Appendix A. Granger Causality of ACD

Here, we show that when constraining the edge-type $e = 0$ to be the zero function, time series i does not Granger cause the model prediction of j in ACD. In the noiseless setting, at the global optimum of sufficiently expressive model classes, this is equivalent to saying that we recover the true Granger causal relations.

Claim. If $z_{ij,0} = 1$, i does not Granger cause the model prediction of j in ACD.

Proof. According to Theorem 1, time-series i does not cause j, if g_j is invariant to $\boldsymbol{x}_{\bar{i}}^{\leq t}$. In our model, the decoder represents this non-linear model g_j and consists of two functions. First, it propagates information across edges using Eq. (12). This function returns a value of zero, if $z_{ij,0} = 1$. This output is used for the second function, described in Eq. (1), which does not introduce any new terms that depend on i. Thus, if $z_{ij,0} = 1$, the decoder's prediction for j is invariant to $\boldsymbol{x}_{\bar{i}}^{\leq t}$, and i does not Granger cause these predictions.

Appendix B. Fully Observed Amortized Causal Discovery

B.1. Experimental Details

B.1.1. DATASETS

Physics Simulations To generate these simulations, we follow the description of the underlying physics of Kipf et al. (2018) for the phase-coupled oscillators (Kuramoto) (Kuramoto, 1975) and the particles connected by springs. In contrast to their simulations, however, we allow the connectivity matrix, which describes which time-series influences another, to be asymmetric. This way, it describes causal relations instead of correlations.

For both datasets, we generate 50,000 training and 10,000 validation samples. We restrict the number of test samples to 200, since the previous methods we compare to must be refit for each individual sample. We simulate systems with $N = 5$ time-series. Our training and validation samples consist of $T = 49$ time-steps, while the test-samples are $T = 99$ time-steps long. This increased length allows us to infer causal relations on the first half of the data, and to test the future prediction performance on the second half (with $k = \{1, ..., 49\}$). In Figure 8, we show some examples of both the particles and Kuramoto datasets in the noiseless setting and with added noise.

Netsim The Netsim dataset simulates blood-oxygen-level-dependent (BOLD) imaging data across different regions within the human brain and is described in Smith et al. (2011). The task is to infer the directed connections, i.e. causal relations, between different brain areas.

The Netsim dataset includes simulations with different numbers of brain regions and different underlying connectivity matrices. In our experiments, we use the data from the third simulation Sim-3.mat as provided by Khanna and Tan (2019). It consists of samples from 50 subjects, each with the same underlying causal graph, each of length $T = 200$ and including $N = 15$ different brain regions. Note, that we report worse results than Khanna and Tan (2019), since we assume self-connectivity for all time-series and only evaluate the causal discovery performance between *different* time-series.

The dataset is very small (50 samples) and due to this, we do not use a training/validation/test split, but use the same 50 points at each phase instead. While this is not standard machine learning practice, it still facilitates a fair comparison to the other methods, each of which is fit to individual

test points. The purpose of including experiments on this dataset is not to demonstrate generalization ability, but rather to show that our method is flexible enough to work reasonably well even in the classical causal discovery setting (with one shared causal graph, and fitting the model on the test set).

B.1.2. ARCHITECTURE AND HYPERPARAMETERS

Our model is implemented in PyTorch (Paszke et al., 2019). We did no hyperparameter optimization for model training, but used the settings as described for the NRI model (Kipf et al., 2018). The latent dimension throughout the model is set to size 256. We optimize our model using ADAM (Kingma and Ba, 2015) with a learning rate of 0.0005. In the experiments on the particles dataset, the learning rate is decayed by a factor of 0.5 every 200 epochs. We set our batch-size to 128 and train for 500 epochs. The temperature of the Gumbel-Softmax is set to $\tau = 0.5$. During testing, this concrete distribution is replaced by a categorical distribution to obtain discrete edge predictions.

