Construction of a Surrogate Model: Multivariate Time Series Prediction with a Hybrid Model

Clara Carlier[1,2], Arnaud Franju[2], Matthieu Lerasle[1], Mathias Obrebski[2]

(1) Statistics Department of CREST, 5 Avenue Henry Le Chatelier, 91120 Palaiseau, France. E-mail: {clara.carlier, matthieu.lerasle} @ (ensae.fr)
(2) DEA-TDV team of the Renault group, 1 avenue du Golf, 78280 Guyancourt, France. E-mail: {clara.carlier, arnaud.franju} @ (renault.com)

Abstract - *Recent developments in advanced driver-assistance systems necessitate an increasing number of tests to validate new technologies. Carrying out these tests on track would take too long, this is why automotive groups rely on simulators to perform most tests.*
These simulators serve various purposes and enable the creation of scenarios with varying degrees of realism. In this article, we focus on a simulator that generates vehicle behavior using time series.
But the reliability of these simulators for constantly refined tasks is becoming an issue and, to increase the number of tests, the industry is now developing surrogate models, that should mimic the simulator's behavior while being much faster to run on specific tasks.
In this paper, we develop a surrogate model that replaces the simulator by generating time series based on user-defined input parameters. We first test several classical methods like random forests, ridge regression, or convolutional neural networks. Then we build three hybrid models that combine these methods and combine them to obtain an efficient hybrid surrogate model.

Keywords: *AD/ADAS, aggregation of experts, generative model, hybrid surrogate model, time series prediction*

1. Introduction

Certification of Autonomous Driver (AD) and Advanced Driver-Assistance Systems (ADAS) are highly sensitive applications that should be carried out very carefully. The numerous onboard sensors in cars give access to a large amount of information. There are many strict regulations, making it necessary to carry out a lot of real on-track experiments over long distances. The Renault group has decided to develop digital platforms to simulate driving assistance and vehicle automation systems to create simulation data. These data will complete or even replace the real experiments done on track in the certification process.

Before integrating the simulations into the certification process, **the simulator must be calibrated** to generate data similar enough to the real on-track experiments. The overall goal is to develop a methodology that will gauge the quality of the simulator by comparing it to real on-track data and then calibrating and readjusting it. Once recalibrated, the simulator should be able to generate more realistic time series.

The proposed methodology for calibrating the simulator necessitates using it repeatedly, **which is not feasible**. Hence, it is necessary to develop **a computationally more efficient surrogate model** that mimics the simulator and then serves as a substitute. This paper presents several surrogate models that can replace the simulator by generating time series.

Section 1.2 describes in more detail the general objective and the resulting specific problem we seek to solve.

1.1. Renault simulator

The simulator is based on SCANeR™ studio software suite (AVSimulation, n.d.). It is dedicated to automotive and transport simulations. Among other things, it is designed to drive and test AD/ADAS by providing all the necessary tools to build a realistic virtual world by defining: road environments, vehicle dynamics, traffic, weather, ..., etc. The simulator requires different input parameters to define the desired experiment (initial speed, braking efficiency, ..., etc) and then generate the associated time series describing vehicles' behavior (speed, acceleration, ..., etc). We define a **scenario** as a combination of input parameters and their associated time series.

1.2. Objectives and full process

The proposed methodology to calibrate the simulator is articulated between the resolution of an *inverse problem* and a *direct problem*. The whole process is similar to the one used in (Giraldi, et al., 2017), it is briefly described in Fig. 1.

The overall problem is the inverse one:

1. we have one so-called reference experiment, on-track time series, and its associated input parameters called nominal values,

