A case Study on Black-Box Modeling of Non-Linear Systems for Rapid Development in ICE Control

PhD. Mario Chaves
Eng. Ariel Bepu
MSc. Oscar Duque
FEV Latin America

ABSTRACT

This paper presents a case study on the black-box modeling of non-linear systems with artificial intelligence (AI) techniques, specifically focusing on their practical applications in the control of internal combustion engines (ICEs).

Non-linear systems, characterized by their complex behavior, pose significant challenges in the field of control engineering. Traditional modeling approaches often fall short by accurately predicting the dynamic responses of such systems and require a comprehensive understanding of the system's dynamic behavior. This work explores various non-linear modeling techniques and evaluates their efficacy in simulating the intricate dynamics of ICEs, with the aim of enhancing control strategies.

The findings indicate that the practical applicability of non-linear modeling is feasible for certain engine sub-systems, especially in a fast development environment added to limited knowledge of the system's governing equations. As expected, this approach allows for better estimates during the iterative process of engine calibration.

ARTIFICIAL INTELLIGENCE METHODS FOR MODELING INTERNAL COMBUSTION ENGINE SYSTEMS

Internal combustion engines are complex non-linear systems, making them challenging to model accurately using traditional methods. These challenges stem from the intricate thermodynamic processes, varying operating conditions, and interactions between different engine subsystems.

Artificial Neural Networks (ANNs) have emerged as a powerful tool for capturing this non-linearity and developing accurate models of engine behavior. ANNs are data-driven models that can learn complex relationships between input and output variables without requiring explicit knowledge of the underlying physical processes, what is commonly called black-box models.

This makes them well-suited for modeling engine parameters such as torque, fuel consumption, emissions, and in-cylinder pressure. By training ANNs on experimental data or data from detailed physics-based simulations, it is possible to create models that can predict engine behavior across a wide range of operating conditions.

Unlike static models, recurrent neural networks (RNNs), can model time-dependent behaviors (e.g., cold starts, load changes). Several studies have demonstrated the effectiveness of ANNs in modeling various aspects of internal combustion engines. For instance, ANNs have been used to:

Predict engine performance and emissions: Researchers have employed ANNs to model the relationship between engine operating parameters (e.g., speed, load, air-fuel ratio) and performance metrics (e.g., torque, power, fuel consumption) and emissions (e.g., NOx, CO, HC) [1] [2].

Estimate in-cylinder pressure: ANNs have been used to predict in-cylinder pressure, a crucial parameter for understanding combustion processes and optimizing engine control [3] [4].

Model engine subsystems: ANNs can model specific engine subsystems, such as the intake manifold, combustion process, and exhaust system, to improve the accuracy of overall engine models [5].

The ability of ANNs to learn complex non-linear relationships, handle noisy data, and provide fast predictions makes them attractive for various applications in engine control and optimization.

Among and beyond ANNs we can also cite other AI, or machine learning, based methods able to capture these dynamics with equal or greater accuracy, still maintaining the generalization capacity. They may be classified like:

Neural Networks: Temporal Convolutional Networks (TCNs) and Long Short-Term Memory (LSTM),

Ensemble Learning: Random Forest and XGBoost,

Kernel based: Support Vector Regression (SVR) and Gaussian Models,

Discovery methods: Sparse Identification of Nonlinear Dynamics (SINDy),

to cite a few.

ANN METHODS - A neural network consists of interconnected layers of neurons. The basic types of layers commonly used in ANNs include (Figure 1):

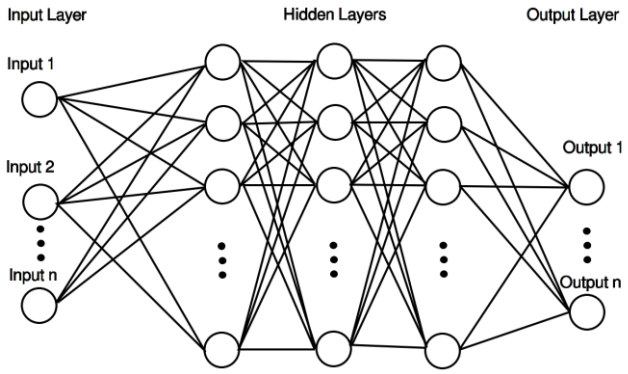

Figure 1. ANN structure.

Input Layer: The first layer in the network. It receives the input data. The number of neurons in this layer corresponds to the number of input features.

Hidden Layers: These layers are located between the input and output layers. A neural network can have multiple hidden layers, allowing it to learn increasingly complex representations of the input data. The neurons in hidden layers extract features from the input and pass them on to the next layer.

Output Layer: The last layer in the network. It produces the final output of the model.

In a typical neural network, each neuron in a layer performs the following operations:

Weighted Sum: It calculates a weighted sum of the inputs from the previous layer.

