2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Synthetic data generation for system identification: leveraging knowledge transfer from similar systems

Dario Piga[1], Matteo Rufolo[1], Gabriele Maroni[1], Manas Mejari[1], Marco Forgione[1]

Abstract— **This paper addresses the challenge of overfitting in the learning of dynamical systems by introducing a novel approach for the generation of synthetic data, aimed at enhancing model generalization and robustness in scenarios characterized by data scarcity. Central to the proposed methodology is the concept of knowledge transfer from systems within the same class. Specifically, synthetic data is generated through a pre-trained meta-model that describes a broad class of systems to which the system of interest is assumed to belong. Training data serves a dual purpose: firstly, as input to the pre-trained meta model to discern the system's dynamics, enabling the prediction of its behavior and thereby generating synthetic output sequences for new input sequences; secondly, in conjunction with synthetic data, to define the loss function used for model estimation. A validation dataset is used to tune a scalar hyper-parameter balancing the relative importance of training and synthetic data in the definition of the loss function. The same validation set can be also used for other purposes, such as early stopping during the training, fundamental to avoid overfitting in case of small-size training datasets. The efficacy of the approach is shown through a numerical example that highlights the advantages of integrating synthetic data into the system identification process.**

I. Introduction

The performance of system identification algorithms, as well as other machine learning tools, is greatly dependent on the amount and quality of the training data. This dependence becomes challenging in situations where data is scarce and expensive to acquire, leading to overfitting when complex models are adopted.

In the machine learning literature, two main strategies have been developed to address this challenge: data augmentation and the use of synthetic data [1], [2], each with its own benefits. Data augmentation is a widely used technique in various domains, such as image and natural language processing. It involves modifying existing data samples to create new variations. This technique helps to enrich the dataset and introduce diversity, which improves the model's ability to generalize from a limited number of samples. However, data augmentation is limited by the scope and characteristics of the original data. On the other hand, the production and application of synthetic data offers a broader solution. Synthetic data goes beyond simply modifying existing data. It entails generating completely new, artificial datasets that mimic the statistical characteristics of real-world data. This approach not only overcomes the constraints of data augmentation but also provides a method to generate large and diverse datasets. However, generating reliable synthetic data might be challenging and may not be always possible.

To the best of our knowledge, only few contributions are available for data augmentation/synthetic data generation in system identification. The work [3] addresses identification of dynamical systems described in terms of nonlinear finite impulse response, where synthetic regressors (with no corresponding outputs) are created by slightly perturbing the original regressors, and manifold regularization is applied using these new synthetic regressors. The contribution [4] proposes data augmentation by modifying the original time sequences through jittering and slicing, with application to estimation of dynamical model describing harbor manouvers in maritime autonomous surface ships.

The popular physics-informed deep learning paradigm, introduced by Raissi *et al.* in [5], can be seen as a method for working with synthetic datasets. Indeed, physical laws are integrated into the training procedure to restrict the set of admissible solutions or to enforce known dynamics, thus enhancing model robustness, generalization, and explainability, even in the case of small-sized training datasets. Essentially, a regularization term is added to the fitting loss of the available training data. This regularization is formed by considering collocation points (distributed in the spatio-temporal domain), where the available *Partial Differential Equation* (PDE) describing the system should be satisfied.

In this paper, we leverage the power of synthetic data in scenarios of small-size training data. We exploit the new modeling paradigm recently proposed by some of the authors in [6] in order to derive a *meta-model* (describing a broad class to which the query system[1] is assumed to belong) that is used to generate a set of synthetic data. As discussed in [6], such a meta-model is trained on a potentially infinite stream of synthetic data, generated by simulators with randomly generated settings. The proposed approach harnesses the power of Transformers, commonly used for Natural Language Processing [7] and representing the key technology behind Large Language Models [8], [9]. At the inference time, a (typically short) input-output sequence generated by the query system (namely, available training data) is used as a context for the Transformer, which implicitly discerns the dynamics of the system, enabling predictions of

Corresponding author: matteo.rufolo@supsi.ch.

This project was partially supported by the Eurostars programme, project "*E3093 - SLIMPEC: A Software suite for LearnIng-based embedded Model PrEdictive Control*" and by HASLER STIFTUNG under the project "*Accurate and Interpretable Deep Learning models for Peak Load Forecasting*", project number: 23040.

