Neural Processes with Event Triggers for Fast Adaptation to Changes

Paul Brunzema* PAUL.BRUNZEMA@DSME.RWTH-AACHEN.DE
Paul Kruse* PAUL.KRUSE@DSME.RWTH-AACHEN.DE
Sebastian Trimpe TRIMPE@DSME.RWTH-AACHEN.DE
Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Germany

Editors: A. Abate, K. Margellos, A. Papachristodoulou

Abstract

Traditionally, first-principle models are used to monitor and control dynamical systems. However, modeling complex systems using first principles can be challenging. Learning the dynamics from data using neural networks has emerged as a viable alternative. In practice, some parameters of a system may vary across different system instances, but training separate neural networks for all possible parameter combinations can be infeasible. Therefore, meta-learning using, e.g., conditional neural processes (CNPs), aims to learn a prior model over the system dynamics for various parameters. These models can then adapt on deployment to the parameters of a system instance using a context set composed of past observations. However, changes in parameters can also occur *online* during operation and naively adding past observations across parameter variations to the context set can distort the model's latent representation, leading to inaccurate predictions over time. This paper introduces an adaptation scheme to enable CNPs to cope with such online variations. We combine a sliding window to accommodate gradual variations with the use of event triggers to detect sudden changes. The event triggers are based on concentration inequalities, they reset the context set of the CNP once observations deviate significantly from the CNP's predictions. We validate our concepts on two nonlinear dynamical systems under parameter variations and demonstrate that our approaches decrease the prediction error over time as well as their efficacy for control.

Keywords: neural process, time-varying environment, event-triggered learning, meta learning

1. Introduction

Dynamical systems are prevalent across various fields including engineering and biology. Effective monitoring and control of these systems are essential for maintaining operation and ensuring safety. Traditionally, first-principle models have been employed for this task. However, modeling complex systems using first principles can be challenging, especially for nonlinear and noisy systems. Machine learning (ML) approaches, such as neural networks (NNs), have emerged as promising alternatives to first-principle modeling. They directly learn the dynamics from data, bypassing the need for explicit modeling. NNs have proven to be effective in approximating complex systems and have been applied in various control applications (Zhang et al., 2019; Salzmann et al., 2023).

A challenge for model learning in practice is that dynamical systems often have deviations in the parameters between different system instances; consider e.g., two robots with different friction coefficients in the joints. Training separate NNs for each combination of parameters and choosing the one closest to the system instance is inefficient and a waste of compute and data resources. Meta-learning using e.g., conditional neural processes (CNPs), offers a solution to this problem by

* The authors contributed equally to this work.

learning a prior model capturing the core dynamics of a system, which can then adapt to different system instances at deployment using a *context set* composed of past observations (Beck et al., 2023). In real applications, the dynamics of a controlled system will often change. Such changes may be gradual (e.g., wear and tear) or more sudden (e.g., driving dynamics due to different payloads). If data is naively added to the context set across time-variations, this can distort the latent representation of a CNP of the dynamics and thus lead to inaccurate predictions. Poor predictions will harm the performance of a downstream task such as lead to sub-optimal control performance when used in a model predictive controller.

In this paper, we propose an adaptation scheme for CNPs (Garnelo et al., 2018a) (see Figure 1) to obtain accurate predictions across time-variations in the dynamics. The adaptation scheme has two components. First, it utilizes a sliding window, a common technique in ML to deal with time-variations, to obtain a context set that characterizes the current dynamics. We will demonstrate that there is a fundamental trade-off between speed of adaptation and accuracy in the prediction. In particular long windows will yield accurate predictions but lead to slow adaptation. Therefore, second, we employ an event trigger (ET) to detect significant deviations in the system behavior based on predictions of the CNP and observations from the dynamical system. This ET resets the context when necessary, allowing the CNPs to rapidly adapt to new dynamics. We evaluate our approach on two nonlinear dynamical systems and demonstrate that with the ET we achieve uniformly better trade-offs than using only a sliding window alone. While we employ CNPs to illustrate our method, the adaptation scheme can be utilized for other neural processes (NPs) or meta-learning models with a latent context representation. In summary, our main contributions are:

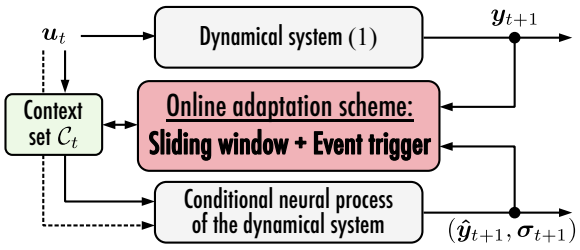

Figure 1: Schematic overview of our adaptation scheme for CNPs.

(i) We demonstrate how to effectively utilize CNPs for online predictions and control of dynamical systems with time-varying parameters.

(ii) An adaptation scheme for CNPs to adapt to online parameter changes composed of a sliding window and an ET.

