Neural networks based speed-torque estimators for induction motors and performance metrics

Sagar Verma, Nicolas Henwood, Marc Castella, Al Kassem Jebai, Jean-Christophe Pesquet

Neural Networks based Speed-Torque Estimators for Induction Motors and Performance Metrics

Sagar Verma,[1,2] Nicolas Henwood,[2] Marc Castella,[3] Al Kassem Jebai,[2]
Jean-Christophe Pesquet,[1]
[1] Université Paris-Saclay, CentraleSupélec, Inria, Centre de Vision Numérique
[2] Schneider Toshiba Inverter Europe
[3] SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris

***Abstract*—This paper focuses on the quantitative analysis of deep neural networks used in data-driven modeling of induction motor dynamics. With the availability of a large amount of data generated by industrial sensor networks, it is now possible to train deep neural networks. Recently researchers have started exploring the usage of such networks for physics modeling, online control, monitoring, and fault prediction in induction motor operations. We consider the problem of estimating speed and torque from currents and voltages of an induction motor. Neural networks provide quite good performance for this task when analysed from a machine learning perspective using standard metrics. We show, however, that there are some caveats in using machine learning metrics to analyze a neural network model when applied to induction motor problems. Given the mission-critical nature of induction motor operations, the performance of neural networks has to be validated from an electrical engineering point of view. To this end, we evaluate several traditional neural network architectures and recent state of the art architectures on dynamic and quasi-static benchmarks using electrical engineering metrics.**

***Index Terms*—induction motor, neural networks, deep learning, time series, training**

I. Introduction

Induction motors have very complex dynamics and it is essential to have a controller that can provide robust control based on these dynamics. Induction motor controllers also offer protection and supervision of the electro-mechanical system [1], [2]. For these services, it is mandatory to know the dynamical physical model of induction motors. Accurate dynamics is derived from the first principles of physics. These dynamical models are dependent on different induction motor physical characteristics like currents, voltages, speed, fluxes, inductances, and resistances, which are measured directly or indirectly using sensors or estimators. Accurately measuring some of these quantities is challenging due to the presence of noise and the operating conditions.

Controllers derived from physical models are widely used in industry and are very reliable. Although moderate complexity models of the induction motor exist, they are nonlinear and include several uncertain parameters. Industrial Internet of Things (IIoT) has made it possible to collect a large amount of data from different electro-mechanical devices. This collection of real-time or near real-time data from induction motor systems has enabled the development of online decision algorithms for electrical system monitoring applications. Recent advances in deep neural networks for time series analysis have also led to better prediction models. End-to-end learning of temporal dynamics from time-series data has been made easier due to techniques like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long-Short Term Memory (LSTM) structures.

Neural networks have been actively used for controller [3]–[5], monitoring and fault prediction in electrical motor operations [6]–[9]. Verma et. al. [10] used encoder-decoder architecture to learn dynamics from induction motor signals. These methods are evaluated using machine learning metrics like mean square error, r-square, accuracy, etc. These metrics are well suited for regression and prediction tasks but have an inherent bias towards the dataset. We analyze how and when these metrics fail and propose a way to properly evaluate neural network methods for induction motor problems using dynamic and quasi-static benchmarks, as well as electrical engineering metrics. We consider the problem of predicting speed and electromagnetic torque from currents and voltages represented in $(d-q)$ frame. We experiment on network architectures similar to those proposed in [10].

This paper is structured as follows: Section II provides a short survey on induction motor physics modeling and neural networks. Section III explains the nonlinear physical model of an induction motor and electrical engineering metrics used in our experiments. Section IV details neural network models used in this paper. Section V describes the training and benchmark data. Section VI contains experimental details and the obtained results. In the last section, conclusions are drawn and some possible extensions of this work are mentioned.

II. Related Work

Development in model-based nonlinear control methods in the last three decades can be grouped in the following categories: feedback linearization methods [11], [12], Lyapunov-based control [13], [14], and passivity-based methods [15], [16]. The state-space model of an induction motor is described in [17]. Modeling of electrical motors based on analytical mechanics and energy consumption is presented in [18]. Existing methods for designing a controller for an induction motor can be done in two ways, when perfect knowledge of

the parameters is available [19]–[21] and when there is an uncertainty associated with the estimation of the parameters [22]–[24]. Model-based control methods for induction motors all require proper knowledge about four independent electrical parameters of the motors.

