrary time shifts. Formally, let τ_s be the translation operator for any $s \in \mathbb{R}$, which acts on a trajectory $\gamma(s)$ as $\big(\tau_s\gamma\big)(t) = \gamma(t-s)$ and on a set of trajectories A as $\tau_s A = \{\tau_s\gamma : \gamma \in A\}$. An operator F that maps a pointed set of trajectories (γ, A) to an output trajectory is time-equivariant if it commutes with this action:

$$F\big(\tau_s\gamma,\ \tau_s A\big) \ = \ \tau_s\big(F(\gamma, A)\big)$$

This property ensures that if the entire input scenario—both the target agent and its context—is shifted in time, the output trajectory is identically shifted without any change to its form.

The principle of agent anonymity requires that the model treats all context agents equally, without regard to any arbitrary labels or ordering. For an operator $F(\gamma, A)$ that produces an output corresponding to the specific agent γ, this means the output should be invariant to permutations of the context agents in the set A. Let $\sigma \in \mathfrak{S}_n$ be a permutation acting on the elements of an n-element set A. The operator F is *permutation equivariant* if

$$F\big(\gamma,\ \sigma\cdot A\big)\ =\ F\big(\gamma,\ A\big).$$

This symmetry is achieved through aggregation operators, and while sum, average, max, and min are all permutation invariant, a crucial distinction exists in their expressive power. For a fixed number of agents n, summation is unique in its ability to fully characterize permutation invariance. That is, a tuple $(x_1, \ldots, x_n)$ is a permutation of $(y_1, \ldots, y_n)$ if and only if

$$\sum_{i=1}^{n} f(x_i)\ =\ \sum_{i=1}^{n} f(y_i) \quad \text{for all functions } f : X \to \mathbb{R}.$$

The importance of these invariant operators extends beyond invariant tasks. They are, in fact, the fundamental building blocks for constructing equivariant functions. The following proposition establishes the precise relationship between a G-equivariant function and a simpler set of invariant functions.

Proposition 5.1.3 (Decomposition of G-Equivariant Functions)**.** *Let G be a finite group acting on the index set $\{1, \ldots, n\}$. This action decomposes the indices into a disjoint union of orbits $O_1, \ldots, O_m$, where each i_j is a chosen representative of its orbit.*

A function $F : \mathbb{R}^n \to \mathbb{R}^n$ is G-equivariant if and only if its structure is determined by a

collection of invariant *functions, one for each orbit.*

Specifically, for each orbit O_{i_j} there exists a single function

$$f_j : \mathbb{R}^n \to \mathbb{R}^n$$

which is invariant under the action of the stabilizer subgroup $\mathrm{Stab}_G(i_j)$.

The G-equivariant function F is then assembled from these invariant functions. For any output component $k \in \{1, \ldots, n\}$, if k belongs to the orbit O_{i_j}, its value is given by

$$F(x)_k \;=\; \big(f_j \circ \tau_k\big)(x),$$

where $\tau_k \in G$ is a group element (specifically, a coset representative) that maps the orbit representative i_j to the index k.

Proof. See [91], Proposition 3.1. □

This proposition is fundamental because it provides a constructive roadmap. It reduces the complex problem of approximating a G-equivariant function F to the simpler problem of approximating a set of $\mathrm{Stab}_G(i_j)$-invariant functions f_j. For the permutation group $\mathfrak{S}_n$, an invariant function for the stabilizer subgroup ($\cong \mathfrak{S}_{n-1}$) can be represented in the form $f(x) = \rho\big(x_1,\ \sum_{i=2}^{n} \phi(x_i)\big)$, which immediately gives an $\mathfrak{S}_n$-equivariant map, $F_i(x) = \rho\big(x_i,\ \sum_{j=1,j\neq i}^{n} \phi(x_j)\big)$. This representation explicitly uses the summation operator ($\sum$)—a permutation-invariant operator—as its core component.

Despite this theoretical underpinning of summation, we deliberately employs $\max$ (or $\min$) pooling. This choice is justified by the practical constraints of the trajectory prediction problem—

namely, the varying number of agents and the physical impossibility of co-location—which make the insensitivity of $\max$ pooling to multiplicity a more robust and logical choice.

The combined requirement for a model to respect both time-homogeneity and agent anonymity means it should ideally be equivariant under the actions of both groups. We can therefore define the idealized symmetry group for this problem as the product group $\mathbb{R} \times \mathfrak{S}_n$. The following subsections will explore how practical models must adapt to this ideal symmetry and the fundamental architectural consequences that arise from its structure.

5.1.3 Causal Convolutions and Set-Pooling for Practical Equivariance

While the product group $\mathbb{R} \times \mathfrak{S}_n$ defines the idealized symmetries of the problem, a direct implementation is not feasible for real-world applications. Any realizable architecture must adapt this ideal group structure to respect fundamental physical and observational constraints. This leads to specific architectural choices that are not merely convenient, but necessary.

The ideal of perfect time-equivariance under the group $(\mathbb{R}, +)$ implies symmetry under time shifts both forward and backward. However, any predictive model is fundamentally constrained by causality: the future cannot influence the past. The model's output at time t can only depend on inputs from times $s \leq t$.

This physical constraint forces us to restrict the full $\mathbb{R}$-action to a causal semigroup action, considering only past-to-future evolution. This restriction directly motivates the use of causal convolutions for processing the temporal dimension of trajectory data. Unlike standard convolutions that use information from both sides of a point, causal convolutions are designed to only use past and present information, thereby respecting the arrow of time while still maintaining the

principle of time-homogeneity for past-looking dynamics.

The symmetric group $\mathfrak{S}_n$ is defined for a fixed integer n. As we saw, theoretical representations of permutation-invariant functions, such as those using summation, are most naturally defined over a fixed number of inputs.

However, in any realistic multi-agent scenario, the number of agents within a sensor's view is not constant. Agents enter and exit the scene, meaning that the cardinality of the context set A varies over time. A robust architecture must therefore be able to handle an input set of arbitrary size. This constraint motivates the use of set-encoding mechanisms. Aggregation operators like max or min pooling are prime examples, as their output is well-defined and stable regardless of the number of elements in the input set. This practical requirement reinforces the decision to use max pooling, a choice already justified by the physical impossibility of co-located agents.

5.1.4 Fundamental Decomposition and Architectural Consequences

A standard technique in mathematics is to simplify a problem by choosing a basis or coordinate system adapted to its symmetries. For a topological group like the additive group $\mathbb{R}^n$, which is isomorphic to a product of n copies of $\mathbb{R}$, this is equivalent to choosing a basis for the vector space. While the standard axes are an obvious choice, it is often highly beneficial to choose a different slant decomposition—like an eigenbasis—to diagonalize an operator and simplify computation.

This flexibility, however, does not extend to the symmetry group of our problem. The group $G = \mathbb{R} \times \mathfrak{S}_n$ exhibits a powerful property: the *uniqueness of its internal direct product decomposition* into nontrivial subgroups.

Theorem 5.1.4. *Let $G = \mathbb{R} \times \mathfrak{S}_n$ be the product topological group. Any nontrivial decomposition of G into an internal direct product $G = H_1H_2$ requires that the set of subgroups $\{H_1, H_2\}$ is equal to the set $\big\{\mathbb{R} \times \{e\}, \{0\} \times \mathfrak{S}_n\big\}$, where e is the identity in $\mathfrak{S}_n$.*

Proof. Assume G has a nontrivial internal direct-product structure $G = H_1H_2$, where H_1 and H_2 are closed normal subgroups of G with $H_1 \cap H_2 = \{e\}$.

Let $p : G \to \mathfrak{S}_n$ be the projection homomorphism, $p(r, \sigma) = \sigma$. Set $K_1 = p(H_1)$ and $K_2 = p(H_2)$.

- Since H_1 and H_2 are normal in G, K_1 and K_2 are normal subgroups of $\mathfrak{S}_n$.
- Commutativity of H_1 and H_2 in G implies elements of K_1 and K_2 commute.
- Since $G = H_1H_2$, we have $\mathfrak{S}_n = K_1K_2$ with $K_1 \cap K_2 = \{e\}$.

However, $\mathfrak{S}_n$ is indecomposable (it cannot be written as a product of two nontrivial subgroups). Therefore one factor, say K_1, must be trivial while the other, K_2, is the entire group, or vice versa. Without loss of generality, we may assume that $p(H_1) = \{e\}$ and $p(H_2) = \mathfrak{S}_n$. Then $H_1 \subset \ker p = \mathbb{R} \times \{e\}$. Since H_1 is a closed subgroup of $\mathbb{R}$, it can only be $\{0\}$, $a\mathbb{Z}$ (for some $a \neq 0$), or $\mathbb{R}$ itself. Nontriviality of the decomposition forces $H_1 \neq \{0\} \times \{e\}$. Suppose $H_1 = a\mathbb{Z} \times \{e\}$. Then

$$H_2 \;\cong\; G/H_1 \;=\; (\mathbb{R} \times \mathfrak{S}_n) \,/\, \big(a\mathbb{Z} \times \{e\}\big) \;\cong\; (\mathbb{R}/a\mathbb{Z}) \times \mathfrak{S}_n \;\cong\; S^1 \times \mathfrak{S}_n,$$

where S^1 is the circle group. But $H_2 \subset \mathbb{R} \times \mathfrak{S}_n$, and $\mathbb{R} \times \mathfrak{S}_n$ has no subgroup isomorphic to S^1

(its only compact subgroup is trivial). This is a contradiction. Thus the only possibility is

$$H_1 = \mathbb{R} \times \{e\}.$$

Since $G = H_1 H_2$ is an internal direct product, $H_2 = \{0\} \times \mathfrak{S}_n$. Therefore,

$$\{H_1, H_2\} = \left\{\mathbb{R} \times \{e\}, \{0\} \times \mathfrak{S}_n\right\}$$

□

The architectural consequence of this uniqueness is profound. It proves that there are no exotic or clever modular architectures that can circumvent the problem's fundamental structure by finding a new basis where temporal evolution and agent permutations become intertwined in a simpler way.

Therefore, any model that fully respects the problem's symmetries is forced to address the temporal $(\mathbb{R})$ and permutational $(\mathfrak{S}_n)$ challenges separately. This provides a fundamental rationale for the *canonical architectural pathways* that we will now formalize, demonstrating that they are not merely sensible designs, but are derived from the only principled modular approaches available.

5.1.5 A Conjecture of Universal Approximation Theorems for Causally-Pointwise Equivariant Maps

Thanks to the unique product structure $\mathbb{R} \times \mathfrak{S}_n$, any principled architecture must disentangle its temporal and permutation-handling modules. This leads to a set of canonical architectural

families. The following theorem, presented here as a conjecture, posits that these families—each a principled combination of universal approximation theorems for set-equivariant functions (Theorem 3.5.6) and for shift-equivariant operators (Theorem 4.4.1)—are sufficient to approximate any continuous operator with the requisite spatio-temporal symmetries.

Conjecture 5.1.5 (A Conjecture of Universal Approximation Theorems for Equivariant Operators on Sets of Functions)**.** *Let $E \subset C_0(\mathbb{R}^n, \mathbb{R}^{d_{\text{in}}})$ be a compact set of base functions. Let*

$$F : \mathrm{Pt}\big(\mathcal{K}(E) + \mathbb{R}^n\big) \;\to\; C_0(\mathbb{R}^n, \mathbb{R}^{d_{\text{out}}})$$

be a continuous and $\mathbb{R}^n$-equivariant operator.

Then for any $\epsilon > 0$, there exists an architecture Ξ from one of the canonical families below such that for all $(\gamma, A) \in \mathrm{Pt}(\mathcal{K}(E) + \mathbb{R}^n)$,

$$\sup_{t \in \mathbb{R}^n} \big\| F(\gamma, A)(t) \;-\; \Xi(\gamma, A)(t) \big\| \;<\; \epsilon.$$

In the following families, Ψ denotes a universal two-layer convolutional network of the form described in Theorem 4.4.1. A single-layer convolution is denoted $\mathrm{CNN}_i(f)(t) \;=\; \mathbf{b}_i \;+\; \int A_i(t-s)\, f(s)\, \mathrm{d}s$. The functions Γ, ϕ, and h are standard Multi-Layer Perceptrons.

Pathway 1: Shift $\to$ Set *This pathway first applies a universal temporal operator (Ψ) to each input function to extract high-level feature functions. The spatial aggregation is then performed pointwise in time on these features.*

- ***Version 1a: Global Summary Model***

$$\Xi(\gamma, A)(t) = \Gamma\Big(\Psi(\gamma)(t),\ \max_{\gamma' \in A} h\big(\Psi(\gamma')(t)\big)\Big)$$

- ***Version 1b: Individualized Interaction Model***

$$\Xi(\gamma, A)(t) = \Gamma\Big(\max_{\gamma' \in A} \phi\big(\Psi(\gamma)(t), , \Psi(\gamma')(t)\big)\Big)$$

Pathway 2: Set $\rightarrow$ Shift *This pathway first aggregates spatial information at each time step from the raw input functions. The universal temporal operator (Ψ) is then applied to the resulting intermediate function.*

- ***Version 2a: Global Summary Model***

$$\Xi(\gamma, A) = \Psi\Big(s \mapsto \Gamma\big(\gamma(s),\ \max_{\gamma' \in A} h(\gamma'(s))\big)\Big)$$

- ***Version 2b: Individualized Interaction Model***

$$\Xi(\gamma, A) = \Psi\Big(s \mapsto \Gamma\big(\max_{\gamma' \in A} \phi\big(\gamma(s), \gamma'(s)\big)\big)\Big)$$

Pathway 3: Shift $\rightarrow$ Set $\rightarrow$ Shift *This more expressive pathway first applies a single convolution, performs spatial aggregation on the intermediate features, and then applies a second single convolution to the result.*

- ***Version 3a: Global Summary Model***

$$\Xi(\gamma, A) = \mathrm{CNN}_2\Big(s \mapsto \Gamma\Big(\mathrm{CNN1}(\gamma)(s),\ \max_{\gamma' \in A} h\big(\mathrm{CNN}_1(\gamma')(s)\big)\Big)\Big)$$

- ***Version 3b: Individualized Interaction Model***

$$\Xi(\gamma, A) = \mathrm{CNN2}\Big(s \mapsto \Gamma\Big(\max_{\gamma' \in A} \phi\big(\mathrm{CNN}_1(\gamma)(s), \mathrm{CNN}_1(\gamma')(s)\big)\Big)\Big)$$

Pathway 4: Shift $\to$ Set $\to$ Shift with Final Pointwise MLP *This family extends Pathway 3 by applying a final pointwise MLP (Γ) to the output feature function, allowing for a final transformation of the deep spatio-temporal features.*

- ***Version 4a: Global Summary Model***

$$\Xi(\gamma, A)(t) = \Gamma\left(\mathrm{CNN}_2\Big(s \mapsto \big[\mathrm{CNN1}(\gamma)(s),\ \max_{\gamma' \in A} h(\mathrm{CNN}_1(\gamma')(s))\big]\Big)(t)\right)$$

- ***Version 4b: Individualized Interaction Model***

$$\Xi(\gamma, A)(t) = \Gamma\left(\mathrm{CNN2}\Big(s \mapsto \max_{\gamma' \in A} \phi\big(\mathrm{CNN}_1(\gamma)(s), \mathrm{CNN}_1(\gamma')(s)\big)\Big)(t)\right)$$

These architectural families represent the complete set of principled, universal, modular architectures for this general class of operators. It is crucial to note, however, that for the specific task of trajectory prediction, these architectures must be further constrained by causality. The general $\mathbb{R}^n$–equivariant operators, h and h_{final}, must be instantiated as *causal operators* that only

use past information to influence the future. This specialization, often realized using architectures like causal convolutional networks, ensures that the resulting models are not only expressive and symmetric but also physically realizable for forecasting.

5.2 A Functional Inequality Characterization of Point-to-Point Compact Set Evolution

In this section, we establish the framework for the point-to-point evolution of a compact set $K_t \subset \mathbb{R}^d$. The point-to-point evolution is governed by the transport of the set's points along the trajectories of a time-dependent vector field $v(t, x)$. The trajectory for a single point is the solution to the ordinary differential equation (ODE):

$$\dot{x}(t) \;=\; v\big(t,\, x(t)\big).$$

The well-posedness of this evolution is governed by the classical theory of ODEs. The foundational result guarantees local existence and uniqueness.

Theorem 5.2.1 (Existence and Uniqueness of Maximal Solutions)**.** *Let $D \subset \mathbb{R} \times \mathbb{R}^d$ be open and let $v : D \;\to\; \mathbb{R}^d$ be continuous in t and locally Lipschitz-continuous in x: for every compact $K \subset D$ there is a constant $L_K > 0$ such that*

$$\|v(t, x) - v(t, y)\| \leq L_K \|x - y\| \quad \textit{for all } (t, x), (t, y) \in K.$$

Then, for every initial condition $(s, x_s) \in D$, there exists a unique noncontinuable C^1 solution on

the maximal interval $(a, b) \ni s$*,*

$$x : (a, b) \to \mathbb{R}^d, \qquad x(s) = x_s,$$

of the ordinary differential equation

$$\dot{x}(t) = v\big(t, x(t)\big), \quad (t, x(t)) \in D.$$

Moreover, if the maximal interval (a, b) *is finite at either end, then as* $t \to a^+$ *or* $t \to b^-$*, the point* $\big(t, \phi(t)\big)$ *approaches* ∂D*.*

Proof. This theorem combines the standard local existence result of Picard–Lindelöf with the continuation theorem. For a full treatment, see, e.g., [92], Chapter 1, Theorems 2.1 and 3.1. □

However, for the evolution of a set—a global object—a local result is not sufficient. We need to guarantee that a unique solution exists for all initial points in an initial set K_0 and for all time $t \in [0, T^*]$. This requires stronger, global conditions on the vector field. To extend the local solution to a unique global flow, we impose the following:

1. **Uniform Regularity in Space:** The vector field is uniformly bounded and uniformly Lipschitz continuous with respect to the spatial variable. Precisely, we require

$$\sup_{t\in[0,T^*]} \Big(\sup_{x\in\mathbb{R}^d} \|v(t,x)\| \; + \; \sup_{x\neq y} \frac{\|v(t,x) - v(t,y)\|}{\|x - y\|} \Big) \; < \; \infty.$$

2. **Continuity in Time:** For any fixed $x \in \mathbb{R}^d$, the map $t \mapsto v(t, x)$ is continuous on $[0, T^*]$.

These global conditions are sufficient to ensure that the local solutions guaranteed by the Picard–Lindelöf theorem can be uniquely continued for all time. We state this as the following proposition.

Proposition 5.2.2 (Existence of a Unique Global Flow)**.** *Let the vector field $v(t, x)$ satisfy conditions (1) and (2) above. Then there exists a unique global flow map, $\Phi_{t,s}$, which continuously maps a point x_s at time s to its position x_t at time t. Consequently, this flow defines a unique evolution for any initial compact set K_0, given by*

$$K_t \;=\; \Phi_{t,0}(K_0) \quad \textit{for all } t \in [0, T^*].$$

Proof. We show that for any initial point, the unique solution to the ODE exists for all time in $[0, T^*]$.

By Theorem 5.2.1, for any initial condition (s, x_s) there exists a unique noncontinuable solution $x(t)$ defined on a maximal interval of existence $(a, b) \subseteq [0, T^*]$.

We argue this by contradiction. Assume $b < T^*$. By the continuation principle in Theorem 5.2.1, if a solution cannot be continued past time b, its trajectory must leave every compact subset of the spatial domain $\mathbb{R}^d$ as $t \to b^-$. Hence $\|x(t)\| \to \infty$ as $t \to b^-$.

However, by Assumption (A1) (Uniform Regularity in Space), the vector field is uniformly bounded: there exists a finite constant M such that

$$\|v(t, x)\| \leq M \quad \text{for all } (t, x).$$

The solution $x(t)$ satisfies the integral equation

$$x(t) = x_s + \int_s^t v\big(\tau, x(\tau)\big)\, d\tau.$$

Applying the triangle inequality gives

$$\|x(t)\| \leq \|x_s\| + \int_s^t \|v(\tau, x(\tau))\|\, d\tau \leq \|x_s\| + \int_s^t M\, d\tau = \|x_s\| + M\,(t-s).$$

This holds for all $t \in [s, b)$. Taking the limit $t \to b^-$ yields

$$\lim_{t \to b^-} \|x(t)\| \;\leq\; \|x_s\| + M\,(b-s),$$

a finite constant. This contradicts the requirement $\|x(t)\| \to \infty$ as $t \to b^-$. Therefore $b < T^*$ is impossible. An identical argument shows $a > 0$ is impossible. Thus the unique maximal solution extends to all $t \in [0, T^*]$.

Since a unique global solution exists for every initial point (s, x_s), we obtain a global flow map $\Phi_{t,s}$ that takes x_s to the unique solution $x(t)$. The continuity and uniqueness of this flow for every point in an initial compact set K_0 guarantee a unique point-to-point evolution of the set itself, given by

$$K_t \;=\; \Phi_{t,0}(K_0).$$

□

5.2.1 The Extremal-Integral Characterization

The unique flow-transported point-to-point evolution K_t admits a powerful characterization based on how its support functional evolves over time. This characterization is given by the following theorem, which defines the point-to-point evolution in terms of a two-sided extremal-integral inequality.

Theorem 5.2.3 (Functional Inequality Characterization of Flow-Transported Point-to-Point Compact Set Evolution)**.** *Let the vector field $v(t, x)$ satisfy the regularity conditions, (1) Uniform Regularity in Space and (2) Continuity in Time. Let $\{L_t\}_{t\in[0,T^*]}$ be a family of non-empty compact sets with $L_0 = K_0$.*

- *It is a* subsolution *($L_t \subset K_t$ for all $t \in [0, T^*]$) to the point-to-point evolution if, for any test function $\eta \in C_c^\infty\big([t_1, t_2] \times \mathbb{R}^d\big)$ and any compact subinterval $[t_1, t_2] \subset [0, T^*]$, it satisfies the upper-bound inequality*

$$\max_{x\in L_{t_2}} \eta(t_2, x) \;-\; \max_{x\in L_{t_1}} \eta(t_1, x) \;\le\; \int_{t_1}^{t_2} \max_{x\in L_s} \big[\partial_s\eta(s, x) + \nabla_x\eta(s, x) \cdot v(s, x)\big]\, ds.$$

- *It is a* supersolution *($L_t \supset K_t$ for all $t \in [0, T^*]$) to the point-to-point evolution if, for any test function $\eta \in C_c^\infty\big([t_1, t_2] \times \mathbb{R}^d\big)$ and any compact subinterval $[t_1, t_2] \subset [0, T^*]$, it satisfies the lower-bound inequality*

$$\int_{t_1}^{t_2} \min_{x\in L_s} \big[\partial_s\eta(s, x) + \nabla_x\eta(s, x) \cdot v(s, x)\big]\, ds \;\le\; \max_{x\in L_{t_2}} \eta(t_2, x) \;-\; \max_{x\in L_{t_1}} \eta(t_1, x).$$

The flow-transported point-to-point compact-set evolution $K_t = \Phi_{t,0}(K_0)$ is the unique

family starting from K_0 that is both a subsolution and a supersolution.

Proof. The proof consists of two parts: first, showing that the true point-to-point evolution K_t is indeed both a subsolution and a supersolution (necessity), and second, showing that any point-to-point evolution satisfying the upper-bound (resp. lower-bound) inequality must be a subsolution (resp. supersolution), so any point-to-point evolution satisfying both must be unique.

(Necessity) Let $[t_1, t_2] \subset [0, T^*]$ be an arbitrary compact subinterval of $[0, T^*]$ and let $\eta \in C_c^\infty\big([t_1, t_2] \times \mathbb{R}^d\big)$. First, we prove

$$\max_{x \in K_{t_2}} \eta(t_2, x) \; - \; \max_{x \in K_{t_1}} \eta(t_1, x) \; \leq \; \int_{t_1}^{t_2} \max_{x \in K_s} \big[\partial_s \eta(s, x) + \nabla_x \eta(s, x) \cdot v(s, x)\big] \, ds.$$

Let $x_2^* \in K_{t_2}$ be a point where the maximum is achieved, so

$$\max_{x \in K_{t_2}} \eta(t_2, x) = \eta\big(t_2, x_2^*\big).$$

Since $x_2^* \in K_{t_2}$, it has flowed from a unique initial point $x_0 \in K_0$. Consider the trajectory

$$x(t) = \Phi_{t,0}(x_0), \quad \text{noting } x(t_2) = x_2^*.$$

For the second term, $\max_{x \in K_{t_1}} \eta(t_1, x) \geq \eta\big(t_1, x(t_1)\big)$, since $x(t_1) \in K_{t_1}$.

Therefore

$$\max_{x \in K_{t_2}} \eta(t_2, x) - \max_{x \in K_{t_1}} \eta(t_1, x) \; \leq \; \eta\big(t_2, x(t_2)\big) - \eta\big(t_1, x(t_1)\big).$$

Now set $g(s) = \eta\big(s, x(s)\big)$. By the Fundamental Theorem of Calculus and the chain rule,

$$g(t_2) - g(t_1) = \int_{t_1}^{t_2} g'(s)\, ds = \int_{t_1}^{t_2} \Big[\partial_s \eta(s, x(s)) + \nabla_x \eta(s, x(s)) \cdot v(s, x(s))\Big]\, ds.$$

At each $s \in [t_1, t_2]$, $x(s) \in K_s$, so

$$\partial_s \eta(s, x(s)) + \nabla_x \eta(s, x(s)) \cdot v(s, x(s)) \;\le\; \max_{y \in K_s} \big[\partial_s \eta(s, y) + \nabla_y \eta(s, y) \cdot v(s, y)\big].$$

Integrating this pointwise bound from t_1 to t_2 gives the desired inequality.

The argument is entirely analogous for the other inequality. We must show

$$\int_{t_1}^{t_2} \min_{x \in K_s} \big[\partial_s \eta(s, x) + \nabla_x \eta(s, x) \cdot v(s, x)\big]\, ds \;\le\; \max_{x \in K_{t_2}} \eta(t_2, x) - \max_{x \in K_{t_1}} \eta(t_1, x).$$

Let $x_1^* \in K_{t_1}$ be where $\max_{x \in K_{t_1}} \eta(t_1, x)$ is attained. Follow the trajectory $x(t) = \Phi_{t,t_1}(x_1^*)$, noting $x(t_1) = x_1^*$. Then $\max_{x \in K_{t_2}} \eta(t_2, x) \ge \eta\big(t_2, x(t_2)\big)$. Hence

$$\max_{x \in K_{t_2}} \eta(t_2, x) - \max_{x \in K_{t_1}} \eta(t_1, x) \;\ge\; \eta\big(t_2, x(t_2)\big) - \eta\big(t_1, x(t_1)\big).$$

As before, let $g(s) = \eta(s, x(s))$. Then

$$g(t_2) - g(t_1) = \int_{t_1}^{t_2} \Big[\partial_s \eta(s, x(s)) + \nabla_x \eta(s, x(s)) \cdot v(s, x(s))\Big]\, ds.$$

At each s, $x(s) \in K_s$, so

$$\partial_s \eta(s, x(s)) + \nabla_x \eta(s, x(s)) \cdot v(s, x(s)) \;\geq\; \min_{y \in K_s} \big[\partial_s \eta(s, y) + \nabla_y \eta(s, y) \cdot v(s, y)\big].$$

Integrating yields the desired other inequality.

(Uniqueness) Let $\{L_t\}$ be a family of non-empty compact sets satisfying the upper-bound (resp. lower-bound) inequality with $L_0 = K_0$. We must show $L_t \subset K_t$ (resp. $L_t \supset K_t$).

Fix an arbitrary $\psi \in C_c^\infty(\mathbb{R}^d)$ and define the special test function

$$\eta(s, x) = \psi\big(\Phi_{0,s}(x)\big),$$

where $\Phi_{0,s}$ is the backward flow map. By construction, η satisfies the material-derivative equation

$$D_s \eta = \partial_s \eta + \nabla_x \eta \cdot v = 0.$$

Substituting η into the upper-bound (resp. lower-bound) inequality for $\{L_t\}$ yields

$$\max_{x \in L_{t_2}} \eta(t_2, x) \;-\; \max_{x \in L_{t_1}} \eta(t_1, x) \;\leq\; \int_{t_1}^{t_2} 0 \, ds$$

$$\text{resp.} \quad \int_{t_1}^{t_2} 0 \, ds \;\leq\; \max_{x \in L_{t_2}} \eta(t_2, x) \;-\; \max_{x \in L_{t_1}} \eta(t_1, x).$$

Hence

$$\max_{x\in L_{t_2}} \eta(t_2,x) \le \max_{x\in L_{t_1}} \eta(t_1,x) \quad \text{for all } 0\le t_1<t_2\le T^*$$

$$\text{resp.} \quad \max_{x\in L_{t_2}} \eta(t_2,x) \ge \max_{x\in L_{t_1}} \eta(t_1,x) \quad \text{for all } 0\le t_1<t_2\le T^*.$$

In particular, for any $t\in[0,T^*]$,

$$\max_{x\in L_t} \psi\big(\Phi_{0,t}(x)\big) \le \max_{x\in L_0} \psi(x)$$

$$\text{resp.} \quad \max_{x\in L_t} \psi\big(\Phi_{0,t}(x)\big) \ge \max_{x\in L_0} \psi(x).$$

Let $L'_t=\Phi_{0,t}(L_t)$ denote the pullback of L_t to time 0. Then the above inequality says

$$\max_{x\in L'_t} \psi(x) \le \max_{x\in L_0} \psi(x) \quad \text{for all } \psi,$$

$$\text{resp.} \quad \max_{x\in L'_t} \psi(x) \ge \max_{x\in L_0} \psi(x) \quad \text{for all } \psi,$$

so $L'_t\subset L_0$ (resp. $L'_t\supset L_0$). Pushing forward by $\Phi_{t,0}$ gives

$$L_t=\Phi_{t,0}(L'_t)\subset \Phi_{t,0}(L_0)=\Phi_{t,0}(K_0)=K_t.$$

$$\text{resp.} \quad L_t=\Phi_{t,0}(L'_t)\supset \Phi_{t,0}(L_0)=\Phi_{t,0}(K_0)=K_t.$$

If $\{L_t\}$ satisfies both the upper-bound and the lower-bound inequalities, we conclude $L_t=K_t$ for all t. This completes the proof of uniqueness. □

5.2.2 Necessity of the Conditions

The uniqueness of the flow-transported point-to-point set evolution, as established in Theorem 5.2.3, depends critically on two distinct features of the functional inequality characterization. First, the point-to-point evolution must satisfy a two-sided inequality, providing both an upper and a lower bound on the change in its support functional. Second, this two-sided condition must hold not just for a single time interval, but for all arbitrary sub-intervals of the point-to-point evolution's domain.

