A APPENDIX

The supplementary materials contain proofs of suggested theorems in Section A.1, results on additional experiments including details on classic CPD methods in subsection A.2.1, performance on unbalanced datasets in subsection A.2.2 and information about inference time in subsection A.2.3. We also provide details about data preprocessing and data examples in subsection A.3, and implementation details in subsection A.4.

A.1 Proofs

We provide proof that our loss function is a lower bound of the principled loss function. We start with two supplementary lemmas that lead to the main theorem.

Lemma A.1. *$\tilde{L}^h_{delay}(f_{\rm w}, D)$ is a lower bound for the expected value of the detection delay.*

Proof. The expected value of the *detection delay* $\mathscr{L}_{delay}$ given the probabilities of a change point for the scenario (B) has the following form for the change point estimate τ:

$$\mathscr{L}_{delay}(f_{\rm w}, D) = \mathbb{E}_\theta(\tau - \theta)^+ = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=\theta_i}^{\infty}(t-\theta_i)p_t^i\prod_{k=\theta_i}^{t-1}(1-p_k^i),$$

the outer sum over i-s reflects that we average the loss over all sequences in a batch with change points at moments θ_i; the inner sum over t-s corresponds to losses associated with each moment in a sequence; p_t^i is the predicted probability of a change point at the moment t for i-th series. The probability $p_t^i\prod_{k=\theta_i}^{t-1}(1-p_k^i)$ reflects, that (i) we didn't report a change point during all previous time moments, this event has the probability $\prod_{k=\theta_i}^{t-1}(1-p_k^i)$, but reported a change point at time t with probability p_t^i.

As it can be seen, the term includes an infinite number of elements. Moreover, their values decrease with each t, as we multiply by quantities equal to 1 minus change point probability at a particular point, smaller than 1.

We rewrite $\mathscr{L}_{delay}(f_{\rm w}, D)$ in the following form for some T:

$$\mathscr{L}_{delay}(f_{\rm w}, D) = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=\theta_i}^{h}(t-\theta_i)p_t^i\prod_{k=\theta}^{t-1}(1-p_k^i) + \sum_{t=h+1}^{\infty}(t-\theta_i)p_t^i\prod_{k=\theta_i}^{t-1}(1-p_k^i)\right).$$

Consider a lower bound for the second term for a single sequence:

$$\sum_{t=h+1}^{\infty}(t-\theta)p_t\prod_{k=\theta}^{t-1}(1-p_k) \geq (h+1-\theta)\sum_{t=h+1}^{\infty}p_t\prod_{k=\theta}^{t-1}(1-p_k).$$

The expression under the sum introduces the probability of detecting the change moment after the moment T. It equals the probability of not detecting the change before moment $h+1$. Thus,

$$(h+1-\theta)\sum_{t=h+1}^{\infty}p_t\prod_{k=\theta}^{t-1}(1-p_k) = (h+1-\theta)\prod_{t=\theta}^{h}(1-p_t).$$

This brings us to the desired result if we sum the first part and the lower bound for the second part:

$$\tilde{L}^h_{delay}(f_{\rm w}, D) = \frac{1}{N}\sum_{i=1}^{N}\tilde{L}^h_{delay}(f_{\rm w}, X_i, \theta_i),$$

$$\tilde{L}^h_{delay}(f_{\rm w}, X_i, \theta_i) = \sum_{t=\theta_i}^{h}(t-\theta_i)p_t^i\prod_{k=\theta_i}^{t-1}(1-p_k^i) + (h+1-\theta_i)\prod_{k=\theta_i}^{h}(1-p_k^i). \quad (6)$$

□

Lemma A.2. *$\tilde{L}_{FA}(f_{\rm w}, D)$ is a lower bound for the expected value of the time to the false alarm.*

The expected *time to false alarm* $\mathscr{L}_{FA}$ given the probabilities for the scenario (B) has the following form:

$$\mathscr{L}_{FA}(f_{\rm w}, D) = -\mathbb{E}_\theta(\tau|\,\tau < \theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{\theta_i}tp_t^i\prod_{k=0}^{t-1}(1-p_k^i). \quad (7)$$

The same situation with the infinite sum appears in (7) in case $\theta_i = \infty$ — when no change point happens. So, we offer an approximation for $\mathscr{L}_{FA}$:

$$\tilde{L}_{FA}(f_{\rm w}, D) = -\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{\tilde{T}_i}tp_t^i\prod_{k=0}^{t-1}(1-p_k^i) - (\tilde{T}_i+1)\prod_{k=0}^{\tilde{T}_i}(1-p_k^i)\right), \text{where } \tilde{T}_i = \min(\theta_i, T). \quad (8)$$

The proof is similar to the one from A.1.

