A RELATED WORK

As previously highlighted in the introduction, several critical bottlenecks restrict universal applications of prompt control algorithms to disease. When it comes to quickly acquiring new skills, meta-learning emerges as an ideal paradigm for achieving rapid mastery in specific scenarios. In terms of data efficiency, both model-based reinforcement learning (MBRL) and meta-learning have the potential to significantly reduce sample complexity.

Neural Temporal Point Process. In the realm of modeling real-world data, the use of constrained models like Multivariate Hawkes Processes (Hawkes, 1971) can often lead to unsatisfactory results due to model misspecification. In recent studies, researchers have started exploring neural network parameterizations of Temporal Point Processes (TPPs) to mitigate these limitations. Common approaches (Du et al., 2016; Mei & Eisner, 2017) involve the employment of recurrent neural networks to evolve a latent state from which the intensity value can be derived. However, this approach falls short in capturing clustered and bursty event sequences, which are prevalent, as it overlooks vital temporal dependencies or necessitates an excessively high sampling rate (Nickel & Le, 2020).

To surmount these challenges, Neural Jump SDEs (Jia & Benson, 2019) and Neural Spatial Temporal Process (NSTT) (Chen et al., 2020a) extend the Neural Ordinary Differential Equation (ODE) framework, facilitating the computation of exact likelihoods for neural TPPs while addressing the limitations of prior methodologies. These two advancements closely align with our dynamic model, and we draw upon their concepts to develop neural excitatory point processes (EPPs) that are governed by an influence matrix. Besides Neural ODE, other streams of research on networked excitatory point processes are also proposed. Hawkes Processes on Large Network (Delattre et al., 2016) extends the construction of multivariate Hawkes processes to encompass a potentially infinite network of counting processes situated on a directed graph. Other approaches of latent structure learning in multivariate point process (Cai et al., 2022; Fang et al., 2023) emphasize accommodating heterogeneous user-specific traits and incorporating both excitatory and inhibitory influences.

Manipulation of Dynamic Processes. The manipulation and control of dynamic processes represent an active area of research. Typically, control policies for temporal process manipulating can be divided into two categories: (1) Gradually introducing exogenous ***event interventions*** into the existing historical events, and (2) Promptly enforcing ***network interventions*** to the influence matrix between different types of events. Currently, most research is centered around the first type of intervention, primarily focusing on low-frequency and low-dimensional event interventions within social media datasets. For example, techniques like dynamic programming (Farajtabar et al., 2014; 2017) and stochastic control on SDE (Wang et al., 2018) with a closed feedback loop are utilized to steer user activities in social media platforms. However, the number of event types in the above work is limited to derive the closed-form solution. Modern Reinforcement Learning approaches, including both model-free (Upadhyay et al., 2018) and model-based RL (Qu et al., 2023), are proposed to mitigate fake news events on social media. Notably, the *event-intervention-based* approach will fail to generalize to high frequency and uncontrollable event data like newly infested disease cases and incoming traffic.

On the other hand, the problem of *network intervention*, particularly node manipulation (e.g., vaccination) to control epidemic processes on graphs has received extensive attention (Hoffmann et al., 2020; Medlock & Galvani, 2009). Most previous studies have adopted a static setup and made a single decision. In recent work (Meirom et al., 2021), the agent performs sequential decision-making to progressively control graph dynamics through node interventions, demonstrating effectiveness in slowing the spread of infections among different individuals. While existing *network-intervention-based* approaches offer a promising solution for individual-level quarantine in pandemic control, they do not inherently adapt to county- or state-level control, which necessitates more complex node status considerations than those assumed in (Meirom et al., 2021), as well as a larger search space incorporating edge interventions. Moreover, it's worth noting that our approach is related to, but more comprehensive than, epidemic control problems, as it accommodates various data distributions, including Poisson, within excitatory point processes.

Model-based Reinforcement Learning. The key to applications within a Reinforcement Learning (RL) framework lies in enhancing sample efficiency, with Model-Based Reinforcement Learning (MBRL) serving the role of approximating a target environment for the agent's interaction. In environments characterized by unknown dynamics, MBRL can either learn a deterministic mapping

or a distribution of state transitions, denoted as $p(\Delta s|[s, a])$. Typically, modeling uncertainty in dynamic systems involves the evolution of a hidden unit to represent a dynamic world model. Several auto-regressive neural network structures are prevalent in this domain, including well-known models such as World Models Ha & Schmidhuber (2018), Decision Transformer (Chen et al., 2021), and the Dreamer family (Hafner et al., 2019; 2023). On the other hand, the integration of Neural Networks with Model Predictive Control (MPC) (Nagabandi et al., 2018) has achieved excellent sample complexity within model-based reinforcement learning algorithms. Our approach closely aligns with MPC and MBRL techniques, wherein MBRL methods, such as Gradient Descent, can be leveraged to refine or adapt the model utilized by MPC. This adaptation holds the potential to enhance performance, especially in scenarios where system dynamics are non-linear, partially unknown, or subject to change.

Meta Reinforcement Learning. The majority of meta Reinforcement Learning (RL) algorithms adhere to a model-free approach and introduce task-specific variational parameters to facilitate learning across various simple locomotion control tasks. Examples of such algorithms include MAESN (Gupta et al., 2018) and PEARL (Rakelly et al., 2019). Simultaneously, there has been notable success in recent times by incorporating the inherent sequential structure of off-policy control into the representation learning process, as demonstrated in works like CURL (Laskin et al., 2020), Sensory (Tang & Ha, 2021), and PSM (Agarwal et al., 2021) particularly when dealing with more complex tasks such as those found in the Distracting DM Control Suite (Stone et al., 2021).

In scenarios where data is limited, such as in disease control, researchers are increasingly focusing on Model-Based Meta Reinforcement Learning (MBMRL), with a specific emphasis on achieving *fast adaptation* within dynamics models. Approaches like AdMRL (Lin et al., 2020) and Amortized Meta Model-based Policy Search (AMBPS) (Wang & Van Hoof, 2022) involve the optimization and inference of task-specific policies within a parameterized family of tasks, often containing parameters related to positions and velocities. Importantly, a key distinction between our method and AMBPS lies in our utilization of network embeddings and the incorporation of inductive bias to facilitate the learning of a meta-policy without parameterizing individual tasks.

B Limitations

It should be noted that our approach assumes that decisions made by considering a receding horizon window size of T for each node provide a reasonably good approximation of the optimal policy. However, it is important to acknowledge that if long-range correlations exist, this approximation may result in decreased performance. Consequently, an intriguing question arises regarding the ability of our approach to effectively tackle the problem of network interventions within long-range correlated data distributions under the proposed training protocol.

C A Comparison Between Neural ODE and Neural Process

In this section, we conduct a comparative analysis of model performance and efficiency between Neural Ordinary Differential Equations (Neural ODE or NODE) (Chen et al., 2018) and Neural Process (NP) (Garnelo et al., 2018). To evaluate these models, we utilized a synthetic dataset consisting of 5-dimensional Multi-Hyperparameter (MHP) data, with each dimension containing 100 sample points in the training set. The results of this comparison are visually presented in Fig. 10.