Neural network-based control methods have also been proposed to learn a better model that can overcome problems associated with the traditional model-based approach. [25] uses radial basis neural networks to learn the relationship between currents and flux linkages. [6] presents a neural network classifier for fault diagnosis in electrical motor operations. The approach does not use dynamics modeling and only relies on supervised labels [8], learn motor dynamics from simulated data, and perform fault detection in simulated motor operations. In [10], encoder-decoder network is proposed to learn dynamics of electrical motor directly from recorded data. The proposed method achieves good performance in modeling the input-output relationship.

The reasons why neural networks fail is analyzed in [26]. Neural networks make fewer assumptions, have a large number of parameters, and have different modeling processes, opening more possibilities for inappropriate uses and problematic applications. Another major pitfall consists of treating neural networks as black boxes. In [27] a robustification technique is proposed to interpret neural network results in terms of the input effects and interactions among input variables. Underfitting and overfitting being well-known problems in machine learning methods, neural networks are prone to these problems due to their data-hungry nature and large number of parameters, as discussed in [28].

III. Physical Modeling

A. Nonlinear State-Space Motor Model

In this section, we present the mathematical model of an induction motor we consider, which was introduced in [17]. Rotating reference frame at pulsation ω_s ($d-q$ model) is used to express all the quantities. Park transformation is used to convert the quantities from the fixed three-axis frame to the orthogonal rotating ($d-q$) frame. The fifth-order nonlinear state space model of the induction motor reads as follows:

$$\frac{J}{n_p}\frac{d}{dt}\omega_r = \frac{3}{2}n_p\Im(\psi_s^* i_s) - \tau_L \tag{1}$$

$$\frac{d}{dt}\psi_s = -j\omega_s\psi_s - R_s i_s + u_s \tag{2}$$

$$\frac{d}{dt}\psi_r = -j\omega_s\psi_r - R_r i_r + j\omega_r\psi_r \tag{3}$$

where $\Im(z)$ and z^* are respectively the imaginary part and conjugate of z. n_p denotes the number of pole pairs, J the motor shaft inertia, τ_L the load torque, and $u_s = u_{sd} + ju_{sq}$ the motor input voltage in the ($d-q$) frame. ω_r (n_p times the mechanical speed), $\psi_s = \psi_{sd} + j\psi_{sq}$ (stator flux), and $\psi_r = \psi_{rd} + j\psi_{rq}$ (rotor flux) are the five state variables. The flux variables are linked to the current variables i_s (stator) and i_r (rotor) by nonlinear relationships [29] given by :

$$\begin{aligned}\psi_s &= L_{fs}i_s + \frac{L_m(i_s+i_r)}{1+\gamma|i_s+i_r|}\\ \psi_r &= L_{fr}i_r + \frac{L_m(i_s+i_r)}{1+\gamma|i_s+i_r|}\end{aligned} \tag{4}$$

The parameters are the stator and rotor resistances R_s and R_r, the stator and rotor leakage inductances L_{fs} and L_{fr}, and the magnetic saturation coupling parameters L_m and γ.

B. Performance Metrics

To evaluate neural networks we use electrical engineering (EE) performance metrics widely used in industrial settings. We report the following metrics, applied to the response signal to a speed or torque reference ramp (whose amplitude is the absolute difference between the starting and target values):

- **2% response time** ($t_{2\%}$) is the time value at which the response signal has covered 2% of the ramp amplitude.
- **95% response time** ($t_{95\%}$) is the time value after which the response signal remains at less than 5% of the ramp amplitude from the target value.
- **Overshoot** ($D\%$) is the difference between the maximum peak value of the response signal and the target value. It is expressed in percentage of the ramp amplitude.
- **Steady-state error** (E_{ss}) is the difference between the response signal and target values once the steady-state has been reached.
- **Following error** (E_{fol}) is the difference between the reference and response signal values at the time when the reference value has covered 50% of the ramp amplitude.
- **Maximum acceleration torque (for speed ramp only)** ($\Delta\tau_{max}$) is the maximum response torque deviation during the speed ramp.
- **Speed drop (for torque ramp only)** (SD) is the maximum response speed deviation during the torque ramp.

IV. Data Driven Modeling

A. Standard Neural Networks

We use three benchmark neural network architectures from [10] for our experiments. Since the focus of this paper is on the evaluation of performance metrics, we only use subset of benchmark networks from [10].