Theorem A.1. *The loss function $\tilde{L}^h(f_{\rm w}, D, c)$ from* (3) *is a lower bound for $\tilde{\mathscr{L}}(\tau)$ from criteria* (2).

Proof. The proof follows from lemmas A.1 and A.2 as we provide a lower bound for both terms in the loss function in them. □

A.2 Additional experiments

A.2.1 Choosing best classic methods. As "classic" CPD approaches, we consider offline methods BinSeg, Pelt, and non-trainable Kernel CPD from the ruptures package [30]. To provide an honest comparison, we choose hyperparameters on the training set that maximize the F1 metric and then apply them to the test set. For search, we varied the method's central models (or kernels for Kernel CPD), the penalty parameter and the number of change points to detect in every sequence. The results with the best hyperparameters are presented in Table 4. We also calculate the metrics when the hyperparameters are set with respect to the maximisation F1 metric on a validation set to better understand the maximum ability of classic approaches.

In our opinion, setting the number of change points to detect in every sequence (n_pred) to 1 is an unsuccessful choice as it provides a change point in every sequence, and the number of false alarms rapidly increases, so we avoid using it. One of the apparent disadvantages of classic methods is the necessity to pick hyperparameters manually, while representation-based methods

Table 4: Main quality metrics for considered classic CPD approaches. ↑ marks metrics we want to maximize, ↓ marks metrics we want to minimize. We present the best options with hyperparameters selected using training or test data. Best values for parameters selected via using training set are highlighted with bold font, best values for parameters selected on test set underlined.

Data for hyper-param. select.	Method	Core model or kernel	Hyperpa-rameters	Mean Time to FA ↑	Mean DD ↓	F1 ↑	Covering ↑
			1D Synthetic data				
Train	BinSeg	L2	pen = 7000	**101.39**	7.72	0.9651	0.9651
	BinSeg	L2	n_pred = 1	64.66	0.46	0.4772	0.8900
	PELT	L2	pen = 7000	**101.39**	7.72	0.9651	0.9651
	PELT	L2	n_pred = 1	14.65	**0.05**	0.0392	0.7721
	Kernel CPD	Linear	pen = 21	94.81	0.64	**0.9872**	**0.9954**
	Kernel CPD	Linear	n_pred = 1	62.77	0.12	0.6842	0.9151
Test	BinSeg	L2	pen = 5400	99.48	5.85	0.6438	0.9679
	PELT	L2	pen = 5600	99.48	5.85	0.6438	0.9679
	Kernel CPD	Linear	pen = 16	94.49	0.33	0.9904	0.9964
			100D Synthetic data				
Train	BinSeg	L2	pen = $2.11 \cdot 10^5$	**96.94**	3.39	0.6201	0.9788
	BinSeg	L2	n_pred = 1	62.69	0.34	0.4772	0.8891
	PELT	L2	pen = $9.6 \cdot 10^4$	95.53	2.13	0.5571	0.9808
	PELT	L2	n_pred = 1	4.95	0.05	0.0328	0.8025
	Kernel CPD	Linear	pen = 0.9	94.20	0.03	**0.9968**	**0.9996**
	Kernel CPD	Linear	n_pred = 1	62.59	**0.01**	0.6871	0.9053
Test	BinSeg	L2	pen = 10^6	100.19	6.52	0.6667	0.9683
	PELT	L2	pen = $4.8 \cdot 10^4$	98.58	4.95	0.6778	0.9730
	Kernel CPD	Linear	pen = 167	94.20	0.03	0.9968	0.9996
			Human Activity Recognition				
Train	BinSeg	RBF	pen = 0.2	4.63	0.37	0.3097	0.6634
	BinSeg	RBF	n_pred = 1	**10.17**	4.83	0.8563	0.6624
	PELT	RBF	pen = 0.5	4.63	0.37	0.3097	0.6634
	PELT	L2	n_pred = 1	9.86	0.92	0.5740	0.8414
	Kernel CPD	RBF	pen = 0.3	5.90	**0.07**	0.6703	**0.8538**
	Kernel CPD	Cosine	n_pred = 1	9.35	1.26	**0.9023**	0.8510
Test	BinSeg	RBF	pen = 0.5	10.17	4.83	0.8563	0.6624
	PELT	RBF	pen = 0.6	10.17	4.83	0.8563	0.6624
	Kernel CPD	Linear	pen = 0.9	9.57	0.09	0.8761	0.9450
			Sequences of MNIST images				
Train	BinSeg	L2	pen = 402	**45.26**	5.08	0.4901	**0.8430**
	BinSeg	L2	n_pred = 1	29.16	1.62	0.4252	0.7029
	PELT	L2	pen = 402	45.26	5.08	0.4901	**0.8430**
	PELT	RBF	n_pred = 1	9.95	**0.05**	0.0583	0.7259
	Kernel CPD	Linear	pen = 375	43.02	4.08	**0.5500**	0.8364
	Kernel CPD	RBF	n_pred = 1	28.88	3.81	0.4887	0.6420
Test	BinSeg	L2	pen = 439	47.99	5.63	0.5172	0.8697
	PELT	L2	pen = 439	47.99	5.63	0.5172	0.8697
	Kernel CPD	Linear	pen = 357	40.94	3.20	0.5724	0.8270

