Framework for Training and Deployment Machine Learning Methods in Real-Time Simulator: Short-Term Kinetic Energy Forecasting in Power Systems

Riquelme-Dominguez, Jose Miguel; Gonzalez-Longatt, F.; Valles, Jose M.; Rueda, Jose Luis

DOI
10.1109/MELECON56669.2024.10608671

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the 2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON)

Framework for Training and Deployment Machine Learning Methods in Real-Time Simulator: Short-Term Kinetic Energy Forecasting in Power Systems

Jose Miguel Riquelme-Dominguez
Electrical Engineering Department
Universidad de Sevilla
Seville, Spain
jmriquelme@us.es

Francisco Gonzalez-Longatt
Centre for Renewable Energy Systems Technology (CREST))
Loughborough University
Loughborough, United Kingdom
fglongatt@fglongatt.org

Jose M. Valles
Instituto de Ingeniería
Universidad Nacional Autónoma de México
Mexico City, Mexico
JVallesC@iingen.unam.mx

Jose Luis Rueda
Department of Electrical Sustainable Energy
Technische Universiteit Delft Delft, Netherlands
Delft, Netherlands
J.L.RuedaTorres@tudelft.nl

2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON) | 979-8-3503-8702-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/MELECON56669.2024.10608671

***Abstract*—Low-inertia power systems require more innovative operation, control, and protection strategies to maintain the operation secure and reliable. One of the challenges related to these systems is the need for knowledge of the inertia level (kinetic energy stored in the rotating masses) in real time. This paper proposes a framework for training and deploying machine learning methods for real-time power systems' kinetic energy forecasting. Linear Regression and Long Short-Term Memory methods are implemented in the Python interpreter of the Typhoon HIL 404 real-time simulator for forecasting the kinetic energy of the Nordic Power System in real-time. This paper provides implementation details together with possible future expansions of the framework. Simulation results show that the trained models can predict the kinetic energy in a forecasting horizon of four hours with a Mean Absolute Error lower than other methods currently available in the literature.**

Keywords— Forecasting, Inertia, Kinetic Energy, Machine Learning, Real-Time

I. Introduction

Historically, power systems have depended primarily on massive, synchronous generators powered by steam or gas turbines to produce electricity. These generators typically have large rotational masses in the rotor, resulting in considerable amounts of physical rotational inertia [1]. The rotational inertia of these machines works as a stabilising force on the grid, helping to maintain the balance between power supply and demand [2], [3].

One of the key benefits of having enough rotational inertia in a power system is that it helps to reduce frequency variations by compensating the active power unbalances. The frequency in a stable system remains relatively constant, usually around the nominal frequency (50 or 60 Hertz) [4], [5]. The frequency, on the other hand, might differ from its nominal value when the balance between generation and load is interrupted due to unexpected changes in demand or generation capacity [6], [7]. This frequency fluctuation can seriously affect the power system's stability and reliability [8].

Rotational inertia (H) is proportional to the kinetic energy (KE) stored in the rotating masses of the machines directly connected to a power system. As the penetration of renewable inverter-based resources (IBR) increases in power systems, physical rotational inertia, and kinetic energy decline. In fact, FINGRID (Finish transmission system operator- TSO) registers the kinetic energy of the synchronous machines connected to the Nordic Power System (NPS) and makes it available to the general public online in [9]. Figure 1 shows the yearly kinetic energy distribution in the NPS in the last six years. The trend shows a higher concentration of values under 200 GW·s in the most recent years and the lowest minimum and maximum values of the kinetic energy registered [10].

