Deep Learning based Design and Optimization of Electrical Machines

Deep Learning basiertes Design und Optimierung Elektrischer Maschinen
Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Vivek Parekh aus Bhavnagar, Indien
Elektrotechnik und Informationstechnik
Tag der Einreichung: October 24, 2023, Tag der Prüfung: March 4, 2024

1. Gutachten: Prof. Dr. Sebastian Schöps
2. Gutachten: Prof. Dr. David Lowther
Darmstadt, Technische Universität Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich 18
Computational Electromagnetics Group

Deep Learning based Design and Optimization of Electrical Machines
Deep Learning basiertes Design und Optimierung Elektrischer Maschinen

genehmigte Dissertation von Vivek Parekh aus Bhavnagar, Indien
Elektrotechnik und Informationstechnik

Date of submission: October 24, 2023
Date of thesis defense: March 4, 2024

Darmstadt, Technische Universität Darmstadt

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, October 24, 2023

Vivek Parekh

Zusammenfassung

Die Entwicklung von technischen Produkten erfordert in der Industrie erhebliche natürliche Ressourcen, menschlichen Einsatz und Zeit. Es kann kostspielig sein, wenn physikalische Phänomene während der Herstellung nicht korrekt vorhergesagt werden. Das virtuelle Prototyping ermöglicht die Analyse physischer Prozesse unter realen Bedingungen, bevor die eigentliche Produktfertigung beginnt. In den letzten Jahrzehnten wurden verschiedene Simulationssoftwares entwickelt, die es ermöglichen, verschiedene Arbeitsbedingungen, komplexe Konstruktionskriterien und Beschränkungen während der Simulation zu berücksichtigen. Diese Simulationstools erfordern jedoch eine beträchtliche Rechenleistung zur Lösung komplexer mathematischer Modelle, wodurch die Kapazität der numerischen Analyse zur Erkundung eines großen Designraums für optimale Designs eingeschränkt wird. In den letzten Jahren haben sich datengetriebene Deep-Learning-Methoden (DL) entwickelt. Sie können den teuren Rechenaufwand erheblich reduzieren, indem sie ein kostengünstiges Metamodell zur Vorhersage der physikalischen Ausgangsgrößen im Entwurfsprozess bereitstellen.

In dieser Arbeit werden verschiedene datengetriebene DL-Ansätze zur Beschleunigung des Entwurfsoptimierungsverfahrens von elektrischen Maschinen untersucht. Alle vorgeschlagenen Ansätze konzentrieren sich darauf, die Erkundung eines hochdimensionalen Designraums zu ermöglichen, um optimale Maschinenentwürfe zu generieren und dabei Rechenressourcen zu sparen.

Zunächst werden verschiedene Geometriedarstellungen von Permanentmagnet-Synchronmaschinen (PMSM) analysiert. Bildbasierte Modelle werden für verschiedene Pixelauflösungen untersucht. Es wird ein quantitativer Vergleich zwischen den bildbasierten und den auf skalaren Parametern basierenden Metamodellen zur Annäherung an bereichsübergreifende Leistungskennzahlen (KPIs) von PMSMs durchgeführt. Die numerischen Ergebnisse zeigen, dass das auf skalaren Parametern basierende Metamodell eine hohe Vorhersagegenauigkeit aufweist und gleichzeitig computational effizient ist. Andererseits sind bildbasierte Modelle in Szenarien flexibler, z.B. bei Querrotor-Topologien und Reparametrisierung. Alle trainierten Meta-Modelle bewerten neue Entwürfe in deutlich kürzerer Zeit als konventionellen Finite-Elemente-Simulationen.

Zweitens wird ein neuartiger hybrider daten- und physikgetriebener Ansatz vorgeschlagen, um die Vorhersagegenauigkeit und Flexibilität der skalaren Darstellung für die Quantifizierung der Leistung von PMSMs zu verbessern. Der hybride Ansatz wird mit einem datengesteuerten Ansatz verglichen. Schließlich wird eine multikriterielle Optimierung mit einem hybriden Ansatz in einem industriellen Umfeld durchgeführt und eine quantitative Analyse vorgenommen.

Es wird eine Methode vorgestellt, um KPIs vorherzusagen, indem ein hochdimensionaler komplexer skalarer Designraum auf einen niederdimensionalen latenten Raum für unterschiedlich parametrisierte Maschinentechnologien und -topologien mit Hilfe eines tiefen generativen Modells abgebildet wird. Dieser Ansatz ermöglicht die gleichzeitige parametrische Optimierung verschiedener Maschinentypen und Rotortopologien mit einem einzigen Metamodell-Training. Die vorgeschlagene Methode wird für zwei Szenarien

demonstriert: erstens, für die gleichzeitige Optimierung von heterogen parametrisierten Rotor-Topologien, und zweitens, für heterogen parametrisierte Maschinentechnologien.

Alle vorgeschlagenen Methoden sind so allgemein, dass sie in jedem industriellen Produktdesign-Workflow angewandt werden können, bei dem physikalische Phänomene als ein System linearer oder nichtlinearer Funktionen beschrieben werden.

Abstract

Developing engineering products requires significant natural resources, human effort, and time in the industry. It can be expensive if physical phenomena are not predicted correctly during manufacturing. Virtual prototyping enables the analysis of physical processes under real-world conditions before actual product manufacturing. Various simulation softwares have been developed in recent decades that allow for considering different working conditions, complex design criteria, and constraints during design simulation. However, these simulation tools require significant computational power to solve complex mathematical models, which limits the capacity of numerical analysis for exploring a large design space for optimal designs. Data-driven deep learning (DL) methods have evolved in recent years. They can notably reduce expensive computational effort by finding a low-cost meta-model to predict physical output quantities in the design process.

In this thesis, different data-driven DL approaches for accelerating the design optimization procedure of electrical machines are investigated. All the proposed approaches are focused on enabling the exploration of a high-dimensional design space to generate optimal machine designs while saving computational resources.

First, various permanent magnet synchronous machine (PMSM) geometry representations are analyzed. Image-based models are studied for different pixel resolutions. A quantitative comparison is made between the image-based and scalar parameter-based meta-models for approximating cross-domain key performance indicators (KPIs) of PMSMs. Numerical results showed that the scalar parameter-based meta-model has high prediction accuracy while being computationally cheap. On the other hand, image-based models are more flexible in scenarios, e.g., cross-rotor topologies and reparameterization. All trained meta-models evaluate new designs in much less time than conventional finite element simulations.

Second, a hybrid data- and physics-driven approach is proposed to improve the scalar representation's prediction accuracy and flexibility for quantifying the performance of PMSMs. The electromagnetic behavior is characterized using a data-driven DL approach, and subsequent KPIs are evaluated using a physics-based post-processing tool. The hybrid approach is compared to a data-driven approach. Finally, multi-objective optimization is performed using a hybrid approach in industrial settings, and quantitative analysis is conducted.

A method is introduced to predict KPIs by mapping a high-dimensional complex scalar design space to a lower-dimensional latent space for differently parameterized machine technologies and topologies using a deep generative model. This approach enables concurrent parametric optimization of different machine types and rotor topologies with a single meta-model training. The proposed method is demonstrated for two scenarios: first, for the concurrent optimization of heterogeneously parameterized rotor topologies, and second, for heterogeneously parameterized machine technologies.

All proposed methods can be applied to any industrial product design workflow where the physical phenomena can be described as a system of linear or nonlinear functions.

Contents

List of figures

List of tables

1 Introduction

In this introductory chapter, first, the motivation for the research is stated. Subsequently, a brief literature review on the application of various machine-learning algorithms in the electrical machines domain is provided. Finally, the significant contribution of this work is described, followed by an outline of the thesis structure.

1.1 Motivation

Electrical machines, an example of electromechanical devices in the form of motors and generators, are considered as one of the most vital devices for energy conversion in recent times. Electrical machines can be operated by renewable energy resources such as solar energy and hydroelectric power, which are more sustainable and eco-friendly energy sources than conventional fossil fuels. A wide range of devices, including household appliances, automobiles, and heavy industrial machinery, are driven by electrical machines. Depending on the application, different classes of electrical machines with diverse design configurations and characteristics are chosen. For example, small permanent magnet (PM) motors are installed in healthcare equipment such as implanting devices, insulin pumps, and many other devices as described in [65]. Due to their mobility and durability features, switched reluctance and brushless DC motors are utilized in the aircraft industry [18]. Permanent magnet synchronous machines (PMSMs) have numerous industrial applications because of their high efficiency, power density, torque characteristics, and fault tolerant capacity [161]. There are diverse examples of the application of PMSMs such as robot applications [79], electric vehicles (EVs) [31, 162, 195], transportation [20, 181], air blowers, [82], compressors [218], high-speed centrifugal machines, turbine generators [194], or high power traction [150]. Similarly, due to simple and robust design, low production cost, and their ability to operate at variable speed, induction machines (IMs) are employed in heating, ventilation, and air conditioning systems [149], systems with reduced costs for low-power applications [85], hybrid vehicles [31] and so on. The physical construction of electrical machines includes materials such as copper, aluminium, electrical steel, and iron which have a moderate to high-cost range depending on the processing and availability in the geographic region. In addition, PMSMs nowadays contain high-cost magnets with rare earth materials, i.e., neodymium-iron-boron (NdFeB) and samarium-cobalt (SmCo). These rare earth magnets produce high torque, power density and efficiency in PMSMs due to their magnetic properties, such as high field strength, energy density, and operating temperature range. Therefore, they are widely used in electric and hybrid electric vehicles (HEVs) [64, 211]. The production rate of EVs (see Figure 1.1) and HEVs is very high in recent years as they are considered a much cleaner way of transportation than fossil combustion-based vehicles [91]. The manufacturing and industrial operations of electrical machines may emit greenhouse gases (e.g., CO_2), which cause global warming effect [124]. On the other hand, the production of electrical machines using rare earth materials is also an environmental challenge. Also, the material consumption contributes significantly to the final cost of the manufactured electrical machine; while other processes such as virtual prototyping,

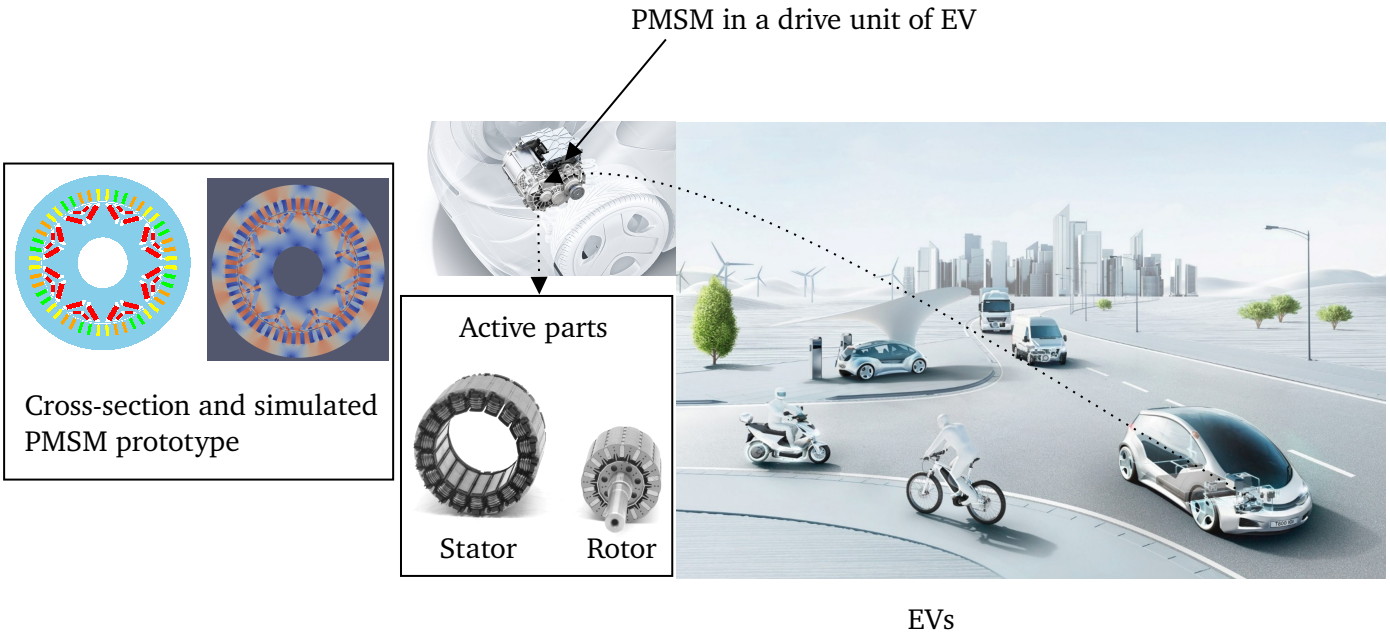


Figure 1.1: Illustration of cross section of PMSM, simulated prototype, active parts, a drive unit developed for EV (car), and EVs (Source: Robert Bosch GmbH).

account for only a minor portion of the cost. Hence, prior to construction, numerical optimization of active parts (e.g., stator and rotor) is necessary to minimize material consumption, reduce carbon footprint, and maximize efficiency while fulfilling dynamic industrial requirements (e.g., continuous peak power, torque). The numerical optimization of an electrical machine incorporates electromagnetic, thermal, and structural analysis, making it a complex multi-domain objective optimization problem [28, 174]. The design simulation is normally performed by finite element (FE) software or through analytical calculations. The simulation takes from minutes to several hours to compute a single design based on the computational settings and available resources. This is because the input design space for optimizing the electrical machine typically contains many design variables (e.g., electrical, geometry, and material) as well as many cross-domain key performance indicators (KPIs) or design objectives. These KPIs may include torque characteristics, power, cost, torque ripple, efficiency, etc. The numerical value of the KPIs assesses the performance of an electrical machine design. Considering these factors, a numerical optimization requires expertise, a large number of simulations of physical phenomena, and huge computational resources. As a result, a numerical optimization is usually time-consuming and computationally expensive. With limited computing resources and time, optimizers can evaluate only a few hundred to a few thousand samples during optimization. The result of an optimization yields a set of optimal designs. Out of these, a final candidate that fulfills all constraints and design criteria can be selected while achieving the desired performance. This choice is often based on specific criteria manually set by the designer, which could include factors like cost, efficiency, durability, or any other relevant KPIs. The chosen final optimal design is constructed as a physical prototype. It is, furthermore, validated by measurement. The optimization process takes a few days to months, depending on the specification of available computing resources. Due to the huge input design space and many target KPIs, the optimization process may not explore the whole design space effectively. Therefore, at the end of optimization, the final set of feasible candidates may not contain the optimal solution due to the non-exploration of design space. This motivates the search for

alternative methods that accelerate optimization by reusing existing simulation data. These methods also aim to efficiently explore a vast design space by evaluating many designs, which may help to obtain designs closer to the real global optimum. A further aim of this study, to enhance the input design space variability, is to find an approach that allows for the concurrent optimization of differently parameterized machine technologies (e.g., IM and PMSM) with varying topological parameters (e.g., pole pairs). This thesis will present a few novel data-driven deep learning-based approaches to accelerate the optimization procedure of rotating electrical machines by exploring large complex design spaces.

1.2 Literature review

Meta-modeling aims to approximate the performance of a computationally expensive simulation model, calculating required target functions with a lower computational cost while maintaining reasonable prediction accuracy. Meta-modeling has emerged as a prominent field for researchers in the recent decade [111]. Polynomial interpolation or regression, which can also be referred to as either the spectral method or polynomial chaos, is a commonly utilized method in uncertainty estimation [25, 39, 60, 227]. Another usual meta-modeling method is Kriging, an interpolation technique to estimate the value of unknown functions based on a set of observed values. It may make some additional prior assumptions, such as the variable being estimated following a specific distribution or kernel (e.g., Gaussian distribution) [36, 54, 55]. Kriging has been particularly popular to solve electromagnetic optimization problems cost-effectively [17, 127, 207, 226].

In the last decade, the use of machine learning (ML) has increased in diverse sectors, e.g., healthcare [215], aerospace industry [34], weather forecast [139], robotics [115], agriculture [213], recommendation systems [144] and so on. Recently, the GPT-3 language model [30], a breakthrough in natural language processing, has drawn widespread public attention towards machine learning. ML has gained popularity among researchers due to several factors, e.g., the development of computational resources concerning hardware and software and the possibility of generating large datasets. The utilization of deep neural networks (DNNs) has become prevalent due to their favourable attributes, such as the ability to manage large datasets, ease of parallelization via graphical processing units (GPUs), automated feature extraction, and capacity to handle high-dimensional data. A Subclass of DNN, a convolutional neural network (CNN), has shown great potential for different computer vision tasks due to its ability to capture visual input features and transfer knowledge through a pre-trained network from one domain to another [112, 118, 120, 210]. The DNN, due to its complex, non-linear function approximating capacity, has been commonly used to solve a multiple-output regression problem [70, 80, 130, 202].

Recently, there has been rapid growth in the use of machine-learning-based algorithms for applications of electrical machines. Meta models have been trained to approximate various KPIs at different stages using state-of-the-art ML algorithms. There are several studies, for example, estimation of magnetic field distribution for low-frequency electromagnetic devices [96], torque estimation [228], tracking of real-time temperature fluctuation in PMSM [110], torque prediction for different states of a PMSM drive [128], approximation of speed, efficiency, and torque for PMSM [87], assessment of stator winding fault [163], predicting performance characteristics [57] and so on. It is demonstrated that machine learning-trained meta-models significantly expedite the optimization procedure for electrical machines, e.g., [8, 48, 66, 72, 116, 231]. A detailed discussion of the state-of-the-art deep learning algorithms for electrical machine applications can be found in Sec. 3.3.

1.3 Contribution

This thesis investigates different data-driven deep learning (DL) algorithms to deal with non-linear multi-target regression problems for electrical machines. The application of the proposed approaches is focused on accelerating the optimization of rotating electrical machines, e.g., PMSM and ASM. The main contribution of this research work is as follows:

Firstly, a data-driven DL approach is presented that facilitates faster prediction of a large number of cross-domain KPIs with reasonable accuracy [152]. The approach focuses on analyzing various input geometry representations of PMSMs, specifically image-based and scalar parameter-based representations. In this study, three distinct artificial neural networks (ANNs) are derived and trained on two different industrial datasets. A quantitative comparison demonstrates that the meta-model based on scalar input representation obtains higher prediction accuracy while being computationally efficient. The applicability and limitations of both input representations are discussed for various scenarios.

Secondly, a hybrid model that combines a data-driven DL model with physics-based post-processing is developed to evaluate PMSM KPIs using a scalar parameter-based input representation [153]. The proposed method addresses the limitations of the data-driven approach [152]. An extensive quantitative analysis illustrates the superior accuracy and flexibility of the hybrid approach compared to the data-driven model. For further analysis from a practical application perspective, the hybrid model is employed alongside the conventional approach in an industrial multi-objective optimization (MOO) loop to navigate a vast and complex design space [155].

A generative model-based probabilistic approach is proposed for the concurrent optimization of differently parameterized rotating electrical machine topologies and technologies over a common set of KPIs [151, 154]. The proposed approach is applied for concurrent MOO in two scenarios: one for heterogeneously parameterized PMSM rotor topologies, and the other for heterogeneously parameterized machine technologies. Additionally, it is quantitatively compared to the individually trained DL models.

In summary, the thesis proposes data-driven deep learning approaches to accelerate the numerical optimization of rotating electrical machines in a vast and challenging design space.

1.4 Outline

The remaining structure of the thesis is as follows. Chapter 2 commences with a brief general introduction to rotating electrical machines, providing a quick overview of the design process from a production standpoint. In Sec. 2.2, first, Maxwell's equations are formulated, followed by a explanation of the FE method. Then, the simulation workflow for PMSMs to calculate KPIs is presented in Sec. 2.2.3. Lastly, the basics of optimization and widely used gradient and population-based optimization methods are briefly explained in Sec. 2.3.

Chapter 3 discusses the fundamentals of deep learning. A brief overview of artificial intelligence, machine learning, and deep learning, along with explanations of commonly used learning strategies, is presented in Sec. 3.1. The deep learning architectures utilized in Chapters 4 to 6 are explained in Sec. 3.2. Lastly, a literature review of recent articles focusing on the applications of deep learning algorithms in the domain of rotating electrical machines is provided in Sec. 3.3.

Chapters 4 to 6 represent the primary contributions of this thesis. Chapter 4 introduces a data-driven DL strategy for predicting a large number of cross-domain KPIs concerning different input geometry representations of PMSMs. A generalized multi-objective optimization problem concerning the design of rotating electrical machines is then outlined in Sec. 4.2. The general process of data generation, which is consistently employed throughout the thesis, is elaborated upon in Sec. 4.3. The procedures for tuning the hyperparameters of ANNs, which are predominantly used in this treatise to propose all meta-models, are detailed in Sec. 4.4. This section also covers the architecture of the proposed networks and the training process. Subsequently, a detailed numerical analysis for both datasets is presented in Sec. 4.5. The chapter concludes with a summary of findings in Sec. 4.6.

Chapter 5 begins by introducing a generalized data- and physics-based hybrid approach, stating the limitations of the data-driven approach introduced in Chapter 4. In Sec. 5.2, the procedure of the hybrid approach is described, and the information regarding the dataset used is given. The proposed multi-branch DNN and its training specifications are discussed in Sec. 5.3. Sec. 5.4 starts with a numerical analysis of the prediction results of FE outputs, then presents a quantitative comparison between the hybrid and data-driven approaches. In Sec. 5.5, the proposed hybrid method is applied to the industrial multi-objective optimization loop, addressing a high-dimensional complex design space. The chapter concludes with a summary of the findings.

The approaches proposed in Chapter 4 and Chapter 5 are limited to quantifying KPIs for a single electrical machine type using a scalar parameter-based input representation. In practical scenarios, KPI requirements may vary based on specific applications, making it unclear which machine type best fits the given needs. Relying on individual meta-models for each machine type can be computationally intensive, especially when multiple options exist. A deep generative network-based meta-modeling approach that concurrently adapts multiple machine types and topologies is introduced in Chapter 6. The motivation behind the proposed approach is given in Sec. 6.1, while the generative network-based training methodology is presented in Sec. 6.2. Both Sec. 6.3 and Sec. 6.4 follow a similar structure, each showcasing an application of the proposed approach in two distinct scenarios: the former focuses on heterogeneously parameterized topologies and the latter heterogeneously parameterized machine types, respectively. The chapter concludes with a summary of the findings.

Lastly, the thesis is concluded in Chapter 7 with an over all conclusion and outlook.

2 Background

The design of efficient electrical machines becomes imperative as steps towards utilizing cleaner energy sources, decreasing global energy demand, and reducing reliance on fossil fuels are taken. Therefore, comprehensive knowledge of different cross-domain key performance measures has become vital. In this chapter, a brief introduction to rotating electrical machines is provided in Sec. 2.1. An electromagnetic analysis and an electromagnetic design simulation workflow for calculating KPIs are presented in Sec. 2.2. In Sec. 2.3, the basics of optimization are explained and well-established gradient-based and population-based optimization methods are briefly reviewed.