There was no thorough hyperparameter optimization done for test-time adaptation (TTA). Since there was no pre-existing implementation, some hand-tuning was performed. We use a learning rate of 0.1 for the Kuramoto and particles datasets and 0.01 for Netsim. For each, we run 1000 iterations.

Encoder In our experiments, the amortized encoder applies a graph neural network $f_{enc,\phi}$ on the input. It implements two edge-propagation steps along the causal graph:

$$\boldsymbol{\psi}_j^1 = f_{\text{emb}}(\boldsymbol{x}_j) \tag{15}$$

$$\boldsymbol{\psi}_{ij}^1 = f_e^1([\boldsymbol{\psi}_i^1, \boldsymbol{\psi}_j^1]) \tag{16}$$

$$\boldsymbol{\psi}_j^2 = f_v^2(\sum_{i \neq j} \boldsymbol{\psi}_{ij}^1) \tag{17}$$

$$\boldsymbol{\psi}_{ij} = f_e^2([\boldsymbol{\psi}_i^2, \boldsymbol{\psi}_j^2]) \quad . \tag{18}$$

f_e^1, f_e^2 and f_v^2 are fully-connected networks (MLPs). On both the particles dataset and Netsim, f_{emb} is an MLP as well (MLP Encoder); on the Kuramoto dataset, we use a 1D CNN with attentive pooling (Lin et al., 2017) instead (CNN Encoder).

When conducting test-time adaptation as described in Eq. (7), we remove the encoder and model a distribution over $\mathbb{G}$ using a non-amortized variational distribution $q(\boldsymbol{z})$ with its initial values sampled from a unit Gaussian.

Decoder The decoder implements a single edge-propagation step according to equations 12-14. It uses MLPs for both f_e and f_v. To improve performance, we train the decoder to predict several time-steps into the future. For this, we replace the true input $\boldsymbol{x}^t$ with the predicted $\boldsymbol{\mu}^t$ for $k = 10$ steps.

Following our causal formulation of the NRI model, we implement Eq. (12) by masking out the values of the corresponding edges. Thus, the ordering of the edge types is not arbitrary in our setting.

We note that while this implementation of the decoder assumes a full-time graph of Markov order 1, the full ACD framework does not, and could be implemented using a recurrent architecture to remove this assumption.

Since our physics simulations are differentiable, we can replace the decoder with the ground-truth dynamics and backpropagate through them. We call this setup the simulation decoder.