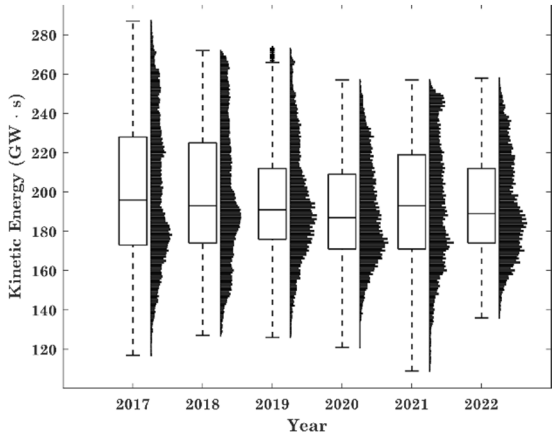

Fig. 1. Yearly distribution of the kinetic energy in the NPS in the last six years (2017-2022) [10].

Knowing the kinetic energy evolution of power systems in advance would greatly help the TSOs, as it would allow preventive actions to be taken before the system performance enters an unsafe/dangerous operating zone, allowing preventive control actions to minimise the operational risk. In this sense, data-driven and specifically Machine learning (ML) has emerged as a robust technique for tackling complicated issues in various domains, including the power systems industry [11]. ML approaches have been widely used to address problems in power system operation, control, planning, and monitoring. Among the techniques are supervised learning, unsupervised learning, reinforcement learning, deep learning, and applications in load forecasting,

fault detection, demand response, renewable energy integration, and grid stability enhancement.

Although there is a consensus in the scientific community about the timing requirements of real-time machine-learning applications [11], not many works have focused on Real-Time Simulators (RTS) [12], which are indispensable tools for power system analysis. This paper proposes a framework for training and deploying machine learning methods for real-time power systems' kinetic energy forecasting using Typhoon HIL RTS. All files for replicating the results of this work are publicly available at [13].

The rest of the paper is organised as follows: Section II presents the proposed methodology for real-time kinetic energy forecasting using Typhoon HIL real-time simulator. Section III shows the simulation results, and finally, Section IV details the conclusions of the work and possible expansions of the framework.

II. Proposed Methodology

The proposed real-time modelling and simulation framework requires the use of appropriate software and hardware. Although the methodology presented in this paper is illustrated using the Typhoon HIL modelling and simulation framework, it can be extended to another manufacturer with only minor changes (not explained in this paper because of space limitations).

The general methodology consists of three main steps:

- *Step 1*. Raw data collection and preparation.
- *Step 2*. Step up the real-time simulation framework to use external Python libraries/packages.
- *Step 3*. Step up and use of the proposed forecasting methodology in the real-time simulation framework.

A. Step 1. Raw data collection and preparation

The starting point of the methodology is collecting and preparing the raw data; this step is performed offline but is essential to ensure the successful development of the other steps. The dataset consists of a time series of the variables under consideration. In this paper, the main application is the forecasting of the kinetic inertia, so the authors took the time series of the kinetic energy available in the Nordic Power System (NPS) that is publicly available in [9]. The authors have created a Python code to use an API to systematically collect the dataset from the FINGRID repository. The automatic collection script allows downloading the time series in a commonly used format, comma-separated values (CSV) file, for further processing.

The authors have created customised code in Python to preprocess the time series to remove wrong (out of scale) fill missing or wrong values, so the data integrity is ensured. The processing time of this step is irrelevant as it is running offline on the CPU of the host PC.

B. Step 2. Step up the real-time simulation framework to use external Python libraries/packages

Real-time simulators are equipped with solid modelling and simulation software; it has the possibility of extending the functionalities by using additional computational libraries and/or packages. Depending on the real-time simulator hardware and manufacturer, the use of some programming languages is more suitable. Some real-time manufacturers such as OPAL-RT and Typhoon HIL extend the testing functionalities by automating the testing process by using the popular interpreted programming language Python. In this paper, the main interest is to create a framework to use of machine learning algorithms for time series forecasting, so Python programming language results are very attractive as several libraries extensively used are available, such as TensorFlow, SciPy, Skit learn, etc.

The centre or core of the Typhon HIL real-time modelling and simulation framework is the Typhoon HIL Control Centre. It enables de use of the Schematic Editor to create the models and HIL SCADA to control and run the real-time simulation; in both cases, a Python interpreter was created when developing the software. However, the user cannot install anything to this Python interpreter because it is unalterable from the building stage [14], [15]. There is a solution, however, that allows the user to install additional packages in HIL SCADA and Schematic Editor. The essential concept behind this solution is to include the relevant path from the host PC into this unalterable Python.