[1]SUPSI-DTI-IDSIA, Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland.

[1]We refer to the system to be identified as a "query system". Measured data generated by the query system is used as a context for the meta-model.

its behavior. By querying the meta-model with different new input sequences, a potentially infinite stream of synthetic data is then constructed. From a different perspective, since the available training data are used as context to the meta model to generate synthetic output samples, the technique can also be seen as a data augmentation strategy, as the training data are manipulated to form new data. Overall, synthetic data for the query system is generated leveraging knowledge transfer from systems within the same class, all integrated within the pre-trained Transformer. Once the synthetic input/output sequences become available, a model (either gray or black box) of the query system, fitting both the training and the synthetic data, is estimated using any system identification tool.

The paper is organized as follows: The problem addressed is described in detail in Section II. The generation of synthetic data is discussed in Section III, which includes a description of the encoder-decoder Transformer architecture used for synthetic data generation and describing the class of systems. Parametric system identification with synthetic data is described in Section IV. A numerical example is reported in Section V to demonstrate the effectiveness of the approach and to show the advantages of using synthetic data in identification problems with small datasets. Concluding remarks and directions for future research are presented in Section VI.

II. Problem description

We consider a dataset $\mathcal{D}_{\text{tr}}$ containing a sequence of input-output pairs generated by a dynamical system $\mathcal{S}_{\text{o}}$, denoted as $\mathcal{D}_{\text{tr}} = \{(u_t, y_t)\}_{t=1}^{T}$, where u_t and y_t respectively represent the input and output of $\mathcal{S}_{\text{o}}$ at time step t, with $t = 1, \dots, T$. For simplicity, and without loss of generality, we assume that the system $\mathcal{S}_o$ is single-input single-output (SISO), i.e., $u_t, y_t \in \mathbb{R}$.

Our focus is on a standard system identification problem, aiming to fit a parametric causal dynamical model $\mathcal{M}(\cdot;\theta)$ to the training dataset $\mathcal{D}_{\text{tr}}$. Here, θ denotes the parameters describing the model $\mathcal{M}(\cdot;\theta)$, which maps an input sequence $u_{1:t}$ up to time t to an output $\hat{y}_t$, as defined by:

$$\hat{y}_t(\theta) = \mathcal{M}(u_{1:t};\theta), \tag{1}$$

where the dependence of $\hat{y}_t(\theta)$ on $u_{1:t}$ is omitted to simplify the notation.

Additionally, our goal is to augment the training dataset $\mathcal{D}_{\text{tr}}$ by generating a potentially infinite-dimensional set of synthetic input-output trajectories $\left\{\tilde{u}_t^{(i)}, \tilde{y}_t^{(i)}\right\}_{t=1}^{\tilde{T}}$, where the superscript $i = 1, 2, \dots$ is used to denote the i-th synthetic trajectory, and $\tilde{T}$ is the length of the synthetic trajectories, which are all assumed to have the same length just to keep the notation simple. The trajectories $\tilde{u}_{1:t}^{(i)}, \tilde{y}_{1:t}^{(i)}$ are assumed to be drawn from a probability distribution $P(\tilde{u}, \tilde{y})$ which ideally should be the same underlying distribution generating the training dataset $\mathcal{D}_{\text{tr}}$.

In the following section we show how to generate the synthetic output sequence $\tilde{y}^{(i)}$ for given input trajectories $\tilde{u}^{(i)}$.

III. Synthetic data generation

In order to generate the synthetic input-output sequences $\tilde{u}^{(i)}, \tilde{y}^{(i)}$, we assume that: (i) the system $\mathcal{S}_{\text{o}}$ we aim to model belongs to a broad class of dynamical systems; (ii) the meta-model, which describes the behavior of this class and is used to generate synthetic data, has been pre-trained using data from systems within the class. This approach is based on the system class modeling paradigm developed by some of the authors in [6], which is briefly reviewed in the following for self-consistency of the paper.

A Transformer with an encoder-decoder architecture, illustrated in Fig. 1, is used for generating synthetic data of the system $\mathcal{S}_{\text{o}}$. It is basically the standard Transformer architecture with attention mechanism proposed in [7], adapted in [6] to process sequences of real input/output data, and further modified in [10] with positional encoding used instead of positional embedding.

