Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Autonomous Process Model Identification using Recurrent Neural Networks and Hyperparameter Optimization

Mehmet Mercangöz, Andrea Cortinovis, and Sandro Schönborn

** ABB Future Labs, Baden-Dättwil, Switzerland (e-mail: andrea.cortinovis@ch.abb.com, sandro.schoenborn@ch.abb.com, mehmet.mercangoez@ch.abb.com)*

Abstract: We demonstrate the application of automated machine learning to the problem of identifying dynamic process models using recurrent neural networks (RNNs). The general concept relies on continuous monitoring of input-output data from a plant and the processing of this data by a collection of algorithms. The data is first processed by a collaborative filtering system, which suggests a classification of system dynamics per output channel. The proposed classification and the number of input channels is used to initialize a search over RNN hyperparameters. The search algorithm uses subsets of historical data for training and validation to select the RNN architecture and determine the network parameters according to preselected objectives for balancing model accuracy and model compactness. The proposed approach is demonstrated on a simulated case study for online system identification of a chemical reactor, where the underlying dynamic characteristics of the simulated system are changed during the simulation as the system undergoes a number of disturbances and handles control tasks. Process models for the system in question are obtained via the automated machine learning approach and the models are updated as the system dynamics change. The results show good prediction accuracy of the models throughout the simulation representing changes in system dynamics.

1. INTRODUCTION

Operator room staff of large industrial complexes such as refineries, chemical plants, or pulp and paper mills have repetitive, stressful, and difficult jobs. It is becoming more and more difficult to replace retiring personnel and, given the aging workforce in most developed countries, the situation is expected to lead to a shortage of skilled labor. The operators play a very critical role for these plants as the existing automation systems are generally built with an assumption of having supervising human operators. At the same time, statistically the leading cause of unexpected shutdowns in these facilities is operator error, and such shutdowns, given the massive inertia in these operations, often means several hours if not days of outage and hundreds of thousands of dollars in lost revenues. These observations are strongly motivating research and development of intelligent systems, which could substitute or at the very least assist human operators to increase their productivity to counter the shortage of co-workers and at the same time minimize the risk of errors and therefore human error related shutdowns. Addressing this challenge requires investigation in multiple dimensions. Since industrial processes operate most of the time under normal conditions, the first aspect to address would be to look at the base automation tasks and substitute the involvement and intervention of operators within highly expected scenarios of disturbances and operational changes. This can be considered analogous to highway driving in autonomous driving systems but rolling out an autonomous plantwide solution is non-trivial and will require significant engineering effort for every plant when using current state-of-the-art methods. Machine learning (ML) solutions that learn from past operator responses and historical data could be a more scalable and economically feasible alternative (Mercangöz et al., 2019).

At a very high level two different approaches can be considered when using ML to handle the challenges described above. The first approach would be the consideration that it is easier to learn control policies directly from data, rather than learning a model. Ongoing attempts at learning to directly control, optimize, and operate systems are using approaches like imitation learning, machine teaching, or reinforcement learning (Spielberg et al., 2017; Zhu, 2015; Evans et al., 2016; Kober&Peters, 2010). Although these approaches have certain advantages like exploiting the possibility of only learning the relevant components of a system for the tasks at hand, most of the demonstrated solutions utilize a simulation environment to generate the massive experiments and datasets required to learn the control policies, which is not feasible to carry out in real-world systems for various reasons and reliable simulations are typically not available for large scale process systems especially for those, whose behavior tends to change in time due to a wide variety of internal and external factors (such as catalyst poisoning, which is used in the case study of this article).

