A Distribution shift in our real-world dataset

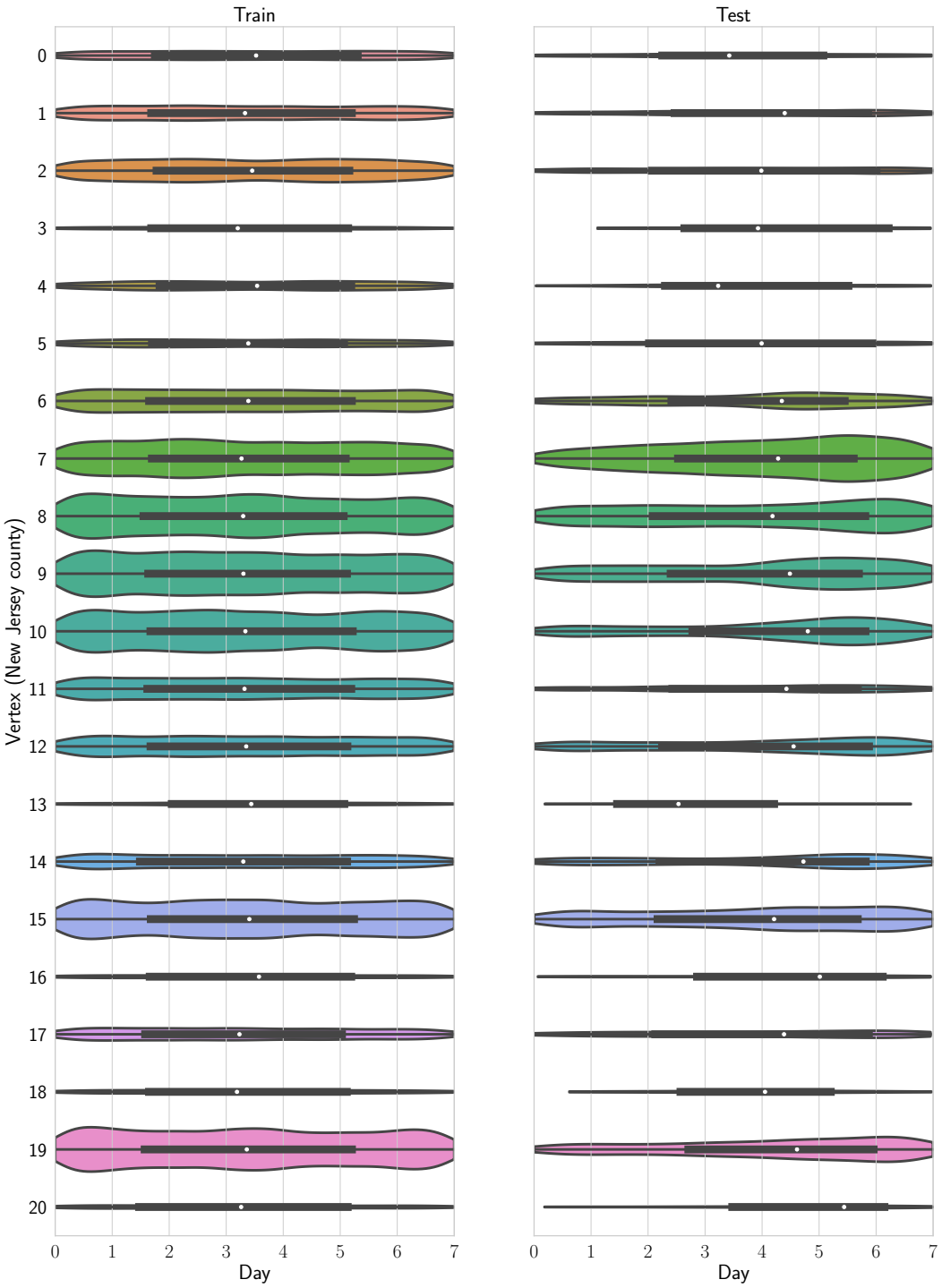

Figure 5: Distribution shift between the train and test distribution of the New Jersey COVID-19 cases created by Chen et al. [5] when binned by the 21 counties. For each of the two distributions, the height of each violin plots is normalized by the total count of observations in that split, i.e. size of training set or size of test set. For most vertices, there are fewer COVID cases later in the 7 day interval in the test set than in the training set.

