Transfer Learning-Based Modular Neural Network for Multi-Objective Optimization of Interior Permanent Magnet Synchronous Motors

Nuo Chen
Electrotechnical Institute
Karlsruhe Institute of Technology
Karlsruhe, Germany
nuo.chen@kit.edu

Martin Doppelbauer
Electrotechnical Institute
Karlsruhe Institute of Technology
Karlsruhe, Germany
martin.doppelbauer@kit.edu

***Abstract*—Finite element analysis is frequently used to optimize the characteristics of interior permanent magnet synchronous motors throughout the design phase. The existing toolchains enable the full automation of simulating and optimizing a reference motor by manipulating the input design parameters within the feasible design space. However, for each motor design, a complete simulation is required, implying a high computational burden and time cost. Moreover, once the input design parameters undergo variations, it becomes necessary to initiate the simulation process from the beginning. The previously obtained simulation results are not helpful for the new task. In this paper, a new method using modular neural networks based on transfer learning (TL) under dimensionally varying input space conditions is presented. By transferring certain parts of the pre-trained neural networks (NNs) of the old task to the new task's NN, the previously learned parameters can be applied as the initial weight for the new network. Finally, the optimization process is completed by combining this approach with multi-objective optimization. The results show that the learning of the new NN is promoted with the help of TL. In addition, highly flexible surrogate models are achieved, enabling accurate prediction capabilities and a fast optimization time of around 12 seconds.**

***Index Terms*—permanent magnet motor, finite element analysis, modular neural network, transfer learning, multi-objective optimization.**

I. Introduction

Over the past few years, there has been an increasing utilization of interior permanent magnet synchronous motors (IPMSMs) in various domains due to several advantageous characteristics, including high efficiency, compact construction, and a broad range of operating speeds [1], [2]. Nevertheless, the motor optimization procedure in the design phase is time-consuming, caused by the high complexity of the input design parameters and the extensive computational demands of the Finite element analysis (FEA). Hence, many researchers adopt surrogate models to optimize electric motors [3]. At present, neural networks (NNs) are being progressively employed as surrogate models with the aim of forecasting the performance parameters of electric motors [4], [5]. A previous investigation applied a convolutional neural network (CNN) to estimate average torque and torque ripple, resulting in a significant acceleration of the optimization procedure with guaranteed accuracy [6]. In [7], a CNN and a recurrent neural network were employed in combination to handle the geometric and operating point variations. At the end, the predictive efficiency map of the electrical machine is generated with this approach. In [8], the exploration and comparison of image-based NNs and parameter-based NNs were undertaken. It was demonstrated that the image-based approach was more flexible in the case of reparametrization. In contrast, parameter-based networks performed better in terms of prediction accuracy and had a lower computational burden. However, managing various topologies is a challenge. In order to address this issue, the paper [9] presented a surrogate model based on the variational autoencoder (VAE), which mapped a high-dimensional design parameters in a lower-dimensional latent space. This approach enabled simultaneous optimization of multiple topologies. In addition, a recent study [10] combined the VAE and NNs with dropout to achieve faster optimization. The VAE was applied to encode different motor shapes into the latent space. However, the aforementioned methods involved a large-scale prediction model and a huge amount of data. To solve this problem, an automatic design system integrated with the generative adversarial network (GAN) and CNN was proposed in [11]. A applied machine learning method enabled the rapid acquisition of an enormous amount of data through the use of a small number of FE results. The proposed GAN was used to design rotor topologies for three different IPMSMs in latent space. Additionally, CNN demonstrated the capability to quickly and precisely predict motor characteristics.

So far, the inputs for NN-based surrogate models can be categorized into three groups: parameter-based (1D vector) [8]–[10], image-based (2D or 3D matrix) [6], [8], [11], and mixed input, which includes both vectors and images [7]. In real-world scenarios, parameter-based datasets are more accessible to obtain without additional storage of images. However, when the reference motor is reparameterized, the NN that was previously trained becomes incompatible for further usage due to alterations in the feature composition of parameter vectors. Retraining new NNs from scratch is demanded for every alteration, thereby leading to the discarding of previously trained parameters.