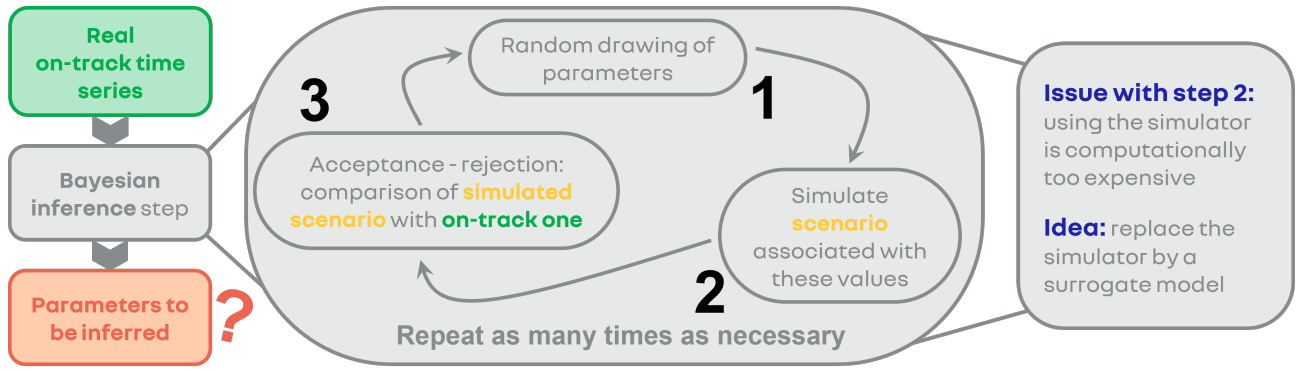

Figure 1: Summary of the general process. The three blocks on the left represent the inverse problem which consists in finding the values of the input parameters associated with the reference on-track test. The middle section describes how ABC methods work. The last part is about the issue we are facing and how we intend to solve it.

2. and we want to recover the input parameters that would simulate the *closest* time series to the reference ones.

To resolve the inverse problem, we use Approximate Bayesian Computation (ABC) methods which are likelihood-free inference schemes. Several steps are repeated iteratively and one of them requires generating time series from a set of input parameters. First, we (1) draw candidate parameters according to priors; then (2) we simulate the time series associated with these candidate values, this is precisely what the simulator is for; and finally (3) we go to the acceptance-rejection step if the simulated time series is close enough to the reference ones, we accept the candidate parameters, otherwise, they are rejected. We repeat these three steps, as many times as necessary or desired.

Fig. 2 shows the benefit of this parameter calibration. We compare the reference time series to the time series simulated with nominal values vs. calibrated parameters. Time series are closer to baseline values when using inferred rather than nominal parameters.

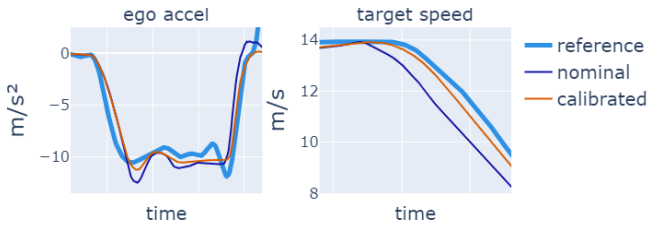

Figure 2: Reference on-track test and time series simulated with nominal parameters vs. calibrated parameters

At this point, **we are facing a problem**. In step (2), the simulator is computationally too expensive to use iteratively. The idea is to develop a surrogate model that will mimic and replace the simulator and then perform this step during the ABC process. It corresponds to solving the direct problem. It is a classical learning problem involving multivariate time series.

In summary, to solve the inverse problem of calibrating the simulator, it is necessary to first solve the direct problem, which involves constructing a surrogate model. This article deals with this specific task.

1.3. Surrogate model

Surrogate models are largely used in all types of domains and contexts, they have already demonstrated their usefulness and efficiency using a wide variety of possible methods: Polynomial Chaos Expansions (Sraj, et al., 2016), Radial Basis Functions and Kriging (Beglerovic, Stolz, and Horn, 2017), Bayesian Surrogate Models (Ford, Moorhead, and Veras, 2011), Surrogate Response Surface Models (Mattis and Wohlmuth, 2018), Artificial Neural Networks (Xu, et al., 2020).

Surrogate models are also widely used in the automotive field and have demonstrated their accuracy in many applications like car seats (Long, Liao, and Yu, 2021), suspension components (Jiang, et al., 2021), human-product interaction (Ahmed, et al., 2018) or autonomous vehicles validation (Beglerovic, Stolz, and Horn, 2017).

In this paper, we construct a surrogate model with supervised machine learning methods that mimics and replaces the Renault simulator by predicting the simulated time series.

Contrarily to the time series forecasting framework where the objective is to predict the future from the past, the surrogate model aims at predicting a whole time series from a set of parameters. More precisely, it is a generative model. The dataset used to build the model is a set of beforehand simulated scenarios output by SCANeR™. The various input parameters need to be carefully chosen so that the database correctly represents the desired parameter definition space.

