On the adaptation of in-context learners for system identification

Dario Piga, Filippo Pura, Marco Forgione

IDSIA Dalle Molle Institute for Artificial Intelligence USI-SUPSI, Via la Santa 1, CH-6962 Lugano-Viganello, Switzerland.

Abstract: In-context system identification aims at constructing meta-models to describe classes of systems, differently from traditional approaches that model single systems. This paradigm facilitates the leveraging of knowledge acquired from observing the behaviour of different, yet related dynamics. This paper discusses the role of meta-model adaptation. Through numerical examples, we demonstrate how meta-model adaptation can enhance predictive performance in three realistic scenarios: tailoring the meta-model to describe a specific system rather than a class; extending the meta-model to capture the behaviour of systems beyond the initial training class; and recalibrating the model for new prediction tasks. Results highlight the effectiveness of meta-model adaptation to achieve a more robust and versatile meta-learning framework for system identification.

1. INTRODUCTION

In traditional system identification, the focus is on learning a model of a specific system through a-priori physical knowledge and measured input-output trajectories. This typical workflow is closely related to supervised machine learning, albeit with distinct peculiarities. The emphasis is on dynamical systems, opting for parsimonious representations like Linear Parameter-Varying and hybrid models (Bamieh and Giarre, 2002; Mejari et al., 2020), and employing the model for applications such as control design (Bombois et al., 2006; Piga et al., 2019). Given the intrinsic link between supervised learning and system identification, recent contributions employ deep learning tools to estimate dynamical systems using neural network structures (Masti and Bemporad, 2021; Forgione and Piga, 2021a,b; Beintema et al., 2023; Pillonetto et al., 2023). Despite these advancements, the conventional approach often overlooks potential insights from previous trainings on similar systems, leaving room for enhanced methodologies that leverage knowledge accumulated across related systems and tasks.

Harnessing this knowledge represents a pioneering shift for system identification and adaptive control (see, e.g., Zhan et al. (2022); Richards et al. (2021); Chakrabarty et al. (2023); Balim et al. (2023)), situated within the domain of meta-learning or "learning to learn" (Schmidhuber, 1987; Finn et al., 2017). Meta-learning involves training models across various tasks to enable these models to adapt to new ones with minimal human intervention, training effort, or data. This facilitates generalization from one task to another, a significant leap towards autonomous system identification and control.

In the in-context system identification framework introduced by the authors in Forgione et al. (2023b), instead of estimating a model of a specific dynamical system, a meta-model describing a whole class of such systems is learned. Meta-model pre-training is performed using data drawn from a potentially infinite stream of input/output sequences generated from different, yet related systems. In practice, data could be sourced from simulators, allowing for the generation of an arbitrary amount of synthetic input/output sequences by varying physical parameters and disturbances. To guarantee substantial representational power of the meta-model, an encoder-decoder Transformer architecture is adopted (Vaswani et al., 2017; Radford et al., 2019) and specialized to process real-valued sequences. The Transformer acts as an in-context learner (Kirsch et al., 2022), taking a short input-output sequence (context) and producing predictions without the need for a specialized model of the system under study. This eliminates the necessity to specify a dynamical model structure, implement a training algorithm, tune hyper-parameters, etc.

Although Forgione et al. (2023b) shows that the meta-model exhibits high performance on test systems belonging to the same class used for training, model adaptation may be necessary to further enhance its predictive accuracy. Indeed, this is a need not only in meta-learning, but also in traditional system identification to take into account, e.g., aging, changing configurations, and different external conditions (see, e.g., Pozzoli et al. (2020); Forgione et al. (2023a)).

In this paper, we extend the work in Forgione et al. (2023b) showing through numerical examples that only minimal data sequences and a few iterations of gradient descent for weight updates are needed to improve the model's predictive performance. We consider three scenarios where adaptation can be applied: refining the meta-model to describe a specific system rather than a class; adapting the

meta-model to describe the behaviour of a system outside the class used for training; and modifying the model for new prediction tasks, specifically transitioning from short-term (easy) to long-term prediction (more difficult) task.

