Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors

Wilhelm Kirchgässner
Department of Power Electronics and Electrical Drives
Paderborn University
33095 Paderborn, Germany
kirchgaessner@lea.uni-paderborn.de

Oliver Wallscheid
Department of Power Electronics and Electrical Drives
Paderborn University
33095 Paderborn, Germany
wallscheid@lea.uni-paderborn.de

Joachim Böcker
Department of Power Electronics and Electrical Drives
Paderborn University
33095 Paderborn, Germany
boecker@lea.uni-paderborn.de

***Abstract*—Most traction drive applications using permanent magnet synchronous motors (PMSMs) lack accurate temperature monitoring capabilities so that safe operation is ensured through expensive, oversized materials at the cost of its effective utilization. Classic thermal modeling is conducted with e.g. lumped-parameter thermal networks (LPTNs), which help to estimate internal component temperatures rather precisely but also require expertise in choosing model parameters and lack physical interpretability as soon as their degrees of freedom are curtailed in order to meet the real-time requirement. In this work, deep recurrent and convolutional neural networks with residual connections are empirically evaluated for their feasibility on the sequence learning task of predicting latent high-dynamic temperatures inside PMSMs, which, to the authors' best knowledge, has not been elaborated in previous literature. In a highly utilized PMSM for electric vehicle applications, the temperature profile in the stator teeth, winding, and yoke as well as the rotor's permanent magnets are modeled while their ground truth is available as test bench data. A model hyperparameter search is conducted sequentially via Bayesian optimization across different random number generator seeds in order to evaluate model training consistency and to find promising topologies as well as optimization strategies systematically. It has been found that the mean squared error and maximum absolute deviation performances of both, deep recurrent and convolutional neural networks with residual connections, meet those of LPTNs, without requiring domain expertise for their design. Code is available at [1] to assist related work.**

***Index Terms*—Machine Learning, Deep Learning, Thermal Management, Permanent Magnet Synchronous Motor, Neural Networks**

I. Introduction

In most automotive motor applications, high power and torque densities as well as high efficiency are required, which makes the permanent magnet synchronous motor (PMSM) the preferred choice. In order to exploit the motor's full capabilities, high thermal stress on the motor's potentially failing components must be taken into account, that are the stator winding insulation, which may melt, and the permanent magnets that can suffer from irreversible demagnetization. A sensor-based temperature measurement would yield a rather precise thermal state, yet for the rotor part, it is technically and economically infeasible due to an electric motor's sophisticated internal structure and the difficult accessibility of the rotor. Stator temperature monitoring is realized with thermal sensors, but these are usually firmly embedded in the stator so that replacement is not an option, although sensor functionality deteriorates steadily. Since competitive pressure demands perpetual reduction of production costs, there is a commercial interest driving the investigation of sufficiently accurate real-time temperature estimation. In the last decades, various research efforts led to approaches that approximate the heat transfer process e.g. with equivalent circuit diagrams [2] called lumped-parameter thermal networks (LPTNs). This kind of model must forfeit physical interpretability of its structure and parameter values by significantly curtailing degrees of freedom in favor of the real-time requirement and, at the same time, expert domain knowledge is mandatory for the correct choice of parameter values.

In contrast, neural networks are known to be universal generalizers [3], with gradual degrees of complexity that can be adapted to the capacities of the application platform. More specifically, in the field of sequence learning tasks, recurrent neural networks (RNNs) and convolutional neural networks (CNNs) denote the state of the art in classification and estimation performance [4]. The concept is to parameterize neural networks entirely on empirical data, exploiting their rich expressive power without the need of domain expertise. The whole process from data acquisition over model training up to temperature monitoring in the field is sketched in Fig. 1. In previous literature on electric machines, neural networks were applied merely once in temperature prediction inside PMSMs [5], which indicated a comparative performance with respect to established real-time methods like LPTNs on low dynamic test bench data. However, only shallow networks were found to work out, and the scope of investigation was restricted to recurrent architectures.

In this work, several innovative and recently elaborated additions to neural network topologies are incorporated to