In this study, we propose parameter-based modular neural networks (MNNs) through transfer learning (TL) to get over the issue of the networks requiring to be retrained from scratch after reparameterization. Transfer learning is an important technique in the field of machine learning that facilitates the training of NNs by transferring previously learned knowledge to a new task [12]. One of the most common ways to reuse knowledge is through the transfer of NNs. This method makes it possible to save part of the old networks' weights and use them to help the new NN update its weights [13]. In the field of design optimization of electrical motors, transfer learning-based CNN has been employed in [14]. An advantage of this proposed approach was that the CNN was effectively trained with a limited amount of input data using TL. Nevertheless, it relied on images but failed to address the issue of reparameterization.

II. Methodology and Datasets

A. Modular Neural Network and Proposed Workflow

Assume we have a task $\mathcal{T}_t$ in the target domain $\mathcal{D}_t$, and an assistance can be obtained from a source task $\mathcal{T}_s$ coming from the source domain $\mathcal{D}_s$. By transferring latent knowledge from $\mathcal{T}_s$ to the target task $\mathcal{T}_t$, TL intends to boost the performance of the prediction function $\mathcal{F}_t$, where $\mathcal{F}_t$ is a representation of a deep NN [12]. A workflow example for the proposed method is depicted in Fig. 1. Assuming an initial learning task based on three input design parameters (d_1, d_2, d_3) and four output performance parameters (o_1, o_2, o_3, o_4) is given. Subsequently, the input space is expanded, and the whole process is described as follows:

1) *Stage I*: A separate training session is performed for each of the input design parameters. Every individual multi-layer NN is called a base module, which consists of m dense layers with n neurons in each layer.
2) *Stage II*: All the base modules are subsequently aggregated to the target NN by transferring the respective $m-1$ dense layers, namely the layers with colors. A multi-branched MNN was applied, with each branch representing an independent input parameter. The quantity of input design parameters directly matches the quantity of branches in the target MNN.
3) *Stage III*: The input space is altered by eliminating the design parameter d_3 and introducing the new design parameters d_4 and d_5. As the input space expands, the NN's architecture goes through an adjustment whereby the layers associated with the branches of d_1 and d_2 stay unchanged while the branch linked to d_3 is removed. Another two basic modules (pink and purple) with d_4 and d_5 as inputs are trained independently to provide pre-trained models resembling Stage I. The required layers are then transferred to various branches. Only the first and last dense layers (white) necessitate retraining from scratch.

A classic MNN consists of three parts: the task-dividing layer, sub-network layer, and output-combining layer [15].

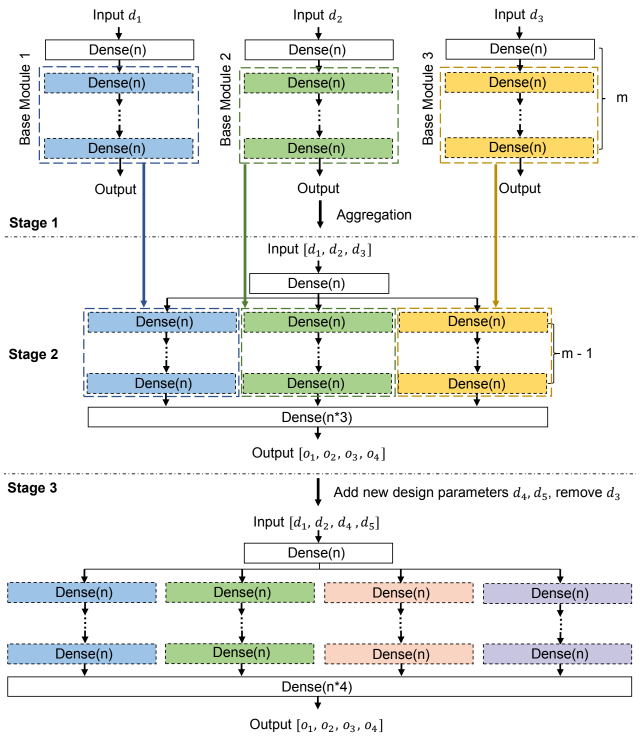

Fig. 1. Proposed workflow for aggregating pre-trained NNs.

The idea of modularity is derived from biological inspiration. Within a biological brain, distinct information is typically processed in designated regions to carry out certain activities [16]. In [17], [18], modularity-based ideas were applied, and images were divided into segments and supplied to NNs for processing. Inspired by the above approaches, every input design parameter can be regarded as a distinct sub-task that corresponds to a branch in our proposed MNN. Through this method, the target MNN exhibited the capability to adapt in response to variations in the input parameters. Meanwhile, the knowledge learned from the previous task can be transferred to the new task.