The rest of the paper is organized as follows. Section 2 discusses the problem setting, elaborates on the meta-learning framework introduced in Forgione et al. (2023b), and provides a more detailed discussion on the model adaptation scenarios addressed in this paper. Architectures and algorithms for training and adaptation are described in Section 3. Numerical examples illustrating the three adaptation scenarios are presented in Section 4 and conclusions are drawn in Section 5.

2. PROBLEM SETTING

2.1 Conventional vs. meta identification

Conventional system identification seeks to estimate the model of a *fixed* unknown dynamical system S using an input-output sequence $\mathcal{D} = (u_{1:N}, y_{1:N})$ generated by S, where $u_k \in \mathbb{R}^{n_u}$ (resp. $y_k \in \mathbb{R}^{n_y}$) represents the system's input (resp. output) at time step k.

In the in-context learning framework introduced in Forgione et al. (2023b) and discussed in this paper, a *probability distribution* of dynamical systems is considered. This distribution is used to generate a sequence of such systems along with their corresponding input-output datasets. Thus, we have access to an *infinite stream* of input/output pairs $\{\mathcal{D}^{(i)} = (u_{1:N}^{(i)}, y_{1:N}^{(i)}), i = 1, 2, \ldots, \infty\}$, each derived by feeding a randomly generated signal $u_{1:N}^{(i)}$ as input to a randomly instantiated dynamical system $S^{(i)}$. Formally, we can sample synthetic datasets $\mathcal{D}^{(i)}$ from an underlying distribution $p(\mathcal{D})$, which may not be explicitly known. Since the datasets $\mathcal{D}^{(i)}$ are drawn from the common distribution $p(\mathcal{D})$, partial knowledge transfer from one dataset to another is possible.

2.2 Learning system classes

As discussed in Forgione et al. (2023b), an in-context learner $\mathcal{M}_\phi$ (called meta-model), with parameters ϕ, can be utilized to describe a whole class of systems. The meta-model $\mathcal{M}_\phi$ processes the input-output dataset $\mathcal{D}^{(i)}$ and directly provides the predictions of interest, without generating a specialized model of the system $S^{(i)}$ generating the dataset $\mathcal{D}^{(i)}$.

Specifically, the meta-model $\mathcal{M}_\phi$ receives the input/output sequence $(u_{1:m}^{(i)}, y_{1:m}^{(i)})$ up to time step m and a test input sequence $u_{m+1:N}^{(i)}$ from time step $m + 1$ to N. Its goal is to produce the corresponding simulated output sequence $\hat{y}_{m+1:N}^{(i)}$:

$$\hat{y}_{m+1:N}^{(i)} = \mathcal{M}_\phi(u_{1:m}^{(i)}, y_{1:m}^{(i)}, u_{m+1:N}^{(i)}). \qquad (1)$$

To solve such a meta-learning problem, the meta-model $\mathcal{M}_\phi$ is expected to "understand" (to a certain degree) the data generating mechanism $S^{(i)}$ from the provided *context* $(u_{1:m}^{(i)}, y_{1:m}^{(i)})$, without returning an explicit representation in a model form.

2.3 Adaptation

Once the meta-model $\mathcal{M}_\phi$ is estimated based on the data drawn from the probability distribution $p(\mathcal{D})$, the model may need to be adapted to enhance its predictive capabilities in various scenarios, such as:

- **From class to system:** Adapt the model to describe data not generated by the distribution $p(\mathcal{D})$, but by the conditional distribution $p(\mathcal{D}|S)$. This case encompasses the scenario where we want to specialize the model $\mathcal{M}_\phi$ not to describe a generic class of systems, but to move towards a narrower model describing a specific system S generating data $\mathcal{D}$.
- **From class to class:** Adapt the model in case of a shift in the data-generating probability distribution $p(\mathcal{D})$. This encompasses the scenario where the model class we aim to describe is not included in the class used for training. The class-to-system and the class-to-class problems can also be considered together to handle a class to out-of-class-system problem. In this case, a shift in the probability distribution occurs and at the same time we are interested in modelling the conditional distribution $p(\mathcal{D}|S)$ for a system not belonging to the class used in the training process.
- **Change of prediction task:** Adapt the meta-model trained to perform a prediction task (for example, n-step ahead prediction) towards performing another prediction task (for example, n'-step ahead prediction). Indeed, when $n \ll n'$, training for n-step ahead is usually easier than training for n'-step ahead. Thus, training can be done first for with an n-step ahead objective. The harder n'-step ahead prediction objective can be employed at a later stage to refine the models through a few iterations of gradient descent or to update only a subset of the model's parameters. This is in line with the concept of *curriculum learning*, where a learning problem is tackled by solving tasks of increasing difficulty (Wang et al., 2021).

Model learning and adaptation is discussed in the following section, and the importance of adaptation in the three scenarios mentioned above is shown in Section 4 through numerical examples.

3. META SYSTEM IDENTIFICATION: ALGORITHMS

3.1 Architecture

The encoder-decoder Transformer architecture used as meta-model in this paper is illustrated in Fig. 1. It is a rather standard Transformer architecture, similar to the one introduced in Vaswani et al. (2017) and adapted to receive as input/produce as output real-valued samples instead of categorical world tokens (Forgione et al., 2023b).

The Transformer is fully specified by the choice of the following hyper-parameters: number of layers n_{layers}; number of units in each layer d_{model}; number of heads n_{heads}; and context window length n_{ctx}.

3.2 Training

The encoder processes an input/output sequence $u_{1:m}, y_{1:m}$ and generates an embedding sequence $\zeta_{1:m}$, which is then
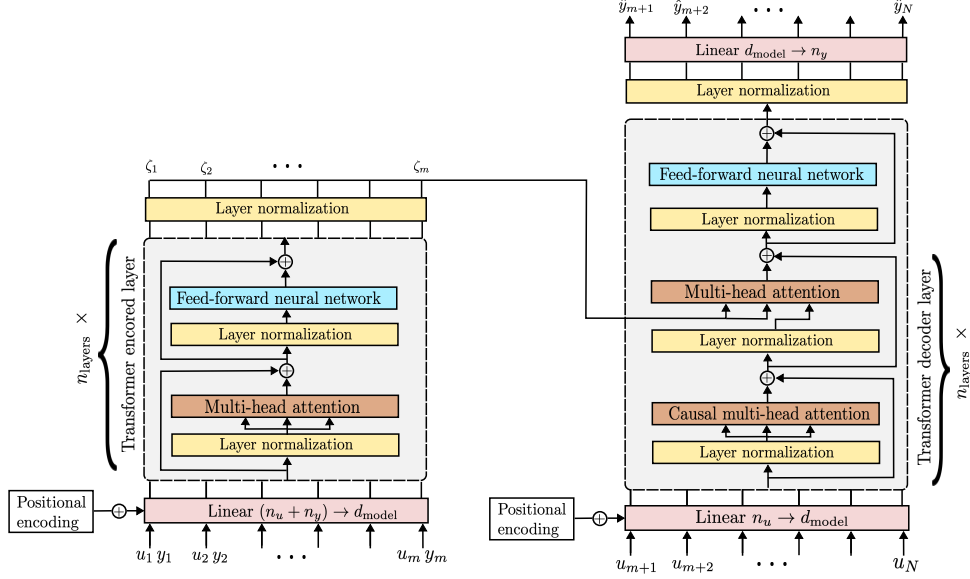
Fig. 1. Encoder-decoder Transformer for multi-step-ahead simulation. With respect to the architecture in Forgione et al. (2023b), we use fixed positional encodings.