The encoder processes an input/output sequence $u_{1:m}, y_{1:m}$ (typically referred to as 'context') and generates an embedding sequence $\zeta_{1:m}$, which is processed by the decoder along with a test input $u_{m+1:N}$ (the latter subject to causal restriction) to produce the sequence of predictions $\hat{y}_{m+1:N}$ up to step N. The parameters ϕ of the Transformer are obtained by randomly sampling systems from the class, and then minimizing over ϕ the empirical loss according to a supervised learning paradigm:

$$J \;=\; \frac{1}{b}\sum_{i=1}^{b} \left\| y_{m+1:N}^{(i)} - \mathcal{T}_{\phi}(u_{1:m}^{(i)}, y_{1:m}^{(i)}, u_{m+1:N}^{(i)}) \right\|^2, \tag{2}$$

where: $\mathcal{T}_{\phi}(u_{1:m}^{(i)}, y_{1:m}^{(i)}, u_{m+1:N}^{(i)})$ denotes the output $\hat{y}_{m+1:N}$ of the Transformer when fed with a context $\{u_{1:m}^{(i)}, y_{1:m}^{(i)}\}$ and query test input $u_{m+1:N}^{(i)}$; and b denotes the number of randomly generated systems, each providing an input/output sequence $\{u_{1:N}^{(i)}, y_{1:N}^{(i)}\}$, split to form the context and to create the empirical loss J in (2). *Mini-batch gradient descent* is then used to minimize J, with b new systems and sequences resampled *at each iteration*. It should be noted that, in practical scenarios, simulators can be employed to produce the input/output sequences $\{u_{1:N}^{(i)}, y_{1:N}^{(i)}\}$ for pre-training the Transformer. Consequently, it becomes possible to create a diverse range of datasets for estimating the Transformer's parameters by adjusting software configurations (for instance, physical parameters) based on insights from the application domain.

The Transformer, pre-trained on data simulated from systems randomly selected from a particular class, serves as an extensive meta-model for that class. It gains the ability to infer the behavior of a specific query system $\mathcal{S}_{\text{o}}$ directly (specifically, through zero-shot in-context learning) from the existing training dataset $\mathcal{D}_{\text{tr}}$, which implicitly contains the main characteristics of the system $\mathcal{S}_{\text{o}}$. The training dataset provides context for the pre-trained Transformer, enabling it to generate synthetic outputs $\tilde{y}_{1:\tilde{T}}$ for any query input sequence $\tilde{u}_{1:\tilde{T}}$. Consequently, this approach allows the generation of a potentially infinite stream of synthetic input/output

