Towards End-to-end Deep Learning Analysis of Electrical Machines

N. Gabdullin¹, *Member, IEEE*, S. Madanzadeh¹, *Student Member, IEEE*, and A. Vilkin²

¹Department of Electrical and Electronic Engineering, Chung-Ang University, Seoul, 06974, South Korea

²Department of Applied Mathematics and Computer Science, National Research Nuclear University, Moscow, 115409, Russia

Convolutional Neural Networks (CNNs) and Deep Learning (DL) revolutionized numerous research fields including robotics, natural language processing, self-driving cars, healthcare, and others. However, DL is still relatively under-researched in fields such as physics and engineering. Recent works on DL-assisted analysis showed emerging interest and enormous potential of CNN applications. This paper explores the possibility of developing an end-to-end DL pipeline for the analysis of electrical machines. The CNNs are trained on conventional finite element method (FEA) data to predict the output torque curves of electric machines. FEA is only used for dataset collections and CNN training, whereas the analysis is done solely using CNNs. The required depth in CNN architecture is studied by comparing a simplistic CNN with three ResNet architectures. The effects of dataset balancing and data normalization are studied and torque clipping inspired by offset normalization is proposed to ease CNN training and improve the prediction accuracy. The relation between architecture depth and accuracy is identified showing that deeper CNNs improve the curve shape prediction accuracy even after torque magnitude prediction accuracy saturates. Over 90% accuracy for analysis conducted under a minute is reported for CNNs, whereas FEA of comparable accuracy required 200 hours. Predicting multidimensional outputs can improve CNN performance, which is essential for multiparameter optimization of electrical machines.

Index Terms—Convolutional Neural Networks, deep learning, electrical machines, finite element analysis.

I. INTRODUCTION

CONVOLUTIONAL Neural Networks (CNNs) and Deep Learning (DL) have revolutionized various research fields becoming indispensable for computer vision, natural language processing, healthcare, robotics, and others real-life applications [1]–[3]. However, future saturation in conventional research areas is expected along with the growth of DL applications in areas where DL is currently underrepresented, including physics and various engineering fields [4]–[6]. Recent engineering applications include fault detection and material characterization along with DL-assisted analysis techniques [7]–[9].

Nowadays, finite element analysis (FEA) is the most widely applied method for analysis and design of electric machines due to its high accuracy and flexibility. This method is remarkably versatile being applied for electrical, mechanical, fluid dynamics and other analysis types. FEA uses numerical methods to find an approximate solution to field problems guided by, for instance, Maxwell equations in electromagnetism that have no analytical solution. However, the high accuracy comes at a price of enormous computational time resulting in high computational costs.

Whereas FEA finds the solution by minimizing errors over numerous iterations, the computation results are eventually dismissed and never used for future analyses. Hence, solving a problem for which a solution is known requires as much time as solving it for the first time. However, would there be a method that takes advantage of previous computations to speed up future analyses, the data from thousands of previously published FEA papers could be used to develop a high-accuracy high-speed analysis tool.

DL can facilitate such tools because after a time-consuming training stage, the prediction stage can be performed

extremely fast. DL previously was applied to predict the distribution of arterial stresses as mechanical FEA surrogate model [10]. This work is computationally similar to electromagnetic field distribution prediction with DL [11]. However, both approaches are hardware-demanding due to extremely large size of input and output data vectors and large DL models (e.g., 32 layer CNN [11]) that are not necessarily optimized for engineering purposes.

Recently, Sasaki and Igarashi [12], as well as Doi *et al.* [13] and Asanuma *et al.* [14], applied DL to assist a genetic optimization algorithm to find optimal motor geometries by minimizing the number of FEA subroutine calls substantially reducing the analysis time. Some researchers approached the torque prediction problem using classification by dividing the complete torque range into sub-areas, whereas others formulated this problem as a regression one. Remarkably, the datasets used for DL training were about 5,000 samples in size showing the possibility of DL training on modest datasets, unlike extremely large ones commonly used in other research areas [15], [16]. This is significant due to extremely high costs associated with dataset collection in engineering unless community effort is utilized.

However, in aforementioned papers DL CNNs were used only as a component and final analyses were still conducted using FEA. Furthermore, applying DL to assist another analysis tool possibly hinders the utilization of full potential of DL [17]–[19]. In this study, DL CNNs are used for end-to-end accurate prediction of the output of Interior Permanent Magnet Synchronous Motors (IPMSMs). Furthermore, it is shown that training DL CNNs to predict multidimensional outputs improves the analysis accuracy and allows more comprehensive information about IPMSM performance to be obtained.