The surrogate model must be as accurate as possible:

- the simulated scenarios have to be close enough to the on-track experiments to prove their reliability,
- and the surrogate model, which replaces it, must be as close as possible to the simulator's behavior, to make its use viable in the calibration process.

Moreover, with a more accurate model, the final time series are even better as can be appreciated in Fig. 3.

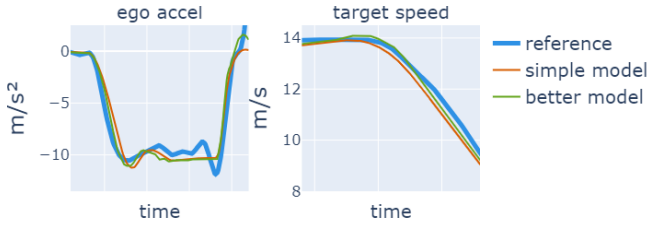

Figure 3: Reference on-track test and time series simulated with parameters inferred with a simple model vs. a better model

1.4. Mathematical description

We consider a dataset $(\mathbf{x}_k, \mathbf{y}_k)$. Each $\mathbf{x}$ is a vector of $\mathbb{R}^M$ and corresponds to a set of parameters given to SCANeR™. Each $\mathbf{y}$ is a vector of N time series $y_t^{(n)}$ of duration T_n output by SCANeR™.

We aim to predict the entire time series: given $\mathbf{x}$, we want to predict $\mathbf{y}$. These two are linked through the deterministic simulator $\mathcal{S}^*$: for a given set of parameters, time series are generated

$$\mathbf{y}_k = \mathcal{S}^*(\mathbf{x}_k) \quad (1)$$

Constructing the surrogate model amounts to building a predictor $\widehat{\mathcal{S}}$ which returns all steps of the time series. Given $\mathbf{x}$, we predict $\mathbf{y}$ by $\widehat{\mathcal{S}}(\mathbf{x})$. This corresponds to a classical supervised learning problem.

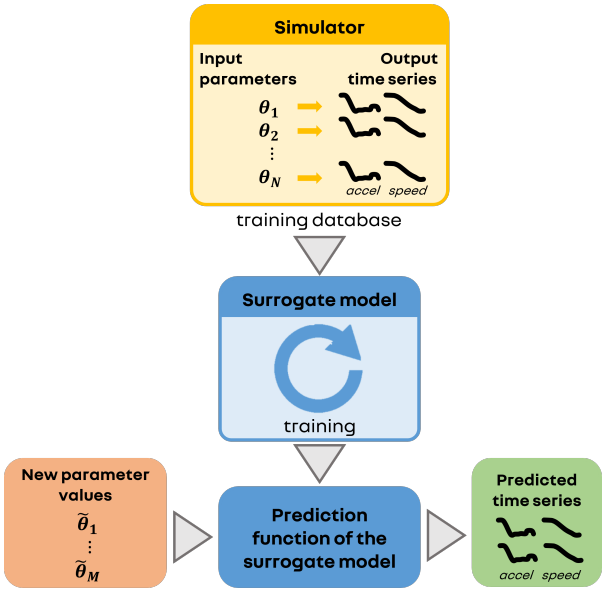

Figure 4: Summary of the training and predicting process of the surrogate model

2. Data description

To build the surrogate model, we construct a dataset with the SCANeR™ simulator. We look specifically for an emergency braking scenario involving two vehicles following each other and driving at given initial constant speeds. The front vehicle (named **target**) brakes and the following one (named **ego**) activates its emergency braking to avoid a collision.

Table 1: Keywords definition

Keyword	Definition
AEB	automatic emergency braking
calibration	methodology which assesses the quality of the simulator compared to real on-track experiments and subsequently readjusts it to optimize its utilization
distance	corresponds to the distance between the two vehicles, ego and target
ego	designates the name of the vehicle whose AEB is being tested
nominal values	the parameter values corresponding to a real on-track experiment
physical = real	defines when the experiment is realized on-track
scenario	corresponds to a combination of input parameters and associated time series, it is one experiment
target	designates the name of the vehicle that brakes and that the ego vehicle must avoid