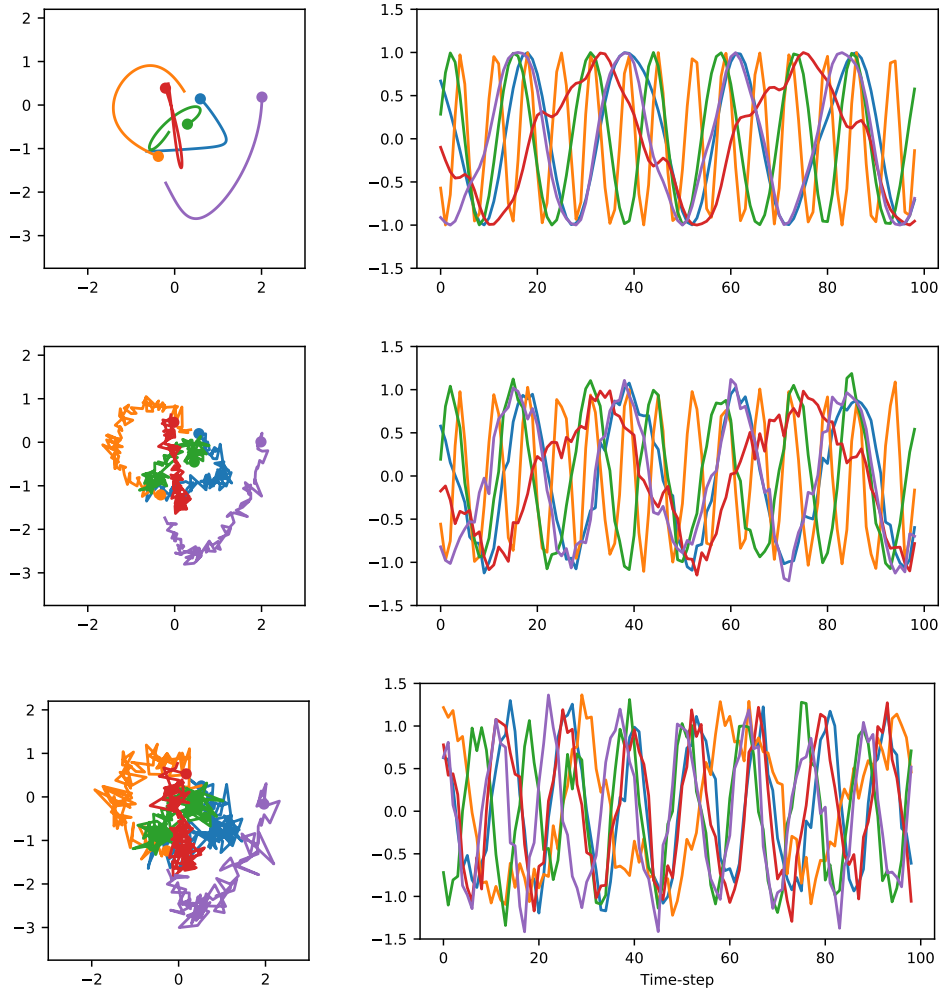

Figure 8: Example trajectories with different levels of observation noise for the Particles dataset (left) and Kuramoto (right). The observation noise is sampled from a zero-mean Gaussian distribution with standard deviation of (from top to bottom) 0.0, 0.1 and 0.2, respectively.

Variance When we report the variance on the ACD results, we collected these across five different random seeds. Baselines in Kuramoto/Netsim use three seeds each, except for NGC, which uses only one due to a longer runtime (the confidence intervals shown for NGC are confidence intervals on the AUROC itself, whereas all other confidence intervals are based on variance of AUROC across seeds).

B.1.3. BASELINES

We compare ACD against several baselines:

Neural Granger Causality From Tank et al. (2018), we optimized an MLP or LSTM to do next step prediction on a sample. We found that the MLP worked best. The causal links are wherever an input weight is non-zero. We used ADAM and then line search to find exact zeros. In this method, we calculate AUROC by running with a range of sparsity hyperparameters ($\lambda = [0, 0.1, 0.2, 0.4, 0.8]$

for Kuramoto and $\lambda = [0, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 1]$ for Netsim). As in Tank et al. (2018), we calculate a score s for each edge, where $s = \min\{\lambda : z_{ij,0} = 1\}$, and use that score to calculate AUROC. Code was used from https://github.com/icc2115/Neural-GC.

ESRU Khanna and Tan (2019) take a similar approach to Tank et al. (2018), but they use economy statistical recurrent units (eSRU), instead of LSTMs. We found one layer worked best, and used their hyperparameters otherwise. We use sparsity hyperparameters $[0.1, 0.2, 0.3, 0.4, 0.5]$ for Kuramoto, and $[0, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 1]$ for Netsim. Code was used from https://github.com/sakhanna/SRU_for_GCI.

MPIR Wu et al. (2020) determine where causal links exist by examining the predictive performance change when noise is added to an input variable. Code for this method and the baselines below was used from https://github.com/tailintalent/causal.

Transfer Entropy Schreiber (2000) suggest this entropy-like measure between two variables to produce a metric which is likely to be higher when a causal connection exists. We use the implementation by Wu et al. (2020).

Mutual Information Using the implementation by Wu et al. (2020), we calculate the mutual information between every pair of time series.

Linear Granger Causality Using the implementation by Wu et al. (2020), this is a linear version of Granger causality where non-zero linear weights are taken as greater causal importance.