IPMSM rotor geometries and output torque curves are used

Fig. 1. Initial IPMSM geometry in JMAG Designer that includes the PM (1), rotor slits (2), rotor core (3), air gap (4), stator back iron (5), stator slots (6), and stator teeth (7).

to train DL CNNs. Predicting torque curves rather than average torque is advantageous for electrical machine analysis [20]. Furthermore, several DL CNN architectures are studied to determine the required architecture complexity, including a simplistic custom architecture and ResNets of varying depth. A modest 4,000 sample dataset collected using FEA is used for training, and the difficulties associated with collecting large datasets are discussed. Furthermore, different types of data normalization, dataset balancing, and potential errors associate with CNN training are also discussed. Finally, the high accuracy of CNN torque prediction is shown and the advantages over FEA are discussed along with future potential of DL CNNs to exceed FEA capabilities.

II. FEA ANALYSIS OF IPMSMs

A. IPMSM rotor geometries in JMAG designer

For DL CNN training, the IPMSM motor geometries were generated using JMAG Designer [21]. This tool provides a typical initial geometry shown in Fig. 1, and several parameters that can be varied including stator and rotor diameters, numbers of poles and slots, slot size, permanent magnet (PM) size, and size of rotor slits.

This study focuses exclusively on rotor geometry, so stator parameters remain unchanged for all models. Furthermore, rotor outer diameter remains constant, leading to the position and size of the PM, and size and geometry of air slits being the variables. Table I summarizes the parameters and parameter variation ranges for studied IPMSMs. Stator current and PM grade remain unchanged for all experiments. The same approach can be used to obtain reluctance motor rotor geometries for DL studies [22]. Three phase 60 Hz excitation is applied in all models.

Because DL was previously used to predict optimal IPMSM performance, commonly only near-optimal geometries were used to train DL models. However, this might lead to DL developing bias towards predicting high-torque outputs, potentially predicting good results for absurd geometries for which extremely low performance is realistically expected [8]. To address this potential issue, an unbiased approach to data collection is utilized in this study resulting in both optimal and

Fig. 2. Example IPMSM rotor geometries for (a) high torque positive, and (b) extremely low torque negative examples.

TABLE I
DESIGN PARAMETERS OF STUDIED IPMSMs

Parameter	Value	Parameter	Value
Number of poles	4	Number of slots	24
PM grade	N35	Coil current	3 A
Rotor inner diameter	13.6 mm	Stator inner diameter	47.7 mm
Rotor outer diameter	46.7 mm	Stator outer diameter	93.4 mm
PM position	$10 \div 20$ mm	Slot depth	12 mm
PM thickness	$1 \div 4$ mm	Slot opening width	0.85 mm
PM width	$2 \div 8$ mm	Tooth width	2.8 mm
Slit depth	$0.5 \div 3$ mm	Tooth depth	0.7 mm
Slit angle	$30 \div 150$ deg	Back iron width	32 mm

suboptimal geometries shown in Fig. 2 analyzed for DL training. This approach allows the features leading to both positive and negative results to be learned by the DL CNNs, hence accurately analyzing a large variety of low- and high-torque IPMSM rotor geometries.

B. FEA model parameters

FEA utilizes numerical methods, hence its accuracy must be considered when preparing the training data. To ensure low computational errors and reasonable analysis time, a mesh study was conducted resulting in 6300 element (3700 nodes) mesh and analysis time of 2 s per step, 91 steps per geometry to obtain smooth torque curves. Thus, the analysis time per model was approximately 3 min. Torque curves can be used to calculate maximum torque, average torque, and torque ripple but provide additional essential information about harmonic spectrum of the torque and its distribution, which can be extremely valuable for IPMSM rotor design.

It should be mentioned that unlike the models used in this study, an extremely high accuracy analysis may require hours for a single model to converge. However, even 3 min per model still yields 50 h of computational time per 1,000 models, and 200 h to analyze the 4,000 sample dataset used in this study not including the FEA model setup time, loading time, exporting and interpreting the results, and other ancillary but necessary operations. This illustrates the difficulties associated with collecting large datasets for DL electrical engineering applications.

Fig. 3. Preparing rotor images for CNN training by converting to grayscale and resizing JMAG Designer exported rotor geometries.

III. PREPARING THE TRAINING DATA

A. Arranging rotor geometry images and torque curves for DL training

DL CNNs take advantage from training on large datasets and deep models are often characterized by extremely high number of parameters and layers making their implementation computationally demanding. In this study the experiments are conducted with smaller input image size and depth to reduce hardware requirements and training time while ensuring that the accuracy of input data representation is not compromised.

With JMAG Designer, it is possible to export rotor geometries as RGB images with default resolution of approximately 180×180 pixels. However, this resolution was found suboptimal due to extensively large size and no information carried by color channels. Thus, the input images are resized to 56×56 pixel resolution and converted to grayscale as shown in Fig. 3. The resolution was also chosen to be a divisible of default input of ResNets to apply the latter without modifying the filters.