do it during the learning process. We can see from Table 4 that the most suitable choices for the training set don't provide the best metric on the test set in most cases.

A.2.2 Experiment on imbalanced datasets. In our main experiments, we mostly use balanced training datasets. They include an equal number of sequences with and without change points. However, we expect to face change points less often than normal sequences without them in real-world data. Thus, to investigate the performance of proposed approaches in more realistic scenarios, we conduct additional experiments with sequences of MNIST images. We try a different number of sequences with change points and calculate the main quality metrics. We test our three main methods based on representation learning (InDiD, BCE, BCE+InDiD) with the same model design as for the main experiments (see 4, A.4). The results are given in the Table 5.

# of seq. with change point	Method	AUC ↓	Max Covering ↑
350 (original)	InDiD	213.76 ± 10.77	0.9656 ± 0.0012
	BCE	237.94 ± 8.54	**0.9661 ± 0.0006**
	BCE+InDiD	**202.19 ± 2.07**	0.9656 ± 0.0012
250	InDiD	224.73 ± 4.80	**0.9671 ± 0.0029**
	BCE	241.31 ± 2.12	0.9654 ± 0.0038
	BCE+InDiD	**224.52 ± 17.75**	0.9665 ± 0.0031
150	InDiD	**230.67 ± 5.57**	0.9644 ± 0.0034
	BCE	239.384 ± 3.802	**0.9657 ± 0.0039**
	BCE+InDiD	237.29 ± 16.59	0.9644 ± 0.0038
50	InDiD	**237.78 ± 7.96**	0.9618 ± 0.0058
	BCE	245.96 ± 7.79	0.9616 ± 0.0046
	BCE+InDiD	247.35 ± 7.78	**0.9621 ± 0.0046**
25	InDiD	**239.82 ± 11.63**	**0.9565 ± 0.0024**
	BCE	248.54 ± 5.65	0.9563 ± 0.0039
	BCE+InDiD	253.21 ± 13.44	0.9562 ± 0.0023

Table 5: Quality metrics for models trained on the datasets with smaller amount of sequences with change points (# of seq. with CP), while we use 350 sequences without change points for all experiments. ↑ marks metrics we want to maximize, ↓ marks metrics we want to minimize. The results are averaged by 5 runs and are given in the format *mean* ± *std*. Best values are highlighted with bold font.

We see that the performances of all the considered models deteriorate as the amount of "abnormal" data available for training decreases: Area under the Detection Curve increases, Covering decreases. However, the models can still detect change points quite precisely, even when the share of sequences with change points in the training dataset is less than 7%. We also see that our InDiD and BCE+InDiD methods are better than baseline BCE for a different share of "abnormal" data in the train set in terms of the AUC metric.

A.2.3 Computational time comparison. Inference time is a crucial parameter for a model's industrial application. We share the running time of considered approaches for two different datasets in Table 6. The presented time is the mean value over 10 runs. KL-CPD and TSCP suffer from the necessity to compare several subsequences to obtain the predictions for the whole data stream in an online regime. So, their inference time is higher than those of other representation-based methods.