Bias Addition: A bias term is added to the weighted sum.

Activation: The result is passed through an activation function.

The output of the activation function is then passed on to the neurons in the next layer. This process is repeated for each layer in the network, allowing the network to learn complex non-linear mappings between the input and output.

The activation function is a key factor and will be chosen depending on the problem to be solved.

ENSEMBLE-METHODS - Ensemble methods combine the predictions of multiple individual "weak learners" (often simple models like decision trees) to create a single, more robust, and accurate "strong learner." The core idea is that a diverse group of models, even if individually flawed, can collectively outperform any single model.

KERNEL-BASED-METHODS - Kernel-based methods transform data implicitly into a higher-dimensional feature space where non-linear relationships in the original space become linearly separable. This allows for the application of linear algorithms to solve complex non-linear problems without explicitly calculating the coordinates in the high-dimensional space.

DISCOVERY-METHODS - These methods aim to uncover fundamental, interpretable mathematical relationships (e.g., differential equations, algebraic equations) that govern a system's behavior directly from observed data, rather than just building a predictive black-box model [6].

EXECUTED AND COMPARED METHODS

With the aim of using these non-linear dynamical models in loops for estimation and optimization of control parameters, it is interesting to have a comparison in effectiveness of machine learning methods of modeling. Following, a set of methods are presented with its main characteristics.

TEMPORAL-CONVOLUTIONAL-NETWORKS (TCNs) - Temporal Convolutional Networks (TCNs) are a class of neural networks specifically designed for sequence modeling tasks, including time series prediction and modeling of dynamical systems. Unlike Recurrent Neural Networks (RNNs) and their variants (like LSTMs and GRUs), which process sequences sequentially, TCNs leverage convolutional layers to capture temporal dependencies [7].

TCNs can effectively learn local dependencies through causal convolutions while also capturing long-term relationships through dilated convolutions, making them well-suited for modeling the intricate dynamics of complex systems.

XGBOOST (Extreme Gradient Boosting) - XGBoost is an optimized distributed gradient boosting library designed for efficiency, flexibility, and portability. It's an ensemble learning method that sequentially builds a strong predictive model from a combination of many weak prediction models, typically decision trees.

At its core, XGBoost uses the principle of gradient boosting. It starts with an initial prediction (often the mean of the target variable for regression). Subsequent models (weak learners, usually decision trees) are trained to predict the residuals (the errors) of the previous models. Each new tree attempts to correct the mistakes made by the preceding trees. It excels at capturing non-linear relationships and interactions between features.

RANDOM-FOREST - Random Forest is another ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the mean prediction (for regression) of the individual trees. It's

known for its robustness and ability to handle high-dimensional data [8] [9].

SUPPORT-VECTOR-REGRESSION (SVR) - Support Vector Regression (SVR) is an extension of Support Vector Machines (SVMs) applied to regression problems. Instead of classifying data points into discrete categories, SVR aims to find a function that best approximates the relationship between input features and a continuous target variable [10].

CONTROL LOOP DESCRIPTION: IDLE SPEED CONTROL WITH PI CONTROLLER

The idle speed control, critical for engine stability and driver comfort when stationary or in idle driving condition, is challenged by disturbances from several energy consumers. These include electrical loads imposed by the alternator (e.g., lights, control modules, and refrigeration system motors) and mechanical loads from components like air conditioner compressors, hydraulic pumps, and driver induced clutch torques.

The control system regulates the engine speed to a desired setpoint by manipulating the amount of air and ignition advance, working with fast and slow torques in different zones of speed error.

A Proportional-Integral (PI) controller is commonly used for this purpose due to its simplicity and effectiveness.

The idle speed control loop typically consists of the engine speed output fed back to the controller, which calculates the proportional and integral actions based on the error of regulation (Figure 2).

The actuators in the control loop are the throttle valve for slow and high torque and the ignition advance for high speed and relatively low torque demands.

Each term of the PI controller is calculated in two steps:

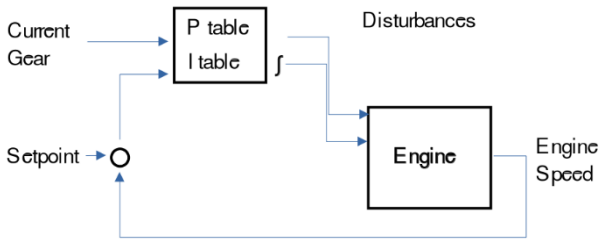

Figure 2. Studied loop control.

P and I factors computation: each factor is calculated throughout a lookup table that has as inputs the regulation error and the engaged gear.

P and I correction signal: the factor for P and I are multiplied by the error and integrated in the case of the integral correction signal.