In this subsection, we demonstrate that both of these conditions are indispensable. We will construct conceptual counterexamples to show that relaxing either condition—that is, enforcing only a one-sided inequality or enforcing the two-sided inequality only on the global time domain—is insufficient to guarantee a unique solution. Following these counterexamples, we will compare this characterization to the standard weak formulation of the continuity equation for measures, highlighting the fundamental structural differences that necessitate these specific conditions for the point-to-point evolution of compact sets.

5.2.2.1 Non-Uniqueness for One-Sided Inequalities

The uniqueness proven in Theorem 5.2.3 relies on enforcing a two-sided inequality. We now show that satisfying only one of these inequalities is insufficient by giving concrete counterexamples in one dimension with zero velocity field, where the true flow-transported set is static.

Let $d = 1$, $v(s, x) \equiv 0$, and $K_0 = [0, 1]$. Then the true evolution is

$$K_t = \Phi_{t,0}(K_0) = [0, 1] \quad \text{for all } t \geq 0.$$

First, consider any family $\{L_t\}$ that only satisfies

$$\max_{x \in L_{t_2}} \eta(t_2, x) \; - \; \max_{x \in L_{t_1}} \eta(t_1, x) \; \leq \; \int_{t_1}^{t_2} \max_{x \in L_s} \big[\partial_s \eta(s, x) + \nabla_x \eta(s, x)\, v(s, x)\big]\, ds$$

for all $0 \leq t_1 < t_2 \leq 1$ and all test functions η. Geometrically, this prevents the support functional from growing faster than allowed, but places no lower bound on shrinking, so one may have

$$L_t \subseteq K_t.$$

For a pecific counterexample, we define

$$L_t = [\, 0,\, 1 - t\,], \quad t \in [0, 1].$$

Then $L_0 = [0, 1] = K_0$, but for any $t > 0$, $L_t \subsetneq K_t$. We verify the upper-bound inequality:

Proof of Upper-Bound Inequality. Fix $0 \leq t_1 < t_2 \leq 1$ and a test function η. Let $x_2^* \in L_{t_2}$ be a maximizer of $\eta(t_2, \cdot)$ on L_{t_2}. Since $1 - t_1 > 1 - t_2$, we have

$$x_2^* \; \in \; [\, 0,\, 1 - t_2\,] \; \subset \; [\, 0,\, 1 - t_1\,] = L_{t_1}.$$

Hence

$$\max_{x\in L_{t_2}} \eta(t_2,x) = \eta\big(t_2,x_2^*\big) \;\le\; \max_{x\in L_{t_1}} \eta(t_1,x) + \Big[\eta\big(t_2,x_2^*\big) - \eta\big(t_1,x_2^*\big)\Big].$$

By the Fundamental Theorem of Calculus,

$$\eta\big(t_2,x_2^*\big) - \eta\big(t_1,x_2^*\big) = \int_{t_1}^{t_2} \partial_s\eta\big(s,x_2^*\big)\,ds.$$

But $x_2^* \in L_s$ for all $s \in [t_1,t_2]$, so

$$\partial_s\eta\big(s,x_2^*\big) \;\le\; \max_{x\in L_s} \partial_s\eta(s,x).$$

Integrating from t_1 to t_2 gives

$$\eta\big(t_2,x_2^*\big) - \eta\big(t_1,x_2^*\big) \;\le\; \int_{t_1}^{t_2} \max_{x\in L_s} \partial_s\eta(s,x)\,ds.$$

Hence

$$\max_{x\in L_{t_2}} \eta(t_2,x) - \max_{x\in L_{t_1}} \eta(t_1,x) \;\le\; \int_{t_1}^{t_2} \max_{x\in L_s} \big[\partial_s\eta(s,x) + \nabla_x\eta(s,x)\,v(s,x)\big]\,ds,$$

since $v \equiv 0$. This verifies the upper-bound inequality, yet $L_t \neq K_t$, proving non-uniqueness. $\square$

Conversely, consider an evolution $\{L_t\}$ that only satisfies the lower-bound inequality. This condition prevents the support functional from shrinking faster than the flow but places no re-

striction on its growth. Thus one may have

$$L_t \;\supseteq\; K_t.$$

Using the same setup $d = 1$, $v \equiv 0$, and $K_0 = [0, 1]$, define

$$L_t \;=\; [\,0,\, 1 + t\,].$$

Then $L_0 = K_0$, but for any $t > 0$, $L_t \supsetneq K_t$. We now prove that $\{L_t\}$ satisfies the lower-bound inequality.

Proof of the Lower-Bound Inequality. We must show that for all $0 \le t_1 < t_2 \le 1$ and any test function η,

$$\int_{t_1}^{t_2} \min_{x \in L_s} \big[\partial_s \eta(s, x)\big]\, ds \;\le\; \max_{x \in L_{t_2}} \eta(t_2, x) - \max_{x \in L_{t_1}} \eta(t_1, x).$$

Let $x_1^* \in L_{t_1} = [0, 1+t_1]$ be a maximizer of $\eta(t_1, \cdot)$ on L_{t_1}. Since $t_1 < t_2$, we have $1+t_1 < 1+t_2$, so $L_{t_1} \subset L_{t_2}$. Hence $x_1^* \in L_{t_2}$ as well. Therefore

$$\max_{x \in L_{t_2}} \eta(t_2, x) - \max_{x \in L_{t_1}} \eta(t_1, x) \;\ge\; \eta\big(t_2, x_1^*\big) - \eta\big(t_1, x_1^*\big).$$

By the Fundamental Theorem of Calculus,

$$\eta\big(t_2, x_1^*\big) - \eta\big(t_1, x_1^*\big) = \int_{t_1}^{t_2} \partial_s \eta\big(s, x_1^*\big)\, ds.$$

But for each $s \in [t_1, t_2]$, $x_1^* \in L_s$ since $L_s = [0, 1+s] \supset [0, 1+t_1]$. Thus

$$\partial_s \eta\big(s, x_1^*\big) \;\geq\; \min_{x \in L_s} \partial_s \eta(s, x).$$

Integrating from t_1 to t_2 gives

$$\int_{t_1}^{t_2} \partial_s \eta\big(s, x_1^*\big)\, ds \;\geq\; \int_{t_1}^{t_2} \min_{x \in L_s} \partial_s \eta(s, x)\, ds.$$

Combining these inequalities yields the desired lower-bound inequality. Since $L_t \neq K_t$, uniqueness fails. □

5.2.2.2 Non-Uniqueness for a Fixed Time Interval

Finally, we demonstrate that uniqueness of the point-to-point evolution is only guaranteed if the two-sided inequality holds for *all* compact sub-intervals $[t_1, t_2]$. If the condition were only imposed on the global interval $[0, T^*]$, it would constrain only the initial and final states, permitting non-physical paths in between.

As a specific counterexample, consider the following system:

- Let the dimension be $d = 1$.
- Let the vector field be $v(t, x) = 0$.
- Let the time interval be $[0, T^*] = [0, 2]$.
- Let the initial set be the singleton $K_0 = \{0\}$.

The true flow-transported evolution is therefore static:

$$K_t = \{0\} \quad \text{for all } t \in [0, 2].$$

Now consider the alternative evolution defined by the interval

$$L_t = [\, t(2-t),\, -t(2-t)\,] = [-t(2-t),\, t(2-t)].$$

Let us check the start and end points of this evolution:

- At $t = 0$, we have $L_0 = [0, 0] = \{0\} = K_0$.
- At $t = 2$, we have $L_2 = [2 \cdot 0,\, -2 \cdot 0] = \{0\} = K_2$.

The point-to-point evolution L_t starts and ends at the correct states. However, for any time in between—for instance at $t = 1$—the set has expanded to

$$L_1 = [-1, 1].$$

Thus L_t expands from a point to a line segment and then contracts back to a point, which is clearly not the true static evolution.

Although this pathological point-to-point evolution L_t is clearly not the true flow, it can be shown to satisfy the two-sided extremal-integral inequality if the condition is only enforced on the global interval $[0, 2]$. For any test function $\eta \in C_c^\infty([0, 2] \times \mathbb{R})$, the middle term of the

inequality reduces to the change at the origin:

$$\max_{x\in L_2}\eta(2,x)-\max_{x\in L_0}\eta(0,x)=\eta(2,0)-\eta(0,0)=\int_0^2\partial_s\eta(s,x)\,ds.$$

Because the point $x=0$ is always contained in L_s for all $s\in[0,2]$, we have the pointwise bounds

$$\min_{x\in L_s}\partial_s\eta(s,x)\ \le\ \partial_s\eta(s,0)\ \le\ \max_{x\in L_s}\partial_s\eta(s,x).$$

Integrating these inequalities over $s\in[0,2]$ shows that the two-sided extremal-integral inequality is indeed satisfied on the global interval $[0,2]$.

However, on a sub-interval such as $[0,1]$, the evolution $L_t=[-t(2-t),\,t(2-t)]$ is purely expansive and thus fails the upper-bound (subsolution) test on $[0,1]$. This illustrates that enforcing the inequalities on *all* compact sub-intervals is essential to rule out such deviant paths and to guarantee uniqueness at every moment in time.

5.2.3 Comparison with the Weak Formulation of the Continuity Equation

It is illuminating to compare the functional-inequality characterization for sets with the standard weak formulation of the continuity equation [55] for measures,

$$\partial_t\mu_t+\nabla\cdot\big(v_t\,\mu_t\big)=0.$$

Both frameworks share a fundamental structure for their solutions. In both cases, the point-to-point evolution is determined by transporting the initial condition along the flow map, $\Phi_{t,0}$, of

the underlying ODE. For sets, the solution is the image of the initial set under the flow,

$$K_t = \Phi_{t,0}(K_0).$$

For measures, the unique solution is the pushforward measure of the initial measure under the same flow [12, Theorem 5.34],

$$\mu_t = (\Phi_{t,0})_{\#}\mu_0.$$

While both frameworks describe the transport of a distribution, they differ fundamentally in the structure of their defining conditions.

The weak formulation of the continuity equation requires that a *single integral equation* holds for all test functions $\eta \in C_c^\infty\big((0,T^*) \times \mathbb{R}^d\big)$:

$$\int_0^{T^*} \int_{\mathbb{R}^d} \big[\partial_t \eta(t,x) + \nabla_x \eta(t,x) \cdot v(t,x)\big]\, d\mu_t(x)\, dt \;=\; 0.$$

This stands in contrast to our framework on two key points:

1. **Inequalities vs. Equation.** Our characterization requires a *two-sided inequality*. This is necessary to provide both an upper and a lower bound on the evolution of a set's boundary, which can undergo complex topological changes. The transport equation for measures, which conserves total mass, can be fully characterized by a single equality.

2. **Arbitrary Sub-intervals vs. a Single Interval.** Our inequalities must hold for *all* arbitrary compact sub-intervals $[t_1, t_2]$. As demonstrated by the counterexamples, this is essential because the inequality is a path-wise condition relating the state at two different

times. Enforcing it only on the global interval $[0, T^*]$ would fail to constrain the trajectory uniquely. In contrast, the weak formulation for measures is stated over the *single time interval* $(0, T^*)$. This is sufficient because the condition must hold for *all* test functions, including those with arbitrarily small support in time. The ability to localize the test function ensures that the single global integral enforces the PDE's validity at every moment.

In essence, our framework for sets requires a more stringent path-constraining condition precisely because the geometry of a set's boundary is more complex than a conserved measure, highlighting a fundamental structural difference from the classical framework.

5.3 A Morphological Characterization of Point-to-Point Set Evolution

The framework of classical ordinary differential equations is foundational to the study of dynamical systems, but it is limited to state spaces that are vector spaces. To analyze the evolution of more general objects, such as changing shapes or deforming geometries, a more powerful theory is needed. The field of set-valued analysis [93] and the theory of differential inclusions [94], pioneered by authors such as Aubin and Frankowska, provide the mathematical foundation for such systems by extending differential calculus to set-valued maps.

Within this broader field, a significant class of problems is described by morphological equations [10], which govern the evolution of compact sets in Euclidean spaces. As detailed in the comprehensive framework of Mutational Analysis, these equations generalize ODEs to describe how a compact set mutates over time based on a given evolutionary rule.

The power of this general framework lies in its ability to handle dynamics governed by set-valued vector fields, where the velocity of a point can be a set, i.e., $\dot{x}(t) \in F(t, A(t))$. This allows

the theory to model very complex phenomena, including the branching of a single point, where the set of possible future trajectories forms a solution funnel emanating from a single initial state.

5.3.1 Primitives and Reachable Sets in Single-Valued Evolutions

The general framework of mutational analysis is designed to handle set-valued dynamics, which can describe complex phenomena such as the branching of trajectories. However, for the specific application of trajectory prediction, an agent's path is assumed to be unique. This physical constraint motivates our specialization of the theory to the case where the evolution is governed by a single-valued vector field.

To make this connection precise, we will now formally define the core objects from the theory of set-valued analysis that are needed to build our argument.

Definition 5.3.1. *A* set-valued map $F\colon \mathbb{R}^d \rightrightarrows \mathbb{R}^d$ *assigns to each* $x \in \mathbb{R}^d$ *a nonempty compact set* $F(x) \subset \mathbb{R}^d$.

For the evolution to be well-behaved, the set-valued map that generates it must satisfy certain regularity conditions, such as uniform boundedness and Lipschitz continuity. This motivates the definition of the following space of maps.

Definition 5.3.2. *Define*

$$\mathrm{LIP}(\mathbb{R}^d, \mathbb{R}^d) := \Big\{ F\colon \mathbb{R}^d \rightrightarrows \mathbb{R}^d : F(x) \textit{ compact}, \sup_{x\in\mathbb{R}^d} \sup_{v\in F(x)} \|v\| < \infty, \ F \textit{ Lipschitz} \Big\},$$

and equip LIP *with the metric*

$$d_\infty(F,G) := \sup_{x\in\mathbb{R}^d} d_H\big(F(x),G(x)\big).$$

The evolution generated by such a map can be understood from a bottom-up, Lagrangian perspective by considering the set of all possible endpoints of individual trajectories. This is known as the reachable set. In the context of shape evolution, the map that generates the reachable set at each time is called a morphological transition.

Definition 5.3.3. *Let $F\colon \mathbb{R}^d \rightrightarrows \mathbb{R}^d$ be set-valued. For an initial compact set $K_0 \subset \mathbb{R}^d$ and time $t \geq 0$, the* reachable set $\vartheta_F(t,K_0)$ *is defined by*

$$\vartheta_F(t,K_0) := \Big\{ x(t)\in\mathbb{R}^d \,:\, \textit{there exists } x(\cdot)\in AC([0,t];\mathbb{R}^d) \textit{ such that } x'(s)\in F\big(x(s)\big) \textit{ for a.e. } s\in[0,t] \textit{ and } x(0)\in K_0\Big\}.$$

Here, $AC(I,X)$ for an interval I is defined to be the set of absolutely continuous curves from I to X.

If $\widehat{F}\colon [0,T^*]\times\mathbb{R}^d \rightrightarrows \mathbb{R}^d$ is time-dependent, one replaces $F(x(s))$ by $\widehat{F}(s,x(s))$ in the above.

A morphological transition $\vartheta\colon [0,1]\times\mathcal{K}(\mathbb{R}^d)\rightrightarrows\mathcal{K}(\mathbb{R}^d)$ *corresponds to a set-valued map $F\in \mathrm{LIP}(\mathbb{R}^d,\mathbb{R}^d)$ so that $\vartheta(\cdot,\cdot)=\vartheta_F(\cdot,\cdot)$.*

An evolution over time traces a path in the space of compact sets. This path is formally called a tube.

Definition 5.3.4. *A* tube *in $\mathbb{R}^d$ is any curve*

$$K(\cdot)\colon [0, T^*] \to \mathcal{K}(\mathbb{R}^d).$$

To create a differential calculus for these tubes, the theory defines a derivative or mutation for a tube. It is the set of all possible set-valued vector fields that are consistent with the tube's infinitesimal motion.

Definition 5.3.5. *Identifying transitions ϑ_F with their generators F, the* mutation *of K at time t is*

$$\mathring{K}(t) := \Big\{ F \in \mathrm{LIP}(\mathbb{R}^d, \mathbb{R}^d) \mid \lim_{h \downarrow 0} \tfrac{1}{h}\, d_H\big(\vartheta_F(h, K(t)),\ K(t+h)\big) = 0 \Big\}.$$

Thus $\mathring{K}\colon [0, T^*] \rightrightarrows \mathrm{LIP}(\mathbb{R}^d, \mathbb{R}^d)$.

The concept of a mutation allows us to rigorously define what it means for a tube to be a solution to an evolution prescribed by a time-varying set-valued map, which provides a top-down, Eulerian approach to set evolution. Such a solution is called a primitive.

Definition 5.3.6. *A tube $K(\cdot)$ is called a* morphological primitive *of a given single-valued map $\mathfrak{F}\colon \mathcal{K}(\mathbb{R}^d) \times [0, T^*] \to \mathrm{LIP}(\mathbb{R}^d, \mathbb{R}^d)$ if $K(\cdot)$ is Lipschitz in d_H and*

$$\mathfrak{F}(K(t), t) \ \in\ \mathring{K}(t),\ \textit{i.e.}\ \lim_{h \downarrow 0} \tfrac{1}{h}\, d_H\big(\vartheta_{\mathfrak{F}(K(t),t)}(h, K(t)),\ K(t+h)\big) = 0\ \textit{a.e. on}\ [0, t].$$

This gives a differential criterion for a tube being a morphological primitive. The following proposition establishes the crucial fact that these two perspectives—the Lagrangian reachable set and the Eulerian primitive—are equivalent.

Proposition 5.3.7. *Suppose* $\mathfrak{F}\colon \mathcal{K}(\mathbb{R}^d) \times [0,T^*] \;\to\; \mathrm{LIP}(\mathbb{R}^d,\mathbb{R}^d)$ *is a Carathéodory function (i.e. continuous in the first argument and measurable in time) satisfying*

$$\sup_{\substack{S\in\mathcal{K}(\mathbb{R}^d)\\ t\in[0,T^*]}} \Big(\sup_{x\in\mathbb{R}^d}\sup_{v\in\mathfrak{F}(S,t)(x)} \|v\| \;+\; \mathrm{Lip}\,\mathfrak{F}(S,t)\Big) \;<\; \infty.$$

Then a continuous tube $K\colon [0,T^*]\to\mathcal{K}(\mathbb{R}^d)$ *is a solution of the morphological equation*

$$\mathring{K}(t) \;\ni\; \mathfrak{F}\big(K(t),t\big)$$

if and only if for each $t\in[0,T^*]$*, the compact set* $K(t)\subset\mathbb{R}^d$ *coincides with the reachable set of the initial compact set* $K(0)\subset\mathbb{R}^d$ *for the non-autonomous differential inclusion*

$$x'(s) \;\in\; \mathfrak{F}\big(K(s),s\big)\big(x(s)\big) \quad \textit{a.e. } s\in[0,t].$$

Proof. See [10], Proposition 70 in Chapter 1. □

Now that we have rigorously defined what a solution is, we must establish that such solutions exist and are unique under appropriate conditions. The following propositions provide the Peano- and Picard–Lindelöf-type theorems for these morphological equations.

Proposition 5.3.8 (Existence of Solutions for Morphological Equations)**.** *Let* $\mathfrak{F} : \mathcal{K}(\mathbb{R}^d)\times[0,T^*]\to\mathrm{LIP}(\mathbb{R}^d,\mathbb{R}^d)$ *be continuous. Assume moreover that* $\mathfrak{F}$ *is uniformly bounded and uniformly Lipschitz in space, i.e.*

$$\sup_{\substack{S\in\mathcal{K}(\mathbb{R}^d)\\ t\in[0,T^*]}} \Big(\sup_{x\in\mathbb{R}^d}\sup_{v\in\mathfrak{F}(S,t)(x)} \|v\| \;+\; \mathrm{Lip}\,\mathfrak{F}(S,t)\Big) \;<\; \infty.$$

For every initial compact set $K_0 \subset \mathbb{R}^N$ there exists at least one Lipschitz tube

$$K\colon [0,T^*] \to \mathcal{K}(\mathbb{R}^N), \qquad K(0) = K_0,$$

satisfying the morphological differential inclusion

$$\mathring{K}(t) \;\ni\; \mathfrak{F}\big(K(t),t\big) \quad \textit{for a.e. } t \in [0,T^*]$$

Proof. See [10], Proposition 71 in Chapter 1. □

Proposition 5.3.9 (Uniqueness of Solutions for Morphological Equations)**.** *Assume we are given a continuous morphological velocity field $\mathfrak{F}\colon \mathcal{K}(\mathbb{R}^d) \times [0,T^*] \to \mathrm{LIP}(\mathbb{R}^d,\mathbb{R}^d)$ i.e. a rule that to every compact set $S \subset \mathbb{R}^d$ and time t assigns a (single-time) set-valued vector field $\mathfrak{F}(S,t) \in \mathrm{LIP}(\mathbb{R}^d,\mathbb{R}^d)$. Suppose:*

1. *Global Lipschitz dependence on the current set:*

$$d_\infty\big(\mathfrak{F}(S_1,t),\,\mathfrak{F}(S_2,t)\big) \;\le\; L\,d_H(S_1,S_2) \quad \forall\, S_1,S_2 \in \mathcal{K}(\mathbb{R}^d),\ \forall\, t \in [0,T^*].$$

2. *Uniform boundedness of speed and spatial Lipschitz constant:*

$$\sup_{\substack{S\in\mathcal{K}(\mathbb{R}^d)\\ t\in[0,T^*]}} \Big(\sup_{x\in\mathbb{R}^d}\sup_{v\in\mathfrak{F}(S,t)(x)} \|v\| \;+\; \mathrm{Lip}\,\mathfrak{F}(S,t)\Big) \;<\; \infty.$$

Then for every initial compact set $K_0 \subset \mathbb{R}^d$ there exists a unique tube

$$K\colon [0,T^*] \to \mathcal{K}(\mathbb{R}^d), \qquad K(0) = K_0,$$

that satisfies the differential inclusion

$$\mathring{K}(t) \ \ni \ \mathfrak{F}\big(K(t),t\big) \quad \textit{for a.e. } t \in [0,T^*]$$

Proof. See [10], Proposition 72 in Chapter 1. □

Finally, we state the stability of these unique solutions with respect to changes in the initial conditions and the governing dynamics.

Proposition 5.3.10 (Continuous Dependence of Solutions for Morphological Equations)**.** *Let $\mathfrak{F},\ \mathfrak{G}\colon \mathcal{K}(\mathbb{R}^d) \times [0,T^*] \to \big(\mathrm{LIP}(\mathbb{R}^N,\mathbb{R}^N), d_\infty\big)$ be two continuous morphological vector fields satisfying*

- *Uniform Lipschitz bound in space:*

$$\widehat{\alpha} := \sup_{S\in\mathcal{K}(\mathbb{R}^d),\, t\in[0,T^*]} \mathrm{Lip}\,\mathfrak{F}(S,t) \ < \ \infty.$$

- *Global Lipschitz dependence on the set argument:*

  there is a constant $L \geq 0$ such that

$$d_\infty\big(\mathfrak{F}(S_1,t),\mathfrak{F}(S_2,t)\big) \ \leq \ L\, d_H(S_1,S_2) \quad \forall\, S_1,S_2 \in \mathcal{K}(\mathbb{R}^d),\ t\in[0,T^*].$$

- *Uniform gap between the two generators:*

$$\delta := \sup_{S \in \mathcal{K}(\mathbb{R}^d),\, t \in [0,T^*]} d_\infty\big(\mathfrak{F}(S,t), \mathfrak{G}(S,t)\big) \;<\; \infty.$$

Let $K_1, K_2 \colon [0,T^*] \to \mathcal{K}(\mathbb{R}^N)$ *be the unique solutions of the two morphological differential inclusions*

$$\mathring{K}_1(t) \;\ni\; \mathfrak{F}\big(K_1(t),t\big), \qquad \mathring{K}_2(t) \;\ni\; \mathfrak{G}\big(K_2(t),t\big), \qquad K_1(0), K_2(0) \in \mathcal{K}(\mathbb{R}^d).$$

Then for every $t \in [0,T^*]$*,*

$$d_H\big(K_1(t),K_2(t)\big) \;\le\; \Big(d_H\big(K_1(0),K_2(0)\big) \;+\; t\,\delta\Big)\, \exp\big[(L+\widehat{\alpha})\,t\big].$$

Proof. See [10], Proposition 73 in Chapter 1. □

We have thus identified two distinct conceptual viewpoints for describing the evolution of a set: the Lagrangian perspective, which builds the evolved set from the trajectories of its individual points, and the Eulerian perspective, which treats the set as a single object in an abstract space.

The main result of this section is to show that for point-to-point evolutions governed by a unique global flow, these two perspectives are equivalent. They are simply different mathematical languages for describing the same, uniquely determined point-to-point evolution.

Proposition 5.3.11 (Equivalence of Lagrangian and Eulerian Characterizations)**.** *Let the evolution of a compact set* $K_0 \subset \mathbb{R}^d$ *be governed by a single-valued vector field* $v(t,x)$ *that satisfies the regularity conditions necessary to guarantee a unique global flow map,* $\Phi_{t,s}$*. The following*

four characterizations of the evolved set, K_t, are equivalent:

1. *The image under the classical ODE flow map (Lagrangian).*
2. *The reachable set $\mathcal{R}_v(t, K_0)$ (Lagrangian).*
3. *The unique primitive of the evolution, $S_t(K_0)$ (Eulerian).*
4. *The unique point-to-point set evolution that satisfies the two-sided functional inequality from Theorem 5.2.3 (Eulerian).*

Proof. (1) $\iff$ (2): The equivalence of the reachable set and the flow map image is a direct consequence of the uniqueness of the ODE solution.

(2) $\iff$ (3): Proposition 5.3.7 applies with the choice $\mathfrak{F}(S,t)(x) = \{\, v(t,x) \,\}$, i.e. we treat the single-valued field v as a set-valued map of singletons. The proposition then asserts that a tube $K(\cdot)$ is a primitive of $\mathfrak{F}$ precisely if, for every t, it coincides with the reachable set generated by $\mathfrak{F}$. Hence, $\mathcal{R}_v(t, K_0) = S_t(K_0)$, so the reachable set $\mathcal{R}_v(t, K_0)$ and the primitive $S_t(K_0)$ are identical.

(1) $\iff$ (4): The equivalence of the flow map image and the solution to the two-sided functional inequality is the main result of Theorem 5.2.3. The proof of that theorem established that (a) the set generated by the unique flow satisfies the inequalities, and (b) any point-to-point set evolution that satisfies the inequalities must be identical to the one generated by the unique flow. Hence, the result extends immediately to any setting that guarantees a unique global flow map $\Phi_{t,s}$. □

5.3.2 The Autonomous Point-to-Point Set Evolution

Having established the theoretical framework for set evolution, we now define the specific autonomous, state-dependent system at the heart of our proposed method. We model the interactive dynamics of a group of agents as a point-to-point set flow, where the velocity of each agent is determined by the configuration of the entire group at that instant.

Formally, let $K_t \in \mathcal{K}(\mathbb{R}^d)$ be the compact set representing all agent positions at time t. The evolution of any point $x(t) \in K_t$ is governed by the ordinary differential equation:

$$\dot{x}(t) = v[K_t](x(t))$$

Here, v is a map that takes an entire compact set K_t as input and outputs a continuous vector field, $v[K_t]\colon \mathbb{R}^d \to \mathbb{R}^d$, which then determines the velocity for every point in the set. This is an autonomous system in the sense that the map v does not explicitly depend on time, but it is state-dependent, as the dynamics are determined by the instantaneous geometry of the set K_t.

To ensure that this formulation leads to a unique, well-defined point-to-point set evolution (a unique primitive in the language of the previous sections), by Proposition 5.3.9, the map v must satisfy a Lipschitz condition. Specifically, the map from the space of compact sets to the space of vector fields, $v\colon \mathcal{K}(\mathbb{R}^d) \to \mathrm{Lip}(\mathbb{R}^d, \mathbb{R}^d)$, must be Lipschitz continuous.

We ensure this crucial Lipschitz-in-set condition holds by neural network model construction. Our model for the map $K \mapsto v[K]$ is a deep neural network architecture designed to be set-equivariant as in Theorem 3.5.6. This architecture is motivated by the ultimate goal of our framework: to model complex dynamics with memory, in the spirit of Retarded Functional Dif-

ferential Equations (RFDEs), while remaining computationally tractable.

To achieve this, we adopt an encoder–decoder approach.

1. First, an encoder network, which is a causal and permutation-equivariant operator as in Conjecture 5.1.5, summarizes the entire past trajectory of each agent into a fixed-size latent vector in $\mathbb{R}^{\ell}$.