A.3 Datasets details

In the paper, we consider a diverse set of datasets, with two of them introduced for the first time for the problem of supervised change point detection. Summary statistics on the used data are shown in Table 7. More information about data generation and preprocessing one can find below. For simplicity, we call sequences with change points "abnormal" and sequences without change points "normal".

Synthetic datasets. To generate an abnormal sequence, we use two subsequences with randomly selected lengths for every dataset sample. The first subsequence corresponds to the normal state and is sampled from $\mathcal{N}(1, 1)$ in the 1D case or from $\mathcal{N}(\mathbf{1}, I)$ in the 100D case. The second one, abnormal, is from a Gaussian with a randomly selected mean between 2 and 100 (constant vector for the multivariate scenario) and the variance 1 or I. The sequence without change a point is generated similarly to the first subsequence. We use balanced datasets with an equal number of normal and abnormal objects.

	Synthetic 100D	MNIST sequence
Method	Inference time, ms	
BinSeg	6.550 ± 0.018	5.850 ± 0.004
PELT	8.450 ± 0.009	4.000 ± 0.006
Kernel_CPD	4.340 ± 0.003	4.750 ± 0.002
KL-CPD	$(1.700 \pm 0.008) \cdot 10^3$	$(6.520 \pm 0.031) \cdot 10^3$
TSCP	$(2.940 \pm 0.039) \cdot 10^3$	$(6.280 \pm 0.037) \cdot 10^3$
BCE	5.130 ± 0.016	2.660 ± 0.013
InDiD	5.070 ± 0.021	2.650 ± 0.018
BCE+InDiD	5.130 ± 0.021	2.650 ± 0.016

Table 6: CPU inference time for one sequence for considered methods.

Human Activity dataset contains sensors' records for 12 types of human physical activity, such as walking, elevator, going upstairs, running, sleeping, etc., collected from the group of 30 individuals. Each record has a length of 200 and consists of 561 features. The frequency of sensor measurements is 50 Hz. We cut these records to get subsequences with changes in the type of human activity and without them. The length of every subsequence is 20. We also select only 28 features out of 561 corresponding to the "mean" parameters. It is important to note that the activity type frequently changes in this dataset, so the number of normal examples is scarce, even for a small length of 20.

Sequences of MNIST images. To generate data, we obtain latent representations for MNIST images using pretty simple Conditional Variational Autoencoder [27] (CVAE). The utilized architecture has a hidden dimension of 256 and a latent dimension of 75. Then we take two points in the latent space corresponding to a pair of digits and add 62 points from the line connecting them. As a result, we have a sequence with a length of 64 of latent representations of images. After reconstruction via CVAE's decoder, we get a sequence of images. Such an approach allows us to generate images with smooth transitions of digits, even if the first and the last digits differ. We expect these gradual changes increase the difficulty of CPD. There are sequences with (e.g. from 4 to 7) and without (e.g. from 4 to 4) a change point in the result dataset. We generate 1000 sequences of length 64 distributed evenly between normal and abnormal samples. Two examples of obtained sequences are in Figure 6.

Explosions and Road Accidents. The whole dataset [28] consists of real-world 240 × 320 RGB videos with 13 realistic anomaly types such as explosion, road accident, burglary, etc., and normal examples. As we've already mentioned, we use only explosion and road accident videos for our research. The CPD specific requires a change in data distribution. We suppose that explosions and road accidents correspond to such a scenario, while most other types correspond to point anomalies. For example, data, obviously, comes

Table 7: Statistics for datasets used in our experiments.

Dataset	Single sample shape	Train size	Test size	% of sequences with changes in train set	% of sequences with changes in test set	Sequence length
1D Synthetic data	1×1	700	300	48.9	52.7	128
100D Synthetic data	100×1	700	300	48.9	52.7	128
Human Activity Recognition	28×1	3580	1337	83.9	87.1	20
Sequences of MNIST images	$28 \times 28 \times 1$	700	300	51.1	47.3	64
Explosions	$256 \times 256 \times 3$	310	315	50.0	4.8	16
Road Accidents	$256 \times 256 \times 3$	666	349	50.0	14.0	16