Initially, the user installs the desired library using the package-management system called Pip; it is written in Python and is used to install and manage software packages. Any library/package available on the PyPI website (https://pypi.org/) can be installed using Pip in the typhoon-python interpreter. For Microsoft Windows cases, the Command prompt must be used, to do so: Press the Windows Key + R, type in cmd.exe, and press Enter. Open search and type in cmd. To install the free and open-source software library for machine learning and artificial intelligence Tensorflow, the user must type in the Command Prompt and then press Enter:

```
1: typhoon-python -m pip install tensorflow
```

Fig. 2 shows a screen capture showing the use of the command-line interpreter to install the very popular Python library TensorFlow.

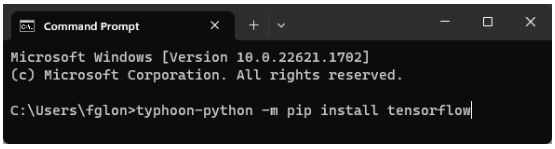

Fig. 2. Illustrative example of Microsoft Windows Command Prompt window showing the command line for installing library TensorFlow in the Typhoon HIL Python interpreter.

As soon as the Python library is installed in the appropriate location and using the process described above, the library can be imported and used in the typhoon-python interpreter. However, the user must be careful to add the appropriate path where the library was installed.

The installed Python libraries can be used in several places inside the Typhoon HIL framework; one of the places is inside the graphical environment used to create a specific interface with the real-time model, Typhoon HIL SCADA. The Panel Initialization inside the SCADA allows to specify global constants, variables, and functions; it includes the use of the recently installed Python Libraries. Code 1 (below) shows the Python code lines needed to make the code of the library available in the module that the user is developing inside the HIL SCADA. Again, the path stored in the variable `sendto_dir` is essential to make possible use of the library. The installed library will be used for the proposed forecasting methodology in the real-time simulation framework in *Step 3*.

Code 1: Importing external Python packages in Typhoon HIL SCADA environment

```
import os
from os import path
import sys
sendto_dir =
path.expandvars(r'C:\Users\user_name\AppData\Roaming\typhoon\20XX.X\python_portables\python3_portable\Lib\site-packages')
add_to_python_path(os.path.normpath(sendto_dir))
import tensorflow as tf
```

C. *Step 3: Step up and use the proposed forecasting methodology in the real-time simulation framework*

The proposed time-series forecasting for real-time hardware is configured to work in three sub-steps, shown below and explained in the following subsections.

- Step 3.1. Initialisation.
- Step 3.2. Real-time forecasting
- Step 3.3. Publishing results

1) Step 3.1. Initialisation

The initialisation process is performed once at the beginning of the real-time simulation. Referring to the Typhoon HIL framework, the Typhoon HIL SCADA is used to initialise the proposed time-series forecasting framework. During this step, the code embedded in the initialisation panel allows importing the CSV files containing the KE raw data. Next, the data is preprocessed according to Code 2, in which the gaps in the raw data are filled using the backfill technique and then scaled to the interval [-1.0,1.0], a common practice in machine learning data preprocessing. Finally, *dropna* function from the Pandas library removes missing values in the time series.