The proportional term responds to the current error, while the integral term eliminates steady-state errors. The resulting control signal is then applied to the actuators in function of a third lookup table that selects the amount of torque to be required by fast or slow torque sources, adjusting the engine's air or fuel intake and ignition advance.

The behavior of the idle speed control system can differ significantly depending on the clutch state:

Neutral Clutch: In this case, the engine load is relatively low and mainly consists of internal friction and accessory loads. The control system needs to compensate for disturbances such as changes in accessory loads (e.g., air conditioning) to maintain a stable idle speed.

Gears Selected (Clutch Engaged): When a gear is selected and the clutch is engaged, the engine load increases significantly due to the transmission and vehicle load. The control system must now counteract larger disturbances, such as the increased load and potential for engine stalling. The PI controller parameters need to be tuned more carefully to ensure stability and prevent oscillations in this condition.

MOTIVATION FOR MODELING WITH MACHINE LEARNING

As pointed above, an engine-controlled system has many sources of nonlinearity and a great set of parameters to be tuned. The development and calibration of engine control systems is a complex and time-consuming process. With increasing demands for fuel efficiency, emission reduction, and performance, the complexity of these systems continues to grow. Traditional calibration methods, which rely heavily on manual tuning and experimental testing, are becoming increasingly challenging and expensive.

In the fast-paced environment of engine development and calibration, there is a strong need for tools that can:

Reduce calibration time: By automating the optimization process, the time required to calibrate engine control parameters can be significantly reduced.

Improve calibration quality: Automated tools can explore a wider range of parameter combinations and find optimal solutions that may not be achievable through manual tuning.

Facilitate rapid prototyping: A virtual environment allows for rapid testing of different control strategies and parameter settings without the need for physical prototypes.

Enable early-stage optimization: Control system optimization can be performed early in the development process, reducing the need for costly late-stage changes.

Therefore, the development of a tool that can efficiently and accurately optimize engine control parameters, such as PI controller tables, is highly desirable.

By using a non-linear engine model based on AI methods and a search method, this tool can enable faster and more effective control system calibration, ultimately leading to improved engine performance, fuel efficiency, and emissions. In this work, we fulfill the first point of obtaining a model.

MODELING OF THE EXAMPLE SYSTEM

The modeling software was developed in Python due to its large support to artificial intelligence tools and data processing [11] [12]. As mentioned before, the techniques used here are of the black-box type, the training data is fed to the software along with the basic model structure resulting in a non-linear dynamical model.

The modeling was made using LSTM, TCN, XGBoost, Random Forest and SVR identification implementations. As a way of comparing the precision of estimation, the indices MSE (Mean Squared Error) and Dynamic Time Warping (DTW) were implemented. The number of regressors for inputs was fixed in 3 temporal steps, as experimentation showed as a reasonable value to all techniques.

Following we present how the sample data is prepared along with the results obtained.

DATA ACQUISITION AND PREPARATION

The dataset comprises of measurements from an internal combustion engine operating under idle and idle drive conditions. The data captures the engine's response to disturbances caused by both electrical and mechanical loads. This is valuable for developing a model that accurately reflects real-world operating conditions.

Idle Condition: This data represents the engine operating with minimal load, primarily experiencing internal friction and the load of essential accessories.

Idle Drive Condition: This data reflects a more realistic scenario where the engine is engaged with the vehicle's transmission, resulting in a higher load and different dynamic behavior.

By including data from both idle and idle drive conditions with these disturbances, the dataset provides a comprehensive representation of the engine's behavior under various operating scenarios. This is crucial for training a robust neural network model.

DATA SEPARATION FOR TRAINING AND VALIDATION - To properly train and evaluate a neural network model, the data must be split into distinct sets:

Training Set: This subset of the data is used to train the neural network.

Validation Set: This subset is used to evaluate the model's performance during training. It provides an unbiased measure of the model's ability to generalize to unseen data. This helps to prevent overfitting, where the model performs well on the training data but poorly on new data.

DATA PREPARATION FOR DYNAMICAL SYSTEMS IDENTIFICATION - To capture the temporal dependencies in the engine data, it's necessary to create temporal redundancy in the input data. This involves including past values of the input as inputs to the machine learning method. The present and past values of loads (disturbances), and control signals (proportional and integral actions) were selected as inputs while the engine speed is the only output.

The data was arranged into sequences or time windows. Each sequence contains a set of past and current input values, and the corresponding future output value.

Normalization/Scaling is often beneficial to be executed before training the model. This can improve training stability and performance. The scaling and normalization steps are included in each procedure adopted from now on.

MODEL IDENTIFICATION

For the model identification, 80% of the data was used for training, while the other 20% was applied to validation. The original training datasets are showed in the Figure 3 and 4. For better testing of the methods capabilities, two data sets were trained.