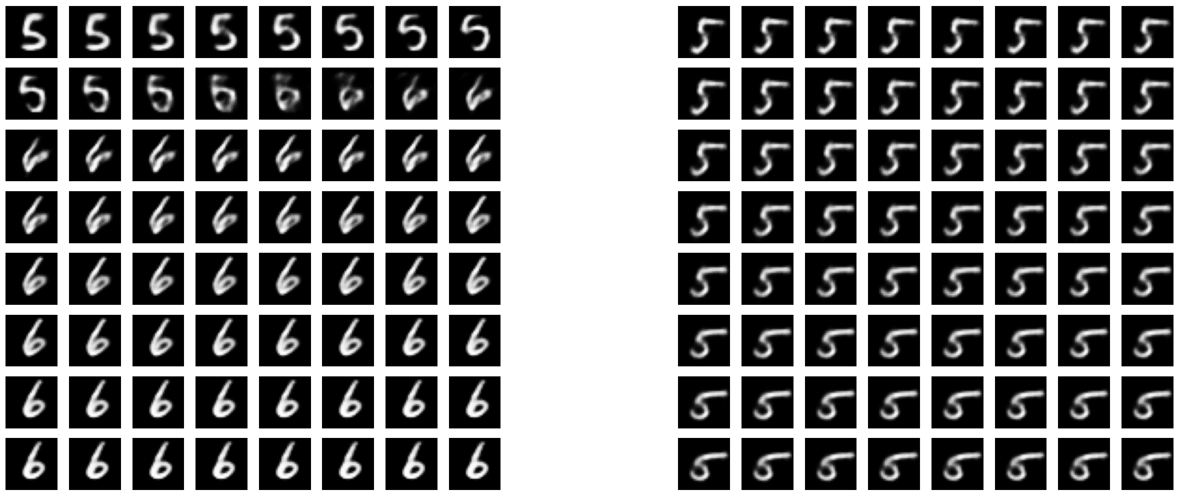
Figure 6: Example of MNIST sequences with (left) and without (right) change point. At each row, a sequence goes from left to right. Then we continue with the most left image in the next row.

.

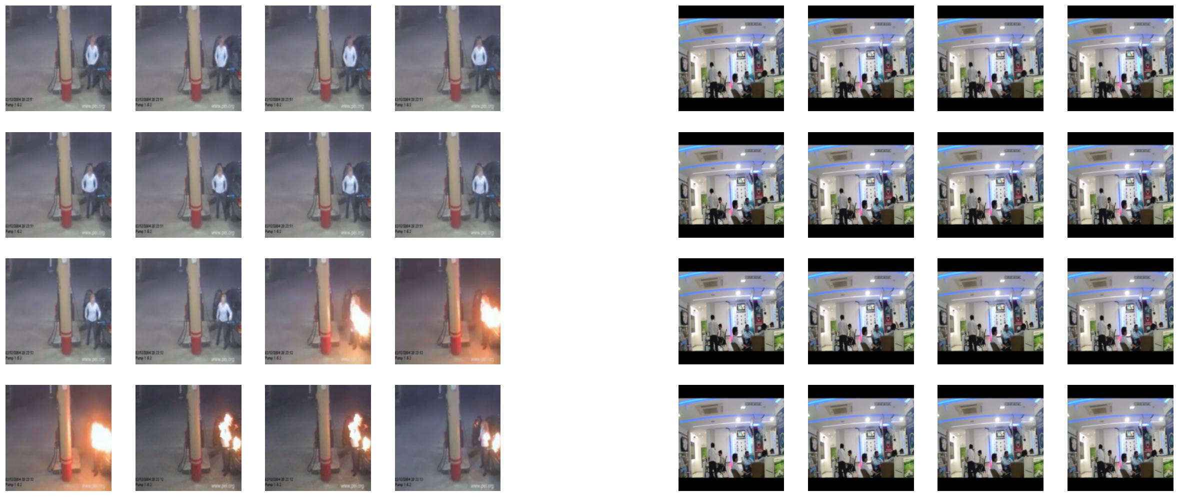
Figure 7: Example from Explosion data with (left) and without (right) change point correspondingly. At each row, a sequence goes from left to right. Then we continue with the most left image in the next row.

.

from a normal regime before the explosion. After it, we can see fire and smoke, which last for some time. Thus, the first moment when an explosion appears is a change point. Along with a volunteer, the authors carefully labelled chosen anomaly types. Their opinions were averaged. We provide the obtained markup and the code and release it for the first time, so other researchers can use it to validate their CPD algorithms[2].