B Learned forecasts on COVID-19 dataset

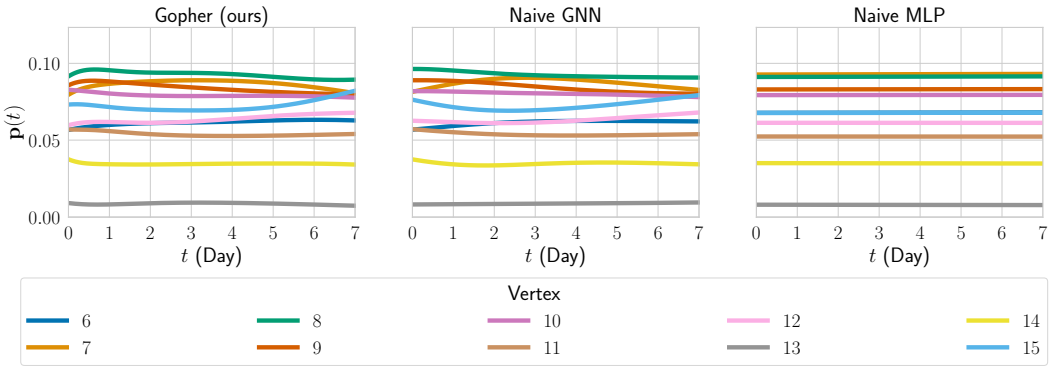

Figure 6: A copy of Figure 4 for easier comparison to the empirical distribution in Figure 5

C Model robustness

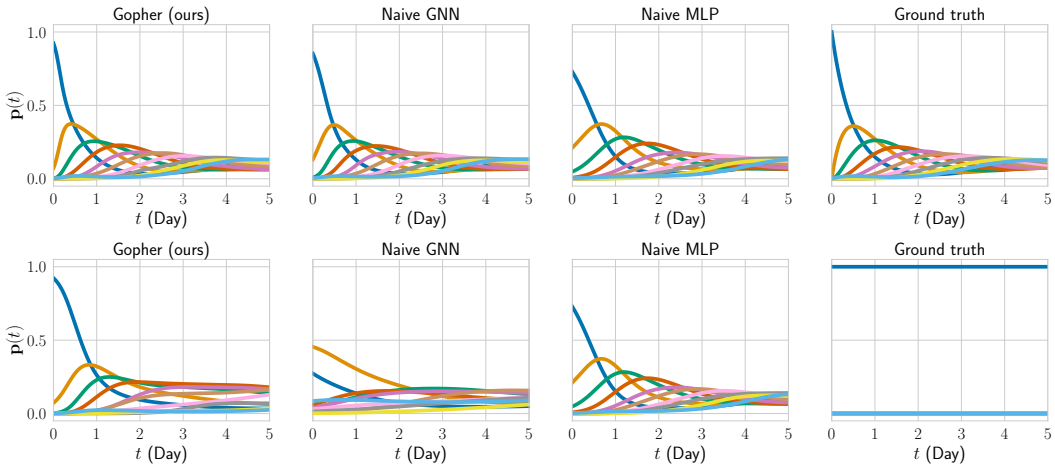

Figure 7: (Top) The learned and ground truth $\mathbf{p}(t)$ for the ring graph dataset. (Bottom) The predicted $\mathbf{p}(t)$ after we remove all edges from the ring graph. Notice that the ground truth of a completely disconnected graph is to have $\mathbf{p}(t) = \mathbf{p}(0)$ for all t. However, all of the models fail completely on this new disconnected graph, suggesting that they do not learn the true dependence of $\mathbf{p}(t)$ on the graph structure.