*1) Four layer Fully Connected Network (**FCN**):* A four layer fully connected network which has been used in different tasks related to induction motor operations.

*2) Two layer Long-Short Term Memory Network (**LSTM**):* In sequential neural networks, we use LSTM as it performs better than RNN. We use two layers of LSTM, followed by three fully connected layers in this network.

*3) Four layer Convolutional Neural Network (**CNN**):* CNNs have been shown to provide competitive results on sequential data. In line with recent advances in the use of 1D convolutions for sequential tasks, we also employ a four-layer convolutional neural network for our experiments.

B. Encoder-Decoder Networks

Encoder-decoder networks have been proven to perform better when input and output dimensions are the same. We consider encoder-decoder variants proposed in [10]. This structure consists of encoding and decoding blocks with convolutional and deconvolutional layers, respectively.

*1) Vanilla Encoder-Decoder **(Vanilla)**:* Vanilla Encoder-Decoder consists of four layers of convolutional and four layers of a deconvolutional block.

*2) Skip Connections **(Skip)**:* We add skip connections to each convolutional layer of the encoding block to consecutive deconvolutional layer of the decoder block.

*3) Recurrent Skip Connections **(RNN)**:* Better temporal dynamics learning is achieved by introducing RNN layers in skip connections.

*4) Bidirectional Recurrent Skip Connections **(BiRNN)**:* Bidirectional RNN in skip connections have a better scope over complete input sequence than unidirectional RNN.

*5) Bidirectional Diagonalized Recurrent Skip Connections **(BiDiagRNN)**:* RNNs in the network are diagonalized by making their parameters independent of each other. This leads to fewer parameters.

C. Loss Function and Evaluation Metrics

Very often, mean square error loss is used to train a regression model. Total variation weighted mean square loss $\mathcal{L}_{\text{TV-MSE}}$ as proposed in [10] gives more weight to dynamic parts of the signal:

$$\mathcal{L}_{\text{TV-MSE}} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\left(\sum_{t=1}^{T-1} |y_t^i - y_{t+1}^i|\right)}_{\text{Total Variation}} \overbrace{\left(\frac{1}{T}\sum_{t=1}^{T}(y_t^i - \hat{y}_t^i)^2\right)}^{\text{MSE}} \tag{5}$$

where y_t^i and $\hat{y}_t^i$ are the values of output and predicted sample i at time-step t, respectively. N is the number of training samples where each sample is of duration T.

To analyse model performance at global scope, we report machine learning (ML) metrics. Mean absolute error (MAE), symmetric mean absolute percentage error (SMAPE), and coefficient of determination (R^2) are used, whose expressions are recalled below:

$$\text{MAE}(y, \hat{y}) = \frac{1}{T}\sum_{t=1}^{T} |y_t - \hat{y}_t| \tag{6}$$

$$\text{SMAPE}(y, \hat{y}) = \frac{100}{T}\sum_{t=1}^{T} \frac{|\hat{y}_t - y_t|}{|\hat{y}_t| + |y_t|} \tag{7}$$

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{t=1}^{T}(\hat{y}_t - \bar{y})^2}{\sum_{t=1}^{T}(y_t - \bar{y})^2} \tag{8}$$

where y_t is the ground truth, $\hat{y}_t$ is the predicted output of the model at time t, and T is the total experiment duration. $\bar{y}$ denotes the mean of ground truth y.

V. Datasets

We use the motor model proposed in [17] to generate our simulated data.

A. Reference Trajectory Generator

To generate simulated training and validation sets, we created a trajectory generator that generates realistic reference speed and load torque trajectories. For every simulation, the number of static states is drawn randomly from a uniform distribution between 5 and 15. The duration of a static state in a simulation is drawn from a uniform distribution between 1 and 5 seconds. Ramp duration between two consecutive static states is generated according to a shifted truncated exponential distribution between 4 and 2000 milliseconds to provide more frequent short duration ramps. For each static state, speed and load torque values are generated according to a uniform distribution on [-70, 70] Hz and [-120, 120] % of nominal torque ($\%\tau_{nom}$), respectively. Figure 1 shows a sample reference trajectory from the training set.