The first dataset (Figure 3) is a small one with a great transient happening in a small part of the training data and the validation data covering a smaller area of the working range. It was acquired in the condition of engine in idle with neutral gear and the application of air conditioning and electrical loads intermittently.

The second dataset (Figure 4) is larger and covers a great range of the working area, it was acquired with gears changing from neutral to the fourth gear in idle driving mode. The disturbances applied are also of air conditioning and electrical ones.

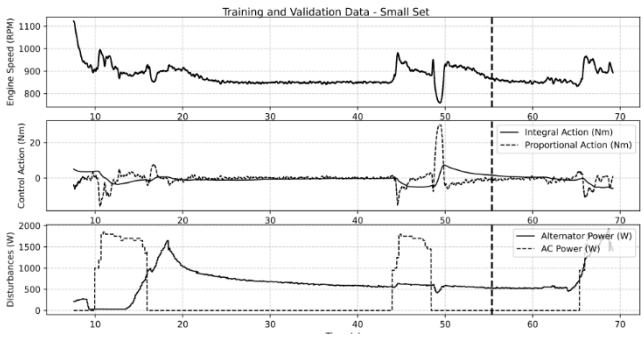

Figure 3. Training and Validation data – Small Dataset

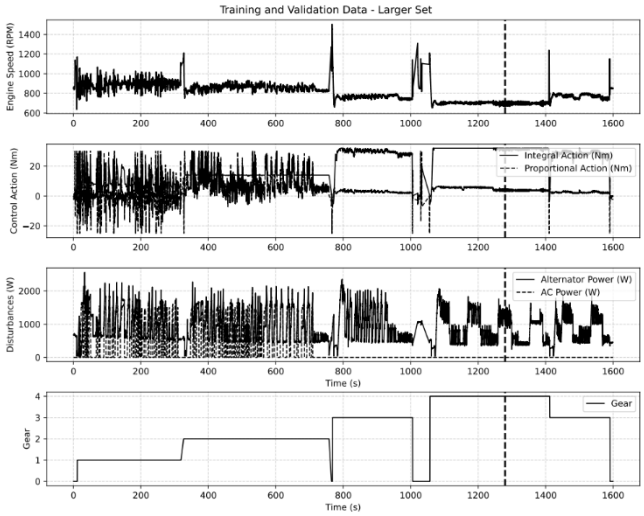

Figure 4. Training and Validation data – Larger Dataset

LSTM – In the following are presented the configurations and results for the Long Short Time Memory method.

The inputs were scaled with mean 0 and standard deviation 1 and the model structure is comprised of (Table 1):

Table 1. LSTM model structure.

Layer (type)	Output Shape	Num Param
LSTM	(None, 3, 64)	1920
Dropout	(None, 3, 64)	0
Dense	(None, 2)	1060

The results for simulation of the validation set are presented in Figure 5 for the smaller dataset and in Figure 6 for the larger dataset. The structure of the model wasn't changed to allow comparison of the results when more information is used during training.

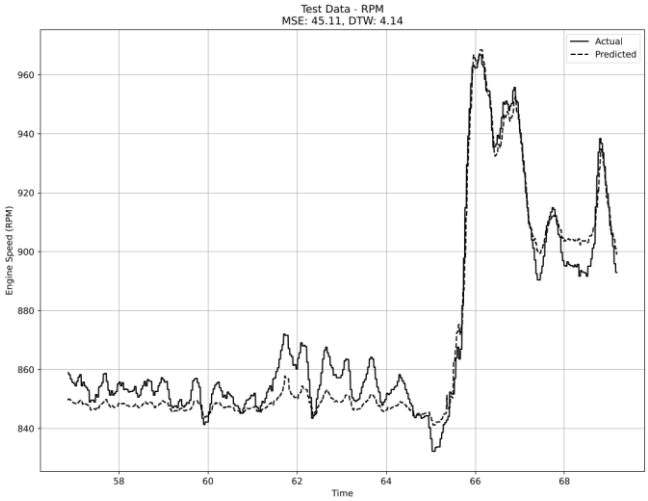

Figure 5. Small dataset for LSTM method.

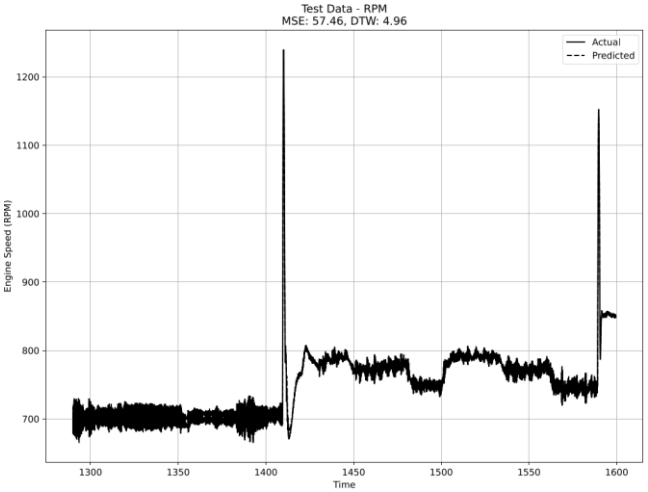

Figure 6. Large dataset for LSTM method.