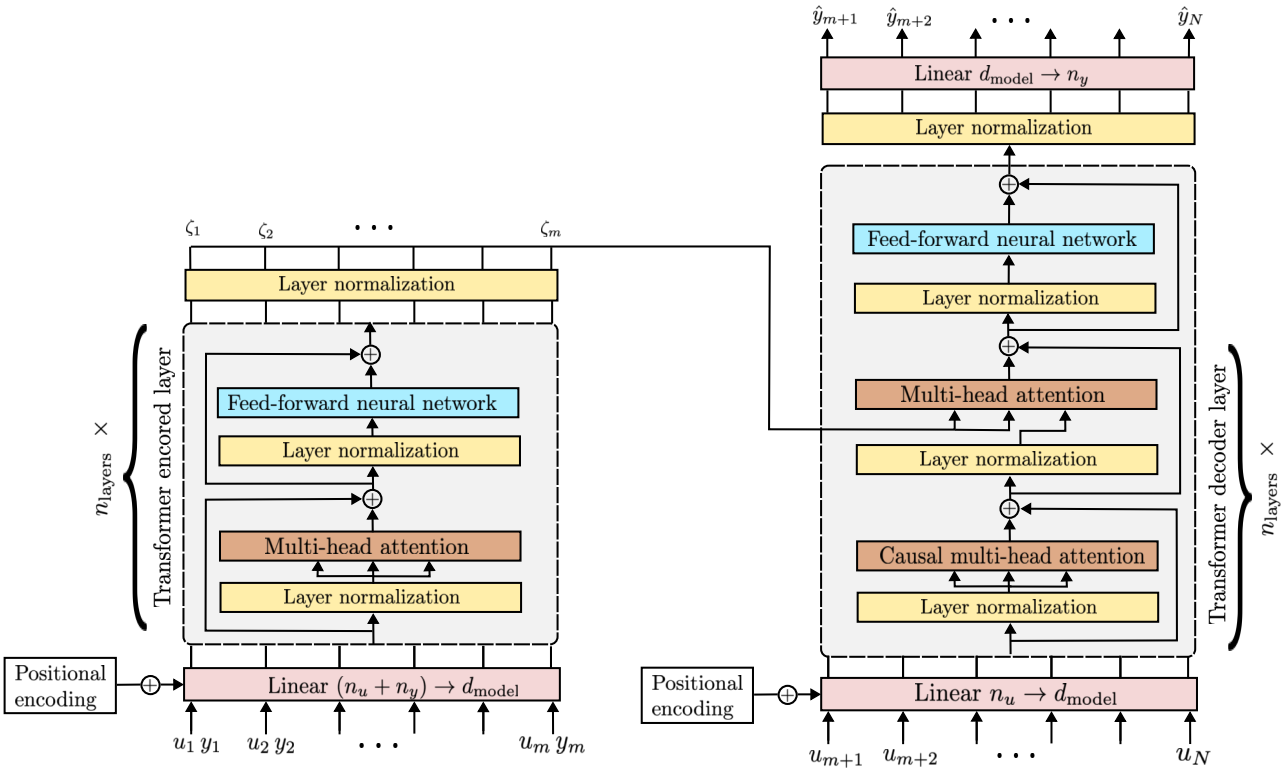


Fig. 1: Encoder-decoder Transformer model for the class of systems, used to generate synthetic data. The Transformer is characterized by: number of layers (n_{layers}), model dimensionality per layer (d_{model}), number of attention heads (n_{heads}), and context window length (m).

data, by implicitly leveraging knowledge transfer from systems within the same class.

IV. Learning with augmented data

The parameters θ characterizing the model $\mathcal{M}(\cdot;\theta)$ of the query system $\mathcal{S}_{\text{o}}$ are then estimated by minimizing the following expected loss, which considers both actual training data (from $\mathcal{S}_{\text{o}}$) and artificially generated synthetic data (from the pre-trained Transformer):

$$\hat{\theta} = \arg\min_{\theta\in\Theta} \frac{1}{T}\sum_{t=1}^{T} \ell\left(y_t, \hat{y}_t(\theta)\right) + \\ + \gamma \mathbb{E}_{P(\tilde{u},\tilde{y})}\left[\frac{1}{\tilde{T}}\sum_{t=1}^{\tilde{T}} \ell\left(\tilde{y}_t, \hat{\tilde{y}}_t(\theta)\right)\right] \quad (3)$$

where $\hat{\tilde{y}}_t(\theta)$ represents the model's output at time t for a synthetic input sequence $\tilde{u}_{1:t}$, and ℓ denotes a chosen loss function, such as the squared error defined by:

$$\ell\left(y_t, \hat{y}_t(\theta)\right) = \left(y_t - \hat{y}_t(\theta)\right)^2. \quad (4)$$

The term γ is a non-negative regularization hyperparameter that balances the influence of training and synthetic data in the model fitting process. A value of $\gamma = 0$ focuses the model on fitting the training data alone, potentially leading to overfitting, especially with overly complex models or small datasets. Increasing γ elevates the significance of synthetic data, which might introduce biases from the synthetic data generation process. The choice of γ should ideally be determined by the quality and amount of real training data and the reliability of synthetic data. In this paper, γ is selected through hold-out validation, using a portion of the training dataset $\mathcal{D}_{\text{tr}}$ or a separate validation dataset $\mathcal{D}_{\text{val}}$ to regulate the balance between real and synthetic data influences. As discussed in the example in Section V, we employ early stopping criteria to prevent overfitting, especially when relying on a small training dataset and when synthetic data is not utilized. Consequently, the same validation dataset used for early stopping can also be adopted to adjust γ, ensuring that no additional data is required beyond what is already used for model estimation with early stopping.