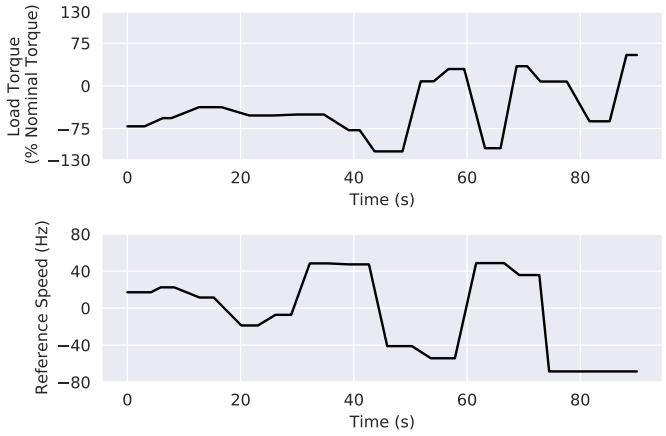
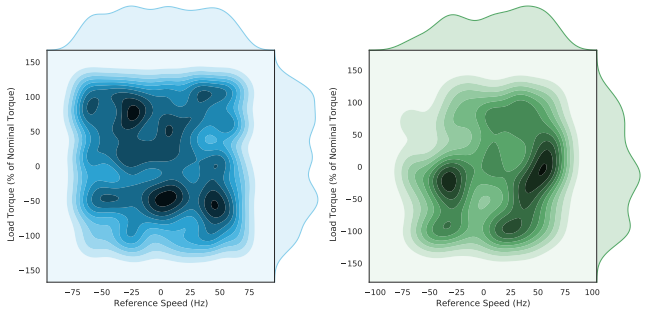


Fig. 1: Reference speed and load torque trajectories from one of the training samples.

B. Training and Validation Dataset



(a) Training zone (b) Validation zone

Fig. 2: Density plots of torque vs speed plans for all the simulations in training set and validation set.

We use our reference trajectory generator to generate 100 simulated paths totaling about 1000 speed and torque ramps in 150 minutes for the training set and 50 simulations totaling

about 200 ramps in 30 minutes for the validation set. Torque-speed plan density plots for training and validation zones are shown in Figure 2. We then simulate these trajectories using our Simulink model of a 4kW induction motor and collect simulation data every 4ms. The simulation dataset consists of the following electrical quantities: currents i_{sd} and i_{sq}, voltages u_{sd} and u_{sq} acting as inputs and rotor speed ω_r, and electromagnetic torque τ_{em} acting as outputs.

C. Test Dataset

The validation dataset is sufficient to evaluate neural network models on ML metrics (equations (6), (7), and (8)). To properly evaluate neural network models on EE performance metrics (see Section III-B), we generate five classical benchmark trajectories. These are divided into two categories:

1) Quasi-Static Benchmarks: At constant torque, reference speed goes from 70 to -70Hz in 50 seconds. Two constant torques are tested: no-load and 50% of the nominal load. We name these benchmarks, **Quasi-Static1** and **Quasi-Static2**, respectively. In the case of the 50% nominal load torque, the torque has already reached the steady-state before the start of the benchmark.

2) Dynamic Benchmarks: We generate three dynamic benchmarks to evaluate our neural network models:

(a) **Dynamic-Speed1:** Reference speed goes from 0 to 50Hz in 1 second at no load.
(b) **Dynamic-Speed2:** Reference speed goes from 50 to -50Hz in 1 second at 50% of nominal load.
(c) **Dynamic-Torque:** Load torque goes from 0 to 100% of nominal torque in 4ms with a constant 25Hz reference speed.

VI. Experimental Results

Model	Speed (ω_r)		Torque (τ_{em})	
	MAE	SMAPE	MAE	SMAPE
FCN	0.79	21.77%	0.57	48.66%
LSTM	0.11	18.76%	0.21	43.01%
CNN	0.06	19.14%	0.09	38.91%
Vanilla	0.05	18.94%	0.10	39.91%
Skip	0.08	19.08%	0.12	43.23%
RNN	0.06	19.31%	0.08	41.81%
BiRNN	0.05	**18.67%**	0.09	42.82%
DiagBiRNN	**0.03**	18.76%	**0.04**	**38.46%**
R^2 is 0.99 for all the networks for both quantities.				

TABLE I: ML metrics for the predictions done on benchmark set using standard models and the encoder-decoder variants. Aggregated results are shown for all 5 benchmarks.