Pytorch 1.4.0 is used in this study for DL CNN implementation, training, and data preparation [23]. The, grayscale rotor geometry images are first converted to numpy arrays and paired with corresponding output torque data [24]. Data normalization and torque clipping discussed in the next subsection are also performed on this stage. The dataset is later converted to Pytorch tensors for DL model training. Numpy arrays and Pytorch tensors with the dataset prepared for training can be exported and saved for later use by various CNN models [25].

B. Min-max and offset torque normalization techniques

In electric machines the output torque range can be extremely large, especially for diverse datasets with negative examples contributing to very low torque values as well as near-optimal geometries with high torques. This is drastically different from DL classification where binary 0 or 1 outputs have to be predicted.

Because predicting the exact numerical values is required for electrical machine analysis, it corresponds to a regression problem. Whereas it is in principle possible to predict any numerical values with CNNs, they perform better when an arbitrary range of outputs (i.e., torques τ) is normalized using, for instance, min-max normalization [26]

$$\tau_{norm_i} = \frac{\tau_i - \tau_{min}}{\tau_{max} - \tau_{min}}, \quad (1)$$

where τ_{norm} is normalized torque, and τ_{min} and τ_{max} are minimum and maximum torque values in the dataset, respectively. This leads to all normalized torque τ_{norm} values being within 0 – 1 region.

However, an additional challenge was encountered when DL CNNs were applied to torque prediction. For classification, a softmax layer is applied when calculating model outputs. However, in case of torque calculation the exact numerical values in output layers of CNN are compared with dataset torques. It was noticed that predicting a large data range with high precision was challenging, as shown in Section V. This might be due to disproportional contribution of smaller values to relative error metric.

However, extremely small torque values are unimportant because they correspond to negative examples. Furthermore, it is desired to predict large outputs of optimal geometries with the highest possible accuracy. Thus, the use of a different normalization that reduces the effect of low torque samples on DL training can be proposed. An offset in normalization range can be introduced so that

$$\tau_{o.norm_i} = \frac{\tau_i - \tau_{min}}{\tau_{max} - \tau_{min}} + a_n, \quad (2)$$

where a_n is normalization offset parameter so that $\tau_{o.norm} \in [a_n, 1 + a_n]$. Using (2) essentially leads to all torques $\tau < a_n$ being treated in exactly the same manner, hence removing the strain of predicting their exact values accurately. The normalized outputs predicted by DL CNN can then be recalculated into real ones as

$$\tau_{real_i} = (\tau_{o.norm_i} - a_n)(\tau_{max} - \tau_{min}) + \tau_{min}. \quad (3)$$

It was observed that training DL CNNs on the data normalized using (2) led to higher accuracy and faster convergence. It should be mentioned that using (2) results in additional errors for values close to a_n when calculating real values using (3). Hence, a_n should be chosen so errors are large only in regions where high accuracy is not required. Choosing small a_n does not result in error increase for average and large torque samples.

Alternatively, the dataset can be clipped for low torques so that

$$\tau_{c.norm_i} = \begin{cases} \delta_c, & \text{if } \tau_i < \delta_c(\tau_{max} - \tau_{min}) + \tau_{min} \\ \tau_{norm_i}, & \text{else} \end{cases} \quad (4)$$

where δ_c is threshold parameter. The effect of δ_c is similar to the effect of a_n , so accuracy at extremely low torques is discarded to ease training for the rest of data range. However, no additional error is introduced for $\tau_{c.norm_i} > \delta_c$ unlike in case of offset normalization, making torque clipping a better option.

Fig. 4. Architecture of the Simplistic CNN with only 5 layers and less than 1 million parameters.

C. Dataset balancing for training and testing

DL CNNs can potentially overfit the training data leading to the model performing well on the training set, but failing on any example outside the initial dataset. Thus, the complete dataset is divided into training, validation, and test sets. The majority of available data is usually allocated for training. In this study, 80% of the dataset is training data. In case training data is limited, the validation and training sets can be merged which applies to 4,000 sample dataset used in this study. Thus, 20% of dataset is allocated for validation and validation accuracy is used to calculate accuracy metrics.

Before dividing into training and validation, the dataset is shuffled to avoid clustering of similar examples. Both training and validation sets are checked for balance, meaning that all types of data (low, average, high torque samples) are equally represented to ensure unbiased training and validation.

IV. DL CNN ARCHITECTURES AND TRAINING PARAMETERS

Electrical machine researchers previously used very deep architectures such as GoogLeNet [13], [27], and even VGG 16 that has 138 m parameters [14], [28]. However, smaller custom architectures were also used successfully [22]. Thus, it is currently hard to determine the optimal CNN architecture required for electrical machine design projects.