B. Dataset Generation

To conduct experiments based on the proposed workflow, we created three main datasets that have slightly different input design parameters and the same outputs. The sampling procedure was carried out using Latin hypercube sampling (LHS) [19] by a simulation toolchain based on Altair Flux. In the first step, each sample model was built, and all unfeasible designs were eliminated. Then we performed FEA based on 36 different current vectors to obtain the current-dependent maps. Finally, we further acquired the speed-torque maps by maximum torque per ampere optimization and obtained the performance indicators for each design. The reference motor is depicted in Fig. 2. Dataset I contains six design parameters (d_1, d_2, d_3, d_4, d_5, d_6) and a total of 968 samples. Dataset II is formed by removing one parameter from dataset I and adding three new design parameters. As a result, dataset II includes

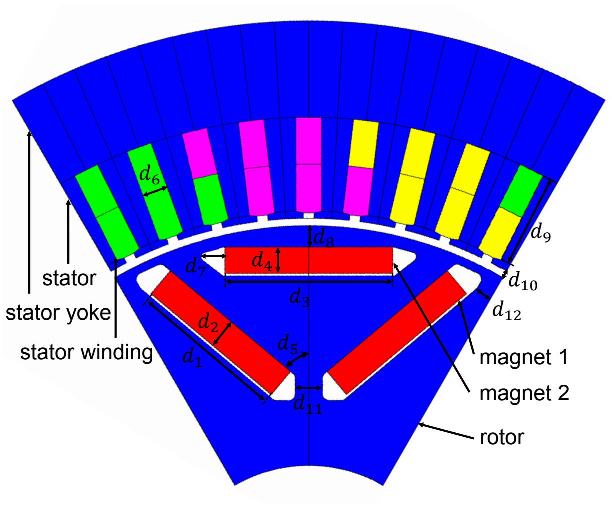

Fig. 2. Reference IPMSM geometry and design parameters.

eight design parameters (d_1, d_2, d_3, d_4, d_5, d_8, d_{11}, d_{12}) and 2000 samples. Dataset III has 2951 samples and consists of ten design parameters (d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}).

In the following, we only analyze dataset III. Table I summarizes the details of all parameters. The upper and lower boundaries for each design parameter are determined empirically. The Pearson correlation coefficients among all parameters can be calculated by:

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \qquad (1)$$

where x_i and y_i denote different values of variables in a sample. $\bar{x}$ and $\bar{y}$ represent sample means. The results are visualized in Fig. 3. The correlation between the design input parameters was generally low, except for the width of magnet 2 and the V-magnet angle. This is due to restrictions on their feasible domains. The design parameters exhibited a strong correlation with the indicators, suggesting a good sampling process.

In order to obtain pre-trained NNs, we performed separate experimental designs for each design parameter individually. A group of small subdatasets was then generated, each containing around 50 samples. Fig. 4 demonstrates rotor images in the subdataset for the design parameters d_1, d_3, d_4, and d_5.

III. Training of Neural Networks

A. General Settings

All the data were standardized by their sample mean and variance before being fed to the NNs. All experiments, including model comparison, were performed utilizing 5-fold cross-validation, with loss computed by the mean squared error (MSE) between prediction and ground truth. The entire dataset was divided into five equal parts. We took four parts as training data and one part for testing. Training was then conducted five times. It should be noted that the model in III-C was analyzed by selecting the fold with the lowest loss in cross-validation for the results. To guarantee that comparisons between various models were based on the same dataset divisions and to make the findings reproducible, random seeds were set. In addition, the early stopping method was implemented during the training process to avoid overfitting, and its patience was set to 10. In order to evaluate the quality of predictions, three different evaluation metrics were employed: the mean absolute error (MAE), symmetric mean absolute percentage error (SMAPE), and coefficient of determination (R^2). The MAE enables the measurement of the absolute error between the predicted value and the true value. The SMAPE aims to address the problem of the mean absolute percentage error producing overly large values when the real value is quite small. The R^2 measures how well the prediction model fits