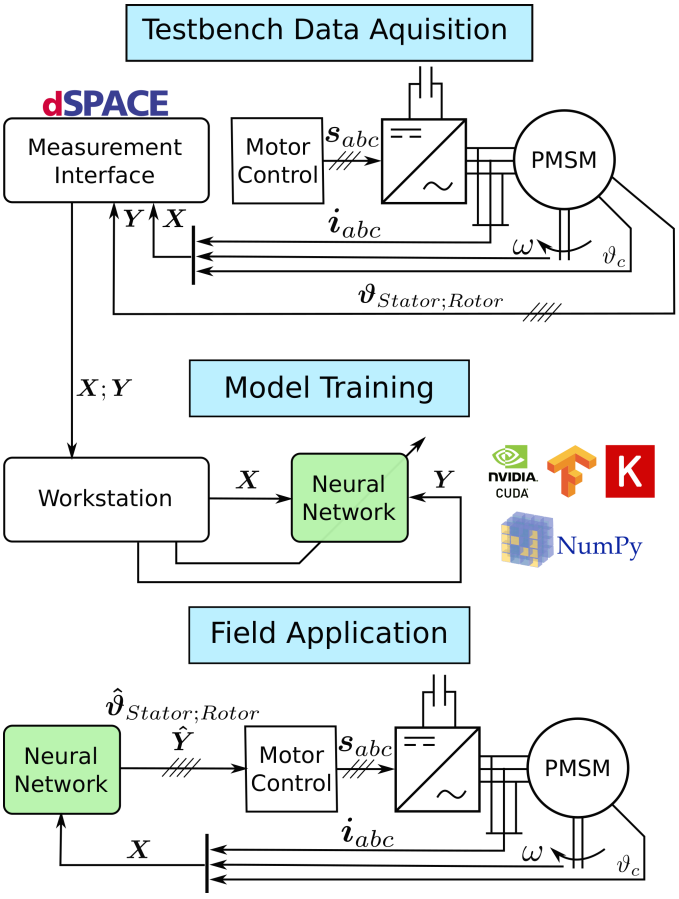

Fig. 1. Simplified scheme of the whole process from data acquisition at the test bench over model training up to the applied temperature monitoring in the field. Being able to precisely estimate stator and rotor temperatures allows for higher material utilization in the PMSM as derating switching schemes s_{abc} can be enabled right before critical operation.

examine the performance of neural networks on the estimation of high-fluctuating thermal time-series inside PMSMs, which is the first time in literature to the authors' best knowledge. These recent additions comprise e.g. residual connections [6], chrono initialization for RNNs [7] as well as CNNs with causal and dilated convolutions [8] along with many layers (up to seven, making it a deep architecture). An extensive hyperparameter search over topology-defining variables and optimization-affecting regularization schemes is conducted via Bayesian optimization across different random number generator seeds.

II. Neural Network Architectures

Neural networks come in many different flavors, and much research effort was put in finding better and better topologies [9], [10]. However, two fundamental architectures crystallized most effective for sequence modeling tasks, and are highlighted in the following.

A. Recurrent Architectures with Memory Blocks

RNNs experienced a success story in various domains e.g. speech recognition [11], [12], machine translation [13], audio sample generation [14] and whenever long-term memory is required [15]. This is established with memory blocks called long short-term memory (LSTM) that have gated forward and recurrent connections, and were first introduced in [16] in their today's popular form. Memory blocks help mitigate the exploding and vanishing gradient problem, which is one of the major challenges in training recurrent networks. A slightly simplified version of an LSTM is the gated recurrent unit (GRU) [17]. There is no clear empirical advantage of one topology over the other [18], besides a fewer amount of learnable parameters in GRU. Recurrence is needed in order to preserve past information (see Fig. 2), which is neglected by standard feed-forward neural networks (FNNs).

The general suitability of these topologies has been proven in [5]. There, a set of important hyperparameters were identified, on which this contribution is based. To ensure comparability to the following fundamentally different topology, in this work state-of-the-art RNN models are also evaluated on a hyperparameter search on the same high-dynamic temperature data.

B. Temporal Convolutional Networks

The RNN was the workhorse in sequential tasks for many years and has commonly been associated with this kind of modeling problem. Recently though, CNNs, which are predominantly applied in image processing, were discovered to be of superior behavior over RNNs, even in classic long-term memory sequential problems [8], [19]. This is on account of temporal convolutions that are causal and dilating - see Fig. 3. Whilst causal convolution means that outputs at time t are convolved only with activations from the previous layer at time t or earlier, dilated convolution denotes that elements of one single filter are not adjacent anymore but have a fixed step size instead. This dilated operation leads to an effectively expanded receptive field of the CNN, which would be directly proportional to the network's depth otherwise. Thus, a CNN is able to detect critical events far in the past and can outperform the long memory capability of the established RNN memory blocks. The specific CNN architecture utilized in this contribution was suggested by [8] and is that of the temporal convolutional network (TCN), which inherits recent advances of applications on sequential data yet distilled to a simpler form than commonly incorporated. Interestingly, [8] point out that this architecture is basically a time delay neural network as proposed in [20] around 30 years ago with the mere addition of zero padding in subsequent layers in order to obtain equal layer sizes and dilating contributions of previous-layer elements during a feed-forward pass.

One drawback of this topology, in contrast to the RNNs, is the necessity of a sliding window over the data requiring the choice of a window length w. Thereby, for a prediction at time t, a TCN requires all input observations from time t to $t-w$ during inference, which increases the memory demand. For an RNN only the current time regressors are sufficient because it condenses past information virtually in its inner state.

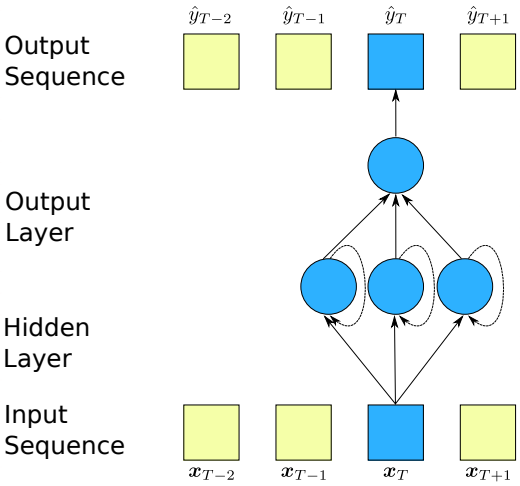

Fig. 2. Sequence learning with RNNs. For each observation $\boldsymbol{x}_T$ at time T there is a prediction $\hat{y}_T$. Past information is retained through the RNN's memory cells and recurrent connections.