D Implementation details

We use 2 layer MLPs with 64 hidden units per layer whenever we use a MLP. We use Swish activations to ensure smoothness of our dynamics. We also use an augmented neural ODE [8], using 16 dimensions as augmented dimensions out of the 64 hidden dimensions.

D.1 Model architectures.

GOPHER. We parameterize the dynamics g from Equation 2 using one graph isomorphism network layer parameterized by a MLP. We also use a MLP to model the projection π.

NAIVEGNN. We use the same architecture as GOPHER, namely a GIN layer followed by a projection π. However, instead of using the model to parameterize ODE dynamics, we directly input the node embeddings concatenated with time t through the GNN.

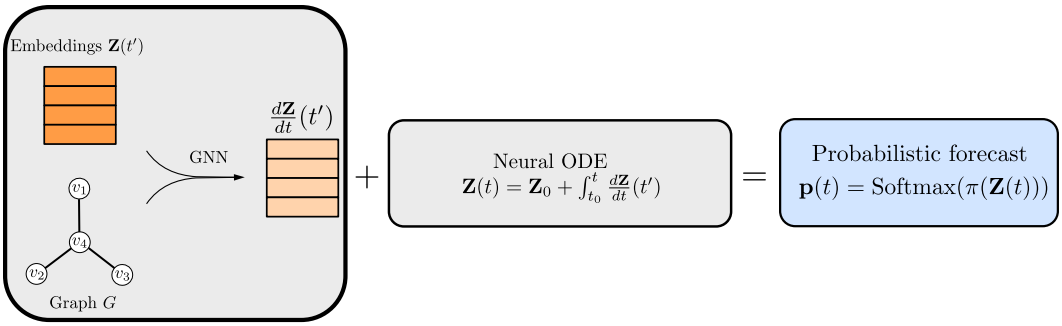

Figure 8: Architecture of GOPHER.

NAIVEMLP. We replace the GIN layer of NAIVEGNN with a MLP, keeping all else the same.

D.2 Training procedure and dataset details.

To maximize hardware parallelism, we parallelize our neural ODE computation across sequence timesteps and across sequences using the time-reparameterization trick outlined in Chen et al. [5]. For the synthetic datasets, we use the AdamW optimizer with 0.01 learning rate and batch size 64 for 30 epochs. For the COVID-19 dataset, we use a 3×10^{-4} learning rate and batch size 4 for 15 epochs. Here, each batch consists of multiple sequences drawn from the training period $[0, T]$. We use $T = 5$ for the ring graph, $T = 1$ for the geometric graph, and $T = 8$ for the New Jersey counties graph. We generate the ring graph dataset by using hand-set coefficients for the edge weights A_{uv} to allow for counter-clockwise transport. We generate the geometric graph dataset by generating a random geometric graph via the `networkx` python package and drawing a random sample of $\{A_{uv}\}$.

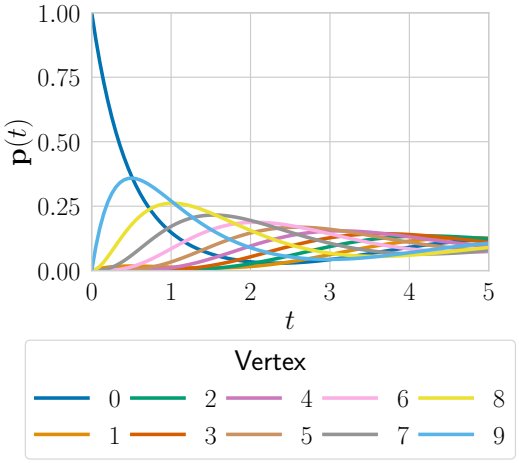

Figure 9: The dynamics of each component of $\mathbf{p}(t)$ on the cyclic graph. Each component corresponds to the probability of a vertex on the graph.