TABLE I
Details of Reference Machine Parameters

Symbol	Design parameter	Min.	Max.	Unit
d_1	Magnet 1 width	10.0	28.0	mm
d_2	Magnet 1 height	2.0	8.0	mm
d_3	Magnet 2 width	15.0	43.0	mm
d_4	Magnet 2 height	2.0	8.0	mm
d_5	Magnet inclination angle	15.0	60.0	deg
d_6	Slot width	3.5	5.0	mm
d_7	Flux barrier width	1.0	6.0	mm
d_8	Width between magnet 2 and air gap	0.5	4.0	mm
d_9	Slot depth	13.0	18.0	mm
d_{10}	Air gap	0.5	2.0	mm
d_{11}	Distance between magnets	1.0	4.0	mm
d_{12}	Distance between poles	4.0	6.5	mm
	Constant parameter	**Value**		
c_1	Number of poles	6		-
c_2	Number of stator slots	54		-
c_3	Maximum rotational speed	16800.0		rpm
c_4	Magnet temperature	75.0		°C
	Indicator			
o_1	Maximum torque			Nm
o_2	Maximum power			kW
o_3	Torque ripple at maximum torque			Nm
o_4	Material cost			Euro

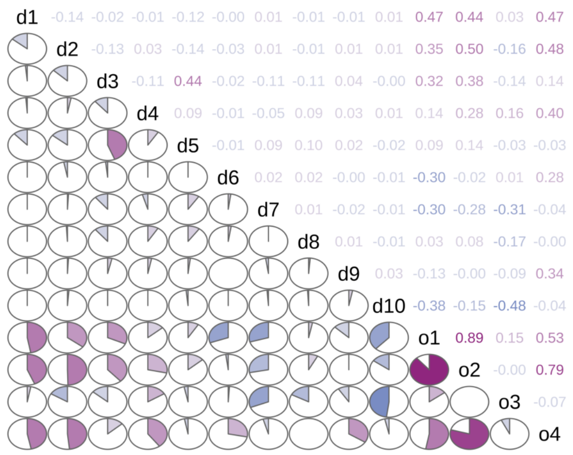

Fig. 3. Correlation between design parameters and performance indicators for dataset III.

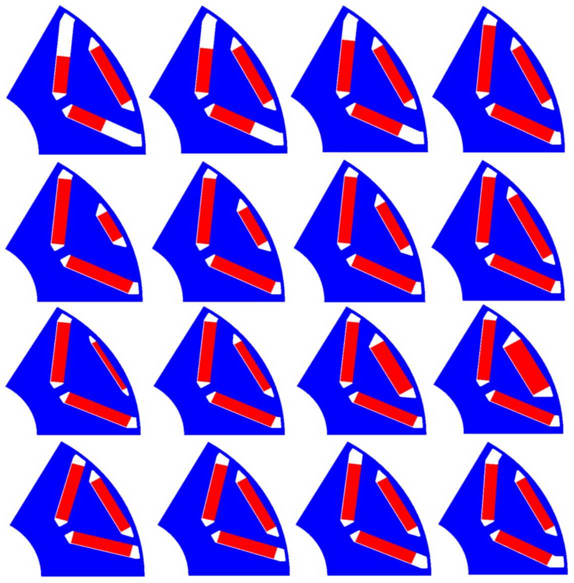
Fig. 4. Example rotor images of subdatasets.

the data. They can be calculated as follows:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - y_i^*|\,, \tag{2}$$

$$\mathrm{SMAPE} = \frac{100\%}{N}\sum_{i=1}^{N} \frac{|y_i - y_i^*|}{(|y_i| + |y_i^*|)/2}, \tag{3}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - y_i^*)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{4}$$

where N represents the total number of test samples. y_i^* describes the prediction value. $\bar{y}$ is the average true value of test data.