To generate the dataset, we apply methods from torchvision [22] that cut video on small clips (a subsequence of the initial frames' sequences) with a given step and frequency. We use 30 frame per seconds and number of frames in clip 16. For training data, we sample clips with step 5, so there are overlapping sections. For test data, we cut the video on non-overlapping clips. We also transform videos into the shape 256×256 and normalize them. The examples from the obtained data are shown in Figure 7.

A.4 Implementation details

The main implementation details of all the considered CPD methods are summarized in Table 8. We denote window size for KL-CPD and TSCP methods (see [5, 7] for details) as "w.s.", the batch size used for training as "b.s.", learning rate as "l.r." and monitoring parameter for early stopping as "mon".

For video preprocessing in our Explosions and Road Accidents experiments, we use a pre-trained feature extractor X3D [9] that is denoted as "Extractor" in Table 8. We freeze its parameters while training all CPD models.

We mostly follow the original procedures for KL-CPD[3] and TSCP[4] presented in papers and official repositories [5, 7], varying only their hyperparameters. During the application of these methods, we faced several difficulties described below. To treat them, we had to implement some procedures out of the initial research scope. We expect deeper investigations to improve the performance, while they should be addressed in separate papers. In particular, for KL-CPD and TSCP, we highlight:

- KL-CPD method processes a sequence of observations and outputs scores based on MMD statistic that corresponds to the similarity of the distributions of the two consecutive windows. To adapt this method to our setting, that works with change point probabilities, we propose transforming these non-negative MMD-scores into predictions in the following way: $pred = \tanh(a \cdot scores)$. Thus, we scale the outputs to $[0, 1]$ and can interpret them as change probabilities similar to our InDiD method. Constant a is a hyperparameter. We chose the following coefficients for our experiments: $a = 100$ – for Synthetic 1D, 100D and MNIST sequences; $a = 5$ – for Human Activity Recognition; $a = 10^7$ – for Explosions and Road Accidents. In a similar way, TSCP outputs cosine similarity of consecutive windows. So, we use sigmoid function to scale the output, i.e. $pred = \sigma(-scores)$.
- As the KL-CPD approach is based on several recurrent networks that encode and decode the original sequence, it requires many parameters in multidimensional scenarios to decode the embedding back into the original space. Direct applying it to video data turned out to be impossible due to the huge resource necessities. We overcome it by adding Linear layers to encoders and a decoder that make an embedding size small enough to be stored and processed.
- Another challenge was connected with training TSCP on synthetically generated data, such as 1D, 100D and MNIST. Because of our data generation procedure, there are many pretty similar sequences, as we always start from f_0 and end with f_∞. TSCP tries by default to separate these close sequences in the embedding space. Thus, we suppose that such peculiarities negatively affect the method's performance and to make it work in such scenarios, we should change the sampling procedure and the training process in the method.

[2]The data labelling are available at https://anonymous.4open.science/r/InDiD-FD23
[3]https://github.com/OctoberChang/klcpd_code
[4]https://github.com/cruiseresearchgroup/TSCP2

Table 8: Implementation details for considered CPD methods.

Method	Core's architecture	Loss' parameters	Learning parameters	Early Stopping parameters
	1D Synthetic data			
Best classic	Kernel CPD (kernel = linear)	pen = 21	na	na

KL-CPD
(w.s. = 8)
	Net D/G Enc GRU(1, 4)
	Net D Dec GRU(4, 1)
	Net G Dec GRU(1, 4) + Lin(4, 1)
		lambda_ae = 0.1
		lambda_real = 10
		med_sqdist = 1
			b.s = 64
			l.r. = 10^{-3}
				mon = val_mmd2_real
				delta = 10^{-5}
				patience = 5

TSCP
(w.s. = 16)
	TSCP (nb_filters = 16, n_steps = 4
	code_size = 4, kernel_size = 5)
		temp_NCE = 0.5	b.s = 4
			l.r. = 10^{-4}
				mon = train loss
				delta = 10^{-4}
				patience = 5

BCE
InDiD
BCE + InDiD
	LSTM(1, 4, dropout = 0.5) +
	+ Linear(4, 1) + Sigmoid()
		T = 32	b.s = 64
			l.r. = 10^{-3}
				mon = val loss
				delta = 0
				patience = 10