2. This encoded history is then combined with the agent's current position to form an augmented state in a higher-dimensional space, $\mathbb{R}^{\ell+d}$.

3. Finally, the future evolution is predicted by a decoder, which is a neural ODE that operates on the set of these augmented states.

Crucially, the vector field of this neural ODE,

$$\dot{y}(t) = v_{\mathrm{NN}}[K_t]\big(y(t)\big),$$

—where K_t is now the set of all agents' augmented states—has exactly the autonomous, point-to-point structure analyzed in this section. As we will detail in the next chapter, the set-equivariant design of the neural network v_{NN} for the dynamic context decoder guarantees that the necessary Lipschitz-in-set condition is satisfied by using a Lipschitz continuous activation function, ensuring the learned dynamics are well-posed and produce a unique evolution.

5.3.3 Comparison with Measure-Theoretic Autonomous Transport

Having established the properties of the autonomous point-to-point set evolution, we now briefly compare this mass-free framework to measure-theoretic approaches. Several powerful

frameworks exist for modeling state-dependent dynamics on spaces of measures, though they differ significantly in their mathematical settings. For instance, [57] developed a comprehensive theory for transport equations with sources and sinks on the space of non-negative Radon measures, $\mathcal{M}_+(\mathbb{R}^d)$, built upon a generalized p-Wasserstein distance which they introduced. A distinct and more general approach is found in the mutational analysis frameworks of [10, 95], who study a broader class of evolutionary problems on the space of all Radon measures, $\mathcal{M}(\mathbb{R}^d)$, employing the dual $W^{1,\infty}$ metric.

More recently, [96] extends the theory of [57] to the space of signed Radon measures with compact support, introducing a Wasserstein norm for this broader space. Significantly, this new Wasserstein norm provides a physical interpretation for the dual $W^{1,\infty}$ metric, which can be viewed as a special case—a connection that was previously lacking. While their methodologies and scope differ, both the mutational and the generalized Wasserstein frameworks provide robust tools for analyzing dynamics where the evolution depends on the entire state of the measure.

The first fundamental contrast lies in the nature of the object being evolved: measure evolution allows continuous mass creation and destruction whereas set evolution does not. A general form of non-conservative transport equation is:

$$\partial_t \mu_t + \nabla \cdot (v_t \mu_t) = g_t \mu_t.$$

This equation models two distinct phenomena. The divergence term, $\nabla \cdot (v_t \mu_t)$, is the transport or advection term; it describes how the mass distribution μ_t is moved by the velocity field v_t. The term $g_t \mu_t$ is a source/sink term that describes the local creation ($g_t > 0$) or destruction ($g_t < 0$) of mass at a rate determined by the scalar function g_t. The entire framework is built around the

transport and evolution of a mass distribution.

Our framework, by contrast, is fundamentally mass-free. Due to the binary nature of sets—a point is either included or not—there is no mechanism for the continuous creation or destruction of points within the evolution itself. A point cannot evolve into multiple points (branching) or merge with another point while remaining a point-to-point evolution. In practice, the source and sink of agents are discontinuous events that arise from sensor limitations, such as an agent entering a camera's field of view or from sensor failure such as a GPS signal failing. These phenomena are handled separately; for instance, boundary-hitting events due to sensor limitations can be modeled using the language of survival analysis (censoring and truncation).

In the set-based framework, the velocity of a point $x(t)$ is determined by a vector-field generated from a map

$$v\colon \mathcal{K}(\mathbb{R}^d) \;\to\; \mathrm{Lip}(\mathbb{R}^d, \mathbb{R}^d),$$

which depends on the geometry of the entire set $K(t)$ at that moment. Similarly, in the measure-theoretic setting, the velocity field $v[\mu(t)]$ and growth rate $g[\mu(t)]$ are functions of the entire measure $\mu(t)$, governed by maps such as

$$v\colon \mathcal{M}_{+,c}(\mathbb{R}^d) \;\to\; \mathrm{Lip}(\mathbb{R}^d, \mathbb{R}^d) \quad \text{and} \quad g\colon \mathcal{M}_{+,c}(\mathbb{R}^d) \;\to\; \mathrm{Lip}(\mathbb{R}^d, \mathbb{R}).$$

This nonlocal dependence, where individual dynamics are influenced by the state of the entire population, is particularly useful for modeling collective behavior in the social sciences [57].

Furthermore, both frameworks require a Lipschitz-type condition on these governing dynamics to ensure that the evolution is well-posed. For a given initial state—whether a compact

set K_0 or a measure μ_0—a unique solution is guaranteed only if the map from the state to the vector field v is Lipschitz continuous with respect to the underlying metric of the space (the Hausdorff distance for sets or the Wasserstein distance for measures). For the measure-theoretic case, the coefficients $v[\mu]$ and $g[\mu]$ are assumed to be Lipschitz continuous maps from the space of measures to the space of vector fields.

This distinction in the required Lipschitz conditions highlights a deeper truth about the two frameworks. In the measure-theoretic case, the entire distribution can influence the dynamics. For set evolution, however, the information is strictly localized. The following proposition makes this precise by demonstrating that the vector field's behavior outside the set's evolving support is irrelevant.

Proposition 5.3.12 (Sufficiency of the Pointed-Set Domain for Set Evolution)**.** *Let* $v, w \colon \mathcal{K}(\mathbb{R}^d) \to \mathrm{Lip}(\mathbb{R}^d, \mathbb{R}^d)$ *be two uniformly Lipschitz maps from compact sets to Lipschitz velocity fields. Assume that* v *and* w *agree on pointed sets, i.e. for every compact* $K \subset \mathbb{R}^d$ *and every* $x \in K$*,*

$$v[K](x) = w[K](x).$$

Let $K_0 \subset \mathbb{R}^d$ *be any initial compact set. Denote by* K_t^v *(resp.* K_t^w*) the evolution of* K_0 *under* v *(resp.* w*). Then the two evolutions coincide for all* $t \geq 0$*:*

$$K_t^v = K_t^w, \qquad \forall\, t \geq 0.$$

Proof. Because v and w coincide on every pointed set (x, K) with $x \in K$, the two differential inclusions driving the tubes K^v and K^w are identical along the trajectory itself. By Proposition

5.2.2, there is only one tube starting from K_0 that satisfies the inclusion, hence $K_t^v = K_t^w$ for all $t \geq 0$. □

Remark 5.3.13. *The evolution of a set depends only on the vector-field values* on the set itself. *How v and w behave off the set—that is, at (x, K) with $x \notin K$—is irrelevant. This mirrors the phenomenon in Theorem 3.5.5: $\mathrm{Pt}\big(\mathcal{K}(\mathbb{R}^d)\big)$ is a proper closed subset of $\mathbb{R}^d \times \mathcal{K}(\mathbb{R}^d)$ whereas $\mathrm{Pt}\big(\mathcal{M}_{+,c}(\mathbb{R}^d)\big)$ is a dense subset of $\mathbb{R}^d \times \mathcal{M}_{+,c}(\mathbb{R}^d)$. This self-contained property sharply distinguishes the set-based approach from classical measure-based transport theories.*

Table 5.1: Comparison of autonomous measure vs. set evolutions

Property	**Autonomous Measure Evolution**	**Autonomous Set Evolution**
Evolving Object	$\mu_t \in \mathcal{M}_c(\mathbb{R}^d)$	$K_t \in \mathcal{K}(\mathbb{R}^d)$
Governing Equation	$\partial_t \mu_t + \nabla \cdot \big(v[\mu_t]\, \mu_t\big) = g[\mu_t]\, \mu_t$	$v[K_t] \ \in \ \mathring{K}_t$
Vector-Field Type	$v : \mathcal{M}_c(\mathbb{R}^d) \to \mathrm{Lip}(\mathbb{R}^d, \mathbb{R}^d)$ $g : \mathcal{M}_c(\mathbb{R}^d) \to \mathrm{Lip}(\mathbb{R}^d, \mathbb{R})$	$v : \mathcal{K}(\mathbb{R}^d) \to \mathrm{Lip}(\mathbb{R}^d, \mathbb{R}^d)$
Required Domain to Define Evolution	$v[\mu](\cdot)$ and $g[\mu](\cdot)$ must be defined on all $(x, \mu) \in \mathbb{R}^d \times \mathcal{M}_c(\mathbb{R}^d)$	$v[K](\cdot)$ needs definition only on $\mathrm{Pt}(\mathcal{K}(\mathbb{R}^d))$
Key Implication	A point x can depend on mass located far away (even outside of the support of μ).	A point $x \in K$ moves purely from the geometry of K; dynamics are strictly self-contained.

5.3.4 RFDEs for Hyperspaces

Our discussion has so far focused on autonomous systems where the dynamics depend on the instantaneous state of the entire set, $K(t)$. This concept of state-dependency can be understood as a form of spatial or geometric memory: the velocity of each point is informed by the geometric configuration of all other points at the same instant. This is powerfully analogous to the temporal memory found in classical Retarded Functional Differential Equations (RFDEs), where the velocity $\dot{x}(t)$ depends on the history of the state over a preceding time interval $[t-r, t]$ or $(-\infty, t]$.

The framework of Mutational Analysis is particularly powerful because it formally unifies these concepts. The theory extends naturally to mutational equations with delay, providing a rigorous foundation for systems where the governing dynamics depend not on the instantaneous state $K(t)$, but on its recent history.

Proposition 5.3.14 (Existence and Uniqueness for Morphological Equations with Delay)**.** *Suppose a fixed delay $r \geq 0$ is given, and a continuous map*

$$F\colon C\big([-r,0], \mathcal{K}(\mathbb{R}^d)\big) \times [0, T^*] \;\to\; \mathrm{LIP}(\mathbb{R}^d, \mathbb{R}^d)$$

satisfies the following:

1. *Uniform Bounds:*

$$\sup_{M(\cdot),\, t} \Big(\|F(M(\cdot), t)\|_\infty \;+\; \mathrm{Lip}\big(F(M(\cdot), t)\big) \Big) \;<\; \infty.$$

2. *Lipschitz Continuity of F: There exists a constant $\Lambda \geq 0$ such that for Lebesgue-almost every $t \in [0, T^*]$ and all histories $M_1(\cdot), M_2(\cdot) \in C([-r, 0], \mathcal{K}(\mathbb{R}^d))$,*

$$d_\infty\big(F(M_1(\cdot), t),\ F(M_2(\cdot), t)\big) \ \leq\ \Lambda \sup_{s\in[-r,0]} d_H\big(M_1(s), M_2(s)\big).$$

Then for every initial history $K_0(\cdot) \in C\big([-r, 0], \mathcal{K}(\mathbb{R}^d)\big)$, there exists a unique curve

$$K\colon [0, T^*] \ \to\ \mathcal{K}(\mathbb{R}^d)$$

satisfying:

- *$K(\cdot)$ is Lipschitz continuous with respect to d_H.*
- *The solution respects the initial history: $K(t)\big|_{[-r,0]} = K_0(\cdot)$.*
- *For Lebesgue-almost every $t \in [0, T^*]$, K solves the delayed morphological equation*

$$\mathring{K}(t)\ \ni\ F\big(K(t+\cdot),\ t\big)\big|_{[-r,0]}\,.$$

Proof. See [97], Theorems 4.1 and 4.2.

Since K_0 is continuous on $[-r, 0]$ and F is Lipschitz (hence uniformly continuous), and since the quantities M and c defined below are finite and, by Proposition 3.1.4, closed balls in $\mathcal{K}(\mathbb{R}^d)$ are compact under the Hausdorff metric, the existence of a solution follows from Theorem

4.1 in [97]. Specifically, define

$$M = \sup_{\substack{\phi \in C([-r,0],\mathcal{K}(\mathbb{R}^d)) \\ t \in [0,T^*]}} \alpha(F(\phi,t)) \leq \sup_{\substack{\phi \in C([-r,0],\mathcal{K}(\mathbb{R}^d)) \\ t \in [0,T^*]}} \mathrm{Lip}\big(F(\phi,t)\big) < +\infty.$$

and

$$c = \sup_{\substack{\phi \in C([-r,0],\mathcal{K}(\mathbb{R}^d)) \\ t \in [0,T^*]}} \beta(F(\phi,t)) \leq \sup_{\substack{\phi \in C([-r,0],\mathcal{K}(\mathbb{R}^d)) \\ t \in [0,T^*]}} \|F(\phi,t)\|_\infty < +\infty.$$

Moreover, since F is Lipschitz and M is finite, the uniqueness of the solution is ensured by Theorem 4.2 in [97]. □

Having established a well-posedness result for morphological equations with delay using the classical existence theorem, Theorem 4.1. in [97], we now turn to a more complex class of problems: nonlinear transport equations. This new setting, which models the evolution of distributions or densities, requires a shift in our analytical framework.

A significant topological challenge arises when moving to the space of Radon measures, $\mathcal{M}(\mathbb{R}^d)$. The existence theorem from [97] fundamentally relies on the compactness of closed and bounded balls in the underlying state space. This condition, which holds for the space of compact sets $\mathcal{K}(\mathbb{R}^d)$ fails for $\mathcal{M}(\mathbb{R}^d)$ when equipped with the natural metric for this problem. A ball defined by the dual $W^{1,\infty}$ metric is never guaranteed to be tight and, therefore, is never compact.

Mutational analysis replaces the classical requirement that all closed balls are compact with *Euler compactness*, a weaker notion tuned to the Euler scheme and to spaces that carry a family of pseudo-metrics $\big(d_j\big)_{j\in\mathcal{J}}$. For each tuple of a point $x_0 \in E$, a horizon $T > 0$, and positive bounds $\hat{\alpha}_j, \hat{\beta}_j, \hat{\gamma}_j$ $(j \in \mathcal{I})$, the associated generalized ball $\mathcal{N}\big(x_0,\, T,\, (\hat{\alpha}_j, \hat{\beta}_j, \hat{\gamma}_j)_{j\in\mathcal{I}}\big)$, the set of all

curves $y(\cdot)\colon [0,T^*] \to E$ obtained by a finite concatenation of transitions satisfying some bound conditions has the property that, for every $t \in [0,T^*]$, the evaluation set $\{\, y(t) \mid y(\cdot) \in \mathcal{N} \}\ \subset E$ is sequentially precompact with respect to each pseudo-metric d_j: every sequence drawn from that set admits a d_j-convergent subsequence (its limit need not lie in the evaluation set).

To precisely state the result for transport equations, we must first define the functional spaces and the metric involved.

Definition 5.3.15. *Let I be a nonempty interval of $\mathbb{R}$. The set of all bounded, Lipschitz continuous functions is denoted* $\mathrm{BLip}\big(I, \mathcal{M}(\mathbb{R}^d)\big)$.

Definition 5.3.16. *The Sobolev space $W^{1,\infty}(\mathbb{R}^m,\mathbb{R}^n)$ consists of all the bounded Lipschitz continuous functions whose weak derivatives are essentially bounded. Indeed, the precise definition is*

$$W^{1,\infty}(\mathbb{R}^m,\mathbb{R}^n) = \{u \in L^\infty(\mathbb{R}^m,\mathbb{R}^n) : D_j u \in L^\infty(\mathbb{R}^m,\mathbb{R}^n),\ \forall j = 1,\dots,m\}\,.$$

The metric $\rho_{\mathcal{M}}$ on $\mathcal{M}(\mathbb{R}^d)$ defined by

$$\rho_{\mathcal{M}}(\mu,\nu) = \sup\Big\{ \int_{\mathbb{R}^d} \psi\ d(\mu-\nu) : \psi \in C^1(\mathbb{R}^d),\ \|\psi\|_\infty,\ \|\nabla\psi\|_\infty \le 1 \Big\}$$

is called the $W^{1,\infty}$ dual metric $\rho_{\mathcal{M}}$ on $\mathcal{M}(\mathbb{R}^d)$.

With the necessary functional analytic setting now established, we can state the main existence and uniqueness result for the nonlinear transport equation with delay. The proposition considers a system governed by a velocity field f_1 and a source term f_2, which are required to belong to the Sobolev space $W^{1,\infty}$ to ensure appropriate regularity. The dynamics at any given time depend on the state's recent history, which is formalized as a function in $\mathrm{BLip}([-r,0],\mathcal{M}(\mathbb{R}^d))$.

Crucially, the well-posedness of the problem hinges on a Lipschitz condition imposed on this feedback map, where the distance between measure-histories is quantified by the $W^{1,\infty}$ dual metric, $\rho_{\mathcal{M}}$.

Proposition 5.3.17 (Existence and Uniqueness for Nonlinear Transport Equations with Delay)**.** *Suppose for a fixed delay $r \geq 0$, a continuous map*

$$F = (f_1, f_2) \colon \mathrm{BLip}\big([-r, 0],\, \mathcal{M}(\mathbb{R}^d)\big) \times [0, T^*] \;\to\; W^{1,\infty}(\mathbb{R}^d, \mathbb{R}^d) \;\times\; W^{1,\infty}(\mathbb{R}^d, \mathbb{R})$$

satisfies the following conditions:

1. *Uniform Bounds:*

$$M := \sup_{\mu(\cdot),\, t} \Big(\|f_1(\mu(\cdot), t)\|_{W^{1,\infty}} \;+\; \|f_2(\mu(\cdot), t)\|_{W^{1,\infty}} \Big) \;<\; \infty.$$

2. *Lipschitz Continuity of F: There exists $\Lambda \geq 0$ such that for Lebesgue-a.e. t and all $\mu^{(1)}(\cdot), \mu^{(2)}(\cdot) \in \mathrm{BLip}([-r, 0], \mathcal{M}(\mathbb{R}^d))$,*

$$\|F(\mu^{(1)}(\cdot), t) - F(\mu^{(2)}(\cdot), t)\|_{L^\infty} \;\leq\; \Lambda \sup_{s \in [-r, 0]} \rho_{\mathcal{M}}\big(\mu^{(1)}(s), \mu^{(2)}(s)\big).$$

Then for every initial history $\mu_0(\cdot) \in \mathrm{BLip}([-r, 0], \mathcal{M}(\mathbb{R}^d))$, there exists a unique *narrowly-continuous curve of measures*

$$\mu(\cdot) \colon [0, T^*] \;\to\; \mathcal{M}(\mathbb{R}^d)$$

such that:

- *The family* $\{\mu(t)\}_{t\in[0,T^*]}$ *is tight and has uniformly bounded total variation.*

- $\mu(t)\big|_{[-r,0]} = \mu_0(\cdot)$.

- *For Lebesgue-a.e.* t*, the restriction* $\mu(\cdot)\big|_{[0,T^*]}$ *is a distributional solution of the delayed transport equation*

$$\partial_t \mu(t) \;+\; \nabla_x \cdot \big(f_1\big(\mu(t+\cdot)\big|_{[-r,0]}, t\big)\,\mu(t)\big) \;=\; f_2\big(\mu(t+\cdot)\big|_{[-r,0]}, t\big)\,\mu(t).$$

Equivalently, for every test function $\varphi \in C_c^\infty(\mathbb{R}^d)$*, the solution satisfies*

$$\begin{aligned}
&\int_{\mathbb{R}^d} \varphi\, d\mu(t) - \int_{\mathbb{R}^d} \varphi\, d\mu(0) \\
&= \int_0^t \int_{\mathbb{R}^d} \Big(\nabla\varphi(x)\cdot f_1\big(\mu(s+\cdot)\big|_{[-r,0]}, s\big)(x) \;+\; \varphi(x)\, f_2\big(\mu(s+\cdot)\big|_{[-r,0]}, s\big)(x)\Big)\, d\mu(s)(x)\, ds.
\end{aligned}$$

Proof. See [10], Proposition 23, Proposition 51, Lemma 52, Theorem 53 in Chapter 2 and see [97], Theorem 4.2.

Lemma 52 in [10] shows that $\mathcal{M}(\mathbb{R}^d)$ is Euler compact with the metric $\rho_{\mathcal{M}}$ and the total variation $|\cdot|$.

The existence of the solution follows by showing the four conditions specified in Proposition 23 in [10].

By Proposition 51 in [10], for each $R > 0$,

1.)

$$\sup_{\substack{\phi\in C([-r,0],\mathcal{M}(\mathbb{R}^d))\\ t\in[0,T^*]}} \alpha(F(\phi,t);R)$$

$$= \sup_{\substack{\phi\in C([-r,0],\mathcal{M}(\mathbb{R}^d))\\ t\in[0,T^*]}} \big(\text{Lip } f_1(\phi,t) + \|f_2(\phi,t)\|_{W^{1,\infty}}\big)$$

$$\leq \sup_{\substack{\phi\in C([-r,0],\mathcal{M}(\mathbb{R}^d))\\ t\in[0,T^*]}} \big(\|f_1(\phi,t)\|_{W^{1,\infty}} + \|f_2(\phi,t)\|_{W^{1,\infty}}\big) = M < +\infty,$$

2.)

$$\sup_{\substack{\phi\in C([-r,0],\mathcal{M}(\mathbb{R}^d))\\ t\in[0,T^*]}} \beta(F(\phi,t);R)$$

$$= \sup_{\substack{\phi\in C([-r,0],\mathcal{M}(\mathbb{R}^d))\\ t\in[0,T^*]}} \big(\|f_1(\phi,t)\|_\infty + \|f_2(\phi,t)\|_\infty\big)e^{\|f_2(\phi,t)\|_\infty}R \leq Me^M R < +\infty,$$

3.)

$$\sup_{\substack{\phi\in C([-r,0],\mathcal{M}(\mathbb{R}^d))\\ t\in[0,T^*]}} \gamma(F(\phi,t))$$

$$= \sup_{\substack{\phi\in C([-r,0],\mathcal{M}(\mathbb{R}^d))\\ t\in[0,T^*]}} \|f_2(\phi,t)\|_\infty \leq M < +\infty,$$

4.) We must show $\lim_{n\to\infty} D_j(F(\phi_n,t_n),F(\phi,t);R) = 0$ whenever $(\phi_n,t_n)\to(\phi,t)$. From Proposition 51 in [10], we know that

$$D_j(F(\phi_n,t_n),F(\phi,t);R) \leq (\|f_1(\phi_n,t_n)-f_1(\phi,t)\|_\infty + \|f_2(\phi_n,t_n)-f_2(\phi,t)\|_\infty)\cdot R.$$

This means we need to show that $\|F(\phi_n, t_n) - F(\phi, t)\|_\infty \to 0$ for a.e. $t \in [0, T^*]$. We use the triangle inequality:

$$\|F(\phi_n, t_n) - F(\phi, t)\|_\infty \leq \|F(\phi_n, t_n) - F(\phi_n, t)\|_\infty + \|F(\phi_n, t) - F(\phi, t)\|_\infty.$$

The second term converges to zero due to Lipschitz Continuity of F and the fact that $\phi_n \to \phi$. The first term converges to zero assuming standard continuity of F with respect to time t. Thus, the continuity condition is satisfied.

Moreover, since F is Lipschitz and

$$\sup_{\substack{\phi \in C([-r,0], \mathcal{M}(\mathbb{R}^d)) \\ t \in [0, T^*]}} \alpha(F(\phi, t)) = \sup_{\substack{\phi \in C([-r,0], \mathcal{M}(\mathbb{R}^d)) \\ t \in [0, T^*]}} \left(\text{Lip } f_1(\phi, t) + \|f_2(\phi, t)\|_{W^{1,\infty}} \right)$$

is finite, the uniqueness of the solution is ensured by Theorem 4.2 in [97].

The preceding steps guarantee a unique mutational solution. By Theorem 53 in [10], this unique mutational solution is equivalent to the distributional solution. □

The preceding propositions (Proposition 5.3.14 and Proposition 5.3.17) concerned the evolution of a single object—either a compact set $K(t)$ or a measure $\mu(t)$—whose dynamics were determined by its own history. We now turn to a more intricate class of systems that captures the co-evolution of an entire set of trajectories. This represents a fundamental shift: instead of one entity evolving, we have a population of interacting agents where the evolution of each member is coupled to the collective state of all other members.

To understand the practical meaning of this system, consider a multi-agent prediction problem, such as modeling pedestrians in a crowded plaza. In the governing equation $\dot{x}(t) =$

$F(t, x_t, K_t)$:

- $x(t) \in \mathbb{R}^d$ represents the physical state (e.g., position and velocity) of a single pedestrian at time t.
- The history segment $x_t \in C([-r, 0], \mathbb{R}^d)$ represents that pedestrian's recent path and gait over the last r seconds, capturing their walking speed, changes in direction, and intended trajectory.
- The set of histories $K_t \in \mathcal{K}\big(C([-r, 0, \mathbb{R}^d)\big)$ represents the collective social context or crowd dynamic over the last r seconds. It is the set of all recent paths of all other people in the scene.
- The function F represents the law of interaction or walking policy. It dictates how a person adjusts their speed and direction based on their own recent path (x_t), the movements and histories of the surrounding crowd (K_t), and possibly external factors like a crosswalk signal or the time of day (t).

A key property of physical laws is that they are consistent over time. To model a system with such time-homogeneity, the rule F must be autonomous, meaning it does not explicitly depend on t. Such autonomous systems are time-equivariant: if the initial conditions are shifted in time, the entire solution is identically shifted. The theorem that follows is stated for the more general non-autonomous case $F(t, x_t, K_t)$, which can model systems with external, time-varying influences. The autonomous case is the special instance where F is constant in its first argument: $F(x_t, K_t)$.

Theorem 5.3.18 (Existence and Uniqueness for Coupled Morphological Equations with Delay)**.** *Let* $F : [0, T^*] \times \mathrm{Pt}\big(C([-r, 0], \mathbb{R}^d)\big) \to \mathbb{R}^d$ *be a continuous function which is Lipschitz continuous in state: There exists a constant* $L > 0$ *such that for all* $t \in [0, T^*]$ *and for any two states* $(\phi_1, K_1), (\phi_2, K_2) \in \mathrm{Pt}\big(C([-r, 0], \mathbb{R}^d)\big)$*, the following inequality holds:*

$$\|F(t, \phi_1, K_1) - F(t, \phi_2, K_2)\| \leq L \cdot (\|\phi_1 - \phi_2\|_\infty + d_H(K_1, K_2))$$

Then for any initial condition $K_0 \in \mathcal{K}\big(C([-r, 0], \mathbb{R}^d)\big)$*, there exists a unique family of trajectories* $\{x^\phi(\cdot)\}_{\phi \in K_0}$*, which in turn defines a unique evolution curve of history sets* $K(t) = \{x_t^\phi : \phi \in K_0\}$ *for* $t \in [0, T^*]$.

Proof. The proof strategy is to reformulate the initial value problem as a fixed-point problem for an integral operator $\mathcal{T}$ on a complete metric space of solution curves. We will then show that for a sufficiently small time horizon, this operator is a contraction mapping. The existence and uniqueness of a solution then follow directly from the contraction mapping theorem.

Observe that $C([-r, 0], \mathbb{R}^d)$ equipped with the supremum norm is a complete metric space as a Banach space. By Proposition 3.1.4, since $C([-r, 0], \mathbb{R}^d)$ is a complete metric space, $\mathcal{K}\big(C([-r, 0], \mathbb{R}^d)\big)$ is also a complete metric space.

For a given time horizon $\delta > 0$, we define the space of candidate solution paths as

$$\mathcal{S}_\delta := C([0, \delta], \mathcal{K}\big(C([-r, 0], \mathbb{R}^d)\big)).$$

This is the space of all continuous curves of history sets. We equip $\mathcal{S}_\delta$ with the supremum metric

$d_{\mathcal{S}_\delta}$, defined for any two curves $K_1(\cdot), K_2(\cdot) \in \mathcal{S}_\delta$ as

$$d_{\mathcal{S}_\delta}(K_1(\cdot), K_2(\cdot)) := \sup_{t\in[0,\delta]} d_H(K_1(t), K_2(t)).$$

Since $\mathcal{K}\big(C([-r, 0], \mathbb{R}^d)\big)$ is a complete metric space, the space of continuous functions into it, $(\mathcal{S}_\delta, d_{\mathcal{S}_\delta})$, is also a complete metric space. This provides the necessary setting for the fixed-point argument.

We reformulate the initial value problem using an integral operator $\mathcal{T}$ on the space $\mathcal{S}_\delta$. Let $K(\cdot) \in \mathcal{S}_\delta$ be a candidate solution curve satisfying the initial condition $K(0) = K_0$. We define a new curve $\tilde{K}(\cdot) = \mathcal{T}(K(\cdot))$ through the following two-step process:

1. **Solve for Individual Trajectories:** For each initial history $\phi \in K_0$, we solve the following standard (uncoupled) functional differential equation, where the set-valued argument is provided by the known curve $K(\cdot)$:

$$\dot{x}^\phi(t) = F(t, x_t^\phi, K(t)), \quad \text{(with initial history } x_0^\phi = \phi)$$

   Under the given continuity and Lipschitz assumptions on F, standard functional differential equation theory guarantees that a unique continuous solution $x^\phi(t)$ exists on $[-r, \delta]$.

2. **Construct the New Set Curve:** We assemble the history segments of these newly found trajectories to define the image curve $\tilde{K}(t)$ at each time $t \in [0, \delta]$:

$$\tilde{K}(t) := \{(x^\phi)_t \mid \phi \in K_0\}$$

A solution to our original coupled problem is a curve $K(\cdot)$ that is a fixed point of this operator, i.e.,

$$\mathcal{T}(K(\cdot)) = K(\cdot).$$

We must show that for a sufficiently small $\delta > 0$, $\mathcal{T}$ is a contraction mapping on $(\mathcal{S}_\delta, d_{\mathcal{S}_\delta})$. Let $K_1(\cdot)$ and $K_2(\cdot)$ be two curves in $\mathcal{S}_\delta$ both starting at K_0. Let $\tilde{K}_1(\cdot) = \mathcal{T}(K_1(\cdot))$ and $\tilde{K}_2(\cdot) = \mathcal{T}(K_2(\cdot))$.