B. Network Architecture and Training

As previously stated in II-A, the quantity of design input parameters in the dataset is equivalent to the number of branches in the MNN. Thus, the MNNs (MNN I, MNN II, and MNN III) corresponding to the three main datasets had 6, 8, and 10 branches, respectively. We initially created six individual NNs with three hidden layers (m = 3) and 128 neurons in each layer (n = 128) for every design parameter. Each NN was then trained based on subdatasets. Subsequently, the last two hidden layers of these six NNs were transferred directly to the six branches in MNN I. The reason behind selecting the last two layers is that the learning speed of the hidden layer rises as the depth increases [20]. Regarding transferred neurons, there are two options: freezing (learning rate equal to 0), or fine-tuning (learning rate lower than that of non-transferred neurons) [21]. The learning rate for the first and last hidden layers (non-transferred layers) was set to 0.001. A test of the learning rate of transferred neurons regarding dataset III was then conducted, as shown in Fig. 5. The ranges for other hyperparameters were reduced firstly

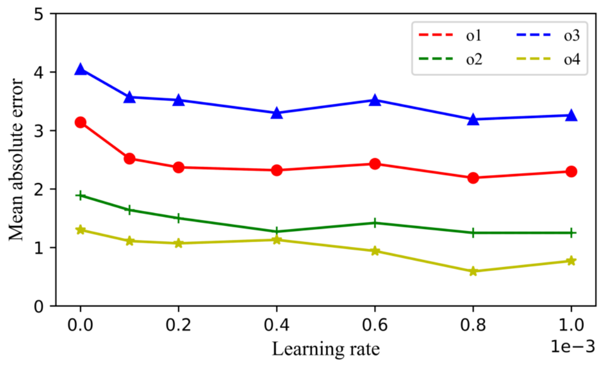

Fig. 5. Evaluation of prediction performance based on learning rate of transferred neurons under 10 sets of random seeds.

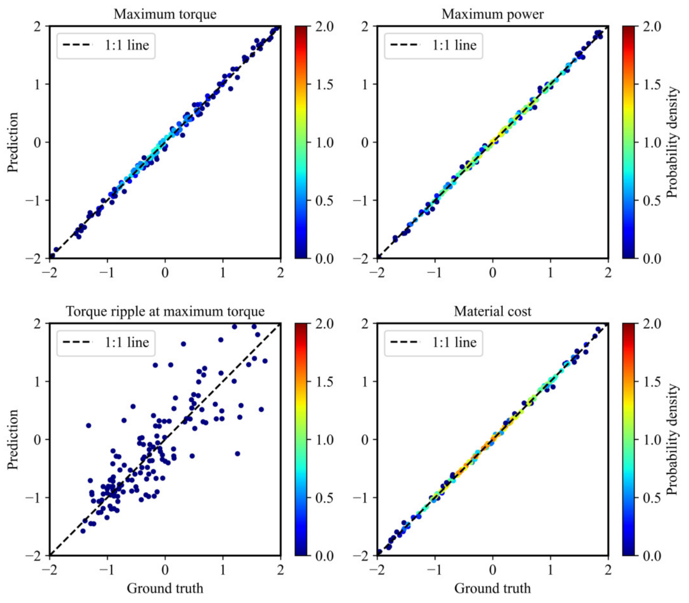

Fig. 6. Predictions for performance indicators based on dataset III.

by trial and error. The final optimized hyperparameters were determined using random research to conserve optimization time. such as optimizer (Adam), threshold of epoch (300), activation function (ELU), and batch size (8) were established.

We added layer normalization (LN) [22] between each two hidden layers. Since the proposed network contains several branches and a large quantity of trainable parameters, the stability and convergence speed of the network can be improved by this trick. Specifically, LN calculates the mean and standard deviation of the input data for each layer and normalizes the data:

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l, \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2}, \tag{5}$$

where μ^l and σ^l denote the mean and standard deviation in the l^{th} hidden layer. a_i^l is the input of the i^{th} neuron, and H is

TABLE II
EVALUATION OF ALL DATASETS

Performance indicators	**Dataset I**		**Dataset II**		**Dataset III**	
	MAE	$\boldsymbol{R^2}$	**MAE**	$\boldsymbol{R^2}$	**MAE**	$\boldsymbol{R^2}$
o_1(max. torque)	1.63	0.9969	1.60	0.9947	2.01	0.9944
o_2(max. power)	0.98	0.9985	0.95	0.9972	1.04	0.9981
o_3(torque ripple at max. torque)	1.17	0.9527	2.39	0.8580	3.19	0.6835
o_4(material cost)	0.60	0.9991	0.58	0.9985	0.59	0.9987

the number of neurons in the l^{th} hidden layer. The normalized value can be computed:

$$\tilde{\boldsymbol{a}}^l = \frac{\boldsymbol{a}^l - \mu^l}{\sqrt{(\sigma^l)^2 + \epsilon}}, \tag{6}$$

where ϵ is a very small value to prevent the denominator from being 0. Two trainable parameters $\boldsymbol{g}$ (gain) and $\boldsymbol{b}$ (bias) are included in the LN for the purpose of maintaining the distribution of the original data as much as possible. Assuming the activation function is f, the output of the LN is

$$\boldsymbol{h}^l = f(\boldsymbol{g}^l \odot \tilde{\boldsymbol{a}}^l + \boldsymbol{b}^l), \tag{7}$$

where $\odot$ indicates element-wise multiplication.