2. Problem Setting

We want to predict the evolution of a nonlinear discrete time dynamical system over discrete time steps $t \in \mathbb{N}_H := \{1, \dots, H\}$ until a time horizon H. The dynamical system with states $\boldsymbol{s}_t \in \mathbb{R}^D$, observation $\boldsymbol{y}_t \in \mathbb{R}^D$ as noisy state measurements, and control inputs $\boldsymbol{u}_t \in \mathbb{R}^U$ is given as

$$\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{u}_t, \boldsymbol{p}_t) + \boldsymbol{v}_t, \qquad \boldsymbol{y}_t = \boldsymbol{s}_t + \boldsymbol{w}_t \tag{1}$$

with unknown time-varying parameters $\boldsymbol{p}_t \in \mathcal{P} \subseteq \mathbb{R}^P$, and $\boldsymbol{v}_t$ and $\boldsymbol{w}_t$ are process and measurement noise, respectively. We assume that the dynamics f are unknown, but we have a data set $\mathcal{S}$ with trajectories from different system realizations, i.e. different $\boldsymbol{p}_t$. Importantly, the parameters $\boldsymbol{p}_t$ are unknown and cannot be measured during inference time. For the training data, we assume that the parameters stay constant for each trajectory. In practice, such setting could be ensured by controlling the (operating) conditions during data recording in a design phase. The data set can be used to train a model of the system dynamics obtaining predictions $\hat{\boldsymbol{y}}_t \in \mathbb{R}^D$ using past outputs and control

inputs from the current episode as $\mathcal{D} = \{(\boldsymbol{y}_{i+1}, \boldsymbol{y}_i, \boldsymbol{u}_i)\}_{i=1}^{t}$. The prediction error as the difference between observation and prediction at time step t is then $\boldsymbol{e}_t \coloneqq \hat{\boldsymbol{y}}_t - \boldsymbol{y}_t$.

Problem Statement. Our objective is to minimize the cumulative prediction error over the time horizon H at test time. Over this time horizon, the parameters $\boldsymbol{p}_t$ governing the system dynamics may change. Since the parameters can not be measured, we seek to develop an adaptation scheme to identify these changes to enable fast adaptation of our prediction model to the system changes.

3. Learning a Prior over Dynamical Systems using Conditional Neural Processes

We first describe how to obtain the CNP prediction model of the dynamical system shown in Figure 1 from the data set $\mathcal{S}$. In Section 4, we will then describe our adaptation scheme for updating the context set of this CNP to account for online time-variations in the parameters.

As discussed, we use a CNP to learn a prior model over the system dynamics in (1). CNPs were introduced by Garnelo et al. (2018a) to model stochastic processes. They use a context set $\mathcal{C}$ consisting of input/output pairs and a target set $\mathcal{T}$ of unlabeled input points. The output of a CNP are posterior predictions at the points in the target set given the context set; with this, they behave similar to Gaussian processes. While CNPs were the first introduced neural processes (NPs), it has recently been shown that they can still perform competitively with newer NP variants (Bruinsma et al., 2023). We will focus on CNPs but our adaptation scheme could also be applied to other variants from the NP family (Garnelo et al., 2018b; Kim et al., 2019; Gordon et al., 2020), or any other meta learning model with a latent space that allows for a context set of varying size.

To model the dynamical system in (1) where the next state depends both on the current state as well as control input we further define $\tilde{\boldsymbol{x}}_{t-1} \coloneqq [\boldsymbol{y}_{t-1}^\top, \boldsymbol{u}_{t-1}^\top]^\top \in \mathbb{R}^{D+U}$ for convenience in notation as the inputs. As outputs, we define $\tilde{\boldsymbol{y}}_t \coloneqq \boldsymbol{y}_t$. Hence, our context set is composed of tuples of the form $(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{y}}_t) \in \mathcal{C}_t$. We make the context set depend on the current time step t since we obtain additional input and output pairs online. Usually, we are only interest in predicting the next state and therefore our target set is, if not explicitly stated otherwise, defined as $\mathcal{T}_t = \{\tilde{\boldsymbol{x}}_t\}$. The CNP then consists of three main building blocks (see also Figure 2 for an illustration of the architecture):

(i) The encoder network $\boldsymbol{r}_i = \texttt{Enc}_\phi(\tilde{\boldsymbol{x}}_{i-1}, \tilde{\boldsymbol{y}}_i)$ that computes a latent representation for all input and output pairs in the context set $(\tilde{\boldsymbol{x}}_{i-1}, \tilde{\boldsymbol{y}}_i) \in \mathcal{C}_t$.

(ii) The aggregator $\boldsymbol{R}_t = \frac{1}{C}\sum_{i=1}^{C} \boldsymbol{r}_i$ to deal with a varying set size where $C = |\mathcal{C}_t|$.

(iii) The decoder $(\hat{\boldsymbol{y}}_{t+1}, \boldsymbol{\sigma}_{t+1}) = \texttt{Dec}_\psi(\tilde{\boldsymbol{x}}_t, \boldsymbol{R}_t)$ that generates independent location and scale parameters for a Gaussian distribution, respectively, for points in the target set $\tilde{\boldsymbol{x}}_t \in \mathcal{T}_t$.

We denote the parameters of the whole CNP as $\theta \coloneqq \phi \cup \psi$. With these components, one can state the predictive distribution of the CNP for the next observation as $p_\theta(\tilde{\boldsymbol{y}}_{t+1} \mid \mathcal{C}_t, \tilde{\boldsymbol{x}}_t) = \mathcal{N}(\hat{\boldsymbol{y}}_{t+1}, \boldsymbol{\sigma}_{t+1})$.