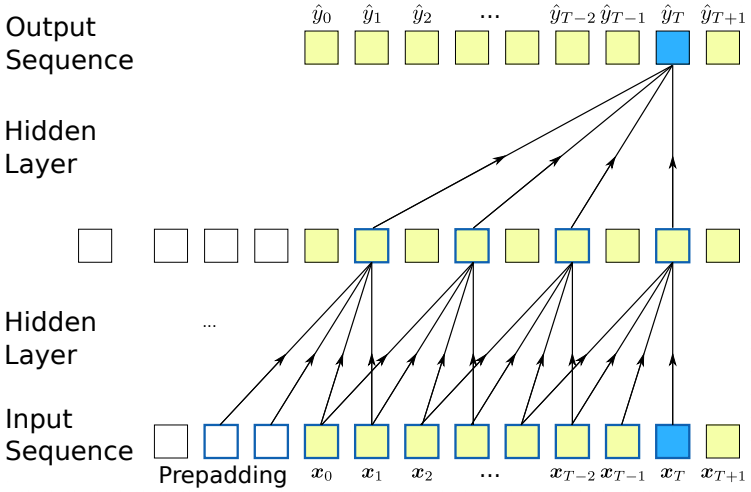

Fig. 3. Sequence learning with TCNs. Each prediction $\hat{y}_T$ is informed by the current observation $\boldsymbol{x}_T$ and a finite set of past observations (framed blue), denoting the network's receptive field, which depends on the number of hidden layers (here, two), the length of the filter l_{kernel} (here, four) and the dilation rate d in each layer. Each filter is convolved with the full sequence. All activations in subsequent layers are the sum of all filters at that time. Padding with zero ensures that layers and the TCN-output are of the same size as the input.

C. Residual Connections

First introduced in [6], residual blocks consist of two branches - one being transformed by a set of neural network layers $f(\cdot)$ and the other being an unaltered projection of the input x - that are added element-wise and transformed through some activation function α forming the residual block's output o:

$$o = \alpha(\boldsymbol{x} + f(\boldsymbol{x})) \tag{1}$$

In terms of the learned target, the accumulated gradients on the neural network weights would be calculated against the identity mapping rather than the complete course of the time series.

III. Bayesian Optimization of Hyperparameters

Neural networks and their optimization are rife with hyperparameters that have individual effects on the final result. Common approaches for finding a suitable or even an optimal set of those parameters are either grid search or better random search [21]. In this contribution, a more systematic approach is followed by exploiting previously evaluated points in the hyperparameter space via Bayesian optimization. A comprehensive review of Bayesian optimization is given in [22].

In the context of automated machine learning, Bayesian optimization describes a sequential model-based approach to maximize (or minimize) a computationally expensive objective function. With S being the search space and g denoting an auxiliary function that maps the given hyperparameter set s to the performance of a machine learning model, the next point to observe is given by

$$\tilde{s} = \underset{s}{\operatorname{argmax}}\, g(s) \quad \text{with} \quad s \in S. \tag{2}$$

Effectively, a surrogate is built for the objective $g(\cdot)$, that is cheap to compute and provides iterative uncertainty information over the search space. In order to obtain new promising points in the search space given already explored points, one has to initially choose a prior over functions e.g. Gaussian processes (GPs) or random forests (RFs), which represent a prior belief over the theoretical neural network predictions. Moreover, one has to select an acquisition function whose maximum will determine the next point to examine. While the prior is updated for each evaluated point in the search space, the acquisition function acts as a heuristic for assessing the cost of the sequence of selected hyperparameter sets. The exploration-exploitation trade-off is dealt with in the acquisition functions through generating maxima where either the surrogate model's uncertainty is large or the cost function is near optimal.

IV. Nonlinear Thermal Modeling

As shown by [2], a three-phase PMSM mounted on a test bench yielded the available data, which is consistently sampled at a frequency of $f_s = 2\,\mathrm{Hz}$. Stator temperatures are measured with classical thermal sensors, whereas the rotor temperature is recorded with a telemetry unit.

In this work, the data comprise the 35-hour investigation basis of [5] but extended by 100 additional hours of measurement with not only low- but also especially high-fluctuating profiles. New profiles are designed to trace different random walks in the torque-speed-space that resemble real-world profiles better than the previously available data did. Tab. I compiles the considered quantities that represent the trained models' input and output. While denoted input parameters are commonly accessible in real-world traction drive systems, target

temperatures are not, hence, supervised learning requires data measured on enhanced motor test equipment.

The Tensorflow-Toolbox with its high-level API Keras [23] is used in all experiments.

A. Data Preprocessing and Feature Engineering

All representations of the data are standardized on their sample mean and sample unit variance exhibited in the training set. The exponentially weighted moving average (EWMA) and standard deviation (EWMS) are taken into account so that for every timestep t the following two terms are computed for each input parameter X and adhered to the models' input (see Tab. I):

$$\mu_t = \frac{\sum_{i=0}^{t} w_i x_{t-i}}{\sum_{i=0}^{t} w_i} \quad \text{and} \quad \sigma_t = \frac{\sum_{i=0}^{t} w_i (x_i - \mu_t)^2}{\sum_{i=0}^{t} w_i}, \quad (3)$$

where $w_i = (1-\alpha)^i$ with $\alpha = 2/(s+1)$ and s being the *span* that is to be chosen. Multiple values for the span can be applied leading to different smoothed versions of all time-series and their standard deviation.