The second approach for utilizing data driven methods for process operations is to follow a two-step procedure of model learning followed by the desired application such as model based control, optimization, or process monitoring. This is also the approach followed in this article, as it forms a basis for explicit analysis to provide insights into the potential decisions coming from systems using the models, which is typically lacking in the former approach of direct decision making from data. The model learning process corresponds to solving the system identification problem (Ljung et al., 2011) and given the original motivation our focus is the possibility to treat any system-of-interest, which rules out many techniques, which are limited in system properties e.g. only applicable to linear time-invariant systems (Coulson et al., 2019) or which have unfavorable scaling properties like Gaussian process models (Chan et al., 2013). Artificial Neural Networks (ANNs) are by now established as very powerful universal function approximators and although they are inferior in some respects to other approaches (e.g. they cannot provide calibrated uncertainty output) they scale very well and there is growing support for both software and hardware implementations, which makes them attractive for industrial applications. For approximating the behavior of dynamic systems there are two possibilities for using ANNs: (i) utilization of an explicit memory in an autoregressive form or (ii) utilization of an implicit memory using a recursive structure within the ANN resulting in what is referred to as recurrent neural networks or RNNs. Generally, RNNs outperform feedforward networks in representing dynamic behavior for similar number of parameters. As an example, Fig. 1 shows a comparison of validation plots for open-loop predictions of the output variable in the case study later to be introduced in this article with NARX and RNN models. RNNs were recognized early on as a suitable approach for modeling dynamic systems (Funahashi &Yuichi, 1993) and there are numerous articles published using these structures for various control systems applications (Pan & Wang, 2011; Yan&Wang, 2012; Lanzetti et al., 2019; Patan, 2014).

Since the motivation of the present article is the development of solutions toward autonomous operations, having a basic method for learning system dynamics alone is not enough. Therefore, in this paper, we tackle the problem of autonomous learning of unknown system dynamics, where the dynamic system in consideration can be subject to changes over time. In Section 2, we provide the concept of a superstructure that can be used for autonomous model learning and discuss the various components and the possible technologies that can be used in this superstructure, including an automated ML component for neural architecture selection. In Section 3 we consider in detail the automated ML component and present in detail, how such an algorithmic

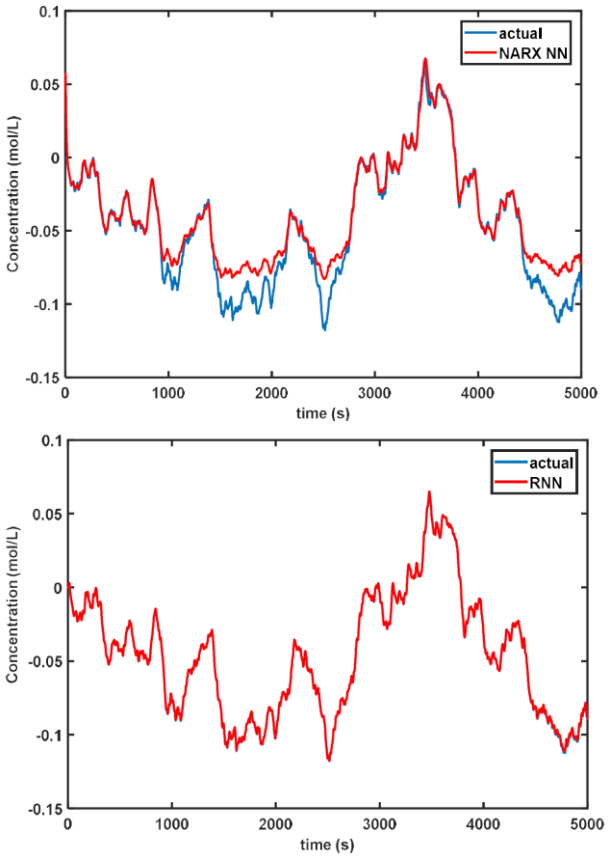

Fig. 1. Open-loop predictions using validation data for changes in product concentration in a chemical reactor using a NARX NN (20 hidden neurons and 4 step delay for inputs and measurements) and an RNN (10 hidden neurons with 4 recurrent layers)

step can be implemented. Finally, in Section 4 we present a case study where we deploy a reduced version of the autonomous model learning superstructure (AMLS) for predicting the discharge concentration of a simulated chemical reactor.