We did not run the baselines on the particles dataset since it is two-dimensional and most baselines did not provide an obvious way for handling multi-dimensional time series. When training ACD on the particles and Kuramoto datasets, we additionally input the velocity (and phase for Kuramoto) of the time-series. Since our chosen NRI encoders and decoders are not recurrent we cannot recover this information in any other way in this model. This enables a more fair comparison to the recurrent methods, which are able to aggregate this information over several time steps.

B.2. Additional Experimental Result - Training Curves

Fig. 9 shows the training curves when training on 100 training samples of the particles dataset. We observe that the encoder overfits on the training samples, as indicated by the AUROC performance. In contrast, the decoder shows less overfitting as indicated by the negative log-likelihood (NLL) performance.

Appendix C. Amortized Causal Discovery with Unobserved Variables

C.1. Temperature Experiments

Implementation Details In this experiment, we use the CNN encoder and a simulation decoder matching the true generative ODE process. Our optimization scheme is the same as before.

For modeling the latent temperature, we output a uniform distribution as our posterior $q_{\phi_c}(c|\boldsymbol{x})$. One tricky aspect about this is the KL-Divergence:

$$KL(q_{\phi_c}(c|\boldsymbol{x})||p(c)) = -\int q_{\phi_c}(c|\boldsymbol{x}) \log \frac{q_{\phi_c}(c|\boldsymbol{x})}{p(c)} d\boldsymbol{z} \quad . \tag{19}$$

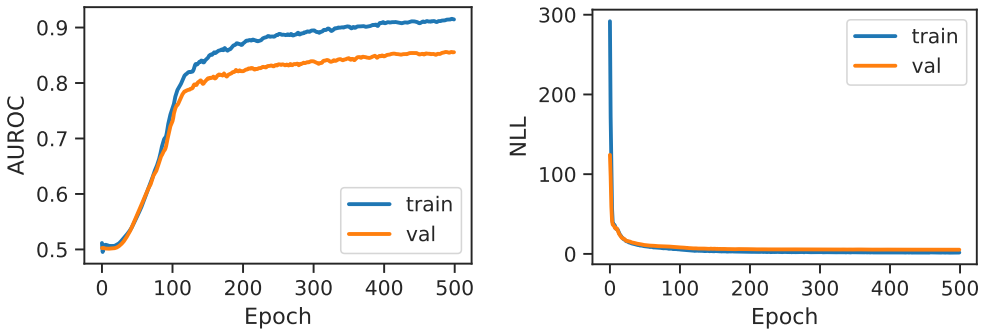

Figure 9: Training curves when training on 100 samples of the particles dataset. The encoder performance (AUROC - left) shows stronger signs of overfitting than the decoder performance (NLL - right).

We must ensure that our posterior support is a subset of our prior support. Otherwise, the KL-Divergence is undefined and optimization impossible. Recall that our prior is a uniform distribution over $[0, 4\alpha]$.

We output two latent parameters $a, b \in \mathbb{R}$ for each input and use these values to parametrize a mean m and a half-width w for the uniform distribution. First, we bound these values to represent a uniform distribution u_1 in $[0, 1]$. To achieve this, we let $m_1 = \sigma(a)$ and $w_1 = \sigma(b) * \min(m_1, 1 - m_1)$ with $\sigma(x) = \frac{1}{1+\exp(-x)}$. We then sample a temperature $\hat{c}_1 \sim u_1 = U(m_1 - w_1, m_1 + w_1)$, which is guaranteed to be bounded within $[0, 1]$. Stopping gradients, we use this temperature sample in the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x}, c)$ to improve the causal discovery performance.

Next, we scale this result to the desired interval $[0, 4\alpha]$. To achieve this, we feed the scaled temperature $\hat{c} = 4\alpha\hat{c}_1$ into the decoder, and use the scaled distribution $u = U(4\alpha m_1 - 4\alpha w_1, 4\alpha m_1 + 4\alpha w_1)$ to find our KL term. We allow gradients to flow through the temperature sample in both the decoder and the distribution in the KL term, which informs our parameter updates.