We train the CNP by minimizing the negative log-likelihood at target and context points (Garnelo et al., 2018a). For this, we use two trajectories from the data set $\mathcal{S}$ that were generated by the same parameter combination. Next, we construct the input and output pairs as discussed above for each trajectory. We then sample a number of context points uniformly as $N \sim \mathcal{U}(1, N_{\max})$ from the first trajectory, where $N_{\max}$ is the maximum context set size. As the target set, we use the whole second trajectory. Letting Q be the distribution of all possible dynamical system captures in $\mathcal{S}$ and let $\mathcal{G}_t$ be the set of ground truth observations, we can define the loss as

$$L(\theta) = -\,\mathbb{E}_Q\left[\mathbb{E}_N\left[\ln p_\theta\left(\mathcal{G}_t \mid \mathcal{C}_t, \mathcal{T}_t\right)\right]\right] = -\,\mathbb{E}_Q\left[\mathbb{E}_N\left[\ln \prod_{i=1}^{|\mathcal{T}_t|} \mathcal{N}(\boldsymbol{y}_i \mid \hat{\boldsymbol{y}}_i, \boldsymbol{\sigma}_i)\right]\right]. \tag{2}$$

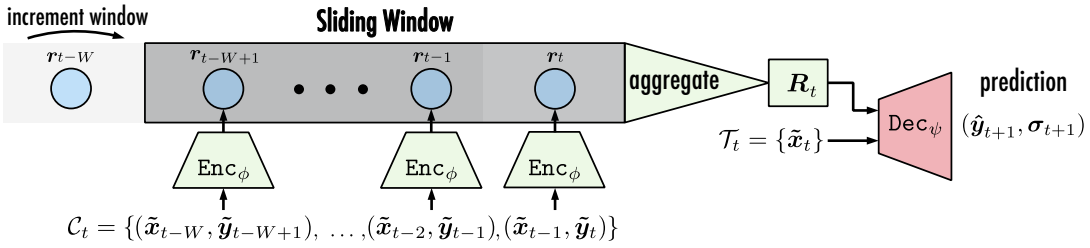

Figure 2: The proposed sliding window of size W for CNPs of dynamical system. The sliding window increments at each time step to obtain a meaningful context set and therefore latent representation that approximates the current systems behavior.

Note that during training, we consider multiple points in our target set as well as a factored structure of the conditional probability to satisfy the conditions for modeling a stochastic process (cf. Garnelo et al. (2018a)). Varying the size of the context set during training is essential for our adaptation scheme for CNPs. Both the moving window and the ET will result in potentially different context sizes over time and for the final performance it its crucial that this is considered during training.

Once trained on data from the dynamical system in (1) as described in this section, a CNP can be deployed onto a specific instance of the system. By incorporating past observations and control inputs into the context set, the CNP will implicitly infer the system's parameters. This enables the CNP to be utilized for monitoring the system or controlling it through a model predictive controller. However, during operation, some of the system parameters might change. Adding data across such time-variations in the parameters to the context set can distort the CNP's latent representation of the dynamics and thus lead to inaccurate predictions or sub-optimal control performance.

4. Adapting to Online Changes Using a Sliding Window and Event Triggers

To obtain accurate predictions through time-variations, we now propose our adaptation scheme (red in Figure 1) consisting of using a sliding window to continuously update the context set of the CNP, and an ET to detect and react to sudden changes (later summarized in Algorithm 1). We will begin by introducing the sliding window mechanism for the CNP's context set, followed by the ET.

4.1. Sliding Window for Continuous Adaptation

Sliding window approaches are common in control as well as machine learning. For CNPs, we fill up the context set after initialization until a set size of W. This is enabled by the architecture of the CNP as the aggregation operation (*(ii)* in Section 3) allows for different context set sizes. The context vector $\boldsymbol{R}_t$ of the CNP is updated every time step using information from the previous observation and control input as (see also Figure 2) $\boldsymbol{r}_t = \texttt{Enc}_\phi(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{y}}_t)$ followed by the aggregation $\boldsymbol{R}_t = \frac{1}{W}\sum_{i=1}^{W} \boldsymbol{r}_{t-i+1}$. The window size directly influences the accuracy of the CNPs predictions. A sliding-window approach enables the CNP to adapt to changes in the system dynamics, including both gradual and sudden changes as $\boldsymbol{R}_t$ is continuously updated. However, we will demonstrate in Section 5, that there is a fundamental trade-off between prediction accuracy and speed of adaptation. Specifically, we will see that for larger W, the prediction is more accurate (see Figure 3); however,

adapting to changes takes longer. Contrary, a short window size W allows the CNP to adapt quickly to a new parameter configuration, but this comes at the cost of less accurate predictions.

4.2. Event Triggers for Detecting Sudden Changes

To obtain both good predictions and fast adaptation, we further introduce the usage of an ET. The idea of such an ET is to detect sudden changes in the system dynamics based on the prediction error of the CNP. Once such changes are detected, we reset the context set and context vector of the CNP. For the design of the ETs, we assume a perfect prediction model, and with this, deviations between predictions and observations are independent and identically distributed Gaussian random variables. This is a strong assumption on the models performance and highly depends on how the model was trained, if the dataset $\mathcal{S}$ is sufficient to model the dynamics, and so on. However, we use it to derive ETs based on theoretical insights from error bounds on such Gaussian random variables and demonstrate that these ETs are useful in practice. We employ concentration inequalities that incorporate the model's uncertainty estimates to design our ETs. In practice, we expect the model to be more uncertain in parts of the state-space with limited data increasing the bounds. We introduce two kinds of ETs: channel-based ETs, i.e. ETs that consider each dimension of the prediction error separately, and ETs that consider the Euclidean norm of the prediction error.

4.2.1. CHANNEL-BASED EVENT TRIGGER

First, we will introduce the channel-based ETs. As stated, we name them 'channel-based' as they consider each entry dimension of the prediction error $\boldsymbol{e}_t = [e_{t,0}, \dots, e_{t,D-1}]^\top$ separately. For this, we first derive bounds on the absolute value of the prediction error in *one* channel. These concentration inequalities guarantee that the absolute value will remain below a certain threshold in probability. Thus, by design, the bounds yield a bound of the false positive rate. By leveraging these results, we can design our ETs at the end of this subsection.