During our experiments, we found that all our models were biased towards long-duration ramps present in the training data. This was due to the fact that when reference trajectories were generated, ramp durations were originally sampled from a uniform distribution. Motor responses to ramps being far more different for a small ramp duration variation when the ramps are short than when they are long, a uniform distribution was not adequate. To overcome this bias, generating data with ramps drawn from an exponential distribution plays a prominent role. This guarantees that our model can see more frequently short duration ramps during training. The benefit of using the exponential distribution for ramps in the reference trajectory generator is illustrated by ramp vs. overshoot plot in Figure 3.

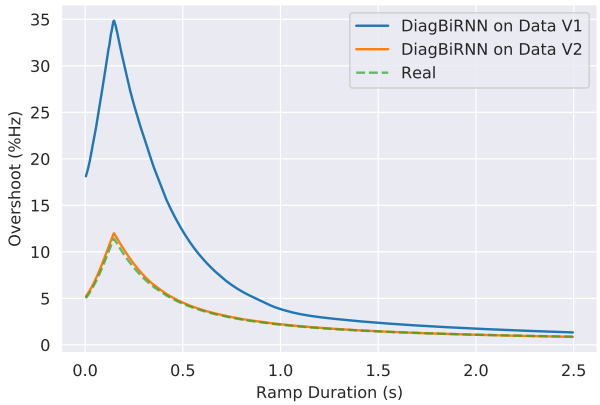


Fig. 3: Overshoot vs. ramp for DiagBiRNN network trained on two versions of data. Data V1 corresponds to training data in which ramps are sampled from a uniform distribution. Data V2 corresponds to training data in which ramps are sampled from an exponential distribution.

ML metrics for the results obtained on the quasi-static and dynamic benchmarks are reported in Table I. Smaller MAE and SMAPE values are desired for a good prediction model and R^2 closer to 1 is considered as a perfect predictor. We observe that ML metrics do not allow a clear comparison between different networks. We can see that any evaluation based on SMAPE and R^2 is difficult.

Model	$t_{2\%}$ (ms)	$t_{95\%}$ (ms)	E_{fol} (Hz)	$D\%$ (%)	E_{ss} (Hz)	$\Delta\tau_{max}$ ($\%\tau_{nom}$)
Real	**48**	**960**	**-0.02**	**2.16**	**0.00**	**32.69**
FCN	8	988	0.56	0.94	1.20	34.18
LSTM	44	933	-0.04	3.30	-0.13	33.49
CNN	40	**964**	-0.04	2.96	-0.04	32.57
Vanilla	44	968	-0.08	2.62	0.02	32.37
Skip	**48**	952	0.12	3.04	**0.01**	32.46
RNN	**48**	952	-0.04	2.28	0.02	32.82
BiRNN	44	944	-0.11	2.29	**0.01**	**32.67**
DiagBiRNN	44	952	**-0.01**	**2.21**	0.03	**32.67**

TABLE II: EE performance metrics obtained by different models on Dynamic-Speed1 benchmark.

ML metrics provide a global performance index on the benchmark set. Evaluating on individual benchmark using ML metrics does not yield a meaningful analysis. ML metrics provide good results because there is a large number of static points which are easy to predict compared to fewer dynamic points. EE metrics focus on these dynamic parts of the signal. For readability, we only plot predictions of the worst performing network (FCN), the best performing standard neural network (CNN) and the overall best performing network (DiagBiRNN) along with reference trajectory and real output

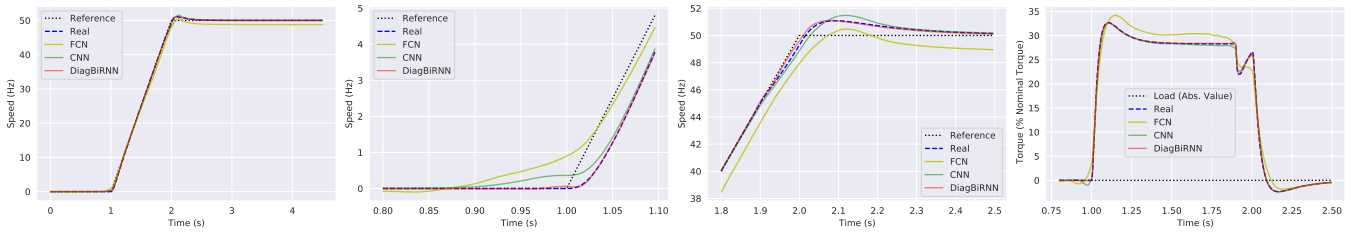


Fig. 4: Results on Dynamic-Speed1 benchmark.