processed by the decoder along with a test input $u_{m+1:N}$ (the latter with causal restriction) to produce the sequence of predictions $\hat{y}_{m+1:N}$ up to step N. The weights ϕ of the meta-model $\mathcal{M}_\phi$ (namely, the learning parameters of the Transformer) are obtained by minimizing over ϕ the expected loss:

$$J^* = \mathbb{E}_{p(\mathcal{D})}\left[\left\|y_{m+1:N} - \mathcal{M}_\phi(u_{1:m}, y_{1:m}, u_{m+1:N})\right\|^2\right]. \quad (2)$$

The expected loss J^* is approximated with an average J over a finite number of sampled systems $S^{(i)}$ and datasets $\mathcal{D}^{(i)}$ according to:

$$J = \frac{1}{b}\sum_{i=1}^{b}\left\|y_{m+1:N}^{(i)} - \mathcal{M}_\phi(u_{1:m}^{(i)}, y_{1:m}^{(i)}, u_{m+1:N}^{(i)})\right\|^2, \quad (3)$$

where b denotes the number of randomly generated systems, each one providing a input/output dataset $\mathcal{D}^{(i)}$ of length N.

The loss (3) is then optimized through *stochastic gradient descent*, with b new datasets resampled *at each iteration*.

3.3 Adaptation

Once a nominal meta-model has been trained, an adaptation step is performed to refine its capabilities to solve new tasks. This is achieved by employing the pre-trained Transformer, with its learned weights ϕ, as an initial condition for subsequent training phases on the new (possibly small-size) dataset, which is specific to the task we aim to describe. This approach leverages the knowledge already acquired by the model, allowing for a more efficient and targeted learning process. This additional training is conducted over a limited number of iterations, applying gradient updates to fine-tune the model's parameters specifically for the new data characteristics. To ensure the model's generalization ability and prevent overfitting, an early stopping criterion is employed based on the performance on a separate validation dataset. This method not only enhances the model's adaptability to new system

dynamics, but also significantly reduces the computational resources and time required for retraining.

4. NUMERICAL EXAMPLES

In this section, we present four numerical examples to demonstrate the effectiveness of meta-modelling and adaptation:

- The first example does not consider adaptation, but illustrates "extrapolation" properties of the meta-model across different classes. We show that the trained meta-models, even without fine-tuning, have a certain degree of meta-generalization capabilities, namely, the ability to "interpolate between classes", enabling them to describe new classes not used in training.
- The second and third examples focus on meta-model adaptation. They address class-to-in-class-systems and class-to-out-of-class systems, respectively. The third example thus demonstrates the relevance of model adaptation in the case of a shift in the data-generating probability distribution $p(\mathcal{D})$.
- The fourth example addresses refinement in the case of a task change, specifically transitioning from short-term to long-term predictions.

The *AdamW* algorithm (Loshchilov and Hutter, 2017) is used to train the parameters ϕ of the meta-model $\mathcal{M}_\phi$, by minimizing the empirical risk (3).

The software has been developed in PyTorch and it is available in the GitHub repository `https://github.com/forgi86/sysid-transformers-transfer`. Computations are performed on an Nvidia RTX 3090 GPU.

4.1 Meta-generalization to unseen system classes

We illustrate the meta-generalization properties of the Transformer to a data distribution outside of the training region. To this end, we consider different data distributions generated by LTI systems with random order
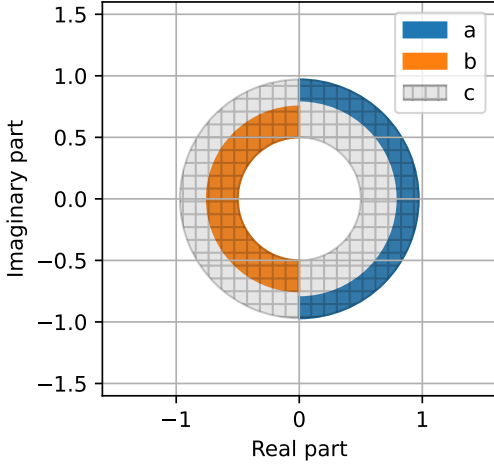
Fig. 2. Regions of the poles for the distributions a, b, and c in the unit circle.

between 1 and 10, and poles constrained in different regions of the unit circle. In particular, let us denote for convenience a, b, c the data distributions generated by linear systems having poles uniformly distributed in the magnitude/phase regions of $(0.8, 0.97)/(-\pi/2, \pi/2)$, $(0.5, 0.75)/(\pi/2, 3/4\pi)$, and $(0.5, 0.97)/(-\pi, \pi)$, respectively. Note that the pole region of c is a superset of a and b (see Fig. 2). Furthermore, let us denote by $a + b$ the mixture distribution obtained by sampling with equal probability from a and b.