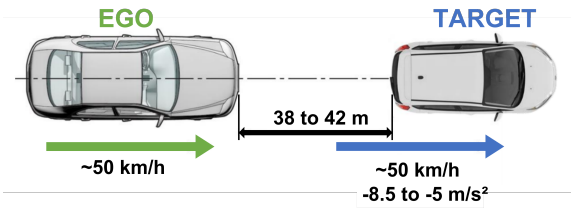

Figure 5: Emergency braking scenario involving ego and target vehicles

We give the simulator the values of the needed input parameters to define the type of scenario and the characteristics of the two vehicles, like initial speeds and braking efficiency of ego. The output time series will describe their behaviors: speed and acceleration of the ego vehicle, the speed of the target vehicle, and the distance between them.

The dataset contained several scenarios: time series generated by distinct sets of parameters. Our goal is to use this dataset to build a surrogate model that is, a machine learning algorithm taking as input parameters and returning time series that should be close to those that SCANeR™ would have generated.

2.1. Input parameters

For this specific scenario, SCANeR™ takes in input 7 parameters, divided into two groups. Scenario parameters define the initial speeds of the vehicles, the initial braking strength of the front vehicle, and the initial distance between vehicles. Then, there is ego vehicle dynamic and environment parameters which define for example front and rear braking efficiency or Autonomous Emergency Braking (AEB) latency.

The simulator is deterministic, so the training dataset's variability only comes from input parameter variability. We draw each parameter from a uniform law, independently from each other. Intervals of each uniform law are defined by taking a percentage around the defined nominal values (Table 2). The automotive certification authorities give these nominal values, defining the values to be tested.

Table 2: Intervals used to generate input parameters from uniform law

Type	Parameter	Nominal value	Interval
scenario	ego initial speed	50	[48, 52]
	target initial speed	50	[48, 52]
	target braking force	-6	[-8.5, -5]
	initial distance between the two	40	[38, 42]
MADA & ENV	front braking efficiency	1	[0.4, 1.6]
	rear braking efficiency	1	[0.4, 1.6]
	AEB latency	0	[0, 55]

The distribution of each parameters values is represented in Fig. 6. We notice that ego and target init speed histograms are not uniform. It's because there are some constraints on the parameter values and if they don't comply, the simulator can't generate the associated time series so the values are automatically rejected. For example, the ego's initial speed must be larger than the target's one.

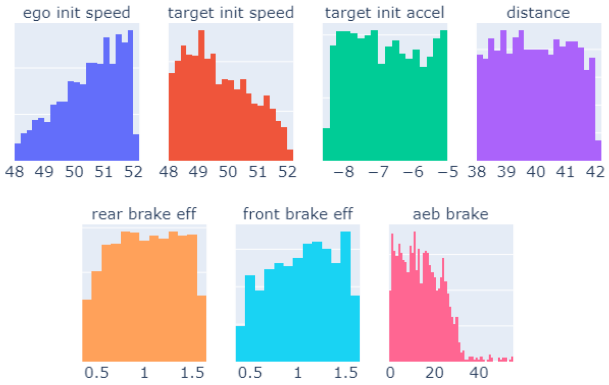

Figure 6: Histograms of input parameters (percentages)

2.2. Output time series

Fig. 7 shows the time series describing the speed and the acceleration of the ego vehicle, the speed of the target vehicle, and the evolution of the distance between them. This gives an idea of the different profiles contained in our initial dataset.

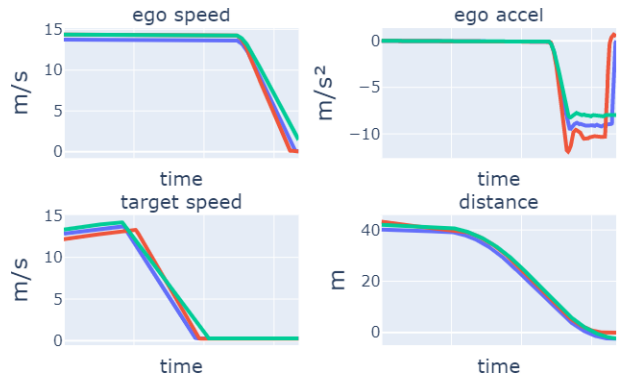

Figure 7: Time series corresponding to three experiments for three distinct sets of parameters

The dataset is divided into three parts:

- **train**: 1442 scenarios to fit the first models (4-RF, CNN, ...),
- **validation**: 100 scenarios to tune hyperparameters for hybrid and aggregated models (choices for each time step),
- **test**: 100 scenarios to evaluate a final model (aggregated or hybrid 1 or 2) unbiasedly.