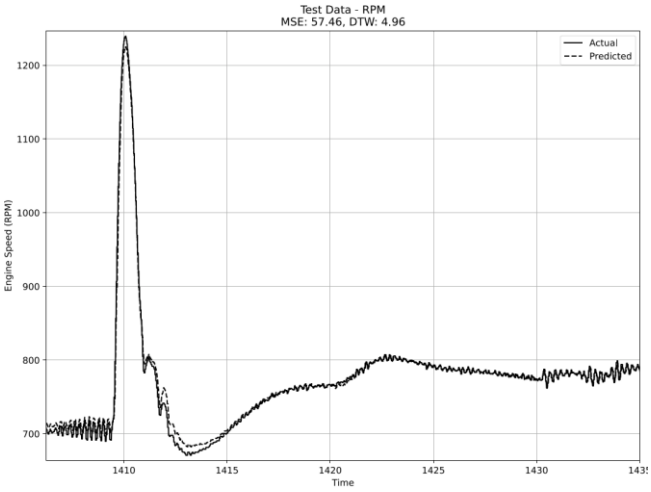

Figure 7. Large dataset for LSTM - Detail.

Comparing the visual and MSE responses it is clear that the model trained with more data seems to make better predictions than the one with less data, better covering the regions of high speed (detail in Figure 7).

TCN – The procedure adopted to execute the simulations for LSTM was replicated here. The final model structure is:

Table 2. TCN model structure.

Layer (type)	Output Shape	Number Parameters	Connected to
Input Layer	(None, 3, 10)	0	
Convolution 1	(None, 3, 64)	1984	Input Layer
Convolution 2	(None, 3, 64)	12352	Convolution 1
Convolution 3	(None, 3, 64)	704	Input Layer
Add/Flatten	(None, 3, 64)	0	Convolution 1 Convolution 2
Dense	(None, 32)	6176	Add/Flatten

Similarly to LSTM case, the model structure was kept the same for the tests with the large and small datasets, resulting in the same behavior as noted in the previous case (improvement for larger dataset), but it is worth to note that, by MSE and DTW indices, the predictions were greatly improved in the case of larger dataset, if compared to LSTM.

The predictions for the small dataset are presented in Figure 8, while for the large dataset are on Figures 10 and 11 (detail).

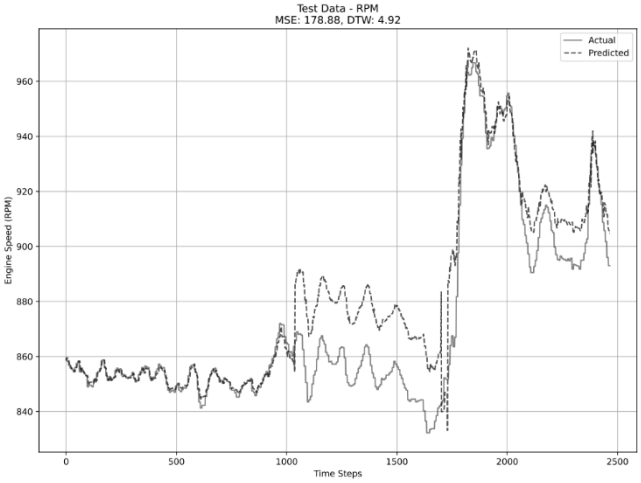

Figure 8. Small dataset for TCN method.

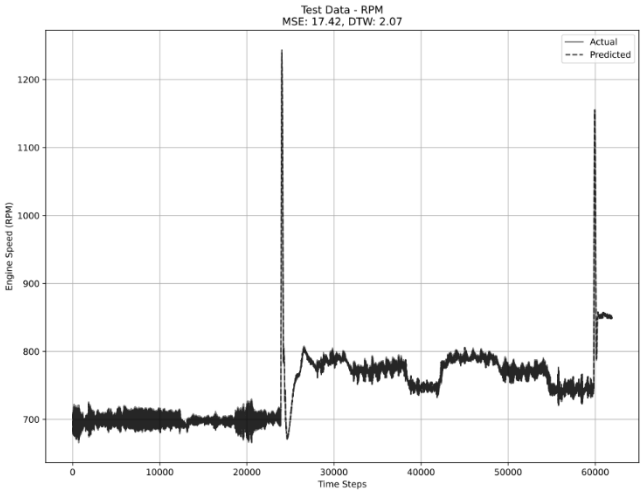

Figure 9. Larger dataset for TCN.