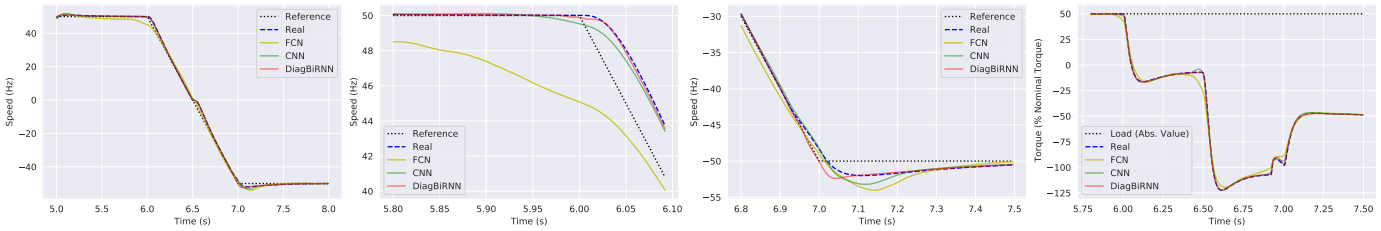


Fig. 5: Results on Dynamic-Speed2 benchmark.

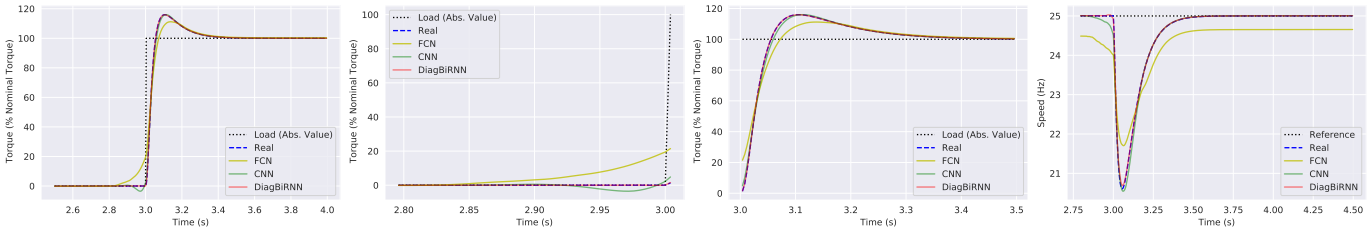


Fig. 6: Results on Dynamic-Torque benchmark.

Model	$t_{2\%}$ (ms)	$t_{95\%}$ (ms)	E_{fol} (Hz)	$D\%$ (%)	E_{ss} (Hz)	$\Delta\tau_{max}$ ($\%\tau_{nom}$)
Real	**52**	**956**	**0.10**	**2.00**	**0.00**	**72.33**
FCN	-144	944	1.06	4.03	0.37	69.35
LSTM	32	1052	**0.10**	7.46	-0.33	73.30
CNN	44	960	0.42	3.21	-0.08	**72.30**
Vanilla	44	**956**	0.34	3.55	-0.11	72.29
Skip	48	1036	0.45	5.07	-0.16	71.28
RNN	48	936	0.26	3.89	**-0.06**	72.25
BiRNN	48	940	0.02	3.28	-0.13	72.45
DiagBiRNN	**52**	948	0.15	**2.39**	-0.12	72.15

TABLE III: EE performance metrics obtained by different models on Dynamic-Speed2 benchmark.

Model	$t_{95\%}$ (ms)	$D\%$ (%)	E_{ss} ($\%\tau_{nom}$)	SD (Hz)
Real	**244**	**15.96**	**0.00**	**4.39**
FCN	252	11.19	-0.32	3.30
LSTM	**244**	**15.95**	-0.02	4.28
CNN	**244**	16.01	0.01	4.45
Vanilla	**244**	15.46	0.04	3.88
Skip	**244**	15.90	-0.02	**4.43**
RNN	**244**	15.87	**0.00**	4.03
BiRNN	**244**	16.01	0.01	4.23
DiagBiRNN	**244**	15.91	-0.02	4.31

TABLE IV: EE performance metrics obtained by different models on Dynamic-Torque benchmark.