3. Surrogate model construction

To measure our risk, we consider the root mean squared error. For u the vector containing the true values and v the predicted values, the classical RMSE is given by

$$\text{RMSE}(u, v) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(u_i - v_i)^2} \qquad (2)$$

The RMSE will be used to select the best approach. We will use it to compare beforehand simulated data with predicted data created with each model.

In Section 3, we will calculate the RMSE on the train and validation set. The test set will be used only at the end to tune hyperparameters of hybrid and aggregated models. In Section 4, we will calculate the RMSE on the validation and test set.

We will also measure the training and prediction time of each model. The prediction time is calculated for 100-time series.

3.1. Selection of the most promising method

We realized a benchmark and tested different prediction methods to compare them, like k-nearest neighbors (k-NN), kernel ridge regression (KRR, *with laplacian kernel*) (Exterkate, et al., 2016), simple convolutional neural networks (CNN), polynomial chaos expansion (PCE) (Blatman and Sudret, 2011; Crestaux, Le Maître, and Martinez, 2009), random forests (RF), Deep Forest (DF) (Zhou and Feng, 2018).

Concerning random forests, we developed a global model that predicts all the time series (1-RF) and another one obtained by training four distinct random forest models, one per time series (4-RF). We also tried to combine random forests with dimensionality reduction methods, like classical or functional PCA (PCA-RF) (Shang, 2014; Wohlenberg, 2021).

Table 3 summarizes the results obtained with each of these methods and compares the RMSE and the computation times obtained.

In terms of RMSE, the k-NN algorithm produces the worst results. The PCA-RF results are not very good either, it reduces a bit the training time compared to RF but not the prediction time. The results of KRR, DF, and 1-RF are quite similar and a bit better. The CNN and the 4-RF results are the best.

Table 3: RMSE and computation times on train and validation sets for the seven methods (*) *laplacian kernel* ; (**) *for 100-time series*

		k-NN	KRR (*)	CNN	DF	1-RF	4-RF	PCA-RF
RMSE $(\times 10^{-2})$	train	9.23	**0.04**	1.22	4.26	1.24	**0.71**	1.95
	validation	30.15	7.21	**2.31**	7.05	7.27	**3.69**	12.35
training time		0.05 sec	0.22 sec	59 min	13 min	42 sec	53 sec	7.42 sec
prediction time (**)		0.01 sec	0.02 sec	1.39 sec	0.59 sec	0.08 sec	0.15 sec	0.16 sec

Table 4: RMSE for each time series on validation set with the seven methods

method	ego speed	ego accel	target speed	dist.	mean
k-NN	11.00	64.77	8.02	36.83	30.15
KRR	2.20	7.14	1.63	17.86	7.21
CNN	0.18	3.85	**1.06**	**4.16**	**2.31**
DF	1.32	8.88	5.80	12.19	7.05
1-RF	1.36	9.54	4.86	13.31	7.27
4-RF	**0.12**	**1.47**	2.34	10.84	3.69
PCA-RF	2.33	20.53	5.58	20.96	12.35

3.2. More detailed comparison

Let's now detail these results by calculating the RMSE associated with each time series obtained with each method. The results are given in Table 4.

We now notice important differences. Each method performs more or less well with each time series. In this case, CNN performs best for target speed and distance while 4-RF performs best for the ego series.

We then want to detail further the error values: for all methods, the RMSE is calculated for each time step, Fig. 8 represents it.
We can distinguish which method performs better at each time step. For distance, CNN outperforms other methods, but it's not so clear for the other series. These results suggest building hybrid models.