To address this problem, we compare a custom simplistic low-demand CNN with different ResNet architectures. The torque prediction accuracies are compared along with size, complexity, and hardware requirements of the architectures.

A. Custom simplistic architecture

In this study a simplistic CNN shown in Fig. 4 is used to evaluate the possibility of training shallow networks for electric machine design projects. Table II shows that this CNN consists of 3 convolutional and 2 fully connected (linear) layers and has less than 1 million parameters. The ratio of the number of channels in convolutional and linear layers mimics the intermediate layers of AlexNet [29]. This architecture was inspired by the success of shallow networks on image classification tasks.

TABLE II
COMPARISON OF PARAMETERS OF STUDIED DL CNN ARCHITECTURES

Characteristic	Simplistic CNN	ResNet-10	ResNet-18	ResNet-50
ResNet blocks	-	Basic, [1,1,1,1]	Basic, [2,2,2,2]	Bottleneck, [3,4,6,3]
Conv. layers	3	9	17	49
Total layers	5	10	18	50
Num. parameters	0.89m	5.8m	11.4m	23.9m

The advantage of using shallow CNNs is fast training and low memory demands. A small size of this architecture allows training to be performed reasonably fast even on CPU, unlike deeper architectures for which CPU training takes unreasonably long and GPU training is performed instead.

B. ResNets

ResNets [30] are deep residual CNNs that were developed to explore the relation between the depth and image classification accuracy inspired by the evidence of accuracy degradation for extremely deep architectures [31]. ResNets introduced bypass connections and trained network to learn desired change of activations instead of activations themselves. The architectures consist of blocks of convolutional layers with block types and their number specifying particular architectures. All ResNets have one initial convolutional layer and one linear output layer, with aforementioned blocks of convolutional layers allocated between them. The softmax operation on the output layer used by classifiers is removed for regression problems.

ResNet-50 and more complex architectures utilize four bottleneck-type blocks with two identity mappings surrounding a 3×3 convolution. The ResNet-50 parameters can be encoded as [3, 4, 6, 3] with each number corresponding to the number of blocks of each type. Table II shows that ResNet-50 has 50 layers of which 49 are convolutional. This is the deepest architecture among the architectures studied in this paper with almost 24 million parameters.

Less deep ResNet architectures use basic blocks with two 3×3 convolutions per block. Table II shows that ResNet-18 utilizes two blocks of each type and has roughly half the ResNet-50 parameters. In addition, we consider even shallower architecture with only one block per type, resulting in the total of 10 layers, labeled ResNet-10.

C. DL CNN training parameters

CNN training, as training of any artificial neural network, is an optimization process where errors calculated during forward pass are backpropagated to update the network weights and minimize errors. Adam optimizer is used for all experiments in this study [32].

There are several important hyperparameters that affect the training. First, learning rate determines the step size in optimization process. On one hand, large learning rate facilitates faster training but may result in oscillations and divergence leading to unstable training. On the other hand, small learning rate may lead to very slow training or solutions stuck at saddle points. Thus, it is often recommended to monitor variation in training and validation losses and update

Fig. 5. Example of CNN outputs in low-torque region. All architectures show similar accuracy.

the learning rate respectively. This can be done manually or automated using learning rate decay [33].

For most experiments on ResNets, a common 0.01 initial learning rate and 0.1 learning rate multiplier were found effective. However, for the Simplistic CNN the latter settings often resulted in poor convergence. A 0.5 initial learning rate and 0.2 learning rate multiplier were empirically found to perform better.

Second, loss function is used for error computation and CNN parameter adjustment during training. For regression problems, least absolute deviation (L1) and least square error (L2) loss functions are primarily used. In this study nearly no difference was observed after comparing the two loss functions. However, L2 loss can be recommended because errors for high torques gain more weight compared to L1 loss, which is desired for accurate prediction in high-torque region.

Training CNNs can be very computationally expensive and requires high-end hardware. However, recent improvements in GPU training allowed fast training on affordable machines, or on GPU clusters for extremely large datasets. This is particularly relevant for CUDA-accelerated GPUs [34]. In this study a NVIDIA GeForce GTX 1660 Ti GPU and CUDA 11.0 were used for all experiment.

The GPU memory should still be carefully managed when working with large datasets. Thus, training data is passed to the CNN using mini-batches, and mini-batch size is the third important hyperparameter. In Pytorch, a dataloader can also be used to automate the dataset splitting. In this study, a mini-batch size of 64 was used for all experiments. Whereas more shallow architectures allow mini-batch size to be increased, it was kept constant for clarity of comparison. Experiments on the Simplistic CNN were also conducted using batch sizes up to 256 which showed no significant deviation.