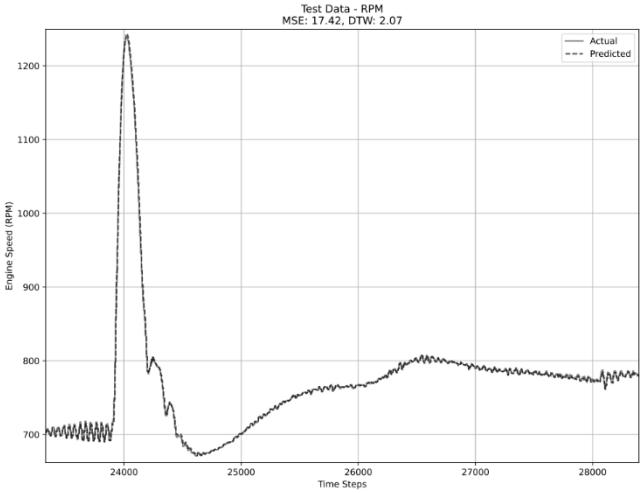

Figure 10. Larger dataset for TCN – Detail.

XGBOOST – This method based in decision trees was configured to run with 500 trees. It was noted that the training time spent was considerably lower than for the ANNs methods. In Figures 11, 12 and 13 (detail) the visual results are presented.

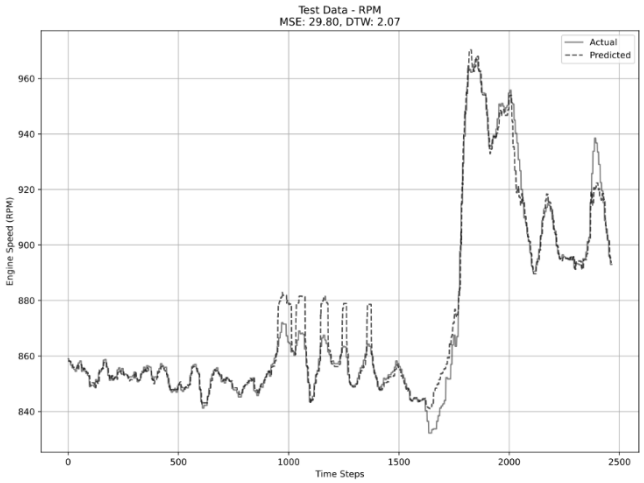

Figure 11. Small Dataset for XGBoost.

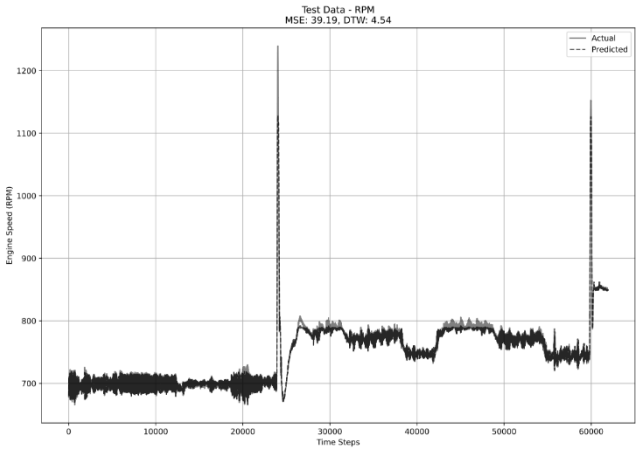

Figure 12. Larger Dataset for XGBoost.

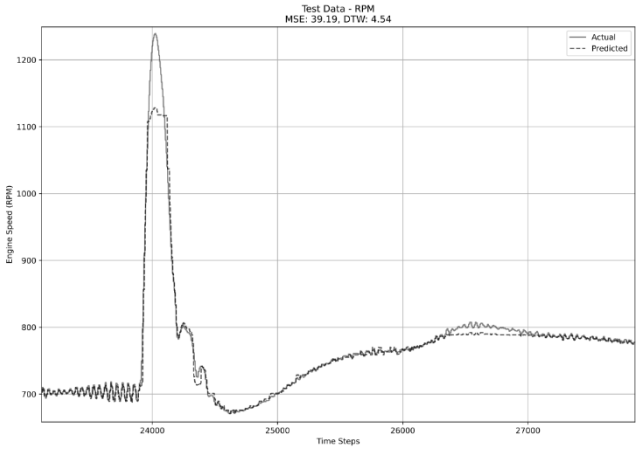

Figure 13. Larger Dataset for XGBoost – Detail.

As noted by the analysis of figures and indices, although the XGBoost predictions seem to fail in great and fast gradients, in the regions where there is more modeling data, it seems to work well, resulting in a reduced fit if compared to the TCN method. But it is important to note that it seems to work better with small datasets.