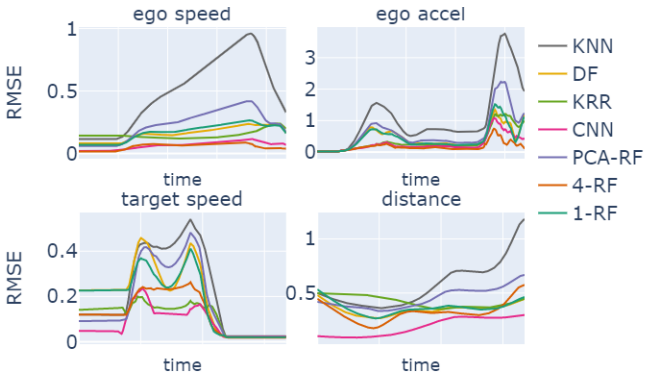

Figure 8: For each method, we compute the RMSE obtained at each time step on validation set

4. Hybrid and aggregated models

We build three new models: two with a hybrid approach and one with aggregation. The first two consist in choosing the best method at each time step and the third one performs a mixture of methods by giving them different weights computed with an expert aggregation.

The description of the three models is:

- **Hybrid 1:** for each time step, we select the best method among the seven proposed;
- **Hybrid 2:** we keep the three most used methods in Hybrid 1 (CNN, 4-RF and PCA-RF, see Table 5) and select the best one for each time step;
- **Aggregated:** it is built with an Exponential Weighted Aggregation (EWA) (Mourtada, 2016).

Fig. 9 shows which method performs best at each time step and the weights given to each method for each time step are shown in Fig. 10.

Table 5: Number of time steps for which the methods are selected in the hybrid model 1

CNN	4-RF	PCA-RF	KRR	1-RF	DF	k-NN
1081	997	430	93	53	28	2
40%	37%	16%	3%	2%	1%	¿1%

4.1. Numerical results

We now summarize all numerical results obtained with these three approaches. First, we calculate RMSE values on the **validation set** which has been used to tune hybrid and aggregated approaches: for each time step, the choice of the methods for hybrid models, and the weights for aggregated one. Then, we calculate RMSE values on the **test set** to confirm the results and evaluate the generalization quality.

4.1.1. Validation (Table 6)

On the validation set, the three new models are better than the CNN and 4-RF models. The aggregated is quite better. Hybrid 1 and Hybrid 2 are similar. To conclude on the best model, let's now see **how these results generalize to the test set**.

Table 6: RMSE for each time series with CNN, 4-RF, hybrid and aggregated models on the validation set

method	ego speed	ego accel	target speed	dist.	mean
CNN	0.18	3.85	1.06	4.16	2.31
4-RF	0.12	1.47	2.34	10.84	3.69
Hybrid 1	0.11	1.46	0.93	4.16	1.66
Hybrid 2	0.11	1.46	1.04	4.16	1.69
Agg.	**0.07**	**0.59**	**0.24**	**1.36**	**0.56**

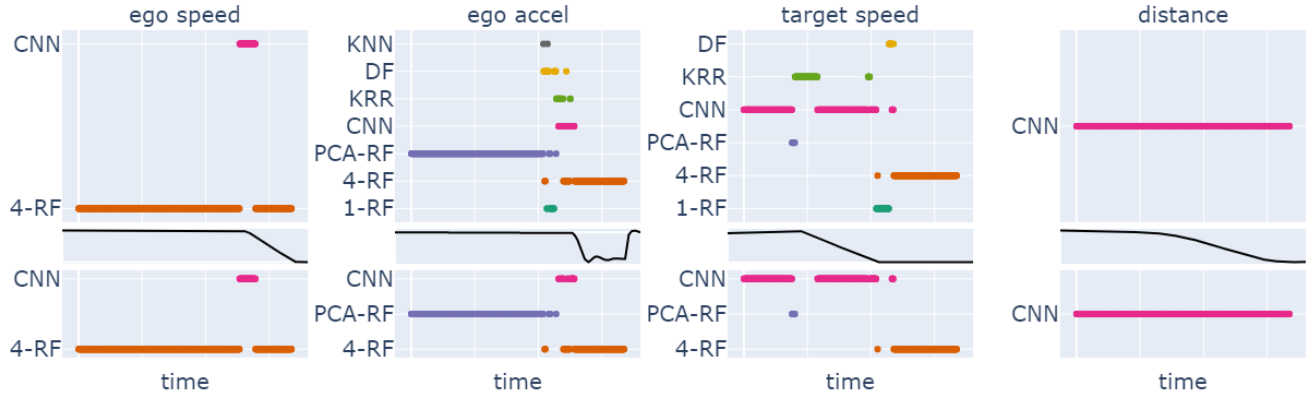

Figure 9: Selected method at each time step on the validation set.
First line is hybrid 1: among the 7 methods; Last line is hybrid 2: among the 3 best methods.