Lemma 1 (One-step error bound) *Let $e_{t,i}$ be the zero mean Gaussian prediction error with variance $\sigma^2_{t,i}$ at time step t. Pick $\delta \in (0,1)$ and set $h_t = \sqrt{2\sigma^2_{t,i} \log \frac{1}{\delta}}$. Then, $\mathbb{P}\{|e_{t,i}| \leq h_t\} \geq 1 - \delta$.*

Proof This follows directly from standard concentration results for normal distributions (e.g., Srinivas et al. (2010)). It uses the fact that if $e_{t,i} \sim \mathcal{N}(0,1)$, then $\mathbb{P}\{e_{t,i} > h_t\} \leq 1/2e^{-h_t^2/2}$. Scaling with $\sigma^2_{t,i}$, considering a two sided bound, and solving the right hand side for δ yields the result. ■

Lemma 1 only holds for one time step. If we consider multiple time steps, the probability of drawing an outlier from the normal distribution increases. Hence, the threshold h_t has to increase over time to still bound the prediction error in one channel with probability $1 - \delta$. Following the approach of Srinivas et al. (2010) we can transform Lemma 1 to the so-called time-uniform case, i.e. a concentration inequality that is satisfied for all time steps.

Lemma 2 (Time-uniform error bound) *Let $e_{1,i}, \dots, e_{t,i}$ be t consecutively observed zero mean Gaussian prediction errors with individual variances $\sigma^2_{t,i}$. Pick $\delta \in (0,1)$. Set $h_t = \sqrt{2\sigma^2_{t,i} \log \frac{\pi_t}{\delta}}$, where $\sum_{t \geq 1} \pi_t^{-1} = 1$, $\pi_t > 0$. Then the following holds: $\mathbb{P}\{|e_{t,i}| \leq h_t, \forall t \geq 1\} \geq 1 - \delta$.*

Proof Using the same approach as in (Srinivas et al., 2010, Lemma 5.1) by setting the right side in the proof of Lemma 1 to $\exp(-h_t^2/(2\sigma^2_{t,i})) = \delta/\pi_t$ and solving for h_t proves the claim. ■

A popular choice for π_t is $\pi_t = \pi^2 t^2/6$ (Srinivas et al., 2010). Based on these results, we design our channel-based ETs to detect sudden changes in the system dynamics: Given a $\delta \in (0, 1)$, we can use Lemma 1 or Lemma 2 to compute a threshold which is then used in the ET. As soon as one of the channels violated the threshold, i.e. the error in one dimension violates the threshold, we reset the context set and context vector of the CNP.

Definition 3 (Channel-based Event Trigger) *Given threshold h_t that may depend on time following either Lemma 1 or Lemma 2 with $\delta \in (0, 1)$, we define the channel-based ET as*

$$\gamma_{\text{reset}} = 1 \iff \bigvee_{i=0}^{D-1} |e_{t,i}| > h_t \tag{3}$$

where γ_{reset} is the binary indicator for whether to reset the context ($\gamma_{\text{reset}} = 1$) or not ($\gamma_{\text{reset}} = 0$).

We refer to the ET following Definition 3 using Lemma 1 and Lemma 2 as `CT(`δ`)` and `UCT(`δ`)`, respectively, where δ specifies the tightness of the concentration inequalities. The presented channel-based ETs consider by design each dimension independently. Next, we propose two ETs based on the norm of the prediction error $\boldsymbol{e}$.

4.2.2. EUCLIDEAN NORM EVENT TRIGGER

To account for the fact that, in practice, the error in each dimension may be correlated, we now consider bounds on the Euclidean norm $\|\cdot\|_2$ of the prediction error. Similar to the last section, we again first derive a one-step error bound followed by a time-uniform error bound. At the end of the subsection, we will then derive our norm-based ETs from these results.

Lemma 4 (One-step norm error bound) *Let $\boldsymbol{e}_t \in \mathbb{R}^D$ be a zero mean Gaussian prediction error with covariance matrix $\Sigma_t \in \mathbb{R}^{D\times D}$. Pick $\delta \in (0, 1)$ and set $h_t = \sqrt{8 \operatorname{Tr}(\Sigma_t) \log(\frac{\sqrt{e}}{\delta})}$. Then, the following holds: $\mathbb{P}\{\|\boldsymbol{e}_t\|_2 \leq h_t\} \geq 1 - \delta$.*

Proof From Pinelis and Sakhanenko (1986, Corollary 3) we have for Gaussian random vectors that

$$P\left\{\|X\| - \mathbb{E}[\|X\|] \geq \bar{h}\right\} \leq \exp\left(-\frac{\bar{h}^2}{2\,\mathbb{E}[\|X\|^2]}\right) \qquad \text{for } \bar{h} \geq 0. \tag{4}$$