2.1 General introduction of rotating electrical machines

The fundamental structure of a rotating electrical machine is broken down into two primary components: the stator and the rotor. The stator is a fixed part that commonly has three-phase windings (spatially shifted by 120°) that use an alternating current (AC) supply (temporally shifted by 120°) to create a rotating magnetic field. The stator winding inside the slots consists of electrically conducive material such as copper or aluminium. The stator can be further decomposed into a stator yoke, stator slots and stator teeth. The stator core is typically made of layers of steel laminations coated with an insulating material to reduce iron losses and mitigate generated heat. The part of the machine that rotates is called a rotor. It contains e.g., coils (electromagnets), permanent magnets, or a squirrel cage winding with aluminium bars (does not require an external source) to produce magnetic fields. The rotor is built to interact with the magnetic field created by the stator. This interaction creates an electromagnetic torque that enables the rotor to rotate. Similar to the stator, the rotor can be partitioned into rotor shaft and rotor core. The rotor core consists of materials, e.g., iron, aluminium, or steel. It can be in various shapes and designs depending on the class of machine and its intended application. For example, in an induction motor, the rotor core may have a cylindrical shape and block of metal with aluminium bars, where in the PMSM, the rotor core incorporates permanent magnets. Figure 2.1 shows cross-sections of different rotating machine types: PMSM, asynchronous machine (ASM), electrically excited synchronous machine (EESM), synchronous reluctance machine (SynRM), and switched reluctance machine (SRM). The method of how to generate a magnetic field inside the rotor is a major differentiating characteristic among all these machines. For example, the PMSM has a rotor with magnets, whereas an ASM rotor contains a multi-phase winding or cage (short circuited with end rings and slots with bars). Similarly, in the EESM rotor the magnetic field can be controlled by an electromagnet (field excitation); the SynRM incorporates a cylindrical rotor with air gaps; in a SRM, switching in coil currents concerning spatial shift generates a magnetic field in the rotor. Each of these machines has different advantages and disadvantages. The PMSM has a high torque density, power, and continuous torque; conversely, it comprises high-price magnets prone to demagnetization in adverse conditions [122, 161]. It also requires a position sensor to detect the rotor position continuously. The EESM has the advantage of overall high efficiency and torque density, but it requires additional

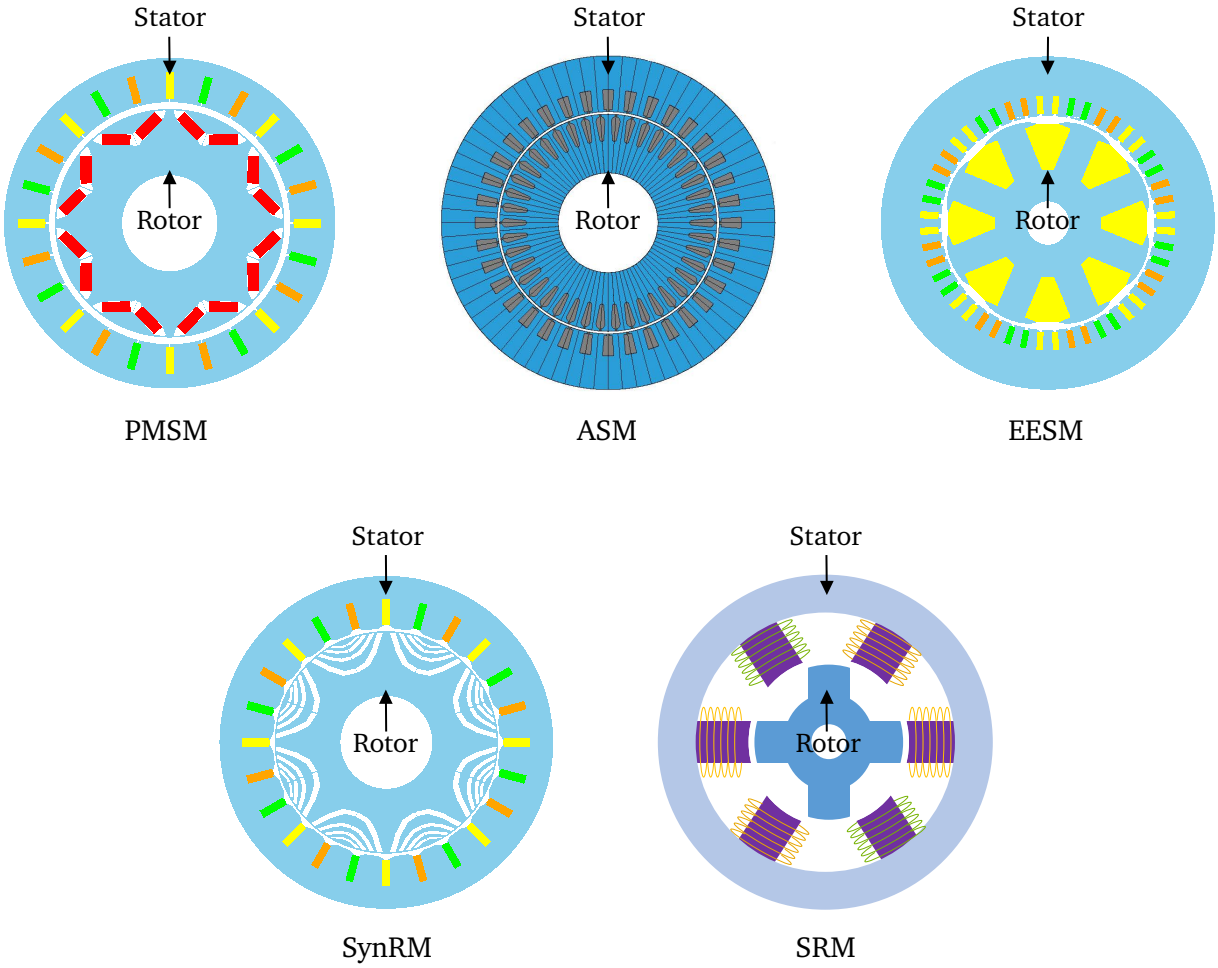


Figure 2.1: Basic structures of different rotating electrical machines.

measures for rotor excitation [81]. Both are preferable in EV applications due to their advantages, but a machine type may be selected based on a precise performance requirement for the target application as demonstrated in [81, 134]. Another performance comparison between the ASM and the SynRM is given in [19]. All these rotating machine designs are quantified by various performance measures based on their design and application domain. These performance measures, alternatively known as KPIs, may include electromagnetic torque, continuous torque, maximum torque, shaft power, critical field strength, iron losses, material cost, sound power level, etc.

2.1.1 Generalized design process for rotating electrical machines

The design flow of a typical electrical machine is illustrated in Figure 2.2. It has several stages before the final design is manufactured for the target application.

In the first stage, a set of design specifications is defined to meet various requirements. It contains an initial design concept. The design specification includes multi-domain KPIs such as rated power, current, duty cycle, physical size, weight, noise impact, temperature, etc. It consists of essential constraints and design parameters. In addition, manufacturers and customers may consider certain environmental factors like

carbon emissions and possible operating scenarios when selecting material for the product.

The next stage is virtual prototyping. It is the crucial phase, where multi-physics and multi-objective optimization are conducted to obtain the optimal design prototype. It is common to start with FE-based or analytical electromagnetic analysis, followed by checking the feasibility of electromagnetic compatibility (EMC). Then, the mechanical analysis is carried out to inspect stress, durability, and structural compatibility for the set of optimal electromagnetic designs. Noise, vibration and harshness (NVH) simulation is run to ensure the safety and performance of the mechanical system especially to have no annoying sounds and no oscillation problems at specific modes. Typically, within the design procedure, the mechanical and NVH analysis can be conducted with the aid of analytical models that can be assessed swiftly and with minimal effort. If the final set of optimal designs fail to meet the criteria of this stage, another run of electromagnetic optimization may be performed. In the next step, thermal analysis is

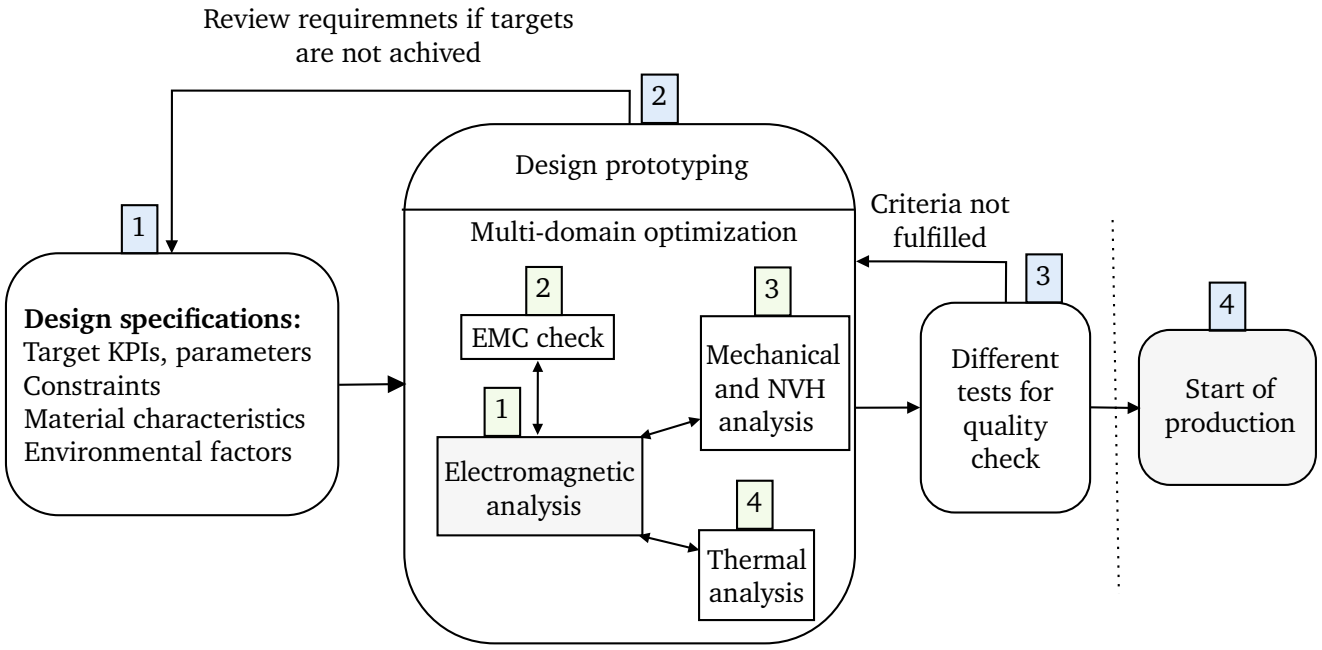


Figure 2.2: Block diagram for the generalized design process for multi-domain, multi-objective optimization of a rotating electrical machine. The numbers in the box show the step sequence.

conducted to monitor the increase in temperature due to excess heat generation as a result of losses (e.g., copper losses, iron losses) for the final set of optimal designs. It must stay within decided safety limits for the machine's smooth operation in the given operating conditions. The execution and repetition of all these analyses is majorly dependent on the target application. If no single configuration is found that fits all the criteria at the end of multi-domain optimization, then a review for revised specifications is required. The workflows for multi-domain optimization of rotating electrical machines are described in [3, 172, 174] with more details.

After finalizing the virtual prototype, various tests and quality checks are performed. These tests may include general manufacturing assessments, such as checking for end winding fitting in stator slots, stamping of lamination sheets, conducting parameter identification tests, and evaluating drawing tolerances. The design prototype is sent for production if all the tests are passed.

2.2 Electromagnetic analysis

In this dissertation, the KPIs are mainly obtained using models based on Maxwell's equations. They involve electric and magnetic field interaction in space and time for energy transformation, e.g., electrical machines, transformers, and antennas. An electromagnetic analysis can be carried out by numerical and analytical methods. Analytical methods seek closed-form solutions but it is often not possible to achieve solutions for electrical machines due to the complex geometry structure (e.g., rotor structure), boundary conditions, and nonlinearities of the materials [183]. However, analytical methods are computationally cheap. On the other hand, most numerical methods solve complex physical problems by decomposing them into smaller parts. They can be divided into solving integral and differential equations for non-linear systems. The integral field equations can be solved using the Boundary Element Method (BEM). With the BEM, the field solutions on boundaries are obtained, and subsequently, the field inside the model can be determined through post-processing. However, non-linearities cannot be incorporated using the BEM. Combining the BEM with other numerical methods, such as finite elements, can be advantageous [182]. Numerical methods can be further segregated into finite difference method (FDM) and finite element method (FEM). FDM is easy to implement but typically requires a structured grid. It approximates the derivatives of the field equation with Taylor series [200]. It does not efficiently discretize complex geometries of electrical machines, and that is where the FEM is particularly useful [180].

2.2.1 Maxwell's equation for electromagnetic analysis

The electromagnetic behavior of electrical machines can be comprehended with the help of Maxwell's equations. James Clerk Maxwell first described them in 1862 by adding an extra term to Ampere's law. The set of Maxwell's equations and a few material equations can describe the electromagnetism phenomena of any electromagnetic device. The differential form of Maxwell's equations is discussed. The first equation represents Faraday's law. It asserts that a change in the magnetic field with time generates an electric field circulating around it. A negative sign indicates that the generated electric field opposes the magnetic field change that caused it. It is represented as

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}, \tag{2.1}$$

where $\nabla\times$ is the curl operator, $\vec{E} = \vec{E}(\vec{r}, t)$ is the electric field strength, and $\vec{B} = \vec{B}(\vec{r}, t)$ is magnetic field. In this context, t denotes time, while $\vec{r} = (x, y, z)$ refers to a spatial vector.

The second equation describes Ampere's law. In simple terms, it states that a magnetic field is generated by time-varying current and electric field displacement, i.e.,

$$\nabla \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t}, \tag{2.2}$$

where $\vec{H} = \vec{H}(\vec{r}, t)$ represents the magnetic field intensity, $\vec{J}$ is electric current density. $\vec{D}(\vec{r}, t)$ is electric field displacement density.

The third equation is Gauss's law, which states that the quantity of the electric field through a closed surface is proportional to the amount of electric charge enclosed by that surface. It is written as

$$\nabla \cdot \vec{D} = \varrho, \tag{2.3}$$

where $\nabla\cdot$ is the divergence operator and $\varrho = \varrho(\vec{r})$ is the charge density.

The last equation states that magnetic monopoles do not exist. It is described by

$$\nabla \cdot \vec{B} = 0. \tag{2.4}$$

The material relations in the linear case are described by

$$\vec{D} = \epsilon\vec{E} \tag{2.5}$$

$$\vec{J} = \sigma\vec{E} \tag{2.6}$$

$$\vec{B} = \mu\left(\vec{H} + \vec{M}_{\mathrm{pm}}\right), \tag{2.7}$$

where $\epsilon = \epsilon(\vec{r})$ is the electrical permittivity in the domain, $\mu = \mu(\vec{r})$ is the magnetic permeability, and $\sigma = \sigma(\vec{r})$ is the electrical conductivity. Equation (2.6) is a *Ohm's law*. $\vec{M}_{\mathrm{pm}} = \vec{M}_{\mathrm{pm}}(\vec{r})$ is the magnetization of the permanent magnets. In real-world problems, these material laws are usually nonlinear. Here, they are assumed linear to simplify the mathematical equations. The electrical permittivity, expressed as a composition of vacuum permittivity (ϵ_0) and relative permittivity (ϵ_r), can be written as $\epsilon = \epsilon_0\epsilon_r$. Similarly, the magnetic permeability, comprising vacuum permeability (μ_0) and relative permeability (μ_r), can be written as $\mu = \mu_0\mu_r$. The reluctivity can be represented as $\nu = \frac{1}{\mu}$.

The magnetic vector potential (MVP) $\vec{A} = \vec{A}(\vec{r}, t)$ is introduced and (2.4) is reformulated such that

$$\vec{B} = \nabla \times \vec{A}. \tag{2.8}$$

If $\vec{A}$ is introduced into (2.1) and then both sides are integrated over position, then

$$\vec{E} = -\frac{\partial\vec{A}}{\partial t} - \nabla V \tag{2.9}$$

is obtained, where $V = V(\vec{r})$ is the electric (scalar) potential.

In this research, simulation datasets are generated using a magnetostatic FEM solver for low frequency PMSM. In the case of magnetostatics, the time derivative of electric flux density is ignored [41], therefore $\frac{\partial\vec{D}}{\partial t} = 0$, and further simplification of (2.2), (2.6), (2.7), (2.8) using the MVP $\vec{A}$ leads to

$$\nabla \times (\nu\nabla \times \vec{A}) + \sigma\frac{\partial\vec{A}}{\partial t} = \vec{J}_{\mathrm{s}} + \nabla \times \vec{M}_{\mathrm{pm}}, \tag{2.10}$$

where $\vec{J}_{\mathrm{s}} = -\sigma\nabla V$, $\vec{J}_{\mathrm{s}}$ is the source current density and $\nabla\times\vec{M}_{\mathrm{pm}}$ shows current generated by the permanent magnets. The source current density ($\vec{J}_{\mathrm{s}}$) concerning winding functions χ_k and currents i_k can be described as $\vec{J}_{\mathrm{s}} = \sum_k \chi_k i_k$ given in [191]. From this point onward, $\vec{J}_{\mathrm{total}} = \vec{J}_{\mathrm{s}} + \nabla \times \vec{M}_{\mathrm{pm}}$ will be considered for the sake of simplicity. Equation (2.10) is the so-called A-formulation, and it is applied in two-dimensional (2D) problems.

In many instances, such as when using laminated steel in the construction of electric machines where the current aligns with the stacking direction of the laminations, it is possible to disregard eddy-currents (from equation (2.6) in [24]); then,

$$\nabla \times (\nu\nabla \times \vec{A}) = \vec{J}_{\mathrm{total}} \tag{2.11}$$

is obtained. Further, for the 2D case, the magnetic field is represented as $\vec{B} = (B_x, B_y, 0)$, which implies that $\vec{J}_{\text{total}} = (0, 0, J_{\text{total},z})$, yielding

$$-\nabla \cdot (\nu \nabla A_z) = J_{\text{total},z}. \tag{2.12}$$

Equation (2.11) may be used to simulate PMSM and IMs. However, equation (2.11) cannot be applied to situations involving transient or time-harmonic analysis because the absence of eddy currents is assumed [184].

2.2.2 Finite element method

The FEM is a widely used numerical technique for solving partial differential equations (PDEs) to approximate the physical behavior of complex engineering systems [7, 184]. The FEM is used for electrical machines such as magnetostatic, magneto-quasistatic, electrostatic, electro-quasistatic, thermal, structural, and acoustic analyses. The basic procedure of how FEM works using magnetostatic settings is explained in this subsection. As illustrated in Figure 2.3, the FEM is typically described using six steps. The process will be followed as outlined in [7, 184] to solve general PDEs.

In the first step, equations are simplified, for example, through the A-formulation (2.12) of Maxwell's

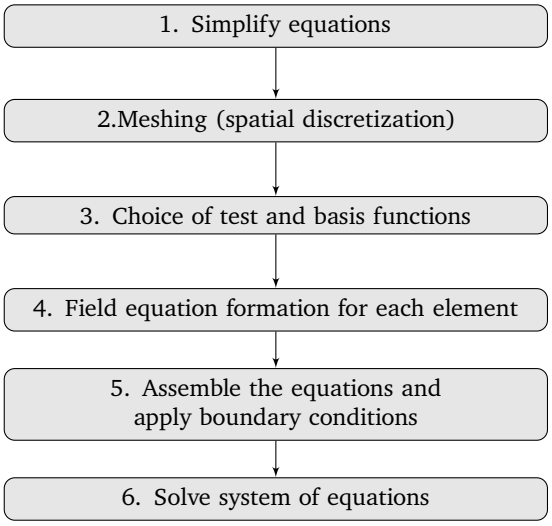


Figure 2.3: General FEM workflow for electrical machines [7, 184].

equations. Further simplification of (2.12) leads to Poisson's equation

$$-\nu \left(\frac{\partial^2 A_z}{\partial x^2} + \frac{\partial^2 A_z}{\partial y^2} \right) = J_{\text{total},z}, \tag{2.13}$$

where it is assumed here for simplicity that the reluctivity ν is space independent.

In the second step, complex geometry is divided into smaller, easily solvable entities (e.g., triangle, square, tetrahedron; see example of discretization in Figure 2.4) known as finite elements. The quality of discretization has a significant impact on the numerical accuracy. The finer the discretization in certain regions the higher the number of elements to solve, which results in a more accurate solution at the cost of computational time and resources.

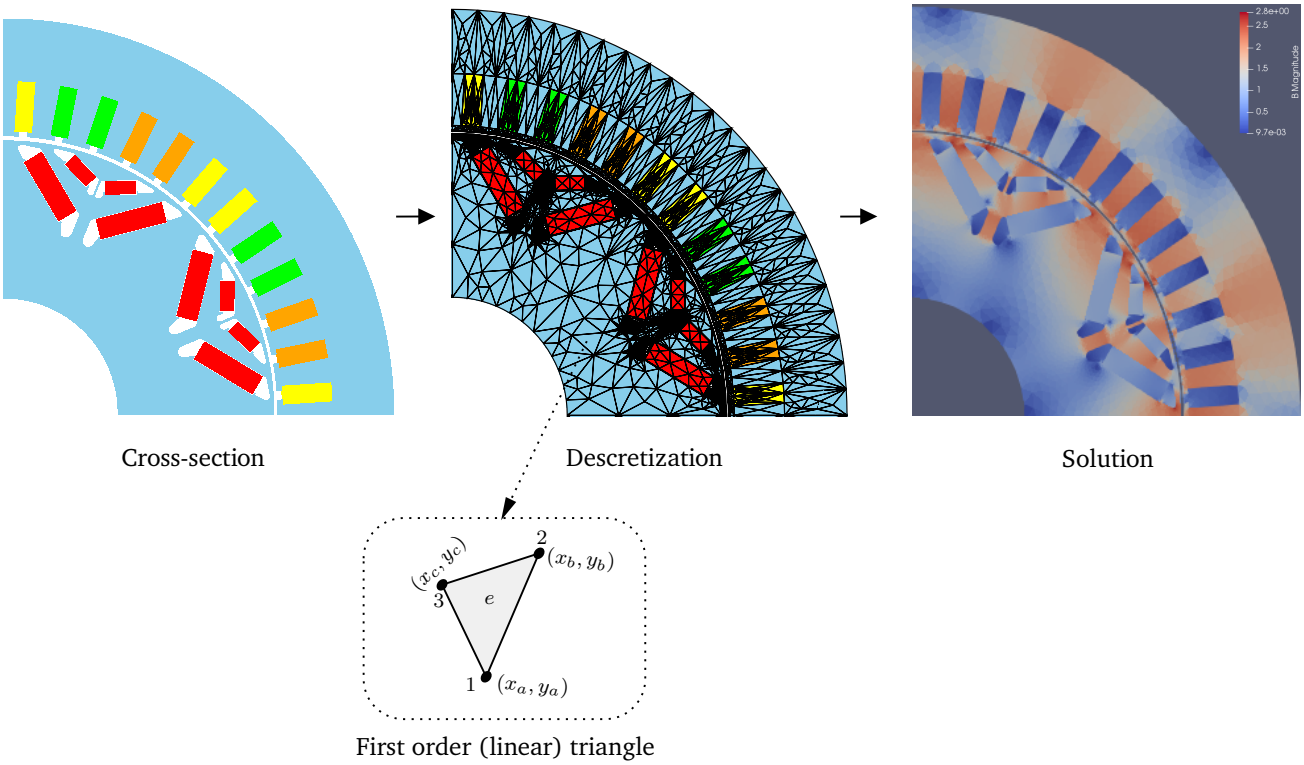


Figure 2.4: Illustration of PMSM geometry cross-section, discretization, and magnetic field solution.

The third step is selecting test and basis functions that approximate the solution. These functions are usually polynomial, and their element order can vary according to the need. The commonly used functions are linear, quadratic, and cubic. Higher-order functions achieve more accuracy, but at the same time, calculation time increases. For illustrative purposes, a single triangular element, denoted as e, is selected from the discretized geometry as shown in the Figure 2.4. If all elements are assumed to be linear, then unknown MVP $A_z^e(x, y)$ in the element e is approximated as

$$A_z^e(x, y) = g_1^e + g_2^e x + g_3^e y, \tag{2.14}$$

where g_1^e, g_2^e, g_3^e are constant coefficients to be determined. Due to the linear variation of the MVP, the flux density, which represents the derivative of the MVP, remains constant within the triangle. There are three nodes $(1, 2, 3)$ at each vertex of a linear triangle, and the values of the MVP at each node are $A_{z,1}^e(x_a, y_a), A_{z,2}^e(x_b, y_b), A_{z,3}^e(x_c, y_c)$, that is described by (2.14)

$$\begin{aligned} A_{z,1}^e(x_a, y_a) &= g_1^e + g_2^e x_a^e + g_3^e y_a^e \\ A_{z,2}^e(x_b, y_b) &= g_1^e + g_2^e x_b^e + g_3^e y_b^e \\ A_{z,3}^e(x_c, y_c) &= g_1^e + g_2^e x_c^e + g_3^e y_c^e \end{aligned} \tag{2.15}$$

Following the explanation provided in [184, Sec. 1.2], after terms are solved and rearranged, the MVP is obtained as follows:

$$A_z^e(x, y) = \sum_{j=1}^{n} N_j^e(x, y) \cdot A_{z,j}^e, \tag{2.16}$$

where n represents the number of nodes per element (three in the displayed element), and the node coordinates denoted as (x, y) define the spatial location of each node. The basis function ($N_j^e(x, y)$) for any node j is given by

$$N_j^e(x,y) = \frac{1}{2\Delta^e}\left(g_{1,j}^e + g_{2,j}^e x + g_{3,j}^e y\right), \tag{2.17}$$

in which Δ^e is the area of the e_{th} triangle and the coefficients, for example, node $N_1^e(x, y)$ are determined by

$$\begin{aligned} g_{1,1} &= x_b y_c - y_b x_c, \\ g_{2,1} &= y_b - y_c, \\ g_{3,1} &= x_c - x_b, \end{aligned} \tag{2.18}$$

Similar to $N_1^e(x, y)$, coefficients of $N_2^e(x, y)$ and $N_3^e(x, y)$ can be determined. As described in [184, Sec. 1.2], when the hat function is utilized as the basis function, it has a value of one at the specified node j and zero at all other nodes. The sum of all basis functions at any node in the triangular element is one.