RANDOM-TREES – Being two ensemble methods, XGBoost and Random Trees work with tree building but have important differences regarding the way how are trained and executed. Here the model used 200 trees to both dataset cases.

The results seen in Figures 14, 15 and 16 show improvements if compared to the XGBoost in the case of large dataset and similar results in the case of small dataset.

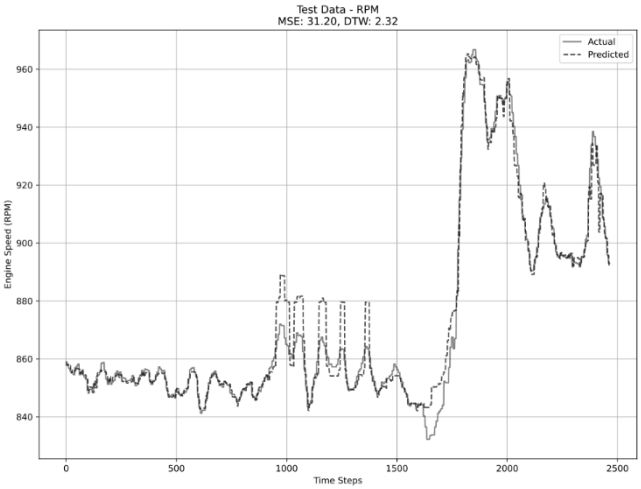

Figure 14. Small Dataset for Random Forest.

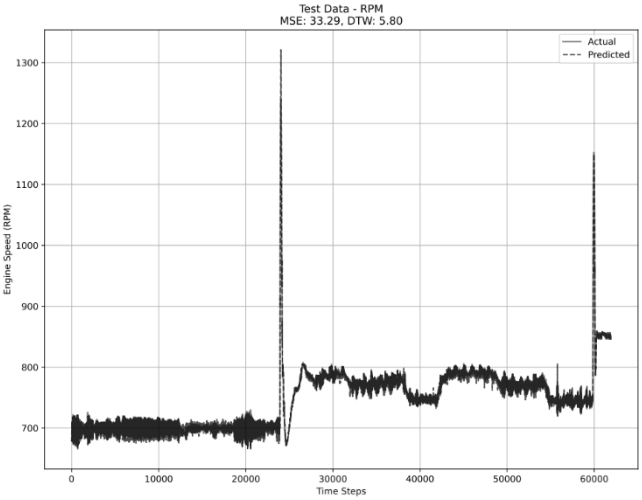

Figure 15. Large Dataset for Random Forest.

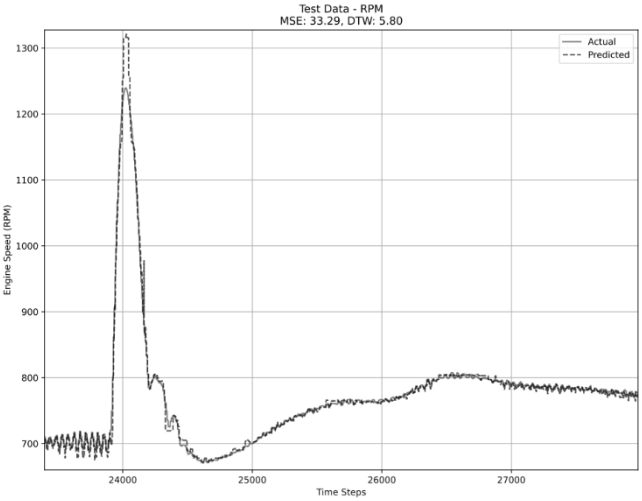

Figure 16. Large Dataset for Random Forest – Detail.

SVR – Being based on a classification method, SVR relies on a set of hyperplanes and a kernel trick to work in higher dimensions, it uses support vectors to define the boundaries of the predicted system curves. The predicted values are inside what is called a ε-tube. This method has great sensitivity to the chosen kernel and has poor computational efficiency for this kind of problem.

These points may be observed in the estimation for the small dataset (Figure 17). The fit is not good and more experimentation with kernels would lead to best results, what is to be done in future works.

With relation to the computational effort, the modeling wasn't possible in acceptable time for the large dataset.