Following Pinelis (2023) of setting $\bar{h} := h_t - \mathbb{E}[\|X\|]$ and multiplying the RHS by $\sqrt{e}$ we obtain

$$P\{\|X\| \geq h_t\} \leq \sqrt{e} \exp\left(-\frac{h_t^2}{8\,\mathbb{E}[\|X\|^2]}\right) \qquad \text{for } h_t \geq 0 \tag{5}$$

using the fact that $h_t - \mathbb{E}[\|X\|] \geq h_t/2$. By design, note that if $h_t^2 < 4\,\mathbb{E}[\|X\|^2]$, the upper bound in (5) is > 1 and therefore trivial. We can therefore state without loss of generality that $h_t^2 \geq 4\,\mathbb{E}[\|X\|^2] \geq 4\,\mathbb{E}[\|X\|]^2$ so that $\mathbb{E}[\|X\|] \leq h_t/2$ ensuring $\bar{h} > 0$ and proving $h_t - \mathbb{E}[\|X\|] \geq h_t/2$. If $X \sim \mathcal{N}(0, \Sigma)$, then for the Euclidean norm we have $\mathbb{E}[\|X\|^2] = \mathbb{E}[\sum_{i=1}^N X_i^2] = \sum_{i=1}^N \mathbb{E}[X_i^2] = \sum_{i=1}^N \sigma_i^2 = \operatorname{Tr}(\Sigma)$. Setting the right hand side of (5) to δ and solving for h_t proves the claim. ■

Lemma 5 (Time-uniform norm error bound) *Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t \in \mathbb{R}^D$ be t consecutively observed zero mean Gaussian prediction errors with individual covariance matrices $\Sigma_t \in \mathbb{R}^{D\times D}$. Pick $\delta \in (0, 1)$ and set $h_t = \sqrt{8 \operatorname{Tr}(\Sigma_t) \log(\frac{\sqrt{e}\pi_t}{\delta})}$, where $\sum_{t\geq 1} \pi_t^{-1} = 1$, $\pi_t > 0$. Then,*

$$\mathbb{P}\{\|\boldsymbol{e}_t\|_2 \leq h_t,\ \forall t \geq 1\} \geq 1 - \delta. \tag{6}$$

Proof We can use the same argumentation as in the proof of Lemma 2 (based on arguments from Srinivas et al. (2010)) and set the upper bound in (5) to δ/π_t and solve for h_t to prove the claim. ■

Different to Definition 3, we now consider the combined prediction error at every time step: Given a $\delta \in (0, 1)$, we can use either Lemma 4 or Lemma 5 to compute a threshold. As soon as the Euclidean norm of the prediction error exceeds this threshold, we reset the context set of the CNP.

Definition 6 (Euclidean Norm Event Trigger) *Given threshold h_t that may depend on time following either Lemma 4 or Lemma 5 with $\delta \in (0, 1)$, we define the Euclidean norm ET as*

$$\gamma_{\text{reset}} = 1 \iff \|\boldsymbol{e}_t\|_2 > h_t \tag{7}$$

where γ_{reset} is the binary indicator for whether to reset the context (γ_{reset} = 1) or not (γ_{reset} = 0).

We refer to the ET following Definition 3 using Lemma 1 and Lemma 2 as $\texttt{NT}(\delta)$ and $\texttt{UNT}(\delta)$, respectively, where δ specifies the tightness of the respective concentration inequalities.

Our adaptation scheme consisting of a sliding window and an ET is summarized in Algorithm 1. In the next section, we will empirically test this scheme on two nonlinear dynamical systems to test if it improves the prediction performance of our CNP across time-variations.

Algorithm 1 Adaptation Scheme for CNPs

Input: Trained CNP, W, H, event trigger (Def. 3 or Def. 6) with $\delta \in (0, 1)$, controller

for $t = 1, \ldots, H$ **do**
  Observe $\boldsymbol{y}_t$ from dynamical system
  if $t \geq 2$ **then**
    $\gamma_{\text{reset}} \leftarrow \text{EVENT TRIGGER}(\boldsymbol{y}_t, (\hat{\boldsymbol{y}}_t, \boldsymbol{\sigma}_t), t)$
    if γ_{reset} **then**
      $\mathcal{C}_t \leftarrow \{(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{y}}_t)\}$
    else
      $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1} \cup \{(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{y}}_t)\}$
      if $t > W$ **then**
        $\mathcal{C}_t \leftarrow \mathcal{C}_t \setminus \{(\tilde{\boldsymbol{x}}_{t-W-1}, \tilde{\boldsymbol{y}}_{t-W})\}$
    end
  Apply $\boldsymbol{u}_t$ from controller to dynamical system
  Update prediction target $\tilde{\boldsymbol{x}}_t \leftarrow [\boldsymbol{y}_t^\top, \boldsymbol{u}_t^\top]^\top$
  $(\hat{\boldsymbol{y}}_{t+1}, \boldsymbol{\sigma}_{t+1}) \leftarrow \text{CNP}(\mathcal{C}_t, \tilde{\boldsymbol{x}}_t)$
end

5. Numerical Experiments

We evaluate our adaptation scheme on two example dynamical systems: the Van-der-Pol oscillator (VDP) in Section 5.1 and a cart-pole system (CP) in Section 5.2.[1] For both, we choose the encoder (*(i)* in Section 3) as an MLP with 2 hidden layers of size 256 and the decoder (*(iii)* in Section 3) as an MLP with 4 hidden layers with size 128. The current state is further encoded by an one layer MLP of size 128. The latent representation is of size 128. We encode absolute angles with $\sin, \cos$, normalize the data in the network and use a global residual connection to learn a difference for numerical stability as recommended by Deisenroth and Rasmussen (2011). This results in a CNP that directly maps one state to the next. We train the model over 80 epochs using Adam (Kingma and Ba, 2015). As initial learning rate, we use $\text{lr} = 0.001$ for the VDP and $\text{lr} = 0.0005$ for the CP and multiply it by 0.2 in epochs $\{40, 65, 70, 75\}$. The batch size is 64 and we use gradient clipping at 0.5. The training data consists of randomly sampled transitions (50%) and initial state rollouts with random inputs (50%). For all simulation, we use a delta time of 0.02s. As baseline, we add the trained CNP with an infinite window size ($\texttt{w}\infty$). This baseline can not adapt to changes.