For each $\phi \in K_0$, we have two associated trajectories, $x^\phi_{(1)}$ and $x^\phi_{(2)}$, satisfying their respective integral equations.

$$x^\phi_{(1)}(t) = x^\phi_{(1)}(0) + \int_0^t F(s, x^\phi_{(1)s}, K_1(s))\, ds$$
$$x^\phi_{(2)}(t) = x^\phi_{(2)}(0) + \int_0^t F(s, x^\phi_{(2)s}, K_2(s))\, ds$$

For any $t \in [0, \delta]$, their difference is bounded as follows:

$$\begin{aligned}\|x^\phi_{(1)}(t) - x^\phi_{(2)}(t)\| &\leq \int_0^t \|F(s, x^\phi_{(1)s}, K_1(s)) - F(s, x^\phi_{(2)s}, K_2(s))\|\, ds, \\ &\leq L \int_0^t \Big(\|(x^\phi_{(1)})_s - (x^\phi_{(2)})_s\|_\infty + d_H(K_1(s), K_2(s)) \Big)\, ds.\end{aligned}$$

where the second line follows from the Lipschitz condition on F.

Let $v_\phi(t) := \sup_{s\in[0,t]} \|(x^\phi_{(1)})_s - (x^\phi_{(2)})_s\|_\infty$. Since

$$\|(x^\phi_{(1)})_s - (x^\phi_{(2)})_s\|_\infty = \sup_{\sigma\in[-\tau,s]} \|x^\phi_1(\sigma) - x^\phi_2(\sigma)\|$$

and the histories match for $\sigma \leq 0$, we have

$$v_\phi(t) = \sup_{\sigma \in [0,t]} \|x_1^\phi(\sigma) - x_2^\phi(\sigma)\|.$$

The integral inequality implies:

$$v_\phi(t) \leq L \int_0^t (v_\phi(s) + d_{\mathcal{S}_\delta}(K_1, K_2))\, ds = Lt \cdot d_{\mathcal{S}_\delta}(K_1, K_2) + L \int_0^t v_\phi(s) ds$$

Note that $t \mapsto Lt \cdot d_{\mathcal{S}_\delta}(K_1, K_2)$ is nondecreasing. Applying Gronwall's inequality to the function $v_\phi(t)$ with the non-negative function L yields:

$$v_\phi(t) \leq Lt \cdot d_{\mathcal{S}_\delta}(K_1, K_2) \exp\Big(\int_0^t L ds \Big) = (Lt \cdot d_{\mathcal{S}_\delta}(K_1, K_2)) e^{Lt}$$

This bound on the individual histories allows us to bound the Hausdorff distance between the resulting sets of histories:

$$\begin{aligned} d_H(\tilde{K}_1(t), \tilde{K}_2(t)) &= d_H\left(\{(x_{(1)}^\phi)_t \mid \phi \in K_0\}, \{(x_{(2)}^\phi)_t \mid \phi \in K_0\} \right) \\ &\leq \sup_{\phi \in K_0} \|(x_{(1)}^\phi)_t - (x_{(2)}^\phi)_t\|_\infty \leq v_\phi(t) \leq (Lt \cdot d_{\mathcal{S}_\delta}(K_1, K_2)) e^{Lt} \end{aligned}$$

Taking the supremum over $t \in [0, \delta]$ gives the final estimate for the distance between the curves:

$$d_{\mathcal{S}_\delta}(\tilde{K}_1, \tilde{K}_2) \leq (L\delta e^{L\delta}) \cdot d_{\mathcal{S}_\delta}(K_1, K_2)$$

By choosing $\delta > 0$ small enough such that the contraction factor $\kappa = L\delta e^{L\delta} < 1$, the

operator $\mathcal{T}$ is a contraction mapping.

We have established that for a sufficiently small $\delta > 0$, the operator $\mathcal{T}$ is a contraction mapping on the non-empty complete metric space $(\mathcal{S}_\delta, d_{\mathcal{S}_\delta})$. By the contraction mapping theorem, $\mathcal{T}$ has a unique fixed point $K(\cdot) \in \mathcal{S}_\delta$. This fixed point is the unique solution to the integral equation system, and therefore the unique solution to the coupled initial value problem on the interval $[0, \delta]$.

Because the Lipschitz constant L is uniform in time, the same contraction factor $\kappa = L\delta e^{L\delta} < 1$ is valid whenever we restart the construction. Let $\delta > 0$ be any value satisfying that inequality and set $N := \lceil T/\delta \rceil$. Starting with the initial set K_0 we obtain, by the previous fixed-point argument, a unique solution on $[0, \delta]$. Taking its terminal history at $t = \delta$ as new initial data and repeating the argument yields a unique extension on $[\delta, 2\delta]$, and so on. Uniqueness on overlapping intervals implies that the segments concatenate to a single continuous curve $K : [0, T^*] \to \mathcal{K}(C([-\tau, 0], \mathbb{R}^d))$. Thus a global solution exists and is unique on $[0, T^*]$.

□

This work has established a rigorous mathematical foundation for a novel class of coupled mutational functional differential equations of the form $\dot{x}(t) = F(t, x_t, K_t)$. Theorem 5.3.18 provides the existence and uniqueness theorem for systems where the evolution of each agent depends not only on its own history but also on the collective state of the histories of all co-evolving agents. The proof successfully adapts the classical Picard–Lindelöf argument to a highly abstract setting by constructing a complete metric space of continuous curves of compact sets and demonstrating that a well-defined integral operator is a contraction on this space. This provides the crucial assurance that such complex, interacting systems with memory are well-posed and

can be analyzed with mathematical precision.

The analysis presented here establishes this foundation for systems with a fixed, finite delay $r < \infty$. A natural and important direction for future research is to extend this framework to systems with more complex memory structures. This includes the case of infinite delay, where the full history on $(-\infty, 0]$ is relevant, or state-dependent delays, where the memory horizon can change dynamically. Such extensions would require the theory to be adapted to more sophisticated phase spaces, such as the fading memory spaces used in classical RFDE theory, and would significantly broaden the applicability of this framework to a new range of physical and biological systems.

Chapter 6: Signature Characteristics of Trajectory Prediction

This chapter examines three core innovations that distinguish truly interactive trajectory predictors from standard, benchmark-focused models. First, we confront the Boundary Challenge: real scenes are open worlds where agents enter and leave the field of view, requiring both a precise taxonomy of partially observed trajectories and principled, differentiable loss terms to handle censoring. Next, we introduce a Causal-Pointwise Equivariant framework realized via an encoder–decoder architecture: the encoder compresses rich, variable-length histories into permutation- and time-equivariant latent summaries, and the decoder evolves these summaries continuously to generate socially aware futures. Finally, we tackle Temporal Irregularity head-on by embracing a continuous-time paradigm—both in the encoder's integral convolutions and in the decoder's Neural ODEs—so that our model naturally accommodates asynchronous, irregularly sampled sensor data without resorting to ad hoc imputation or resampling heuristics. Together, these three signature characteristics—boundary awareness, causal-pointwise equivariance, and continuous-time modeling—form a unified, robust foundation for next-generation interactive trajectory prediction.

6.1 The Boundary Challenge

Trajectory prediction benchmarks, while invaluable for comparing models under controlled conditions, rely on a critical simplification: they process and evaluate only on fully-observed trajectories [30, 43]. This common practice of filtering out partial histories—from agents that enter a scene late or exit early—creates a significant disconnect between the benchmark task and the complexities of real-world scenarios. In any practical application, an autonomous system must constantly reason about an open world where agents appear and disappear, and their trajectories are frequently censored by the physical or sensory limits of the observable region.

This simplification has two profound consequences. First, it discards a significant portion of real-world data, limiting the model's exposure to the full spectrum of behaviors. Second, and more importantly, it ignores the critical interactions that occur between fully-observed agents and those with censored (partial) histories. The sudden appearance of a new agent or the departure of an existing one can drastically alter the dynamics of a scene, yet models trained only on complete data are not explicitly optimized to handle these crucial events. This failure to model agent appearances and disappearances is a key barrier to creating robust, real-world predictive systems.

6.1.1 A Complete Taxonomy of Trajectory Censoring

To build a framework that addresses the complexities of real-world data, we must first establish a rigorous classification of how any trajectory sequence can be observed within a finite spatio-temporal window. A trajectory's relationship with the observation window is defined by two independent conditions: its status at the start of the window (the left temporal boundary) and its status at the end (the right temporal boundary).

- **Condition at the Left Boundary (Start of Observation):**

    1. **Spatially Untruncated (Left):** The agent is already present within the observable region $\mathcal{B}$ at the start of the observation window ($t = 0$). The agent's entrance into the scene is therefore *left-censored*.

    2. **Spatially Truncated (Left):** The agent is not present at $t = 0$ and is first observed entering the boundary of $\mathcal{B}$ at some time $\tau_{\text{enter}} > 0$.

- **Condition at the Right Boundary (End of Observation):**

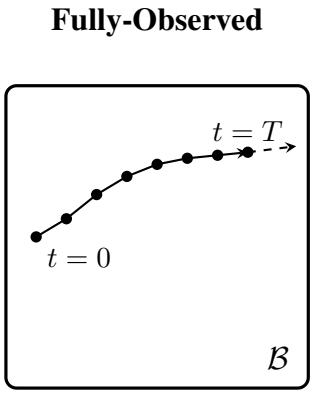
    1. **Spatially Untruncated (Right):** The agent is still present within $\mathcal{B}$ at the final moment of observation ($t = T$). The agent's exit is therefore *right-censored*, with the true exit time $\tau_{\text{exit}} > T$.

    2. **Spatially Truncated (Right):** The agent is observed to leave the boundary $\mathcal{B}$ at some time $\tau_{\text{exit}} \leq T$. This constitutes an *observed exit*.

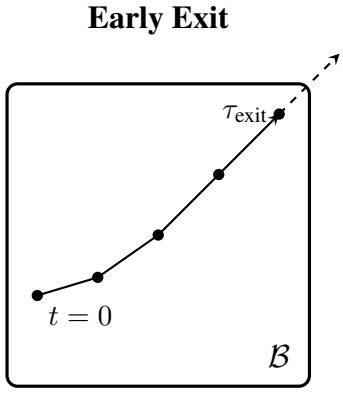
Since these two conditions are independent, their combination yields a complete taxonomy of four mutually exclusive and exhaustive categories that describe any possible observed trajectory segment:

1. **Fully-Observed Trajectory:** *Spatially Untruncated Left* and *Spatially Untruncated Right*. The agent is present for the entire observation period. This is the only category of data used by models that adhere to the simplified standard benchmark protocol.

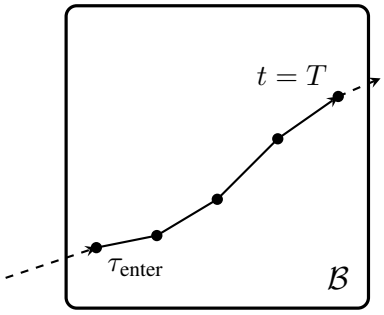
2. **Early Exit Trajectory:** *Spatially Untruncated Left* and *Spatially Truncated Right*. The agent is present at the start but is observed to exit the scene during the observation period.



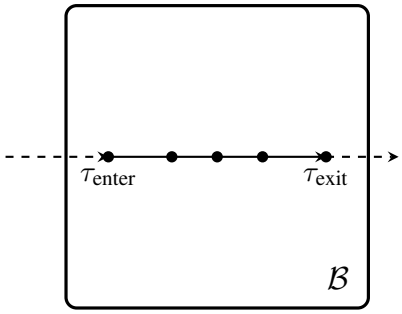
(a) Spatially untruncated on both left and right temporal boundaries.



(b) Spatially untruncated left, but truncated right.



(c) Spatially truncated left, but untruncated right.



(d) Spatially truncated on both left and right temporal boundaries.

Figure 6.1: A complete taxonomy of trajectory censoring within a finite observation window $\mathcal{B}$. Each trajectory must belong to one of these four categories based on its interaction with the temporal boundaries of the observation.

3. **Late Arrival Trajectory:** *Spatially Truncated Left* and *Spatially Untruncated Right*. The agent enters the scene after the observation period has begun and is still present at the end.

4. **Pass-Through Trajectory:** *Spatially Truncated Left* and *Spatially Truncated Right*. The agent both enters and exits the scene entirely within the observation period.

While standard models are often designed to handle only the first, simplest case, a truly robust framework must correctly handle all four. The subsequent sections will detail the architec-

tural and probabilistic solutions our framework employs to address each of these real-world data scenarios.

It is important to note that this taxonomy focuses specifically on censoring and truncation events caused by the finite spatio-temporal boundaries of an observation window. Other real-world phenomena, such as temporary occlusions (e.g., a pedestrian hidden behind a pillar) or sensor failures, can also lead to missing data within an agent's trajectory [1]. While these represent important challenges for robust prediction, they constitute a different class of missing data problems often addressed with interpolation or short-term imputation techniques.

The framework presented in this thesis deliberately concentrates on the geometrically well-defined problem of boundary-induced censoring, as the datasets used for our experiments have been pre-processed to represent continuous tracks within the observable region. By focusing on this specific challenge, we can develop a principled, survival-analysis-inspired approach to model agents entering and exiting the scene—a critical and often overlooked aspect of real-world trajectory forecasting.

6.1.2 General Log-Likelihood

The challenge of handling left-censored trajectories is primarily one of state initialization from an incomplete history, which we address architecturally in Section 6.2. The challenge of handling the future path, however, requires a more rigorous probabilistic formulation. For any given trajectory of observed length T, we must distinguish between an observed exit and a right-censored one

To formalize this, we introduce a binary indicator variable, δ, for each trajectory, where

$\delta = 1$ signifies an observed exit and $\delta = 0$ signifies a right-censored observation. The general likelihood of observing a path prefix $x_0, \ldots, x_T$ and its censoring status δ can be decomposed into a common path probability and a censoring term that depends on δ:

$$L(\theta) = \left(\prod_{t=0}^{T-1} p_\theta(x_{t+1}, x_{t+1} \in \mathcal{B} \mid x_{0:t}) \right) \times \begin{cases} P\big(X_{T+1} \notin \mathcal{B} \mid X_{0:T} = x_{0:T}\big), & \delta = 1, \\ P\big(X_{T+1} \in \mathcal{B} \mid X_{0:T} = x_{0:T}\big), & \delta = 0. \end{cases}$$

To train the model, we work with the log-likelihood, $\mathcal{L}(\theta) = \log L(\theta)$, which takes an elegant, additive form:

$$\mathcal{L}(\theta) = \underbrace{\sum_{t=0}^{T-1} \log p_\theta(x_{t+1}, x_{t+1} \in \mathcal{B} \mid x_{0:t})}_{\text{Path-Matching Term}} + \delta \underbrace{\log P\big(X_{T+1} \notin \mathcal{B} \mid X_{0:T} = x_{0:T}\big)}_{\text{Boundary Exit Term}}$$
$$+ (1 - \delta) \underbrace{\log P\big(X_{T+1} \in \mathcal{B} \mid X_{0:T} = x_{0:T}\big)}_{\text{Boundary Survival Term}}.$$

where the individual terms are defined as:

- Path-Matching Term: $\displaystyle\sum_{t=0}^{T-1} \log p_\theta(x_{t+1}, x_{t+1} \in \mathcal{B} \mid x_{0:t})$
- Boundary Exit Term: $\log P\big(X_{T+1} \notin \mathcal{B} \mid X_{0:T} = x_{0:T}\big)$
- Boundary Survival Term: $\log P\big(X_{T+1} \in \mathcal{B} \mid X_{0:T} = x_{0:T}\big)$

This is the complete and correct log-likelihood for the problem, providing a principled objective for learning from both censored and uncensored trajectory data.

6.1.3 A Differentiable Surrogate for the Boundary Exit Term

The Path-Matching Term Under the common and convenient assumption of an isotropic Gaussian noise model for the trajectory, the first term of the log-likelihood corresponds directly to the standard L_2 reconstruction error used during training. This term forces the model to learn the internal dynamics of movement.

The Challenge of the Boundary Terms The boundary exit and survival terms, while theoretically sound, present a fundamental implementation challenge. For any given stochastic trajectory generated by the model, the predicted exit time $\hat{\tau}_{\text{exit}}$ is a deterministic function of that path. This deterministic mapping, combined with the fact that exit times in benchmark datasets are recorded as discrete integer multiples of a frame rate, causes the gradient of the loss with respect to the model parameters to be zero almost everywhere. A small change in model weights is highly unlikely to cause a predicted path to cross the boundary at a different discrete time step.

The Differentiable Surrogates To overcome this, we introduce carefully designed, differentiable surrogates that are handled by our loss function.

1. **For Observed Exits** ($\delta = 1$): The anti-lingering penalty serves as the differentiable surrogate for the boundary exit term. Its role is to provide a smooth, non-zero gradient that encourages the model to predict a timely exit, fulfilling the objective of maximizing the exit probability. It has a dual justification:

   (a) **As a Spatial Surrogate:** It acts as a continuous proxy for the discrete exit event. Instead of a binary in/out signal, it provides a gradient that pushes lingering trajectories

towards the boundary.

(b) **As a Geometric Lower Bound:** For any time $t > \tau_{\text{exit}}$, the ground-truth position x_t is outside $\mathcal{B}$. If the model's prediction $\hat{x}_t$ remains inside $\mathcal{B}$, then the true L_2 error is bounded below by the distance from the prediction to the boundary:

$$\|\hat{x}_t - x_t\| \;\geq\; \operatorname{dist}(\hat{x}_t, \partial\mathcal{B})\,.$$

Penalizing this distance is therefore a principled method of minimizing a lower bound on the true, unobservable error.

2. **For Right-Censored Data** ($\delta = 0$): The boundary survival term is handled implicitly. The optimization is driven solely by the L_2 path-matching term, which forces the predicted path to stay close to the ground-truth path inside the boundary. This implicitly encourages the model to predict trajectories that survive within the boundary, thus maximizing the survival probability without needing a separate, explicit loss term.

   As a result, the true squared L_2 loss

$$\sum_{t=0}^{T} \|\hat{x}_t - x_t\|^2$$

is bounded below by

$$\sum_{t:\, x_t \in \mathcal{B}} \|\hat{x}_t - x_t\|^2 \;+ \sum_{t:\, x_t \notin \mathcal{B},\, \hat{x}_t \in \mathcal{B}} \operatorname{dist}(\hat{x}_t, \partial\mathcal{B})^2,$$

since whenever $x_t \notin \mathcal{B}$ and $\hat{x}_t$ lingers inside $\mathcal{B}$, we have $\|\hat{x}_t - x_t\| \geq \operatorname{dist}(\hat{x}_t, \partial\mathcal{B})$.

6.2 The Encoder-Decoder Architecture and Causal-Pointwise Equivariance

6.2.1 The Encoder-Decoder Architecture

The most complete and theoretically satisfying way to model systems with memory is through the framework of Retarded Functional Differential Equations (RFDEs). In this paradigm, the state of a system at any given time is not merely its current position, but the entire continuous history function leading up to that moment. The future evolution is then described by a functional that operates on this infinite-dimensional history space.

While theoretically powerful, a direct neural implementation of this concept presents a formidable practical challenge. In any real-world application, the continuous history must be discretized into a sequence of observations. A naive but theoretically complete model would then need to consider the influence of every past point on the current prediction. This would require modeling all pairwise temporal interactions within the history, a process analogous to the self-attention mechanism in Transformer [19] architectures. Such an approach is subject to a computational complexity that scales quadratically with the length of the observation history, rendering it prohibitively expensive for modeling long-term dependencies.

To overcome this challenge while still capturing the rich influence of the past, we adopt the encoder-decoder architecture as a principled and efficient solution. This architecture decouples the complex task of history processing from the task of future prediction.

The encoder's role is to serve as a learned summarization function. It is designed to take the high-dimensional, variable-length history of an agent—including its interactions with its environment and other agents—and compress it into a compact, finite-dimensional latent vector.

This latent vector serves as a prior condition for the decoder, providing a rich summary of the past. The nature of this summary can be conceptualized in two ways:

1. **Deterministic Encoder:** In the simplest case, the encoder can be deterministic, mapping a given history to a single, fixed latent vector. This vector represents the best guess or a rich feature summary of the agent's state and intent. This approach is computationally efficient but is fundamentally limited to producing only a single future prediction, failing to capture the inherent multi-modality of real-world behavior.

2. **Probabilistic Encoder:** To address the stochastic nature of human movement, a more sophisticated approach is to design a probabilistic encoder. In this paradigm, the encoder's output is not a single vector, but rather the parameters of a probability distribution (e.g., the mean and covariance of a Gaussian) over the latent space. This distribution acts as a true *probabilistic prior* for the prediction, capturing not just a summary of the past but also the *uncertainty* associated with that summary.

This probabilistic approach allows the model to directly connect the quality of the input history to the confidence of its forecast. For example, a short and information-poor history—as seen in a Late Arrival or Pass-Through trajectory from our taxonomy—should lead the encoder to produce a prior with high variance, signaling high uncertainty and a wide range of plausible futures. Conversely, a long and information-rich history—as in a Fully-Observed trajectory—should result in a prior with low variance, reflecting higher confidence in a more constrained set of future possibilities. The encoder, therefore, does not just compress what happened; it learns to quantify how much to trust that information.

The decoder then operates on this summary. In the deterministic case, it takes the single

latent vector as an initial condition. In the more general probabilistic case, it takes a sample from the latent distribution as an initial condition to generate a single, plausible future trajectory. By sampling multiple times, the probabilistic model can generate a diverse set of predictions whose spread is naturally governed by the uncertainty encoded in the prior.

By structuring the model in this way, we avoid the quadratic cost of invoking the full history at every predictive step. The encoder performs the computationally intensive task of history summarization once, producing an information-rich latent vector that equips the decoder with the necessary context to generate a plausible and accurate future path.

6.2.2 Implementing Causal-Pointwise Equivariance in the Encoder

The encoder's primary role is to transform a rich, high-dimensional history of agent interactions into a compact latent vector that can effectively initialize the decoder. For this process to be generalizable and physically meaningful, the encoder's architecture must be designed to respect the fundamental symmetries of the problem.

Data Representation and the Challenge of Ragged Histories Before detailing the encoder's specific mechanisms, it is essential to first describe the structure of the raw data it must process. A batch of data from a scene can be conceptualized as a three-dimensional tensor of shape (`num_agents, num_timesteps, num_features`), where the features are the (x, y) coordinates.

However, due to the various forms of censoring defined in Section 6.1.1, this tensor is inherently ragged. As illustrated in the data sample from the Zara03 dataset [98] (Table 6.1), many entries in the tensor are `NaN` because agents enter the scene late, exit early, or both. The presence

of `NaN` values illustrates the practical challenge of handling the four types of trajectory censoring. For example, Agents 0, 1, 7, and 8 represent the fully-observed case. Agents 2-6 and 13 demonstrate early exits (spatially truncated right), while Agents 9-12 are late arrivals (spatially truncated left). This raggedness presents a significant practical challenge for modern deep learning frameworks, which are optimized for dense, uniform tensor operations. The primary task of the encoder's initial stages is to handle this ragged structure in a principled and efficient manner.

The Sequential Processing of Symmetries The encoder must adhere to the principle of causal-pointwise equivariance, which is composed of two distinct symmetries: permutation equivariance for the spatial arrangement of agents, and causal temporal equivariance for the evolution of their histories. A profound result from group theory dictates how these symmetries can be implemented. The symmetry group of the problem, $\mathbb{R} \times \mathfrak{S}_k$, has a unique internal direct product decomposition. The architectural consequence of this uniqueness is that there are no exotic or clever modular architectures that can circumvent the problem's fundamental structure by finding a new basis where temporal evolution and agent permutations become intertwined in a simpler way.

Therefore, any principled architecture that fully respects the problem's symmetries is forced to process the temporal ($\mathbb{R}$) and spatial/permutational ($\mathfrak{S}_k$) information sequentially. This constraint gives rise to two canonical architectural pathways:

1. Time-First, Space-Second (Shift $\to$ Set): In this approach, a temporal summary is first generated for each agent's history independently. Subsequently, the spatial interactions between these complete historical summaries are processed.

2. Space-First, Time-Second (Set $\to$ Shift): In this alternative, the spatial interactions between

Agent	0	1	2	3	4	5	6	7	8	9
0	8.8 6.8	8.3 6.8	7.9 6.8	7.4 6.8	7.0 6.8	6.5 6.8	6.1 6.8	5.6 6.8	5.1 6.8	4.6 6.7
1	9.1 7.2	8.6 7.2	8.1 7.3	7.7 7.3	7.2 7.3	6.7 7.3	6.2 7.3	5.8 7.3	5.3 7.3	4.8 7.3
2	13.8 5.0	14.2 4.8	14.7 4.7	15.1 4.6	NaN	NaN	NaN	NaN	NaN	NaN
3	2.1 8.5	1.5 8.6	1.0 8.8	0.7 9.1	0.4 9.4	0.0 9.6	NaN	NaN	NaN	NaN
4	1.0 6.8	0.5 6.7	0.1 6.7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	1.2 6.1	0.8 6.0	0.3 6.0	-0.2 5.9	NaN	NaN	NaN	NaN	NaN	NaN
6	13.6 6.6	14.3 6.6	14.9 6.6	15.6 6.6	NaN	NaN	NaN	NaN	NaN	NaN
7	11.6 4.4	11.9 4.4	12.3 4.4	12.8 4.4	13.2 4.4	13.6 4.4	14.0 4.4	14.5 4.5	14.9 4.5	15.4 4.5
8	11.5 5.2	11.9 5.2	12.3 5.2	12.8 5.2	13.1 5.2	13.5 5.3	13.9 5.3	14.3 5.4	14.7 5.4	15.0 5.4
9	NaN	15.0 6.0	14.5 6.1	14.0 6.2	13.5 6.3	13.0 6.3	12.5 6.4	12.0 6.4	11.5 6.5	11.0 6.6
10	NaN	15.1 5.4	14.6 5.4	14.1 5.5	13.6 5.5	13.1 5.5	12.6 5.4	12.0 5.4	11.5 5.4	11.0 5.5
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.3 4.9	0.9 5.1
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.1 4.5	0.4 4.5	1.0 4.5
13	0.4 4.5	1.0 4.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 6.1: A sample of the raw, ragged trajectory data tensor for the 10 to 19 time steps of 14 agents from the Zara03 dataset. Each cell contains the (x, y) coordinates.

all agents are computed at each discrete time step. The encoder then processes the temporal evolution of these interaction-aware features to form the final summary.

Furthermore, Conjecture 5.1.5 suggests that a two-layer convolutional network is necessary to approximate any well-behaved equivariant operator. This provides a strong theoretical motivation for a two-stage encoder architecture (e.g., a Shift $\rightarrow$ Set $\rightarrow$ Shift process), which is more expressive and appealing from a universality perspective, despite the increased computational cost.

The Adaptive Causal Kernel for Left-Censored Histories The core challenge of temporal processing is handling the ragged, variable-length histories that result from the different forms of censoring in our taxonomy. Given the distinct statistical nature of left-censored versus left-truncated trajectories, one might argue for designing separate temporal kernels for each case. However, creating and justifying such specialized models is complex. A more elegant and parameter-efficient solution is to create a single, adaptive causal kernel that learns to handle these distinctions implicitly.

To achieve this, we define the adaptive kernel as a neural network that generates a temporal matrix-valued convolution kernel K as a function of two inputs: the total observed length of a trajectory, T_{obs}, and the time lag, τ. That is, $K = A_\theta(T_{\text{obs}}, \tau)$. By explicitly conditioning on the observed duration, this single, continuous function can produce a meaningful historical summary for any partial trajectory, avoiding the need for multiple, specialized kernels.

This adaptive mechanism is also crucial for modeling uncertainty. The network learns to map summaries from short, information-poor histories to high-variance latent priors, while mapping summaries from long, information-rich histories to low-variance priors. This directly

connects data quality to forecast confidence.

Finally, to ensure robustness even for very short histories, our framework employs two types of convolutions in parallel. For a trajectory $x(t)$, observed on the interval $[t_0, t_*]$, the standard integral convolution is given by:

$$\int_{t_0}^{t_*} A_\theta(t_* - t_0, t_* - t)x(t)\ dt.$$

While this provides a universal summary, its output can approach zero for a short duration $t_* - t_0$, potentially losing information. To counteract this, a maxitive convolution is used simultaneously to capture the most salient feature from the history, regardless of its length:

$$\max_{t_0 \le t \le t_*} M_\theta(t_* - t_0, t_* - t)x(t).$$

The final historical summary is a concatenation of both results, providing a representation that is robust to trajectories of any observed length.

A Practical Technique for Parallel Processing: Temporal Conjugation A fundamental challenge in multi-agent modeling is performing meaningful, time-synchronous spatial interactions on a batch of ragged, temporally misaligned trajectories. To solve this, we introduce a data processing pipeline we term *temporal conjugation*, inspired by the concept of conjugation ($g^{-1}hg$) in group theory. The technique transforms the data into a computationally convenient frame for processing, and then transforms it back to its original temporal frame to ensure spatio-temporal consistency. This is achieved through a three-step process:

1. **Detect and Align** (g): First, the true start index and observed length of each agent's partial trajectory are detected. Then, an Align operation transforms each history into a temporary, left-aligned computational frame, creating a dense tensor suitable for vectorized operations.

2. **Process** (h): The primary temporal processing, such as the first adaptive causal convolution, is applied in a highly parallel manner to this aligned batch.

3. **Un-align** (g^{-1}): The resulting sequence of historical summaries is then mapped back to its original, correct temporal positions for each agent. This Un-align step is crucial, as it ensures that the inputs to the subsequent spatial interaction modules are temporally consistent across all agents in the scene.