Our observations reveal notable distinctions between the learned behaviors of NODE and the Neural Process model. Specifically, the curve learned by NODE exhibits greater diversity, as depicted in the middle panel of Fig. 10. An important metric to consider is the log-likelihood, where we find that Neural ODE achieves a log-likelihood of -33, significantly outperforming Neural Process with a log-likelihood of -180.

It's essential to consider the computational cost associated with each model. For Neural ODE, a significant portion of the computational burden arises from the numerical integration process itself. The runtime complexity of this integration process is denoted as $O(\text{NFE})$, where NFE represents the number of function evaluations. The worst-case scenario for NFE depends on two factors: the minimum step size δ required by the ODE solver and the maximum integration time of interest, denoted as Δt_{max}.

In contrast, the runtime complexity of Neural Process, which employs n context points and m target points, is represented as $O(n + m)$. In our specific experiment, the number of total function

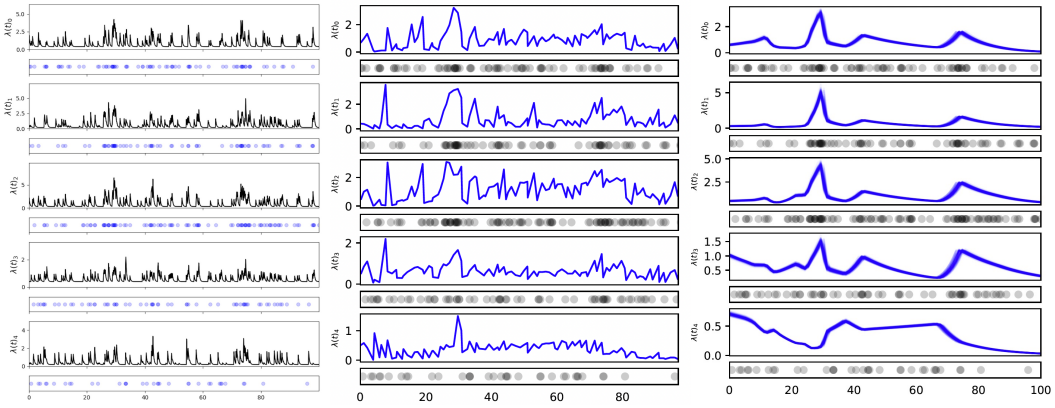

Figure 10: Temporal Point Process Model learned by different methods. **Left**: ground truth intensity function. **Middle**: Learned intensity function plot by Neural Jump ODE (Log-Likelihood: -33). **Right**: Learned intensity function plot by Neural Process. (Log-Likelihood: -183)

evaluations (NFE) for NODE was approximately 5000, while for Neural Process, we selected 50 context points and 100 target points.

It is worth noting that the integration steps, as quantified by $\frac{\Delta t_{max}}{\delta}$, can potentially result in a large constant factor, which may be hidden within the big-O notation. However, it is reassuring to acknowledge that modern ODE solvers, such as *dorpi5*, are designed with adaptive step size mechanisms that adjust dynamically to the supplied data. This adaptive behavior mitigates concerns related to the scalability of the integration process with respect to dataset size and complexity.

D Technical Details of a Probabilistic Networked Model

Table 3: Table of conditional spike count distributions, their parameterizations, and their properties

Distribution	$p(x\|\psi,\upsilon)$	$\mathbb{E}(x)$	Var(x)
Poi($exp(\psi)$)	$\frac{exp(-exp(\psi))(exp(\psi))^x}{x!}$	$exp(\psi)$	$exp(\psi)$
Bern($\sigma(\psi)$)	$\sigma(\psi)^x\sigma(-\psi)^{1-x}$	$\sigma(\psi)$	$\sigma(\psi)\sigma(-\psi)$
Bin($\upsilon,\sigma(\psi)$)	$\binom{\upsilon}{x}\sigma(\psi)^x\sigma(-\psi)^{\upsilon-x}$	$\upsilon\sigma(\psi)$	$\upsilon\sigma(\psi)\sigma(-\psi)$
NB($\upsilon,\sigma(\psi)$)	$\binom{\upsilon+x-1}{x}\sigma(\psi)^x\sigma(-\psi)^{\upsilon}$	$\upsilon\cdot exp(\psi)$	$\upsilon exp(\psi)/\sigma(-\psi)$

E Technical Details of Mean Field Approximation for Reward Model

We begin by defining some notation which will be used throughout these results

- Spectrum of a nonlinear operator f: $L_f = ||f||_{Lip} := \sup_{x\neq y}\frac{||f(x)-f(y)||}{||x-y||} < \infty$

E.1 Theoretical Justification

Given event sequences $\mathcal{H}^{t-} := \{(t_i, n_i)\}_{t_i<t}$, we use $\mathbf{x}^t = \{x_1^t, \ldots, x_N^t\}$ to record the incremental number of events for different nodes starting from within $[t + \Delta t)$. We consider a stochastic dynamic system based on the nonlinear system (2-4) presented in the manuscript:

$$\begin{cases} & \mathbf{h}_n^{t_0} = \mathbf{h}_n^0, \\ & \frac{d\mathbf{h}_n^t}{dt} = f(\mathbf{h}_n^t), \\ \lim_{\epsilon\to 0} & \mathbf{h}^{t_i+\epsilon} = \sum_{m\in\mathcal{N}_n} w_{m\to n}\cdot\phi(\mathbf{h}_m^{t_i}, x_m^{t_i}), \end{cases} \tag{12}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ and $\phi : \mathbb{R}^d \to \mathbb{R}$ are two Lipschitz functions. The emission/intensity function for the dynamic is defined as $\Lambda(t|\mathbf{h}^{t-}) := g_\Lambda(\mathbf{h}^{t-}\cdot\mathbf{w}) = g_{\Lambda,\mathbf{w}}(\mathbf{h}^{t-})$ where $g_{\Lambda,\mathbf{w}} : \mathbb{R}^{N\times d} \to \mathbb{R}^N$ is a Lipschitz activation function, $\mathbf{w} \in \mathbb{R}^d$ is a linear layer, and $\mathbf{h}^{t-} \in \mathbb{R}^{N\times d}$ is the left continuous points before jump. We use λ_m^t to denote m-th element of $\Lambda(t)$ and thus $\mathbf{x}_m^{t_i} \sim \text{Poi}(\lambda_m(t_i|\mathbf{h}_n^{t_i-}))$ is the number of incremental events in $[t_i, t_i + \Delta t)$ for node m. The ground truth cumulative cost $J(t)$

at finite horizon time t is given by the Monte Carlo Estimator, i.e.,

$$J(t) := \sum_{n=1}^{N} \sum_{i=1}^{t} \mathbb{E}[x_n^i]. \tag{13}$$

Instead of extensively performing Monte Carlo simulation to obtain the cost estimator, we introduce a ***Mean Field Estimator*** $\hat{J}(t)$ for $J(t)$ by averaging out the interaction effect by $x_m^{t_i}$ in the dynamic system (12), i.e.,

$$\hat{J}(t) := \sum_{n=1}^{N} \sum_{i=1}^{t} \hat{\lambda}_n(i). \tag{14}$$

where $\hat{\lambda}_n(t) := \lambda_n(t|\hat{\mathbf{h}}_n^{t-})$ and $\hat{\mathbf{h}}_n^{t-}$ is the mean field estimator for $\mathbf{h}_n^{t-}$ and is derived from a deterministic dynamic by replacing the stochastic discrete update term in system (12) with $\phi(\mathbf{h}_m^{t_i}, \hat{\lambda}_m^{t_i})$. We let the starting point of $\hat{\mathbf{h}}_n^{t-}$ be the same point as $\mathbf{h}_n^{t-}$, i.e., $\hat{\mathbf{h}}_n^0 = \mathbf{h}_n^0, \forall n \in \{1, 2, \cdots, N\}$.