Code 2: HIL SCADA panel initialisation dialog

```
# I: Import Libraries
 import tensorflow as tf
 import numpy as np
 import pandas as pd
 import datetime as dt
 from sklearn.linear_model import LinearRegression
 from sklearn.preprocessing import MinMaxScaler
# II: Define a dictionary
 return_dict = {"predicted":ypred,
                "realvalue":ytest,
                "elapsed":elapsed,
                "timecount":time_count}
# II: Import the time-series from the CSV files
 files = os.listdir(data_path)
df = pd.concat([pd.read_csv(f"{data_path}/{filename}") for filename in files])
# III: Preprocess the data
df = df.asfreq("1min", method="backfill")
SCALER = MinMaxScaler((-1,1)).fit([[130.0], [260.0]])
df[['Value']] = SCALER.transform(df[['Value']])
df = df.drop("Value", axis=1)
df = df.dropna()
# III. Training process
# A. Linear Regression model
model = LinearRegression()
model.fit(X_train,y_train)
# B. LSTM model
X_train = np.expand_dims(X_train, axis=2)
X_test = np.expand_dims(X_test, axis=2)
model = tf.keras.Sequential([tf.keras.layers.LSTM(64,…)])
model.fit(X_train, y_train,validation_data=(X_test,y_test), batch_size=512, epochs=100)
```

The panel initialisation dialogue can be used either to load or train the models. In Code 2, it is shown how to train Linear Regression (LR) and Long Short-Term Memory (LSTM) models by using *model.fit* function. At this point, the data is loaded and preprocessed, and the models are ready to start the real-time forecasting process.

2) Step 3.2. Real-Time forecasting

The real-time forecasting process takes place in HIL SCADA macro blocks. Depending on the configuration, these blocks execute at the beginning of the simulation, on user clicks, on a pre-defined timer and on the simulation stop. To mimic real-time operation and control of a power system, the forecasting is performed on timer event, executed each 1000 ms. The Python code executed periodically is shown in Code 3.

Code 3: HIL SCADA macro

```
# I: Load the dictionary
 global return_dict
# Extract the correspongding rows of X_test and y_test matrixes
 X_test_row=X_test[time_count,:]
 y_test_row=y_test[time_count,:]
# Perform the forecasting
 y_pred_row = model.predict(X_test_row.reshape(1, -1))
# De-normalize predictions
 y_pred_row = SCALER.inverse_transform(y_pred_row.reshape(1, -1))
 y_test_row = SCALER.inverse_transform(y_test_row.reshape(1, -1))
# Store the results in a matrix
 y_test_matrix[time_count,:] = y_test_row
# Calculate Mean Absolute Error
 mae = np.mean(np.abs(y_pred_row-y_test_row))
# Update the fields of the dictionary
 return_dict={"prediccion":y_pred_row[0,0],
              "valorreal":y_test_row[0,0],
              "elapsed":elapsed,
              "timecount":time_count}
# Update the time counter
time_count = time_count + 1
```

3) Step 3.3. Publishing results

The final step of the proposed methodology is to publish the results of the forecasted time series. The Typhoon HIL framework uses the graphical environment available in the HIL SCADA to create a specific interface with the real-time model. Several widgets available in HIL SCADA, such as Text Displays and Trace Graphs widgets, achieve real-time monitoring. The former block helps show some information about the real-time simulation, for example, the model used to make predictions, the past history considered or the forecasting horizon length, and the current date. The latter block represents in real-time the evolution of the selected signals. Code 4 shows the code expression to depict simultaneously the actual value and the predicted one in the last few predictions.

Code 4: Expression code of the Trace Graph widget

```
# I: Load the dictionary
 global return_dict
# II: Extract the correspongding fields
 ypred=return_dict.get("predicted")
 ytest=return_dict.get("realvalue")
# III: Define the data to be displayed
 data = {
          "analog_names": ["y_pred","y_test"],
          "analog_values": [float(ypred),float(ytest)],
          }
 displayValue = data
```

III. Simulations and Results

The proposed framework was tested with the KE data of the NPS in 2020 with a resolution of one sample per minute. The data is preprocessed as detailed in Section II. The data from the first nine months of the year are used to train both LR and LSTM, considering different sizes of the past history (number of samples considered to make a new prediction) and the forecasting horizon (number of future values to be

predicted). The trained models are tested with the data of October, November, and December 2020.