The entire training process was accomplished through the NVIDIA Quadro P4000 GPU and took approximately $\sim$3 minutes to complete a whole 5-fold cross-validation. Following cross-validation using ten distinct random seeds, we selected the model with the minimum loss for qualitative analysis. Subsequently, this model was also employed in the optimization procedure.

C. Training Results

Table II summarizes the performance of the proposed MNNs on three main datasets. Dataset III exhibited the lowest prediction accuracy. This is because it possesses the largest dimension of input design parameters and lacks sufficient data. Since MNN III is constructed on top of MNN I and MNN II, we only analyze the results of MNN III below. Fig. 6 provides evaluation details for each performance output. Our proposed TL-based MNN demonstrated superiority in predicting maximum torque, maximum power, and material cost. The color bar indicates the density of test data. The error in the predicted torque ripple is noteworthy. Insufficient sample size could be a potential explanation for this phenomenon. 2956 samples are relatively few for a dataset containing ten input parameters.

Fig. 7 demonstrates a comparison of the proposed MNN III (multi-branched MNN with transferred layers), MNN III without TL (the architecture is completely identical to the former but without the transferred layers, so all the neurons need to be trained from scratch), and traditional multi-layer perceptron (MLP) based on dataset III. Ten random seeds were employed to ensure that different models were compared across ten different divisions of the dataset. It can be observed that the MNN III outperformed the MNN III without pre-trained transferred layers, which indicated transfer learning was able to boost the network to obtain improved prediction

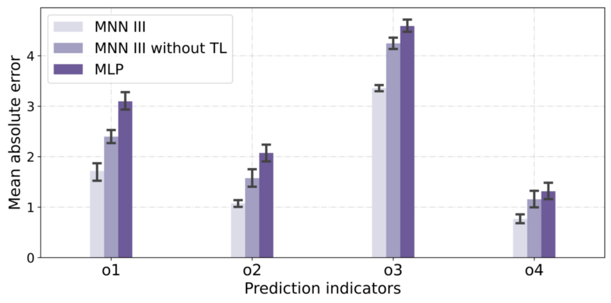

Fig. 7. Predictive performance comparison among MNN III, MNN III without TL, and MLP.

performance. The primary rationale behind this is that the network with transferred layers can leverage prior knowledge to more effectively determine the optimum. Besides, the multi-branched architecture offered superior predictive capability in comparison to MLP. We additionally conducted a comparison between MNN III with and without LN, and the outcome showed a $\sim$31% improvement in prediction accuracy with the inclusion of LN.

IV. OPTIMIZATION DESIGN

The proposed MNN III was applied to speed up the optimization process. We conducted a comparative experiment with two common and widely used multi-objective optimization (MOO) algorithms, namely NSGA-II [23] and NSGA-III [24]. The implementation was carried out by the framework Pymoo [25]. The objective functions were maximizing maximum torque and maximum power. The associated generalized optimization problem is formulated as follows:

$$o_1, o_2 \rightarrow \max \tag{8}$$

$$\text{s.t.} \quad d_i^L \leq d_i \leq d_i^U, \qquad i = 1, \cdots, n_{\text{input}} \tag{9}$$

$$c_j \leq 0, \qquad j = 1, \cdots, n_{\text{cons}} \tag{10}$$

where d_i indicates input design parameters and d_i^L and d_i^U are predefined bounds. c_j represents constraints for the optimization process. The number of iterations (200), population size (100), crossover probability (0.9), and polynomial mutation (15) were included.