(given by Simulink) for each of the dynamic benchmark. Figures 4 and 5 show plots for Dynamic-Speed1 and Dynamic-Speed2 benchmarks. From left to right, plots show speed, speed during the start of ramp, speed during end of the ramp, and torque. Tables II and III show EE performance metrics on Dynamic-Speed1 and Dynamic-Speed2, respectively. Overall, DiagBiRNN can be considered as the best choice as its metrics are closest to the metrics of the real output. Among standard neural networks, FCN performs worst and CNN performs better when compared to some of the encoder-decoder variants.

Figure 6 shows plots for Dynamic-Torque benchmark. From left to right, plots show torque, torque during start, torque during the end, and speed. Table IV shows results obtained on Dynamic-Torque benchmark. Acceptable values are less than 0.1Hz / 10ms / 0.2 percent point from real values and bad values are more than 0.25Hz / 25ms / 0.5 percent point from real values.

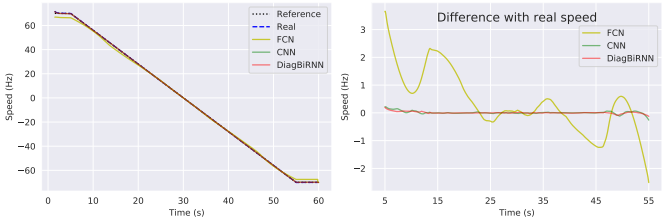


Fig. 7: Results on Quasi-Static1 benchmark.

For quasi-static benchmarks, we plot the speed during the long ramp and the difference between neural network speed

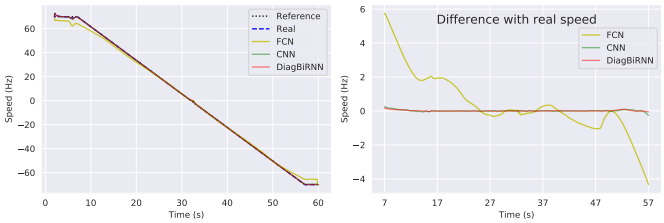


Fig. 8: Results on Quasi-Static2 benchmark.

Model	FCN	LSTM	CNN
Quasi-Static1	3.66	0.992	0.261
Quasi-Static2	5.751	0.629	0.259

Model	Vanilla	Skip	RNN	BiRNN	DiagBiRNN
Quasi-Static1	0.178	0.549	0.341	0.236	**0.198**
Quasi-Static2	0.336	0.444	0.258	0.346	**0.171**

TABLE V: Max absolute error (Hz) for static benchmarks.

prediction and real output speed. Figures 7 and 8 show plots for Quasi-Static1 and Quasi-Static2 benchmarks, respectively. The max absolute error for the predictions on quasi-static benchmarks are reported in Table V. It can be seen that DiagBiRNN has the smallest error and therefore is again closest to the real output speed, whereas FCN leads to the largest error.

VII. Conclusion

We present neural network methods to estimate speed and torque from currents and voltages of an induction motor. We show that, without care, these networks can be biased towards the data. We provide a realistic trajectory generator that helps in learning better dynamics. We also emphasize the limitations of machine learning metrics in understanding neural network real performance. By using dynamic and quasi-static benchmarks, we show that electrical engineering metrics are better suited to evaluate the merits of different neural networks. Both types of metrics show that our proposed DiagBiRNN network performs better on benchmarks. In the future, we plan to work with real motor data and generalize our network architectures for modeling different motor types.

References

[1] S. J. Campbell, *Solid-State AC Motor Controls*, 1987.

[2] C. S. Sisking, *Electrical Control Systems in Industry*, 1978.

[3] P. H. Truong, D. Flieller, N. K. Nguyen, J. Mercklé, and M. T. Dat, "Optimal efficiency control of synchronous reluctance motors-based ann considering cross magnetic saturation and iron losses," in *Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 4690–4695.

[4] M. Zhou, Y. Feng, C. Xue, and L. Xu, "Neural network based non-singular terminal sliding-mode control of induction motors," in *Annual Conference of the IEEE Industrial Electronics Society*, 2019, pp. 6538–6542.

[5] Z. Wang, C. Hu, Y. Zhu, S. He, K. Yang, and M. Zhang, "Neural network learning adaptive robust control of an industrial linear motor-driven stage with disturbance rejection ability," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2172–2183, 2017.