Additional Results Similarly to Fig. 6, we show the future prediction performance in MSE across different values of α in Fig. 10. Again we find a slight improvement in performance when using ACD with *Latent* compared to the baselines, although this is a noisier indicator.

Additionally, we evaluate how well the introduced latent variable learns to predict the unobserved temperature. To do so, we use the mean of the predicted posterior uniform distribution. When a discrete categorical prediction is needed for evaluation, we quantize our results into three bins based on their distance in log-space. To calculate AUROC in this three category ordinal problem, we average the AUROC between the two binary problems: category 1 vs not category 1, and category 3 vs not category 3 (category 2 vs not 2 is not a valid regression task for the purposes of AUROC which is concerned with ordering, since it is the middle temperature and hence the labels would not be linearly separable).

The confusion matrix between true and predicted temperature in Fig. 11 indicates that ACD with Latent's prediction tends to be conservative: it is more likely to predict a too low temperature than a too high one. This is probably due to higher temperatures incurring larger MSEs, since higher temperature systems are more chaotic and thus less predictable.

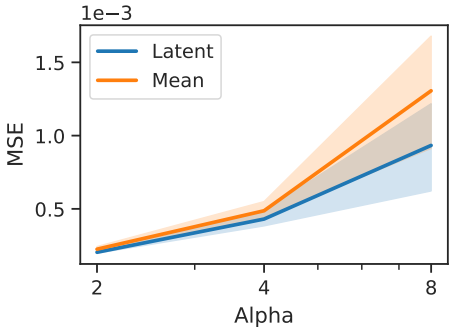

Figure 10: MSE (lower better) averaged across 5 random seeds for hidden temperature experiment. MSE for *None* baseline was much worse with MSE $= 0.009, 0.02, 0.04$ for $\alpha = 2, 4, 8$ (not shown in plot).

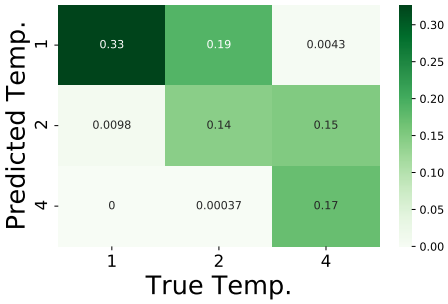

Figure 11: Confusion matrix for latent temperature prediction with $\alpha = 2$. ACD with Latent's prediction tends to be conservative: it is more likely to predict a too low temperature than a too high one.

Table 3 lists the temperature prediction results across all tested values of α. We find that we can predict the unobserved temperature quite well, especially with respect to ordering (as measured by correlation and AUROC).

	α		
	2	4	8
Correlation	0.888	0.844	0.661
Accuracy	0.644	0.384	0.346
AUROC (1vAll)	0.966	0.935	0.843

Table 3: Latent Temperature Prediction Metrics. We treat the mean of the outputted interval of the uniform posterior as the predicted temperature. For accuracy, this value discretized in log space to get a ternary prediction. *AUC (1vAll)* averages the two one-vs-all AUC values, which can be calculated in a 3-category ordinal problem.

C.2. Unobserved Time-series

Implementation Details For modeling the unobserved time-series, we employ a two-layered, bi-directional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) with a latent dimension of size 256.

Additional Results The full evaluation of our experiments with an unobserved time-series can be found in Table 4. Our results indicate that our proposed method *ACD with latent* predicts the trajectory of the unobserved time-series (unobserved MSE) more accurately than the *Mean* imputation baseline. Even though this prediction is worse than for the *Supervised* baseline, ACD with *Latent* manages to recover the performance of the fully *Observed* baseline better than the *None* and the *Mean* imputation baselines.

Fig. 12 shows the performance of the tested methods dependent on the number of time-series that are influenced by the unobserved one. In addition to Fig. 6 in our Experiments section, these

Method	AUROC	Accuracy	MSE	unobserved MSE
Observed	(0.99)	(0.993)	(0.00301)	-
Supervised	0.982	0.931	0.00822	0.0164
None	0.946	0.882	0.0119	-
Mean	0.951	0.881	0.0106	0.0397
ACD with latent	0.979	0.918	0.00747	0.0375

Table 4: Experiments with an unobserved time-series.

plots show the achieved accuracy and MSE results. The general trends are the same. Fig. 13 shows example trajectories and the corresponding predictions for all tested methods.