1. The full code base can be found here: https://github.com/NKPmedia/np-et-learn.

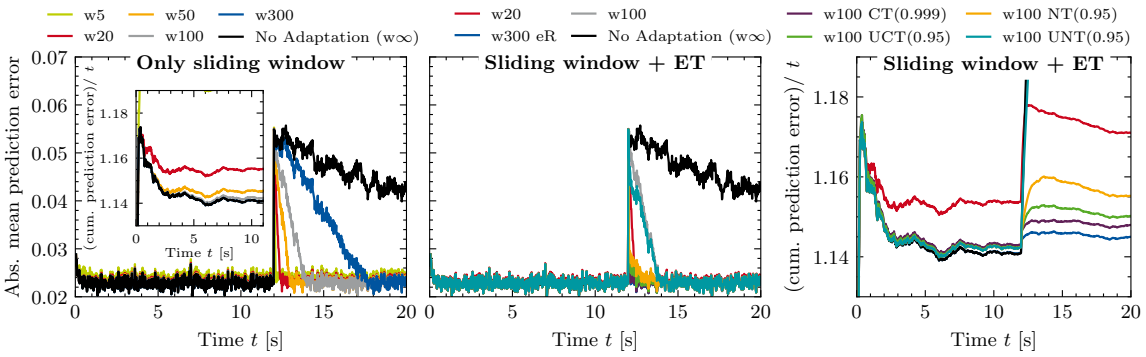

Figure 3: Results on the Van-der-Pol oscillator. Absolute pred. error for different window sizes (left), and combining a sliding window with the proposed event triggers (middle, right).

5.1. Adaptive CNP for the Van-der-Pol Oscillator

First, we consider the two dimensional nonlinear VDP. Here, μ is the only free parameter and we set its parameter range to $\mu \in [0.1, 2]$. In our experiments, we consider 300 different initial states, apply random control inputs to the system, and induce Gaussian zero mean measurement noise with $\sigma_w^2 = 0.001$ and state-dependent process noise with $\sigma_v^2 = 0.01 \cdot |s_t - s_{t-1}|$, respectively. At $t = 12$, we induce a random system change by adding ± 0.5 to the current μ and clipping it to the initial range. We first look at the influence of different sizes W of the sliding window. The results in Figure 3 (left) show the mean prediction error and verify the statements made in Section 4. A larger window size improves the prediction error of the CNP. This effect stagnates for very large W. We can further observe that large window sizes take significantly longer to adapt to the new system parameters, as it takes longer to overwrite the context set. To have both, a small prediction error using a large sliding window and fast adaptation to changes, we proposed various ETs. The different ETs combined with a sliding window of 100 are compared in Figure 3 (middle, right). Here, `w300 eR` is the baseline with $W = 300$ and an exact reset at the time of change as the best possible performance given our CNP. We can observe that considering the cumulative prediction error `CT` performs best. However, note that before the change `CT` is outperformed by the other triggers, indicating multiple false positives by `CT`. Given the low-dimensional system and only one free parameter, these false positives are not significant to the cumulative error. The norm based triggers are more conservative and some changes go undetected. Noteworthy is that `UNT` does not detect any changes resulting in a performance equivalent to only using a sliding window.

5.2. Adaptive CNP for the Cart-Pole System

Next, we consider the four dimensional cart-pole system which is a classic benchmark in control and reinforcement learning. We use the model as described by Green (2020). Here, the free parameters of the system and their ranges are as follows: $m \in [0.1, 0.5]$ is the mass of the pole, $\ell \in [0.1, 0.5]$ is the length of the pole, $M \in [0.5, 2]$ is the mass of the cart, and we set $\mu_c = 0.01$ and $\mu_p = 0.001$ for the friction of the cart and the pole, respectively (all parameters are in SI units). We use the same experimental setup as before, but adding ± 0.1 to M, m, ℓ. Figure 4 summarizes the results. We see the same tendencies for different window sizes; however, the differences are more distinct for this higher dimensional system with more free parameters. We can further observe that the

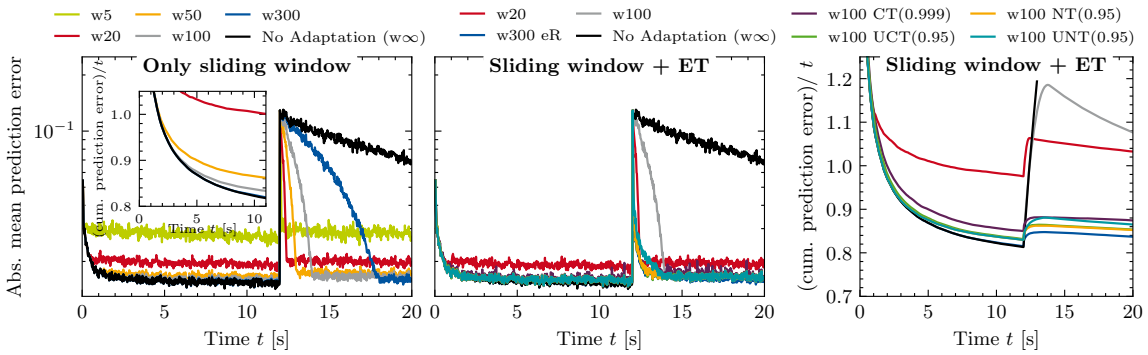

Figure 4: Results on the cart-pole system. Absolute prediction error for different window sizes (left) and combining a sliding window with the proposed event triggers (middle, right).

false positives of CT are now more significant for the cumulative mean prediction error. The more conservative variant UCT as well as the norm based ETs outperform CT.