The fourth step is the formulation of the field equation per element. In this explanation, the Galerkin approach is followed as described in [184, Sec. 1.2]. Suppose $\hat{A}_z$ is an approximate solution; then equation (2.13) can be written as:

$$\nu\left(\frac{\partial^2 \hat{A}_z}{\partial x^2} + \frac{\partial^2 \hat{A}_z}{\partial y^2}\right) + J_{\text{total},z} = \Gamma(\hat{A}_z), \tag{2.19}$$

where $\Gamma(\hat{A}_z)$ represents the residual. The residual is multiplied by a test function (φ) and the integral over the computational domain Ω is set to zero:

$$\iint_{\Omega} \Gamma(\hat{A}_z)\varphi \, \mathrm{d}x\mathrm{d}y = 0. \tag{2.20}$$

This is called the weighted residual. Substituting (2.19) in (2.20)

$$-\iint_{\Omega} \nu\left(\frac{\partial^2 \hat{A}_z}{\partial x^2} + \frac{\partial^2 \hat{A}_z}{\partial y^2}\right)\varphi \, \mathrm{d}x \, \mathrm{d}y = \iint_{\Omega} J_{\text{total},z}\varphi \, \mathrm{d}x \, \mathrm{d}y. \tag{2.21}$$

Then, integrating the left hand side term by parts,

$$\begin{aligned} -\iint_{\Omega} \nu\left(\frac{\partial^2 \hat{A}_z}{\partial x^2} + \frac{\partial^2 \hat{A}_z}{\partial y^2}\right)\varphi \, \mathrm{d}x \, \mathrm{d}y = \iint_{\Omega} \nu\left(\left(\frac{\partial \varphi}{\partial x}\right)\left(\frac{\partial \hat{A}_z}{\partial x}\right) + \left(\frac{\partial \varphi}{\partial y}\right)\left(\frac{\partial \hat{A}_z}{\partial y}\right)\right) \mathrm{d}x \, \mathrm{d}y \\ - \oint_{\mathcal{C}} \nu\varphi \frac{\partial \hat{A}_z}{\partial \hat{n}} \, \mathrm{d}\mathcal{C} \end{aligned} \tag{2.22}$$

The line integral term is to be calculated over elements that share a common side with the boundary $\mathcal{C}$. The term $\frac{\partial \hat{A}_z}{\partial \hat{n}}$ represents Neumann boundary conditions, where $\hat{n}$ is the outward unit normal vector.

Homogeneous Dirichlet or Neumann boundary conditions are assumed for simplification; therefore, the boundary integral vanishes. Equation (2.22) can be rewritten as

$$\iint_{\Omega} \nu \left(\left(\frac{\partial \varphi}{\partial x} \right) \left(\frac{\partial \hat{A}_z}{\partial x} \right) + \left(\frac{\partial \varphi}{\partial y} \right) \left(\frac{\partial \hat{A}_z}{\partial y} \right) \right) \mathrm{d}x \, \mathrm{d}y = \iint_{\Omega} J_{\mathrm{total},z} \varphi \, \mathrm{d}x \, \mathrm{d}y. \tag{2.23}$$

From this point onwards, for better readability, is $\hat{A}_z$ will simply be referred as A. The region Ω is discretized into $i = 1, \ldots, N$ number of linear triangle elements and then (2.23) is applied over each element individually. Afterward, the summation of the integral is taken as described below

$$\sum_{i=1}^{N} \left[\nu \iint_{\Omega^i} \left(\left(\frac{\partial \varphi^i}{\partial x} \right) \left(\frac{\partial A^i}{\partial x} \right) + \left(\frac{\partial \varphi^i}{\partial y} \right) \left(\frac{\partial A^i}{\partial y} \right) \right) \mathrm{d}x \, \mathrm{d}y - \iint_{\Omega^i} J_{\mathrm{total},z} \varphi^i \, \mathrm{d}x \, \mathrm{d}y \right] = 0, \tag{2.24}$$

over each element, where Ω^i is the domain of the i_{th} element. The test (weighting) function (φ^i) is selected to be identical to the basis function mentioned in (2.17) in order to follow the Galerkin approach. The values of the MVP (A) need to be determined only in the nodes of each element. Substituting (2.16) in (2.24)

$$\begin{aligned} &\sum_{i=1}^{N} \left[\sum_{j=1}^{n} \sum_{k=1}^{n} \left[A_j^i \iint_{\Omega^i} \nu \left(\left(\frac{\partial N_k^i}{\partial x} \right) \left(\frac{\partial N_j^i}{\partial x} \right) + \left(\frac{\partial N_k^i}{\partial y} \right) \left(\frac{\partial N_j^i}{\partial y} \right) \right) \mathrm{d}x \, \mathrm{d}y \right] \right] \\ &- \sum_{i=1}^{N} \left[\sum_{k=1}^{n} \left[\iint_{\Omega^i} J_{\mathrm{total},z} N_k^i \, \mathrm{d}x \, \mathrm{d}y \right] \right] = 0. \end{aligned} \tag{2.25}$$

where $j = 1, \ldots, n$ denotes the index of the nodes used for approximating the solution within each element, with n being the total number of nodes per element, and $k = 1, \ldots, n$ represents the index for the test functions associated with each node of the element.

The fifth step is assembling all the element equations into one system of equations using the connectivity matrix. Equation (2.25) is rewritten as

$$\sum_{i=1}^{N} ([\mathbf{K}^i][\mathbf{A}^i] - [\mathbf{T}^i]) = 0 \tag{2.26}$$

$$\begin{aligned} \text{where} \quad [\mathbf{A}^i] &= \left[A_1^i, \ldots, A_j^i, \ldots, A_n^i \right]^\top, \\ T_k^i &= \iint_{\Omega^i} J_{\mathrm{total},z} N_k^i \, \mathrm{d}x \, \mathrm{d}y, \\ K_{j,k}^i &= \iint_{\Omega^i} \nu \left(\left(\frac{\partial N_k^i}{\partial x} \right) \left(\frac{\partial N_j^i}{\partial x} \right) + \left(\frac{\partial N_k^i}{\partial y} \right) \left(\frac{\partial N_j^i}{\partial y} \right) \right) \mathrm{d}x \, \mathrm{d}y, \end{aligned}$$

where T_k^i denotes the kth entry of the vector $[\mathbf{T}^i]$ and $K_{j,k}^i$ represents the (j,k) entry of the matrix $[\mathbf{K}^i]$. Then, system of equations can be written as

$$\mathbf{K}\mathbf{A} = \mathbf{T}, \tag{2.27}$$

where the matrix $\mathbf{K}$ is constructed by assembling the elements $[\mathbf{K}^i]$, the vector $\mathbf{A}$ is formed from the elements $[\mathbf{A}^i]$, and the vector $\mathbf{T}$ is created using the elements $[\mathbf{T}^i]$. The matrix $\mathbf{K}$ is called the "stiffness matrix".

After forming the system of equations, boundary conditions are applied to describe the field behavior on the model's boundaries (or at the interfaces between different materials). Popular examples of boundary conditions are the Dirichlet boundary condition and the Neumann boundary condition.

- **Dirichlet boundary condition** is also called the essential boundary condition. It specifies the value of the field potential (here, the MVP) at the boundary ($\mathcal{C}$), which corresponds to the computational domain (Ω). The homogeneous Dirichlet boundary condition can be expressed as

$$\vec{A}(\vec{r}) = 0, \ \forall \vec{r} \text{ on } \mathcal{C}, \tag{2.28}$$

  where $\vec{r} = (x, y, z)$ spatial vector and $\vec{A}(\vec{r})$ is the MVP value. If $\vec{A}(\vec{r}) = c$, where c is the constant value on the boundary and $c \neq 0$, then it is a non-homogeneous Dirichlet boundary condition.

- **Neumann boundary condition** is called the natural boundary condition. It specifies the normal derivative of the MVP at the boundary of a domain. The homogeneous Neumann boundary condition can be expressed as

$$\frac{\partial \vec{A}(\vec{r})}{\partial \vec{n}} = 0, \ \forall \vec{r} \text{ on } \mathcal{C}, \tag{2.29}$$

  where $\vec{n}$ is the outward unit normal vector. If $\frac{\partial \vec{A}(\vec{r})}{\partial \vec{n}} = c$, where c is the constant value on the boundary and $c \neq 0$, then it is a non-homogeneous Neumann boundary condition.

There are other boundary conditions such as periodic and anti-periodic boundary conditions; refer to [7] for more details.

The sixth step is to solve the global system of equations (2.27). If the system is linear then direct solvers (e.g., Gaussian Elimination or Cholesky Decomposition) or an iterative solver such as the generalized minimal residual method [178] can be used. For a non-linear system, an iterative approach like the Newton-Raphson method can be employed (refer to [184, Chapter 2]). After solving the system of equations, post-processing is done to obtain the quantity of interest (e.g., magnetic flux density, torque, and iron losses).

2.2.3 Simulation process of calculating KPIs for PMSM

In this study, ground truth data for different topologies of PMSMs is generated using 2D magnetostatic FE simulation [184]. Therefore, in this subsection, the industrial simulation flow to evaluate KPIs for PMSM will be explained. Figure 2.5 gives a high-level gist of the simulation workflow for the PMSM.

Design parameters: The simulation process starts with defining design parameters. Any parameter that directly or indirectly affects the construction and performance of the electrical machine is called a design parameter. In this thesis, the design parameters are broadly classified into three categories:

- **Geometry and topology:** The set of parameters defines the shape and size of an electrical machine in space. These parameters include, for example, stator dimensions (i.e., outer diameter, tooth height, tooth width, etc.), rotor dimensions (i.e., diameter, magnet pocket size, etc.), machine axial length, air-gap length, magnet height, magnet width, pole pairs and so on. Some influential parameters are depicted in Figure 2.6. Every parameter has an impact on the final KPIs of the machine. For instance,

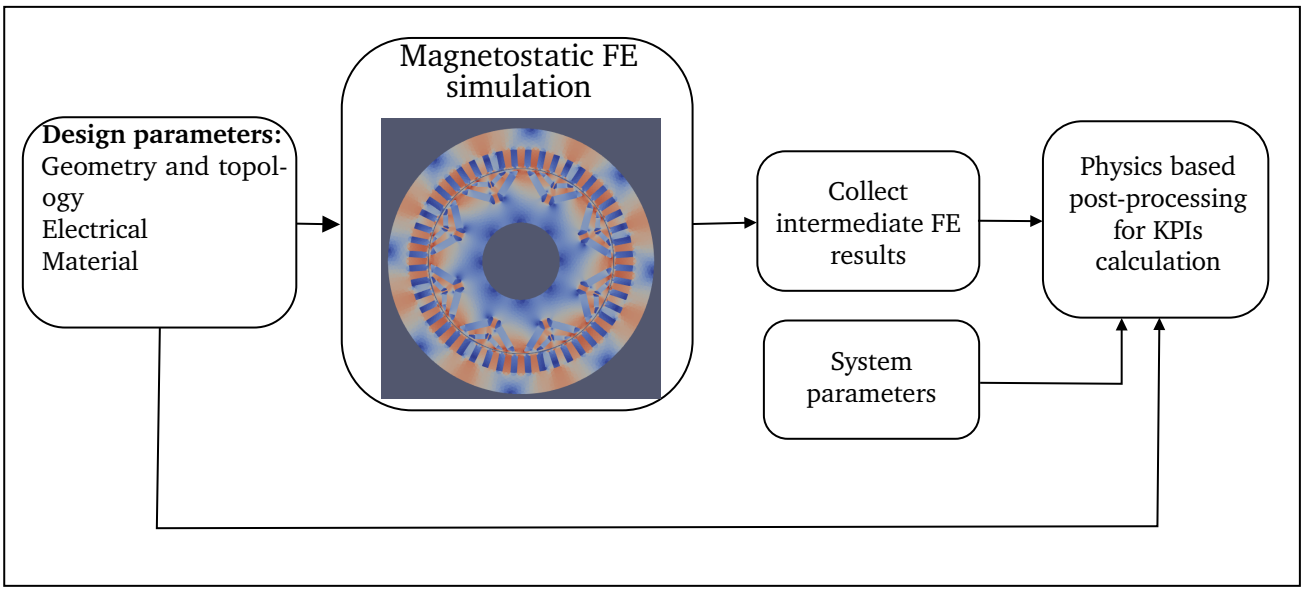


Figure 2.5: Diagram for the calculating of electrical machine design KPIs.

the torque density of the PMSM to some extent is dependent on its axial length and stator outer diameter; however, it decreases with an increase in air-gap length and the number of poles [143]. Sensitivity analysis quantifies the impact of parameters on the machine's performance. For further details, refer to the study in [148].

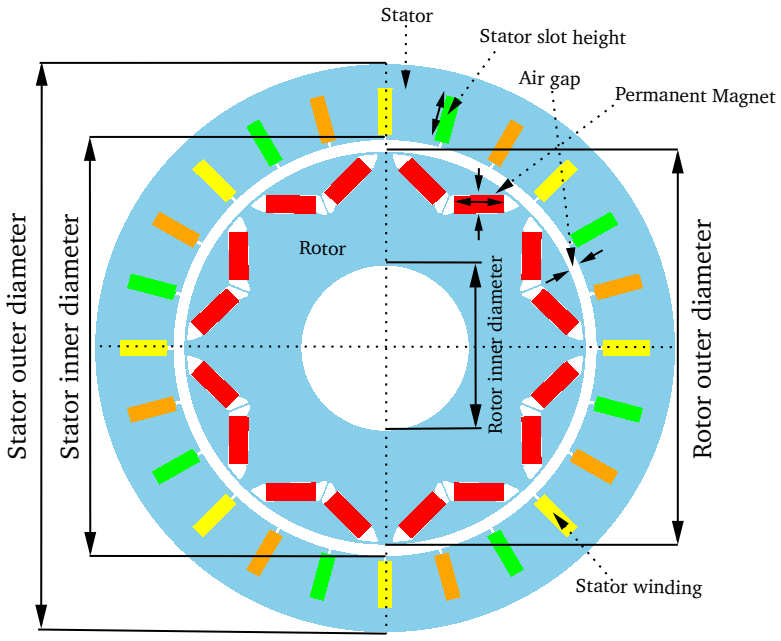


Figure 2.6: Illustrative PMSM cross-section (8 poles/24 slots).

- **Electrical:** Electrical parameters specify the electrical excitation of the machine. These parameters usually determine the steady-state and dynamic behaviour of the machine. Some key electrical

parameters are input voltage, input coil current, resistance, winding type (short pitch or full pitch), power supply connection (star or delta), power supply frequency, etc.

- **Material:** Material parameters in electrical machines are linked to the properties of the materials used to build the machine, such as copper, aluminium, iron and magnets. These parameters considerably affect the machine's performance, efficiency, and reliability. Some examples of magnet material parameters include the type, weight, density, remanence, and relative permeability; other material parameters include the iron stacking factor, iron correction factor, lamination type, and copper filling factor.

Magnetostatic-FE simulation: After defining the design variables, a geometrically consistent computer-aided design (CAD) model is first built. Here, geometric consistency means a manufacturable CAD model without any distortions, e.g., no intersections in geometry. A CAD model is then meshed with sufficiently many elements, guided either by experience or driven by an error estimator [14, 184]; for example, as shown in Figure 2.4. Each machine design can be associated with a design vector $\mathbf{p} \in \mathbb{P} \subset \mathbb{R}^n$, where n is the dimension of the input space $\mathbb{P}$ which is restrained by constraints, e.g. to ensure geometric consistency. The design vector for the l_{th} design during the simulation can be represented as $\mathbf{p}^{(l)} = [p_1^{(l)}, p_2^{(l)}, \ldots, p_n^{(l)}]$. For each design $\mathbf{p}^{(l)}$, the magnetostatic FE approximation is performed on a 2D parameterized geometry $\Omega(\mathbf{p}) \subset \mathbb{R}^2$. From the work [138, 152, 184], rewriting (2.12),

$$-\nabla \cdot (\nu \nabla A_z) = J_{\text{s},z} + \nabla \times \mathbf{M} \cdot \mathbf{e}_z, \tag{2.30}$$

where the z-component of the MVP $\vec{A}$, the source current density $J_{\text{s},z}$, and the permanent magnets' magnetization $(\mathbf{M})$ depends on the input vector $(\mathbf{p})$. The FEM computes the MVP $(\vec{A})$ from which intermediate FE results can be derived.

Intermediate results: From the MVP $(\vec{A})$, different quantities of interest such as magnetic flux density, flux linkage on a winding, non-linear iron losses, and electromagnetic torque can now be calculated. From (2.8), the relation between magnetic flux density $\vec{B}$ and $\vec{A}$ is known. The magnetic flux (Φ) over a computational domain Ω can be calculated as

$$\Phi = \iint_{\Omega} \nabla \times \vec{A} \, \mathrm{d}\Omega. \tag{2.31}$$

Rewriting (2.31) in line integral form by considering enclosed boundary $(\mathcal{C})$ over the region using Stokes theorem as

$$\Phi = \oint_{\mathcal{C}} \vec{A} \, \mathrm{d}\mathcal{C}. \tag{2.32}$$

The magnetic flux linkage (ψ) on a stator winding can be calculated by

$$\psi = \iint_{\Omega} \vec{\chi} \cdot \vec{A} \, \mathrm{d}\Omega, \tag{2.33}$$

as explained in [191, Equation 16]. Where $\vec{\chi}$ is the winding function. The other intermediate measure, the electromagnetic torque $(\mathcal{T}_e)$ applied on the rotor, can be calculated using the Maxwell stress method [113, 165, 184] [89, Section 1.5], as written below:

$$\mathcal{T}_e = \iint_{\Omega_{rot}} \vec{r} \times (\sigma \cdot \hat{n}) \, \mathrm{d}\Omega_{rot}, \tag{2.34}$$

where Ω_{rot} represents the surface surrounding the rotor, $\hat{n}$ denotes unit normal vector, The vector $\vec{r}$ is the position vector that connects the surface (Ω_{rot}) to the rotor origin. The Maxwell stress tensor (σ) [184, Chapter 6.3] is given by

$$\sigma_{ij} = \nu_0 \left(B_i B_j - \frac{1}{2}\delta_{ij}|B|^2 \right), \tag{2.35}$$

where ν_0 denotes reluctivity in the air, δ_{ij} is the Kronecker delta and $i, j = 1, 2, 3$.
The non-linear iron losses (P_{fe}) comprise hysteresis and eddy current losses. They are calculated at base speed ($n_{\text{rpm,base}}$) and base machine length (l_{base}) according to Jordan's model [171] as over the computation region Ω,

$$P_{\text{fe,hys,base}} = \sum_k \left[K_{\text{hys}} f_k^{\beta} \iint_{\Omega} \left(\hat{B}_{x,k}^{\gamma}(x,y) + \hat{B}_{y,k}^{\gamma}(x,y) \right) \mathrm{d}\Omega \right] \quad \text{and} \tag{2.36}$$

$$P_{\text{fe,eddy,base}} = \sum_k \left[K_{\text{eddy}} f_k^{\delta} \iint_{\Omega} \left(\hat{B}_{x,k}^{\xi}(x,y) + \hat{B}_{y,k}^{\xi}(x,y) \right) \mathrm{d}\Omega \right],$$

where f_k is the operating frequency in k_{th} harmonic order, $P_{\text{fe,hys,base}}$ is the hysteresis loss and $P_{\text{fe,eddy,base}}$ describe the eddy current loss at the base speed and length, the coefficients K_{hys} and K_{eddy} are calculated using the base value of material measurement data, i.e., nominal conductivity ($\sigma_{\text{hys,base}}$, $\sigma_{\text{eddy,base}}$) and the constant values, e.g., $\gamma = 2$, $\beta = 1$, $\xi = 2$, and $\delta = 2$ are set in this thesis for data generation. The magnetic flux density $\hat{B}$ is integrated over regions and calculated in the frequency domain for each frequency harmonic.

The input design is simulated for one electrical period at each defined operating point. The electrical machine's operating point is a variable electrical excitation input. The input phase current and corresponding control angle together define the operating point. The control angle is the angle between the excitation current and no-load induced voltage (back electromotive force (EMF)). The intermediate measures, i.e., non-linear iron losses, electromagnetic torque, and flux linkages for all the operating points are evaluated.

Physics based post-processing: Following the time-consuming FE simulation, the results are post-processed together with the system parameters to calculate KPIs using physics-based models. A few examples of system parameters are inverter input voltage, current, rotational speed (n_{rpm}) as the desired operating point, ohmic resistance (R), and scaling parameters (e.g., machine length). The basic block diagram for the electrical drive system is shown in Figure 2.7, which displays certain system parameters (e.g., inverter input DC current and voltage). The KPIs quantify the performance of electrical machine design. It may include maximum torque, shaft power, copper loss, material cost, efficiency and so on. A few examples are given in this subsection. To calculate the torque and power related KPIs under a specified load, the FEM computed three-phase coil flux-linkages (ψ) are first converted to the dq-frame, denoted by ψ_d and ψ_q, using the Park transformation [156]. In this context, the d-axis, designated as the direct axis, aligns with the magnet's magnetic field, while the q-axis, the quadrature axis, is perpendicular to it. The inner torque of the electrical machine, also denoted as electromagnetic torque (T_{i}), for a given input three-phase line currents, can be described by:

$$T_{\text{i}} = \frac{1}{2} mp(\psi_d i_q - \psi_q i_d) = \frac{1}{2} mp(\psi_{\text{pm}} i_q + (L_d - L_q) i_d i_q), \tag{2.37}$$

as described in [140, 220], where m is the number of phases (usually 3-phase), p is the number of pole pairs (input topology parameter). The d-axis and q-axis currents, denoted as i_d and i_q respectively, are

obtained in the dq-frame by applying the Park transformation to the three-phase currents. L_d and L_q represent the d-axis and q-axis inductances, respectively. They are derived from the intermediate results using the formulas: $L_d = \frac{\psi_d - \psi_{\mathrm{pm}}}{i_d}$ and $L_q = \frac{\psi_q}{i_q}$. Here, ψ_{pm} denotes the permanent magnet flux linkage, which is also an output of the FE results.

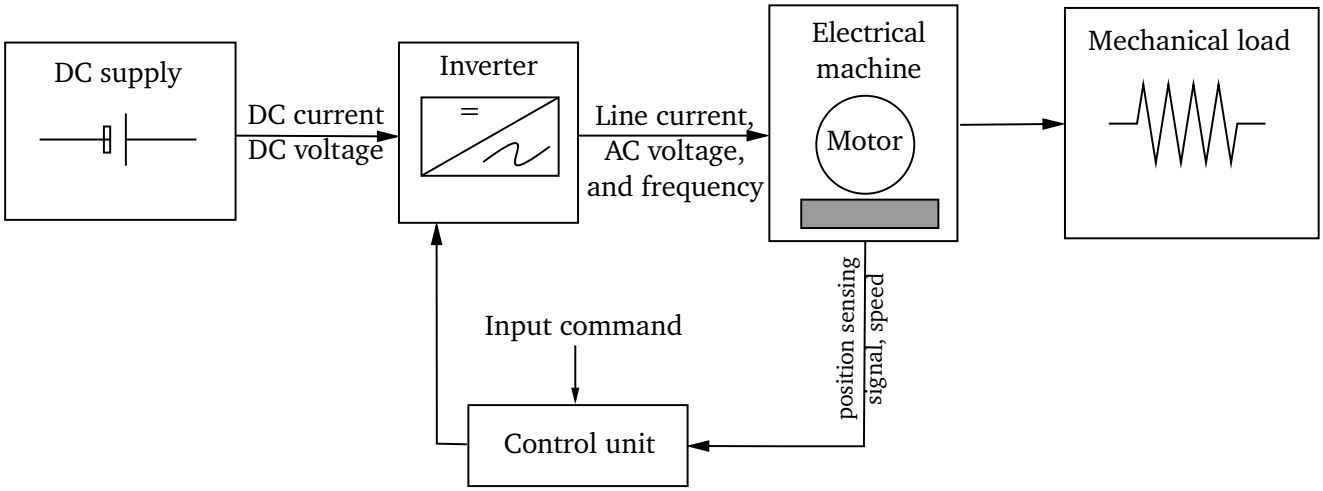


Figure 2.7: General block diagram of electrical drive system.