We assess the generalization properties of a meta-model learned by repeating the learning procedure on a, b, and $a + b$, while monitoring the performance obtained on c. Results are visualized in Fig. 3, where we report the training loss curves obtained by training on a, b, and $a + b$, together with the corresponding "test" loss, always evaluated on c. It is evident that, if we train on the data distributions a or b, the meta generalization on c is rather poor, i.e. there is a significant gap between the training performance on a or b and test performance measured on c (top and middle panel). However, if we train on $a + c$, the meta generalization gap is much narrower (bottom right panel). Intriguingly, when training on $a + b$, the Transformer appears to be able to extrapolate effectively from magnitude/phase values that are only seen in one of the two distributions.

4.2 Adaptation from class to system

We consider a nominal meta-model $\mathcal{M}_\phi$ trained according to the setting in Forgione et al. (2023b) and describing the class of stable *Wiener-Hammerstein* (WH) dynamical systems with structure G_1–F–G_2, where G_1, G_2 are SISO LTI blocks randomly generated with order comprised between 1 to 5, and F is a static feed-forward neural network with one hidden layer and randomly generated parameters.

The Transformer structure is defined by $n_{\text{layers}} = 12$ layers; $d_{\text{model}} = 128$ units in each layer; $n_{\text{heads}} = 4$ attention heads; $m = 400$ and $n = N - m = 100$ encoder's and decoder's context window length, respectively. The total
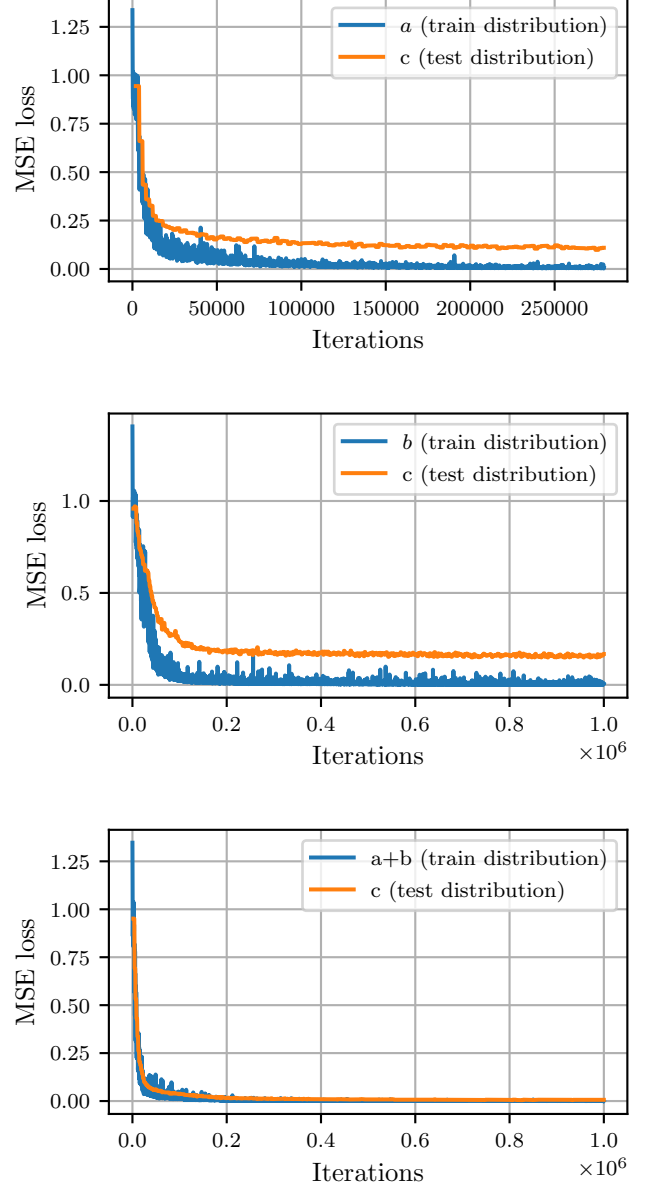