This framework makes it possible to apply various sequence models to ragged data. While traditional recurrent architectures like RNNs can be parallelized across the batch of agents using this technique, their processing remains inherently serial along the time axis. In contrast, our choice of an adaptive causal kernel fully leverages the parallel potential of this technique. The entire aligned sequence for each agent can be processed in a single, parallel operation, analogous to the causal masking used during the training of Transformer decoders [19].

This advantage persists at inference time. While an RNN must still iterate step-by-step through the known history, our causal kernel can process the entire available history in one pass. This makes the convolutional approach a more computationally efficient choice for both training and inference on ragged, real-world data.

The Latent Vector as a Projection of the Jet Fiber The output of this entire encoding process is a finite-dimensional latent vector for each agent at each point in time. This vector can be under-

stood as a practical and learned approximation of a richer mathematical object. Conceptually, we frame this latent vector as being inspired by the jet fiber.

While a true jet fiber would contain an infinite sequence of higher-order derivatives (velocity, acceleration, jerk, etc.), our encoder's latent vector can be seen as a learned projection from this infinite-dimensional space into a compressed, finite-dimensional representation. It is trained to implicitly capture the most salient higher-order dynamic information from the agent's own history and its interactions with others. This latent vector is then stacked on top of the agent's current position to form the complete state that initializes the decoder, providing it with a much richer picture of the agent's state than position alone.

6.2.3 Implementing Causal-Pointwise Equivariance in the Decoder

While the encoder is responsible for summarizing the past, the decoder's role is to generate a plausible future. For a model to produce a continuous, time-varying trajectory, rather than a fixed-size set of future points in a single pass, a recurrent or evolving dynamic is unavoidable. This stands in contrast to architectures like Social-STGCNN [38], which are designed for the specific benchmark task of observing 8 frames to produce a 12-frame future simultaneously. Such single-pass models are efficient for a fixed prediction horizon but lack the architectural flexibility to generate continuous paths or adapt to events that unfold over time.

Our framework provides this necessary recurrent structure through the use of a Neural ODE [13]. The decoder evolves an initial state—provided by the encoder—forward in time. Equivariance in this generative, recurrent process is realized through how the decoder handles the social context during this evolution. We explore two distinct models of contextual equivariance,

each representing a different philosophy of how past information should influence the future.

1. **Static-Context Model:** In this architecture, the decoder is initialized with a latent vector that summarizes the entire historical context up to the point of prediction. This context, which is permutation-equivariant with respect to the agents in the past, remains fixed throughout the entire decoding process. The agent effectively plans its future based on its understanding of the past and executes that plan without incorporating new information from its neighbors' evolving predicted paths. The equivariance is respected at the start of the prediction, but the social context does not co-evolve with the agent's state.

2. **Dynamic-Context Model:** In this more sophisticated architecture, the decoder continuously re-evaluates the social context at every intermediate step of the ODE solver. The interactions between all agents are re-computed based on their current predicted states at that future moment in time. This creates a more powerful, dynamically equivariant model where agents are continuously reacting to the anticipated movements of others throughout the prediction horizon. The social context is not a fixed prior condition, but a dynamic input that co-evolves with the state of the entire system.

A key advantage of the dynamic-context model is its ability to adapt to new information that becomes available during the prediction window, such as the appearance of a previously unobserved agent (a late arrival from our taxonomy). Because the dynamic-context model re-evaluates the set of all agents at every step, it can immediately incorporate the new agent into its interaction calculations and adjust the trajectories of existing agents in response. The static-context model, having fixed its plan based only on the past, is architecturally incapable of such real-time adaptation.

6.3 The Advantage of Continuous-Time Modeling: Handling Irregular Data

Beyond the challenges of censoring and interaction, real-world trajectory data presents another fundamental difficulty: temporal irregularity. While many benchmark datasets are pre-processed to provide observations at uniform time intervals, the raw data from physical sensors is often not captured by a perfect metronome [99]. Sensor lag, packet loss, or the inherent asynchronicity of different sensor systems in a fusion pipeline can lead to non-uniform time intervals between observations. This section details how our framework's continuous-time philosophy provides a principled and robust solution to this challenge.

A fundamental weakness of standard recurrent architectures, such as RNNs and LSTMs, is their reliance on a fixed, discrete time step, Δt. These models are architecturally designed to process sequences where observations occur at regular, uniform intervals, making them inherently ill-equipped to handle irregularly sampled data.

When faced with non-uniform time intervals, discrete models are forced to either ignore the precise timing information and treat the data as a simple ordered sequence, or to use heuristics to fit the data into a rigid time grid. Both approaches are suboptimal, as they either discard valuable information contained in the timestamps or risk distorting the true underlying dynamics of the system by imposing an artificial temporal structure.

In contrast, the continuous-time paradigm offers a more natural and robust solution to this problem. Instead of modeling the discrete steps of a sequence, this approach aims to learn the underlying differential equations that govern the system's evolution. By modeling the dynamics themselves, the framework is no longer tied to the specific, and potentially irregular, time points at which the data was sampled. This allows the model to learn the intrinsic properties of an

agent's motion, not the artifacts of a particular sampling rate.

6.3.1 Neural Implementation of Continuous-Time Processing

Our framework implements this continuous-time philosophy in both the encoder and decoder, granting the architectural flexibility needed to handle temporal irregularities.

Encoder: Causal Integral Convolution The history encoder is built on inherently continuous-time operators designed to process the continuous trajectory segments described in §4.2.6. We assume that within any single observed segment (e.g., $\gamma : [a, b] \to \mathbb{R}^d$), the time between samples is small enough that the path can be faithfully represented, a task handled naturally by the numerical integration within our convolutions. This approach avoids large-scale imputation for significant occlusions or sensor failures, which is ill-suited for scenarios where agent behavior is fundamentally uncertain (e.g., on a highway, a one-second gap is too large to be faithfully interpolated). As detailed in Section 6.2, our adaptive causal kernel processes these continuous segments. The framework uses both an integral convolution to accurately accumulate information and a maxitive convolution to robustly capture salient features from even very short histories. This dual approach enables the encoder to generate a meaningful summary from any observed trajectory segment, regardless of its length or internal sampling density.

Handling Interaction under Fully Asynchronous Sampling Despite the flexibility of adaptive causal kernels, modeling social interactions requires aligning all agents' states to a common timestamp set. Each agent i is observed at its own irregular timestamps $\{t_j^i\}$. To compute interaction features:

1. **Space-First** (Set→Shift): This approach first synchronizes the data by interpolating each agent's trajectory onto the union of all timestamps $\mathcal{T} = \bigcup_i \{t_j^i\}$ (or a chosen dense grid). Interaction features are then computed on these synchronized observations. However, this approach is often computationally inefficient and can introduce artifacts due to its heavy reliance on interpolation.

2. **Time-First** (Shift→Set): Reconstruct each agent's latent trajectory as a smooth function $z_i(t)$ via the continuous-time encoder, then evaluate $z_i(t)$ at $\mathcal{T}$ to obtain synchronized latent states for interaction computations.

This ensures that even timestamps with sparse observations (e.g., a single observed agent) are properly considered. By treating missing observations as interpolable gaps in a continuous trajectory rather than agent absences, we maintain fidelity to both asynchronous sampling and underlying continuous dynamics before applying permutation-equivariant interaction operations.

While our continuous-time operators can handle irregular sampling for individual agents, modeling social interactions presents a further challenge in the most general case of fully asynchronous sampling, where each agent is observed at its own unique set of timestamps. A theoretical approach would be to create a master time grid from the union of all timestamps and interpolate each agent's trajectory onto it. However, this strategy is often impractical, as it can be computationally expensive and introduce significant interpolation error if the sampling rates vary widely between agents.

Even if this master grid were pruned to a sparser, more computationally tractable set of timestamps, the resulting common time grid would still be irregular. Our Temporal Conjugation pipeline is robust enough to handle such a scenario; once the irregular timestamps are sorted, the

same align-process-unalign logic can be applied.

However, since the fully asynchronous case is rare in many common applications (e.g., video surveillance) where data is captured on a common clock, our framework focuses on the more prevalent challenge of raggedness: agents are observed on the same underlying grid, but only for partial, misaligned segments. The Temporal Conjugation technique is therefore presented as our practical and efficient solution for this common problem.

Decoder: Neural Ordinary Differential Equations The decoder leverages a Neural ODE architecture, a fundamentally continuous-time generative model [13]. The numerical solver evolves the latent state continuously from an initial time t_0 to an endpoint t_1, and can be evaluated at arbitrary specified timestamps. This continuous-time evolution provides two powerful capabilities not present in discrete recurrent models:

- **Interpolation:** The decoder generates smooth, physically plausible trajectories between irregularly spaced observations, inherently filling temporal gaps in a dynamics-aware manner.
- **Prediction:** It seamlessly predicts future states at arbitrary timestamps, independent of the input data's original sampling regularity or density.

Moreover, the continuous evolution of the decoder mirrors the continuous-time summary from the encoder, providing end-to-end coherence. The choice of numerical solvers [100] (e.g., fixed-step versus adaptive) further balances trade-offs between computational efficiency and solution accuracy, adapting naturally to practical application constraints.

Overall, the continuous-time approach robustly accommodates irregular temporal sampling

and asynchronous multi-agent scenarios, offering significant practical advantages over conventional discrete-time methods.

Chapter 7: Neural Architecture & Implementation

The preceding chapters established a theoretical foundation for trajectory prediction built on three signature characteristics: boundary awareness, causal-pointwise equivariance, and continuous-time dynamics. This chapter presents the concrete realization of this framework through a series of end-to-end neural architectures. We begin by justifying our choice of the JAX ecosystem, whose features for composable vectorization and high-performance compilation were instrumental in implementing our models efficiently. We then detail the core components of our architectures: the temporal conjugation pipeline for handling ragged real-world data; the progression of causal-pointwise equivariant encoders—from simpler canonical pathways to a final, multi-stage design; the static- and dynamic- context decoders built on Neural ODEs; and finally, the probabilistic VAE framework and the specific loss functions used to train each model. Together, these components translate our theoretical principles into a practical, high-performance forecasting system.

7.1 The Choice of the JAX Ecosystem

The implementation of our theoretical framework requires a software environment that is both highly performant and expressive enough to handle complex, nested operations on ragged data. For readers less familiar with our chosen tools, **JAX** [14] is a high-performance numerical

computing library that combines a NumPy-like application programming interface (API) with a functional programming paradigm and composable function transformations. One of its core features is *Just-In-Time* (**JIT**) compilation, where a function is traced and compiled into a highly optimized computational kernel using the *Accelerated Linear Algebra* (**XLA**) compiler [101]. This process fuses operations and generates efficient machine code tailored for accelerators like graphics processing units (GPUs) and tensor processing units (TPUs).

While frameworks like PyTorch and TensorFlow are common, we selected the JAX ecosystem for its unique capabilities in functional programming, automatic differentiation, and compilation, which are particularly well-suited to the challenges of our model. This section details the motivation for this choice and the specific libraries used.

7.1.1 Key Advantages for Spatio-Temporal Modeling

Our decision to use JAX is grounded in three key advantages that were critical for implementing our complex, multi-stage architecture efficiently.

Composable Vectorization with `vmap` A core challenge in our work is the parallel processing of interactions between agents with ragged, misaligned histories. While other frameworks offer some vectorization, JAX's `vmap` is a fully composable function transformation. This allows for the deep nesting of `vmaps`, which is essential for our architecture. For example, our final model requires an outer `vmap` to iterate over the batch, which contains another `vmap` for each agent, which in turn contains another `vmap` for each initial time step. Within this structure, further vmaps are used to implement the actual interaction (`pairwise_interactions`) and convolution (`conv_list`) logic. JAX handles this deep nesting of transformations elegantly,

which simplifies bookkeeping and avoids the complex manual flattening that would be required in other frameworks.

This composability has profound implications for performance, particularly memory. To compute pairwise interactions for `N` agents via broadcasting, other frameworks would require materializing an intermediate tensor of shape `(N, N, ...)` leading to an $O(\texttt{N}^2)$ memory footprint. With JAX's nested `vmap`, this large tensor is never created, resulting in a more efficient $O(\texttt{N})$ memory footprint, which is critical in memory-limited GPU environments.

High-Performance Differential Equation Solvers with Diffrax A central component of our decoder is a Neural ODE. **Diffrax** [45] is a JAX-native library for numerical differential equation solving that offers a wide range of high-order and adaptive solvers that are superior to those available for other deep learning frameworks. Its seamless integration with JAX's `grad` and `jit` transformations was essential for building and training our continuous-time decoders.

JIT Compilation for Speed JAX's primary performance advantage comes from its Just-In-Time (JIT) compiler, which fuses operations and compiles Python functions into highly optimized XLA kernels using the `@jit` decorator. This provides significant speed improvements during training and inference and motivates the careful design of our JIT-compatible temporal conjugation pipeline.

7.1.2 The Library Stack

Our implementation is built on a small, powerful stack of JAX-native libraries:

- **Equinox** [45]: Provides a simple and effective way to organize model parameters and state

into PyTorch-like modules (`eqx.Module`) while remaining fully compatible with JAX's functional paradigm.

- **Diffrax** [45]: Used for all Neural ODE integration in our decoders.
- **Optax** [102]: Used for all gradient processing and optimization, providing a flexible library of standard optimizers like Adam [103] and AdamW [104].

7.2 The Temporal Conjugation Pipeline

A primary challenge in processing real-world trajectory data is that our models require dense, uniform tensors for efficient JIT-compilation, but the data itself is often ragged. To bridge this gap, we developed a JIT-compatible pipeline of temporal conjugation in §6.2.2. The pipeline transforms the data into a computationally convenient frame, applies the primary processing, and then transforms the result back to the original temporal frame to ensure spatio-temporal consistency.

The pipeline consists of the following three stages:

1. **Detect and Align (g):** This stage identifies the start index and valid length of each partial trajectory and creates a dense, left-aligned view of the data. To illustrate, consider a partial trajectory

$$X = [\texttt{NaN},\ \texttt{NaN},\ 1,\ 2,\ 3,\ 4,\ \texttt{NaN},\ \texttt{NaN},\ \texttt{NaN}].$$

First, the Detect step finds $\texttt{start_ind} = 2$ and $\texttt{time_length} = 4$. Next, the Align step constructs an index array, `left_aligned_inds`, to produce a dense array where the last valid observation is used as right-padding:

```
left_aligned_inds = (
    start_inds[..., None]
    + jnp.cumsum(
        jnp.arange(T_max) <= (time_lengths[..., None] - 1),
        axis=-1
    )
)
```

For our example, this generates the index map $\big[2, 3, 4, 5, 5, 5, 5, 5, 5\big]$. A vectorized `gather` operation then uses this map to select

$$\big[X[2], X[3], X[4], X[5], X[5], \dots\big]$$

from the original trajectory, producing the left-aligned array

$$\big[1, 2, 3, 4, 4, 4, 4, 4, 4\big].$$

2. **Process (h):** The primary temporal operation, such as the first adaptive causal convolution, is then applied in a highly parallel manner to the aligned batch produced in the previous step.

3. **Un-align (g^{-1}):** Finally, the processed features are mapped back to their original, correct temporal positions. Continuing the example, if `F_aligned` are the processed features for the aligned data, the goal is to produce an un-aligned array `F_unaligned` that is padded

on both sides, like $\big[f(1), f(1), f(1), f(2), f(3), f(4), f(4), f(4), f(4)\big]$. This is achieved by creating an inverse index map, `aligned_inds`, by clamping indices to the valid range for each trajectory:

```
aligned_inds = jnp.minimum(
    jnp.maximum(
        0,
        jnp.arange(T_max)-start_inds[..., None]
    ),
   time_lengths[..., None]-1,
)
```

For our example, this operation clamps the indices to the valid local range of $[0, 3]$, producing the final index map $\big[0, 0, 0, 1, 2, 3, 3, 3, 3\big]$. A final `gather` operation uses this map to produce the un-aligned, padded, and temporally consistent feature tensor.

7.3 Encoder Architectures

The encoder is the core component responsible for processing historical data in a manner that respects the fundamental symmetries of the multi-agent prediction problem. As established by the unique decomposition of the $\mathbb{R} \times \mathfrak{S}_n$ symmetry group, any principled architecture must process the temporal and spatial dimensions sequentially. This section details the three encoder architectures we developed. We first present two canonical pathways designed for standard benchmarks, which operate on relative coordinates. We then introduce our final, more expressive architecture, which uses absolute coordinates—a more natural choice for our target robotics

applications and for implementing boundary-awareness.

7.3.1 Canonical Pathways for Benchmark Models

For benchmark comparisons on the fixed 8/12 prediction horizon [30, 35, 36, 38, 39, 43], we implemented two architectures that use agent-centric (relative) coordinates. Given this simpler task where the observation duration is always fixed, these models employ a more efficient, non-adaptive single causal convolution. The kernel for this convolution is only a function of the time lag, as the observed duration is constant. This streamlined design helps to offset the computational cost incurred by creating `N` different agent-centric views for a scene with `N` agents.

The interaction modules in these models are a direct implementation of the global summary model ($\gamma_{\text{MLP}}\big(x,\ \max_{y\in K} h_{\text{MLP}}(y)\big)$), the individualized interaction model ($\gamma_{\text{MLP}}\big(\max_{y\in K} \phi_{\text{MLP}}(x,y)\big)$), or the combination of the two from Theorem 3.5.7. The global summary function h_{MLP} is implemented by `sumr_model` and the pairwise interaction function ϕ_{MLP} is implemented by `intr_model`, and the efficient, parallel computation of the interactions for each agent is achieved by applying `jax.vmap` to this function.

The interaction modules now include three dummy bias terms—one as a pseudo-agent for self-interaction (`H_bias`), one as an extra max-pooling bias (`I_bias`), and one as a global summary bias (`S_bias`)—so that every pooling operation can fall back on a learned constant if no real neighbor provides a stronger signal. Concretely, for each agent state x with neighbors K we compute

$$\hat{\phi}(x) \;=\; \max\Big\{ \max_{y\in K} \phi_{\text{MLP}}(x,y),\ \phi_{\text{MLP}}(x,\texttt{H_bias}),\ \texttt{I_bias}\Big\},$$

and then feed $\hat{\phi}(x)$ through γ_{MLP}. Similarly, the global summary at time t is

$$\hat{h} \;=\; \max\Big\{ \max_{x \in K} h_{\text{MLP}}(x),\; \texttt{S_bias} \Big\}.$$

By learning these biases end-to-end, the model can gracefully handle sparse interactions without resorting to ad hoc fallbacks.

The Agent→Time (Set→Shift) Architecture The VAEST model follows a spatial-first processing pathway, as illustrated in Figure 7.1. At each of the 8 observed time steps, its interaction module computes social context from the relative agent coordinates. A subsequent skip connection combines these features with the agent's own motion before a final causal convolution summarizes the sequence.

The Time→Agent (Shift→Set) Architecture The VAETS model follows a temporal-first processing pathway, as shown in Figure 7.2. A causal convolution is first applied to each agent's relative trajectory to produce a set of hidden summary vectors. The Interaction Module then computes social context by applying the individualized interaction model to this set of abstract hidden vectors.

7.3.2 The Time→Agent→Time Encoder for Partial Trajectories and Multi-Horizon Prediction

For our final encoder, which is designed to handle partial trajectories across multi-horizons and to be boundary-aware, we use absolute coordinates. This is a more natural fit for robotics

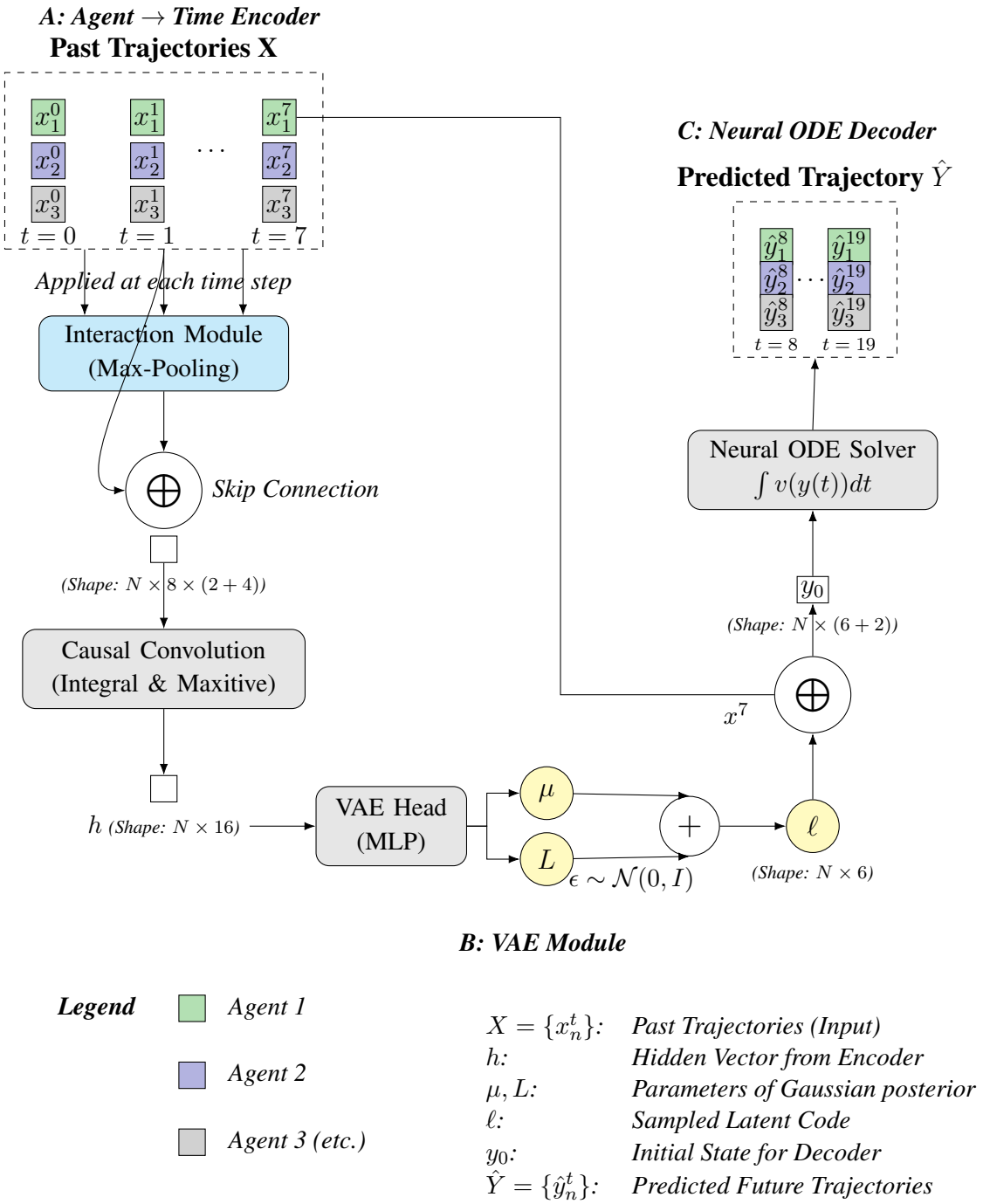


Figure 7.1: A complete diagram of the VAEST (Agent → Time) architecture, showing the encoder, VAE module, and Neural ODE decoder.

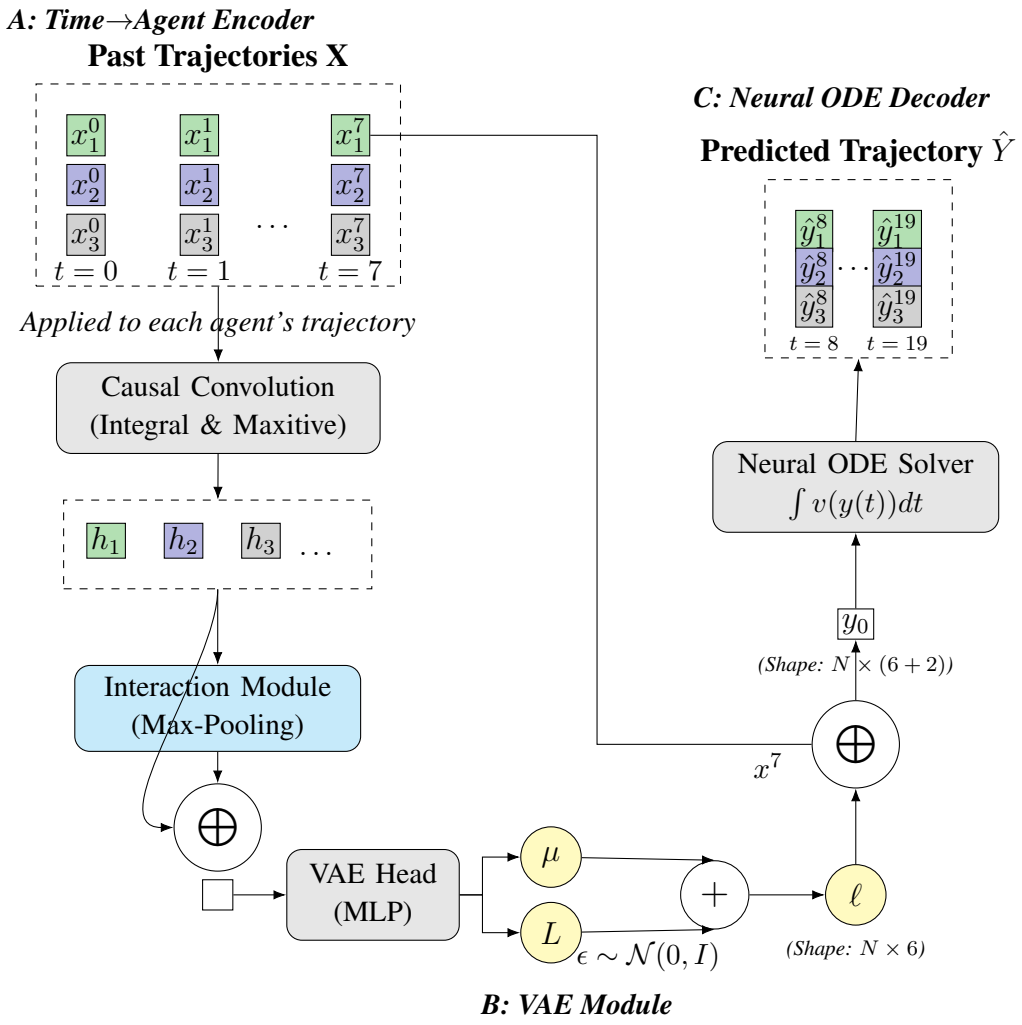


Figure 7.2: A complete diagram of the VAETS (Time $\rightarrow$ Agent) architecture, showing the encoder, VAE module, and Neural ODE decoder.

applications, where an agent typically operates within a known, confined environment with fixed boundaries. Using a shared, absolute frame makes reasoning about these boundaries direct and elegant.

Our final encoder uses a more expressive, three-stage Time→Agent→Time encoder, shown in Figures 7.3 and 7.4. This encoder architecture is inspired by our conjectured Conjecture 5.1.5 with the three learned dummy biases, `H_bias`, `S_bias`, and `I_bias`. To enable multi-horizon prediction, this multi-stage process is applied across a range of potential observation endpoints to produce a sequence of initial states $\{y^{t_0}\}$. In our experiments, t_0 ranges from 5 to 13. The encoder uses a multi-stage process with repeated, highly-parallelizable adaptive causal convolutions. The implementation of these convolutions is made efficient through the use of a `vmap`-based `conv_list` function. A detailed, one-to-one correspondence between the components of the universality theorem and the code modules for this architecture is provided in Appendix B.

7.4 Decoder Architectures: Continuous-Time Trajectory Generation

The decoder is responsible for taking the compact, probabilistic summary from the encoder and generating a continuous future trajectory. To achieve this in a way that is robust to irregular time steps and allows for prediction at any future point, we use a Neural ODE [13]. The decoder integrates an initial state vector forward in time using a learned neural vector field. We present two versions of this decoder: a simpler static-context model and a more advanced dynamic-context model.