The expected value in (3) is approximated by applying q synthetic sequences $\tilde{u}^{(i)}$ (with $i = 1, \ldots, q$) as input of the decoder and then generating corresponding output sequences $\tilde{y}^{(i)}$. The estimation problem (3) thus becomes:

$$\hat{\theta} = \arg\min_{\theta\in\Theta} \frac{1}{T}\sum_{t=1}^{T} \ell\left(y_t, \hat{y}_t(\theta)\right) + \\ + \gamma \frac{1}{q}\frac{1}{\tilde{T}}\sum_{i=1}^{q}\sum_{t=1}^{\tilde{T}} \ell\left(\tilde{y}_t^{(i)}, \hat{\tilde{y}}_t^{(i)}(\theta)\right). \quad (5)$$

The optimization problem (5) is then solved through mini-batch stochastic gradient descent by generating, at each iteration, q new synthetic input/output sequences.

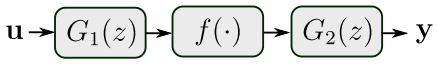


Fig. 2: The Wiener-Hammerstein system structure.

V. Example

In this section we illustrate the effectiveness of the proposed method with a numerical example focused on the identification of Wiener-Hammerstein models, which provide a block-oriented representation for numerous nonlinear dynamical systems encountered in practice [11]. The pre-trained meta model, along with the Python scripts for synthetic data generation, are available in the GitHub repository associated with this paper [12].

A. Data-generating system

For data generation, we consider a SISO stable Wiener-Hammerstein dynamical system $\mathcal{S}_{\text{o}}$, as visualized in Fig. 2, with the structure G_1–F–G_2, where G_1 and G_2 are Linear-Time-Invariant (LTI) blocks, and F is a static non-linearity sandwiched between G_1 and G_2. The LTI blocks G_1 and G_2 are randomly chosen, with dynamical order between 1 to 10, and poles randomly generated and constrained to have magnitude and phase in the range $(0.5, 0.97)$ and $(0, \pi/2)$, respectively. The non-linear block F is a feed-forward neural network with one hidden layer of size 32, and parameters randomly drawn from a Gaussian distribution.

A training dataset $\mathcal{D}_{\text{tr}}$ and a validation dataset $\mathcal{D}_{\text{val}}$, of lengths $T = 250$ and $T_{\text{val}} = 100$ respectively, are generated by exciting the system with a white input signal drawn from a normal distribution with zero mean and unit variance. The output samples are corrupted by white Gaussian noise with a standard deviation of $\sigma_e = 0.35$, corresponding to a Signal-to-Noise Ratio (SNR) of 9.1 dB.

A test dataset D_{test} consisting of $4,000$ samples is also generated. For simplification in evaluating the model's performance, the test outputs are considered without noise. It is important to note that, in practical applications, the size of the test dataset should ideally not be 16 times larger than that of the training dataset. Nevertheless, employing a large test dataset enhances the statistical significance of the results obtained. To further increase the statistical reliability of these results, a Monte Carlo study comprising 100 runs is conducted, with new data-generating system $\mathcal{S}_{\text{o}}$, training, validation, and test datasets being generated for each run.

B. Synthetic Data Generation

The Transformer $\mathcal{T}_\phi$ describing the considered class of systems was pre-trained as described in [6] using an Nvidia RTX 3090 GPU. The Transformer, which comprises 5.6 million weights, is characterized by $n_{\text{layers}} = 12$ layers, $d_{\text{model}} = 128$ units in each layer, $n_{\text{heads}} = 4$ attention heads, and an encoder's context window length of $m = T = 400$. Such a pre-trained Transformer describes the class of SISO Wiener-Hammerstein systems with LTI blocks of dynamical order up to 10.

The context provided to the encoder is the input-output training sequence. Synthetic output samples are generated by applying query input sequences $\tilde{u}_{1:\tilde{T}}$ of length $\tilde{T} = 200$, which share the same statistical distribution as the training input.

C. Parametric model

We aim at estimating a parametric model $\mathcal{M}(\cdot;\theta)$ describing the behavior of the query Wiener-Hammerstein data-generating system $\mathcal{S}_{\text{o}}$.

As a structure for the parametric model $\mathcal{M}(\cdot;\theta)$ we consider a Wiener-Hammerstein one, where the LTI blocks have a dynamical order of 10. The non-linear block F is a feed-forward neural network with one hidden layer of size 32. Overall, the considered model structure is characterized by 137 parameters. Such a parametric model $\mathcal{M}$ have the same structure of the class of systems described by the pre-trained Transformer $\mathcal{T}_\phi$. Therefore, the prior information about the class to which the query system is supposed to belong has been used to both generate synthetic data and select the structure of the parametric model $\mathcal{M}$.