2. AUTONOMOUS MODEL LEARNING

We will pose the autonomous model learning problem for a so-called brownfield setting. This entails that at the initial step historical data will be available for the inputs and outputs of the plant in consideration. In the best possible case, the historical data will be frequency rich to learn a satisfactory model and assuming the resulting applications created from the learned models provide a similar persistency of excitation, relearning of the models will also be possible from historical data as the system experiences changes over time. If this is not the case, an interaction with the system via the inputs to generate the necessary output data will be required and this will then be equivalent to starting with a greenfield plant, where no historical data is present.

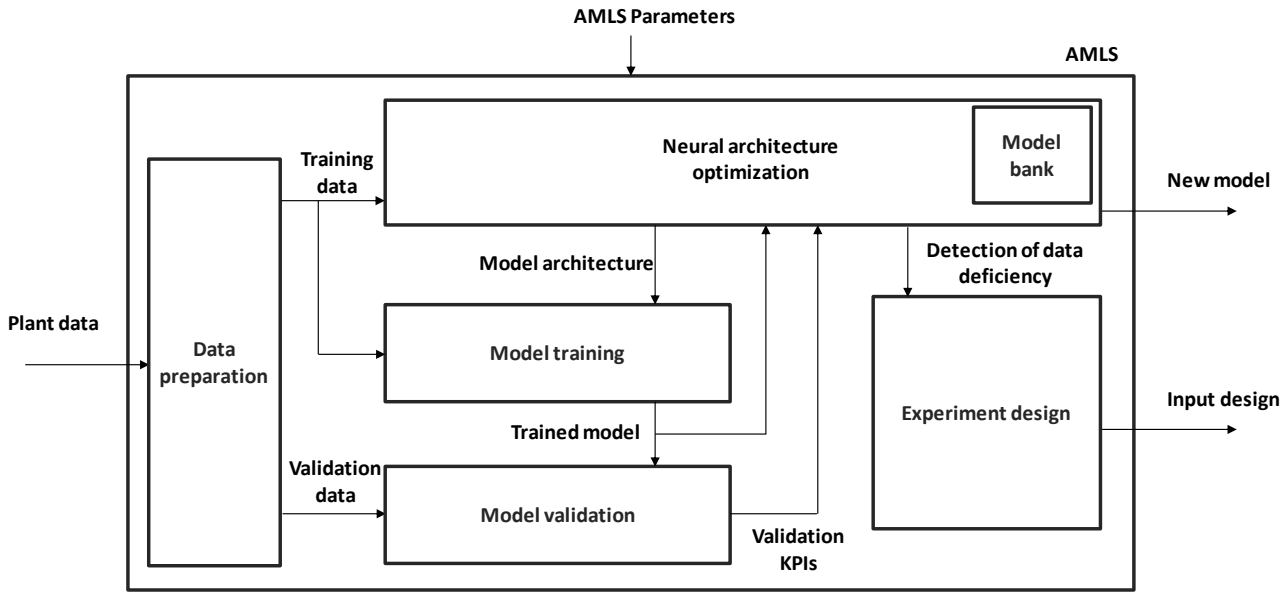

Fig. 2. The AMLS with six distinct components is shown. The arrows and the associated text represent the information flow and the boxed elements represent the components with the processing and storage tasks

Fig.2 shows the general arrangement of the functions needed in the AMLS. A combination of six distinct components are envisioned to provide the necessary functionality. The flow of information and the execution of the different tasks are carried out in the following order:

1. Plant data is ingested by the data preparation block, which carries out various filtering tasks including outlier removal and missing data imputation. Data preparation block can also be envisioned to scan over large time series signals to seek frequency rich periods corresponding to start-ups, shut-downs, or grade changes. Data is also organized here to form batches of training and validation sets.
2. The training data is received by the neural architecture optimization block, where the data is processed first by a classification algorithm, which is linked to a model bank containing models and corresponding architecture parameters. This classification provides an initial guess to start the architecture optimization process.
3. The training data and the model architecture is provided to a model training block, which comes up with a parametrization corresponding to the provided architecture. The training data can be further split into multiple batches by this block depending on the underlying algorithms and the software used. The model training block will also control the training process according to default or user specified AMLS parameters, which determine among others termination criteria or number of repeated trainings with different initial guesses for the parameter optimization
4. The trained model and validation data are provided to a model validation block, which simulates the provided model with validation input data and compares simulation results with the validation output data to generate validation results. The validation results can contain various key performance indicators (KPIs) such as integral errors, maximum deviations, or other criteria of importance.
5. The validation KPIs and the trained model are received by the neural architecture optimizer, which assesses based on the overall targets, if the model is satisfactory, if it needs to be further optimized, or if the optimization process needs to be terminated due to training data deficiency. The model and the intermediate results are also saved at the model bank. If the model is satisfactory, the model is provided to applications downstream of the AMLS.
6. If a data deficiency is detected the experiment design block is executed. It is envisioned that this block will not only come up with the plant testing plan, but it will also carry out and monitor the plant testing effort in coordination with the various control and optimization loops for manipulating the plant inputs. This block is not further discussed in this paper and is one of the subjects for future work.
7. Finally, the model validation block takes on the task of validating the existing model continuously with new validation data and in case a significant drop in validation KPIs is detected a new cycle will be started to relearn a new model

As noted in the description of the workflow, there are also configuration parameters, which will specify the thresholds used in the automated evaluation of the plant data by the AMLS. These parameters can be provided or adjusted externally by plant operators or a default setting can be considered to have a completely autonomous system. The details of such configuration options will not be discussed here as the purpose in this section is to provide the AMLS at a concept level. The details of the experiment design block of the AML is also not discussed in this paper. The question on how the existing historical data or the unsuccessful modeling attempts can be used in the experiment design are not considered. However, there is considerable prior work in this area and interested readers can follow e.g. Hjalmarsson, Gevers, & De Bruyne, 1996; Forssell & Ljung, 2000; Rojas et.al., 2007; Bavdekar & Mesbah, 2016. A detailed discussion of the neural architecture optimization step is provided in the next section.

3. NEURAL ARCHITECTURE OPTIMIZATION

As discussed in the introduction, when it comes to modeling time-series data, RNNs enjoy several advantages over other methods such as high scalability and capability of modeling multiple-output data. Because an RNN simulates a discrete-time dynamical system, it has the build-in capability of modeling sequential data, which is typically not offered by other variants of neural networks. At the same time RNNs have the disadvantage of being more difficult to handle in the training, mainly due to vanishing or exploding gradients. Therefore, it is important to have a good guess for hyperparameters. Therefore, we are considering the problem of how to suggest RNN hyperparameters for unseen time-series. The hyperparameters we consider are: the number of recurrent units (R) and the number of layers (N) for a fully connected architecture specifically in the form of so called "Elman Network" (Elman, 1990). In this paper we will focus on these two parameters, however the idea can be extended to different machine learning models and model structures. We also only consider multiple-input-single-output (MISO) systems in the current paper to show the feasibility of the approach as multiple-input-multiple-output (MIMO) systems can be considered as a collection of MISO systems, where the state and hence output interactions are assumed to be captured by the RNNs.

The problem at hand is composed of finding a pair of R and N for an unseen time-series such that the model complexity is acceptable while having a good prediction accuracy of the model. This trade-off needs to abstract the system complexity arising from system delays, nonlinearities, internal states, etc. and find a model structure that is large enough to be able to predict the system output. If R and N are chosen too small, the model will not be able to capture the dynamics, whereas if R and N are excessively large the prediction performance will either saturate or worsen depending on the information content and the amount of training data.

The proposed solution consists of a two-step approach. In the first step prior knowledge is used in order to determine the best starting pair of R and N for a previously unseen time-series. In the second step a heuristic search finds the most suitable R and N.

3.1. Search initialization

Two approaches are considered for the search initialization step: collaborative filtering and classification via convolutional neural networks (CNNs). Both approaches rely on the availability of a model bank formed by a heuristic search process as illustrated in Fig. 3. The collaborative filtering approach also requires the trace of the search process.