Other KPIs such as the copper loss and the inner power without losses are calculated by

$$P_{\mathrm{CU}} = mI^2R \quad \text{and} \quad P_{\mathrm{i}} = \frac{2\pi n_{\mathrm{rpm}} T_{\mathrm{i}}}{60}, \tag{2.38}$$

where T_{i} is the inner torque. The shaft power (P_{shaft}) is the mechanical power transferred from an electrical machine at a given speed (n_{rpm}). It can be calculated from the inner power P_{i} using the formula $P_{\mathrm{shaft}} = P_{\mathrm{i}} - P_{\mathrm{fe}} - P_{\mathrm{friction}}$. In this formula, P_{fe} represents the iron losses, while P_{friction} accounts for power losses due to mechanical friction, which may occur due to bearings, windage (air resistance), and other sources of mechanical friction within the electrical drive system. Given the speed vector (N) and torque vector ($\mathcal{T}$), the shaft power P_{shaft} for the given operation points (speed-torque combination) can be calculated. Ultimately, the maximum shaft power can be calculated by taking the maximum of all calculated powers.

Torque ripple deviation is the difference between the torque ripple and the average target torque T_{avg} over one electrical period. The torque ripple is calculated by

$$T_{\mathrm{ripple}} := T_{\mathrm{i,max}} - T_{\mathrm{i,min}}. \tag{2.39}$$

Here, $T_{\mathrm{i,max}}$ is maximum electromagnetic torque and $T_{\mathrm{i,min}}$ is minimum electromagnetic torque over one electrical period.

The efficiency of an electrical machine at a given torque and speed combination is derived from the mapping of FEM results in relation to d-axis and q-axis currents. Essentially, it is the ratio of the output mechanical power to average input electrical power during one cycle, applicable in both motor and generator modes. The efficiency of the electrical machine η_{E}, at any given operating point described as a torque and rotor speed, can be quantified by

$$\eta_{\mathrm{E}} = \frac{P_{\mathrm{shaft}}}{P_{\mathrm{el}}}, \tag{2.40}$$

where P_{el} is input electrical power.

During post-processing, the iron losses can be scaled with desired speed (n_{rpm}), machine length (l), and material physical values ($\sigma_{\mathrm{hys}},\ \sigma_{\mathrm{eddy}}$) as

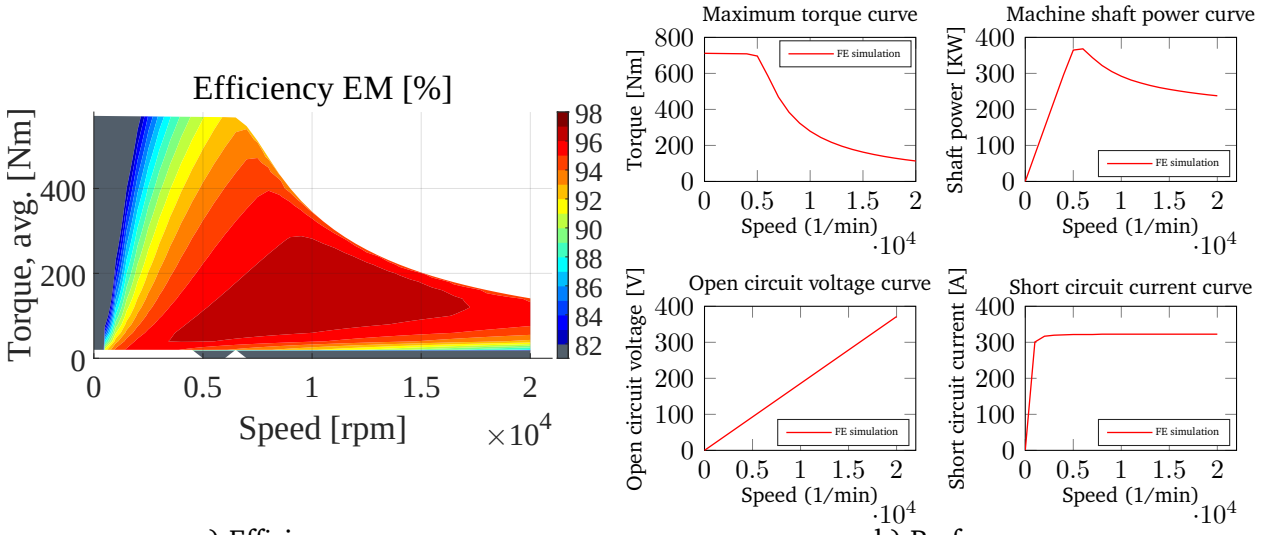
$$P_{\mathrm{fe,hys},op} = P_{\mathrm{fe,hys,base}} \cdot \left(\frac{l}{l_{\mathrm{base}}} \frac{f}{f_{\mathrm{base}}} \frac{\sigma_{\mathrm{hys}}}{\sigma_{\mathrm{hys,base}}} \right) \quad \text{and} \tag{2.41}$$

$$P_{\mathrm{fe,eddy},op} = P_{\mathrm{fe,eddy,base}} \cdot \left(\frac{l}{l_{\mathrm{base}}} \left(\frac{f}{f_{\mathrm{base}}} \right)^2 \frac{\sigma_{\mathrm{eddy}}}{\sigma_{\mathrm{eddy,base}}} \right),$$

where $f = \frac{n_{\mathrm{rpm}} p}{60}$, p is number of poles. One such example of scaling losses of IM is described in [145].

During the post-processing, in addition to single-valued KPIs, complex performance measures are computed when needed. Figure 2.8 demonstrates complex performance measures, such as the efficiency map and various performance curves, for one sample of a PMSM. In Figure 2.8a, the efficiency map plot is a graphical depiction of the PMSM's maximum efficiency at various speed and torque combinations. Each point on the map corresponds to a specific rotor speed and torque, with the color indicating the efficiency. This plot facilitates the analysis of the PMSM's performance, highlighting the most efficient operational regions across various conditions [135].

The maximum torque curve represents a performance curve (in Figure 2.8b) with the maximum torque for each given rotational speed. It is a crucial characteristic that defines the operating limits of the motor, showing where the torque starts to drop off as speed increases. To obtain this, the physical post-processing requires the torque matrices $\mathcal{T}_e$ (output of FEM intermediate results) for the interpolation. Then, on the interpolated torque matrices with the help of control strategies such as maximum-torque-per-ampere, flux weakening, and maximum-torque-per-volt for example, as explained in [140], the optimal i_d and i_q combination is determined to maximize inner torque T_{i} for the given torque-speed combination, within the boundary constraints of the inverter's input current I_{L} and voltage V_{line}. In the post-processing, the numerical optimization is performed since a non-linear relationship between (ψ_d, ψ_q) and (i_d, i_q) (i.e., saturation) is considered. If a linear relationship is present (i.e., no saturation), then it can be obtained



a) Efficiency map b) Performance curves

Figure 2.8: Illustration of complex performance measures.

analytically. The maximum shaft power curve illustrates the maximum mechanical power that can be delivered by the motor's shaft at different rotor speeds.

The open-circuit voltage V_{oc} is the voltage induced across the motor terminals (windings) due to permanent magnets when the terminals of the electrical machine are open (i.e., $I_{\mathrm{L}} = 0$). It is described as $V_{\mathrm{oc}} = \omega\psi_{\mathrm{pm}}$, under the condition $i_d = i_q = 0$. Here, $\omega = \frac{2\pi n_{\mathrm{rpm}}}{60p}$ represents the electrical angular velocity. The open-circuit voltage curve, represented in Figure 2.8b, explains the behavior of the motor's induced voltage at various speeds without the influence of load.

During the short circuit condition, the terminals of the electrical machine are short-circuited, meaning that the terminal voltage, $V_{\mathrm{line}} = 0$. The permanent magnets induce a voltage in the windings, which drives a short-circuit current, I_{sc}. For simplicity in analysis, the voltage drop across the resistance can be neglected at medium and high rotor speeds (when ohmic resistance $R \ll \omega L_d L_q$). This results in the dq-axis voltage equations being described by

$$V_d = -\omega L_q i_q = -\omega\psi_q \quad \text{and} \quad V_q = \omega\psi_{\mathrm{pm}} + \omega L_d i_d = \omega\psi_d, \tag{2.42}$$

as given in [220, Equations (10-11)]. In this context, since $V_{\mathrm{line}} = 0$, both V_d and V_q are equal to zero. The simplification of voltage equations (2.42) yields the short-circuit current, denoted as $I_{\mathrm{sc}} = i_d = -\frac{\psi_{\mathrm{pm}}}{L_d}$. The short circuit current curve, depicted in Figure 2.8b, shows how the current behaves when the terminals are shorted for different rotor speeds. This curve can be useful for assessing the motor's behavior under fault conditions.

These are a few examples of electrical machine KPIs calculated by the post-processing. It should be noted that, compared to magneto-static FE simulation ($3-5$ hours/design), the post-processing step is much faster ($3-5$ minutes/design). For the sake of simplicity in this thesis, all the datasets are generated without considering additional harmonic losses. These losses can occur due to factors such as the total harmonic distortion caused by pulse width modulation and space harmonics.

2.3 Basics of optimization

In this section, the basic notions of optimization and the general mathematical framework for multi-objective optimization are introduced. The gradient-based approaches used for training DNNs, as well as the general population-based evolutionary algorithm employed for the MOO of rotating electrical machines in this work, are briefly explained.

2.3.1 General definition of optimization

Generally speaking, optimization refers to utilizing resources or circumstances optimally by selecting the best element from a range of options based on specific criteria. In mathematical terms, a problem typically involves maximizing or minimizing a function (known as the objective function) concerning a set of criteria called constraints [45]. For example, designing electrical machines by optimizing KPIs subject to constraints.

To describe a general, nonlinear MOO optimization problem, Suppose $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^m$, $\mathbf{e} : \mathbb{R}^d \to \mathbb{R}^n$, and $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^p$ be continuously differentiable functions. The MOO formulation as explained in [45, Chap.1]

is written by

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}} \quad & \mathbf{f}(\mathbf{x}), && (2.43)\\ \text{subject to} \quad & \mathbf{e}(\mathbf{x}) \leq \mathbf{0}, && (2.44)\\ & \mathbf{g}(\mathbf{x}) = \mathbf{0}, && (2.45)\\ & x_b^{\mathrm{L}} \leq x_b \leq x_b^{\mathrm{U}}, \quad b = 1, \ldots, d && (2.46) \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^d$ represents the continuous variables, $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}), \ldots, f_m(\mathbf{x})]$ is the row vector of objective functions to be minimized, $\mathbf{e}(\mathbf{x}) = [e_1(\mathbf{x}), \ldots, e_i(\mathbf{x}), \ldots, e_n(\mathbf{x})]$ is the row vector of inequality constraints, and $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_j(\mathbf{x}), \ldots, g_p(\mathbf{x})]$ is the row vector of equality constraints. The bounds for continuous variables are denoted as x_b^{L} and x_b^{U}. If $\mathbf{f}(\mathbf{x})$ contains only a single objective function, i.e., $m = 1$, then (2.43) is a single objective optimization. Equation 2.43 can be written as maximization problem if (2.43) is reformulated as below

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{maximize}} \quad -\mathbf{f}(\mathbf{x}). \tag{2.47}$$

If there are no constraints, meaning that the optimization problem only involves (2.43), then the problem is considered unconstrained. The bound constraint problem can be described by (2.43) and (2.46), where no inequality or equality constraints are included in the problem.

During the optimization process, the vector $\mathbf{x} \in \mathbb{R}^d$ is considered a feasible solution for the problem defined by (2.43)-(2.46) if it satisfies the conditions outlined in (2.44)-(2.46). The collection of all feasible solutions is referred to as the feasible set and represented as $\Omega \subseteq \mathbb{R}^d$ [59]. In the MOO, the best solution for one objective does not necessarily imply the best solution for other objectives. Therefore, the typical goal of MOO is to find a set of optimal solutions $\mathbf{x}_{\mathrm{opt}} \in \Omega$ that are Pareto optimal. As explained in [59], the Pareto optimal refers to a solution in MOO that cannot be improved in one objective without sacrificing the performance in at least one other objective. In other words, a solution is Pareto optimal if there is no other feasible solution $\mathbf{x}$ that can obtain better performance in all objectives simultaneously. i.e.,

$$\begin{aligned} & \mathbf{f}_k(\mathbf{x}) \leq \mathbf{f}_k(\mathbf{x}_{\mathrm{opt}}) && \text{for } k \in \{1, \ldots, m\} \\ \text{and} \quad & \mathbf{f}_l(\mathbf{x}) < \mathbf{f}_l(\mathbf{x}_{\mathrm{opt}}) && \text{for some } l \in \{1, \ldots, m\}. \end{aligned} \tag{2.48}$$

The set of Pareto optimal solutions is referred to as a *Pareto-front*. In Figure 2.9 the Pareto fronts for a bi-objective function are displayed. Figure 2.9a shows the Pareto front for objectives $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, which aim to be minimized simultaneously. On the other hand, Figure 2.9b shows the Pareto front for two competing objectives, where $f_1(\mathbf{x})$ is a minimization problem and $f_2(\mathbf{x})$ is a maximization problem.

2.3.2 Brief overview of optimization methods

There are numerous optimization algorithms available for solving different types of optimization problems [6, 13, 50, 93, 203]. In this thesis, gradient-based approaches are specifically used for training DNNs, and population-based evolutionary algorithms are employed for the MOO of rotating electrical machines. Therefore, in this subsection, a few gradient-based algorithms and a general framework for population-based evolutionary algorithms will be briefly introduced.

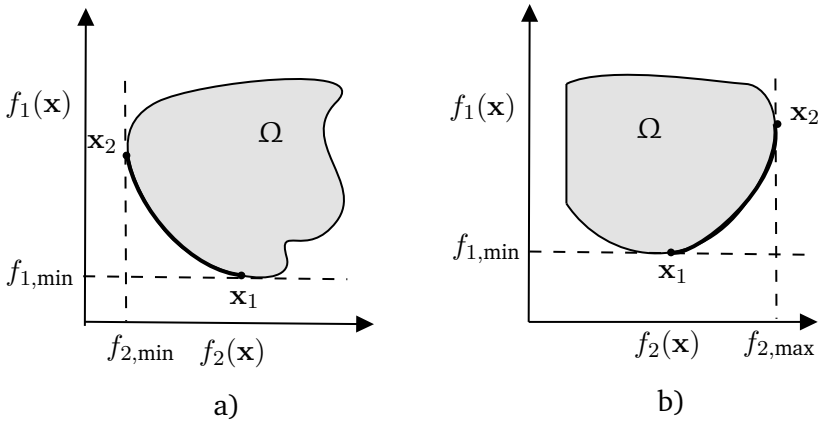


Figure 2.9: Illustration of Pareto-fronts for a bi-objective function: a) Min-Min and b) Min-Max.

2.3.2.1 Gradient based optimization methods

Generally, a gradient-based optimization algorithm iteratively finds the function's extremum by utilizing a gradient, representing the steepest descent (or ascent) direction at a given point. Usually, these algorithms use the function's first-order and second order derivative information to guide the search for optimal solutions. These algorithms converge toward the optimum of the function by repeatedly updating the current solution based on the gradient.

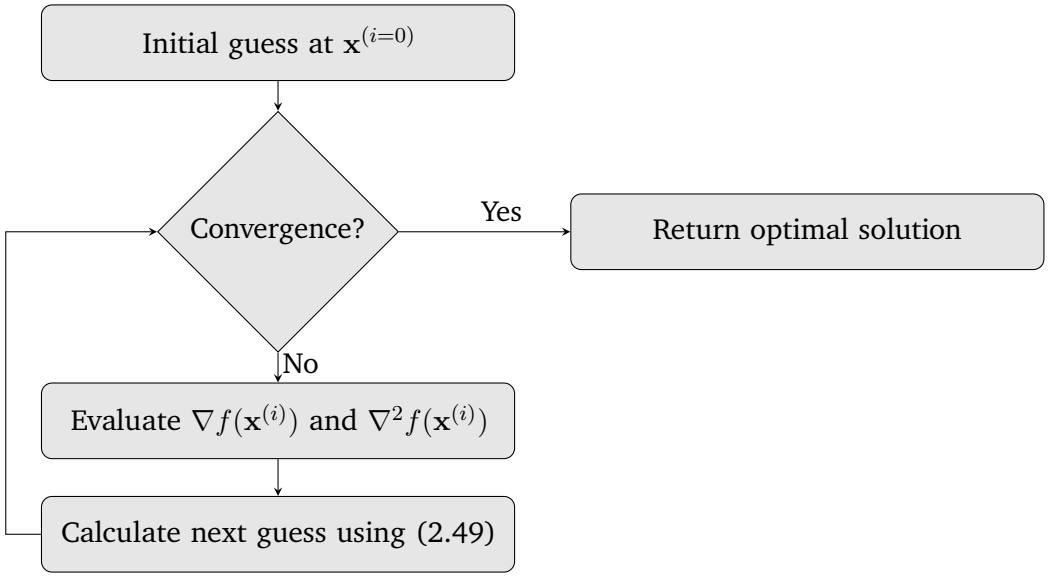


Figure 2.10: Flow chart of Newton's algorithm

Newton's method is an iterative, gradient-based optimization algorithm used to find the minimum or maximum of a given differentiable scalar objective function f. As explained in [147, Chapter 3], the goal is to find the optimal input $\mathbf{x}$ that satisfies the first-order optimality condition $\nabla f(\mathbf{x}) = \mathbf{0}$, where $\nabla f(\mathbf{x})$ represents the gradient. For a continuous and twice-differentiable nonlinear function $f : \mathbb{R}^n \to \mathbb{R}$, the

iterative scheme is formulated as follows:

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \left(\nabla^2 f(\mathbf{x}^{(i)})\right)^{-1} \nabla f(\mathbf{x}^{(i)}), \tag{2.49}$$

where $\mathbf{x}^{(i)}$ is the guess for the ith iteration, $\nabla f(\mathbf{x}^{(i)})$ is the gradient, and $\nabla^2 f(\mathbf{x}^{(i)})$ is the Hessian matrix. The updated guess for the next iteration is given by $\mathbf{x}^{(i+1)}$. The convergence criterion is typically defined by a small tolerance limit ϵ with $|\nabla f(\mathbf{x}^{(i)})| \leq \epsilon$. Figure 2.10 displays the Newton's algorithm in a flow chart form.

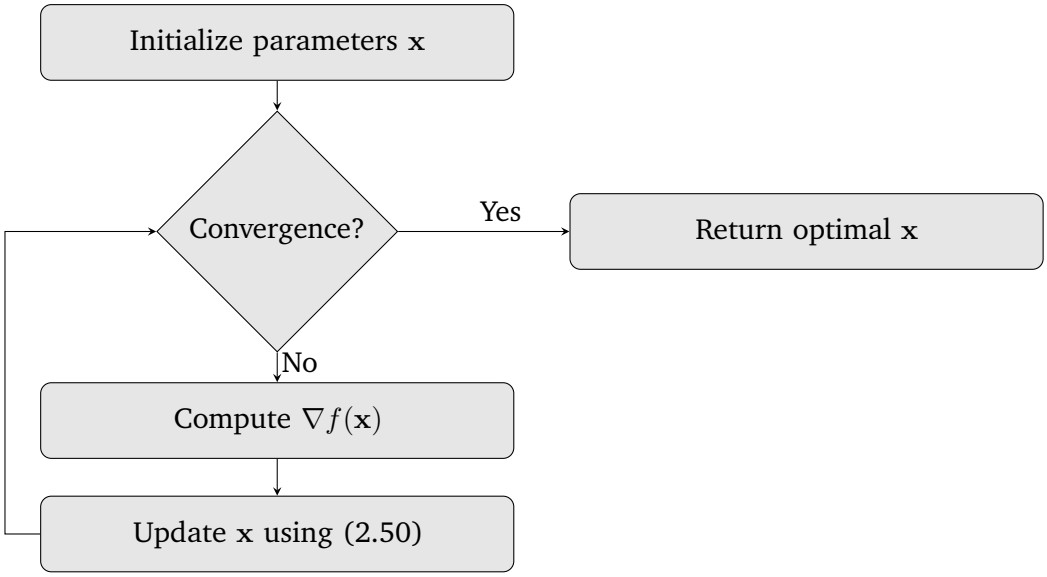


Figure 2.11: Flow chart of gradient descent algorithm

Gradient descent [4] is a gradient-based optimization algorithm with applications spanning various domains such as signal processing, deep learning, and more. It is commonly used in the DL field for training DNNs (for further details, refer to Sec. 3.2.1.1). It minimizes a DNN's loss function by iteratively updating the training parameters (weights and biases). This iterative process helps in obtaining parameter values that yield the lowest possible loss. As illustrated in Figure 2.11, the algorithm starts with an initialization of continuous variables $\mathbf{x}$, for example, a DNN training parameters. Then, the gradient of an differentiable objective function (e.g., a DNN loss function) is computed with respect to these parameters $\mathbf{x}$. The gradient indicates the direction of the steepest ascent, so to minimize the objective function, the parameters are updated in the opposite direction of the gradient using

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}), \tag{2.50}$$

where $f(\mathbf{x})$ is the objective function, positive scalar α explains learning rate, and i indicates the current training iteration. The learning rate α controls the change in the value of parameters to determine the size of the steps taken during each iteration. Small α leads to slower convergence but potentially more accurate results, whereas a large α can speed up convergence with the risk of overshooting the optimal values. The algorithm continues to iterate until a convergence criterion is satisfied, which can be determined by reaching a maximum number of iterations or applying regularization techniques like early stopping. Popular variants of gradient descent include stochastic gradient descent (SGD) (which updates $\mathbf{x}$ for each training sample) and mini-batch gradient descent (which processes a small batch of

training samples for updating $\mathbf{x}$). From (2.50), it can be seen that gradient descent requires the manual setting of a learning rate that determines the size of each step, which may impact the speed and success of convergence. On the other hand, Newton's method (2.49) automatically determines the step size based on the function's second-order derivative. It may often lead to larger steps when far from the minimum and smaller steps when near, but it requires more computational effort.

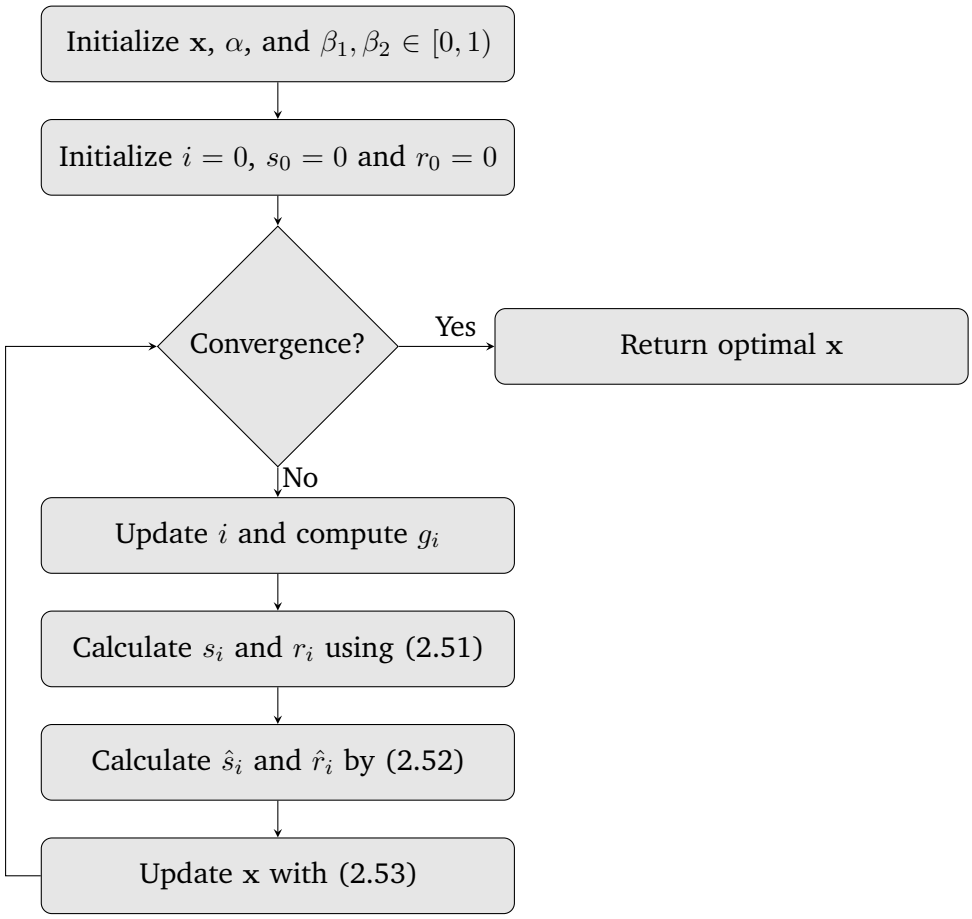


Figure 2.12: Flow chart of Adam algorithm based on Algorithm 1 in [104].