The loss (5) is minimized through stochastic gradient descent for maximum 6000 iterations, with batch size $q = 1$, and by generating at each iteration new synthetic input-output sequences. In minimizing the loss, we exploited the approach in [13] for fast differentiation of the linear dynamical blocks. An early stopping criterion [14] is also adopted to avoid overfitting, which mainly occurred when no synthetic data was used (i.e., for hyperparameter $\gamma = 0$) or for small values of γ (i.e., $\gamma \leq 1$).

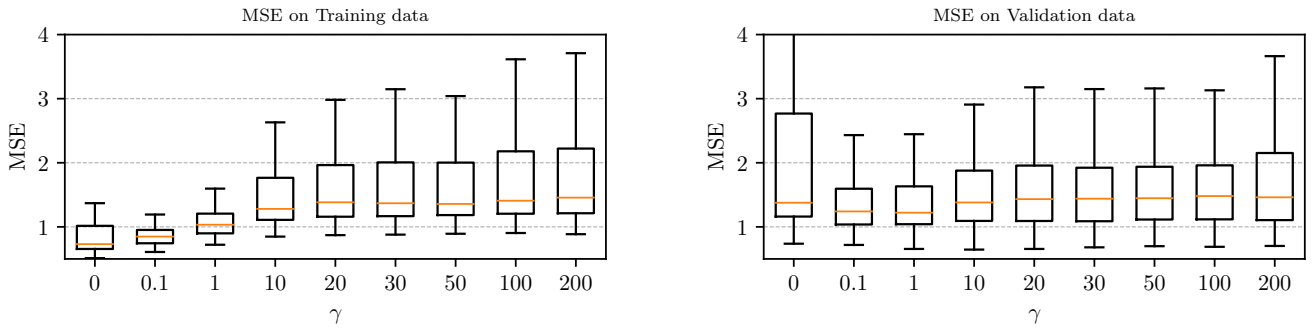
The hyperparameter γ is selected through a coarse grid search, considering the following values: $0, 0.1, 1, 10, 20, 30, 50, 100, 200$ and by minimizing the mean squared error (MSE) over the validation dataset $\mathcal{D}_{\text{val}}$, which is defined as:

$$\text{MSE} = \frac{1}{T_x} \sum_{t=1}^{T_x} \left(y_t - \hat{y}_t(\theta)\right)^2, \tag{6}$$

where T_x is the length of the dataset where the MSE is computed ($T_x = T_{\text{val}} = 100$ for validation set), y_t denotes the true output at time step t, and $\hat{y}_t(\theta)$ indicates the predicted model's output. We remark that the same validation dataset is adopted both for early stopping and to select the hyperparameter γ.

D. Results

Fig. 3 shows the boxplots of the Mean Squared Error on the training (left panel) and validation dataset (right panel) obtained for different values of the hyperparameter γ, with $\gamma = 0$ corresponding to fit only the available training data. To illustrate the presence of overfitting, the MSE on training data is not the one obtained by the model estimated with early-stopping, but the one achieved when the maximum number of iterations is reached. Results in the figure show the regularization role played by synthetic data. Indeed, for $\gamma = 0$ the MSE in the validation data is about 5 times larger



(a) MSE *vs* γ on training data.

(b) MSE *vs* γ on validation data.

Fig. 3: Impact of regularization hyparameter γ on mean squared error in training (left panel) and validation (right panel) dataset. Boxplots of average squared error over 100 Monte Carlo runs. In the right panel, the vertical limit is set to 4 for a better visualization of the boxplots associated to $\gamma \neq 0$.

than the one in training, while similar MSEs are achieved in training and validation when the relevance of synthetic data w.r.t. training data is significant (i.e., for $\gamma \geq 10$). We also notice a benefit in validation performance when synthetic data is used ($\gamma > 0$). Improvement is visible, although we observe that when the ratio of synthetic data in the loss is more than 10 times larger than training data, performance in validation decreases, intuitively due to the fact that measured training data, although affected by noise, are more reliable than synthetic data generated by the meta-model, which can be affected by epistemic uncertainty.