B. Cross-Validation

In all experiments, every model is cross-validated on two load profiles corresponding to seven hours that are unseen during training and hold time characteristics similar to those in the remaining data. These two sequences denote the test set as opposed to the training set, which is utilized iteratively during neural network training. Moreover, one certain profile of roughly 4.6 hours out of the training set is also withheld from training (validation set) in order to validate the generalizability of the trained model after each iteration (epoch) over the training set. The validation set is used to apply *early stopping* [4] i.e. stopping training after the cost function on this set is not improving anymore for a certain delta during a given amount of epochs.

C. Training Strategy

During training, the learning rate of the optimizer (here, Adam [24], as it has asserted itself in [5]) is reduced by factor ten after there is no improvement anymore in training set loss for ten consecutive epochs. Reducing the learning rate on such loss plateaus is a common heuristic in neural network training for improving convergence and local minima exploration [4].

Each supervised training is conducted upon a chosen loss or cost function. Here, the mean squared error (MSE) between predicted and real temperature profile is consulted throughout all evaluations.

Due to the fundamentally different topologies between RNNs and TCNs, the following training strategies are unique to either of them. For a nonlinear dynamic system as a PMSM, the initial starting point of all quantities at the first observation point seems likely to be important for the estimation of all following series trends, such that for RNNs the starting point is always among the very first of a profile. The batch size is bounded on the amount of available profiles and can be increased by downsampling all sequences by e.g. every fourth element, repeated on an increasing offset up to three, which would result in quadrupling the number of sequences. In contrast, the TCN depends on sliding windows, hence, the initial observation is not necessarily within the window, yet the higher bound for the batch size is much larger - specifically, at the amount of total observations. The amount of patterns the TCN might learn from a window is determined by each layer's number of filters (corresponds to nodes in an RNN), that already inherently downsample the input window through the convolving kernel, thus, making the choice of a downsample rate redundant.

Another adjusting training knob unique to an RNN is the choice of the truncated backpropagation through time (TBPTT) length. It defines the ratio of forward passes per backward pass during training with backpropagation.

TABLE I
MEASURED INPUT AND TARGET PARAMETERS

Parameter name	**Symbol**
Inputs	
Ambient temperature	ϑ_a
Liquid coolant temperature	ϑ_c
Actual voltage d-axis component	u_d
Actual voltage q-axis component	u_q
Actual current d-axis component	i_d
Actual current q-axis component	i_q
Motor speed	n_{mech}
Derived inputs	
Voltage magnitude $\sqrt{u_d^2 + u_q^2}$	u_s
Current magnitude $\sqrt{i_d^2 + i_q^2}$	i_s
Electric apparent power $1.5 * u_s * i_s$	S_{el}
Targets	
Permanent magnet temperature	ϑ_{PM}
Stator teeth temperature	ϑ_{ST}
Stator winding temperatue	ϑ_{SW}
Stator yoke temperature	ϑ_{SY}

D. Hyperparameters

The choice over the hyperparameter set is informed by the work in [5] insofar as those hyperparameters whose optima were found around the very same value were excluded from this work's search, whereas other tunable factors with auspicious effect on training were introduced. Tab. II lists the considered hyperparameters and their search interval. Some entries that are not self-explanatory or have not been already mentioned shall be illuminated: Weight shrinkage regularization is applied via adding the L2 norm of the weight matrices as penalty term to the MSE cost function (*weight decay* [25]). Further regularizing effects are achieved with dropout [26], [27], gradient clipping and normalization [28], as well as gradient noise on layer activations [29]. The dropout parameter determines the ratio of nodes or filters per layer that are randomly disabled during training, while normalizing and clipping gradients help to mitigate the exploding and vanishing gradient problem. Gradient noise on layer activations introduces variety and can be understood as enriching the data [29]. For each layer, there is the same amount of nodes n_{units}. In

TABLE II
HYPERPARAMETER SEARCH INTERVALS

Hyperparameter (abbrev.)	Interval discretization[a]/ choices	
No. hidden layers (n_l)	[1,7]	integers
No. hidden units (n_{units})	[4, 256]	integers
L2 regularization rate (α)	[10^{-9}, 10^{-1}]	log-space
Dropout rate (dropout)	[0.2, 0.5]	uniform
Initial learn rate (η)	[10^{-7}, 10^{-2}]	log-space
Span 1 (s_1)	[500, 1500]	integers
Span 2 (s_2)	[2000, 3000]	integers
Span 3 (s_3)	[4000, 6000]	integers
Span 4 (s_4)	[7000, 9000]	integers
RNN only		
Architecture (arch)	LSTM, GRU or Residual LSTM	
Grad. clip-norm (clipnorm)	[0.25, 15]	uniform
Grad. clipping (clipvalue)	[10^{-2}, 1]	log-space
Gradient noise (σ_{GN})	[10^{-9}, 10^{-2}]	log-space
TBPTT length (tbptt)	[1, 128]	integers
Downsample rate (f_S)	[1, 32]	integers
TCN only		
Architecture (arch)	Plain or Residual	
Kernel size (l_{kernel})	[2, 7]	integers
Window size (w)	[8, 128]	integers

[a]uniformly sampled (possibly over log space)

order to limit memory consumption, only four possible values for the span in EWMA and EWMS are explored.