Adaptive Moment Estimation (Adam) [104] is a first-order gradient-based optimization technique that is widely employed for training DNNs in this thesis due to several advantages such as fast convergence, ability to deal with sparse gradients, handling of high dimensional space and large datasets, and invariance to rescaling of gradients. Adam can explore the optimization region by adapting the learning rates individually for each parameter using the first and second moments of gradients. Consider an objective function $f(\mathbf{x})$, which is stochastic and differentiable with respect to the parameters $\mathbf{x}$. The first gradient of the objective function $f(\mathbf{x})$ can be calculated as $g_i = \nabla_{\mathbf{x}} f_i(\mathbf{x})$ at iteration i. As explained in Algorithm 1 of [104], the first moment s_i (the mean) and second moment r_i (the uncentered variance) at iteration i are estimated by

$$\begin{aligned} s_i &= \beta_1 s_{i-1} + (1-\beta_1) g_i \quad \text{and} \\ r_i &= \beta_2 r_{i-1} + (1-\beta_2) g_i \odot g_i, \end{aligned} \tag{2.51}$$

where hyperparameters $\beta_1, \beta_2 \in [0, 1)$ are exponential decay rates, $\odot$ denotes element-wise dot product. Initialization of s_i and r_i with zero at $i = 0$ make them biased towards zero, and can be corrected with

$$\hat{s}_i = \frac{s_i}{1 - \beta_1^i} \quad \text{and}$$
$$\hat{r}_i = \frac{r_i}{1 - \beta_2^i}. \tag{2.52}$$

$\hat{s}_i$ and $\hat{r}_i$ are bias-corrected estimation. Then, the parameters $\mathbf{x}_i$ are updated with

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \alpha \frac{\hat{s}_i}{\sqrt{\hat{r}_i} + \epsilon}, \tag{2.53}$$

where the learning rate α is a hyperparameter that determines the step size of the update, and ϵ is a very small value (10^{-8}) for numerical stability. The Adam algorithm is illustrated in Figure 2.12.

2.3.2.2 Population based optimization method

Population-based optimization methods are powerful techniques for solving complex optimization problems with many conflicting objectives and are widely used in different fields such as engineering, healthcare, and finance [203]. They do not rely on gradients but draw inspiration from natural processes such as biological evolution and swarm intelligence. Some popular examples include genetic algorithms (e.g., non-dominated sorting genetic algorithm (NSGA) I-II [44, 203]), particle swarm optimization [93], ant colony optimization [50]. In this thesis, to demonstrate the usefulness of the proposed approaches, the evolutionary algorithms, i.e., NSGA-II [44] and general evolutionary strategy [5, 13] are employed for the MOO of rotating electrical machines.

The flowchart presented in Figure 2.13 illustrates the typical process of evolutionary algorithms. The first step involves initialization, where input variables $\mathbf{x}$, the multi-objective functions $\mathbf{f}(\mathbf{x})$, and constraints $\mathbf{e}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are defined for the MOO problem (see (2.43)–(2.46)). Thereafter, an initial population is created using an appropriate sampling technique that effectively covers the design space, such as Latin hypercube sampling (LHS) [137]. The evaluation of $\mathbf{f}(\mathbf{x})$, $\mathbf{e}(\mathbf{x})$, and $\mathbf{g}(\mathbf{x})$ is then performed for each sample (individual) in the initial population. The term "population"refers to the collection of solutions in the current generation. After assessing the fitness value (quantification of performance) of each individual using methods such as crowding distance [44], tournament selection [203], the Pareto front is determined for the current generation. Subsequently, the convergence condition is checked, typically based on a certain number of completed generations or when there is stagnation in the generations. If convergence has not been achieved, a subset of individuals is selected for reproduction in the next generation based on fitness assessment or randomly. The crossover, a way of recombination, is then executed. Chosen individuals exchange genetic attributes (chromosomes) with each other to produce new individuals (offspring) to enhance the quality of genetic attributes and preserve genetic diversity. In this context, each chromosome represents a possible solution, reflecting specific characteristics or parameters of the solution. The attributes of these new individuals go through mutation, resulting in random changes to chromosomes and the introduction of new genetic traits to the population. While the crossover merges traits from two parents to generate enhanced offspring, mutation introduces variability, improving exploration and reducing the likelihood of the optimization converging to suboptimal solutions. The selection, crossover, and mutation steps are repeated until convergence. Finally, the non-dominated set of individuals (Pareto optimal solutions) from the final generation is returned as the Pareto front, and from this set, one individual is chosen as the optimal solution.

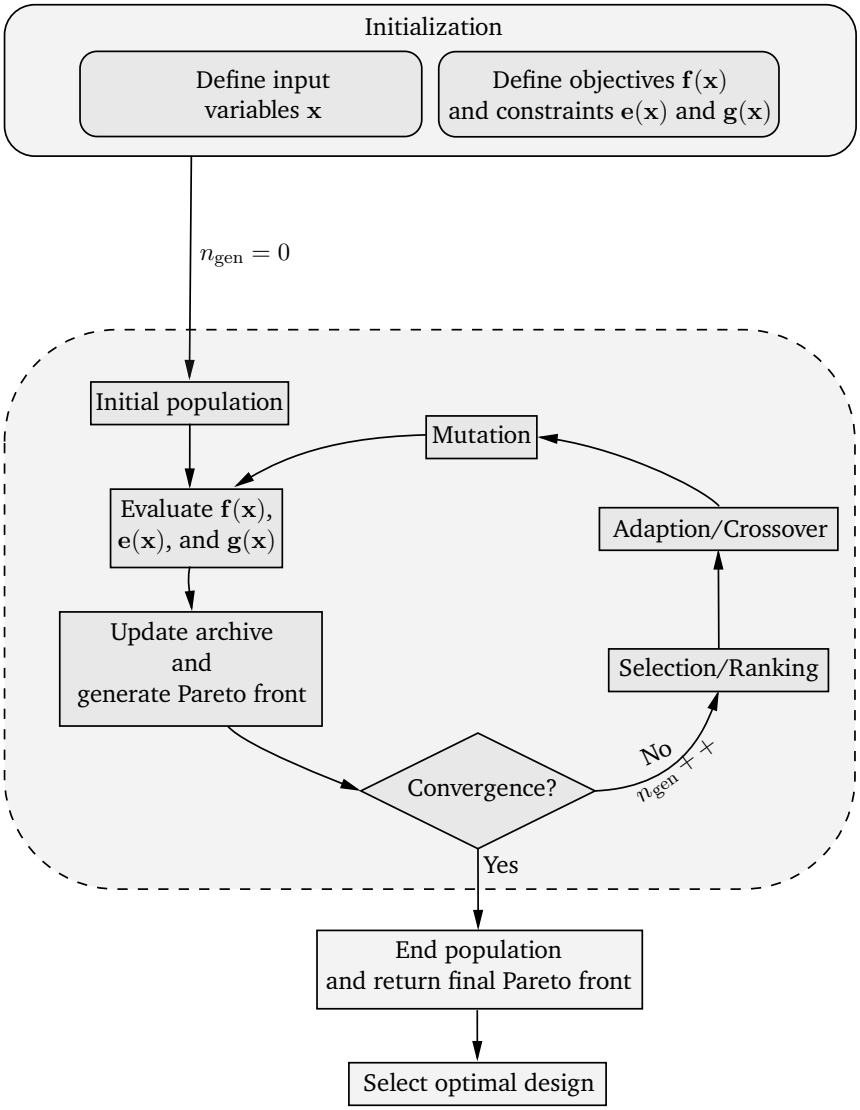


Figure 2.13: General flowchart of gradient free population based evolutionary algorithms

2.4 Summary

In this chapter, first, the general introduction to rotating electrical machines is presented. Then, the industrial design process for electrical machines is explained at a high level. Then, an electromagnetic analysis overview is described briefly, along with Maxwell's equations and the FEM process. Afterward, the simulation process for calculating KPIs for PMSM is illustrated. In the end, basics of optimization is discussed and the brief overview of gradient- and population-based optimization algorithms is given. In the next chapter, the fundamentals of DL will be discussed, a few deep learning architectures that will be employed in subsequent chapters will be explained, and applications of DL to rotating electrical machines will be reviewed.

3 Fundamentals of deep learning and literature review

In the area of meta-modeling, different approaches can be used to approximate costly objective functions during design simulations. For example, Kriging, Polynomial regression, and Radial Basis Function (RBF) are a few approaches discussed in [88]. These meta-modeling techniques are primarily examined in systems with a restricted set of design criteria [95, 160]. The classical ML algorithms, such as support vector machine [129] and random forests [29], can also be employed for meta-modeling. Generally, all these conventional approaches perform comparably to DL in scenarios where the number of training data samples is small. Nevertheless, DL becomes a more reasonable choice in the case of big data and higher-dimensional, complex design spaces, since it demonstrates superior performance compared to classical machine learning algorithms [49, 185, 223, 232]. In the late 2000s, there has been a significant surge in the adoption of DL [11, 78]. This can be attributed to multiple factors, such as the growth of the semiconductor industry leading to high computing power and the availability of information technology infrastructure for generating, storing, and processing large datasets. Further, the introduction of DL frameworks that support automatic differentiation (e.g. PyTorch [158] and TensorFlow [1]) has also played a pivotal role in fostering the use of DL. The ability to learn from raw data makes DL more flexible and potentially more powerful than conventional ML algorithms for tasks such as natural language processing, image recognition, speech recognition, and other applications involving large and complex data sets [10, 43, 120, 166]. Throughout the treatise, large datasets are utilized that incorporate very high-dimensional design spaces. Hence, this work is focused on applying various data-driven DL approaches for meta-modeling at different electrical machine design simulation steps.

In this chapter, a brief introduction will first be provided, and different types of learning approaches will be explained. Then, a few DL architectures, i.e. DNN, CNN, and generative models, will be discussed. Finally, the applications of DL in the domain of rotating electrical machines will be reviewed.

3.1 Short introduction

In 1950, Alan Turing published a landmark paper [216] in which he assumed the possibility of developing machines with real intelligence through the famous Turing test which was the first proposal in the evolution of Artificial Intelligence (AI). At the Dartmouth Conference in 1956, American Computer scientist John McCarthy introduced the term *Artificial Intelligence* for the first time. AI is a broad field within computer science that focuses on developing systems, models or computer program that can perform tasks typically associated with human intelligence. These tasks may include logic, perception, problem-solving, natural language processing, and decision-making. ML is a subset of AI that builds models without straightforward programming and performs tasks by learning patterns or features within data [70]. The process of feeding features can be handcrafted or automatic. Manual features are typically time consuming, hard and may be unscalable in practice. DL is a particular ML type that utilizes ANNs with multiple layers to automatically

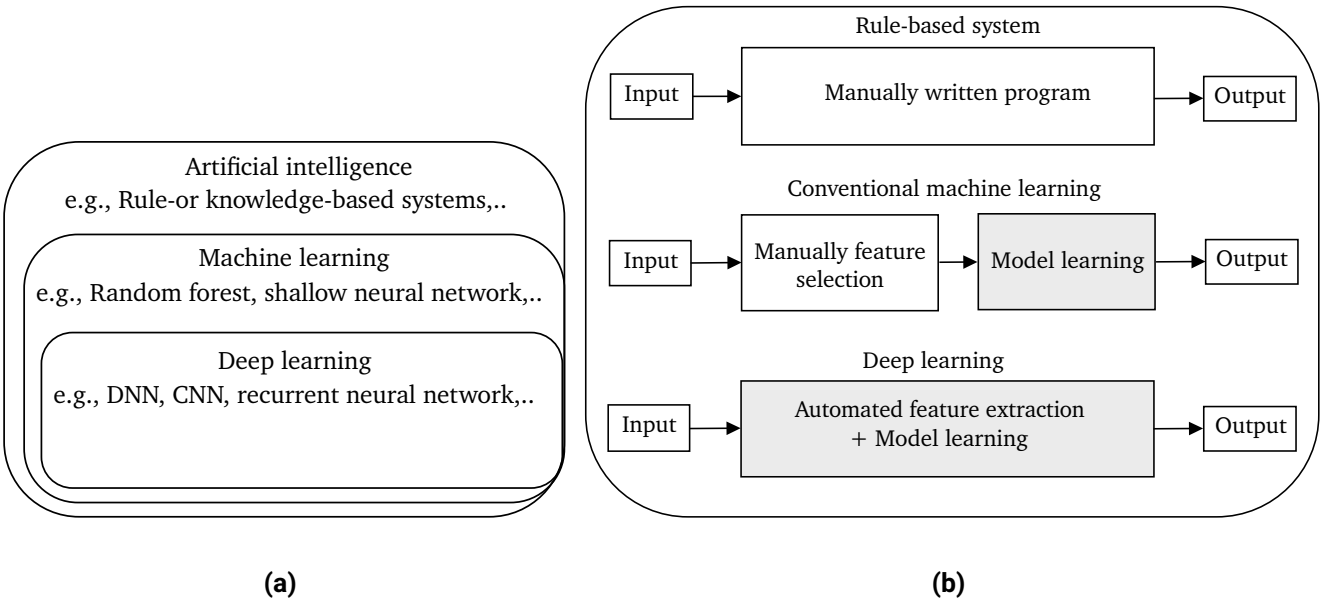


Figure 3.1: Inspired by [70] page 9-10, a) Venn diagram for different AI concepts b) Flow charts for different approaches of AI related to model building to accomplish target task.

learn complex features and underlying patterns from unstructured or structured data and solve complex engineering tasks. DL algorithms decide how to process given input data without explicitly being fed feature descriptions, unlike other conventional ML algorithms. Figure 3.1a displays a Venn diagram illustrating the relationships among AI, ML, and DL, along with examples of popular algorithms. Figure 3.1b visualizes the basic process of learning a specific task using various AI approaches. The rule-based system exemplifies the classical AI methodology. This approach executes the given task using manually programmed rules and logic tailored for specific inputs. In conventional ML, essential features are manually extracted from the inputs using the domain knowledge before being fed into an algorithm that automatically performs the target task. In contrast, the DL-based approach autonomously identifies and learns necessary features from inputs to execute the task.

3.1.1 Different types of learnings

DL algorithms can roughly be classified into three learning strategies based on their training nature: supervised learning, unsupervised learning, and reinforcement learning [16, Chapter 1],[70, Chapter 5].

Supervised learning: Supervised learning involves a labelled dataset, where the input and corresponding output values are known. This means the model learning process is guided by the true output values corresponding to each input. The network fine-tunes its parameters throughout the training to minimize the defined loss function concerning the predicted and actual values. After the training, the model can make predictions on new samples. The target can be a probability of category (e.g., classification task) or a continuous value (e.g., regression task). Figure 3.2a depicts the general workflow for training a supervised learning-based ML model.

Unsupervised learning: In the unsupervised learning, DL algorithms learn patterns from data without labelled output. In other words, the algorithm discovers structure or patterns in the data without being given explicit target values. The goal of unsupervised learning is often to identify clusters of data points

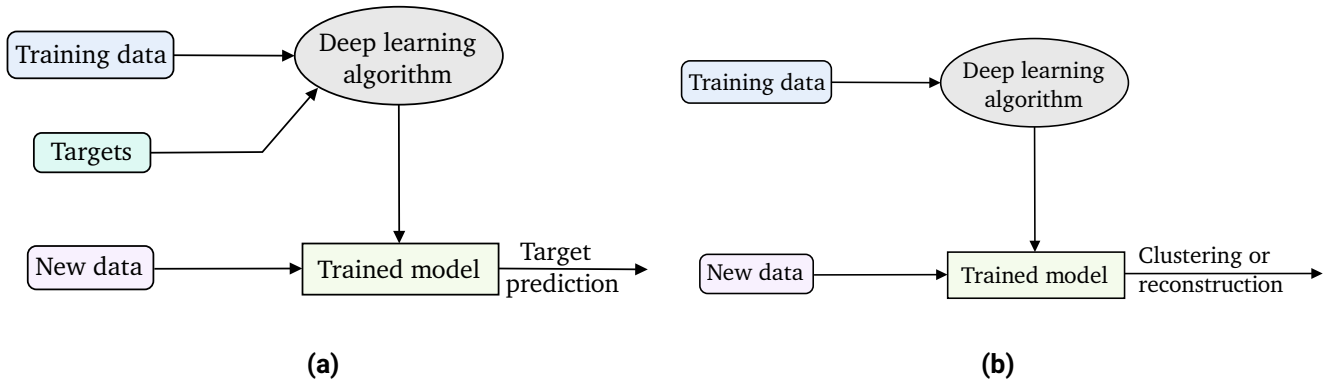


Figure 3.2: Basic flowcharts of a) supervised and b) unsupervised learning strategies, from the explanation in [70, Chapter 5].

that are similar to one another, or to reduce the dimensionality of a input by finding a smaller set of representative features. Figure 3.2b shows the usual workflow for training an unsupervised learning-based ML model.

Reinforcement learning: Reinforcement learning (RL) is a way of learning to make a good sequence decisions by interacting with a system or environment to maximize a reward or minimize the penalty. The system is usually defined as a Markov Decision Process (MDP) consisting of a set of actions, states, and a transition function, that determines the transition probability from one state to another when a specific action is taken. Further, a reward function gives an immediate reward for each correct action taken by an agent. The goal of training an agent (e.g., DNN) is to learn a policy that maps state to action for optimizing a total cumulative reward received. Various methods, such as value-based or policy-based approaches, are used by RL algorithms to learn optimal policy by exploring and exploiting the environment (refer to [206] for more details). RL algorithms do not require a labelled dataset but a true reward function which helps an agent to learn the target policy. Figure 3.3 illustrates the basic workflow for training the ML-model using RL.

Supervised learning is primarily used for classification, regression, and pattern recognition tasks. Examples include email spam detection [114], image classification [112], sentiment analysis [164], and performance analysis of electrical machines [152]. Unsupervised learning is commonly used in dimensionality reduction, clustering, and anomaly detection tasks. Applications include networking [217], image segmentation [101], and generative modeling for electrical machines [154]. RL can be used for problems such as decision-making (e.g., autonomous driving [106]), control policy learning (e.g., robotics [73]), and game playing

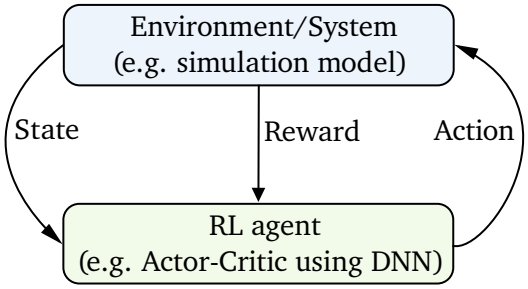


Figure 3.3: Basic flowchart for RL strategy from the explanation in [206].

(e.g., chess [198]). These learning categories may not be mutually exclusive, and there can be scenarios where multiple types of learning techniques are combined to tackle complex problems. For example, a hybrid supervised and unsupervised learning approach is used for solar flare prediction [12]. Another popular example is the training of the AlphaGo program, which utilized both supervised and reinforcement learning to master the game of Go, as proposed in [199]. In this thesis, different data-driven DL algorithms are investigated, focusing on a supervised learning approach to deal with non-linear multiple-output regression problems for electrical machines. Part of this work presents the utilization of an unsupervised learning approach in conjunction with supervised learning to quantify performance measures of rotating electrical machines for the transformed lower dimensional input space. In the next section, different DL network structures that will be employed for the proposed approaches in the subsequent chapters will be explained.

3.2 Different deep learning architectures

3.2.1 Deep neural network

In 1943, Warren McCulloch and Walter Pitts made the initial attempt to represent neural activity in a model form. Their proposed model consisted of fundamental binary components with predetermined thresholds that generated logic functions characterized by binary "zero or one" neural activity [133, 136]. The groundbreaking work on the perceptron by psychologist Frank Rosenblatt in 1958 [173], laid the foundation for today's advancements and ongoing research and development in ANNs. The ANN, in general, is inspired to some degree by human neurological activity by trying to mimic its functionality in mathematical form. In recent years, the DNN, a special type of ANN, gained popularity due to its universal function approximating capability, meaning it can model any continuous mathematical function and capture complex patterns in data with a desired level of accuracy [70]. Unlike other classical ML algorithms that focus on task-specific rules, DNNs exploit data to learn underlying functional relationships. From a network building point of view, DNNs comprise a large number of interconnected individual processing artificial neurons or perceptrons. The artificial neuron is a structural building block of any type of ANN. The perceptron takes a set of inputs and biases. Each of these inputs is multiplied with the corresponding weight, and they are summed up along with the bias. The purpose of bias in a neural network (NN) is to shift the neuron's non-linear activation function, making it more flexible in fitting the training data. The scalar result is fed into a non-linear activation function, and the output of this activation function is a prediction of the perceptron. Mathematically, the working of a perceptron can be described as follows,

$$\hat{y} = f_{\text{act}}(\mathbf{w}^\top \mathbf{x} + b_0) \tag{3.1}$$

where $\mathbf{x}$ is an input column vector ($\mathbf{x} = [x_1, x_2, \ldots, x_i, \ldots, x_m]$), $\mathbf{w}$ is a weight column vector and described as $\mathbf{w} = [w_1, w_2, \ldots, w_i, \ldots, w_m]$ containing the weights concerning each input in $\mathbf{x}$, f_{act} is the non-linear activation function, b_0 is a bias term and, lastly, $\hat{y}$ is the scalar output associated to the perceptron.