The obtained Pareto-fronts are displayed in Fig. 8. FEA was performed by choosing a candidate solution from each Pareto-front. The selected optimal designs are illustrated in Fig. 9. Table III summarizes the FEA and predicted results for the selected designs. It can be seen that the predictions for o_1, o_2 and o_4 were in high agreement with the FEA. A minimum SMAPE of 0.01% can be achieved. However, the prediction accuracy for torque ripple was relatively low (over 20%). A possible reason is the low correlation between the input parameters and this motor characteristic. At last, we executed both optimization approaches 10 times each in order to calculate the average optimization time. The results indicated that it took approximately 12 seconds to perform NSGA-III, whereas NSGA-II could be finished in only about 8 seconds.

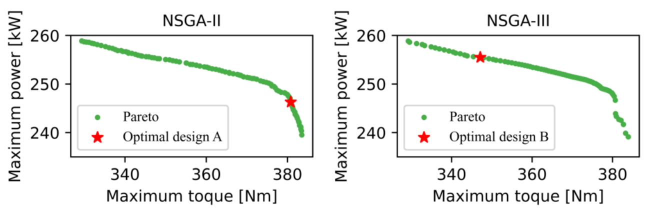

Fig. 8. Pareto-fronts for maximum torque and maximum power based on two different optimization algorithms.

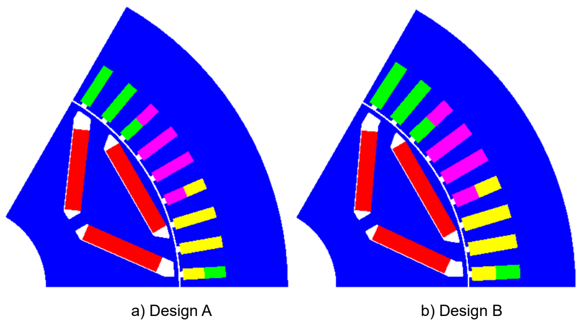

Fig. 9. Optimal designs.

TABLE III
EVALUATION OF THE OPTIMAL DESIGNS FROM PARETO-FRONTS

Indicator	Unit	Optimal design A | B		
		FEA	Prediction	SMAPE
o_1	Nm	368.66 | 347.47	380.80 | 347.12	3.24% | 0.10%
o_2	kW	242.70 | 253.45	246.30 | 255.48	1.47% | 0.80%
o_3	Nm	13.95 | 18.33	18.06 | 14.62	25.69% | 22.53%
o_4	Euro	334.61 | 357.60	332.55 | 359.72	0.62% | 0.59%

V. CONCLUSION

In this paper, we proposed the application of the transfer learning-based modular neural network for the design optimization of the IPMSM. This approach can be adapted dynamically in response to changes in the parameter-based input space. By transferring the pre-trained neurons to the new NN as initial weights, the previous knowledge was reused. After reparametrization, it was no longer needed to retrain the new NN from scratch. The experiments demonstrated that the prediction accuracy of MNN was higher than classical MLP. Furthermore, the TL-based MNN outperformed the normal MNN, attributed to the employment of transferred neurons. The MOO results showed high accuracy in predicting maximum torque, maximum power, and cost, with the optimization completed in around 12 seconds. However, accurate estimation of torque ripple is still a challenge. In future work, this method will be employed for other motor types, such as reluctance motors and electrically excited motors. Furthermore, it is interesting to investigate the degree of confidence associated with the predicted outcomes.