Collaborative filtering is a technique extensively used in recommendation systems (Su & Khoshgoftaar, 2009). In the present work collaborative filtering is used with a similarity-based vector model as illustrated in Table 1.

	R^1N^1	R^1N^2	R^2N^1	…	R^iN^j
System 1	-	KPI^1	KPI^2		-
System 2	KPI^3	-	KPI^4		KPI^5
System 3	KPI^6	KPI^7			KPI^8
…					
System N		KPI^k			KPI^{k+1}

Table 1. The system and hyperparameters interaction matrix.

When a previously unseen time series is received the system takes the current median combination of R and N values and trains a model with the new training data. The resulting performance is then used as an input to a prediction step, where multipliers obtained from the matrix factorization of the interaction matrix is used to generate an R and N combination that is recommended as a good initial guess for the heuristic search. A widely used algorithm for this purpose is Funk SVD (Funk, 2006), which can handle interaction matrices with missing elements, which makes it suitable for using in the current setting, where the heuristic search for different systems will never cover all possible R and N combinations.

In the alternative method, standard size snapshots of augmented time-series with both input and output measurements are labelled with corresponding system identifiers and CNNs are trained to classify snapshot inputs to system identifier outputs. Multiple snapshots from the same system can be used as training data for the classifier. When a snapshot from a previously unseen system is entered, an assignment to one of the known systems will be made and optimized R and N values for the known system will be used as initial guesses for the new one.

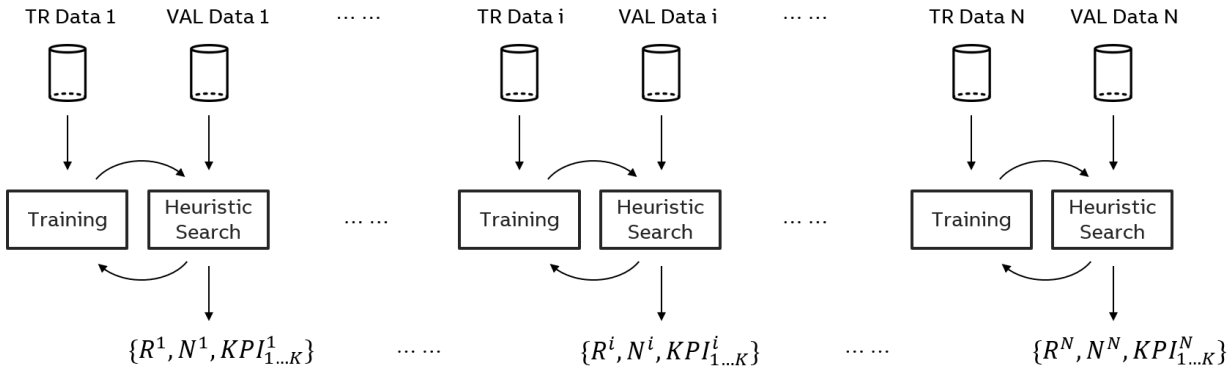

Fig. 3: Building a model bank. As the system learns more and more system models, it keeps a history of how different combination of hyperparameters were performing during the heuristic search procedure. This trace of performance and hyperparameter combinations are useful for using methods like collaborative filtering to recommend hyperparameters to previously unseen time series by means of few trials.

3.2. Hyperparameter search

The heuristic search used in the case study of this paper is carried out in two parts. The algorithm tries to reach the complexity-accuracy trade-off by first increasing R and N values from the initialization point linearly and after the prediction performance improvement of linear increase on validation data becomes smaller than a threshold, the search tries to compactify the model structure by decreasing R and N iteratively. This complexity reduction is only accepted if the prediction accuracy loss is tolerable. The pseudo-code for search is shown below:

```
R=R_init, N=N_init
error = error_init
yval = sim(real_model)
while improvement > imp_threshold
      model_i = Train_model(data, R, N)
      y=sim(model_i)
      error_new = sum(abs(yval-y))
      improvement = error – error_new
      if improvement > impr_ threshold
            increment(R)
            increment(N)
      else
            retry K times
      end
end
while loss < loss_threshold
      model_i = Train_model(data, R-1, N)
      y=sim(model_i)
      error_new_R = sum(abs(yval-y))
      model_i = Train_model(data, R, N-1)
      y=sim(model_i)
      error_new_N = sum(abs(yval-y))
loss = min(error_new_R,error_new_N)-error
      if loss < loss_threshold
reduce(R) if error_new_R< error_new_N
      else
            retry K times
      end
reduce(N) if error_new_N< error_new_R
      else
            retry K times
      end
end
```