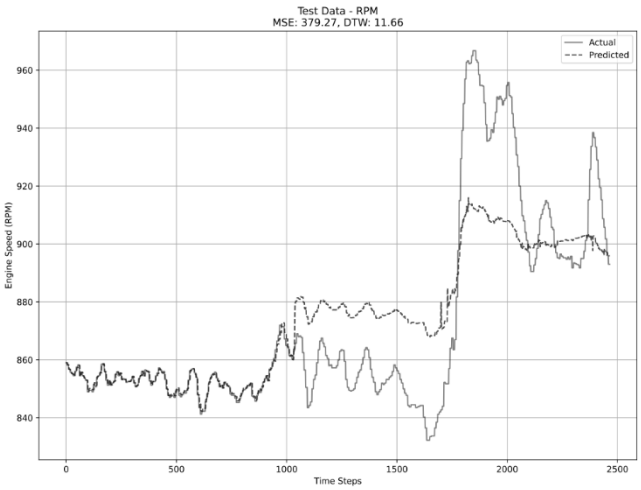

Figure 17. Small Dataset for SVR.

COMPARATIVE-TABLES – In Tables 3 and 4 comparatives of the indices for the cases of small and large datasets are presented.

Table 3. Small dataset indices.

Method/Indice	MSE	DTW
LSTM	45.11	4.14
TCN	178.88	4.92
XGBoost	29.80	2.07
Random Forest	31.20	2.32
SVR	379.27	11.66

Table 4. Large dataset indices.

Method/Indice	MSE	DTW
LSTM	57.46	4.96
TCN	17.42	2.07
XGBoost	39.19	4.54
Random Forest	33.29	5.80
SVR	N/A	N/A

CONCLUSIONS

This comparative analysis of LSTM, TCN, XGBoost, Random Forest, and SVR for non-linear dynamical system modeling across varying dataset sizes reveals some nuances, where no single method unequivocally dominates.

For small datasets, the methods Random Forest and XGBoost demonstrated better performance, as indicated by MSE and DTW. Their ensemble nature provides robustness against overfitting, a common challenge with limited data.

In case of SVR, an extended study on its configurations (kernel selection) have to be done to verify its competitiveness. Its high computational cost is a limiting factor for the application in a fast-paced development, even more when considering that the other methods work very well without need of great adaptation to the problem studied.

The need of larger datasets in the case of LSTM and TCN may be critical depending on the problem studied due to recurrent high costs in tests and even lack of time to spend in experimentation. This may be an important factor in the selection of a method for a generic tool for optimization of closed loop systems, like the one used in this work.

XGBoost and Random Forest remain formidable tools for large datasets, what was noticed by the simulated data. In the literature, it is stated that its ability to capture extremely complex, long-range temporal patterns in some dynamical systems may be surpassed by TCNs or LSTMs with sufficient data, what is observed in TCNs simulations with the larger dataset.

In conclusion, although TCN has presented the best performance with the larger dataset, the XGBoost and Random Forest methods proved to be better scalable, with reduced training times and improved generalization if compared with TCNs and LSTMs. The use of SVRs proved impractical for large datasets training, and even more impractical for optimization of maps based on search methods, due to its computational cost. Future work on small datasets and kernel selection may be interesting. The next step for this work is to evaluate simulation times for ANNs

and ensemble methods to be employed on a search method for control maps optimization.

References

[1] B. Yao, "Prediction of kerosene properties at supercritical pressures by Artificial Neural Network," in *2018 Joint Propulsion Conference*, 2018.

[2] S. R. Narayanan, "Local-Transfer Gaussian Process (LTGP) Learning for Multi-Fuel Capable Engines," in *AIAA SCITECH 2025 Forum*, 2025.

[3] M. Zameni, M. Ahmadi and A. Talebi, "Creating a neural network-based model to predict the exhaust gas temperature of the internal combustion engine," *GSC Advanced Research and Reviews,* 2024.

[4] R. Federico, A. Massimiliano and M. Francesco, "Artificial Neural Networks as a Tool for High-Accuracy Prediction of In-Cylinder Pressure and Equivalent Flame Radius in Hydrogen-Fueled Internal Combustion Engines," *Energies,* 2025.

[5] M. Meli, Z. Wang, P. Bailly and S. Pischinger, "Neural Network Modeling of Black Box Controls for Internal Combustion Engine Calibration," *SAE Technical Paper,* 2024.

[6] S. L. Brunton, J. L. Proctor and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," in *Proceedings of the National Academy of Sciences*, 2016.

[7] J. L. Wang, J. H. a. P. Chen, L. Zhu, Y. Wu, Z. Cai, H. Wu and Maoxuan, "Prediction of the transient emission characteristics from diesel engine using temporal convolutional networks," *Engineering Applications of Artificial Intelligence,* 2024.

[8] W. T. Chung, A. A. Mishra, N. Perakis and M. Ihme, "Data-assisted combustion simulations with dynamic submodel assignment using random forests," *Combustion and Flame,* 2021.

[9] L. Breiman, "Random Forests," *Machine Learning,* 2001.

[10] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing - Springer,* 2004.

[11] F. Chollet, "Keras," Github, 2015. [Online]. Available: https://github.com/fchollet/keras.

[12] M. Abadi, P. Barham, E. Brevdo and e. al, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/.