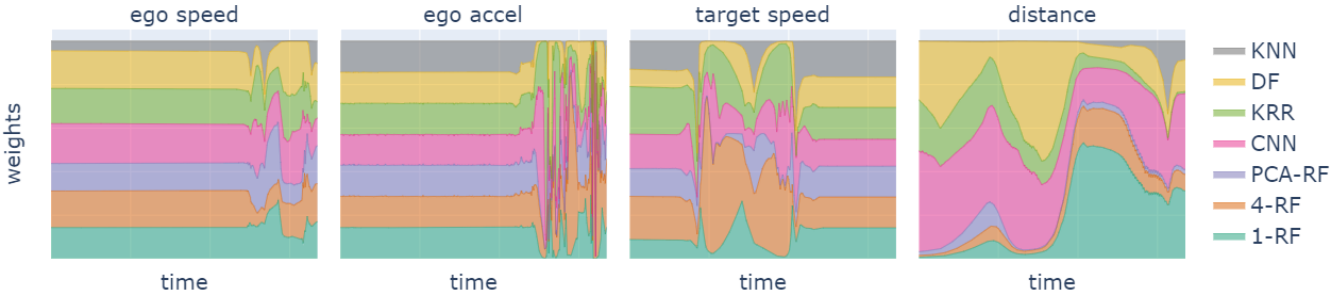

Figure 10: Associated weights for each expert at each time step on the validation set

4.1.2. Test (Table 7)

On the test set, the aggregated model is no longer the best, it is even worse than the CNN model. We conclude that the distribution of the weights cannot be generalized and is too specific to the validation set. However, the results of the two hybrid models are still better than the CNN and 4-RF ones.

With the Hybrid 1 model, the prediction of target speed is deteriorated. From these results, we can see **the advantage of the Hybrid 2 model**: for each time series, it improves predictions although the choice of methods was made on the validation set. Whatever the database used to calibrate, it is the Hybrid 2 model that generalizes best.

Table 7: RMSE for each time series with CNN, 4-RF, hybrid and aggregated models on the test set

method	ego speed	ego accel	target speed	dist.	mean
CNN	0.23	3.34	1.00	2.52	1.77
4-RF	0.13	1.36	2.36	9.84	3.42
Hybrid 1	0.12	1.35	1.12	2.52	1.28
Hybrid 2	0.12	1.35	**1.00**	2.52	**1.25**
Agg.	0.50	3.66	1.08	3.38	2.16

4.1.3. Computation times (Table 8)

The aggregated model is very time-consuming, although it is still faster than SCANeR™. This model does not bring anything more than the other approaches either in terms of accuracy or performance.

The two hybrid models improve the results but still multiply the prediction times by 10. Depending on the case, it will be necessary to decide which approach is the most judicious.

Table 8: Computation times for CNN, 4-RF, hybrid and aggregated models
(*) *for 100 time series*
(+) *add the prediction times of each method*

time	CNN	4-RF
train	59 min	53 sec
pred. (*)	1.39 sec	0.15 sec

time	Hybrid 1	Hybrid 2	Agg.
train (+)	0.29 sec	0.18 sec	2 min
pred. (*)	10.25 sec	8.59 sec	2 min

5. Conclusion

The true objective of this paper was to solve the direct problem: mimic and replace the simulator which is computationally too expensive. An iteration on the simulator SCANeR™ used at Renault takes at least 15 minutes, which becomes prohibitive when repeated simulations are required, as it will be the case in the ABC algorithm we intend to develop to solve the inverse problem.

After testing several methods and approaches, kernel ridge regression, convolutional neural network, and random forests stood out. The computation times of these surrogate models are much more reasonable.

A closer look at the prediction errors shows that some methods are more efficient in predicting certain time series and that difference can even be more interesting at a different time step of each time series.

We thus built three new models taking advantage of these preliminary remarks:

- Hybrid 1 selects the method with the lowest RMSE at each time step;
- Hybrid 2 selects the method with the lowest RMSE at each time step only among the three globally more efficient methods in Hybrid 1, which seems to avoid overfitting.
- Aggregated model uses an aggregation of the experts with an exponentially weighted algorithm.