Proposition 1. *Given f and g are Lipschitz in Sys. (12), let $\mathbf{h}_n^{t_i^-}$ be the left continuous point of $\mathbf{h}_n^{t_i}$ in the Sys. (12), then it can be recursively expressed by a **Lipschitz** operator $\mathcal{T}_n : \mathbb{R}^{N\times d} \times \mathbb{R}^N \to \mathbb{R}^d$:*

$$\mathbf{h}_n^{t_i^-} = \mathcal{T}_n(\mathbf{h}^{t_{i-1}^-}, \mathbf{x}^{t_{i-1}}) \tag{15}$$

where $\mathbf{h}^{t_{i-1}^-} \in \mathbb{R}^{N\times d}$, and $\mathbf{x}^{t_{i-1}} \in \mathbb{R}^N$. More importantly, let λ_{max} be the maximum spectrum of influence matrix $\mathbf{W}$, when L_f, L_ϕ, and λ_w are smaller than 1, we have $L_{\mathcal{T}_n} < 1$.

Proof. We define $\Phi(\mathbf{h}^{t_i}, \mathbf{x}^{t_i}) := [\phi(\mathbf{h}_1^{t_i}, \mathbf{x}_1^{t_i}), \phi(\mathbf{h}_2^{t_i}, \mathbf{x}_2^{t_i}), \cdots, \phi(\mathbf{h}_N^{t_i}, \mathbf{x}_N^{t_i})]^T \in \mathbb{R}^{N\times d}$ as the discrete kernel in System (12), we can represent $\mathbf{h}_n^{t_i^-}$ as:

$$\mathbf{h}_n^{t_i^-} = \mathcal{T}_n(\mathbf{h}^{t_{i-1}^-}, \mathbf{x}^{t_{i-1}}) = \mathbf{w}_n^T \Phi(\mathbf{h}^{t_{i-1}^-}, \mathbf{x}^{t_{i-1}}) + \int_{t_{i-1}^+}^{t_i^-} f(\mathbf{h}_n^s)\, ds,$$

where $\mathbf{w}^T$ is the n-th row of the influence matrix $\mathbf{W}$. Since f and ϕ are Lipschitz and the composition of Lipschitz functions is also Lipschitz, so $\mathcal{T}_n$ is Lipschitz. Since $L_{\mathcal{T}} \le \lambda_{max} \cdot (L_f)^m \cdot (L_\phi)^N$ where m is the number of summation in the intergral term, we have $L_{\mathcal{T}_n} < 1$ when L_f, L_ϕ, and λ_w are smaller than 1.

□

Proposition 2. *Given f and g are Lipschitz in Sys. (12), let $\hat{\mathbf{h}}_n^{t_i^-}$ be the left continuous point of $\hat{\mathbf{h}}_n^{t_i}$ in the deterministic version of the presented Sys. (12) by replacing $x_m^{t_i}$ with $\hat{\lambda}_m(t_i)$, then it can be recursively expressed by a **Lipschitz** operator $\mathcal{T}_n : \mathbb{R}^{N\times d} \times \mathbb{R}^N \to \mathbb{R}^d$:*

$$\hat{\mathbf{h}}_n^{t_i^-} = \mathcal{T}_n(\mathbf{h}^{t_{i-1}^-}, \hat{\Lambda}(t_{i-1})) \tag{16}$$

where $\hat{\Lambda}(t_{i-1}) = [\hat{\lambda}_1(t_{i-1}), \hat{\lambda}_2(t_{i-1}), \cdots, \hat{\lambda}_N(t_{i-1})]$.

Proof. This is a direct result from Prop. 1. □

Lemma 1. *Let $\hat{\mathbf{h}}_n^i$ be the mean field estimator for $\mathbf{h}_n^i$, suppose f and ϕ are Lipshitz , and $\forall n, t,\ \lambda_n(t)$ is bounded by K, we have:*

$$\mathbb{E}_{\mathbf{h}_n^{i-}}\Big[||\mathbf{h}_n^{i-} - \hat{\mathbf{h}}_n^{i-}||\Big] \le L_{\mathcal{T}_n} \mathbb{E}_{\mathbf{h}_n^{i-1-}}\Big[||\mathbf{h}_n^{i-1-} - \hat{\mathbf{h}}_n^{i-1-}||\Big] + M_n, \tag{17}$$

where $M_n = L_{\mathcal{T}_n}(K^{1/2} + K)$. Moreover, when $L_f < 1$ and $L_\phi < 1$, we have:

$$\mathbb{E}\Big[||\mathbf{h}_n^i - \hat{\mathbf{h}}_n^i||\Big] \le \Big(1 - (L_{\mathcal{T}_n})^i\Big) \cdot \frac{M_n}{1 - L_{\mathcal{T}_n}}. \tag{18}$$

Proof. For simplicity, we use $\mathbf{h}_n^i$ to denote $\mathbf{h}_n^{i-}$ and $\hat{\mathbf{h}}_n^i$ to denote $\hat{\mathbf{h}}_n^{i-}$. Here we prove the first inequality: By Prop. 1 and Prop. 2, we can decompose $\mathbf{h}_n^i$ and $\hat{\mathbf{h}}_n^i$, i.e.,

$$\mathbb{E}_{\mathbf{h}_n^i}\left[||\mathbf{h}_n^i - \hat{\mathbf{h}}_n^i||\right] = \mathbb{E}_{\mathbf{h}_n^{i-1},\mathbf{x}^{i-1}}\left[||\mathcal{T}_n(\mathbf{h}^{i-1},\mathbf{x}^{i-1}) - \mathcal{T}_n(\hat{\mathbf{h}}^{i-1},\hat{\Lambda}^{i-1})||\right] \tag{19}$$

$$\leq \mathbb{E}_{\mathbf{h}_n^{i-1},\mathbf{x}^{i-1}}\left[L_{\mathcal{T}_n}||(\mathbf{h}^{i-1},\mathbf{x}^{i-1}) - (\hat{\mathbf{h}}^{i-1},\hat{\Lambda}^{i-1})||\right] \tag{20}$$