For the implementation and testing of the proposed methodology, a Typhoon HIL 404 real-time simulator with Typhoon HIL Control Center V2022.4 sp1 software is used. Python programming language version 3.9.6 was installed on the host PC. The host PC has Intel® Core™ i7-7700 CPU-based processor running Microsoft Windows 10 Pro. Table I shows the results of the experiments in terms of the Mean Absolute Error (MAE) and the mean execution time in milliseconds for different combinations of the model, past history and forecasting horizon.

TABLE I. SIMULATIONS RESULTS AND PERFORMANCE INDICATORS

Model	*Past history*	*Forecasting horizon*	*MAE (GW·s)*	*Mean execution time (ms)*
LR	90	15	0.49	2.69
LR	180	30	0.72	2.57
LR	360	60	1.16	2.73
LR	1440	240	2.71	3.21
LSTM	90	15	0.46	56.8
LSTM	180	30	0.71	63.8
LSTM	360	60	1.09	83.5
LSTM	1440	240	2.64	126.6

The results of Table I can draw different conclusions: first, the forecasting accuracy decreases when the forecasting horizon increases for the two ML approaches. In all the combinations of past history and forecasting horizon values, LSTM performs slightly better than LR. However, the forecasting error is lower for both approaches than that presented in [7], [8]. Finally, the mean execution time was computed for the algorithms. As seen, the execution time is almost constant for the LR approach, whereas it increases as well as the past history and the forecasting horizon increase in the case of LSTM is used. Nevertheless, these times are low enough when compared to the execution rate of the forecasting (minute by minute), which means that the algorithms are suitable for real-time operation.

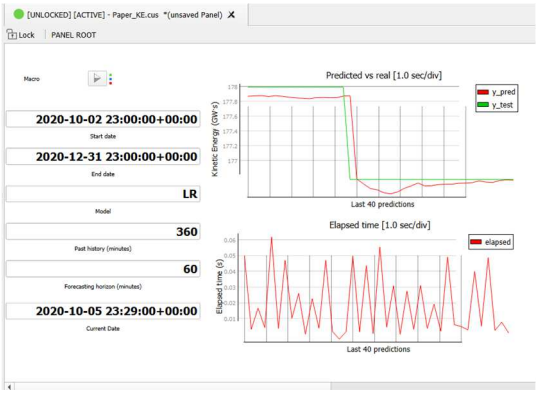

Fig. 3. Screen capture of the HIL SCADA panel during the real-time simulation.

Fig. 3 shows the HIL SCADA panel during the real-time simulation. As depicted, the panel gives online information about the forecasting, such as the ML model in use, the past history and forecasting horizon values or the current date. In addition, visual information is provided employing Trace Graph widgets, which represent the evolution of the predicted KE over the last 40 predictions and the mean execution time elapsed in each of them.

IV. CONCLUSIONS

Integration of high levels of inverter-based resources in power systems requires system operators to develop more advanced tools for power systems planning and control. This work proposes a framework for training and deploying machine learning algorithms for real-time kinetic energy forecasting. The framework is designed to operate in the Python interpreter of the Typhoon HIL Control Center software. Implementation details and the forecasting results of Linear Regression and Long Short-Term Memory algorithms are provided. The real-time simulations showed that the algorithms could run online, and the Mean Absolute Error was lower than other methods available in the literature.

Future developments of the proposed framework will focus on expanding it to train models in real time.

ACKNOWLEDGMENT

This research has been funded by the Andalusian Regional Government under the project PAIDI 2021-PROYEXCEL_00588. The authors would like to recognise the strong support provided by the technical team of Typhoon HIL.

REFERENCES

[1] B. Tan, J. Zhao, M. Netto, V. Krishnan, V. Terzija, and Y. Zhang, "Power system inertia estimation: Review of methods and the impacts of converter-interfaced generations," Int. J. Electr. Power Energy Syst., vol. 134, p. 107362, Jan. 2022, doi: 10.1016/J.IJEPES.2021.107362.

[2] J. M. Valles, C. Angeles-camacho, and F. Gonzalez-longatt, "Modelica Implementation and Validation of Virtual Synchronous Machine Control for a VSC in ePHASORSIM," in IEEE PES General Meeting 2023, Orlando, FL, USA: IEEE, 2023.