REFERENCES

[1] P. Stumpf and T. Tóth-Katona, "Recent Achievements in the Control of Interior Permanent-Magnet Synchronous Machine Drives: A Comprehensive Overview of the State of the Art," *Energies*, vol. 16, no. 13, Jul. 2023.
[2] X. Liu, H. Chen, J. Zhao and A. Belahcen, "Research on the Performances and Parameters of Interior PMSM Used for Electric Vehicles," IEEE Transactions on Industrial Electronics, vol. 63, no. 6, pp. 3533-3545, Jun. 2016.
[3] J. Kudela and R. Matousek, "Recent advances and applications of surrogate models for finite element method computations: A review," Soft Computing, vol. 26, no. 24, pp.13709-13733, Jul. 2022.
[4] S. Zhao, F. Blaabjerg, and H. Wang, "An overview of artificial intelligence applications for power electronics," IEEE Transactions on Power Electronics, vol. 36, no. 4, pp. 4633–4658, 2021.
[5] J. Gu, et al., "Surrogate Model-Based Multiobjective Optimization of High-Speed PM Synchronous Machine: Construction and Comparison," IEEE Transactions on Transportation Electrification, vol. 9, no. 1, pp. 678-688, March 2023.
[6] H. Sasaki and H. Igarashi, "Topology optimization accelerated by deep learning," IEEE Transactions on Magnetics, vol. 55, no. 6, pp. 1–5, 2019.
[7] A. Khan, M. H. Mohammadi, V. Ghorbanian, and D. Lowther, "Efficiency map prediction of motor drives using deep learning," IEEE Transactions on Magnetics, vol. 56, no. 3, pp. 1–4, 2020.
[8] V. Parekh, D. Flore, and S. Schöps, "Deep learning-based prediction of key performance indicators for electrical machines," IEEE Access, vol. 9, pp. 21786–21797, 2021.
[9] P. Vivek, D. Flore, and S. Schöps, "Variational autoencoder-based metamodeling for multi-objective topology optimization of electrical machines," IEEE Transactions on Magnetics, vol. 58, no. 9, pp. 1-4, 2022.
[10] H. Sato and H. Igarashi, "Fast Topology Optimization for PM Motors Using Variational Autoencoder and Neural Networks With Dropout," IEEE Transactions on Magnetics, vol. 59, no. 5, pp. 1-4, May 2023.
[11] Y. Shimizu, S. Morimoto, M. Sanada, and Y. Inoue, "Automatic design system with generative adversarial network and convolutional neural network for optimization design of interior permanent magnet synchronous motor," IEEE Transactions on Energy Conversion, vol. 38, no. 1, pp. 724–734, 2023
[12] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu, "A survey on deep transfer learning," International Conference on Artificial Neural Networks, Oct. 2018.
[13] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?" Advances in neural information processing systems, 2014.
[14] J. Asanuma, S. Doi and H. Igarashi, "Transfer Learning Through Deep Learning: Application to Topology Optimization of Electric Motor," IEEE Transactions on Magnetics, vol. 56, no. 3, pp. 1-4, March 2020.
[15] J. Qiao, X. Meng, W. Li, and B. M. Wilamowski, "A novel modular RBF neural network based on a brain-like partition method," Neural Computing and Applications, vol. 32 pp. 899-911, 2020.
[16] W. Wu, and P. Yun, "Adaptive modular convolutional neural network for image recognition," Sensors, vol. 22, no. 15, p. 5488, 2022.
[17] A. Dosovitskiy, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929 (2020).
[18] K. He, et al., "Masked autoencoders are scalable vision learners," Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022.
[19] K. R. Davey, "Latin hypercube sampling and pattern search in magnetic field optimization problems," IEEE Transactions on Magnetics, vol. 44, no. 6, pp. 974–977, 2008.
[20] Szandała, T., "Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks," Bio-inspired Neurocomputing. Springer. Singapore, 2021, vol. 903, pp. 203-224.
[21] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1299–1312, 2016.
[22] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," arXiv preprint arXiv:1607.06450, 2016.
[23] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002.
[24] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," IEEE Trans. Evol. Comput., vol. 18, no. 4, pp. 577–601, Aug. 2014.

[25] J. Blank and K. Deb, “Pymoo: Multi-objective optimization in python,” IEEE Access, vol. 8, pp. 89497–89509, 2020.

VI. Biographies

Nuo Chen received the bachelor’s degree in automotive engineering from Harbin Institute of Technology, China, in 2018, and the master’s degree in autonomous systems from the University of Stuttgart, Germany, in 2022. He is currently working toward the Ph.D. degree at the Institute of Electrical Engineering, Karlsruhe Institute of Technology, Germany. His research interests include machine learning, deep learning, and the optimization design of electrical machines.

Martin Doppelbauer was born in Altenhundem, Germany. He received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from the University of Dortmund, Germany in 1990 and 1995, respectively. He was with the companies Danfoss Bauer in Esslingen and SEW Eurodrive in Bruchsal, Germany, as Senior Manager for electric motor development. Since 2011 he is full professor at the Institute of Electrical Engineering (ETI) at the KIT. He holds a chair for Hybrid Electric Vehicles. Martin Doppelbauer is also active in national and international standardization and the chairman of IEC Technical Committee 2 Rotating Electrical Machines.