$$= \mathbb{E}_{\mathbf{h}_n^{i-1},\mathbf{x}^{i-1}}\left[L_{\mathcal{T}_n}\sqrt{||\mathbf{h}^{i-1} - \hat{\mathbf{h}}^{i-1}||_{\mathbf{h}}^2 + ||\mathbf{x}^{i-1} - \hat{\Lambda}^{i-1}||_{\mathbf{x}}^2}\right] \tag{21}$$

$$\leq \mathbb{E}_{\mathbf{h}_n^{i-1},\mathbf{x}^{i-1}}\left[L_{\mathcal{T}_n}\left[||\mathbf{h}^{i-1} - \hat{\mathbf{h}}^{i-1}||_{\mathbf{h}} + ||\mathbf{x}^{i-1} - \hat{\Lambda}^{i-1}||_{\mathbf{x}}\right]\right] \tag{22}$$

$$= \mathbb{E}_{\mathbf{h}_n^{i-1},\mathbf{x}^{i-1}}\left[L_{\mathcal{T}_n}\left[||\mathbf{h}^{i-1} - \hat{\mathbf{h}}^{i-1}||_{\mathbf{h}} + ||\mathbf{x}^{i-1} - \Lambda^{i-1} + \Lambda^{i-1} - \hat{\Lambda}^{i-1}||_{\mathbf{x}}\right]\right] \tag{23}$$

$$\leq L_{\mathcal{T}_n}\mathbb{E}_{\mathbf{h}_n^{i-1}}\left[||\mathbf{h}^{i-1} - \hat{\mathbf{h}}^{i-1}||\right] + \mathbb{E}_{\mathbf{x}^{i-1}}\left[L_{\mathcal{T}_n}||\mathbf{x}^{i-1} - \Lambda^{i-1}||\right] + L_{\mathcal{T}_n}||\Lambda^{i-1} - \hat{\Lambda}^{i-1}|| \tag{24}$$

$$= L_{\mathcal{T}_n}\mathbb{E}_{\mathbf{h}_n^{i-1}}\left[||\mathbf{h}^{i-1} - \hat{\mathbf{h}}^{i-1}||\right] + L_{\mathcal{T}_n}\mathbb{E}_{\mathbf{x}^{i-1}}\left[||\mathbf{x}^{i-1} - \Lambda^{i-1}||\right] + L_{\mathcal{T}_n}||\Lambda^{i-1} - \hat{\Lambda}^{i-1}|| \tag{25}$$

$$\leq L_{\mathcal{T}_n}\mathbb{E}_{\mathbf{h}_n^{i-1}}\left[||\mathbf{h}^{i-1} - \hat{\mathbf{h}}^{i-1}||\right] + L_{\mathcal{T}_n}\cdot(K^{1/2} + K) \tag{26}$$

$$= L_{\mathcal{T}_n}\mathbb{E}_{\mathbf{h}_n^{i-1}}\left[||\mathbf{h}^{i-1} - \hat{\mathbf{h}}^{i-1}||\right] + M_n \tag{27}$$

where the second inequality follows by the Lipschitz property of $\mathcal{T}_n$, the forth inequality follows triangular inequality, and the last inequality follows by Jensen's inequality, i.e., $\left(\mathbb{E}_{\mathbf{x}^{i-1}}\left[||\mathbf{x}^{i-1} - \Lambda^{i-1}||\right]\right)^2 \leq \mathbb{E}_{\mathbf{x}^{i-1}}\left[||\mathbf{x}^{i-1} - \Lambda^{i-1}||^2\right] = \Lambda^{i-1} \leq K$ and $||\Lambda^{i-1} - \hat{\Lambda}^{i-1}|| \leq ||\Lambda^{i-1}|| \leq K$ □

Proof. Here we prove the second inequality:

From Eq. (17), we can derive its fixed point $x^* := \mathbb{E}_{\mathbf{h}_n^*}\left[||\mathbf{h}_n^* - \hat{\mathbf{h}}_n^*||\right]$, i.e.,

$$x^* = L_{\mathcal{T}_n}x^* + M_n \tag{28}$$

$$x^* = \frac{M_n}{1 - L_{\mathcal{T}_n}} \tag{29}$$

Since $L_{\mathcal{T}_n} < 1$ given $L_f < 1$ and $L_g < 1$, Eq. (17) is a contraction and thus starting from any arbitrary point x_k will converge to this fixed point x^*. So we have:

$$\mathbb{E}_{\mathbf{h}_n^i}\left[||\mathbf{h}_n^i - \hat{\mathbf{h}}_n^i||\right] - \frac{M_n}{1 - L_{\mathcal{T}_n}} \leq L_{\mathcal{T}_n}\left(\mathbb{E}_{\mathbf{h}_n^{i-1}}\left[||\mathbf{h}_n^{i-1} - \hat{\mathbf{h}}_n^{i-1}||\right] - \frac{M_n}{1 - L_{\mathcal{T}_n}}\right) \tag{30}$$

$$\leq (L_{\mathcal{T}_n})^i\left(\mathbb{E}_{\mathbf{h}_n^0}\left[||\mathbf{h}_n^0 - \hat{\mathbf{h}}_n^0||\right] - \frac{M_n}{1 - L_{\mathcal{T}_n}}\right), \tag{31}$$

$$\mathbb{E}_{\mathbf{h}_n^i}\left[||\mathbf{h}_n^i - \hat{\mathbf{h}}_n^i||\right] \leq \left(1 - (L_{\mathcal{T}_n})^i\right)\cdot\frac{M_n}{1 - L_{\mathcal{T}_n}}, \tag{32}$$

where the last inequality follows by $\mathbf{h}_n^0 = \hat{\mathbf{h}}_n^0$. Thus $\mathbb{E}_{\mathbf{h}_n^i}\left[||\mathbf{h}_n^i - \hat{\mathbf{h}}_n^i||\right]$ is bounded. □

Theorem 1 (Error Bound of Mean Field Estimator). *Let $J(t)$ and $\hat{J}(t)$ be the Monte Carlo Estimator and Mean Field Estimator defined in Eq. (13) and (14). Suppose we have N nodes. When satisfying:*

1. *The dynamic is Lipschitz, i.e., f and ϕ are Lipschitz. And the Lipschitz constants for f and ϕ are smaller than 1, i.e., $L_f < 1$ and $L_\phi < 1$,*
2. *The maximum spectrum $\lambda_{\mathbf{w}}$ of the influence matrix $\mathbf{W}$ is smaller than 1,*
3. *For any $n = \{1, 2, \cdots, N\}$, the intensity function $g_{\lambda_n,\mathbf{w}}$ is Lipschitz and is bounded above by K.*