[3] R. D. Rodriguez-Soto, E. Barocio, F. Gonzalez-Longatt, F. R. S. Sevilla, and P. Korba, "Robust Three-Stage Dynamic Mode Decomposition for Analysis of Power System Oscillations," IEEE Trans. Power Syst., pp. 1–10, 2023, doi: 10.1109/TPWRS.2023.3275102.

[4] J. Fang, H. Li, Y. Tang, and F. Blaabjerg, "On the Inertia of Future More-Electronics Power Systems," IEEE J. Emerg. Sel. Top. Power Electron., vol. 7, no. 4, pp. 2130–2146, 2019, doi: 10.1109/JESTPE.2018.2877766.

[5] H. R. Chamorro, A.-J. Guel-Cortez, E. Kim, F. Gonzalez-Longatt, Á. Ortega, and W. Martinez, "Information Length Quantification and Forecasting of Power Systems Kinetic Energy," IEEE Trans. Power Syst., vol. 37, no. 6, pp. 4473–4484, 2022, doi: 10.1109/TPWRS.2022.3146314.

[6] J. M. Riquelme-Dominguez, M. N. Acosta, F. Gonzalez-Longatt, M. A. Andrade, E. Vazquez, and J. L. Rueda, "Improved Harmony Search Algorithm to Compute the Underfrequency Load Shedding Parameters," Int. Trans. Electr. Energy Syst., vol. 2022, pp. 1–15, Nov. 2022, doi: 10.1155/2022/5381457.

[7] A. J. Veronica, N. S. Kumar, and F. Gonzalez-Longatt, "Design of Load Frequency Control for a Microgrid Using D-partition Method," Int. J. Emerg. Electr. Power Syst., vol. 21, no. 1, Feb. 2020, doi: 10.1515/ijeeps-2019-0175.

[8] J. M. Riquelme-Dominguez, F. M. Gonzalez-Longatt, and S. Martinez, "Decoupled Photovoltaic Power Ramp-rate Calculation Method for Perturb and Observe Algorithms," J. Mod. Power Syst. Clean Energy, vol. 10, no. 4, pp. 932–940, 2022, doi: 10.35833/MPCE.2021.000603.

[9] "Kinetic energy of the Nordic power system - real time data - Dataset - Fingridin avoin data." https://data.fingrid.fi/en/dataset/kinetic-energy-nordic-realtime (accessed May 22, 2023).

[10] F. Gonzalez-Longatt, M. N. Acosta, H. R. Chamorro, and D. Topic, "Short-term Kinetic Energy Forecast using a Structural Time Series Model: Study Case of Nordic Power System," in International Conference on Smart Systems and Technologies (SST), Osijek, Croatia, 2020.

[11] J. Bian et al., "Machine Learning in Real-Time Internet of Things (IoT) Systems: A Survey," IEEE Internet Things J., vol. 9, no. 11, pp. 8364–8386, Jun. 2022, doi: 10.1109/JIOT.2022.3161050.

[12] J. M. Riquelme-Dominguez et al., "A machine learning-based method-

ology for short-term kinetic energy forecasting with real-time application: Nordic power system case", International Journal of Electrical Power Energy Systems 156 (2024) 109730. doi:https://doi.org/10.1016/j.ijepes.2023.109730.

[13] J. M. Riquelme-Dominguez, M. Carranza-García, P. Lara-Benítez, F. GonzalezLongatt. Kinetic-energy-forecasting. 2023, URL https://github.com/carranza96/ kinetic-energy-forecasting. (Accessed 12 November 2023).

[14] "(PDF) Import external Python packages in Typhoon HIL." https://www.researchgate.net/publication/370341383_Import_external_Python_packages_in_Typhoon_HIL (accessed May 22, 2023).

[15] "Typhoon HIL." https://ticket.typhoon-hil.com/kb/faq.php?id=271 (accessed May 22, 2023).