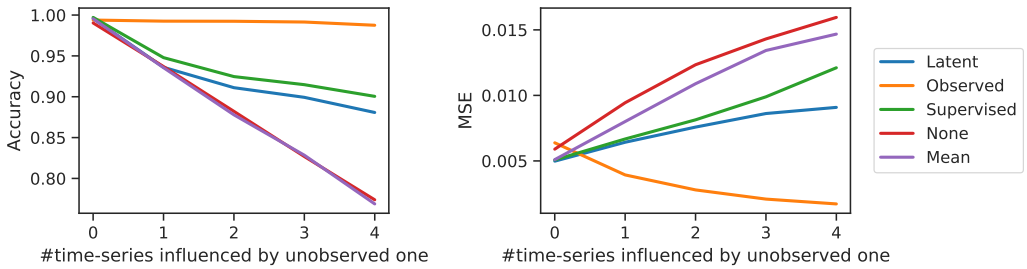

Figure 12: Experiments with an unobserved particle. Performance of the various methods depends strongly on how many observed particles are influenced by the unobserved one (x-axis). The more particles that are influenced by the unobserved particle, the stronger the benefit of using an additional *Latent* variable for modeling its effects. Left - causal relation prediction accuracy (higher = better), right - MSE (lower = better).

Additional Experiment: Uninfluenced Influencer Predicting the trajectory of a time-series that influences only a small number of observed time-series and is (invisibly) influenced by them is arguably very difficult. In this follow-up experiment, we reduce the difficulty of this problem by adding two assumptions: (1) the unobserved time-series influences *all* observed time-series and (2) it is not influenced by any of the observed time-series. This way, we gain more information about its trajectory (due to (1)) and its trajectory becomes easier to predict (due to (2)). Indeed, in this setup, *ACD with latent* manages to almost completely recover the performance of the fully observed baseline (Table 5). In contrast, the performance of the *None* and *Mean* imputation baselines worsens considerably in this setting. Now, all time-series are influenced by the unobserved particle – making their prediction harder when not taking into account this hidden confounder. Fig. 14 shows example trajectories and the corresponding predictions for all tested methods in this setting.