Then we have:

$$|J(t) - \hat{J}(t)| \leq N(t+1)\cdot L_{g_{\lambda_n,\mathbf{w}}}\cdot\frac{\max_n M_n}{1 - \max_n(L_{\mathcal{T}_n})} \tag{33}$$

where $M_n = L_{\mathcal{T}_n}(K^{1/2} + K)$ *and* $\mathcal{T}_n : \mathbb{R}^{N\times d} \times \mathbb{R}^N \to \mathbb{R}^d$ *is the composite transition function in Proposition 1.*

Proof.

$$|J(t) - \hat{J}(t)| = \left| \sum_{n=1}^{N}\sum_{i=0}^{t} \mathbb{E}[x_n^i] - \sum_{n=1}^{N}\sum_{i=0}^{t} \hat{\lambda}_n(i) \right| \tag{34}$$

$$= \left| \sum_{n=1}^{N}\sum_{i=0}^{t} \mathbb{E}_{\mathbf{h}_n^i}\mathbb{E}_{x_n^i}[x_n^i|\mathbf{h}_n^i] - \sum_{n=1}^{N}\sum_{i=0}^{t} g_{\lambda_n,\mathbf{w}}(\hat{\mathbf{h}}_n^i) \right| \tag{35}$$

$$= \left| \sum_{n=1}^{N}\sum_{i=0}^{t} \mathbb{E}_{\mathbf{h}_n^i} g_{\lambda_n,\mathbf{w}}(\mathbf{h}_n^i) - \sum_{n=1}^{N}\sum_{i=0}^{t} g_{\lambda_n,\mathbf{w}}(\hat{\mathbf{h}}_n^i) \right| \tag{36}$$

$$= \left| \sum_{n=1}^{N}\sum_{i=0}^{t} \mathbb{E}_{\mathbf{h}_n^i} \left[g_{\lambda_n,\mathbf{w}}(\mathbf{h}_n^i) - g_{\lambda_n,\mathbf{w}}(\hat{\mathbf{h}}_n^i) \right] \right| \tag{37}$$

$$\leq \sum_{n=1}^{N}\sum_{i=0}^{t} \mathbb{E}_{\mathbf{h}_n^i} \left[|g_{\lambda_n,\mathbf{w}}(\mathbf{h}_n^i) - g_{\lambda_n,\mathbf{w}}(\hat{\mathbf{h}}_n^i)| \right] \tag{38}$$

$$\leq \sum_{n=1}^{N}\sum_{i=0}^{t} \mathbb{E}_{\mathbf{h}_n^i} \left[L_{g_{\lambda_n,\mathbf{w}}} ||\mathbf{h}_n^i - \hat{\mathbf{h}}_n^i|| \right] \tag{39}$$

Apply Lemma 1, we have,

$$\leq \sum_{n=1}^{N}\sum_{i=0}^{t} L_{g_{\lambda_n,\mathbf{w}}} \cdot \left(1 - (L_{\mathcal{T}_n})^i\right) \cdot \frac{M_n}{1 - L_{\mathcal{T}_n}} \tag{40}$$

$$\leq \sum_{n=1}^{N} (t+1) \cdot L_{g_{\lambda_n,\mathbf{w}}} \cdot \frac{M_n}{1 - L_{\mathcal{T}_n}} \tag{41}$$

$$\leq N(t+1) \cdot L_{g_{\lambda_n,\mathbf{w}}} \cdot \frac{\max_n M_n}{1 - \max_n(L_{\mathcal{T}_n})} \tag{42}$$

□

E.2 Empirical Evaluation

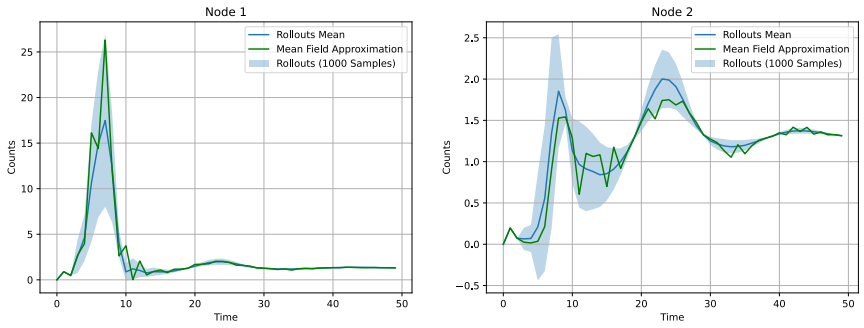

Figure 11: Empirical Evaluation of Mean Field Approximation

We consider the setting of a simplified linear dynamic model based on the nonlinear system (2-4) presented in the manuscript. Specifically, the nonlinear ODE drift term $f(\mathbf{h}_n^t)$ is replaced by a linear function $\mathbf{A}\mathbf{h}_n^t + \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{d\times d}$ and $\mathbf{b} \in \mathbb{R}^d$. The nonlinear discrete jump kernel $\phi(\mathbf{h}_m^{t_i}, x_m^{t_i})$ is replaced by a linear kernel $C\mathbf{h}_m^{t_i} + Dx_m^{t_i}$, where $C \in \mathbb{R}$ represents history forgetness and $D \in \mathbb{R}$ is the scaling factor for the new events. Mathematically, the simplified linear dynamic is given in System (43).

$$\begin{cases} & \mathbf{h}_n^{t_0} = \mathbf{h}_n^0, \\ & \frac{d\mathbf{h}_n^t}{dt} = \mathbf{A}\mathbf{h}_n^t + \mathbf{b}, \\ \lim_{\epsilon\to 0} & \mathbf{h}^{t_i+\epsilon} = \sum_{m\in\mathcal{N}_n} w_{m\to n}(C\mathbf{h}_m^{t_i} + Dx_m^{t_i}). \end{cases} \tag{43}$$

Empirically, we performed a toy experiment on the simplified 2-D linear dynamic and we found the mean field approximator provides an efficiency and accurate approximation for the rollout means (Fig. 11).

F Difference between SimCLR and the proposed method

Given a random sampled minibatch of N examples and the SimCLR contrastive prediction task is defined on pairs of augmented examples derived from the minibatch, resulting in $2N$ data points. SimCLR treats the other $2(N-1)$ augmented examples within a minibatch as negative examples. Then the loss function for a positive pair of examples (i, j) is defined as,

$$l_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \tag{44}$$

where $\mathbf{z}_i$ is the projected embedding from augmented samples.

Compared with SimCLR, our proposed loss does not require sampling additional batch data as negative examples and projecting the augmented sample to one embedding, instead, we only use one anchor sample (anchor network) and project the sample to two different embeddings (p and m), and augment the anchor sample by permuting its node orders or adding noise to the node embeddings such that it can form two negative pairs naturally (as depicted in Fig. 1). Moreover, we augment our network on its latent embedding space by utilizing the graph augmentation techniques, while SimCLR directly performs augmentations on the original visual representations.