Fig. 3. Training and test loss. Training on a (upper panel), on b (middle panel), and on (a, b) (bottom panel). Test is always performed on c.

number of weights characterizing the Transformer is 5.6 M. Training was performed on 6 M iterations, corresponding to a run time of 5.4 days. At each iteration, $b = 32$ new WH systems and white Gaussian input sequences are randomly sampled, and corresponding output sequences are simulated.

In-class system The nominal meta-model is then adapted to describe *specific* WH systems. A Monte Carlo analysis is carried out over 50 randomly sampled system. At each Monte Carlo run, we sample one WH system and generate from that system 140 input/output sequences, each one of length $N = 500$. Out of these sequences, 100 are used for training, 20 for validation, and 20 for test. Model adaptation is performed on 10'000 iterations, corresponding to about 20 minutes per run. Training and validation losses
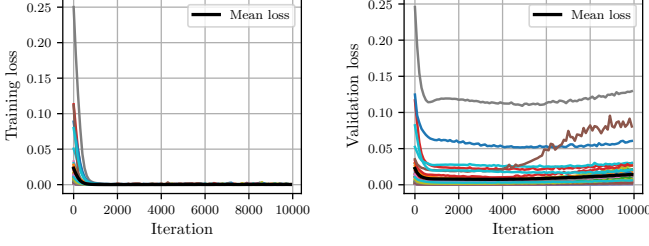
Fig. 4. WH model adaptation: training (left) and validation (right) over 50 Monte Carlo runs. The mean loss is reported as a thick black line.
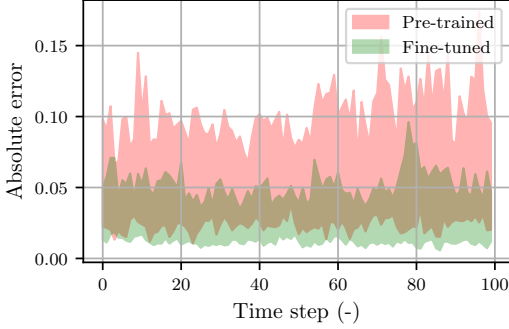

Fig. 5. WH model adaptation: area between the 25th and the 75th quantiles of the absolute error over time for the pre-trained (red) and fine-tuned (green) models.

are reported in Fig. 4 *vs.* number of iterations. We observe that, in average, the validation loss achieves its minimum in about $4'000$ iterations. Fig. 5 shows the area between the 25th and 75th quantiles of the absolute value of the simulation error over test data and for each simulation time step. For comparison, the quantiles corresponding to the pre-trained (non fine-tuned) meta-model are reported in the same figure, showing the effectiveness of the adaptation procedure.

Out-of-class system   In this numerical example, we adapt the nominal meta-model $\mathcal{M}_\phi$ (pre-trained on the WH class) to describe *Parallel Wiener-Hammerstein* (PWH) systems. The same experimental settings considered in the previous example are adopted.

Training and validation losses are reported in Fig. 6 *vs.* the iteration index. We observe that, in average, the validation loss achieves its minimum in about $6'000$ iterations. Fig. 7 shows the area between the 25th and 75th quantiles of the absolute value of the simulation error over test data and for each simulation time step. For comparison, the quantiles corresponding to the pre-trained (non fine-tuned) meta-model are reported in the same figure. The reported results show the capabilities of the meta-model to cope (after fine-tuning) with data distribution shifts.