These three hybrid models provide a clear improvement in predictions over the basic methods. The aggregated model seems to generalize worse on the test set. Hybrid 1 is quite good on both validation and test set but **Hybrid 2 generalizes better**.

Concerning training times, these three new models are more time-consuming. But the CNN model trains for at least one hour. So adding a few minutes will not be restrictive.
However, the prediction time is deteriorated, which might become restrictive for their use in ABC methods. The computation times in Table 9 show that there will be a trade-off to find for practitioners between accuracy and computation time.

To conclude, we easily built an overall reasonable model using random forests. To improve this benchmark, we specify which method to use at each time step of each time series. We emphasized the cost of this refinement and leave the final choice to the user's time constraints.

Moreover, the described methodology allows to systematically build a better model than those obtained with basic methods. The only question lies in determining whether the choices made at each time step are generalizable or not. In the presented application, we deliberately limited the scenarios to study a specific case and obtain the best possible model for this particular case. Consequently, the constructed models will not be applicable to other cases, but the methodology itself remains applicable and will allow to build a more accurate model.

Table 9: Computation time to generate 50.000 simulations one by one

4-RF	Hybrid 2	SCANeR™
1 minute	1 hour	5 days

References

Ahmed, S., Gawand, M. S., Irshad, L., and Demirel, H. O., 2018. Exploring the Design Space Using a Surrogate Model Approach With Digital Human Modeling Simulations. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1B.

AVSimulation, n.d. `https://www.avsimulation.com/scaner-studio/`.

Beglerovic, H., Stolz, M., and Horn, M., 2017. Testing of autonomous vehicles using surrogate models and stochastic optimization. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6.

Blatman, G. and Sudret, B., 2011. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, 230(6), pp. 2345–2367.

Crestaux, T., Le Maître, O., and Martinez, J.-M., 2009. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering & System Safety*, 94(7). Special Issue on Sensitivity Analysis, pp. 1161–1172.

Exterkate, P., Groenen, P. J., Heij, C., and Dijk, D. van, 2016. Nonlinear forecasting with many predictors using kernel ridge regression. *International Journal of Forecasting*, 32(3), pp. 736–753.

Ford, E. B., Moorhead, A. V., and Veras, D., 2011. A Bayesian surrogate model for rapid time series analysis and application to exoplanet observations. *Bayesian Analysis*, 6(3), pp. 475–499.

Giraldi, L., Maître, O. P. L., Mandli, K. T., Dawson, C. N., Hoteit, I., and Knio, O. M., 2017. Bayesian inference of earthquake parameters from buoy data using a polynomial chaos-based surrogate. *Comput Geosci 21*, pp. 683–699.

Jiang, R., Jin, Z., Liu, D., and Wang, D., 2021. Multi-Objective Lightweight Optimization of Parameterized Suspension Components Based on NSGA-II Algorithm Coupling with Surrogate Model. *Machines*, 9(6).

Long, J., Liao, Y., and Yu, P., 2021. Multi-Response Weighted Adaptive Sampling Approach Based on Hybrid Surrogate Model. *IEEE Access*, 9, pp. 45441–45453.

Mattis, S. A. and Wohlmuth, B., 2018. Goal-oriented adaptive surrogate construction for stochastic inversion. *Computer Methods in Applied Mechanics and Engineering*, 339, pp. 36–60.

Mourtada, J., 2016. Prédiction séquentielle par agrégation d'experts. *Master dissertation*.

Shang, H., 2014. A survey of functional principal component analysis. *AStA Advances in Statistical Analysis*, pp. 121–142.

Sraj, I., Mandli, K., Knio, O., Dawson, C., and Hoteit, I., 2016. Quantifying uncertainties in Fault Slip Distribution During the Tōhoku Tsunami using Polynomial Chaos. *Ocean Dynamics*, 67.

Wohlenberg, J., 2021. Functional Principal Component Analysis and Functional Data. *Toward Datascience*.

Xu, Z., Yu, C., Sun, H., and Yang, Z., 2020. The response of sediment phosphorus retention and release to reservoir operations: Numerical simulation and surrogate model development. *Journal of Cleaner Production*, 271, p. 122688.

Zhou, Z.-H. and Feng, J., 2018. Deep forest. *National Science Review*, 6(1), pp. 74–86.