[6] A. A. Silva, A. M. Bazzi, and S. Gupta, "Fault diagnosis in electric drives using machine learning approaches," in *2013 International Electric Machines Drives Conference*, May 2013, pp. 722–726.

[7] R. Zhang, Z. Peng, L. Wu, B. Yao, and Y. Guan, "Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence," *Sensors*, vol. 17, no. 3, p. 549, 2017.

[8] Y. L. Murphey, M. A. Masrur, Z. Chen, and B. Zhang, "Model-based fault diagnosis in electric drives using machine learning," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 3, pp. 290–303, June 2006.

[9] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-d convolutional neural networks," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 7067–7075, 2016.

[10] S. Verma, N. Henwood, M. Castella, F. Malrait, and J.-C. Pesquet, "Modeling electrical motor dynamics using encoder-decoder with recurrent skip connection," in *AAAI conference on artificial intelligence*, New York, United States, 2020, pp. 1–8.

[11] R. Marino and P. Tomei, *Nonlinear control design: geometric, adaptive and robust*. Prentice Hall, 1996.

[12] A. Isidori, M. Thoma, E. D. Sontag, B. W. Dickinson, A. Fettweis, J. L. Massey, and J. W. Modestino, *Nonlinear Control Systems*, 3rd ed., Berlin, Heidelberg, 1995.

[13] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, *Nonlinear and Adaptive Control Design*, 1st ed., USA, 1995.

[14] R. Sepulchre, M. Jankovic, and P. V. Kokotovic, *Constructive Nonlinear Control*, 1st ed., USA, 1997.

[15] A. van der Schaft, *L2-Gain and Passivity Techniques in Nonlinear Control*, 3rd ed., 2016.

[16] "Passivity-based control of euler-lagrange systems: Mechanical, electrical and electromechanical applications," *Industrial Robot: An International Journal*, vol. 26, no. 3, 1999.

[17] F. Jadot, F. Malrait, J. Moreno-Valenzuela, and R. Sepulchre, "Adaptive regulation of vector-controlled induction motors," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 3, pp. 646–657, May 2009.

[18] A. K. Jebai, P. Combes, F. Malrait, P. Martin, and P. Rouchon, "Energy-based modeling of electric motors," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 6009–6016.

[19] G. Espinosa-Perez and R. Ortega, "State observers are unnecessary for induction motor control," *Systems & Control Letters*, vol. 23, no. 5, pp. 315 – 323, 1994.

[20] G. Espinosa-Perez and R. Ortega, "An output feedback globally stable controller for induction motors," *IEEE Transactions on Automatic Control*, vol. 40, no. 1, pp. 138–143, Jan 1995.

[21] P. J. Nicklasson, R. Ortega, G. Espinosa-Perez, and C. G. J. Jacobi, "Passivity-based control of a class of Blondel-Park transformable electric machines," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 629–647, May 1997.

[22] C. C. Chan and H. Wang, "An effective method for rotor resistance identification for high-performance induction motor vector control," *IEEE Transactions on Industrial Electronics*, vol. 37, no. 6, pp. 477–482, Dec 1990.

[23] J. Stephan, M. Bodson, and J. Chiasson, "Real-time estimation of the parameters and fluxes of induction motors," in *IEEE Industry Applications Society Annual Meeting*, Oct 1992, pp. 578–585 vol.1.

[24] R. Marino, S. Peresada, and P. Tomei, "Global adaptive output feedback control of induction motors with uncertain rotor resistance," *IEEE Transactions on Automatic Control*, vol. 44, no. 5, pp. 967–983, May 1999.

[25] L. Ortombina, F. Tinazzi, and M. Zigliotto, "Magnetic modeling of synchronous reluctance and internal permanent magnet motors using radial basis function networks," *Annual Conference of the IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1140–1148, 2018.

[26] G. P. Zhang, "Avoiding pitfalls in neural network research," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 1, pp. 3–16, 2007.

[27] O. Intrator and N. Intrator, "Interpreting neural-network results: a simulation study," *Computational Statistics & Data Analysis*, vol. 37, no. 3, pp. 373 – 393, 2001.

[28] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[29] F. Malrait, A. K. Jebai, and K. Ejjabraoui, "Power conversion optimization for hydraulic systems controlled by variable speed drives," *Journal of Process Control*, vol. 74, pp. 133 – 146, 2019.