Fig. 4 shows results obtained on the test dataset $\mathcal{D}_{\text{test}}$. For a better interpretation of the performance, the R^2 coefficient is plotted in the figure instead of the MSE, where the R^2 is defined as:

$$R^2 = 1 - \frac{\sum_{t=1}^{T_x} \left(y_t - \hat{y}_t(\theta)\right)^2}{\sum_{t=1}^{T_x} \left(y_t - \bar{y}\right)^2} \tag{7}$$

where $\bar{y}$ is the mean output and $T_x = 4,000$ is the length of the test dataset. In the test, we only considered (and thus reported) a comparison between performance achieved without using synthetic data and performance achieved using synthetic data, with regularization hyperparameter γ selected, at each Monte Carlo run, through hold-out validation. As expected from the results in validation, also test results show a significant improvement in the model performance thanks to the usage of synthetic data, with a median of the R^2 coefficient which improved from 0.889 (in case of no synthetic data) to 0.956.

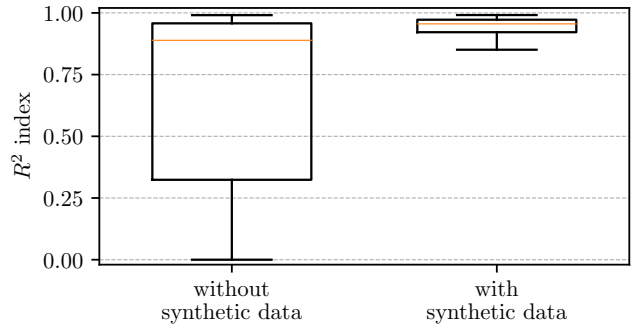


Fig. 4: Impact of synthetic data on R^2 performance in test dataset. Boxplots of R^2 coefficients over 100 Monte Carlo runs: without using synthetic data (left); using synthetic data (right).

VI. Conclusions

In this paper, we demonstrate the feasibility of generating an extensive stream of synthetic data for system identification by employing knowledge transfer from analogous systems. This allows to mitigate the challenges posed by data scarcity, offering enhancements in model performance and generalization capabilities. A practical numerical example highlights the efficacy of the methodology, showcasing an increasing of the R^2 coefficient when synthetic data is integrated with traditional training datasets.

Current research directions are focused on:

- Refinement and scaling-up of the meta-model to encompass a broader spectrum of dynamical systems. This ensures its applicability across a diverse range of system identification scenarios.
- Estimation of the uncertainty associated with the meta-model's outputs. This allows to reformulate the minimization of the loss with synthetic data as a Maximum Likelihood estimation problem. This advancement will allow for a proper weighting of synthetic data based on their reliability, ensuring that less certain synthetic samples exert a smaller influence on the model estimation process compared to more reliable ones and actual training data.
- Enhancing Bayesian estimation algorithms, such as Gaussian Process Regression, by incorporating the meta-model's output as a prior. This strategy is particularly beneficial in areas of the input space lacking observations, where the model increasingly relies on the prior. This incorporation seeks to utilize the knowledge

derived from analogous systems, encapsulated by the meta-model, to make more informed inferences in domains with few observations.

REFERENCES

[1] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[2] S. I. Nikolenko, *Synthetic data for deep learning*. Springer, 2021, vol. 174.

[3] S. Formentin, M. Mazzoleni, M. Scandella, and F. Previdi, "Nonlinear system identification via data augmentation," *Systems & Control Letters*, vol. 128, pp. 56–63, 2019.

[4] K. Wakita, Y. Miyauchi, Y. Akimoto, and A. Maki, "Data augmentation methods of parameter identification of a dynamic model for harbor maneuvers," *arXiv preprint arXiv:2305.18851*, 2023.

[5] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[6] M. Forgione, F. Pura, and D. Piga, "From system models to class models: An in-context learning paradigm," *IEEE Control Systems Letters*, pp. 1–1, 2023.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[10] D. Piga, F. Pura, and M. Forgione, "On the adaptation of in-context learners for system identification," in *Proceedings of the 20th IFAC Symposium on System Identification*, Boston, MA, 2024.

[11] F. Giri and E.-W. Bai, *Block-oriented nonlinear system identification*. Springer, 2010, vol. 1.

[12] D. Piga, "Synthetic data generation," https://github.com/dariopi/synthetic_data_generation, 2024, accessed February 29, 2024.

[13] M. Forgione and D. Piga, "dynoNet: A neural network architecture for learning dynamical systems," *International Journal of Adaptive Control and Signal Processing*, vol. 35, no. 4, pp. 612–626, 2021.

[14] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, 1995.