4.3 Short-to-long simulation

We consider the problem of meta-modelling the WH class to make predictions over the longer horizon of $n = 1000$ steps. We set-up training of the Transformer with the same configurations used in the previous cases. Fig. 8
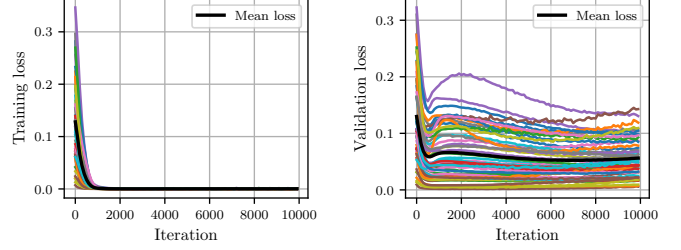

Fig. 6. PWH model adaptation: training (left) and validation (right) losses over 50 Monte Carlo runs. The mean loss is reported as a thick black line.
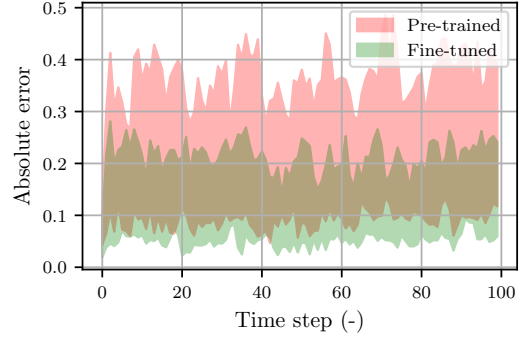

Fig. 7. PWH model adaptation: area between the 25th and the 75th quantiles of the absolute error over time for the pre-trained (red) and fine-tuned (green) models.

(orange curve) shows the training loss vs. the optimization iteration. The loss is stuck around the numerical value 1.0, which corresponds to no meaningful learning. [1] The training procedure was thus interrupted after about 250'000 iterations where no progress occurred.

Conversely, training from scratch a n-step-ahead simulator with $n = 100$ was found to be feasible (blue curve), with the loss decreasing to a very low value after 1'000'000 iterations (approximately 1 day), similarly to the results reported in Forgione et al. (2023b). The trained meta-model obtains an average rmse index of 0.103. We then executed the optimization over the longer horizon $n = 1000$ using the weights previously learned on the $n = 100$ case as initialization, instead of starting from random ones. Interestingly, with this initialization (green curve), the loss decreased rapidly in the first 20'000 iterations and reached a plateau after about 80'000 iterations (approximately 21 hours), when the experiment was interrupted. The adapted meta-model achieves an average simulation rmse of 0.113 over 1000 steps.

5. CONCLUSION

The paper demonstrates the potential of in-context learning for system identification, emphasizing the importance of model adaptability. It showcases how meta-modelling can extrapolate between different classes, enabling the description of new, unseen system classes. The key scenarios explored include refining meta-models for specific systems,

---

[1] Signals are normalized to zero mean and unit variance. A unit loss is thus achieved by a (dummy) predictor returning the zero constant.
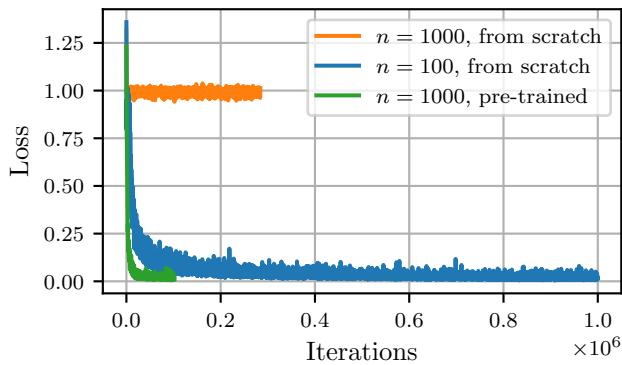
Fig. 8. Training loss of the Transformer for short-term simulation ($n = 100$) initialized from scratch (blue); for long-term simulation ($n = 1000$) initialized from scratch (orange) and for long-term simulation ($n = 1000$) with initialization from the weights trained on short-term sequences (green). In all cases, the context length is $m = 400$.

adapting to behaviours outside the initial training class, and transitioning models for new prediction tasks. The adaptation process is shown to require limited computational time and data compared to meta-model pretraining, with an early stopping criterion ensuring model generalization and preventing overfitting.