DNNs typically comprise an input layer, multiple hidden layers, and an output layer. The input layer usually accepts the input data as a vector and transmits the weighted input data to the next hidden layer. Typically, the inputs and outputs of the hidden layer are not directly observable, which is why it is called the hidden layer. Each hidden layer consists of several neurons that are stacked in parallel. The output of each hidden

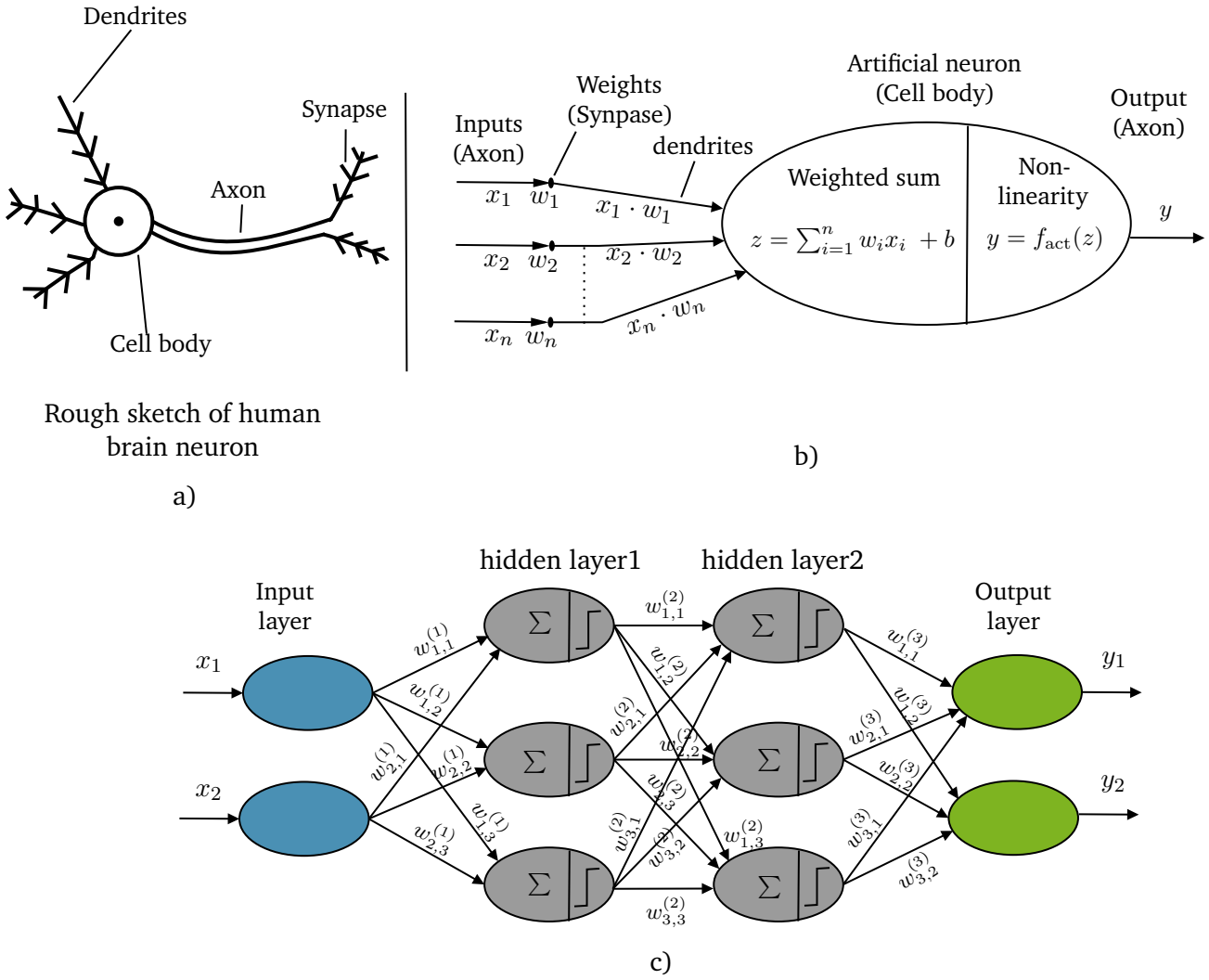


Figure 3.4: a) Biological model of a neuron b) Mathematical model of an artificial neuron c) DNN.

layer serves as the input for the next hidden layer, and multiple hidden layers are linked together in a cascade fashion. The final hidden layer is connected to the output layer, which has a number of neurons equal to the number of classes or continuous values for prediction. Depending on the network structure, DNNs can be densely connected or sparsely connected. In a densely connected DNN, all neurons in each layer are connected to all neurons in the next layer. This is also known as a fully connected DNN. However, in a sparsely connected DNN, some connections between neurons are pruned or eliminated to reduce the number of parameters and lower the computational effort. Figure 3.4a illustrates a rough sketch of a biological neuron, while Figure 3.4b portrays the comparable mathematical representation within a single artificial neuron [52, 125]. Figure 3.4c depicts a representative DNN architecture consisting of one input layer, two hidden layers, and an output layer.

As previously mentioned, each neuron in DNN calculates a weighted sum of its input and bias, which is then passed through a non-linear function to produce the scalar output. The non-linear activation function aims to introduce non-linearity in the network, which allows DNN to approximate arbitrarily complex

functions. It is essential to add non-linearity as most real-world datasets include non-linear patterns. A range of activation functions $f_{\text{act}}(\cdot)$ can be employed to model and reproduce various behaviors. A few commonly used activation functions are illustrated in Figure 3.5. The following are explanations for some of these functions:

Sigmoid: A continuous and monotonically increasing function maps any input value to a value between 0 and 1. Because of this, it is often used in DNNs for classification tasks where the function's output represents a particular class's probability. Additionally, the sigmoid function is differentiable and its derivative can be easily computed. However, it has a drawback of vanishing gradients for both extremely small and large input values, which can pose a challenge during the training process. It is defined as:

$$f_{\text{act}}(z) = \frac{1}{1 + \exp^{-z}} \in [0, 1], \tag{3.2}$$

where from (3.1) $z = \mathbf{w}^\top \cdot \mathbf{x} + b_0$ is a scalar value.

Tanh: The hyperbolic tangent activation function is zero-centred and also a monotonically increasing and differentiable function. It is similar to the sigmoid function in shape. However, it maps inputs to outputs between -1 and 1. It also suffers from a vanishing gradient problem. It is written as

$$f_{\text{act}}(z) = \frac{\exp^{z} - \exp^{-z}}{\exp^{z} + \exp^{-z}} \in [-1, 1]. \tag{3.3}$$

Rectified Linear Unit (ReLU): A piece-wise linear function that yields an unchanged input when it is positive and produces zero otherwise. It was first time introduced in [142] for a restricted Boltzmann machine. It is described as:

$$f_{\text{act}}(z) = \max(0, z) \in [0, \infty). \tag{3.4}$$

It has several advantages when utilized in DNNs, such as being computationally efficient to evaluate, allowing a simple gradient calculation during backpropagation (0 for $z \leq 0$, and 1 when $z > 0$, the

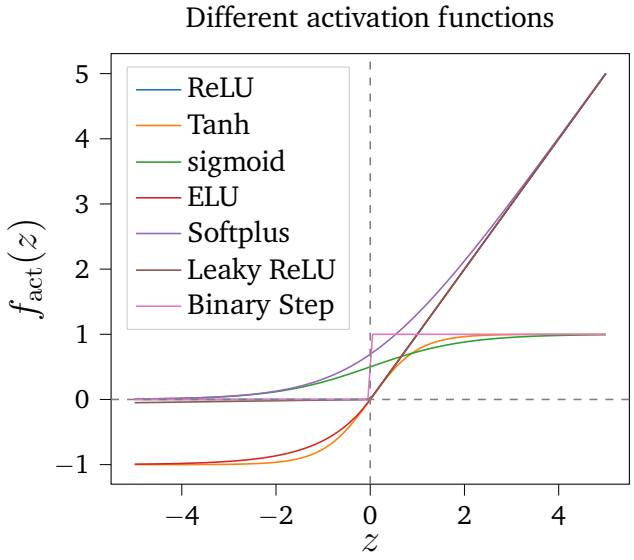


Figure 3.5: Illustration of different activation functions.

discontinuity at $z = 0$ is ignored), and its ability to mitigate the vanishing gradient problem [68, 142]. As a result, ReLU is widely used as an activation function, particularly in computer vision tasks. However, ReLU can suffer from the "dying ReLU"problem, in which gradients during backpropagation may adjust the weights such that the neuron consistently outputs negative values for all inputs. This makes these neurons inactive in the learning process, which can potentially impact the network's performance [208]. Variants of ReLU, such as leaky-ReLU and PReLU, have been proposed in the literature to overcome this problem [76].

Exponential Linear Unit (ELU): The ELU was proposed in [40]. It has a smooth curve that goes to negative infinity instead of abruptly stopping at zero. It allows negative inputs to produce non-zero outputs, which helps to keep the gradient flowing during training and can lead to easier and faster convergence. Additionally, it has been shown to improve the performance of DNNs in some applications compared to ReLU and its variants [40]. ELU is defined as follows:

$$f_{\text{act}}(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha(\exp(z) - 1) & \text{if } z \leq 0, \end{cases} \tag{3.5}$$

where α is a positive constant determining the function's slope when $z < 0$. The advantage of this function is that it has negative values for $z < 0$, which helps to mitigate the vanishing gradient problem.

Softplus: As described in [233], it takes any real input value and returns a positive output value. It also provides an alternative to solve the dead ReLU problem [208]. The derivative of the softplus is a sigmoid. It is defined as

$$f_{\text{act}}(z) = \ln\left(1 + \exp^{z}\right) \in [0, \infty). \tag{3.6}$$

These are a few examples of the many activation functions that can be set while defining a network. The selection of the activation function should be based on the type of the problem and the characteristics of the available data, typically determined through a process of trial and error. Once familiarized with the general network structure, which encompasses the input layer, hidden layer, output layer, and activation functions, the DNN training process will be explained in the following subsection.

3.2.1.1 Training

The training of neural network can be formulated as an optimization problem. The loss function of the network quantifies the cost that occurs due to incorrect or erroneous predictions. The trainable network parameters (weights and biases) are adjusted to minimize the loss function with the ground truth and prediction for the given input-output data points (training samples). It can be written as an optimization problem for minimizing the loss function as below

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathbf{F}(\mathbf{x}^{(i)}; \Theta), \mathbf{y}^{(i)}) \tag{3.7}$$

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \mathcal{J}(\Theta), \tag{3.8}$$

$$\text{where, } \mathcal{J}(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathbf{F}(\mathbf{x}^{(i)}; \Theta), \mathbf{y}^{(i)}),$$

where Θ denotes the trainable parameters of the network, and $\mathcal{L}$ represents the loss function. For each input vector $\mathbf{x}^{(i)}$, where $i \in 1, \ldots, n$ training samples in the dataset and $\mathbf{x}^{(i)} \in \mathbb{R}^d$, the network's prediction

Application	Loss function
Classification	Binary cross entropy, Hinge loss,
Regression	Mean squared error (MSE), Mean absolute error (MAE)

Table 3.1: Examples of commonly used loss functions.

is given by $\mathbf{F}(\mathbf{x}^{(i)};\Theta)$. The corresponding target vector for each $\mathbf{x}^{(i)}$ is represented as $\mathbf{y}^{(i)}$, with $\mathbf{y}^{(i)} \in \mathbb{R}^m$. The choice of the loss function depends on the application. A few typically practiced loss functions are given in Table 3.1. For example, if there are m target classes for the classification, and if $\mathbf{y}^{(i)}$ is a one-hot encoded target vector for instance i, and $\hat{\mathbf{y}}^{(i)}$ representing the model's predicted probabilities, then the mean categorical cross-entropy loss function for n samples is given by:

$$\mathcal{J}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{m} y_c^{(i)} \log(\hat{y}_c^{(i)}). \tag{3.9}$$

Here, the c-th component of vector $\mathbf{y}^{(i)}$, denoted as $y_c^{(i)}$, is 1 for the correct class and 0 for all other classes, while $\hat{y}_c^{(i)}$ is the predicted class probability. Similarly, the loss function for the regression task can be described as a MSE by

$$\mathcal{J}(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)})^2, \tag{3.10}$$

where $\mathbf{y}^{(i)}$ is the actual target vector, and $\hat{\mathbf{y}}^{(i)}$ is the predicted target vector for the i-th input sample.

Once a loss function has been determined, the next step in training DNNs involves adjusting the weights of the network to minimize the difference between the predicted output and the target output for each sample in the dataset. This is achieved by resorting to the backpropagation algorithm [177] which is an efficient automatic differentiation technique to compute loss derivatives with respect to the network parameters. As illustrated in Figure 3.6 inspired from [2], a simple neural network with one input (a), one hidden neuron, and one output neuron, the backpropagation can be explained using a chain rule as below

$$\delta_1 = \frac{\partial \mathcal{L}(\theta)}{\partial w_2} = \frac{\partial \mathcal{L}(\theta)}{\partial c}\frac{\partial c}{\partial w_2} \tag{3.11}$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial w_1} = \frac{\partial \mathcal{L}(\theta)}{\partial c}\frac{\partial c}{\partial w_2}\frac{\partial w_2}{\partial b}\frac{\partial b}{\partial w_1} = \delta_1 \frac{\partial w_2}{\partial b}\frac{\partial b}{\partial w_1}, \tag{3.12}$$

where $\mathcal{L}(\theta)$ represents the loss function of the neural network, θ denotes trainable parameters (w_1 and w_2 in this example) in the network. w_1 and w_2 represent the neural network weights for the first and second layers, respectively. The value b represents the output of the hidden neuron, while c denotes the output of the neural network. The partial derivative of the loss function with respect to the weight w_2 is

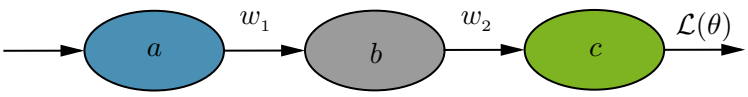


Figure 3.6: Illustration of one input, one hidden neuron, and one output neuron NN inspired from [2],

represented by δ_1. The quantity of interest to be calculated using backpropagation is the partial derivative of the loss function with respect to the weight w_1, and it is denoted by $\frac{\partial \mathcal{L}(\theta)}{\partial w_1}$. The gradient of the loss function $\mathcal{L}(\theta)$ with respect to the weight w_2 is computed using the chain rule. Starting from the loss function $\mathcal{L}(\theta)$, it can be decomposed backwards through the output to the weight w_2; see (3.11). The chain rule is again applied to calculate the gradient with respect to w_1, and the previously computed gradient for w_2 is utilized; see (3.12). This entire process is termed backpropagation because gradients are propagated back from the output loss function $\mathcal{L}(\theta)$ to the weight (w_1). Once the gradients are available, different types of gradient based optimization algorithms can be used for weights updation (refer to Sec. 2.3.2.1).

Algorithm 1 General training algorithm of DNN

1: Initialize trainable network parameters: Θ ▶ random initialization
2: **repeat**
3: Select batch of M data points
4: Make forward pass and compute loss $J(\Theta)$
5: Compute gradient using backpropagation: $\frac{\partial J(\Theta)}{\partial \Theta} := \sum_{k=1}^{M} \frac{\partial J_k(\Theta)}{\partial \Theta}$
6: Update parameters Θ with the suitable optimizer, e.g., Adam, SGD, AdaGrad
7: **until** convergence of Θ
8: Return Θ: Model training completed

Algorithm 1 explains the general training algorithm for an ANN. The algorithm iteratively includes two main steps: forward propagation and backpropagation. In the forward propagation step, the ANN makes a prediction based on the input data and computes the loss. In the backpropagation step, it computes the gradient concerning each weight and bias in the network. These gradients are then used to update the network's weights and biases using suitable gradient-based optimization algorithms, such as standard gradient descent, SGD, and more advanced variants like Adam, the adaptive gradient algorithm (AdaGrad), and root mean square propagation (RMSProp). For further information, refer to Sec. 2.3.2.1 and [175]. Before initiating the training process of a DNN, it is necessary to consider other factors to ensure convergence and prevent undesirable effects such as overfitting or underfitting. This entails the optimization of various network hyperparameters, which will be detailed in chapter 4.

The fully connected DNNs are more useful in predictive modeling (regression tasks) when dealing with scalar parameter-based input data, where the parameters are usually independent. However, they are not as useful for handling computer vision tasks that involve spatial hierarchies or require understanding geometric relationships, such as object recognition and image recognition. To address this, a specialized ANN called a CNN is explicitly designed for handling visual data [38]. The following subsection will explain it in detail.

3.2.2 Convolutional Neural Network

Vision allows humans to perform object identification, spatial navigation, and situational awareness. Computers interpret images as a matrix of numbers, also called *pixels*. A feature in an image refers to a distinguishing structure or pattern that carries meaningful information. Extracting features manually from images can be a time-consuming task, especially when numerous visual changes occur within the image. For example, when dealing with a picture of an electric machine, the high-level features may include components such as the stator, rotor, magnets, stator yoke, winding construction, etc. An algorithm should

be able to analyze the image, starting from the pixel level and progressing to high-level features with semantic meaning, without losing spatial information and while being sensitive to inter-class variation. In this context, inter-class variation describes how one class or category, e.g., different types of electrical machines or their components, is distinct or different from another. Spatial information in an image does refer to the arrangement of pixels and their relationships, capturing the patterns and structures within the image. Therefore, it is essential to exploit the spatial information of the image efficiently with an affordable computational cost. The use of a fully connected DNN can be computationally very expensive for high-resolution images, as it loses spatial information by treating each pixel independently and ignoring their relative positions and connections. The CNN can preserve spatial information and learn complex features (e.g., edges, sharpness, and shape) via a convolution operation with many filters. The CNN was first introduced in the 1980s by Kunihiko Fukushima [56]. It was named "Neocognitron" and was designed to recognize visual patterns inspired by the hierarchical organization of the human visual cortex. Then in 1989, Lecun et al. [119] presented the initial practical illustration of CNN with backpropagation to recognize handwritten digits. It laid a foundation for developing modern CNNs for different computer vision tasks, e.g., image classification, object detection, autonomous driving, and medical imaging. In 2012, a deep CNN architecture called AlexNet won the ImageNet challenge with a significant margin in error rates compared to traditional methods [112]. AlexNet used several convolutional layers that could automatically learn hierarchical representations of image features, making it much more accurate at recognizing images. This breakthrough showed that CNNs can be more effective than traditional methods in image recognition tasks. Consequently, the application of CNNs became popular among researchers and practitioners for solving vision-related tasks. Figure 3.7a illustrates an example of the typical structure of a CNN, which includes an input layer, two convolutional layers with non-linear activation functions, two pooling layers, a flatten layer, a dense layer, and an output layer. The different commonly used layers in building a CNN in practice are explained as follows:

Convolution layer: The convolutional layer is a basic layer of a CNN, where a filter or weight matrix is convolved over a feature map to generate a new feature map. This convolution operation is a powerful technique used to combine two signals and create a processed signal. It has broad applications in domains such as signal processing and speech processing. During convolution, a filter or kernel, which is a small window of weights, slides over the feature map, and a dot product is calculated between the filter and the overlapping input values. The resulting output of the convolutional layer is then passed through a non-linear function (e.g., ReLU, ELU), which generates feature maps that capture various characteristics of the input data, such as edges, textures, or shapes. Introducing non-linearity after a convolutional layer is essential since image data is typically highly non-linear by nature. The key distinction between convolutional layers and standard fully connected layers is that the former only applies convolution operations to a subset of the input feature map. Specifically, each convolutional neuron only processes data within its receptive field. The primary advantage is the reduction in the number of trainable parameters while exploiting spatial relations in the input feature map. As mentioned in [70], the convolution operation in some neural network libraries is implemented as its related function cross-correlation and described as a convolution that can be written

$$H(i,j) = (W * G)(i,j) = \sum_{a} \sum_{b} G(i+a, j+b) W(a,b), \tag{3.13}$$

where H is the output feature map of the convolution operation, W is the filter, and G is the input feature map. The filter W is a small matrix of learnable weights, and it is convolved over input feature map G and computes a dot product between the filter and the overlapping input values at each location (a, b). Equation 3.13 explains a 2D convolution operation commonly used in image processing tasks. However, there are variations of this equation for different types of convolutions, such as one-dimensional (1D) or

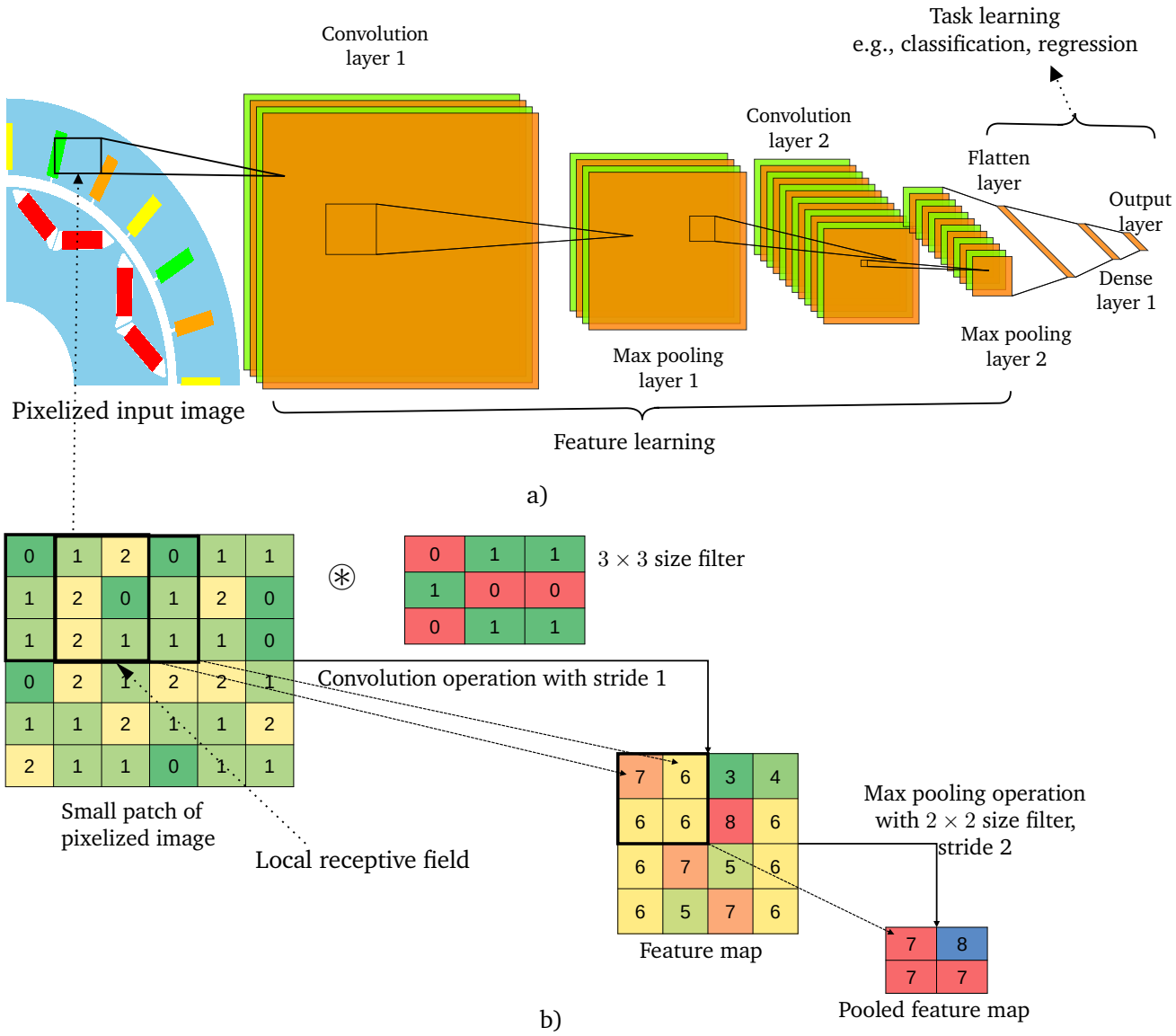


Figure 3.7: Illustration of CNN.

three-dimensional convolutions, and for different padding and stride configurations (refer to [70, 125] for more details). Figure 3.7b represents examples of a convolution operation for 6×6 input image patch and a filter of size 3×3, with stride one that outputs 4×4 feature-map. As described in [125], the output feature map size can be calculated with the below formula

$$b_H = \frac{b_G - n_W + 2p}{s} + 1 \quad \text{and} \quad h_H = \frac{h_G - n_W + 2p}{s} + 1, \tag{3.14}$$

where b_H and h_H represent the width and height of the output feature map, respectively. p is zero padding, s is stride, and n_W is the filter size. b_G and h_G are the width and height of the input feature map.

Pooling layer: It reduces the spatial dimensions of the feature map, which is the previous layer's output. Two of the most widely used pooling operations are average and max pooling. Figure 3.7b displays max

pooling operation for feature map of size 4×4 with a stride two and outputs feature with smaller size to 2×2. These layers are often employed to preserve various rotational and spatial invariances, which helps to mitigate the risk of over-fitting [125].

Flatten layer: It reshapes the output from the convolutional or pooling layers (multi-dimensional arrays) into a 1D vector, which can then be fed into a fully connected layer for further processing, i.e., classification or regression. It is necessary because the fully connected layer only accepts inputs as a 1D vector.

Fully connected layer: A layer in which all the neurons receive input from every neuron in the previous layer and produce output that is fed to every neuron in the next layer. It functionally maps the high-level features extracted by the convolutional and pooling layers to the ground truth, allowing CNNs to perform complex tasks like regression, object detection, and image classification.

The process of training a CNN is typically carried out in a similar way to that of a DNN, by using the backpropagation algorithm as explained in subsubsection 3.2.1.1.

CNNs have proven to be highly effective in tasks involving image and pattern recognition, due to their capacity to preserve the spatial hierarchy of data and their efficiency in handling high-dimensional inputs. However, they do not directly allow to generate new data by learning underlying distributions. This limitation leads to the exploration of a different type of DL model, known as generative models, which can effectively generate new data based on lower-dimensional learned representations. These models also enable to perform downstream tasks by making use of these learned representations in subsequent processes. In the following subsection, the generative model to be utilized later in Chapter 6 is described.