All CNNs were trained over $100 \div 180$ epochs with termination point determined by the decay in the learning rate. As might be expected, training deeper CNNs took longer, with ResNet-50 training taking almost 6 times longer than the training of the Simplistic CNN.

TABLE III
A₁ ACCURACY OF PREDICTING AVERAGE TORQUE AND TORQUE CURVES USING CONVENTIONAL AND CUSTOM NORMALIZATION

CNN architecture	Average τ , conv. norm.	Average τ , cust. norm.	τ curve, conv. norm	τ curve, cust. norm
Simplistic CNN	20	29	34	64
ResNet-10	29	58	59	89
ResNet-50	36	60	62	93

D. Accuracy metrics for regression

Because training and validation losses are hard to interpret, the accuracy is calculated for final result interpretation using specific accuracy metrics. For classification problems, the percentage of correct predictions is the most commonly used accuracy metric. However, this is different for regression problems where purpose-driven metrics are needed. Whereas a single comprehensive accuracy metric for electrical machine design problems is hard to propose, analyzing multiple metrics simultaneously can provide vital information about DL CNN performance.

First, mean average percentage error (MAPE) can be used [22], [35] so that

$$M = \frac{1}{n} \sum_{i=1}^n \left| \frac{\tau_{exp_i} - \tau_{pred_i}}{\tau_{exp_i}} \right|, \quad (5)$$

where n is dataset size, τ_{exp} and τ_{pred} are expected and predicted torque values, respectively. The accuracy is calculated using (5) as

$$A_1 = 1 - M. \quad (6)$$

Second, the accuracy of predicting the results within an error margin can be calculated as

$$A_2 = \frac{\sum_{i=1}^n C_i}{n}, \quad (7)$$

where

$$C_i = \begin{cases} 1, & \text{if } \left| \frac{\tau_{exp_i} - \tau_{pred_i}}{\tau_{exp_i}} \right| < \delta_m \\ 0 & \text{else} \end{cases} \quad (8)$$

and δ_m is error margin.

The first metric A_1 gives a more general overview and is often used to analyze the accuracy of conventional modeling techniques. However, for a large dataset it can give misleading results when errors are localized in a particular part of the dataset. It will be shown in the next Section that A_1 and A_2 can give considerably different results, yet combined provide some important insight into the accuracy variation throughout the dataset.

V. RESULTS AND DISCUSSION

A. Comparison of accuracy of average and torque curve prediction by different CNNs

In order to determine how dimensions of the output affect the CNN training, the Simplistic CNN along with ResNet-10

Fig. 6. Example of CNN outputs in low-mid torque region. Whereas ResNet 18 and ResNet 50 show good agreement with FEA, ResNet10 does not capture the curve shape correctly, and the Simplistic CNN predicts almost 50% lower torque.

Fig. 7. Example of CNN outputs in mid-torque region. The error in ResNet10 prediction increases, and the Simplistic CNN gives a completely incorrect estimation.

and ResNet-50 were trained to predict torque curves and average torques. For the latter experiment, the dimension of output layer was reduced from 91 to 1. The effects of the min-max and offset normalization were also studied by training the CNNs on differently normalized datasets.

The analysis results are summarized in Table III with accuracy calculated using (6). The experiments conducted on conventional (min-max) normalized datasets are labeled “conv. norm.” in Table III. It should be noted that the offset normalization and torque clipping discussed in Section III had nearly identical effects on CNN training, so “cust. norm.” results in Table III are relevant for both techniques.

Table III shows that all CNNs performed better when trained to predict torque curves rather than average torques. This can be explained by multidimensionality of torque curves providing CNNs with more information to learn the features from. This aligns with multi-task learning approach where training on several similar tasks in some cases provides better performance than training for a single task [36]–[38], or single torque in case of this study. Thus, training CNNs to predict

Fig. 8. Example of CNN outputs in high-torque region. ResNet 18 and 50 show high accuracy, ResNet 10 has some error in magnitude and shape predictions, and the Simplistic CNN has even higher errors than ResNet10.

output curves, or any type of distributed parameters or even different parameters types, is easier compared to predicting average values.

Furthermore, all models performed better when offset normalization (or torque clipping) was used. This can be attributed to the reduction in the variations in output torque. On one hand, Table III shows that accuracy of min-max normalized data prediction was gradually increasing with CNN depth, and it might be expected that deeper architectures with more parameters could perform better on such datasets. However, the accuracy exhibits saturation for extremely deep networks [30]. Furthermore, deeper networks are associated with more computational demands, longer training and higher costs. Hence, introducing a meaningful threshold in the torque range can reduce the analysis costs by making CNN training faster and less dependent on the architecture depth.