This work aims to open research directions for new learning and modelling paradigms in system identification, leveraging the power of neural network architectures like Transformers and by exploiting the shared features among similar dynamical systems to identify new systems of interest with a limited amount of data and time.

REFERENCES

Balim, H., Du, Z., Oymak, S., and Ozay, N. (2023). Can transformers learn optimal filtering for unknown systems? *arXiv preprint arXiv:2308.08536*.

Bamieh, B. and Giarre, L. (2002). Identification of linear parameter varying models. *International Journal of Robust and Nonlinear Control*, 12(9), 841–853.

Beintema, G.I., Schoukens, M., and Tóth, R. (2023). Deep subspace encoders for nonlinear system identification. *Automatica*, 156.

Bombois, X., Scorletti, G., Gevers, M., Van den Hof, P.M., and Hildebrand, R. (2006). Least costly identification experiment for control. *Automatica*, 42(10), 1651–1662.

Chakrabarty, A., Wichern, G., and Laughman, C.R. (2023). Meta-learning of neural state-space models using data from similar systems. In *World Congress of the International Federation of Automatic Control (IFAC)*.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Forgione, M., Muni, A., Piga, D., and Gallieri, M. (2023a). On the adaptation of recurrent neural networks for system identification. *Automatica*, 155, 111092.

Forgione, M. and Piga, D. (2021a). Continuous-time system identification with neural networks: Model structures and fitting criteria. *European Journal of Control*, 59, 69–81.

Forgione, M. and Piga, D. (2021b). dynoNet: A neural network architecture for learning dynamical systems. *International Journal of Adaptive Control and Signal Processing*, 35(4), 612–626.

Forgione, M., Pura, F., and Piga, D. (2023b). From system models to class models: An in-context learning paradigm. *IEEE Control Systems Letters*, 1–1. doi: 10.1109/LCSYS.2023.3335036.

Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. (2022). General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Masti, D. and Bemporad, A. (2021). Learning nonlinear state–space models using autoencoders. *Automatica*, 129, 109666.

Mejari, M., Breschi, V., and Piga, D. (2020). Recursive bias-correction method for identification of piecewise affine output-error models. *IEEE Control Systems Letters*, 4(4), 970–975.

Piga, D., Forgione, M., Formentin, S., and Bemporad, A. (2019). Performance-oriented model learning for data-driven MPC design. *IEEE control systems letters*, 3(3), 577–582.

Pillonetto, G., Aravkin, A., Gedon, D., Ljung, L., Ribeiro, A.H., and Schön, T.B. (2023). Deep networks for system identification: a survey. *arXiv preprint arXiv:2301.12832*.

Pozzoli, S., Gallieri, M., and Scattolini, R. (2020). Tustin neural networks: a class of recurrent nets for adaptive mpc of mechanical systems. *IFAC-PapersOnLine*, 53(2), 5171–5176.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9.

Richards, S., Azizan, N., Slotine, J.J.E., and Pavone, M. (2021). Adaptive-control-oriented meta-learning for nonlinear systems. *Proceedings of the Robotics: Science and Systems (RSS)*. URL https://arxiv.org/abs/2103.04490.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Ph.D. thesis, Technische Universität München.

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

Wang, X., Chen, Y., and Zhu, W. (2021). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9), 4555–4576.

Zhan, S., Wichern, G., Laughman, C., Chong, A., and Chakrabarty, A. (2022). Calibrating building simulation models using multi-source datasets and meta-learned bayesian optimization. *Energy and Buildings*, 270, 112278.