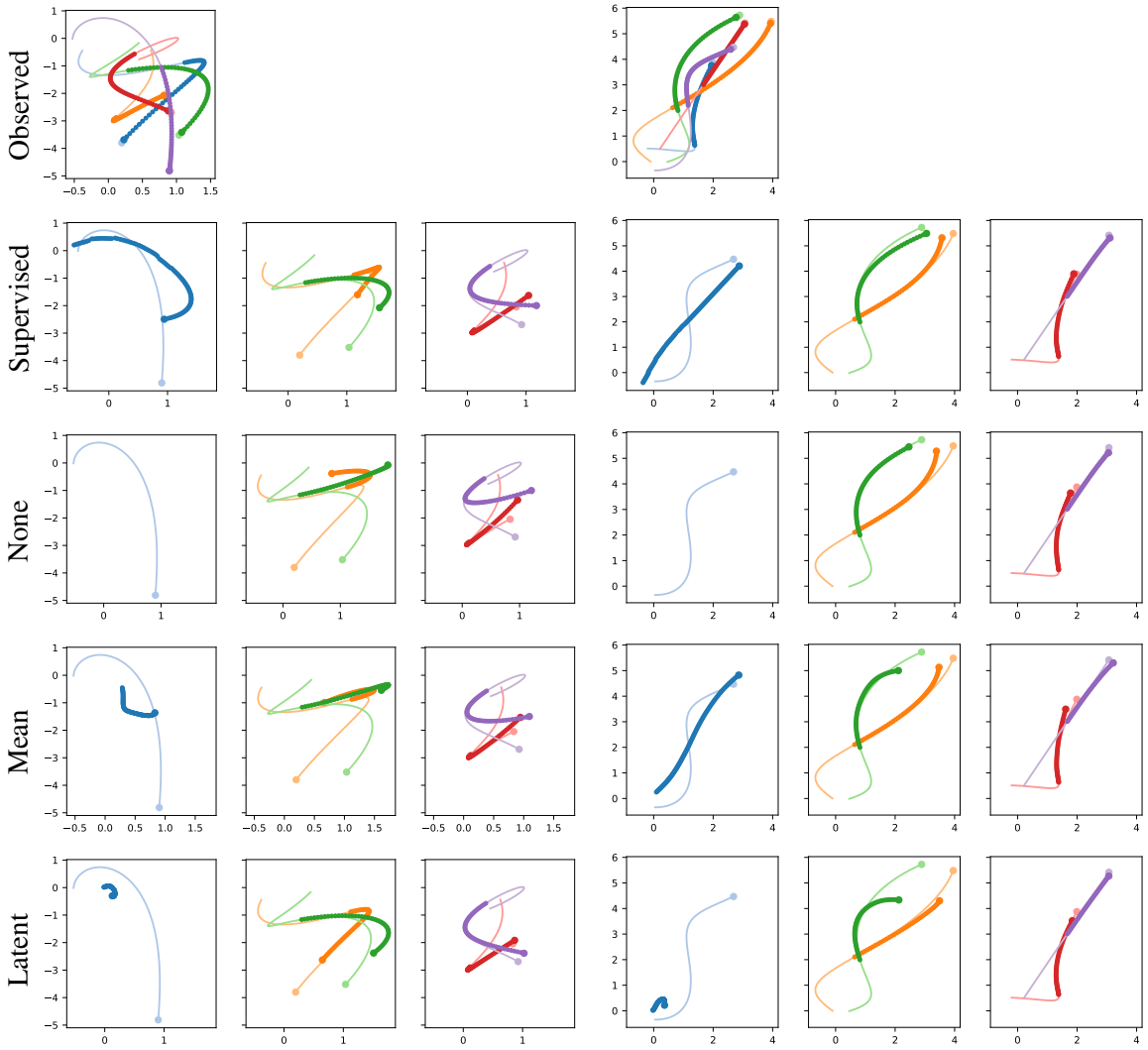

Figure 13: Predicted trajectories for all tested methods in the unobserved time-series experiment for two samples (left/right). From top to bottom: Baselines – observed, supervised, none and mean; proposed ACD with latent. The faded lines depict the ground truth trajectory; bold lines are the trajectories predicted by the model and they start after initializing the model using first half of the ground truth. Dots denote the end of the trajectories. Except for the fully observed baseline, the first panel shows the ground truth and prediction for the unobserved time-series. The second panel shows the trajectories of all time-series that are directly influenced by the unobserved one. The third panel shows the trajectories of all time-series that are not directly influenced by the unobserved one.

Method	AUROC	Acuracy	MSE	unobserved MSE
Observed	(1.0)	(0.997)	(0.0193)	-
Supervised	1.0	0.993	0.024	0.000615
None	0.829	0.76	0.0431	-
Mean	0.853	0.782	0.0365	0.0357
ACD with latent	1.0	0.994	0.0251	0.137

Table 5: Experiments with an unobserved time-series that influences all observed time-series, but is not influenced by them.