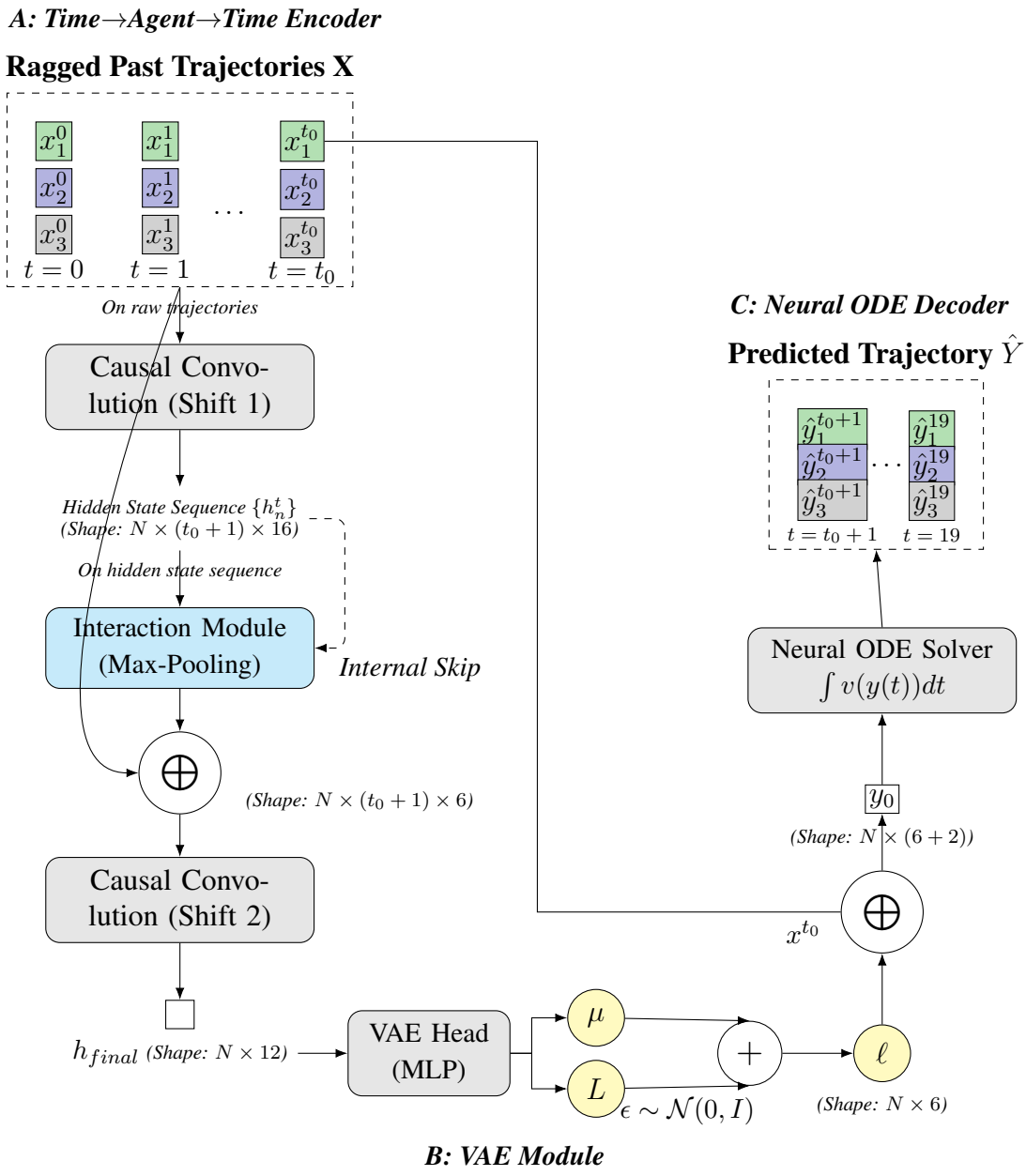


Figure 7.3: A complete diagram of the Time→Agent→Time Architecture with a Static-Context Decoder. This model handles partial trajectories and is evaluated across multiple prediction horizons. The decoder's vector field $v(y(t))$ evolves each agent based on an initial context that remains fixed during prediction.

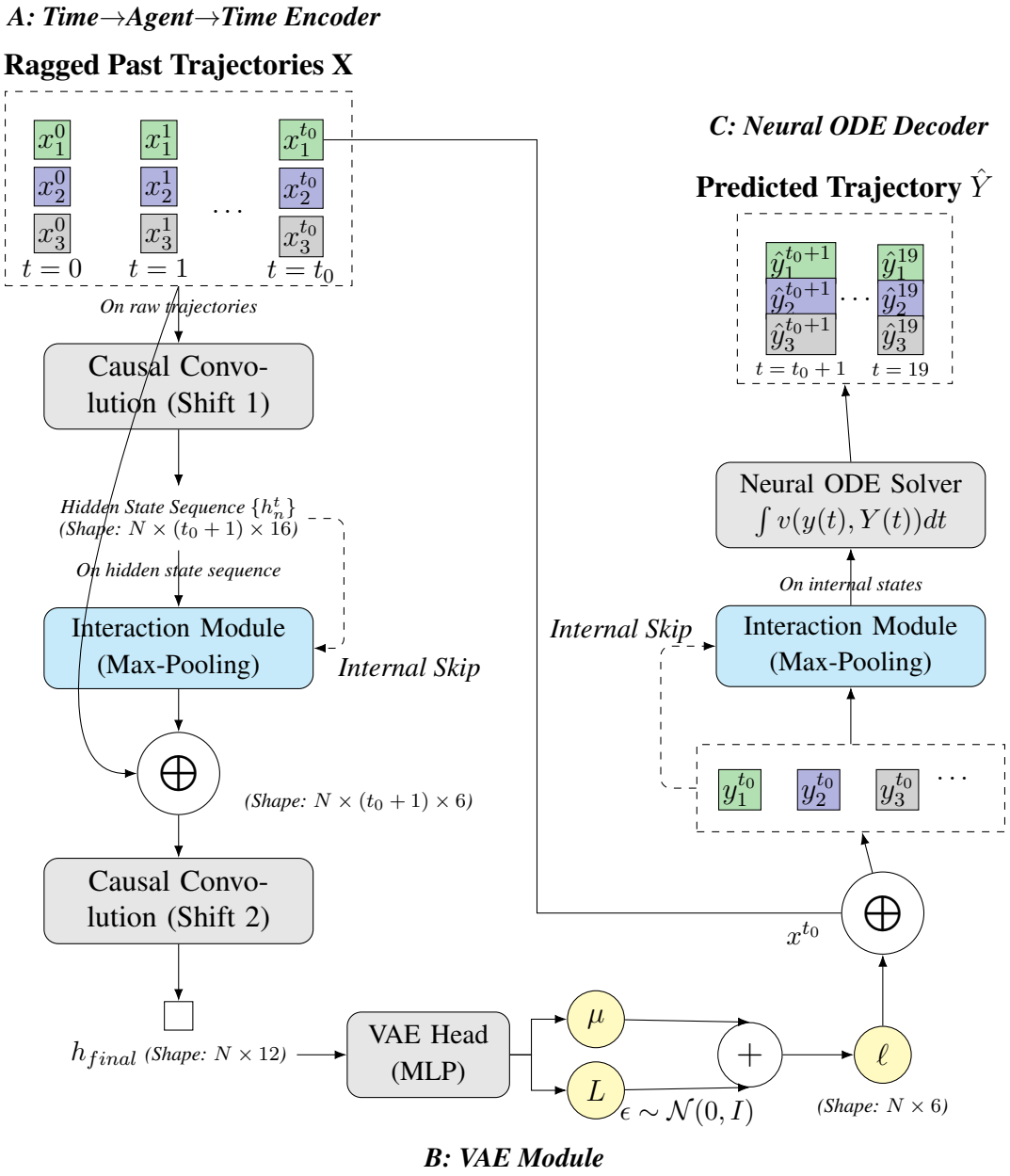


Figure 7.4: A complete diagram of the Time→Agent→Time Architecture with a Dynamic-Context Decoder. This model handles partial trajectories and is evaluated across multiple prediction horizons. The decoder's vector field $v(y(t), Y(t))$ continuously re-evaluates the social context $Y(t)$ at each step of the ODE solve, as shown by the internal feedback loop.

7.4.1 Static-Context Decoder

The static-context decoder, implemented as `StaticNeuralODE`, evolves each agent's trajectory independently during the prediction phase, shown in Figures 7.1, 7.2, and 7.3. The learned vector field is a function of only the agent's own current state, $v(y(t))$. In this model, all social context learned by the encoder is baked into the initial state y_0. The prediction is therefore an offline execution of a plan based on this fixed initial context; the agent does not react to the predicted movements of its peers during the rollout. This is a computationally efficient approach suitable for simpler, fixed-horizon benchmark tasks.

7.4.2 Dynamic-Context Decoder

The dynamic context decoder, implemented as `DynamicNeuralODE`, is a more powerful and adaptive model, as described in Figure 7.4. The vector field is autonomous but depends on the states of all agents in the scene at the current prediction time, $v(y(t), Y(t))$. This is achieved through a dynamic-context feedback mechanism. At each step of the ODE solve, interactions between the current predicted states of all agents, $Y(t)$, are re-evaluated. This re-evaluation is performed by an interaction module that implements the universality theorems from Theorem 3.5.7, combining the agent's own state with a permutation-equivariant summary of its peers and the two learned dummy biases, `v_sumr_bias` and `v_intr_bias`. This allows for a more realistic, online prediction where agents continuously react to each other's anticipated movements. This adaptability is crucial for handling open-world scenarios, such as reacting to a Late Arrival agent that was not present in the initial observation .

7.4.3 Solver Selection for Real-Time Application

For all decoders, we use a fixed-step solver (`diffrax.Euler()`) rather than a more complex adaptive solver. While adaptive solvers can be more efficient for certain problems, fixed-step methods offer a crucial advantage for real-time applications: predictable latency [105]. In such systems, the consistency of computation time is paramount; a result that arrives unpredictably, even if early, can be as detrimental as one that arrives late. An adaptive solver's computation time can vary depending on the complexity of the dynamics, whereas a fixed-step solver's is consistent. Furthermore, our empirical observations during development indicated that more advanced adaptive solvers not only failed to provide a significant performance improvement but in some cases even resulted in a minor degradation in predictive accuracy. The Euler method was therefore chosen as it provides the best balance of speed, predictable performance, and accuracy for our application.

7.5 Probabilistic Framework and Training Objectives

All of our models are cast in a variational autoencoder (VAE) framework discussed in §2.4.1: the encoder maps history into a latent code $z \sim q_\phi\big(z \mid X_{1:t_0}\big)$, and the decoder generates continuous trajectories conditioned on z and the last observed positions. We use a single-Gaussian posterior $\mathcal{N}\big(\mu, LL^\top\big)$ (with full covariance) and a fixed standard normal prior $p(z) = \mathcal{N}(0, I)$. Training minimizes a β-weighted evidence lower bound (ELBO), decomposed into three terms: reconstruction, boundary penalty (for multi-horizon models), and KL divergence. Following [106], we anneal the KL weight β linearly from 0 to 1 over the course of training, which empirically stabilizes learning and yields richer latent representations.

7.5.1 The VAE Module

The final encoder layer is a small MLP that takes the encoder's output vector and produces

- a mean vector $\mu \in \mathbb{R}^6$,
- the entries of a lower-triangular matrix $L \in \mathbb{R}^{6\times 6}$.

We then sample

$$z = \mu + L\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

enabling backpropagation through the reparameterization trick. In early experiments we tried full Gaussian mixtures, but the components collapsed to one, and the single-Gaussian posterior proved both simpler and equally effective.

Latent-Space Dimension Selection In all of our experiments we fixed the latent dimension at $\ell = 6$. This choice was guided by a number of practical heuristics:

1. **Sufficient capacity.** Six latent degrees of freedom proved large enough to capture the primary modes of variation in pedestrian trajectories without overfitting.
2. **Power-of-two alignment.** When concatenated with the 2D position (x, y) to form the decoder's initial state, a $6 + 2 = 8$-dimensional vector aligns naturally with a power-of-two representation, which can simplify certain GPU kernels and memory layouts.
3. **Manageable covariance size.** A full 6×6 covariance entails only 15 off-diagonal parameters, keeping the posterior's parameter count modest.

4. **Empirical plateau.** Increasing ℓ beyond 6 yielded negligible improvements in ADE/FDE, whereas reducing ℓ below 6 led to clear degradation in predictive accuracy.

5. **Mixture failure.** Early tests with higher-dimensional diagonal-covariance posteriors plus a linear projection into 6 D collapsed to a nearly degenerate subspace; direct sampling in 6 D was both simpler and more robust.

Together, these observations suggest that $\ell = 6$ strikes the right balance between expressivity, computational efficiency, and stable training.

7.5.2 Loss Functions

- **Reconstruction Loss (Standard Benchmarks):** For VAEST/VAETS, we simply use the mean squared error over the fixed 8→12-step window:

$$\mathcal{L}_{\text{rec}} \;=\; \frac{1}{12N} \sum_{i=1}^{N} \sum_{t=8}^{19} \big\| \hat{x}_t^{(i)} - x_t^{(i)} \big\|^2 .$$

- **Boundary-Aware Surrogate (Multi-Horizon Models):** To handle partial histories and censoring, define selection masks $\text{mask}_{i,t} \in \{0, 1\}$ based on each agent's valid window $[t_i^{\text{enter}},\, t_i^{\text{last}}]$, and let $\mathbf{1}_{\{\text{inside}\}}$ indicate whether a true position lies inside $\mathcal{B}$.

  Let for each agent i its first-appearance time be

$$t_i^{\text{enter}} \;=\; \min\{\, t : x_t^{(i)} \in \mathcal{B}\},$$

and its last-appearance time

$$t_i^{\text{last}} \;=\; \max\{\, t : x_t^{(i)} \in \mathcal{B}\}.$$

Let t_0 be the prediction start (so in our multi-horizon experiments $t_0 \in \{5, \ldots, 13\}$). We define the binary mask

$$\text{mask}_{i,t} = \begin{cases} 1, & x_{t_0}^{(i)} \in \mathcal{B}, \quad t_0 \le t \le t_i^{\text{last}}, \quad \big(\text{static-context}\big), \\ 1, & t_i^{\text{enter}} \le t \le t_i^{\text{last}}, \quad \big(\text{dynamic-context}\big), \\ 0, & \text{otherwise.} \end{cases}$$

In words, the static-context decoder only scores agents that were present at t_0, and then for every t from t_0 up to that agent's last observation whereas the dynamic-context decoder scores *every* agent from its own entry time onward, so that late-arrivals immediately peek in.

Then:

1. *Path-Matching Term:* sum of squared errors over all (i, t) with $\text{mask}_{i,t} = 1$, normalized by the total count of those entries.
2. *Anti-Lingering Penalty:* whenever $x_t^{(i)} \notin \mathcal{B}$ but $\hat{x}_t^{(i)} \in \mathcal{B}$, compute the squared distance to the boundary. This penalty is then normalized by the number of agents present at the start of the prediction. This provides a strong and stable per-agent gradient signal, ensuring that the model is properly incentivized to learn correct exit behavior.

Combined, the boundary-aware reconstruction loss is

$$\mathcal{L}_{\text{rec}} = \frac{\sum_{i,t} \text{mask}_{i,t} \, \|\hat{x}_t^{(i)} - x_t^{(i)}\|^2}{\sum_{i,t} \text{mask}_{i,t}} \; + \; \frac{\sum_{i : x_{t_0}^{(i)} \text{ exists}} \; \sum_{t :\, x_t^{(i)} \notin \mathcal{B},\, \hat{x}_t^{(i)} \in \mathcal{B}} \text{dist}\big(\hat{x}_t^{(i)}, \partial\mathcal{B}\big)^2}{\sum_i \mathbf{1}_{\{x_{t_0}^{(i)} \text{ exists}\}}}.$$

- **KL Divergence:** We compute the closed-form KL between $q_\phi = \mathcal{N}(\mu, LL^T)$ and $\mathcal{N}(0, I)$. Denoting the diagonal standard deviations $\sigma_i = L_{ii}$ and the off-diagonals by ℓ_{ij}, we have

$$\mathcal{L}_{\text{KL}} = \frac{0.5}{\sum_i \mathbf{1}_{\{x_{t_0}^{(i)} \text{ exists}\}}} \left[\sum_j \big(\mu_j^2 + \sigma_j^2 - 2\ln\sigma_j - 1\big) \; + \; \sum_{j \neq k} \ell_{jk}^2 \right].$$

Putting it all together, the full training objective for multi-horizon models is

$$\mathcal{L} \;=\; \mathcal{L}_{\text{rec}} \;+\; \beta\, \mathcal{L}_{\text{KL}}\,,$$

where β is annealed from 0 to 1 in training to stabilize latent learning.

Early-Stopping Criterion During training we record average displacement errors (ADE) and final displacement errors (FDE) on a held-out validation set every 20 iterations (for VAEST/-VAETS) or every 100 iterations (for multi-horizon models). We select the checkpoint with the lowest ADE, breaking ties with FDE. These metrics are reported in Chapter 8.

Chapter 8: Experimental Results & Analysis

This chapter presents a comprehensive empirical validation of the theoretical framework and neural architectures developed in this thesis. We structure our evaluation to first establish a strong baseline against existing methods on standard benchmarks before demonstrating the unique capabilities of our final, more expressive models on the challenging task of multi-horizon prediction from partial, ragged trajectories. The quantitative results, measured in Average and Final Displacement Error (ADE/FDE), are supplemented by a qualitative analysis that provides deeper insight into the learned interaction dynamics. The experiments collectively demonstrate that our framework is not only competitive with state-of-the-art models in standard settings—while being significantly more parameter-efficient—but also offers a robust and flexible solution for the complexities of real-world, multi-agent forecasting.

8.1 Experimental Setup

This section details the datasets, evaluation metrics, and implementation details used to train and evaluate our models.

8.1.1 Datasets

All experiments are conducted using the widely-adopted ETH [107] and UCY [98] datasets, which consist of real-world pedestrian trajectories captured from a bird's-eye view. We use the pre-processed and split versions provided by the Social GAN [43] repository, which is standard practice in the field.

- **Benchmark Models (VAEST & VAETS):** For the fixed-horizon benchmark comparisons, we train and evaluate on all five standard datasets: ETH, HOTEL, UNIV, ZARA1, and ZARA2, using the leave-one-out cross-validation strategy. These experiments use only the fully-observed trajectories, as is common for this benchmark.

- **Multi-Horizon Models (Time→Agent→Time):** For the primary experiments on partial trajectories, we focus on the ZARA01, ZARA02, and ZARA03 datasets. Because these datasets were all captured at the same physical location, their absolute coordinate systems are aligned. This shared frame of reference makes them particularly well-suited for a principled evaluation of our final models, which use absolute coordinates to handle boundary-awareness in a way that is natural for robotics applications. Furthermore, these datasets feature a high density of complex, interactive scenarios in a confined space, which is most relevant to our work.

8.1.2 Evaluation Metrics

We evaluate our models using two standard metrics for trajectory prediction:

1. **Average Displacement Error (ADE):** The average L_2 distance between the predicted and

ground-truth positions over all predicted time steps:

$$\text{ADE} = \frac{1}{(19 - t_0)\, N} \sum_{i=1}^{N} \sum_{t=t_0}^{19} \left\| \hat{x}_t^{(i)} - x_t^{(i)} \right\|.$$

2. **Final Displacement Error (FDE):** The L_2 distance between the predicted and ground-truth final positions at the end of the prediction horizon:

$$\text{FDE} = \frac{1}{N} \sum_{i=1}^{N} \left\| \hat{x}_{19}^{(i)} - x_{19}^{(i)} \right\|.$$

For our probabilistic models, we follow the standard protocol of generating 20 samples per scene and reporting the ADE and FDE of the sample that is closest to the ground truth [30, 35, 36, 38, 39, 43].

8.1.3 Implementation Details

Our models were implemented using the JAX ecosystem, as detailed in Chapter 7. A key factor for stable training was the use of differential optimizers and learning rates. We observed in early experiments that using a single optimizer for the entire model often led to training collapse.

To remedy this, we partitioned the model's parameters. The `main_linear` layer of the decoder's vector field, which is responsible for the final physical `(x, y)` velocity, was trained with the AdamW optimizer [104] and a larger learning rate (`lr_main`). All other model parameters, which learn the more abstract latent representations, were trained with the standard Adam optimizer [103] and a smaller, more conservative learning rate (`lr_latent`). This allows the model to aggressively fit the primary physical dynamics while ensuring the stable learning of the

underlying latent space. A KL annealing schedule was used for all models, where the weight of the KL-divergence term, β, was linearly increased from 0 to 1. The specific hyperparameters for each of our six main experiments are summarized in Table 8.1.

For all experiments, we used an early-stopping criterion based on the ADE on a held-out validation set, breaking ties with FDE, to select the best model checkpoint for final evaluation.

Table 8.1: Hyperparameter settings for all models and datasets

Hyperparameter	**VAEST**	**VAETS**	**Relative (ZARA)**	**Absolute (ZARA)**	**Static (Multi-Horizon)**	**Dynamic (Multi-Horizon)**
`lr_latent_init`	1e-4	1e-4	1e-4	2e-4	2e-4	1e-4
`lr_latent_peak`	1e-5	1e-5	1e-5	2e-5	1e-5	1e-5
`lr_main_init`	1e-3	1e-3	1e-3	2e-3	2e-3	1e-3
`lr_main_peak`	1e-4	1e-4	1e-4	2e-4	1e-4	1e-4
`warmup_steps`	2000	2000	2000	3500	10000	7500
`total_steps`	3000	3000	3000	5000	15000	20000
`kl_anneal_steps`	3000	3000	3000	5000	10000	10000
`batch_size`	128	128	128	128	128	128
`latent_size`	6	6	6	6	6	6
`hidden_size_1`	16	8	16	16	16	16
`hidden_size_2`	–	–	–	–	12	12
`interaction_size`	8	8	8	8	8	8
`v_intr_size`	–	–	–	–	–	4

8.2 Benchmark Performance and Encoder Comparison

To establish a baseline for our models, we first evaluate their performance on the standard, fixed-horizon (8 observe, 12 predict) benchmark task using only fully-observed trajectories. This allows for a direct comparison with existing methods and provides an analysis of our different encoder architectures on this simplified problem.

8.2.1 Quantitative Results

Table 8.2 presents the Average Displacement Error (ADE) and Final Displacement Error (FDE) for our two canonical benchmark models, VAEST (Agent→Time) and VAETS (Time→Agent). The models were evaluated using the standard leave-one-out protocol on the five ETH/UCY datasets. We compare our results against several strong baselines, including Social-LSTM, Social-GAN, and the highly-efficient Social-STGCNN.

Table 8.2: ADE / FDE on the ETH/UCY benchmark

Model	**ETH**	**HOTEL**	**UNIV**	**ZARA1**	**ZARA2**	**Average**	**Params**
Social-LSTM [30]	1.09/2.35	0.79/1.76	0.67/1.40	0.47/1.00	0.56/1.17	0.72/1.54	∼264k
Social-GAN [43]	0.87/1.62	0.67/1.37	0.76/1.52	0.35/0.68	0.42/0.84	0.61/1.21	∼46.3k
Social-STGCNN [38]	**0.64/1.11**	0.49/0.85	0.44/0.79	**0.34/0.53**	**0.30/0.48**	**0.44/0.75**	∼7.6k
VAEST (Ours)	0.70/1.17	0.49/0.77	**0.43/0.77**	0.45/0.57	0.33/**0.48**	0.48/**0.75**	∼2.5k
VAETS (Ours)	0.77/1.43	**0.46/0.70**	0.44/0.79	0.43/0.58	0.31/0.49	0.48/0.80	∼2.3k

8.2.2 Discussion

The results show that both of our canonical encoder architectures, VAEST and VAETS, achieve performance that is comparable to the state-of-the-art Social-STGCNN model. While the baseline model achieves a slightly better average ADE, our models are competitive on FDE, with our VAEST model matching the average FDE of the baseline.

Most notably, our models achieve this performance with remarkable efficiency. The VAEST and VAETS models have approximately 2.5k and 2.3k total parameters, respectively—more than 3 times fewer than Social-STGCNN [38] (∼7.6k). This efficiency is particularly evident in our decoder design; the entire continuous-time Neural ODE decoder is implemented with only 352

parameters. This demonstrates that our architectures, even with a simple non-adaptive convolution, provide a powerful and exceptionally lightweight baseline for the standard trajectory prediction task

8.3 Performance on Partial Trajectories and Multi-Horizons

The primary contribution of this thesis is a model that can robustly handle partial, ragged trajectories and generate predictions across a range of time horizons. This section evaluates the performance of our final Time→Agent→Time architecture on this more realistic and challenging task. All experiments are conducted on the ZARA01, ZARA02, and ZARA03 datasets. Because these datasets were all captured at the same physical location, their absolute coordinate systems are aligned. This shared frame of reference makes them particularly well-suited for a principled evaluation of our final models, which use absolute coordinates to handle boundary-awareness in a way that is natural for robotics applications. Furthermore, these datasets feature a high density of complex, interactive scenarios in a confined space, which is most relevant to our work.

8.3.1 Bridging Experiment: Relative vs. Absolute Coordinates

We first present a bridging experiment to methodologically connect the standard relative-coordinate benchmark with the absolute-coordinate system used by our final models. For this, we used the VAEST (Agent→Time) architecture. As shown in Table 8.3, when evaluated on the fixed 8/12 task, the model using absolute coordinates performs better than its relative-coordinate counterpart on the ZARA datasets. We hypothesize that this improvement arises because an absolute

frame directly encodes the shared scene geometry and boundary constraints—information that the encoder would otherwise need to infer from relative displacements alone. This result, while expected, is a crucial step that validates our use of absolute coordinates for the final, boundary-aware models, for which an absolute frame is a more natural and direct fit for robotics applications in a known environment.

Table 8.3: Relative vs. Absolute Coordinates on ZARA (8/12 Split)

Model	**ZARA01 ADE/FDE**	**ZARA02 ADE/FDE**	**ZARA03 ADE/FDE**
Relative	0.451 / 0.639	0.315 / 0.489	0.450 / 0.711
Absolute	0.363 / 0.657	0.278 / 0.490	0.362 / 0.661

8.3.2 Multi-Horizon Performance

Tables 8.4, 8.5, and 8.6 present the main results for our final architecture, detailing the Average Displacement Error (ADE) and Final Displacement Error (FDE) for both the static- and dynamic-context decoders across nine different observation/prediction horizons. For direct comparison, we also include the results of the simpler benchmark models using relative and absolute coordinates on the standard 8/12 task.

8.3.3 Discussion

The results demonstrate the flexibility and robustness of our final architecture. Across all three datasets, performance on both ADE and FDE consistently improves as the observation horizon increases, showing that the model effectively leverages longer histories to produce more accurate short-term forecasts.

Table 8.4: Multi-Horizon Performance (ADE / FDE) on ZARA01

Horizon	Relative	Absolute	Static	Dynamic
06/14	—	—	0.387/0.722	0.381/0.686
07/13	—	—	0.353/0.670	0.345/0.630
08/12	0.451/0.639	0.363/0.657	0.332/0.624	**0.326/0.600**
09/11	—	—	0.311/0.580	0.307/0.563
10/10	—	—	0.290/0.535	0.293/0.537
11/09	—	—	0.276/0.506	0.277/0.500
12/08	—	—	0.258/0.466	0.261/0.466
13/07	—	—	0.239/0.425	0.240/0.422
14/06	—	—	0.222/0.384	0.223/0.383

* The Dynamic-context decoder sees only each new agent's first appearance time and location (no further future peek). This ensures (1) existing agents are unaffected until the newcomer arrives, and (2) the newcomer only influences others from the moment it appears.

Table 8.5: Multi-Horizon Performance (ADE / FDE) on ZARA02

Horizon	Relative	Absolute	Static	Dynamic
06/14	—	—	0.319/0.586	0.303/0.561
07/13	—	—	0.292/0.538	0.283/0.525
08/12	0.315/**0.489**	0.278/0.490	0.275/0.499	**0.264/0.489**
09/11	—	—	0.259/0.471	0.247/0.453
10/10	—	—	0.241/0.434	0.232/0.421
11/09	—	—	0.224/0.401	0.215/0.389
12/08	—	—	0.201/0.357	0.200/0.357
13/07	—	—	0.184/0.325	0.186/0.325
14/06	—	—	0.168/0.289	0.168/0.289

Table 8.6: Multi-Horizon Performance (ADE / FDE) on ZARA03

Horizon	Relative	Absolute	Static	Dynamic
06/14	—	—	0.435/0.803	0.417/0.760
07/13	—	—	0.407/0.760	0.374/0.692
08/12	0.450/0.711	0.362/0.661	0.377/0.691	**0.349/0.647**
09/11	—	—	0.346/0.627	0.327/0.596
10/10	—	—	0.323/0.579	0.306/0.553
11/09	—	—	0.300/0.537	0.287/0.513
12/08	—	—	0.277/0.490	0.271/0.476
13/07	—	—	0.257/0.450	0.250/0.435
14/06	—	—	0.233/0.400	0.229/0.389

When comparing the decoders, the dynamic-context decoder model consistently outperforms the static-context decoder model across nearly all datasets and horizons. A caveat for this comparison is that the dynamic-context decoder, by design, has access to minimal future information: the first appearance time and location of agents who enter the scene during the prediction window. However, this information is revealed causally within the ODE solver; the new agent only begins to influence the interactive vector field at the precise moment it appears. This setup makes the dynamic-context model an excellent tool for simulations that study how existing agents react to new arrivals in real-time. It could also serve as a powerful component in a larger system where the appearance of new agents is predicted by a separate model.

Despite this informational advantage, the performance gain of the dynamic-context model is marginal. As discussed, this is likely because the 8-second prediction window is often too short for the model's adaptability to become a decisive factor, as a powerful encoder can anticipate and bake in most interactions into the static-context decoder's initial state.

8.4 Qualitative Analysis

While quantitative metrics like ADE and FDE are essential for evaluating performance, they do not fully capture the nuanced, dynamic behavior of the models. To provide a more intuitive understanding of our framework's capabilities, this section presents a qualitative analysis of the learned interaction mechanisms at the critical `t=5` transition point—the final moment of observation and the first moment of prediction. We use a representative scene from the ZARA01 test set, analyzed with our final Time→Agent→Time model with a dynamic-context decoder.

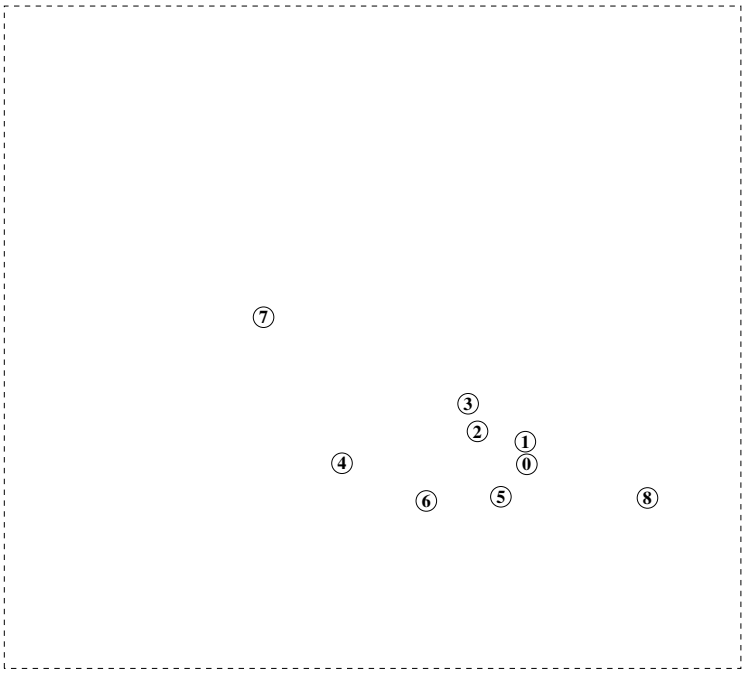


Figure 8.1: Agent positions at `t=5` in ZARA01 (agents 0–8).