V. EXPERIMENTAL RESULTS

There are four searches conducted: In experiment Ξ and Φ, RNNs are examined with targets being the stator quantities (ϑ_{SW}, ϑ_{SY}, and ϑ_{ST}) for the former and the permanent magnet temperature ϑ_{PM} in the rotor for the latter (see Tab. I). In experiment Ψ and Ω, the same target distribution as in the previous two experiments is followed but with TCNs instead. Each set of hyperparameters is repeated five times with a different random number generator seed in order to assess consistency and the impact of initialization noise.

For all experiments, the scikit-optimize framework [30] is used with Gaussian processes as surrogate model. The acquisition function is either of upper confidence bound (UCB), expected improvement (EI) or probability of improvement, which calculate new candidate points independently, and where the most promising proposal serves as next evaluation point in every iteration.

The hyperparameter searches are depicted in Fig. 4, and their results are compiled in Tab. III.

A. Model performance

In absolute numbers, the found optima for both, RNNs and TCNs, achieve state-of-the-art scores on the task of temperature estimation for the stator and rotor on a separated test set. It becomes evident, that the TCN search achieved an overall better MSE score than the RNN search and exhibits less scatter across different random number generator seeds. Moreover, three out of four searches ended up with complex models inheriting a lot of model parameters with an only slightly improved score compared to the small neural network found in experiment Ξ.

TABLE III
HYPERPARAMETER SEARCH RESULTS

Experiment	Ξ	Φ	Ψ	Ω
Total iterations	88	147	91	93
Runtime in days	> 25	> 38	> 42	> 51
	Characteristics of the best model found			
Performance (MSE in K^2)	2.78	3.26	**1.91**	**1.52**
Model parameters	>850k	**1.9k**	>320k	>67k
n_l	2	1	4	2
n_{units}	256	4	121	126
α	10^{-9}	0.1	10^{-8}	10^{-9}
dropout	0.37	0.5	0.29	0.35
η	$1.4{\cdot}10^{-3}$	10^{-2}	$1.4{\cdot}10^{-4}$	10^{-4}
s_1	500	1500	620	500
s_2	2204	2000	2915	2161
s_3	6000	4000	4487	4000
s_4	9000	7000	8825	8895
$arch$	Res. LSTM	Res. LSTM	Res.	Res.
clipnorm	9.4	0.25	-	-
clipvalue	0.076	0.01	-	-
σ_{GN}	10^{-9}	10^{-2}	-	-
tbptt	42	128	-	
f_S	32	1	-	-
l_{kernel}	-	-	6	2
w	-	-	32	33

The TCN optima's performance (i.e. from experiments Ψ and Ω) for the stator and rotor temperatures on one of the two test sets is illustrated in Fig. 5. Noteworthy, in contrast to RNNs, the TCNs have a very high estimation error up to $30\,\mathrm{K}$ right in the beginning of a time-series estimation, which is due to the reliance on sliding windows that have no complete information there. However, after this initial phase of a few seconds, the absolute deviation is rigorously below $6\,\mathrm{K}$ throughout the full sequence. Taking into account that in this work also load profiles of very high dynamics are considered, the aforementioned results denote a significant improvement in terms of the MSE and the L_∞-norm (i.e. the maximum absolute deviation $\|e\|_\infty$) over the shallow networks investigated in earlier work [5] ($\|e\|_\infty > 13\,\mathrm{K}$).

B. Optimal Hyperparameters

A closer look at the found hyperparameters in Tab. III reveals that no notably deep neural network was found fruitful in this work: Three out of four searches ended up with moderate topologies of two to four layers and over 100 nodes per layers. Surprisingly, experiment Φ suggests a minimal model with one layer and four nodes trained with a dropout rate of 0.5, which translates to effectively half the amount of the already few nodes during training time. On top of that, this is the only experiment where strong weight regularization α proved beneficial although only few weights exist. This finding advocates that patterns in the preprocessed dataset are already distinct enough to be captured by low order models. The next chapter will further elaborate on this finding.

Furthermore, residual architectures were found to perform best without exception, and the dropout rate should be at