G Architecture and Hyperparamters

For the dynamic model, we parameterized the ODE forward function f_h in Eq. (3) as a Time-dependent multilayer perception (MLP) with dimensions [64-64-64]. We used the Softplus activation function. We first attempted to use MLP to parameterize the instantaneous jump function ϕ_h in Eq. (4); however, it led to unstable results for long sequences. Thus we switched to the GRU parameterization, which takes an input, the latent state $\mathbf{h}^{t_i}$, and outputs a new latent state. Importantly, we found letting f_h and ϕ_h share all the parameters across different node trajectories will also lead to the failure of capturing the mode diversity. We therefore created an independent biased term added between the MLP layers in f_h to compensate for diversity loss. Lastly, we use MLP to parameterize the emission function g_λ.

We initialized all Neural ODEs (for the hidden state) with zero drift by initializing the weights and biases of the final layer to zero. All integrals were solved using (Chen et al., 2018) to within a relative and absolute tolerance of 1E-4 or 1E-6, chosen based on preliminary testing for convergence and stability. We also use Seminorms (Kidger et al., 2021) to accelerate neural ODE learning and apply temporal regularization (Ghosh et al., 2020) to mitigate the effect of stiff ODE systems.

For the policy model, we parameterized the policy network by 4 heads, and 128 d-model Transformer layers. For the synthetic dataset, we used a 2-layer transformer for representation learning and another 2-layer transformer for policy generation. For the real-world dataset, we used a 4-layer transformer for representation learning and another 4-layer transformer for policy generation.

For the Policy Equivalent Metric presented in Definition 2, we learned a value function parameterized by a 2-layer transformer by optimizing Least Square Temporal Difference (LSTD).

We trained the dynamic, policy, and PEM-value network by using the ADAM optimizer with a 1E-2 decay rate across 4 RTX3090 GPUs. The initial learning rates for dynamic learning, policy learning, and PEM-value function learning were set to be 1E-3, 1E-4, and 1E-4 respectively.

H Additional Details of Synthetic Data Experiment

H.1 Dataset Setup

We generated synthetic networked point process data by simulating Multivariate Hawkes processes (MHP), which are doubly stochastic point processes with self-excitations (Hawkes, 1971). Specifically, the underlying ground truth influence matrix $\mathbf{W}$ was generated with $n = 10$ nodes and the weights were set as $w_{ij} \sim \mathcal{U}[0, 0.5]$. We set the graph sparsity to 0.1, *i.e.*, each edge is kept with probability 0.1. The generated influence matrix was adjusted appropriately so that its maximum spectral radius was smaller than one, ensuring the stability of the process. We further set the Hawkes kernel to be an exponential basis kernel, where the parameter was set to $\beta = 4$, meaning roughly

losing 98% of influence after one unit of time. The simulation of MHP was based on a thinning algorithm (Ogata, 1981) on a $T = 100$ horizon.

H.2 Additional Results and Figures

We additionally trained the amortized policy on a synthetic star graph and a circular graph with different ground-truth weight matrices and tested the performance on a new star or cycle graph with random weights. The results are shown in Fig. 12. Surprisingly, we find the amortized policy only slightly outperforms the non-amortized on both environments. We conjecture this is because we are adapting the amortized policy to a small local region (only 10 nodes) in this experiment so that the non-amortized policy already can achieve relatively good results.

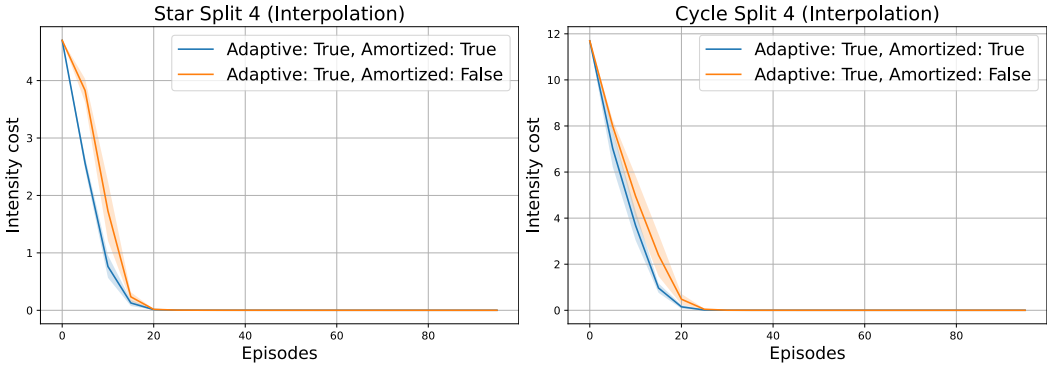

Figure 12: Generalization Results over synthetic data

I Additional Details of Covid Data Experiment

I.1 Dataset Setup

We used data released publicly by (NYTimes, 2020) on daily COVID-19 to learn the excitatory point processes of the pandemic outbreak. The data contains the cumulative counts of coronavirus cases in the United States, at the state and county level, over time. Specifically, we separated the U.S. COVID-19 data into state-wise records and further split a state-wise record into different county corpus where each split is named as "a local region" or "a split", containing distinct intensity trajectories from no more than 25 counties.

I.2 Additional Results and Figures

Additional Baselines for COVID data We applied two additional baselines SAC and PPO on a randomly chosen community that contains nine counties. We used the learned NJODE model as the covid environment simulator and the learned model (Without control) is shown in Fig. 13 along with the observed ground truth counts. We also attempted to use the plain Hawkes model (only influence matrix A and baseline b are learnable) to learn the underlying COVID dynamic. As Fig. 13 depicted, however, plain multivariate Hawkes processes (Purple) fail to distinguish the intensity difference between different counties while the NJODE model (Black) correctly captures the characteristic of different counties and also preserves multimodality within each county. We conjecture this is because the plain multivariate Hawkes process does not have enough parameters to characterize the individual variation on the complex COVID data dynamic correctly. Regarding the intervention effect, we observe that ANI has successfully reduced the number of infested people (having a positive reduced intensity value) in six counties among nine while SAC and PPO struggle to have a positive intervention effect on the nine counties. We speculate it is caused by the disconnected gradients between the consecutive latent states (without backpropagating through the learned COVID dynamic model itself) when using model-free RL algorithms like SAC and PPO.