8.4.1 Encoder Interactions: Summarizing the Past

The encoder's role is to summarize the entire observed history up to the point of prediction. Our framework achieves this without explicit graph structures by using permutation-equivariant max-pooling. To interpret what the model has learned, we can look inside this operation by applying an `argmax` to the pre-pooling features to identify the most salient peer for each of the 8 channels in the interaction summary.

Table shows the raw output of this analysis for the 9 agents present at `t=5`. Each row corresponds to a target agent, and each column to a channel, with the value indicating the index of the most influential agent (where 9 is the environment's `H_bias` and 10 is the maxitive `I_bias`). Table tab:encoder-agg aggregates this information into an adjacency matrix, showing the total number of channels through which one agent influences another.

Discussion The tables reveal a clear and plausible interaction structure. For example, in the aggregate matrix, agent 7 is a strong influencer of agents 0-3, which corresponds to their spatial proximity and potential for future interaction. The model has learned to use different channels to monitor different peers, effectively creating a dynamic, attention-like mechanism without explicit weights.

Table 8.7: Most Salient Agents per Channel for the Encoder at `t=5`

Agent	Ch0	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7
0	7	7	7	8	7	0	0	9
1	7	7	7	8	7	0	0	9
2	7	7	7	8	7	0	0	9
3	7	10	7	8	7	0	0	9
4	9	10	7	8	7	0	0	9
5	7	10	9	8	7	0	0	9
6	7	10	7	8	7	0	0	9
7	9	10	7	8	9	0	8	9
8	7	7	9	8	7	8	0	9

Table 8.8: Aggregate Encoder Adjacency Matrix at `t=5`
(Entry (i, j) is the # of channels where agent j influences agent i.)

Target	0	1	2	3	4	5	6	7	8	H_bias (9)	I_bias (10)
0	2	0	0	0	0	0	0	4	1	1	0
1	1	0	0	0	0	0	0	4	1	1	0
2	1	0	0	0	0	0	0	4	1	1	0
3	1	0	0	0	0	0	0	4	1	1	1
4	1	0	0	0	0	0	0	3	1	2	1
5	1	0	0	0	0	0	0	3	1	2	1
6	1	0	0	0	0	0	0	3	1	1	1
7	1	0	0	0	0	0	0	3	3	1	1
8	1	0	0	0	0	0	0	3	2	2	0

8.4.2 Decoder Interactions: Planning the Future

At the same `t=5` transition point, the dynamic-context decoder begins its forward prediction. Its internal interaction module re-evaluates the social context to plan the initial step of the future trajectory. This module uses a smaller, 4-dimensional interaction vector.

Discussion The decoder's interaction patterns are different from the encoder's. It has learned a distinct policy for future planning. For example, agent 1 is now a more prominent source of influence for several agents. This analysis shows that the decoder is not merely re-using the encoder's summary but is performing its own active, forward-looking social reasoning at the start of the prediction. This qualitative result provides strong evidence for the sophisticated, dynamic, and interpretable nature of our framework.

Table 8.9: Most Salient Agents per Channel for the Decoder at `t=5`

Agent	Ch0	Ch1	Ch2	Ch3
0	1	4	7	9
1	1	1	7	9
2	1	4	7	9
3	1	4	7	9
4	8	4	7	9
5	1	1	7	9
6	1	4	7	9
7	1	1	7	9
8	1	4	9	9

8.5 Chapter Summary

The experiments in this chapter provide a comprehensive empirical validation of our theoretical framework and neural architectures. The results demonstrate that our models are not only

Table 8.10: Aggregate Decoder Adjacency Matrix at `t=5`
(Entry (i, j) is the # of channels where agent j influences agent i.)

Target	0	1	2	3	4	5	6	7	8	v_bias (9)
0	0	1	0	0	1	0	0	1	0	1
1	0	2	0	0	0	0	0	1	0	1
2	0	1	0	0	1	0	0	1	0	1
3	0	1	0	0	1	0	0	1	0	1
4	0	0	0	0	1	0	0	1	1	1
5	0	2	0	0	0	0	0	1	0	1
6	0	1	0	0	1	0	0	1	0	1
7	0	2	0	0	0	0	0	1	0	1
8	0	1	0	0	1	0	0	1	0	2

competitive in standard benchmark settings but also offer unique capabilities for handling the complexities of real-world trajectory data.

Our investigation began by establishing a strong baseline. The results showed that our canonical encoder architectures, VAEST and VAETS, achieve performance comparable to state-of-the-art models like Social-STGCNN [38] on the fixed-horizon ETH/UCY benchmark, while being substantially more parameter-efficient. The entire continuous-time decoder, for instance, requires only 352 parameters.

The core contribution of our work was evaluated through the Time→Agent→Time architecture, which was tested on partial trajectories across a wide range of prediction horizons. The quantitative results confirmed the model's flexibility, with performance improving predictably as the observation window increased. The dynamic-context decoder consistently, though marginally, outperformed its static-context counterpart, validating its more sophisticated, adaptive design.

Finally, our qualitative analysis provided deeper, interpretable insights into the model's learned behavior. By visualizing the internal states of the interaction modules, we demonstrated that our max-pooling mechanism learns a dynamic, attention-like structure that effectively mim-

ics a message-passing [37] framework. Furthermore, a direct visual comparison highlighted the superior behavioral fidelity of the dynamic-context decoder in complex interactive scenarios.

Taken together, these findings confirm that our framework successfully translates deep theoretical principles into a practical system. The final model is efficient, robustly handles partial and variable-length histories, and provides a powerful and flexible foundation for real-world, multi-horizon trajectory forecasting.

Chapter 9: Conclusion & Future Work

9.1 Summary of Contributions

This thesis addressed the critical gap between standard trajectory prediction benchmarks and the complexities of real-world, interactive scenarios. We argued that robust forecasting systems must be built on a foundation that respects three signature characteristics: boundary awareness, causal-pointwise equivariance, and continuous-time dynamics. To this end, we developed a complete theoretical framework and a series of novel neural architectures that successfully translate these principles into a practical and high-performance prediction model.

Our contributions began with a deep dive into the mathematical underpinnings of the problem. We established universality theorems for set-equivariant functions, proving that any such function can be approximated by architectures using permutation-equivariant pooling, such as the max-pooling in our models . We also proved that the spatio-temporal symmetry group of the problem, $\mathbb{R} \times \mathfrak{S}_n$, has a unique decomposition, providing a rigorous justification for the canonical Shift$\rightarrow$Set and Set$\rightarrow$Shift encoder pathways . The temporal dynamics were grounded in the theory of retarded functional differential equations (RFDEs), for which we established that causal convolutional networks are universal approximators .

This theoretical work motivated our practical framework, which was built around the three signature characteristics. For boundary awareness, we introduced a complete taxonomy of trajec-

tory censoring and developed a principled, survival-analysis-inspired loss function with a differentiable anti-lingering penalty . For causal-pointwise equivariance, we implemented a series of encoder architectures, including the final, expressive Time→Agent→Time model, and developed the temporal conjugation pipeline as a novel, JIT-compatible technique for handling ragged, real-world data . For continuous-time modeling, our framework uses adaptive causal convolutions in the encoder and Neural ODEs in the decoder, allowing it to naturally handle variable-length histories and predict at any future time point .

Finally, we implemented these principles in a series of highly efficient, JAX-based models. Our experimental results demonstrated that our architectures are competitive with state-of-the-art models on standard, fixed-horizon benchmarks while using significantly fewer parameters. More importantly, our final model successfully handles the more challenging task of multi-horizon prediction from partial trajectories, showcasing a unique flexibility and robustness that is essential for real-world applications in robotics and autonomous systems.

9.2 Discussion and Broader Implications

The contributions of this thesis extend beyond the specific architectures and performance metrics presented. The work as a whole offers a new perspective on trajectory prediction, grounded in deep theoretical principles and motivated by the practical demands of real-world autonomous systems. This section discusses the broader implications of our framework.

9.2.1 Bridging Rigorous Theory and Practical Implementation

A central theme of this work has been the demonstration of how deep mathematical principles can directly and powerfully inform the design of novel neural network architectures . Rather than relying on purely empirical or heuristic designs, our models are constructed to respect the fundamental symmetries of the problem, as dictated by group theory . Our use of set-based, weight-free interaction modules is not just a practical choice but is theoretically justified by our work on support functionals and idempotent integration, which provides a parallel to the measure-theoretic foundations of other models . Similarly, our continuous-time dynamics are grounded in the theory of RFDEs . By showing how these abstract mathematical concepts can be translated into efficient, JAX-based implementations, this thesis serves as a case study in principled, theory-driven architectural design.

9.2.2 Moving Beyond Standard Benchmarks

Our experimental results highlight the limitations of the standard, fixed-horizon trajectory prediction benchmarks. While these benchmarks are invaluable for comparing models, their exclusive focus on fully-observed trajectories creates a significant disconnect from real-world scenarios where agents constantly enter and exit the scene . Our work confronts this Boundary Challenge directly, not only by proposing a model that robustly handles partial and ragged histories but also by advocating for an evaluation paradigm that measures performance across a wide range of observation and prediction horizons. This multi-horizon evaluation provides a more complete and realistic assessment of a model's capabilities and represents a step towards more meaningful and practical benchmarks for the field.

9.2.3 Implications for Robotics

The architectural choices in our final model were deliberately made with the specific needs of robotics and other embodied agents in mind. The use of an absolute coordinate system is the natural choice for a robot operating in a known, fixed environment like a warehouse or a city block . The emphasis on boundary awareness is critical for any real-world system that must reason about agents entering and leaving its finite sensor field . Finally, the dynamic-context decoder, with its ability to react in real-time to the appearance of new agents, is a crucial feature for safe and adaptive planning in cluttered, interactive environments . By focusing on these practical requirements, this thesis provides a robust and theoretically-grounded framework that is well-suited for the next generation of autonomous navigation and decision-making systems.

9.3 Limitations and Critique

While this thesis presents a robust and theoretically-grounded framework, it is important to acknowledge its limitations, which in turn motivate promising directions for future research.

Scope of Boundary Awareness Our work focuses specifically on the problem of boundary-induced censoring, where agents enter or exit a fixed sensor area. We deliberately scope our framework to this challenge, and it does not explicitly model other forms of missing data, such as temporary occlusions from sensor failures or objects blocking the line of sight. While our continuous-time decoders can gracefully handle small, irregular gaps in data through interpolation, a principled framework for long-duration occlusions would likely require a different approach.

Empirical Performance of the Dynamic-Context Decoder Our results consistently showed that the dynamic-context decoder marginally outperformed its static-context counterpart. We hypothesize that this is a limitation of the evaluation protocol rather than the model itself; the short, 8-second (20-frame) prediction windows of the standard benchmarks are often not long enough or complex enough for the full adaptive capabilities of the dynamic-context decoder to become a decisive advantage. A more thorough validation on longer, more complex, and more densely interactive scenarios would be needed to fully quantify the benefits of the dynamic approach.

Absence of Formal Ablation Studies A primary focus of this thesis has been the development of a unified theoretical framework and its end-to-end neural implementation. Due to the significant scope of this undertaking, a comprehensive suite of ablation studies—controlled experiments that systematically isolate the individual performance contributions of each component—was not performed. While our results include crucial comparisons between the different high-level architectures, a more granular investigation remains an important area for future work. While the current work successfully demonstrates the performance of the complete, integrated system, these targeted ablation studies would provide a deeper understanding of the interplay between its components and are a rich avenue for future research.

The Max-Operator as a Physical Simplification A final, more philosophical critique concerns the use of the max-operator for all social pooling. While this choice is theoretically justified by our work on support functionals and provides a weight-free alternative to measure-theoretic approaches, it is also a strong simplification of the underlying physics. Real-world agents are not weightless points; they are entities with volume and momentum, and their interactions are

governed by complex physical and social forces, not just a collection of extremal influences.

Our model abstracts these complex interactions by selecting, for each of the 8 (or 4) channels in our interaction summary, the single most salient peer. While our universality theorems show that this top-b aggregation is highly expressive, it is still an extremal mechanism. A valuable direction for future research would be to explore alternative, inclusionary permutation-equivariant pooling mechanisms such as attention scores to investigate whether a model that aggregates influences from multiple peers, rather than just the most salient ones per feature, could capture more nuanced interactive behaviors and lead to further improvements in predictive accuracy.

9.4 Future Work

The framework developed in this thesis provides a robust and theoretically-grounded foundation for interactive trajectory prediction, but it also opens up several promising avenues for future research. This section outlines three key directions: extending the probabilistic model to capture a richer class of uncertainties, applying the learned dynamics to control and planning problems, and performing a rigorous suite of ablation studies to further dissect the contributions of each architectural component.

Probabilistic and Stochastic Extensions Our current model uses a single, full-covariance Gaussian in its VAE module to represent uncertainty. While effective, a natural extension would be to replace this with a more expressive probabilistic model, such as a Gaussian Mixture Model (GMM), to better capture complex, multi-modal future behaviors. A more profound extension would be to move beyond latent-space uncertainty and model the inherent randomness of the dynamics directly. This could be achieved by replacing the Neural ODE decoder with a neural

stochastic differential equation (SDE). A neural SDE learns both a drift and a diffusion term, allowing it to generate a distribution of trajectories that captures the aleatoric uncertainty of the system. This would be a significant step towards a more complete and realistic model of stochastic multi-agent dynamics.

Inverse Planning and Control The dynamic-context decoder provides a powerful, forward-in-time simulation of how agents react to one another. An exciting direction for future work is to leverage this learned model for inverse planning and control [108]. For example, one could use the model to plan a path for an autonomous robot through a crowded space by optimizing a trajectory that minimizes the disruption to the predicted paths of the surrounding pedestrians. Furthermore, this framework could be connected to more formal control theory approaches, such as Hamilton-Jacobi-Bellman (HJB) formulations, to solve for optimal, task-driven policies in complex, interactive environments.

Rigorous Ablation Studies Finally, a comprehensive suite of ablation studies would provide a deeper, quantitative understanding of the specific contributions of each novel component in our framework. Key experiments would include:

- A direct comparison of the final Time→Agent→Time encoder against simpler two-stage encoders on the multi-horizon task to definitively measure the benefit of its increased expressivity.
- A formal benchmark of the inference latency of the static- vs. dynamic-context decoders to quantify the practical trade-offs for real-time robotics applications.

- An evaluation of the model trained with and without the anti-lingering penalty to isolate the performance gain from our boundary-aware loss function.
- An ablation study that trains models with only integral, only maxitive, and both convolutions to verify our hypothesis that the combined approach is more robust for short histories.
- An investigation into alternative, inclusionary pooling mechanisms such as attention scores to compare against the extremal max-pooling operator used in our current interaction modules.

These targeted experiments would provide valuable insights for the design of future trajectory prediction architectures and further validate the principles established in this thesis.

Appendix A: Core Architectural Modules

This appendix contains the JAX implementations of the core, reusable components of the architectures described in Chapter 7.

A.1 Causal Convolutions

The temporal processing in our encoders is handled by a custom causal convolution module. The core of this module is the `Causal_Dynamics` class, which acts as a neural network that generates the convolution kernel itself. We use two versions of this class: a non-adaptive version for the fixed-horizon benchmark models and a more powerful adaptive version for the final multi-horizon architecture.

Activation Function All MLPs in the kernel generators use `lipswish`, a 1-Lipschitz continuous activation function [109] that is a scaled version of the SiLU activation. This choice aids in maintaining stable gradients during training.

```
def lipswish(x):
    return 0.909 * jax.nn.silu(x)
```

Adaptive Causal Kernel The `Causal_Dynamics` class used in the final Time→Agent→Time model is adaptive. Its `__call__` method takes both the observed length (`ob_len`) and the time

lag (time) as inputs. This allows the MLP to generate a kernel that is explicitly conditioned on the length of the partial trajectory it is processing, which is crucial for handling ragged data.

```
class Causal_Dynamics(eqx.Module):

    data_size: int

    mlp: eqx.nn.MLP

    norm: eqx.nn.LayerNorm


    def __init__(self, data_size, hidden_size, depth,

                                *, key, **kwargs):

        super().__init__(**kwargs)

        self.data_size = data_size

        self.mlp = eqx.nn.MLP(

            in_size=1 + 1,  # Takes both ob_len and time

            out_size=data_size * hidden_size,

            width_size=data_size * hidden_size,

            depth=depth - 1,

            activation=lipswish,

            final_activation=lipswish,

            key=key,

        )

        self.norm = eqx.nn.LayerNorm(

            data_size * hidden_size,

            eps=1e-6,

        )
```

```
    def __call__(self, ob_len, time):
        ob = jnp.asarray(ob_len)
        t = jnp.asarray(time)
        x = self.mlp(jnp.concatenate([ob[None], t[None]]))
        x = self.norm(x).reshape((-1, self.data_size))
        return (t[None] >= 0) * x # Enforce causality
```

Non-Adaptive Causal Kernel The `Causal_Dynamics` class used in the benchmark VAEST and VAETS models is simpler and non-adaptive. Since these models operate on a fixed 8-step observation window, the kernel does not need to adapt to the trajectory length. Its `__call__` method is therefore only a function of the time lag.

```
class Causal_Dynamics_NonAdaptive(eqx.Module):
    data_size: int
    mlp: eqx.nn.MLP
    norm: eqx.nn.LayerNorm

    def __init__(self, data_size, hidden_size, depth,
                                *, key, **kwargs):
        super().__init__(**kwargs)
        self.data_size = data_size
        self.mlp = eqx.nn.MLP(
            in_size=1,  # Takes only time
```

```
            out_size=data_size * hidden_size,
            width_size=data_size * hidden_size,
            depth=depth,
            activation=lipswish,
            final_activation=lipswish,
            key=key,
        )
        self.norm = eqx.nn.LayerNorm(
            data_size * hidden_size,
            eps=1e-6,
        )

    def __call__(self, time):
        t = jnp.asarray(time)
        x = self.mlp(t[None])
        x = self.norm(x).reshape((-1, self.data_size))
        return (t[None] >= 0) * x # Enforce causality
```

Convolution Helpers The following functions implement the integral and maxitive convolutions. They take a time grid ti, a data sequence `yi`, and a kernel-generating model as input. The `combined_conv` function concatenates the results of both convolution types, and the `conv_list` function efficiently applies this combined convolution across all time steps of a sequence using `jax.vmap`.

```
def intgr_conv(tt, ti, yi, model, hidden_size_int):
    # Build the kernel matrix K where K[s] = model(T_obs, tt - s)
    K = jax.vmap(
    lambda t: model(tt - ti[0], tt - t)[:hidden_size_int]
    )(ti)
    val = (K @ yi[..., None]).squeeze()
    return jnp.trapezoid(val, ti, axis=0)
```

```
def max_conv(tt, ti, yi, model, hidden_size_int):
    # Build the kernel matrix M where M[s] = model(T_obs, tt - s)
    M = jax.vmap(
    lambda t: model(tt - ti[0], tt - t)[hidden_size_int:]
    )(ti)
    scores = (M @ yi[..., None]).squeeze()
    # Mask out future values to ensure causality
    mask = jnp.where(ti <= tt, 0.0, -jnp.inf)[...,None]
    return jnp.max(scores + mask, axis=0)
```

```
def combined_conv(tt, ti, yi, model, hidden_size_int):
    intgr = intgr_conv(tt, ti, yi, model, hidden_size_int)
    max_part = max_conv(tt, ti, yi, model, hidden_size_int)
    return jnp.concatenate([intgr, max_part], axis=-1)
```

```
def conv_list(ti, yi, model, hidden_size_int):
    return jax.vmap(combined_conv,
```

```
    in_axes=(0, None, None, None, None))(
        ti, ti, yi, model, hidden_size_int
    )
```

A.2 Interaction Modules

This section details the two distinct interaction modules developed for our different coordinate systems. Both are implementations of the universality theorem, Theorem 3.5.6, but they differ in their input structure to accommodate either a shared, absolute reference frame or an agent-centric, relative one.

Interaction Module for Absolute Coordinates The `Tf_with_sumr_intr` function is designed to operate on absolute coordinates. It takes an entire scene tensor, representing all agents in a shared reference frame, and efficiently computes all-pairs interactions in a single, parallelized pass using nested `vmaps`. It includes components for both a global `scene_summary` and individualized interactions, though the global summary was disabled in our final experiments.

```
def Tf_with_sumr_intr(
        scene,
        mask,
        max_bias,
        sumr_model,
        intr_model,
        tf_model
        ):
```

```
s_bias, i_bias, h_bias = max_bias


# Compute a global summary of the scene by pooling over all agents
scene_summary = jnp.max(
    jnp.vstack([
        jax.vmap(sumr_model)(scene) - jnp.inf * ~mask[...,None],
        s_bias
    ]), axis=0
)


# Add a learnable "dummy" agent (H_bias) to the scene
mask = jnp.hstack([mask, True])
scene = jnp.vstack([scene, h_bias])
indices = jnp.arange(scene.shape[0])


# Define a function to compute pairwise interactions
# for a single agent (idx)
def pairwise_interaction(idx):
    return jax.vmap(
        lambda ind: intr_model(jnp.hstack([scene[idx],


    )(indices) - jnp.inf * ~(mask | (indices == idx))[...,None]
```

```
    # Vmap the full interaction logic over all agents in the scene
    interactions = jax.vmap(lambda idx: tf_model(
        jnp.hstack([
            scene[idx],
            scene_summary,
            jnp.max(
                jnp.vstack([pairwise_interaction(idx), i_bias]),
                axis=0
            )
        ])
    ))(indices)

    return interactions[:-1]
    # Exclude the dummy agent from the final output
```

Interaction Module for Relative Coordinates The `Tf_with_intr` function is designed to operate on relative coordinates. Unlike the absolute version, it is built to process a pointed set—a single target agent `x` and a `scene` tensor that has been re-centered relative to that agent. In the full benchmark encoders, this function is vectorized using an outer `vmap` to process all `N` agent-centric views of a scene in parallel.

```
def Tf_with_intr(
    x,
    scene,
```

```
    mask,
    max_bias,
    intr_model,
    tf_model,
):

    i_bias, h_bias = max_bias

    # Add a learnable "dummy" agent (H_bias) to the scene
    mask = jnp.hstack([mask, True])
    scene = jnp.vstack([scene, h_bias])

    # Define a function to compute pairwise interactions
    # for the target agent 'x'
    def pairwise_interaction(x):
        return jax.vmap(
            lambda y: intr_model(jnp.hstack([x,y]))
        )(scene) - jnp.inf * ~mask[...,None]

    # Compute the final interaction summary
    # for the target agent 'x'
    interactions = tf_model(
        jnp.hstack([
            x,
```

```
            jnp.max(
                jnp.vstack([pairwise_interaction(x), i_bias]),
                axis=0
            )
        ])
    )

    return interactions
```

A.3 Probabilistic Module (VAE)

This section provides the implementation of the Variational Autoencoder (VAE) module, which is responsible for creating the probabilistic latent space. The core of this module is a function that takes the final hidden vector from the encoder and outputs a sample from a learned, full-covariance Gaussian distribution.

The first helper function, `vector_to_cholesky_jax`, is a utility that takes a flat vector of parameters from the encoder's VAE head and reshapes it into a lower-triangular Cholesky factor L. This matrix is then used to construct the full covariance matrix $\Sigma = LL\top$ of the posterior distribution.

```
def vector_to_cholesky_jax(vec, d):

    L = jnp.zeros((d, d))
```

```
    # Get indices for the diagonal and lower-triangular parts
    diag_indices = jnp.diag_indices(d)
    tril_indices = jnp.tril_indices(d, k=-1)

    # The first d elements are the log-diagonals,
    # which we exponentiate
    log_diagonals = vec[:d]
    L = L.at[diag_indices].set(jnp.exp(log_diagonals))

    # The remaining elements fill the strictly lower part
    off_diagonals = vec[d:]
    L = L.at[tril_indices].set(off_diagonals)

    return L
```

The main `VAE` function takes the hidden vector from the encoder, partitions it into the mean and Cholesky components, and then uses the reparameterization trick to draw a differentiable sample `tf` from the learned posterior distribution.

```
def VAE(
    hidden,
    vae_size,
    key,
):
```

```
    # Partition the encoder's output vector
    # into mean, log-diagonals, and off-diagonals
    mean, logsd, cov = (
        hidden[:vae_size],
        hidden[vae_size:2*vae_size],
        hidden[2*vae_size:],
    )

    logsd = jnp.clip(logsd, -10, 10)

    # Construct the Cholesky factor L from its components
    cholesky_vec = jnp.concatenate([logsd, cov])
    L = vector_to_cholesky_jax(cholesky_vec, vae_size)

    # Sample from the standard normal distribution
    epsilon = jr.normal(key, (vae_size,))

    # Apply the reparameterization trick: z = mu + L * epsilon
    tf = mean + L @ epsilon

    return tf, mean, jnp.exp(logsd), cov
```

A.4 Decoder Vector Fields and Solvers

This section provides the implementation of the decoder modules. The core of the decoder is the `Jet_Dynamics` class, which defines the neural vector field that is integrated by a Neural ODE. The name is inspired by the mathematical concept of a jet bundle, as the state `y(t)` that is being evolved is a concatenation of the abstract latent code and the physical `(x, y)` coordinates. We present both a static and a dynamic version of the vector field, along with their corresponding `NeuralODE` wrapper classes.

Static Decoder The `Static_Jet_Dynamics` class defines a vector field that is a function of only the agent's own current state `y(t). The StaticNeuralODE` class wraps this vector field and uses the `diffrax.Euler()` solver to integrate it forward in time.

```
class Static_Jet_Dynamics(eqx.Module):
    mlp: eqx.nn.MLP
    latent_linear: eqx.nn.Linear
    main_linear: eqx.nn.Linear

    def __init__(self, data_size, latent_size, depth,
                                *, key, **kwargs):
        super().__init__(**kwargs)
        mlp_key, ll_key, ml_key = jr.split(key, 3)
        self.mlp = eqx.nn.MLP(
            in_size=latent_size + data_size,
```

```
            out_size=latent_size + data_size,
            width_size=2 * (latent_size + data_size),
            depth=depth - 1,
            activation=lipswish,
            final_activation=lipswish,
            key=key,
        )
        self.latent_linear = eqx.nn.Linear(
            latent_size + data_size, latent_size, key=ll_key
        )
        self.main_linear = eqx.nn.Linear(
            latent_size + data_size, data_size, key=ml_key
        )

    def __call__(self, t, y, args):
        x = self.mlp(y)
        return jnp.hstack([self.latent_linear(x),
                self.main_linear(x)])
```

```
class StaticNeuralODE(eqx.Module):
    func: Static_Jet_Dynamics

    def __init__(self, data_size, latent_size, depth, hidden_depth,
                                 *, key, **kwargs):
```

```
        super().__init__(**kwargs)
        self.func = Static_Jet_Dynamics(
        data_size, latent_size, depth, key=key
        )

    def __call__(self, ts, y0, mask):
        solution = diffrax.diffeqsolve(
            diffrax.ODETerm(self.func),
            diffrax.Euler(),
            t0=ts[0],
            t1=ts[-1],
            dt0=2/5,
            y0=y0,
            saveat=diffrax.SaveAt(ts=ts),
        )
        return solution.ys
```

Dynamic Decoder The `Dynamic_Jet_Dynamics` class is the final component of the more complex dynamic vector field. The main logic is contained within the `DynamicNeuralODE` class, which re-evaluates the social context at each step of the ODE solve. Its vector field, `v_interactions`, computes a global summary and individualized interactions from the set of all current agent states `Y(t)`, and then passes the result along with the agent's own state `y(t)` to the `v_compensator` (`Dynamic_Jet_Dynamics` instance) to compute the final velocity.

```
class Dynamic_Jet_Dynamics(eqx.Module):
    mlp: eqx.nn.MLP
    latent_linear: eqx.nn.Linear
    main_linear: eqx.nn.Linear


    def __init__(self, in_size, data_size, latent_size, depth,
        *, key, **kwargs):
        super().__init__(**kwargs)
        mlp_key, ll_key, ml_key = jr.split(key, 3)
        self.mlp = eqx.nn.MLP(
            in_size=in_size,
            out_size=in_size,
            width_size=in_size,
            depth=depth - 1,
            activation=lipswish,
            final_activation=lipswish,
            key=key,
        )
        self.latent_linear = eqx.nn.Linear(
            in_size, latent_size, key=ll_key
        )
        self.main_linear = eqx.nn.Linear(
            in_size, data_size, key=ml_key
        )
```

```
    def __call__(self, t, y, args):
        x = self.mlp(y)
        return jnp.hstack([self.latent_linear(x),
                self.main_linear(x)])
```

```
class DynamicNeuralODE(eqx.Module):
    data_size: int
    latent_size: int
    v_sumr: eqx.nn.MLP
    v_intr: eqx.nn.MLP
    v_compensator: Dynamic_Jet_Dynamics
    v_sumr_bias: jax.Array
    v_intr_bias: jax.Array

    def __init__(
        self,
        data_size,
        latent_size,
        v_sumr_size,
        v_intr_size,
        depth,
        hidden_depth,
        *,
```

```
        key,
        **kwargs
        ):
        super().__init__(**kwargs)
        (v_sumr_key,
        v_sumr_bias_key,
        v_intr_key,
        v_intr_bias_key,
        comp_key) = jr.split(key, 5)
        v_size = data_size + latent_size + v_sumr_size + v_intr_size


        self.data_size = data_size
        self.latent_size = latent_size


        self.v_sumr = eqx.nn.MLP(
            in_size=data_size + latent_size,
            out_size=v_sumr_size,
            width_size=data_size + latent_size + v_sumr_size,
            depth=hidden_depth,
            activation=lipswish,
             key=v_sumr_key,
        )
        self.v_sumr_bias = jr.normal(v_sumr_bias_key, (v_sumr_size,))
        self.v_intr = eqx.nn.MLP(
```

```
            in_size=2 * (data_size + latent_size),
            out_size=v_intr_size,
            width_size=2 * (data_size + latent_size) + v_intr_size,
            depth=hidden_depth,
            activation=lipswish,
            key=v_intr_key,
        )
        self.v_intr_bias = jr.normal(v_intr_bias_key, (v_intr_size,))
        self.v_compensator = Dynamic_Jet_Dynamics(
            v_size, data_size, latent_size, depth, key=comp_key
        )

    def __call__(self, ts, y0, mask, bounds):
        x_min, x_max, y_min, y_max = bounds

        v_mask = lambda t, y: (
            jnp.any(
            mask & \
            jnp.tile(jnp.arange(mask.shape[1]) <= \
            (5/2)*t, (mask.shape[0], 1)), axis=1)
        ) & (
            (y[:, -2] >= x_min) & (x_max >= y[:, -2]) & \
            (y[:, -1] >= y_min) & (y_max >= y[:, -1])
        )
```

```
v_mask_fix_data_size = lambda t, y: jnp.concatenate([
    jnp.ones((mask.shape[0], self.latent_size), dtype=bool),
    jnp.broadcast_to(v_mask(t, y)[..., None],
    (mask.shape[0], self.data_size))
], axis=-1)


v_summary = lambda t, y: jnp.max(jnp.vstack([
    jax.vmap(self.v_sumr)(y) - jnp.inf * ~v_mask(t, y)[:,None],
    self.v_sumr_bias
]),axis=0)


v_interactions = lambda t, y, args: v_mask(t, y)[:,None] *\
 jax.vmap(
    lambda v: self.v_compensator(
        t,
        jnp.hstack([
            v,
            v_summary(t, y),
            jnp.max(jnp.vstack([
                jax.vmap(
                    lambda w: self.v_intr(jnp.hstack([v,w]))
                )(y)- jnp.inf * ~v_mask(t, y)[:,None],
                self.v_intr_bias
```

```
                ]), axis=0)
            ]), args)
    )(y)

    solution = diffrax.diffeqsolve(
        diffrax.ODETerm(v_interactions),
        diffrax.Euler(),
        t0=ts[0],
        t1=ts[-1],
        dt0=2/5,
        y0=y0,
        saveat=diffrax.SaveAt(ts=ts),
    )

    return solution.ys
```

Appendix B: Main Encoder Functions and Implementation of Universality Theorems

This appendix shows how the core modules from Appendix A are assembled into the different end-to-end encoder functions used in our experiments. For each architecture, we also provide a one-to-one correspondence between the theoretical components of the universality theorems and the specific JAX/Equinox modules used in the implementation.