Cart-Pole Swing-up Using MPPI. To demonstrate the usefulness of an adaptive meta-learning model for control, we use the trained CNP in a model predictive path integral (MPPI) controller (Williams et al., 2017) to swing up the pole. We perform five consecutive swing-ups (Run 1-5) and the parameters in the system change before starting the last one, i.e. before Run 5. We combine the NT that performed best for this system with a window size of 300. Per run, we used 300 random seeds for the initial action sequence of MPPI. The number of upright poles achieved across the random seeds during the runs are shown in Figure 5. Note that until Run 5, the blue, yellow, violet, and black curve lay on top of each other. We observe that using a longer window spanning across different runs, MPPI achieves more swing-ups due to more accurate predictions of the dynamics. After the change, our method with the ET performs only slightly worse than the exact reset; both outperform w20. Using only a long sliding window, it takes a considerable amount of time to update the context set and regain accurate predictions, hindering the ability to swing up the pole effectively.

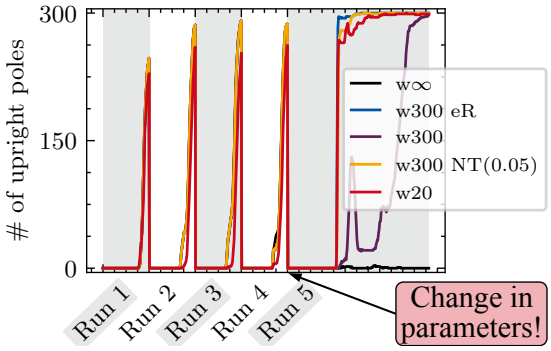

Figure 5: Consecutive swing ups with MPPI.

6. Related Work

While we discussed and validated our adaptation scheme using CNPs as a prediction model, our approaches can also be used for different predictors that use a latent representation that it allows for varying context sizes. Next, we discuss related work to our approaches presented in this paper.

Neural processes for dynamical systems. Galashov et al. (2019) used an attention extended NP for also predicting the dynamics of a cart pole. Cart-pole dynamics were also considered in Wang and Van Hoof (2020) with a doubly stochastic latent NP with global and local stochastic latent variables. In contrast to our work, both did not consider data arriving sequentially as well as online changes in

the parameters. Still, our adaptation scheme could also be applied to their models. Norcliffe et al. (2021) proposed a NP architecture designed to model ODEs based on neural ODEs (Chen et al., 2018). They consider various dynamical systems, but also no online parameter changes.

Meta-learning models for dynamics. Rakelly et al. (2019) use a model similar to CNPs to generate a latent representation based on state transitions for a soft actor critic agent. In contrast to our concepts, they use all past transitions as the context set, allowing for no adaption capabilities after changes. Wang and Van Hoof (2022) use a graph neural network to encode the context allowing for flexible window sizes and preservation of dependencies; hence our adaptation scheme could also be applied. Our scheme could also be applied to GP based meta-learning models as in Sæmundsson et al. (2018). For the work in (Lee et al., 2020; Fu et al., 2021) this is more challenging as they use an encoder with a fixed input size. Applying our ETs would require, e.g., padding with zeros. For gradient and RNN-based meta-learning, a sliding window has also been used by Nagabandi et al. (2018) building on top of (Santoro et al., 2016; Finn et al., 2017). Similarly, Kaushik et al. (2020) propose multiple initial network weights and environment encodings as well as combining a sliding window with regular resets. None of these approach consider an ET to adapt to changes.

Adapting to changes online. Several approaches have been proposed for adapting to changes online for dynamical systems. McKinnon and Schoellig (2017) use a model library approach with different GPs, explicitly selecting the model that best fits the current data. Using NPs, this model selection is performed implicitly by conditioning on the context set. Also using GPs, Meier and Schaal (2016) and Ordóñez-Conejo et al. (2022) employed a sliding window for online model learning and adaptive filtering, respectively. Using an ET to detect changes and adapt to them is inspired by *event-triggered learning* where the aim is to re-learn a model or add data to the model only when necessary (Solowjow and Trimpe, 2020; Brunzema et al., 2023; Umlauft and Hirche, 2019). Our ETs determine when to reset the context set of the CNP and with this implicitly start a re-learning. An ET is also similar in spirit to Bayesian online change point detection (Adams and MacKay, 2007; Agudelo-España et al., 2020; Altamirano et al., 2023). In contrast to these approaches, the model parameters of our CNP remain fixed through time and the ET acts solely as a peripheral for fast adaptation. This makes our approach computationally more efficient as evaluating the ETs is a simple comparison and updating the posterior for CNPs is a forward pass through the encoder compared to updating posterior predictive distributions. It is therefore also suitable for online control tasks. This efficiency however comes at the cost of being less robust to outliers due to the thresholding of the ETs. Interesting future work would be to merge ideas from both spectra by considering multi-step ETs. Also adaptive control (Åström and Wittenmark, 2013) aims to cope with parameter variation. In this context, also meta-learning models have been considered (Richards et al., 2021; Shi et al., 2021). To the best of our knowledge, this is the first work proposing to combine meta-learning models with explicit ETs to detect and react to online parameter changes.