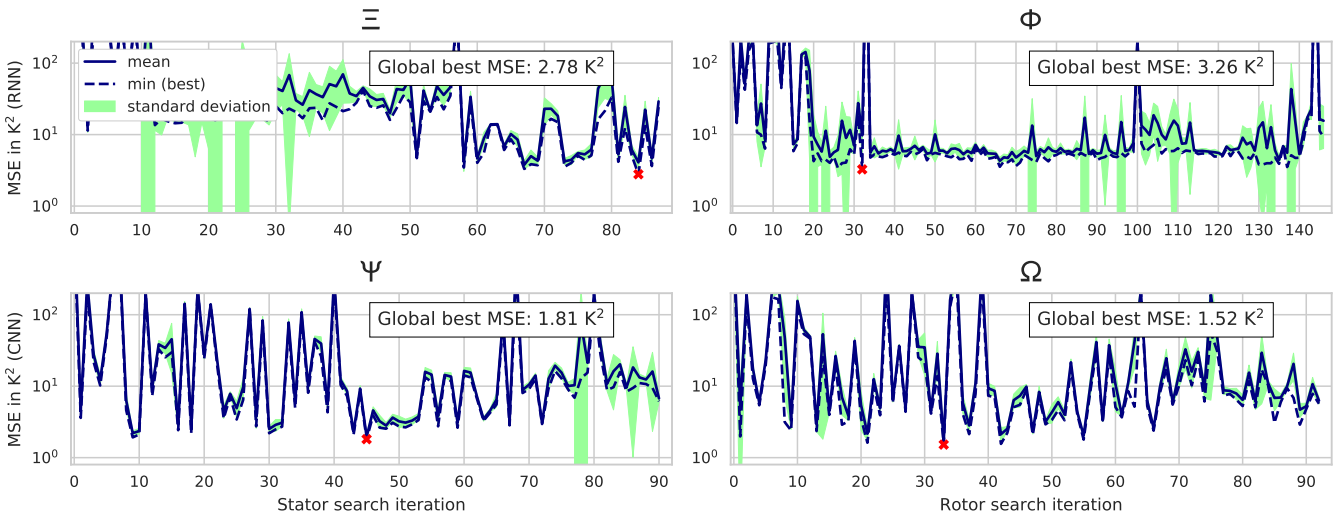

Fig. 4. Score trend over search iterations. Each iteration involves five repetitions of the same hyperparameter set with a different random number generator seed. First 15 iterations are randomly sampled. Global best iteration is marked with a red cross.

least 30 %. Moreover, TCNs perform best with a window length of $32 - 33$. All other hyperparameters cover most of each hyperparameter's interval across the experiments and no overall best choice can be recommended.

VI. Discussion

Building on the results in Tab. III, minimal topologies are examined also for the stator temperatures. Residual RNNs with as few as four memory blocks in one hidden layers are able to estimate the stator temperatures with an MSE $= 2.99\,\mathrm{K}^2$ and $\|e\|_\infty = 7.11\,\mathrm{K}$, when using the optimal hyperparameters found in experiment Φ, albeit the training consistency exhibits a larger scatter across different random number generator seeds. This inconsistency was already mentioned in [5].

However, three insights can be inferred from the fact, that it is possible to train an RNN on stator temperatures with hyperparameters from an experiment that optimized rotor temperatures:

- One does not need to distinguish between stator and rotor targets when designing a neural network i.e. four targets are also possible here (confirmed by subsequent experiments),
- Bayesian optimization with GPs tends to get stuck in local optima, as each search came to an end at very diverse solutions. In view of the drastically reduced model order found to be optimal here, other hyperparameter optimization techniques e.g. particle swarm optimization or genetic algorithms, might tend to converge better in a yet feasible time,
- high estimation accuracy can be achieved with very small neural networks for both, stator and rotor temperatures, as long as residual architectures are utilized on several exponentially weighted moving averages or standard deviations of the raw sensor data. This can be deduced by comparing this work's experiment design with that in [5].

Deliberately designing a small residual TCN e.g. with $n_l = 2; n_{units} = 4; l_{kernel} = 2$ (corresponds to 1200 model parameters), and training on all four target temperatures under optimal hyperparameters from experiment Ψ, leads to a still decent performance with MSE $= 3.6\,\mathrm{K}^2$ and $\|e\|_\infty = 13.8\,\mathrm{K}$. This further affirms the aforementioned insights.

A. Related Work

In this contribution, temperature estimation in PMSMs under high dynamic load profiles are investigated the first time in literature to the authors' best knowledge, thus, making comparisons to previous modeling approaches skewed. Nonetheless, to mention a few, [2] and [31] achieved an L_∞ norm in the range of $3\,\mathrm{K}...8\,\mathrm{K}$ with LPTNs, while [32] was able to estimate the rotor temperature with an $\|e\|_\infty$ lower than $5\,\mathrm{K}$ by utilizing a fundamental wave flux observer. Downsides worth mentioning for either approach is the requirement of precise motor geometry and cooling system data for the former, as outlined already in section I, and high sensitivity to magnetic saturation as well as lower accuracy at low motor speed in the latter case. A neural network or black-box approach, however, is independent of motor data sheets through being fitted on empirical measurements exclusively, and, for the same reason, does not suffer in estimation accuracy during operation points where physical model assumptions are violated.