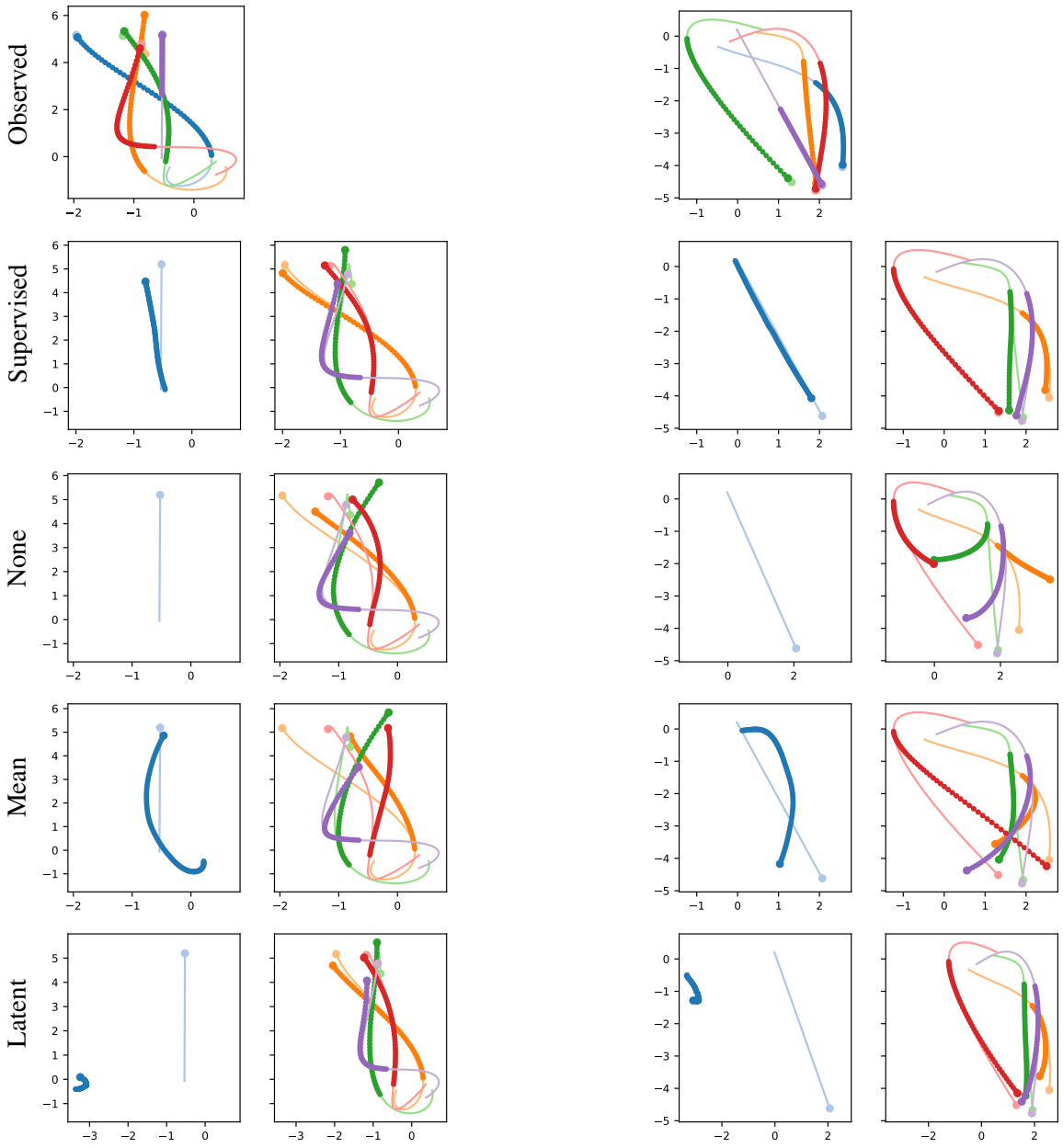

Figure 14: Predicted trajectories for all tested methods when the unobserved time-series influences all observed ones, but stay uninfluenced itself for two samples (left/right). From top to bottom: Baselines – observed, supervised, none and mean; proposed ACD with latent. The faded lines depict the ground truth trajectory; bold lines are the trajectories predicted by the model, and they start after initializing the model using the first half of the ground truth. Dots denote the end of the trajectories. Except for the fully observed baseline, the first panel shows the ground truth and prediction for the unobserved time-series. The second panel shows the trajectories of all observed time-series (which are all influenced by the unobserved one).