	100D Synthetic data			
Best classic	Kernel CPD (kernel = rbf)	pen = 0.9	na	na

KL-CPD
(w.s. = 8)
	Net D/G Enc GRU(100, 16)
	Net D Dec GRU(16, 100)
	Net G Dec GRU(100, 16) + Lin(16, 100)
		lambda_ae = 0.1
		lambda_real = 10
		med_sqdist = 3
			b.s = 64
			l.r. = 10^{-3}
				mon = val_mmd2_real
				delta = 10^{-5}
				patience = 1

TSCP
(w.s. = 16)
	TSCP (nb_filters = 4, n_steps = 4
	code_size = 4, kernel_size = 5)
		temp_NCE = 0.5	b.s = 4
			l.r. = 10^{-4}
				mon = train loss
				delta = 10^{-4}
				patience = 5

BCE
InDiD
BCE + InDiD
	LSTM(100, 8, dropout = 0.5) +
	+ Linear(8, 1) + Sigmoid()
		T = 32	b.s = 64
			l.r. = 10^{-3}
				mon = val loss
				delta = 0
				patience = 10

	Human Activity Recognition			
Best classic	Kernel CPD (kernel = rbf)	pen = 0.3	na	na

KL-CPD
(w.s. = 3)
	Net D/G Enc GRU(28, 8)
	Net D Dec GRU(8, 28)
	Net G Dec GRU(28, 8) + Lin(8, 28)
		lambda_ae = 0.2
		lambda_real = 1
		med_sqdist = 2
			b.s = 64
			l.r. = 10^{-4}
				mon = val_mmd2_real
				delta = 10^{-5}
				patience = 5

TSCP
(w.s. = 4)
	TSCP (nb_filters = 64, n_steps = 8
	code_size = 4, kernel_size = 5)
		temp_NCE = 0.1	b.s = 8
			l.r. = 10^{-3}
				mon = train loss
				delta = 10^{-4}
				patience = 5

BCE
InDiD
BCE + InDiD
	LSTM(28, 8, dropout = 0.5) +
	+ Linear(8, 1) + Sigmoid()
		T = 5	b.s = 64
			l.r. = 10^{-3}
				mon = val loss
				delta = 0
				patience = 10

	Sequences of MNIST images			
Best classic	Kernel CPD (kernel = linear)	pen = 375	na	na

KL-CPD
(w.s. = 8)
	Net D/G Enc GRU(784, 32)
	Net D Dec GRU(32, 784)
	Net G Dec GRU(784, 32) + Lin(32, 784)
		lambda_ae = 0.001
		lambda_real = 0.01
		med_sqdist = 10
			b.s = 64
			l.r. = $3 \cdot 10^{-4}$
				mon = val_mmd2_real
				delta = 10^{-5}
				patience = 5

TSCP
(w.s. = 16)
	TSCP (nb_filters = 256, n_steps = 128
	code_size = 8, kernel_size = 5)
		temp_NCE = 0.1	b.s = 64
			l.r. = 10^{-3}
				mon = train loss
				delta = 10^{-4}
				patience = 5

BCE
InDiD
BCE + InDiD
	LSTM(784, 32, dropout = 0.25) +
	+ Linear(32, 1) + Sigmoid()
		T = 32	b.s = 64
			l.r. = 10^{-3}
				mon = val loss
				delta = 0
				patience = 10

	Explosions / Road Accidents			

KL-CPD
(w.s. = 8)
	Net D/G Enc Extractor + Lin(12288, 100) +
	+ ReLU() + GRU(100, 16)
	Net D Dec GRU(16, 100) +
	+ Lin(100, 12288) + ReLU()
	Net G Dec GRU(100, 16) +
	+ Lin(16, 12288)
		lambda_ae = 0.1
		lambda_real = 10
		med_sqdist = 50
			b.s = 8
			l.r. = 10^{-4}
				mon = val_mmd2_real
				delta = 10^{-5}
				patience = 5

TSCP
(w.s. = 4)
	Extractor +
	+ TSCP (nb_filters = 256, n_steps = 256
	code_size = 128, kernel_size = 5)
		temp_NCE = 0.1	b.s = 16
			l.r. = 10^{-3}
				mon = train loss
				delta = 10^{-4}
				patience = 5

BCE
InDiD
BCE + InDiD
	Extractor +
	+ LSTM(12288, 64, dropout = 0.5) +
	+ Linear(64, 1) + Sigmoid()
		T = 8	b.s = 16
			l.r. = 10^{-3}
				mon = val loss
				delta = 0
				patience = 10