Another noteworthy approach is rotor temperature monitoring through high-frequency signal injection [33], [34]. Here, stator winding resistances and the magnets' magnetization level are gauged to obtain thermal indicators. With no further details on the validation technique, [35] state to achieve an absolute error $\|e\|_\infty$ of less than $3\,\mathrm{K}$. This approach also requires sophisticated modeling efforts i.e. the precise identification of

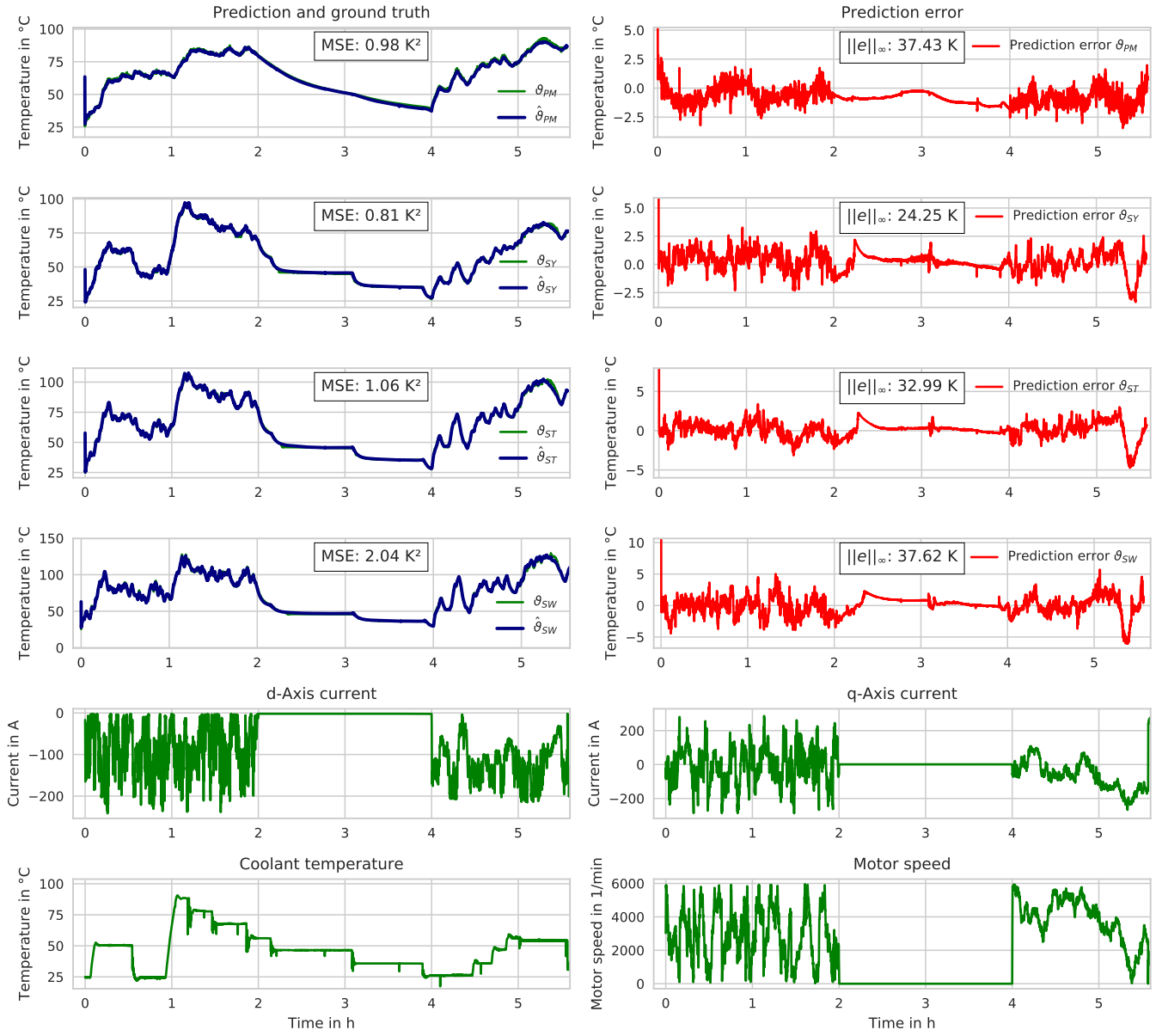

Fig. 5. TCN temperature estimation on the first test set. Utilized Models are those optima found in experiments Ψ and Ω. Left: Absolute temperature profiles of ground truth (green) and the neural network's prediction (blue). Right: Corresponding prediction error. Below four plots represent selected input parameters to showcase the highly dynamic operation.

the motor and inverter. Moreover, it inflicts additional losses on the system and may prove unsuitable in some applications due to electromagnetic interference [36].

B. Implementation Considerations

Since neural networks comprise substantially more parameters than usual white- or grey-box models, all optimal topologies were further monitored for their runtime on embedded hardware. In this work, a *Raspberry Pi 3 Model B+* with *Raspbian Stretch 4.14* and *tensorflow 1.13* were utilized to measure runtime during inference. Mean, minimum, and standard deviation of the models' estimation runtime across 10000 iterations are shown in Tab. IV. It becomes obvious that moderate topologies of almost 1 million parameters even complete an estimation in under 40 ms on average. Taking into account that the used dataset is sampled at 2 Hz, an estimation would be necessary every half second at maximum. Thus, neural networks contemplated in this contribution are real-time capable on embedded hardware, assuming the performance gap between the considered hardware and those usually applied in industry is not too large.