B. Accuracy of torque curve prediction

The accuracy of different CNNs predicting torques curves is summarized in Table IV. Accuracy metrics are applied to the complete dataset, and also exclusively to the high-torque samples. The latter allows the CNN performance in the region of interest to be evaluated with higher precision.

First, Table IV shows that the Simplistic CNN could not provide sufficient prediction accuracy. Whereas its overall MAPE accuracy is over 60%, it performs extremely poorly in mid- and high-torque ranges, as illustrated by other metrics. Fig. 5 shows all models performing similarly on a low-torque sample, implying that the Simplistic CNN could successfully identify negative examples by predicting low torques and distinguishing such geometries from average-torque ones shown in Figs. 6 and 8. However, Fig. 7 shows that sometimes the Simplistic CNN was giving completely incorrect predictions. Thus, whereas some success in identifying positive and negative examples qualitatively is shown by this architecture, it cannot be recommended for quantitative analysis.

All ResNet architectures performed well on the dataset, showing MAPE accuracy of 88% or higher. However, some

TABLE IV
ACCURACY OF TORQUE CURVE PREDICTION BY DIFFERENT DL MODELS

Accuracy metric	Simplistic CNN	ResNet-10	ResNet-18	ResNet-50
A_1^1	64	88	90	93
A_1^2	30	86	92	96
A_2 , 20% margin ¹	24	82	89	92.5
A_2 , 20% margin ²	10	67	74	86
A_2 , 10% margin ¹	0	63	81	90
A_2 , 10% margin ²	0	57	66	77

¹Complete dataset, ²High torques only

deviations illustrated by Figs. 6 – 8 were observed. It can be noticed that ResNet-10 tends to predict the torque magnitude reasonably well, but does not capture the shapes of the torque curves. This implies that the ResNet-10 parameters are not sufficient to capture all torque variations in the dataset, yet those can be crucial for electric machine performance analysis. Low performance of ResNet-10 on error margin accuracy metrics also suggests that it tends to predict average values over groups of input samples, without further refinement into special features of particular geometries and their effects on the output torque.

On the contrary, ResNet-18 and ResNet-50 showed good agreement with FEA by capturing magnitudes and shapes of torque curves. As might be expected, ResNet-50 outperformed ResNet 18, but the latter still shows almost 90% accuracy on 20% error margin metric on the complete dataset, and about 80% accuracy on the 10% one.

It thus can be concluded that deep CNNs are capable of predicting output torque curves with high accuracy. The accuracy is proportional to CNN depth, and for sufficiently deep CNNs the number of CNN parameters correlates with the accuracy of curve shape prediction. However, increasing the depth indefinitely does not result in accuracy increase. This is shown by the similar performance of ResNet-18 and ResNet-50, and can be extrapolated to deeper CNNs.

C. Analysis time of CNNs and FEA

Whereas ResNets show accuracy comparable to FEA, the analysis (torque prediction) time is significantly reduced when using CNNs, as shown by Table V. Evaluating the complete dataset even with ResNet-50 takes less than 1 min whereas FEA requires 200 h of computations. Whereas the used FEA models were relatively compact, an extremely high-accuracy FEA may take 1 h or more, further escalating the difference between FEA and CNN prediction times. Training CNNs on high-accuracy FEA data does not affect CNN training time.

Because of high speed of CNN analysis, IPMSM rotor geometry optimization can be performed extremely fast when FEA is substituted for a DL tool and optimization algorithms are applied. Whereas DL CNNs were previously used to speed-up the optimization while final results were obtained using FEA [12], [14], this paper shows the possibility of accurately evaluating rotor geometries using DL CNNs. Thus, the previously reported DL-accelerated optimization results can be further improved by developing a complete FEA-free end-to-end DL analysis method.

TABLE V
ANALYSIS TIME OF DIFFERENT CNN ARCHITECTURES AND FEA

Architecture	Single sample	Complete dataset
Simplistic CNN	1.2 ms	5 s
ResNet-10	6.2 ms	25 s
ResNet-18	10.5 ms	42 s
ResNet-50	14.5 ms	58 s
JMAG FEA	180 s	200 h

D. Inheriting computational errors from training data

Whereas CNN performance is compared with FEA in this study, FEA uses computational techniques that are not free of errors. These include discretization and numerical errors along with potential misrepresentation of some electric machine parameters in a model. Furthermore, there are additional errors that become apparent on prototype verification stage, e.g. the effects of manufacturing tolerances. Thus, training DL CNNs on FEA data makes the DL model inherit these potential errors.

Inheriting computational errors can be avoided if experimental data is used for DL CNN training. However, obtaining large experimental datasets is extremely challenging. For instance, all various rotors geometries must be fabricated and tested implying great expenses.