3.2.3 Generative network

The objective of generative modeling is to train a model that can capture the underlying probability distribution of the given data points, enabling the generation of new samples from the same distribution. Essentially, the goal is to learn a model that represents the distribution from which the original data was drawn, allowing for sample generation.

3.2.3.1 Autoencoder

In 1986, Rumelhart et al. [176] introduced the autoencoder as an unsupervised learning approach to learn patterns from the input data without any explicit need for ground truth data. An autoencoder is a specific type of ANN that comprises two interconnected networks: an encoder and a decoder. The encoder network transforms the input data into a lower-dimensional representation while the decoder reconstructs the original input from its compressed representation. During training, the autoencoder aims to minimize the difference between the input and the output, enabling the encoder to learn a compressed and meaningful representation of the input, where "meaningful"refers to an encoding that retains and emphasizes essential features relevant for reconstructing the input. When the encoder and decoder networks have many dense layers in the structure, it is called a deep autoencoder. Figure 3.8a displays the general structure of a deep autoencoder. Suppose training input samples $\mathcal{D} = \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(i)}, \ldots, \mathbf{x}^{(N)}$ are given, where each sample vector $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is without a label. The dataset is represented by a $d-$dimensional random variable $\mathbf{x}$. The first part of the network encodes the input $\mathbf{x}$ into a hidden representation $\mathbf{z}$, typically $\mathbf{z} \in \mathbb{R}^n$ with $n \ll d$. Using this hidden random variable $\mathbf{z}$, the second part of the network reconstructs the $d-$dimensional vector $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}} \approx \mathbf{x}$.

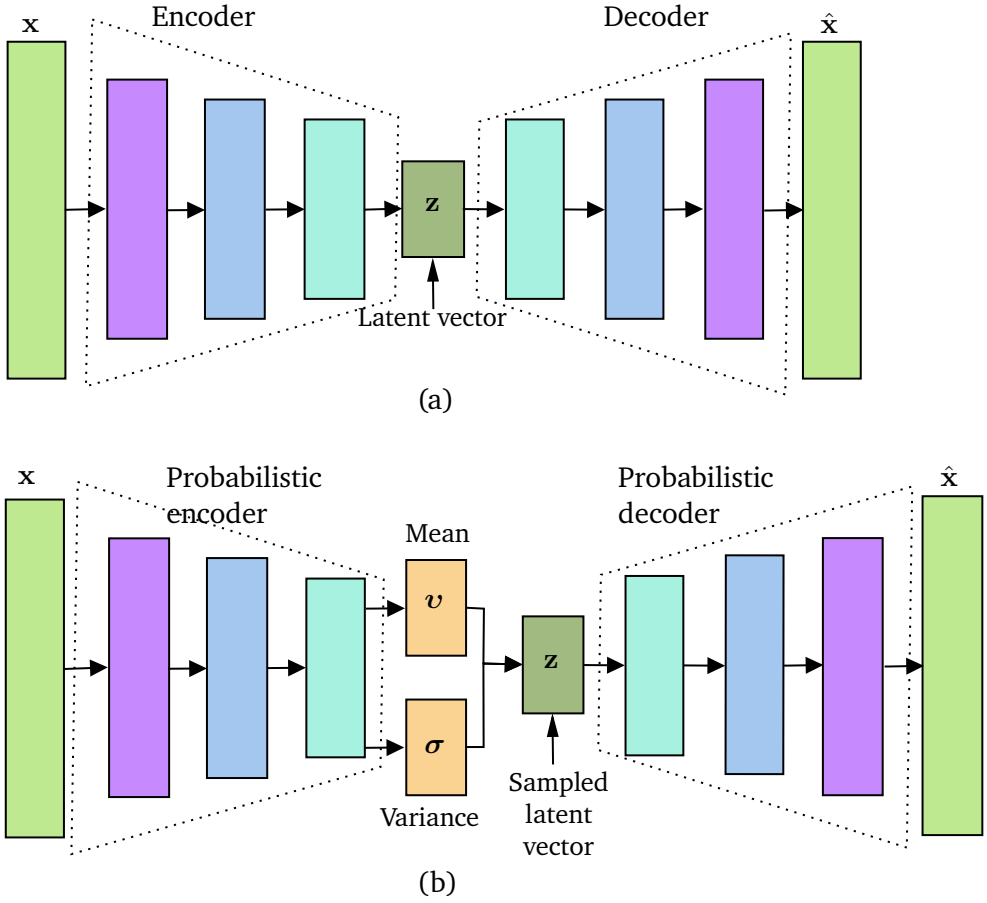


Figure 3.8: Illustration of a) Autoencoder and b) VAE.

The general mathematical framework of the autoencoder can be described below:

$$\mathbf{z} := \mathbf{F}_\theta(\mathbf{x}) \tag{3.15}$$

$$\hat{\mathbf{x}} := \mathbf{F}_\phi(\mathbf{z}), \tag{3.16}$$

where $\mathbf{F}_\theta$ and $\mathbf{F}_\phi$ are the encoder network and the decoder network, respectively. θ and ϕ are learnable parameters, i.e., weights and biases that adjust during the training. Generally, the reconstruction loss function $\mathcal{L}(\theta, \phi; (\mathbf{x}, \hat{\mathbf{x}}))$, parameterized by learnable parameters (θ, ϕ), with respect to the input sample and its reconstruction ($\mathbf{x}^{(i)}$, $\hat{\mathbf{x}}^{(i)}$), for training the autoencoder can be defined in scalar form as a MSE by

$$\mathcal{L}(\theta, \phi; (\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)})) := \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)} \right\|^2. \tag{3.17}$$

The training is carried out using a standard backpropagation by minimizing the loss function $\mathcal{L}(\cdot)$; refer to subsubsection 3.2.1.1.

3.2.3.2 Variational autoencoder

A conventional autoencoder has a deterministic latent layer that is not regularized, which means there are no explicit penalties or constraints applied to the representations in the latent layer during the training process.

This limits its ability to generate new data and results in a lack of generalization. In 2013, the variational autoencoder (VAE) was introduced to address the limitations of the standard autoencoder [103]. The VAE uses a probabilistic approach to describe an observation in the latent space. The VAE is different from the standard autoencoder as it specifies a probabilistic encoder that produces a probability distribution for each latent attribute instead of a single value, along with a probabilistic decoder that generates samples in the original design space. Figure 3.8b displays the general architecture of the VAE. Similar to a conventional autoencoder, the mathematical formulation for the VAE can be written as

$$(\boldsymbol{\upsilon}, \boldsymbol{\sigma}) := \mathbf{F}_\Theta(\mathbf{x}) \tag{3.18}$$

$$\mathbf{z} = \boldsymbol{\upsilon} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon} \tag{3.19}$$

$$\hat{\mathbf{x}} := \mathbf{F}_\Phi(\mathbf{z}). \tag{3.20}$$

The probabilistic encoder $\mathbf{F}_\Theta$ and decoder $\mathbf{F}_\Phi$ networks compute the conditional distributions $\mathcal{P}(\mathbf{z}|\mathbf{x})$ and $\mathcal{P}(\mathbf{x}|\mathbf{z})$, respectively. Where Θ and Φ are trainable network parameters of the VAE and $\odot$ is an element-wise dot product. Assuming that the latent distribution $\mathbf{z}$ follows a standard normal distribution, which is commonly adopted in practice, the encoder network $\mathbf{F}_\Theta$ outputs the distribution parameters $\boldsymbol{\upsilon}$ and $\boldsymbol{\sigma}$ as vectors of size n. To efficiently calculate the gradients during network training, the reparameterization trick (3.19) is used to sample the latent vector $\mathbf{z}$ by adding a noise vector $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ of size n, as described in [103, 105].

The training process aims to enhance the encoding and decoding processes by simultaneously optimizing the network parameters Θ and Φ. In contrast to the training loss for autoencoders given in (3.17), the training loss for the VAE involves an additional regularization term, namely Kullback–Leibler (KL) divergence. The training loss is described by

$$\mathcal{L}(\Theta, \Phi; (\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)})) = \frac{1}{N} \sum_{i=1}^{N} \Big(\Big\| \mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)} \Big\|^2 + \mathcal{D}_{\mathrm{KL}} \Big(\mathcal{P}(z^{(i)}|\mathbf{x}^{(i)}, \Theta) \;||\; \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \Big) \Big). \tag{3.21}$$

The KL divergence $\mathcal{D}_{\mathrm{KL}}$ works as an additional penalty term in the loss function to minimize the distance between the encoder distribution and the prior distribution of the latent variable $\mathbf{z}$. This additional regularization term helps to gain two useful properties: continuity and completeness. Essentially, this means that points that are located close to each other in the latent space will have similarities when decoded, leading to more meaningful interpretations [2].

The encoder network $\mathbf{F}_\Theta$ can be seen as an inference network that approximates the posterior distribution of the latent space $\mathbf{z}$. On the other hand, the decoder network is viewed as a generative model that can generate new samples in the input space that are similar to the samples in the training distribution.

3.3 Literature review:deep learning applications to rotating electrical machines

In Sec. 1.2, ML applications to electrical machines were briefly reviewed. In this section, DL applications to the electrical machines domain are presented in greater detail. Since the 1990s, ANNs with shallow structures, such as feed-forward networks (with one or two layers) or recurrent neural network (RNN), have been employed primarily for control and parameter identification tasks. Several of these ANN applications in the electrical machine field are illustrated in articles [27, 131, 193, 222, 225]. By the late 2010s, after the breakthrough of CNNs in the ImageNet challenge [112], DL began to be incorporated into various stages of virtual prototyping for electrical machines.

The DL-based regression model, consisting of three hidden layers, two inputs, and one output, is proposed to predict PMSM efficiency in relation to speed and torque [87]. This model is a Deep Belief Network (DBN) [78] composed of multiple layers of stochastic units. During pre-training, each pair of successive layers forms a Restricted Boltzmann Machine (RBM) [53], and the final layer is treated as a sigmoid belief network. DBN training is carried out in two phases: first, each pair of layers is trained as an RBM in a layer-wise fashion, followed by tuning the entire network with supervised learning using backpropagation. The dataset, obtained using a commercial FE solver, comprises the electromagnetic torque and speed as inputs and PMSM efficiency as the output. Numerical results indicate that the proposed DL model offers high prediction accuracy, effectively approximating the nonlinear relationship between the PMSM's efficiency, speed, and torque, thus shortening the motor design cycle. This study, focused on a low-dimensional scalar input-output space, merely demonstrates the application of DL in PMSM performance analysis while keeping geometric parameters constant.

In [66], a multi-layer perceptron (MLP) with only three layers (input-hidden-output) is employed as a meta-model to predict the characteristics of a PMSM, specifically efficiency through predicting electrical power consumption, based on scalable scalar input design parameters such as stack length, active diameter, and turns per coil. Compared to [87], this work incorporates varying scalar geometry parameters for dataset generation along with various torque and speed combinations. A separate MLP is trained to predict the maximum torque on the limit curve. Similarly to [66], the dataset is generated using FE simulations. After supervised training, the MLP was tested on a fraction of the dataset that was not used during training. Numerical results indicate reasonable prediction accuracy for most torque-speed combinations, although there are significant deviations in the zero torque region.

The exploration of DNN as a meta-model for optimizing a flux switching machine (FSM) based on scalar geometry parameters is presented in [116]. The authors propose a MOO workflow using multiple DNNs (30 in total). This proposed workflow includes several execution branches to address computational resource challenges (Fig. 3 of [116]). Each DNN is trained to predict either a single or multiple objectives using supervised learning. It is observed that DNNs trained to predict multiple objectives perform better than a single objective, likely due to the hidden correlation between outputs. The training data for training is derived from conventional FE simulations. The DNN uses a vector of selected geometric parameters as inputs and predicts FE outcomes. These results encompass KPIs like electromagnetic torque ripple, average torque per magnet mass, and cogging torque. The distribution prediction error serves as an evaluation metric for quantifying DNN predictions. Notably, all DNNs exhibited relatively weak prediction accuracy for the cogging torque. The scope of this study was confined to predicting single-domain KPIs in a low-dimensional design space. Furthermore, the approach of training multiple DNNs during each generation of the MOO might become computationally intensive as the parameter space expands.

The feasibility of predicting magnetic field solutions using a deep CNN for low-frequency electromagnetic devices, including a four-pole 24-slot interior permanent magnet synchronous motor (IPMSM), a coil,

and a transformer, is explored in [96]. The dataset for DL model training was generated using an FE solver. All input CAD models in the dataset were converted into four-channel pixelated images containing information on geometry, material properties, and excitation. The deep CNN's encoder-decoder architecture was designed using dilated filters in the CNN layers, obtained after evaluating about thirty-five configurations through a random approach. The CNN was trained using supervised learning for dense regression task, with the input being a four-channel $400 \times 400 \times 4$ pixelized cross-sectional image of all three electromagnetic devices and the output as the corresponding magnetic field distribution estimation image sized $400 \times 400 \times 1$. This implies that deep CNN predicts the magnetic field intensity for each pixel in the input image. Fig. 1 in [96] illustrates examples of the input, network structure, and predicted fields for each input pixel. However, models trained purely using supervised learning lack physical inference, which leads to their inability to assess geometries outside the training distribution. To partially mitigate this and enhance the network's generalization capabilities, dropout [204] was employed by introducing probabilistic weights during training to quantify prediction uncertainties and produce uncertainty maps (refer to Fig. 4 in [96]). Dropout randomly removes a certain percentage of neurons in each training epoch. The deep CNN's limitation is noted in handling unstructured meshes. A notable disadvantage of the proposed approach is its computational intensity due to processing high-resolution images. Additionally, the study was limited to a parameterized IPMSM geometry, varying only five geometric parameters for dataset generation while treating other challenging parameters, such as pole pairs and stator outer diameter, as constant.

One important KPI for a detailed analysis of an electrical machine is the efficiency map (refer to [135] and Sec. 2.2.3), which provides a graphical representation of a machine's performance by indicating the ratio of mechanical power (shaft power) to electrical power across various speed-torque operating points. However, deriving an efficiency map is both time-consuming and computationally demanding. In response, [94] introduces two DL based strategies to swiftly predict the efficiency map. The first strategy employs various DL models, namely feed-forward DNN, CNN, and RNN, at specific stages of the conventional workflow, mainly where time-intensive FE simulations are typically conducted (refer to Fig. 1, 3, and 4 in [94]). In its initial phase, a combination of CNN and feed-forward DNN is utilized to predict flux linkage using both scalar parameters- and pixelated image-based inputs. The results from this phase assist in determining optimal excitation conditions (phase current, control angle) for the subsequent step, which are then fed as input along with geometry information to calculate efficiency values for different excitation conditions. A significant merit of this modular method is its ability to manage variable input sequences. Conversely, the second approach utilizes a deep CNN-based encoder-decoder framework, similar to the structure in [96] and the training process. After training, the pixelated CAD input, which encapsulates geometry, material, and excitation data, is fed into the network to project a 2D efficiency map directly. Like efficiency maps, power factor maps can also be predicted [98]. Both approaches require significant computational demand for training different DL models and processing high-resolution images. In [100], transfer learning is proposed to demonstrate generalizability and address the computational needs for the training. This method leverages a pre-trained network from one task as a starting base network for the another, followed by subsequent fine-tuning of it via, for example, supervised training. All the proposed approaches are purely based on data-driven training and lack physical interpretation, i.e., their predictions are unreliable beyond the training data distribution.

So far, DL applications for parametric MOO of various rotating electrical machines using a supervised learning strategy have been discussed. It involves varying specific machine design parameters to enhance performance characteristics, such as the active component's diameter, the number of pole pairs, coil winding configurations, magnet dimension, and other variables within the bounded range. The parametric optimization aims to find the optimal combination of parameter values to achieve desired outcomes, such

as improved efficiency, torque output, or other performance metrics, without altering the components' fundamental structure and layout. The DL algorithms can also be utilized for the topology optimization of electrical machines. For simplicity and distinction, topology optimization is termed as free-form structural optimization (FFSO) in this section. In the present context, FFSO extends beyond mere parameter adjustments to fundamentally redesign the core structure of electrical machines. Unlike parametric optimization, which may be constrained by specific geometric forms, FFSO allows for unconventional shapes. FFSO focuses on the optimal material distribution and strategic arrangement of key components, such as magnets, coils, and laminations, within a predefined design region. By experimenting with various layouts, the objective is to enhance KPIs such as electromagnetic efficiency, mechanical resilience, or thermal performance [146]. At its core, the aim is to achieve optimal machine performance through effective material distribution. In Table 1 of [146], authors present a high-level comparison between parametric and FFSO. It is noted that the industry predominantly adopts parametric optimization due to its high yield of manufacturable optimal designs and ease of implementation. However, parametric optimization might exhibit biases rooted in the parameter template it adheres to, a limitation not observed with FFSO. The latter holds the potential for yielding unconventional designs, which might be less attainable with parametric optimization. FFSO can be computationally burdensome with population-based algorithms when dealing with multi-material scenarios.

In [48], the application of the CNN using a supervised learning strategy was investigated for the multi-objective FFSO of IPMSM. The CNN was trained using three-channel (RGB) pixelated cross-section images of the IPMSM. Two distinct workflows were proposed. In the first workflow, the dataset for training was generated by conducting preliminary FFSO with low population size, aiming to include designs possessing high average torque properties that might not be achievable through random data generation. The deep CNN, based on the GoogleNet architecture [209], served as a grade classifier, classifying the given IPMSM design based on torque ripple range and average torque determined by FE analysis. Once trained, the CNN was used as a meta-model to assess the performance of the given IPMSM design, replacing the FE solver in the primary MOO where the objective was to minimize torque ripple while maximizing the machine's average torque. This MOO was executed using NSGA-II [44]. The primary motivation was to reduce the number of FE simulations during optimization with a large population using the CNN. The CNN's performance was gauged for each MOO generation by calculating a probability (as shown in equation 2 of [48]) to determine the number of samples needing FE analysis per generation. The second method used the trained CNN as a torque constraint estimator while minimizing iron losses. Lower prediction accuracy for torque ripple was observed, potentially due to a weak correlation between material and torque ripple. However, numerical results showed reduced computational costs while ensuring MOO quality. Extending this work in [186] explored the trained CNN's applicability to different problems and models with slightly different shape and performance criteria variations. It should be noted that this research was confined to a fixed magnet shape, position, and a predetermined number of pole pairs. In the latest study [190], a method was presented using a CNN to interpret and visualize the influence of local structures in electric motors based on shapes generated by FFSO using genetic algorithm NSGA-II [44]. The CNN was trained using supervised learning to predict motor characteristics from the three-channel pixelated shapes. By repeatedly processing deformed shapes, heatmaps were created to highlight the importance of specific motor structures (refer to Fig. 3 in [190]). This approach was applied to a PM motor model, allowing for targeted optimization, and it provided insights into enhancing the motor's mechanical stress resistance without compromising torque, with the capability to refine the original shape based on a specific objective function. The trained CNN model is limited to shapes similar to the training geometries of PM motor. Supervised learning based training relies on training data with specific input-output pairs, limiting its generalizability and introducing potential biases based on the data generation method. Such models can be highly accurate within their training data's design space, but their performance can be unreliable

for unseen geometries or conditions, often requiring new datasets and model training for variations in problem conditions. To address these challenges to some extent, a deep RL-based method for FFSO of SynRM is proposed in [99]. To apply the deep RL algorithm, the FFSO problem is defined as an MDP by discretizing the design space with the aim of obtaining an optimal material distribution when maximizing the average torque performance and fulfilling volume constraints. An MDP consists of state, action, reward, environment, and episode [206]. The state represents the current status of material distribution in the design space of SynRM, assisting the RL agent (CNN) in decision-making for action selection. With each step in the design space, the agent predicts an action; it receives feedback as a reward score from the environment. The reward function conducts conventional FE simulations to evaluate objectives (e.g., average torque) to produce the reward. In this context, when the agent predicts an action, the module or controller, functioning like a cell assembly, displays a material pathway (refer to Fig. 1 in [99]). The objective is to maximize the cumulative sum of rewards over episodes. The Actor-Critic RL algorithm [206, Chapter 13] is implemented where the Actor and Critic both represent network structures, combining CNN and dense layers to form two networks (refer to Fig. 5 [99]). The Actor network outputs the probability distribution of each potential action, whereas the Critic network predicts a value, evaluating the Actor network's predicted actions. Both networks are trained simultaneously. The Actor-Critic agent was trained on three different design space scenarios and tested on scenarios not part of the training. Numerical results showed that the trained Actor-Critic agent can optimize SynRM performance while strictly adhering to volume constraints across diverse design scenarios, reducing computational costs significantly compared to traditional evolutionary optimization algorithms. This work is limited to a single RL agent system, and the training phase is computationally demanding.

In the latest study [86], the authors specifically focused on using DNN-based RL for PMSM control. They presented a field-oriented control algorithm that incorporates a deep RL feedback loop as an alternative to the traditional Proportional Integral controller for inner current control. The deep RL agent receives feedback (reward) and the following state (current errors) from the PMSM drive (environment; refer to Fig. 5 of [86]) and takes action by outputting the required voltage for the PMSM through an inverter. Numerical results indicate that this approach surpasses the traditional Proportional Integral controller in speed tracking without need for additional speed feedback.

In recent years, several articles have featured the use of DL-based generative models, such as VAE (refer to Sec. 3.2.3.2) and generative adversarial networks (GANs) [71], for FFSO of electrical machines by learning the underlying input data distribution. For instance, in [121], a deep convolutional GAN is trained on pixelated images of partial IPMSM rotor cross-sections in an unsupervised manner to generate IPMSM rotor shapes for FFSO. The GAN comprises two CNNs, a generator and a discriminator, which are trained together in an adversarial manner [71]: the generator aims to produce rotor images resembling actual samples, while the discriminator strives to differentiate between real and generated fake images. The dataset utilized for this training originated from the time-consuming FE based conventional FFSO process. Another CNN was trained via supervised learning to classify designs based on back-EMF range values. This CNN was subsequently used to classify IPMSM designs produced by the deep convolutional GAN. The final evaluation compared prediction results with those from a commercial FE solver. However, this study was confined to generating rotor designs by targeting a small portion of rotor geometry with a fixed number of magnets and constant stator geometry. The application of VAE for optimizing three different single-layer magnet IPMSM rotor topologies was presented in [189]. In this study, pixelated images of rotor cross-sections, representing the minimal symmetry part of the full machine, were used as inputs for training. The CNN encoder-decoder was trained using unsupervised learning to map the high-dimensional image matrix input to a lower-dimensional latent vector representation and to reconstruct images from the latent input, respectively. Using the latent representation, a DNN was trained with Monte Carlo dropout [58] to

predict flux linkages. The authors carried out MOO in the latent space using trained meta-models and significantly reduced computational time compared to the conventional FE solver. The results showed that Pareto solutions have high prediction errors when considering small PMs, whereas other size PMs have reasonable accuracy. This study was limited to single-layer magnet rotor topologies. In a recent paper [197], FFSO of three distinct PMSM rotor topologies with single and multiple magnet layers was proposed. The authors presented a deep CNN-based generator and KPI predictor workflow for faster PMSM design evaluation. Following unsupervised training, the generator network produces a pixelated image cross-section of the PMSM rotor topology using random latent input. This image is then fed into a trained deep CNN (ResNet-18 [75]) employing supervised learning to predict motor characteristics (refer to Fig. 9 in [197]). In all three articles [121, 189, 197], generative models based on deep CNNs are trained using pixelated images that capture the minimal symmetry portion of electrical machines by considering a fixed stator topology. The training of generative models necessitates a careful selection of hyperparameters. This is especially crucial for GANs. Due to their adversarial training process, where the generator and discriminator pursue opposing objectives, GANs can exhibit numerical instability[71, 84]. In contrast, while VAEs are generally more stable than GANs, they might yield blurry images or less sharp reconstructions[23].

A few DL models applied to rotating electrical machines were explained. These models were based on supervised, unsupervised, or deep RL learning. In a deep RL scenario, one can imagine that an agent interacts with the environment (system) and collects data in the form of experiences. Recently, a physics-based DL model, known as a physics-informed neural network (PINN) [168], has gained the attention of researchers. Distinct from purely data-driven DL models, PINNs are structured to incorporate known laws of physics, commonly expressed as complex nonlinear PDEs, into the loss function. This integration allows PINNs to make meaningful predictions, even with minimal data, serving as a regularizer alongside conventional data loss (e.g., MSE). There are a few recent works where PINNs have been employed for electromagnetic analysis.