B.1 Benchmark Encoders

The following two encoder functions are used for the fixed-horizon (8-observe, 12-predict) benchmark models. They operate on fully-observed, relative-coordinate data and use the non-adaptive causal convolution.

Agent→Time (Set→Shift) Encoder This encoder (`encoder_8_VAEST.py`) implements the spatial-first processing pathway. It first computes interactions at each of the 8 time steps and then applies a single temporal convolution to the resulting feature sequence.

```
import equinox as eqx  # https://github.com/patrick-kidger/equinox
import jax
import jax.numpy as jnp
```

```
import jax.nn as jnn
import utils


def encoder(models, inputs, sizes, keys):
    temp_model, max_bias, intr_model, tf_model = models
    tf_model_1, tf_model_2 = tf_model
    (data_size,
    latent_size,
    hidden_size_int,
    hidden_size_max
    ) = sizes
    hidden_size = hidden_size_int + hidden_size_max


    mask = jnp.isfinite(inputs[...,0])


    ti = (2/5)*jnp.arange(8)
    shape = inputs.shape
    centered_inputs = jnp.broadcast_to(inputs, (shape[0],) + shape) \
                    - jnp.broadcast_to(inputs[:, 0, None, None, :],
                         (shape[0],) + shape)


    interactions = jax.vmap(jax.vmap(
    utils.Tf_with_intr,
    in_axes = (0, 1, None, None, None, None)),
```

```
                        in_axes = (0, 0, None, None, None, None))(
        inputs,
        centered_inputs,
        mask[:,7],
        max_bias,
        intr_model,
        tf_model_1,
    )

augmented_interactions = jnp.concatenate([
    interactions, inputs[:,:8]-inputs[:,:1],
],axis=-1)

tf_1 = jax.vmap(utils.combined_conv,
                        in_axes = (None, 0, None, None))(
    ti,
    augmented_interactions,
    temp_model,
    hidden_size_int)

hiddens = jax.vmap(tf_model_2)(tf_1)

tf, mean, sd, cov= jax.vmap(utils.VAE,
        in_axes = (0, None, 0))(
```

```
            hiddens,
            latent_size,
            keys,
        )

    tf_n_y_lists = jnp.concatenate([
        tf, inputs[:,7]-inputs[:,0],
    ],axis=-1)

    return tf_n_y_lists, mean, sd, cov
```

Time→Agent (Shift→Set) Encoder This encoder (`encoder_8_VAETS.py`) implements the temporal-first pathway. It applies a temporal convolution to each agent's history independently and then computes interactions on the resulting set of hidden summary vectors.

```
import equinox as eqx
import jax
import jax.numpy as jnp
import jax.nn as jnn
import utils

def encoder(models, inputs, sizes, keys):
    temp_model, max_bias, intr_model, tf_model = models
    (data_size,
```

```
 latent_size,
  hidden_size_int,
   hidden_size_max
   ) = sizes
hidden_size = hidden_size_int + hidden_size_max


mask = jnp.isfinite(inputs[...,0])


ti = (2/5)*jnp.arange(8)
shape = inputs.shape
centered_inputs = jnp.broadcast_to(inputs, (shape[0],) + shape) \
                - jnp.broadcast_to(inputs[:, 0, None, None, :],
                     (shape[0],) + shape)


conv_lists = jax.vmap(jax.vmap(utils.combined_conv,
                                  in_axes = (None, 0, None, None)),
                         in_axes = (None, 0, None, None))(
    ti,
    centered_inputs,
    temp_model,
    hidden_size_int)


diag_conv_lists = conv_lists[jnp.arange(shape[0]),
     jnp.arange(shape[0])]
```

```
    interactions = jax.vmap(utils.Tf_with_intr,
            in_axes = (0, 0, None, None, None, None))(
            diag_conv_lists,
            conv_lists,
            mask[:,7],
            max_bias,
            intr_model,
            tf_model,
        )

    tf, mean, sd, cov= jax.vmap(utils.VAE,
            in_axes = (0, None, 0))(
            interactions,
            latent_size,
            keys,
        )

    tf_n_y_lists = jnp.concatenate([
        tf, inputs[:,7]-inputs[:,0],
    ],axis=-1)

    return tf_n_y_lists, mean, sd, cov
```

B.2 Multi-Horizon Encoder

This encoder (`encoder_double_temp.py`) is the final, most expressive architecture. It implements the Time→Agent→Time pathway, uses the Temporal Conjugation pipeline to handle ragged data in absolute coordinates, and employs two stages of adaptive causal convolutions.

Note on Padded Tensor Dimensions The implementation of this encoder operates on input tensors with a time dimension of 21, even though the trajectory data spans only 20 time steps (indices 0-19) . This 21st timestamp is a technical device added to ensure robust, JIT-compatible processing of ragged batches. It specifically helps manage non-existent agent rows—that is, placeholder rows in a padded batch that consist entirely of `jnp.nan`. Such rows can cause unexpected behavior in vectorized indexing operations. To prevent this, we explicitly set the mask for the final, 21st timestamp to `False` for all agents using `.at[...,-1].set(False)` . This ensures that this final time step is always ignored during interaction calculations, providing a clean and stable way to handle the varying number of agents per scene within a fixed-shape, JIT-compiled function.

```
import equinox as eqx
import jax
import jax.numpy as jnp
import utils


def encoder(models, inputs, sizes, keys):
    temp_models, max_bias, sumr_model, intr_model, tf_models = models
    temp_1, temp_2 = temp_models
```

```
    tf_1, tf_2 = tf_models
    data_size, latent_size, hidden_size_int, hidden_size_max = sizes
    hidden_size = hidden_size_int + hidden_size_max

    # --- Temporal Conjugation: Detect & Align (g) ---
    start_inds = jnp.argmax(jnp.isfinite(inputs[...,0]), axis=-1)
    time_lengths = jnp.sum(jnp.isfinite(inputs[...,0]), axis=-1)

    ti = (2/5)*jnp.arange(21)
    left_aligned_inds = start_inds[..., None] + \
        jnp.cumsum(jnp.arange(21) <= \
        time_lengths[..., None]-1, axis=-1)-1
    left_aligned_1 = jnp.take_along_axis(
        inputs, jnp.tile(left_aligned_inds[:,:,None],(1,1,data_size)),
         axis=-2
    )
    mask = jnp.isfinite(inputs[...,0]).at[...,-1].set(False)

    # --- Process (h): Shift 1 ---
    conv_lists_1 = jax.vmap(
    utils.conv_list, in_axes = (None, 0, None, None)
    )(ti, left_aligned_1, temp_1, hidden_size_int)

    # --- Un-align (g^-1) ---
```

```
aligned_inds = jnp.minimum(
    jnp.maximum(0, jnp.arange(21)-start_inds[...,None]),
    time_lengths[...,None]-1)
aligned_conv_lists_1 = jnp.take_along_axis(
    conv_lists_1,
    jnp.tile(aligned_inds[:,:,None],(1,1,hidden_size)),
    axis=-2
)


# --- Process: Set ---
tf_n_y_lists_1 = jnp.concatenate([
    jnp.transpose(jax.vmap(utils.Tf_with_sumr_intr,
              in_axes = (1, 1, None, None, None, None))\
    (aligned_conv_lists_1, mask, max_bias,
     sumr_model, intr_model, tf_1), (1,0,2)), inputs,
],axis=-1)


# --- Temporal Conjugation: Align (g) for second convolution ---
left_aligned_2 = jnp.take_along_axis(
    tf_n_y_lists_1,
    jnp.tile(left_aligned_inds[:,:,None],(1,1,3*data_size)),
    axis=-2
)
```

```
    # --- Process (h): Shift 2 ---
    conv_lists_2 = jax.vmap(
        utils.conv_list,
        in_axes = (None, 0, None, None)
    )(ti, left_aligned_2, temp_2, data_size)


    # --- Final VAE Head ---
    priors = jax.vmap(jax.vmap(tf_2))(conv_lists_2)


    tf, mean, sd, cov= jax.vmap(
        jax.vmap(
            utils.VAE, in_axes = (0, None, 0)),
        in_axes = (0, None, 0))(
            priors,
            latent_size,
            keys,
        )


    tf_n_y_lists_2 = jnp.concatenate([tf, inputs],axis=-1)


    # --- Un-align final features ---
    tf_n_y_lists_2_no_nan = jnp.take_along_axis(jnp.take_along_axis(
        tf_n_y_lists_2,
        jnp.tile(left_aligned_inds[:,:,None],
```

```
        (1,1,data_size + latent_size)),
        axis=-2
    ), jnp.tile(aligned_inds[:,:,None],(1,1,data_size + latent_size)),
     axis=-2)


    return tf_n_y_lists_2_no_nan, mean, sd, cov
```

B.3 Implementation of the Universality Theorems

This section provides a detailed, one-to-one correspondence between the theoretical components of the universality theorems (from Theorem 3.5.7 and Conjecture 5.1.5) and the specific JAX/Equinox modules used in the final implementations of our architectures. We present a separate table for the simpler static-context models and the more complex dynamic-context model.

Table B.1: Code-level implementation of each theoretical component

Theoretical Component	**VAEST (Agent → Time)**	**VAETS (Time → Agent)**	**Time → Agent → Time (Static)**
Encoder			
CNN_1	`--`	`Causal_net`	`Causal_net_1`
ϕ (pairwise interaction)	`Interaction_net`	`Interaction_net`	`Interaction_net`
max (set pooling)	`jnp.max` inside `Tf_with_intr`	`jnp.max` inside `Tf_with_intr`	`jnp.max` inside `Tf_with_sumr_intr`
CNN_2	`Causal_net`	`--`	`Causal_net_2`
γ (final MLP/VAE head)	`Tf_net_2`	`Tf_net`	`Tf_net_2`
Decoder			
$v(y(t))$ (static-context vector field)	`Static_Jet_Dynamics` in `utils_8_VAEST.py`	`Static_Jet_Dynamics` in `utils_8_VAETS.py`	`Static_Jet_Dynamics` in `utils.py`

Table B.2: Correspondence for the Dynamic-Context Decoder

Theoretical Component	**Code Implementation** (`DynamicNeuralODE`)
$\gamma\big(y(t)\big)$ (Set of all states)	The `y` argument passed to the `v_interactions` vector field
$\phi\big(y_i(t),\, y_j(t)\big)$ (Pairwise function)	`self.v_intr(jnp.hstack([y[idx], y[ind]]))`
$\max_i\ y_i(t)$ (Permutation pooling)	`jnp.max` on the output of the pairwise `vmap`
$\gamma\big(y_i(t), Y(t)\big)$ (Final aggregation)	`self.v_compensator` (a `Dynamic_Jet_Dynamics` instance)

Appendix C: Loss Function Implementation

This appendix provides the JAX implementation of the full, boundary-aware loss function used to train our final multi-horizon models, as described in §7.5.2. The `grad_loss` functions for the simpler VAEST and VAETS benchmark models are omitted, as their core logic—an L_2 reconstruction error combined with a KL-divergence term—is a straightforward subset of the final implementation shown here.

The `grad_loss` function takes a batch of data and the models as input and returns the final training loss. It calculates the three main components: the L_2 path-matching error, the KL-divergence for the VAE, and the anti-lingering penalty for boundary awareness. The function is decorated with `@eqx.filter_value_and_grad` to be JIT-compatible and to efficiently compute both the loss and its gradients in a single pass. The following `grad_loss` function is for the `StaticNeuralODE` decoder

```
import equinox as eqx
import jax
import jax.numpy as jnp
import jax.random as jr


@eqx.filter_value_and_grad
def grad_loss(models, inputs, key, beta):
```

```
encoder_models = models[:-1]

jet_model = models[-1]

keys = jr.split(key, (batch_size, max_agents, 21))

# initial time range for the decoder is a:(21-b)

a, b = init_time_range


# Create a time grid for the full 21-step window

ti = (2/5)*jnp.arange(21)


# Create a boolean mask for valid (non-NaN) data points.

# The final timestamp is explicitly masked out

# as a technical handling step.

mask = jnp.isfinite(inputs[...,0]).at[...,-1].set(False)


# --- 1. Run the Encoder ---

# This runs the full Time->Agent->Time encoder

# on the ragged input data.

# It returns the final encoded state for each agent

# at each possible t0,

# as well as the parameters of the VAE posterior distribution.

aligned_tf_n_y_lists, mean, sd, cov = jax.vmap(

    encoder_double_temp.encoder,

    in_axes = (None, 0, None, 0)

)(encoder_models, inputs, sizes, keys)
```

```
# --- 2. Run the Decoder ---
# Select the initial states (y0)
# for each potential prediction start time t0.
# The complex indexing creates a view of the data
# for all start times in parallel.
latents = aligned_tf_n_y_lists[:,:, a+ jnp.minimum(
    jnp.arange(21-a-b).reshape(-1, 1) + jnp.arange(21-a), 21-a-1)]

# Run the Neural ODE decoder in parallel
# for each potential start time t0.
latents_pred = jax.vmap(jax.vmap(
jax.vmap(jet_model, in_axes=(None, 0, 0)),
    in_axes=(None, 0, 0)), in_axes=(None, 0, 0))(
    ti[a:], latents[:,:,:,0], mask[:,:,a + jnp.minimum(
    jnp.arange(21-a-b).reshape(-1, 1) + jnp.arange(21-a), 21-a-1)])

# --- 3. Compute Loss Components ---

# --- L2 Path-Matching Error ---
# Create a mask for all valid agent-timesteps
# in the prediction windows.
masks = (jnp.tile(mask[:,:,None],(1,1,21-b-a,1)) &\
      jnp.less((a + jnp.arange(21-a-b])).reshape(-1, 1),
```

```
                        jnp.arange(21)))


masks = (masks[...,a]:] & mask[:, :, a:-b, None])[...,None]


# Calculate the deviation
# between predicted and ground truth trajectories.
dev = masks*((latents-latents_pred)\
            [:,:,jnp.minimum(
            jnp.arange(21-a-b)[:,None,None],
            jnp.arange(21-a)[None,:,None]
            ), jnp.maximum(
            0,
            jnp.arange(21-a)[None,:,None]\
            - jnp.arange(21-a-b)[:,None,None]
            ), jnp.arange(data_size + latent_size)[None,None,:]])


# Compute the L2 error on the (x,y) coordinates,
# normalized by the number of valid timesteps.
l2_2_error = jnp.sum(dev[...,-2:]**2)/jnp.sum(masks)


# --- Anti-Lingering Penalty (Boundary Error) ---
# Find the true exit time for each agent.
exit = jnp.sum(jnp.cumprod(jnp.isfinite(
inputs[:,:,a + jnp.minimum(
```

```
    jnp.arange(21-a-b).reshape(-1, 1) \
    + jnp.arange(21-a), 21-a-1),0]),
    axis=-1), axis=-1)


# Calculate the distance to the boundary
# for any predicted point that is lingering
# inside the boundary after the agent's true exit time.
dist_to_boundary = mask[:,:,a:-b,None]*jnp.min(jnp.stack([
    (latents_pred[...,-2]-x_min),
    (x_max-latents_pred[...,-2]),
    (latents_pred[...,-1]-y_min),
    (y_max-latents_pred[...,-1]),
], axis=-1), axis=-1)*\
((jnp.arange(21-a).reshape(1,1,1,-1) > exit[...,None]) &\
 ((jnp.arange(21-a).reshape(1,1,1,-1) < \
 (21-a-b-jnp.arange(21-a-b)).reshape(1, 1, -1, 1))))


linger = jnp.maximum(0,dist_to_boundary)


# --- KL Divergence ---
# Calculate KL divergence terms
#for the full-covariance Gaussian posterior.
diag_kl_term = jnp.sum(mask[...,a:-b, None] *\
     (mean**2 + sd**2 - 2 * jnp.log(sd) - 1)[:,:,a:-b])
```

```
    off_diag_kl_term = jnp.sum(mask[...a:-b, None] * \
        (cov**2)[:,:,a:-b])
    total_kl = 0.5 * (diag_kl_term + off_diag_kl_term)
    normalization_factor = jnp.sum(mask[...,a:-b])

    gaussian_kl = total_kl / normalization_factor

    boundary_error = jnp.sum(linger**2)/normalization_factor

    return l2_2_error + boundary_error + beta*gaussian_kl
```

Dynamic Decoder Loss Code The `grad_loss` function for the dynamic model shares the same structure as the static version. The key differences lie in the decoder call, which now requires the scene's spatial `bounds`, and in how the loss `masks` are constructed to account for late-arriving agents.

The `bounds` variable is a tuple `(x_min, x_max, y_min, y_max)` that defines the overall observable region of the dataset. It is passed to the DynamicNeuralODE so that, at each step of the ODE solve, the internal `v_mask` can dynamically deactivate agents whose predicted trajectory has traveled outside the known area. This allows the model to correctly handle early exit events during the simulation, as agents who leave the scene cease to influence those who remain.

```
latents_pred = jnp.transpose(jax.vmap(
    jax.vmap(jet_model, in_axes=(None, 1, 1, None)),
```

```
    in_axes=(None, 0, 0, None))(
        ti[a:],
        latents[:,:,:,0],
        mask[:,:,a+ jnp.minimum(
        jnp.arange(21-a-b.reshape(-1, 1) + jnp.arange(21-a),
            21-a-1
        )],
        bounds,
    ),(0,3,1,2,4))
```

The corresponding `masks` are also constructed differently to ensure that late-arriving agents are included in the loss calculation from the moment they appear:

```
    masks = (jnp.tile(mask[:,:,None],(1,1,21-a-b,1)) &\
          jnp.less((a + jnp.arange(21-a-b)).reshape(-1, 1),
                           jnp.arange(21)))[...,a:,None]
```

The remainder of the loss calculation then proceeds identically to the static decoder version.

Bibliography

[1] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[2] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022.

[3] Yeping Hu, Xiaogang Jia, Masayoshi Tomizuka, and Wei Zhan. Causal-based time series domain generalization for vehicle intention prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7806–7813. IEEE, 2022.

[4] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting multiple agent trajectories. *arXiv preprint arXiv:2106.08417*, 2021.

[5] Song Wen, Hao Wang, and Dimitris Metaxas. Social ode: Multi-agent trajectory forecasting with neural ordinary differential equations. In *European Conference on Computer Vision*, pages 217–233. Springer, 2022.

[6] Christian Bueno and Alan Hylton. On the representation power of set pooling networks. *Advances in Neural Information Processing Systems*, 34:17170–17182, 2021.

[7] Rossana Capuani, Antonio Marigonda, and Marc Quincampoix. Set-driven evolution for multiagent system. *Journal of Optimization Theory and Applications*, 200(1):280–307, 2024.

[8] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[9] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 1998.

[10] Thomas Lorenz. *Mutational analysis: a joint framework for cauchy problems in and beyond vector spaces*. Springer, 2010.

[11] Filippo Santambrogio. *Optimal transport for applied mathematicians*, volume 87. Springer, 2015.

[12] Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

[13] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[14] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[15] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[18] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[21] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[22] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[23] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[25] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[26] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707, 2020.

[27] Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International conference on machine learning*, pages 5453–5463. PMLR, 2021.

[28] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020.

[29] John P Klein and Melvin L Moeschberger. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media, 2006.

[30] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[31] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12085–12094, 2019.

[32] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12, 2016.

[33] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[34] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *Advances in neural information processing systems*, 28, 2015.

[35] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9813–9823, 2021.

[36] Fan Wen, Ming Li, and Ruiyang Wang. Social transformer: A pedestrian trajectory prediction method based on social feature processing using transformer. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2022.

[37] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[38] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14424–14432, 2020.

[39] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020.

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[41] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[42] Benjamin Kedem and Konstantinos Fokianos. *Regression models for time series analysis*. John Wiley & Sons, 2005.

[43] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264, 2018.

[44] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[45] Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

[46] Yuxuan Liang, Kun Ouyang, Hanshu Yan, Yiwei Wang, Zekun Tong, and Roger Zimmermann. Modeling trajectories with neural ordinary differential equations. In *IJCAI*, pages 1498–1504, 2021.

[47] Bruce W Turnbull. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 38(3):290–295, 1976.

[48] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

[49] Terry Therneau et al. A package for survival analysis in s. *R package version*, 2(7):2014, 2015.

[50] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pages 261–268. IEEE, 2009.

[51] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2375–2384, 2019.

[52] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11:417–487, 2011.

[53] Christian Bueno. *Universal Approximation for Neural Nets on Sets*. PhD thesis, University of California, Santa Barbara, Santa Barbara, CA, 2021. Advisor: Paul J. Atzberger.

[54] Ilya Molchanov and Ilya S Molchanov. *Theory of random sets*, volume 19. Springer, 2005.

[55] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[56] Leonid Kantorovitch. On the translocation of masses. *Management science*, 5(1):1–4, 1958.

[57] Benedetto Piccoli and Francesco Rossi. Generalized wasserstein distance and its application to transport equations with source. *Archive for Rational Mechanics and Analysis*, 211:335–358, 2014.

[58] Benedetto Piccoli and Francesco Rossi. On properties of the generalized wasserstein distance. *Archive for Rational Mechanics and Analysis*, 222:1339–1365, 2016.

[59] Alessio Figalli and Nicola Gigli. A new transportation distance between non-negative measures, with applications to gradients flows with dirichlet boundary conditions. *Journal de mathématiques pures et appliquées*, 94(2):107–130, 2010.

[60] Matthias Liero, Alexander Mielke, and Giuseppe Savaré. Optimal entropy-transport problems and a new hellinger–kantorovich distance between positive measures. *Inventiones mathematicae*, 211(3):969–1117, 2018.

[61] Vassili Nikitich Kolokoltsov and Victor Pavlovich Maslov. General form of endomorphisms in the space of continuous functions with values in a numerical commutative semiring (with the operation $\oplus = \max$). In *Doklady Akademii Nauk*, volume 295, pages 283–287. Russian Academy of Sciences, 1987.

[62] Vassili N Kolokol'tsov and Victor P Maslov. Idempotent analysis as a tool of control theory and optimal synthesis. i. *Functional Analysis and Its Applications*, 23(1):1–11, 1989.

[63] Gerald B Folland. *Real analysis: modern techniques and their applications*. John Wiley & Sons, 1999.

[64] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3rd edition, 1976.

[65] Marshall Harvey Stone. Applications of the theory of boolean rings to general topology. *Transactions of the American Mathematical Society*, 41(3):375–481, 1937.

[66] João B Prolla. *Approximation of vector valued functions*, volume 25. Elsevier, 2011.

[67] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[68] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[69] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[70] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[71] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[72] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[73] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[74] Peter J Brockwell and Richard A Davis. *Time series: theory and methods*. Springer science & business media, 1991.

[75] R. A. Singer. Estimating optimal tracking filter performance for manned maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, 6(4):473–483, 1970.

[76] Yaakov Bar-Shalom and Edison Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975.

[77] Shen-Yong Chen. Kalman filter for robot vision: a survey. *IEEE Transactions on industrial electronics*, 59(11):4409–4420, 2011.

[78] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[79] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[80] Ivan Kolár, Peter W Michor, and Jan Slovák. *Natural operations in differential geometry*. Springer Science & Business Media, 2013.

[81] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in neural information processing systems*, 32, 2019.

[82] Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *Advances in Neural Information Processing Systems*, 33:3952–3963, 2020.

[83] Yang Kuang. *Delay differential equations*. Academic press New York, 1993.

[84] Qunxi Zhu, Yao Guo, and Wei Lin. Neural delay differential equations. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021. Poster presentation.

[85] Xunbi A Ji and Gábor Orosz. Trainable delays in time delay neural networks for learning delayed dynamics. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[86] Jack K Hale and Sjoerd M Verduyn Lunel. *Introduction to functional differential equations*, volume 99. Springer Science & Business Media, 2013.

[87] Wataru Kumagai and Akiyoshi Sannai. Universal approximation theorem for equivariant maps by group cnns. *arXiv preprint arXiv:2012.13882*, 2020.

[88] Alfredo Bellen, Stefano Maset, Marino Zennaro, and Nicola Guglielmi. Recent trends in the numerical solution of retarded functional differential equations. *Acta Numerica*, 18:1–110, 2009.

[89] Ba-ngu Vo, Mahendra Mallick, Yaakov Bar-Shalom, Stefano Coraluppi, Richard Osborne, Ronald Mahler, and Ba-tuong Vo. Multitarget tracking. *Wiley encyclopedia of electrical and electronics engineering*, (2015), 2015.

[90] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[91] Akiyoshi Sannai, Yuuki Takai, and Matthieu Cordonnier. Universal approximations of permutation invariant/equivariant functions by deep neural networks. *arXiv preprint arXiv:1903.01939*, 2019.

[92] Jack K Hale. *Ordinary differential equations*. Courier Corporation, 2009.

[93] Jean-Pierre Aubin and Hèléne Frankowska. *Set-Valued Analysis*. Birkhäuser, 1990.

[94] Jean-Pierre Aubin and Hèléne Frankowska. *Differential inclusions*. Springer, 2009.

[95] Peter E Kloeden and Thomas Lorenz. Nonlocal multi-scale traffic flow models: analysis beyond vector spaces. *Bulletin of Mathematical Sciences*, 6(3):453–514, 2016.

[96] Benedetto Piccoli, Francesco Rossi, and Magali Tournus. A Wasserstein norm for signed measures, with application to non-local transport equation with source term. *Communications in Mathematical Sciences*, 21(5):1279–1301, 2023.

[97] Karine Pichard and Serge Gautier. Equations with delay in metric spaces: The mutational approach. *Numerical Functional Analysis and Optimization*, 21(7-8):917–932, 2000.

[98] Laura Leal-Taixé, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. Learning an image-based motion context for multiple people tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3542–3549, 2014.

[99] Deepak Ganesan, Sylvia Ratnasamy, Hanbiao Wang, and Deborah Estrin. Coping with irregular spatio-temporal sampling in sensor networks. *ACM SIGCOMM Computer Communication Review*, 34(1):125–130, 2004.

[100] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[101] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: Autograd and xla. *Astrophysics Source Code Library*, pages ascl–2111, 2021.

[102] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[103] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *ICLR: international conference on learning representations*, 2015.

[104] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR: international conference on learning representations*, 2019.

[105] Giorgio C Buttazzo. *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer, 1997.

[106] Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 10–21, 2016.

[107] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *European conference on computer vision*, pages 452–465. Springer, 2010.

[108] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

[109] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in neural information processing systems*, 32, 2019.