Thus, using FEA or another high-accuracy analysis tool can be recommended when numerous geometry variations have to be studied. However, multiple experiments can be conducted on a single prototype to obtain output characteristics as function of excitation and use to train DL CNNs. Furthermore, it is possible to combine different types of data to device a unique analysis tool with CNNs.

VI. CONCLUSION

This study develops an end-to-end DL-assisted pipeline for analysis of electric machines. Four DL CNNs were trained to predict output torque curves of IPMSM showing over 90% prediction accuracy. Predicting curves, i.e. high dimensional outputs, is advantageous because more information about machine performance can be obtained compared to a single value average torque prediction. It also provides more data for CNN training further increasing prediction accuracy. The comparison of DL architectures shows that a certain degree of complexity in CNN depth is required for accurate torque prediction, and for sufficiently deep networks further increase in depth allows torque curve shapes to be represented more accurately, with accuracy gradually saturating for extremely deep CNNs. The choice of a normalization technique and effects of dataset balancing on CNN training and torque prediction are studied and possible recommendations are given. Upon training, CNNs performed extremely fast allowing more than 10,000 times faster analysis compared to FEA. The possibility of predicting multidimensional outputs facilitates DL-assisted multiparameter optimization of electrical machines showing its potential for a major future analysis and design method.

REFERENCES

- [1] H. I. Lin, "Learning on robot skills: Motion adjustment and smooth concatenation of motion blocks," *Eng. Appl. Artif. Intell.*, vol. 91, no. April 2019, art. no. 103619, 2020, doi: 10.1016/j.engappai.2020.103619.
- [2] O. Fink, Q. Wang, M. Svensén, P. Dersin, W.-J. Lee, and M. Ducoffe, "Potential, challenges and future directions for deep learning in prognostics and health management applications," *Eng. Appl. Artif. Intell.*, vol. 92, no. May, art. no. 103678, 2020, doi: 10.1016/j.engappai.2020.103678.
- [3] C. Bousquet-Jette, S. Achiche, D. Beaini, Y. S. Law-Kam Cio, C. Leblond-Ménard, and M. Raison, "Fast scene analysis using vision and artificial intelligence for object prehension by an assistive robot," *Eng. Appl. Artif. Intell.*, vol. 63, pp. 33–44, 2017, doi: <https://doi.org/10.1016/j.engappai.2017.04.015>.
- [4] P. Mehta *et al.*, "A high-bias, low-variance introduction to Machine Learning for physicists," *Phys. Rep.*, vol. 810, pp. 1–124, 2019, doi: 10.1016/j.physrep.2019.03.001.
- [5] J. Yin and W. Zhao, "Fault diagnosis network design for vehicle on-board equipments of high-speed railway: A deep learning approach," *Eng. Appl. Artif. Intell.*, vol. 56, pp. 250–259, 2016, doi: 10.1016/j.engappai.2016.10.002.
- [6] Y. Dong, "Implementing Deep Learning for comprehensive aircraft icing and actuator/sensor fault detection/identification," *Eng. Appl. Artif. Intell.*, vol. 83, pp. 28–44, 2019, doi: 10.1016/j.engappai.2019.04.010.
- [7] H. J. Carpenter, A. Gholipour, M. H. Ghayesh, A. C. Zander, and P. J. Psaltis, "A review on the biomechanics of coronary arteries," *Int. J. Eng. Sci.*, vol. 147, art. no. 103201, 2020, doi: 10.1016/j.iengsci.2019.103201.
- [8] A. Khan, M. H. Mohammadi, V. Ghorbanian, and D. Lowther, "Efficiency Map Prediction of Motor Drives Using Deep Learning," *IEEE Trans. Magn.*, vol. 56, no. 3, art. no. 7511504, 2020, doi: 10.1109/TMAG.2019.2957162.
- [9] M. A. Omari, A. Almagableh, I. Sevostianov, M. S. Ashhab, and A. B. Yaseen, "Modeling of the viscoelastic properties of thermoset vinyl ester nanocomposite using artificial neural network," *Int. J. Eng. Sci.*, vol. 150, art. no. 103242, 2020, doi: 10.1016/j.iengsci.2020.103242.
- [10] L. Liang, M. Liu, C. Martin, and W. Sun, "A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis," *J. R. Soc. Interface*, vol. 15, no. 138, p. 20170844, 2018, doi: 10.1098/rsif.2017.0844.
- [11] A. Khan, V. Ghorbanian, and D. Lowther, "Deep learning for magnetic field estimation," *IEEE Trans. Magn.*, vol. 55, no. 6, art. no. 7202304, 2019, doi: 10.1109/TMAG.2019.2899304.