The important step to note is the retry commands before termination of the search algorithms. This step is needed since the RNN training process is stochastic and can fail or result in inferior performance depending on the initial conditions of the training algorithm. For larger values of R and N values the training starts to become computationally demanding and building a control layer to terminate the training depending on the evolution of the learning rate becomes advisable. The hyperparameter search problem lends itself to parallel computing and although we do not state an efficient way of parallelizing the described approach here, the dual search in the R and N directions during compactification parallelizes without any effort. The search heuristic we present here can be substituted with more elaborate optimization algorithms. We have used also a genetic algorithm (GA) configured for search over integer variables and observed that a larger number of training instances was needed, and the performance improvement was not significant. For illustration purposes we provide the response surface showing the validation error observed during the GA run for the case study system in Fig. 4.

4. CASE STUDY

We build a simplified AMLS with fixed initial hyperparameters and without an experiment design module for a non-isothermal continuous stirred tank chemical reactor (CSTR) problem, a variant of which can be found at Chen et al., 1995. The reactor has two reactants, a product, and a by-product in two reactions. The objective of the model-based application is to build a soft sensor for the prediction of the product concentration in the output stream of the reactor.

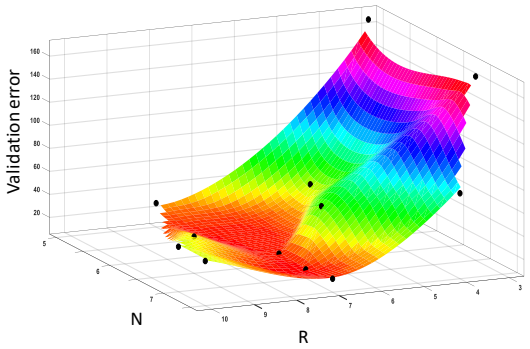

Fig. 4. Surface plot illustrating the dependence of validation error on the RNN hyperparameters. The hyperparameter combinations are shown with the black dots.

The CSTR is simulated as a six state ODE system with mass balances for the four species, a volume balance, and an energy balance equation. A level and a sluggishly tuned temperature control loop are included in the simulations. During the simulations the concentrations of the reactants in the feed stream is varying in a stochastic way, which is affecting the reaction rates and therefore the energy balance causing the temperature control loop to react. Fig 5. Illustrates the performance of the soft sensor in predicting the concentration of the main product. At 5000s, the rate of the main reaction is reduced by 25% to represent a catalyst poisoning event. The AMLS detects the loss of predictive performance and trains a new model, which is engaged at 10000s and recovers good prediction performance. The temperature is not provided as a measurement to the soft sensor and despite the disturbance caused by temperature variations on the reaction rates the system is able to provide good predictions.

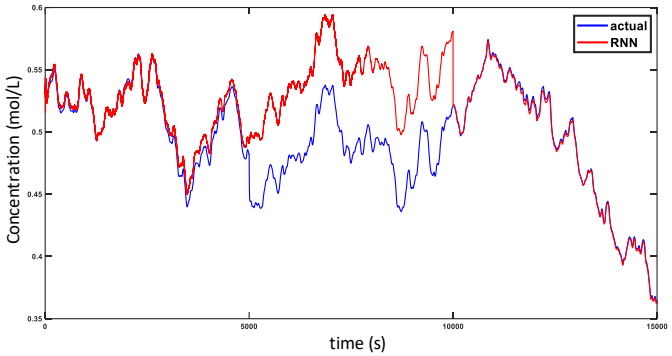

Fig. 5. Time trajectory of the product concentration variation in the CSTR of the case study.