Fairness constraints and more We also present the results and trade-offs for different policies under various fairness constraints. Specifically, we use λ_1 to control the weight of an intervention budget cost and use λ_2 to control the weight of a policy smoothing cost. The intervention cost used here is defined by the distance between two counties when an intervention is implemented and the smoothing cost is defined by the distance between two consecutive policies. We use the average reduced intensity and the average lockdown probability for each edge during the total horizon to

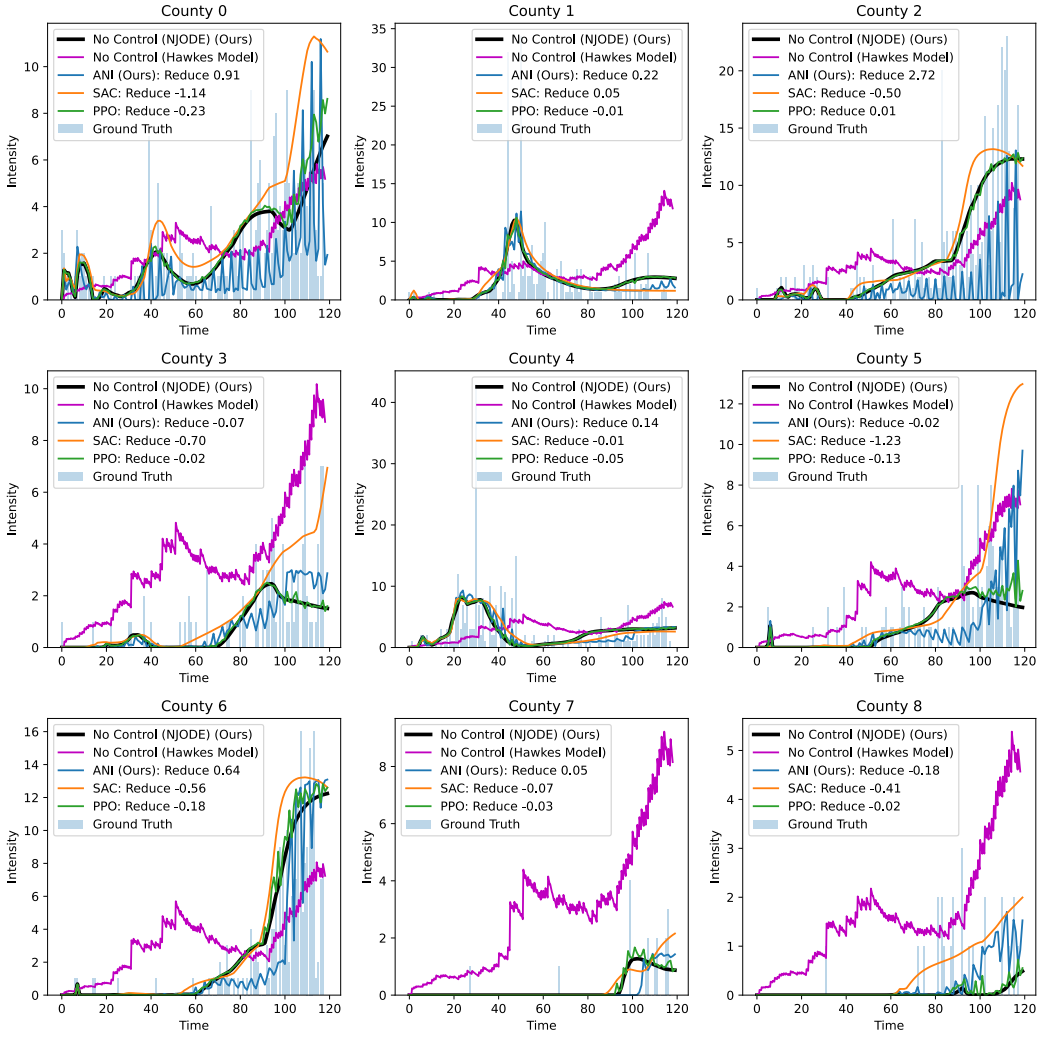

Figure 13: Additional baselines on nine counties

measure fairness and the result is shown in Table 4. Interestingly, we find imposing heavy constraints on the policy simultaneously ($i.e.,\ \lambda_1 = 0.1, \lambda_2 = 0.1$) does not lead to the lowest lockdown probability (1.29%) but does give the lowest control effect (0.01). Instead, only enforcing smoothing constraints ($i.e., \lambda_1 = 0.0, \lambda_2 = 0.1$) gives us a fairer policy, i.e., less effort to intervene (average lockdown probability: 0.43 %) but achieve fair control effect (average reduced intensity: 0.10). We conjecture that adding extra intervention cost constraints will discourage the agent from exploring and thus underperform policy smoothing constraints. We illustrate the detailed lockdown for different constraints in Fig. 14.

Table 4: Average lockdown probabilities and reduced intensity under different soft constraints.

Fairness / Parameters	$\lambda_1 = 0.0$		$\lambda_1 = 0.1$	
	$\lambda_2 = 0.0$	$\lambda_2 = 0.1$	$\lambda_2 = 0.0$	$\lambda_2 = 0.1$
Average Reduced intensity	0.25(0.06)	0.10(0.04)	0.06(0.05)	0.01(0.03)
Average Lockdown probability (%)	1.31(0.53)	0.43(0.12)	1.29(0.27)	1.29(0.17)

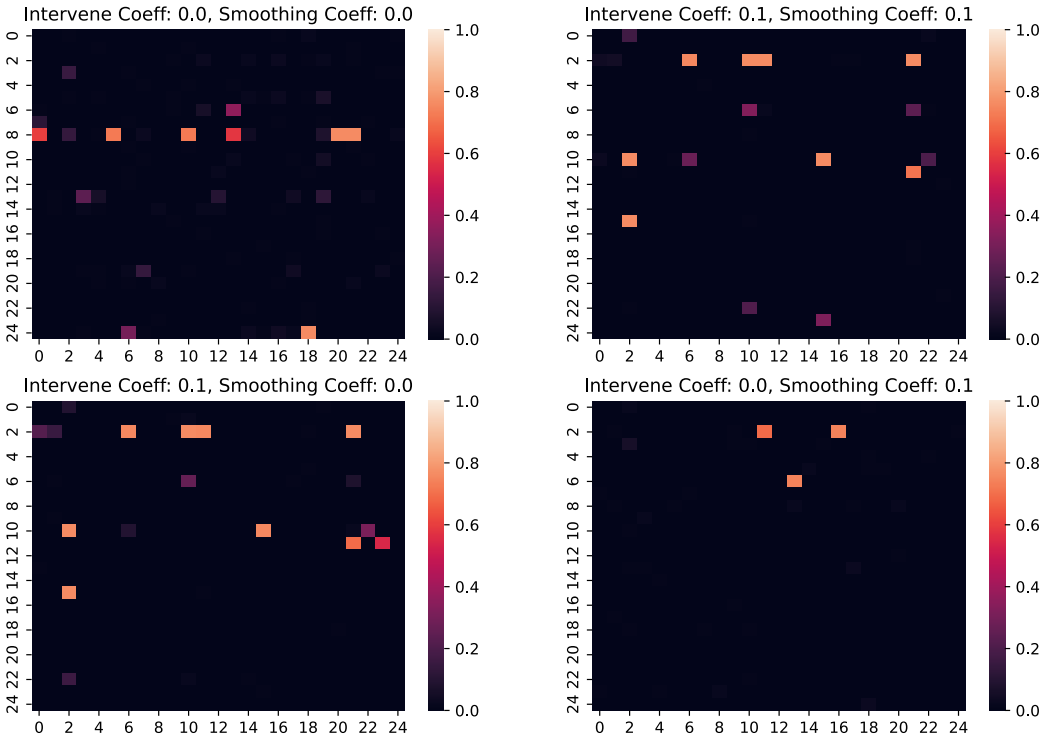

Figure 14: Amortized Networks Interventions in probabilities under different constraints.