- [12] H. Sasaki and H. Igarashi, "Topology optimization accelerated by deep learning," *IEEE Trans. Magn.*, vol. 55, no. 6, art. no. 7401305, 2019, doi: 10.1109/TMAG.2019.2901906.
- [13] S. Doi, H. Sasaki, and H. Igarashi, "Multi-objective topology optimization of rotating machines using deep learning," *IEEE Trans. Magn.*, vol. 55, no. 6, art. no. 7202605, 2019, doi: 10.1109/TMAG.2019.2899934.
- [14] J. Asanuma, S. Doi, and H. Igarashi, "Transfer Learning through Deep Learning: Application to Topology Optimization of Electric Motor," *IEEE Trans. Magn.*, vol. 56, no. 3, art. no. 7512404, 2020, doi: 10.1109/TMAG.2019.2956849.
- [15] F. L. J. Deng, W. Dong, R. Socher, L. Li, K. Li, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.
- [16] Google Open Source, "Open Images Dataset," 2020. [Online] Available: <https://open-source.google/projects/open-images-dataset> (accessed May 22, 2020).
- [17] A. G. Kusne *et al.*, "On-the-fly machine-learning for high-throughput experiments: Search for rare-earth-free permanent magnets," *Sci. Rep.*, vol. 4, p. 6367, 2014, doi: 10.1038/srep06367.
- [18] M. Gusenbauer *et al.*, "Extracting local switching fields in permanent magnets using machine learning," pp. 1–21, 2019, [Online]. Available: <http://arxiv.org/abs/1910.09279>.
- [19] L. Exl *et al.*, "Magnetic microstructure machine learning analysis," *J. Phys. Mater.*, vol. 2, no. 1, art. no. 014001, 2018, doi: 10.1088/2515-7639/aaf26d.
- [20] T. C. Kwang, M. L. M. Jamil, and A. Jidin, "Torque improvement of PM motor with semi-cycle stator design using 2D-finite element analysis," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 6, pp. 5060–5067, 2019, doi: 10.11591/ijece.v9i6.pp5060-5067.
- [21] JSOL Corporation, "JMAG Designer," 2020. [Online]. Available: <https://www.jmag-international.com/products/jmag-designer/> (accessed Jun. 04, 2020).
- [22] S. Barmada, N. Fontana, L. Sani, D. Thomopoulos, and M. Tucci, "Deep Learning and Reduced Models for Fast Optimization in Electromagnetics," *IEEE Trans. Magn.*, vol. 56, no. 3, art. no. 7513604, 2020, doi: 10.1109/TMAG.2019.2957197.
- [23] Pytorch, "PyTorch - an open source machine learning framework," 2020. [Online]. Available: <https://pytorch.org/> (accessed May 26, 2020).
- [24] Numpy, "Numpy - open source numerical computation library," 2020. [Online]. Available: <https://numpy.org/> (accessed May 28, 2020).
- [25] N. Gabbdullin, S. Madanzadeh, and A. Vilkin, "IPMSM rotor geometries and torque curves training data for DL CNNs," *Mendeley Data*, V1, 2020. [Online]. Available: doi.org/10.17632/j7bbrmmn4k.1
- [26] J. Grus, *Data Science from Scratch*. Sebastopol, CA: O'Reilly, 2015.
- [27] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.
- [28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, pp. 1–14, 2015.
- [29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012, pp. 1–9, doi: 10.1145/3065386.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
- [31] K. He and J. Sun, "Convolutional Neural Networks at Constrained Time Cost," in *CVPR*, 2015, pp. 5353–5360.
- [32] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *ICLR*, pp. 1–15, 2015.
- [33] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.
- [34] Nvidia Corporation, "CUDA - a parallel computing platform," 2020. [Online]. Available: <https://developer.nvidia.com/cuda-zone> (accessed May 31, 2020).
- [35] A. De Myttenaere *et al.*, "Mean Absolute Percentage Error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, 2016, doi: 10.1016/j.neucom.2015.12.114.
- [36] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," pp. 1–14, 2017, [Online]. Available: <http://arxiv.org/abs/1706.05098>.
- [37] Z. Tan *et al.*, "Combined electricity-heat-cooling-gas load forecasting model for integrated energy system based on multi-task learning and least square support vector machine," *J. Clean. Prod.*, vol. 248, art. no. 119252, 2020, doi: <https://doi.org/10.1016/j.jclepro.2019.119252>.
- [38] M. Li, D. Chen, S. Liu, and F. Liu, "Grain boundary detection and second phase segmentation based on multi-task learning and generative adversarial network," *Measurement*, vol. 162, art. no. 107857, 2020, doi: <https://doi.org/10.1016/j.measurement.2020.107857>.