7. Concluding Remarks

We presented a method to enable fast reaction to changes. For this, we used a CNP to learn a prior model over non-linear, parameter-varying systems. To maintain accurate predictions though online time-variations of these parameters, we proposed an adaptation scheme for CNPs consisting of a sliding window, to adapt to slow changes, and an ET, to detect and react to sudden changes. Important for training the CNP model were trajectories where $\boldsymbol{p}_t$ remained constant. Extending this with approaches that can deal with *in-trajectory* changes such as Harrison et al. (2020) is promising.

Acknowledgments

We thank A. von Rohr, F. Solowjow, and H. Hose for insightful discussions, and the anonymous reviewers for their valuable feedback. This work is partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)–RTG 2236/2 (UnRAVeL). Simulations were performed with computing resources granted by RWTH Aachen University under project thes1455.

References

Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

Diego Agudelo-España, Sebastian Gomez-Gonzalez, Stefan Bauer, Bernhard Schölkopf, and Jan Peters. Bayesian online prediction of change points. In *Conference on Uncertainty in Artificial Intelligence*, pages 320–329. PMLR, 2020.

Matias Altamirano, François-Xavier Briol, and Jeremias Knoblauch. Robust and scalable bayesian online changepoint detection. In *International Conference on Machine Learning*, pages 642–663. PMLR, 2023.

Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

Wessel Bruinsma, Stratis Markou, James Requeima, Andrew Y. K. Foong, Tom Andersson, Anna Vaughan, Anthony Buonomo, Scott Hosking, and Richard E Turner. Autoregressive conditional neural processes. In *The Eleventh International Conference on Learning Representations*, 2023.

Paul Brunzema, Alexander von Rohr, Friedrich Solowjow, and Sebastian Trimpe. Event-triggered time-varying Bayesian optimization. *arXiv preprint arXiv:2208.10790*, 2023.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pages 465–472, 2011.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Haotian Fu, Hongyao Tang, Jianye Hao, Chen Chen, Xidong Feng, Dong Li, and Wulong Liu. Towards effective context for meta-reinforcement learning: an approach based on contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Alexandre Galashov, Jonathan Schwarz, Hyunjik Kim, Marta Garnelo, David Saxton, Pushmeet Kohli, SM Eslami, and Yee Whye Teh. Meta-learning surrogate models for sequential decision making. *arXiv preprint arXiv:1903.11907*, 2019.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018b.

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020.

CD Green. Equations of motion for the cart and pole control task. *Sharpneat*, 2020. URL https://sharpneat.sourceforge.io/research/cart-pole/cart-pole-equations.html.

James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *Advances in neural information processing systems*, 33:17571–17581, 2020.

Rituraj Kaushik, Timothée Anne, and Jean-Baptiste Mouret. Fast online adaptation in robotics through meta-learning embeddings of simulated priors. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5269–5276. IEEE, 2020.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 5757–5766. PMLR, 2020.

Christopher D McKinnon and Angela P Schoellig. Learning multimodal models for robot dynamics online with a mixture of Gaussian process experts. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 322–328. IEEE, 2017.

Franziska Meier and Stefan Schaal. Drifting Gaussian processes with varying neighborhood sizes for online model learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 264–269. IEEE, 2016.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2018.

Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. Neural ODE processes. In *International Conference on Learning Representations*, 2021.

Alejandro J Ordóñez-Conejo, Armin Lederer, and Sandra Hirche. Adaptive low-pass filtering using sliding window Gaussian processes. In *European Control Conference*, pages 2234–2240. IEEE, 2022.

Iosif Pinelis. Gaussian concentration inequality. MathOverflow, 2023. URL https://mathoverflow.net/q/351639. (version: 2020-02-24).

Iosif F Pinelis and Aleksandr Ivanovich Sakhanenko. Remarks on inequalities for large deviation probabilities. *Theory of Probability & Its Applications*, 30(1):143–148, 1986.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

SM Richards, N Azizan, J-JE Slotine, and M Pavone. Adaptive-control-oriented meta-learning for nonlinear systems. In *Robotics science and systems*, 2021.

S Sæmundsson, K Hofmann, and MP Deisenroth. Meta reinforcement learning with latent variable gaussian processes. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, volume 34, 2018.

Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2016.

Guanya Shi, Kamyar Azizzadenesheli, Michael O'Connell, Soon-Jo Chung, and Yisong Yue. Meta-adaptive nonlinear control: Theory and algorithms. *Advances in Neural Information Processing Systems*, 34:10013–10025, 2021.

Friedrich Solowjow and Sebastian Trimpe. Event-triggered learning. *Automatica*, 117:109009, 2020.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *International Conference on Machine Learning*, pages 1015–1022, 2010.

Jonas Umlauft and Sandra Hirche. Feedback linearization based on Gaussian processes with event-triggered online learning. *IEEE Transactions on Automatic Control*, 65(10):4154–4169, 2019.

Qi Wang and Herke Van Hoof. Doubly stochastic variational inference for neural processes with hierarchical latent variables. In *International Conference on Machine Learning*, pages 10018–10028. PMLR, 2020.

Qi Wang and Herke Van Hoof. Model-based meta reinforcement learning using graph structured surrogate models and amortized policy search. In *International Conference on Machine Learning*, pages 23055–23077. PMLR, 2022.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic MPC for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

Liang Zhang, Gang Wang, and Georgios B Giannakis. Real-time power system state estimation and forecasting via deep unrolled neural networks. *IEEE Transactions on Signal Processing*, 67(15): 4069–4077, 2019.