J Additional Details of Traffic Data Experiement

J.1 Dataset Setup

To simulate real-world traffic, based on the road types shown in Fig. 6, we design a road network with four types: intersections with one or two lanes, and T-Junction with one or two lanes. Specifically, we let the speed of the road be 8m/s or 11m/s randomly. Then, we generate car trips by the random generation tool from the SUMO package. We make such a simulation for 1000s at one run. After this playing, we can get the simulation results including emissions (e.g. CO_2, CO), positions, speed, and lane id (i.e. the car runs in which lane with lanes more than one in a road) of each car at each time step. Given the generated summary data from SUMO, we then count the congestion event (i.e. the car speed less than 0.5m/s) for the following analysis.

J.2 Additional Results and Figure

We show the intensity cost of the learning process for four types of roads in our simulation setup in Fig. 15. For both our model (meta) and our model (train from scratch), the cost trend will converge after several time steps, which proves that our model has great learning and generalization ability.

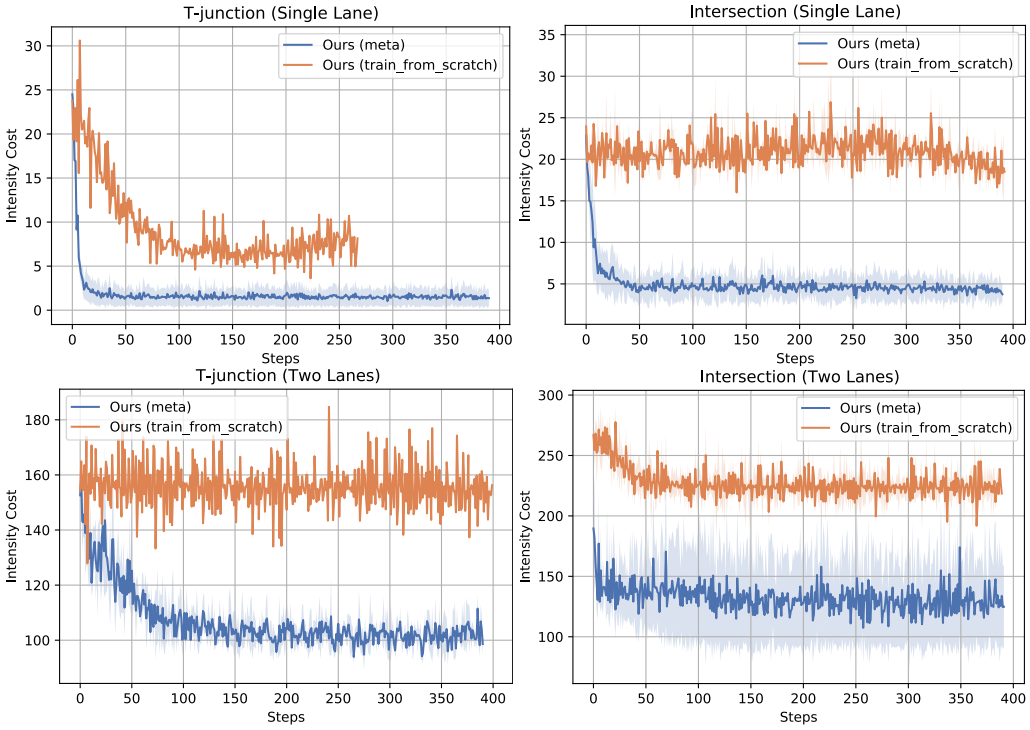

Figure 15: Intensity cost of four types road.

J.3 Case Study

We further provide a case study to show the great interpretation capacity of our model. In the T-junction with single lane scenario, there are 2 discrete actions, corresponding to the following green phase configurations in Fig. 16. The traffic light is marked as node 10, while the other three lanes are marked as node 8, node 11, and node 12 in our Sumo simulation setup.

Real-world traffic can be represented as an NJODE model corresponding with the change of traffic lights. In our simulation, when node 10 (i.e. traffic light) becomes green, the lane controlled by this traffic signal will connect, which is represented as 1 in Fig. 17; otherwise, the connection between two lanes is disconnected and is represented as 0. For instance, in the first sub-graph of Fig. 17, the car can move from node 11 to node 12 under the control of its traffic signal. The learned policy of our model is exactly consistent with real traffic trips, which demonstrates that our model has great adaptation ability in uncovering real-world network interventions.

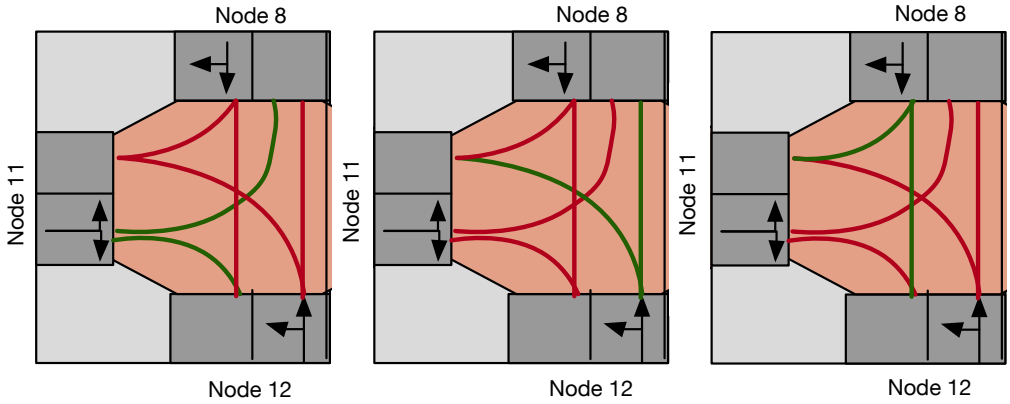

Figure 16: Discrete actions of the T-junction (single lane).

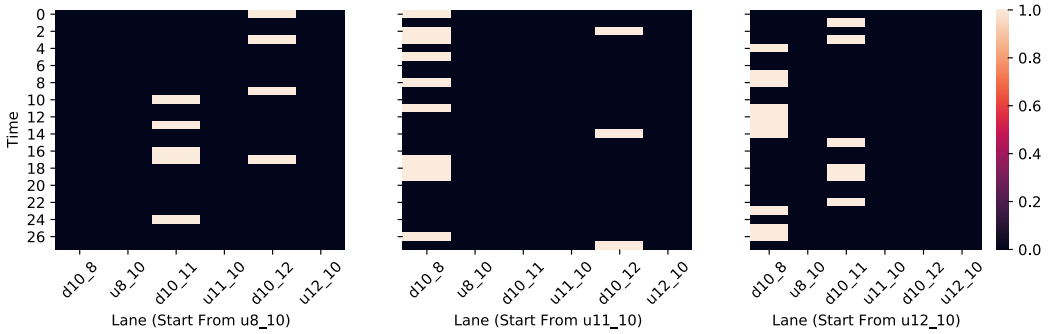

Figure 17: The learned policy of traffic generated from our model.