5. SUMMARY

We pose a problem of autonomous system identification and propose a conceptual construct as a solution. The various components and features of this concept is discussed and possible methods to address the emerging challenges are studied. A partial realization of this AMLS concept is deployed on a chemical reactor soft sensor application as a case study and promising results are obtained. The points discussed in this paper can motivate several research activites relevant for the industry.

References

Mercangöz, M., Cortinovis, A. and Dominguez, L. (2019). Machines learning machines: AI Learns to Mimic Process Dynamics. in ABB Review Q4 / 2019.

Spielberg, S. P. K., R. B. Gopaluni, and P. D. Loewen. "Deep reinforcement learning approaches for process control." *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*. IEEE, 2017.

Zhu, X. "Machine teaching: An inverse problem to machine learning and an approach toward optimal education." Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

Evans, R., and Gao, J. "Deepmind AI reduces Google data centre cooling bill by 40%." *DeepMind blog* 20 (2016).

Kober, J., and Peters, J.. "Imitation and reinforcement learning." IEEE Robotics & Automation Magazine 17.2 (2010): 55-62.

Coulson, J., Lygeros J., and Dörfler, F., "Data-enabled predictive control: in the shallows of the DeePC." *2019 18th European Control Conference (ECC)*. IEEE, 2019.

Kashiwagi, H. "Nonparametric system identification." *CONTROL SYSTEMS* (2009).

Ljung, L. (1998). System identification, Signal analysis and prediction, Springer, pp. 163–173, 1998.

Ljung, L.; Hjalmarsson, H.; Ohlsson, H. Four encounters with system identification. Eur. J. Control, 2011,17(5−6), 449−471.

Chan, L.L.T., Liu, Y., and Chen, J., "Nonlinear system identification with selective recursive Gaussian process models." *Industrial & Engineering Chemistry Research* 52.51 (2013): 18276-18286.

Funahashi, K., and Yuichi N. "Approximation of dynamical systems by continuous time recurrent neural networks." *Neural Networks* 6.6 (1993): 801-806.

Pan, Y., & Wang, J. (2011). Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Transactions on Industrial Electronics*, *59*(8), 3089-3101.

Lanzetti, N., Lian, Y. Z., A., Cortinovis, Dominguez, L., Mercangöz, M. and Jones, C. (2019). Recurrent Neutral Network based MPC for Process Industry, 18th European Control Conference (ECC), Napoli, Italy, June 25-28, 2019.

Yan, Z., & Wang, J. (2012). Model predictive control of nonlinear systems with unmodeled dynamics based on feedforward and recurrent neural networks. *IEEE Transactions on Industrial Informatics*, *8*(4), 746-756.

Patan, K. (2014). Neural network-based model predictive control: Fault tolerance and stability. *IEEE Transactions on Control Systems Technology*, *23*(3), 1147-1155.

Rojas, C. R., Welsh, J. S., Goodwin, G. C., & Feuer, A. (2007). Robust optimal experiment design for system identification. Automatica, 43(6), 993-1008.

Forssell, U., & Ljung, L. (2000). Some results on optimal experiment design. *Automatica*, *36*(5), 749-756.

Bavdekar, V. A., & Mesbah, A. (2016). Stochastic model predictive control with integrated experiment design for nonlinear systems. *IFAC-PapersOnLine*, *49*(7), 49-54.

Hjalmarsson, H., Gevers, M., & De Bruyne, F. (1996). For model-based control design, closed-loop identification gives better performance. *Automatica*, *32*(12), 1659-1673.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, *2009*.

S. Funk. Netflix update: Try this at home.http://sifter.org/˜simon/journal/20061211.html, 2006

Chen, H., Kremling, A., & Allgöwer, F. (1995, September). Nonlinear predictive control of a benchmark CSTR. In Proceedings of 3rd European control conference (pp. 3247-3252).