TABLE IV
RUNTIME OF OPTIMAL MODELS DURING INFERENCE

Experiment	Ξ	Φ	Ψ	Ω
mean $\pm$ std. dev. in ms	36 ± 26	9 ± 8	28 ± 6	14 ± 5
minimum in ms	23.2	7.1	22.5	11.5

VII. CONCLUSION AND OUTLOOK

In this work, it has been shown that engineers can circumvent special domain expertise when designing real-time capable models monitoring important component temperatures inside a PMSM, by relying solely on recorded test bench data. Through autonomously conducted hyperparameter searches, it was possible to demonstrate that residual RNNs with memory blocks and residual TCNs achieve state-of-the-art estimation accuracy also during high dynamic drive cycles. Diverse optimal topologies were found during Bayesian optimization of hyperparameters that differ in the amount of model parameters and execution speed on embedded hardware. However, estimation accuracy is similar for all optima and even bigger architectures were found to run in under real-time on a *Raspberry Pi*.

Although the empirical results are promising, it remains unanswered, whether neural networks may generalize across different motors from the same manufacturer overcoming production tolerances - not to mention different motor types from diverse providers. Moreover, future research may investigate the uncertainty of neural network predictions in order to enable probabilistic estimations, as well as the fusion of black-box models with established approaches.

REFERENCES

[1] Wilhelm Kirchgässner. Deep-PMSM. https://github.com/wkirgsn/deep-pmsm, 2018.

[2] Oliver Wallscheid and Joachim Böcker. Global Identification of a Low-Order Lumped-Parameter Thermal Network for Permanent Magnet Synchronous Motors. *IEEE Transactions on Energy Conversion*, 2016.

[3] K. Hornik, M. Stinchcombe, and H. White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 1989.

[4] I. A. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[5] Oliver Wallscheid, Wilhelm Kirchgässner, and Joachim Böcker. Investigation of Long Short-Term Memory Networks to Temperature Prediction for Permanent Magnet Synchronous Motors. In *Proceedings of the International Joint Conference on Neural Networks*, 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, 2015.

[7] Corentin Tallec and Yann Ollivier. Can Recurrent Neural Networks Warp Time? *CoRR*, 2018.

[8] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *CoRR*, 2018.

[9] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems.*, 2016.

[10] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An Empirical Exploration of Recurrent Network Architectures. In *JMLR*, 2015.

[11] Alex Graves and Jürgen Schmidhuber. Framewise Phoneme Classification With Bidirectional LSTM Networks. In *Proceedings of the International Joint Conference on Neural Networks*, 2005.

[12] H. Sak, A. Senior, K. Rao, and F. Beaufays. Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition. *CoRR*, 2015.

[13] Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the Rare Word Problem in Neural Machine Translation. *CoRR*, 2014.

[14] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. *CoRR*, 2016.

[15] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR*, 2015.

[16] Felix A. Gers and Fred Cummins. Learning to Forget: Continual Prediction with LSTM. In *Ninth International Conference on Artificial Neural Networks*, 1999.

[17] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, 2014.

[18] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, 2014.

[19] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *CoRR*, 2016.

[20] K. Lang, A. Waibel, and G. E. Hinton. A Time-Delay Neural Network Architecture for Isolated Word Recognition. *Neural Networks*, 1990.

[21] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 2012.

[22] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the Human out of the Loop: A Review of Bayesian Optimization. In *Proceedings of the IEEE*, 2016.

[23] François Chollet and Others. Keras. https://keras.io, 2015.

[24] Diederik P. Kingma and Jimmy Lei Ba. Adam: Method for Stochastic Optimization. *CoRR*, 2015.

[25] A. Krogh and J. A. Hertz. A Simple Weight Decay Can Improve Generalization. *Advances in Neural Information Processing Systems*, 1992.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 2014.

[27] Yarin Gal and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, 2016.

[28] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the Exploding Gradient Problem. *Proceedings of The 30th International Conference on Machine Learning*, 2012.

[29] Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding Gradient Noise Improves Learning for Very Deep Networks. 2015.

[30] Tim Head and Others. Scikit-Optimize. https://scikit-optimize.github.io/, 2018.

[31] T. Huber, W. Peters, and J. Böcker. Monitoring Critical Temperatures in Permanent Magnet synchronous Motors Using Low-Order Thermal Models. In *International Power Electronics Conference*, 2014.

[32] A. Specht, O. Wallscheid, and J. Böcker. Determination of Rotor Temperature for an Interior Permanent Magnet Synchronous Machine Using a Precise Flux Observer. In *International Power Electronics Conference*, 2014.

[33] Simon Delamere Wilson, Paul Stewart, and Benjamin P. Taylor. Methods of Resistance Estimation in Permanent Magnet Synchronous Motors for Real-Time Thermal Management. *IEEE Transactions on Energy Conversion*, 2010.

[34] M. Ganchev, C. Kral, H. Oberguggenberger, and T. Wolbank. Sensorless Rotor Temperature Estimation of Permanent Magnet Synchronous Motor. In *IECON Proceedings (Industrial Electronics Conference)*, 2011.

[35] D. D. Reigosa, F. Briz, P. Garcia, J. M. Guerrero, and M. W. Degner. Magnet Temperature Estimation in Surface PM Machines Using High-Frequency Signal Injection. *IEEE Transactions on Industry Applications*, 2010.

[36] O. Wallscheid, T. Huber, W. Peters, and J. Böcker. Real-Time Capable Methods to Determine the Magnet Temperature of Permanent Magnet Synchronous Motors — A Review. In *40th Annual Conference of the IEEE Industrial Electronics Society*, 2014.