In the study by [97], PINNs-based DL models are trained to predict solutions to PDEs across three different problems: a single-domain 2D electrostatic box, a multi-domain electrostatic box, and a multi-domain magnetostatic domain. Here, with the PINN, two learning strategies are employed: the DNN network is trained in a supervised manner using boundary data points, while the PDE component of the loss function, expressed as a residual function, is treated as unsupervised learning. The input to the PINN network consists of collocation points defined by 2D spatial coordinates, and the output is the field solution at those points, for instance, in the form of the MVP (refer to Sec. 2.2.1) for a multi-domain magnetostatic problem (refer to Fig. 1 in [97]). Numerical evaluations are carried out by comparing field results with solutions from the FE solver. The potential for transfer learning across different materials in the multi-domain scenario is also explored to expedite training. Additionally, the authors propose a hybrid modeling approach. This involves incorporating actual solutions at specific collocation points within the domain as data loss during training, thereby merging PINN with data-driven supervised training. This hybrid strategy outperforms the PINN approach that trains only with boundary points. However, this work was limited to analyzing a very simple geometric problem with fixed geometry parameters. Therefore, every time the geometry changes, the network must be retrained for the new geometry. In [9], the authors sought to tackle this issue by formulating a variational principle for a parameterized version of 2D magnetostatic problems. The potential of training a PINN to predict the magnetic flux density concerning changing design parameters, such as geometry and electrical properties, is explored. In this study, the problem domain under consideration is the EI-core electromagnet, which has ten varying design parameters. The PINN network is fed varying design parameters and collocation points. It predicts the MVP, and the subsequent calculations lead to the determination of the magnetic flux density solution at each input collocation point for the given geometry, all via automatic differentiation (refer to Fig. 1 in [9]). However, the proposed approach is limited to very

simple geometries. It is unclear how to scale it up for more complex geometries, such as rotating electrical machines; thus, applying the proposed method may be difficult. In a recent study [201], the PINN-based method was proposed to handle complex geometries like the PMSM. This method suggests a domain decomposition approach where separate networks (as illustrated in Fig. 1 of [201]) are trained for the rotor and stator domains, given their heterogeneous properties during operation; for instance, the stator remains stationary while the rotor rotates. An additional loss term, termed as "interface loss", has been counted to manage discontinuities and inconsistencies between networks for these domains. These trained PINNs are designed to capture the complete electromagnetic responses comprising electric field, magnetic flux density, MVP, and magnetic field under varying operational conditions for each spatial coordinate. The results were evaluated against those from an FE solver and showcased reasonable accuracy. From an application perspective, the PINN can be viewed as a mesh-free solver, offering reduced computational costs compared to the FE solver. However, the authors only considered PMSMs with static geometric parameters in this proposed approach. This means PINNs were trained for spatial coordinates specific to a single machine (fixed geometry parameters).

Applying PINNs presents particular challenges. PINNs can be computationally expensive, often require retraining for new designs, and might not accurately represent solutions when dealing with complex geometries or boundary conditions. They necessitate careful tuning of hyperparameters and selection of network structure, and can exhibit numerical instability during training (e.g., vanishing gradient). Additionally, their convergence is not always guaranteed for complex problems [42, 221].

Table 3.2 provides a high-level summary of the reviewed literature. All these recent papers have highlighted the use of DL in electrical machines. From the literature review, it is evident that most, if not all, DL networks rely on either image-based (2D cross-section) or parameter-based inputs (geometry, electrical properties, system specifications, materials, collocation points, etc.) to approximate target KPIs. These networks utilize either a single learning strategy or a combination of those outlined in Sec. 3.1.1.

Table 3.2: Summary of literature on DL applications in electrical machines

References	DL network	Input	Optimization	Electrical machine	Learning strategy	KPIs
Liang et al.[87]	DBN	Scalar parameters (system)	Parametric	PMSM	Supervised	Efficiency
Gletter et al. [66]	MLP	Scalar parameters (geometry and system)	Parametric	PMSM	Supervised	Power, max.torque
Kurtović et al. [116]	DNN	Scalar parameters (geometry)	Parametric	FSM	Supervised	Torque related KPIs
Khan et al. [96]	CNN	Image (2D cross-section)	Parametric	PMSM	Supervised	Magnetic field
Khan et al. [94, 98, 100]	DNN, CNN, RNN	Scalar parameters (geometry, electrical) Image (2D cross-section)	Parametric	PMSM	Supervised	Efficiency map Power factor map
S.Doi [48]	CNN	Image (2D rotor cross-section)	FFSO	PMSM	Supervised	Torque related KPIs
Sasaki et al. [186]	CNN	Image (2D rotor cross-section)	FFSO	PMSM	Supervised	Average torque Torque ripple
Sato et al. [190]	CNN	Image (2D rotor cross-section)	FFSO	PMSM	Supervised	Stress, torque
Khan et al. [99]	CNN	Image (2D design space)	FFSO	SynRM	RL	Average torque
Jegan et al. [86]	DNN	Scalar parameters (system parameters)	-	PMSM	RL	Voltages for motor control
Lee et al. [121]	GAN, CNN	Image (Small part of 2D rotor cross-section)	FFSO	PMSM	Unsupervised,Supervised	Back-EMF
Sato et al. [189]	VAE, DNN	Image(symmetry portion of 2D rotor cross-section)	FFSO	PMSM	Unsupervised,Supervised	Average torque
Shimizu et al. [197]	GAN, CNN	Image(symmetry portion of 2D rotor cross-section)	FFSO	PMSM	Unsupervised,Supervised	Motor characteristics
Khan et al. [97]	PINN	Scalar parameters (2D collocation points)	-	-	Unsupervised,Supervised	Electromagnetic analysis (fixed geometry of 2D box domain)
Beltrán et al. [9]	PINN	Scalar parameters (2D collocation points, geometry)	-	-	Unsupervised	Magnetostatic analysis of simple 2D EI-core
Seho et al. [201]	PINN	Scalar parameters (2D collocation points)	-	PMSM	Unsupervised,Supervised	Electromagnetic analysis (fixed geometry)

Research goals

The main aim of this thesis is to develop methods based on deep learning algorithms to accelerate the optimization process of electrical machines during the design phase. In the subsequent chapters, the following research goals will be addressed:

- Examine the comparative performance of data-driven models based on various input representations, precisely image- and parameter-based representations, for quantifying electrical machine performance in a high-dimensional design space.

- Improve the prediction accuracy and generalization by investigating the possibility of combining data-driven and physics-based models. A sub-goal related to this is to analyze the performance of the proposed methodology compared to the conventional FE-based workflow in industrial multi-objective optimization settings.

- Develop a methodology for the simultaneous optimization of electrical machines with different parameterizations when using scalar parameter-based representations, aiming to improve generalization across varied design spaces.

3.4 Summary

In this chapter, the basics of DL have been covered. A brief introduction to AI and various learning methods was provided initially. Several DL architectures, including DNN, CNN, and VAE, which will be used in the subsequent research, were then detailed. Recent works highlighting DL applications to electrical machines were reviewed, and the research goals of this treatise were outlined. In the following chapter, a data-driven DL approach for approximating a large number of cross-domain KPIs for different PMSM representations will be introduced.

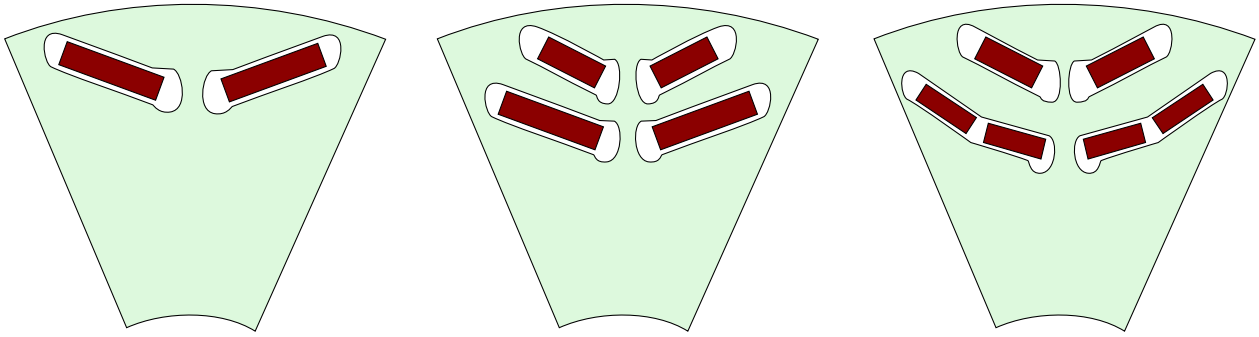
4 Data-driven models for optimization of electrical machines

In this chapter, a data-driven DL approach for predicting a large number of cross-domain KPIs for PMSMs is introduced, utilizing various DL models: the DNN (Sec. 3.2.1) and the CNN (Sec. 3.2.2). The main emphasis is on analyzing how different input representations of PMSMs, specifically parameter- and image-based, impact the prediction accuracy of the DL models with the given number of training samples.

The first section (Sec. 4.1) briefly provides an introduction, and the reparametrization scenario pertaining to parameter- and image-based meta-models is discussed. Next, the generalized MOO problem for designing electrical machines is formulated (Sec. 4.2). Section 4.3 describes the data generation process and details of industrial datasets related to various input representations of PMSMs. Section 4.4 illustrates hyperparameter tuning, ANN architectures for different PMSM input representations, and their training details. This is followed by numerical analysis in Sec. 4.5. The majority of the content and structure of this chapter is drawn from our work in [152].

4.1 Introduction

The performance of any electrical machine design is quantified by evaluating various cross-domain KPIs. These KPIs include material cost, maximum torque, sound power level, efficiency, etc. Typically, these KPIs are obtained during the post-processing stage of the simulation workflow. In this thesis, the simulation workflow is explained in Figure 2.5 and provided a few examples in the Sec. 2.2.3. When dealing with a large design space, the numerical optimization of electrical machines becomes computationally intensive due to the time-consuming FE simulations. Generally, the MOO of any rotating electrical machine entails ten to twenty KPIs and a vast input design space that consists of approximately fifty to seventy different design parameters. Operating within such a large design space, the FE-based workflow prolongs the duration of the design development cycle in industry. To address this issue, the use of different ANN-based meta-models has increased in the past decade. A few articles are reviewed in Sec. 3.3. For example, in [87], DL-based multiple-output regression was presented to predict KPIs such as efficiency, speed, and torque for the performance analysis of PMSMs. Deep CNNs were utilized to predict magnetic field solutions for various electromagnetic devices, such as a transformer, a coil in the air, and an interior PM machine [96]. To monitor the real-time fluctuating temperature of PMSMs, RNNs and CNNs were employed [110]. The CNN-based meta-model was trained to assess electric motor performance, aiming to reduce FE calculations for topology optimization [187]. This approach was further developed for multi-objective topology optimization using a deep CNN [48]. The fully connected DNNs were trained to estimate torque for distinct states of (transient or steady) interior PMSM drives [128]. DL-based models such as RNN, CNN, and DNN compute efficiency maps for a motor drive in [94]. In [66], it is illustrated how the optimization of hybrid EVs at the system



(a) Single V **(b)** Double V **(c)** VC-Design

Figure 4.1: Different PMSM rotor topologies. Figure taken from [152, Fig. 1].

level is performed by approximating non-linear system behavior using neural networks. In [231], a DNN was employed as a meta-model for the shape optimization of PMSMs. The acceleration of optimization in electromagnetics through a combination of a CNN-based model and a reduced FE model is studied in [8]. These are a few examples where various data-driven DL models for quantifying different KPIs are presented. In all of these works, electrical machine design is primarily described in two forms: first, in an image form, i.e., the cross-section of the machine (e.g. Figure 4.1), and second, as scalar parameters that include various design parameters such as geometry, electrical, and material. Also, most of the approaches mentioned above are limited to fewer single-valued KPIs with lower-dimensional scalar input. In this chapter, a supervised learning-based data-driven DL approach is presented to predict a large number of cross-domain KPIs of PMSM with high-dimensional scalar input. Two queries are investigated: first, to what extent a conventional scalar input-trained model can accurately predict the KPIs when provided with a specific number of samples. The other query pertains to the performance of image input-trained models, which possess greater generality and topology invariance, with respect to image resolution and in comparison to scalar-trained models.

4.1.1 Reparameterization scenario for parameter- and image-based meta-models

In a reparameterization scenario, scalar parameter-based and image-based meta-models exhibit distinct behaviors. Once a scalar parameter-based meta-model is trained, it becomes deterministic, meaning its performance solely depends on the scalar parameters used during training. On the other hand, an image-based meta-model provides a more general way of evaluating the performance of electrical machines. Its performance relies solely on the specific image domain it was trained on, without concern for the process of generating the image. If the system is re-parameterized in such a way that the image domain remains unchanged, it becomes possible to predict the KPIs using the same image-based meta-model that was trained. However, this is not applicable to parameter-based meta-models, as any re-parameterization modifies the input space. For instance, as shown in Figure 4.2, two plates with different parameterizations produce the same image domain. The rectangular steel sheet 1 is generated using scalar parameters a and b, while the rectangular steel sheet 2 is created using parameters d and e, which are different from a and b. If only a scalar meta-model is trained using the input domain of parameters a and b, and then tested using the input domain of parameters d and e to make predictions about the stiffness of the steel sheet, reasonably accurate predictions will not be achieved unless a correct transformation is applied. In contrast, an image-based meta-model will still yield accurate predictions in such cases.

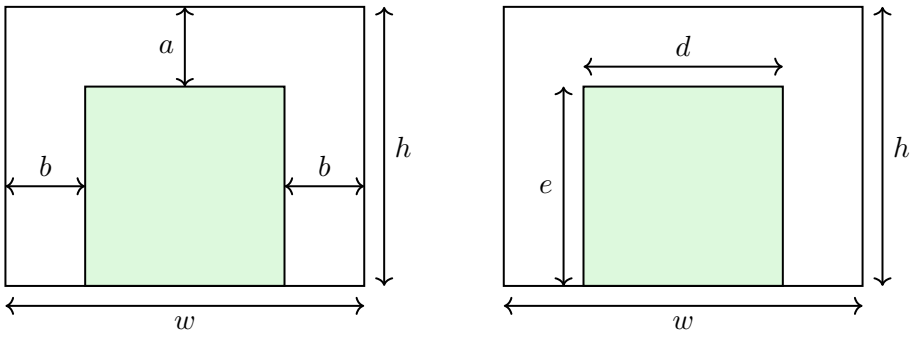


Figure 4.2: Illustration of differently parameterized rectangular steel sheet with the identical image domain ($d = w - 2b$ and $e = h - a$). Figure taken from [152, Fig. 3].

4.2 Problem formulation

The broader goal while designing an electrical machine is to find the optimal design which satisfies different conflicting criteria, such as maximizing torque, power, and efficiency while minimizing cost, carbon footprint etc. It can be achieved with MOO. All the KPIs are dependent on the design vector $\mathbf{p} \in \mathbb{P} \subset \mathbb{R}^n$. The design vector includes a set of selected parameters (geometry, electrical, and material) which is varied in a defined range to obtain the optimal design for the given targets. The MOO problem can be described as follows

$$\min_{\mathbf{p}} \quad k_e(\mathbf{p}), \qquad e = 1, \ldots, m \tag{4.1}$$

$$\text{s.t.} \quad c_f(\mathbf{p}) \leq 0, \qquad f = 1, \ldots, l \tag{4.2}$$

$$p_g^{\mathrm{L}} \leq p_g \leq p_g^{\mathrm{U}}, \qquad g = 1, \ldots, n \tag{4.3}$$

where $c_f(\mathbf{p})$ are constraints, e.g. geometry feasibility check and boundary conditions. $k_e(\mathbf{p})$ calculates target KPI. p_g^{L} and p_g^{U} are parameter bounds.

As explained in the subsection 2.2.3, the process of calculating a KPI ($y_e = k_e(\mathbf{p})$) starts with defining the design vector $\mathbf{p}$, followed by a series of steps such as magnetostatic FE simulation and post-processing. The output of one step may be input to another step, ultimately leading to the target KPIs calculation. Each of these intermediate steps involves different functions, which may not directly require a design vector $\mathbf{p}$ but only the output of other functions as an input. Each target KPI function ($k_e(\mathbf{p})$) can be a combination of many other non-linear functions those directly or indirectly dependent on the design vector $\mathbf{p}$. Therefore, it can be assumed that the whole non-linear system is an unknown or "black box"function to some extent concerning the design vector $\mathbf{p}$ for calculating target KPIs. The magneto-static FE simulation process is time-consuming and computationally expensive. It restricts exploring high-dimensional design space during MOO due to limited computing sources.

Suppose a simulation dataset $\mathcal{D}$ is given as

$$\mathcal{D} := \left\{ (\mathbf{p}^{(1)}, \mathbf{y}^{(1)}), ..., (\mathbf{p}^{(\mathbf{N})}, \mathbf{y}^{(\mathbf{N})}) \right\}, \tag{4.4}$$

where $\mathbf{y} \in \mathbb{Y} \subset \mathbb{R}^m$ is a target KPIs vector for each machine design in the dataset $\mathcal{D}$. Using the dataset $\mathcal{D}$, the aim is to approximate an expensive multiple output function $\mathbf{K} : \mathbb{P} \to \mathbb{Y}$ for KPIs calculation,

i.e.,

$$\hat{\mathbf{y}} := \hat{\mathbf{K}}(\mathbf{p}), \tag{4.5}$$

which is computationally cheap and faster while maintaining high prediction accuracy. $\hat{\mathbf{K}}(\mathbf{p})$ is considered as a metamodel or surrogate model. $\hat{\mathbf{y}}$ is a predicted target KPIs vector.

4.3 Dataset generation

The quality and diversity of a dataset are fundamental to a DL algorithm's ability to generalize and make accurate predictions in real-world scenarios. A dataset that is both sufficient in size and of high quality ensures that the DL model captures essential features and minimizes biases during training. Conversely, an inadequate dataset can lead to overfitting and poor generalization. Figure 4.3 depicts the general workflow for data generation of rotating electrical machines. This workflow is employed consistently throughout the

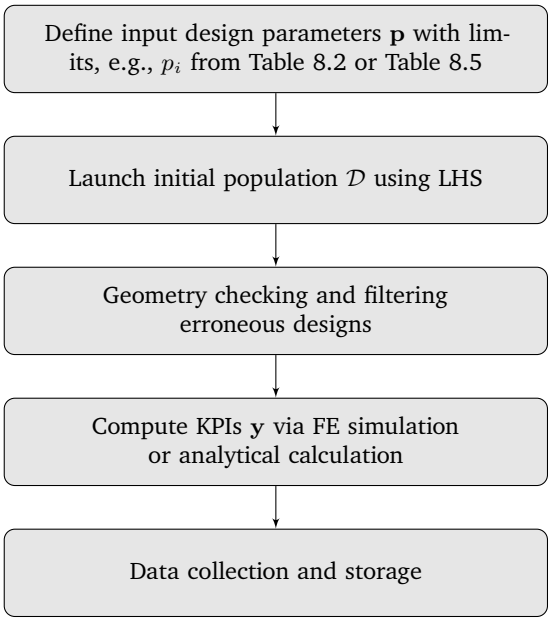


Figure 4.3: General workflow for generating a dataset of rotating electrical machines. Figure based on [151, Fig. 3].

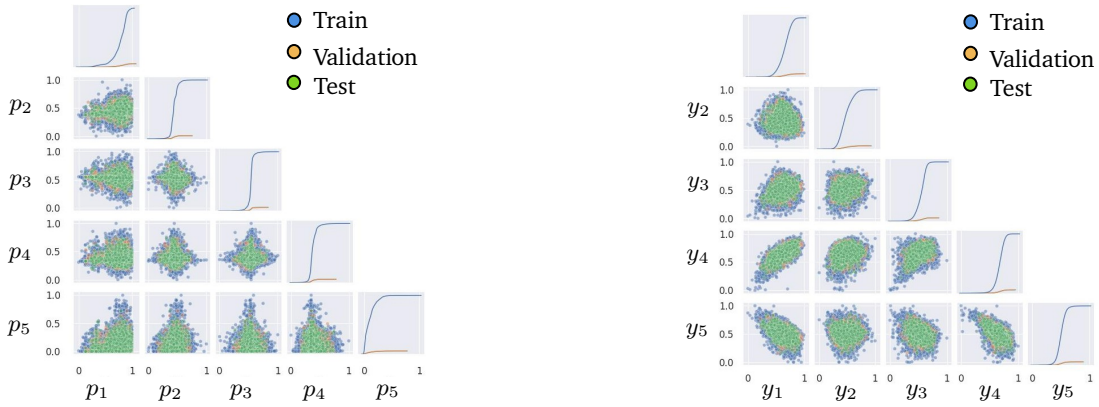
treatise in real-world industrial settings. As described, the first step is about defining design parameters with their respective ranges. The population is then triggered using a suitable sampling technique in a multidimensional parameter space. Several sampling techniques are available for this purpose, such as Monte Carlo sampling [192], LHS [137], Stratified sampling [157], and many other similar methods briefly reviewed in [196]. Any of these methods can be chosen by considering specific needs. The LHS is used in this work since it efficiently explores the parameter space by ensuring that the generated samples are evenly distributed. With the LHS, the parameter space is divided into uniformly probable intervals across each dimension. Then, from each of these intervals, a sample is randomly chosen, adhering to the constraint that only one sample can be selected from each interval in any given dimension. This provides more comprehensive coverage of the parameter space [137, 196]. After generating the population, the

geometric feasibility of each design is assessed using the CAD building software (e.g., Gmsh [62]) to identify and eliminate any erroneous designs. Then, the KPIs are computed with FE simulations or analytical calculations (refer subsection 2.2.3) and stored in a suitable database.

All PMSMs datasets throughout this treatise are generated assuming magnetic state symmetry. Therefore, either a half-pole or a full-pole cross-section is considered in all datasets, leveraging the magnetic-state symmetry of the electrical machines. This chapter examines two such datasets, with each machine design simulated using magneto-static FE simulation. The specifics of these two PMSM datasets are discussed in the following subsections. The content of the next two subsections section is based on our work in [152].

4.3.1 Dataset 1

Each PMSM design realization can be defined with a parameter vector $\mathbf{p}^{(i)}$, and the corresponding output KPIs vector $\mathbf{y}^{(i)}$, where i is the design number in the dataset. The dataset is abstractly expressed by equation (4.4). In this dataset, the rotor model (VC type) is solely incorporated, utilizing $n_{\text{rotor}} = 49$ scalar parameters for sample generation. All rotor parameters are listed in Table 8.3. The entire figure cannot be annotated due to corporate secrecy reasons. The details of seven significant stator parameters ($n_{\text{stator}} = 7$), which provide information about the stator geometry, can be found in Table 8.2. The total number of scalar parameters is $n_1 = n_{\text{rotor}} + n_{\text{stator}} = 56$. The details of other input electrical parameters, such as phase current, phase voltage, and the number of slots per pole per phase, which remain constant during the simulation, are provided in Table 8.1. Similarly, material properties, for example, remanence factor, copper filling factor, and type of magnet cluster remain constant. A brief description of the KPIs is given in the 8.4. The distributions of input parameters and KPIs are visualized on affine 2D subspaces in Figures 4.4a and 4.4b, respectively. After filtering out erroneous designs (e.g., violating geometric constraints) from the initial population, a total of $\mathcal{D}_1 = 68099$ valid samples are generated. The input parameters are independent and exhibit an inhomogeneous distribution, as depicted in Figure 4.4a. A few parameters and KPIs are displayed as pair plots. In the initial stage of data analysis, a pair plot provides a faster way to visualize a comprehensive overview of the dataset. It can quickly visualize the pairwise relationships between multiple parameters in a dataset. This may help in understanding how variables are correlated or how they change relative to each other. By visualizing the data, it can be easier to identify any clear trends



(a) Parameters distribution.

(b) KPIs distribution.

Figure 4.4: Dataset 1: parameters and KPIs distribution. Figure taken from [152, Fig. 4].

or outliers that might exist. The outlier in a dataset refers to a sample that possesses substantially different values than other samples. Diagonal plots show a smoothed version of histograms, which visualize the univariate distribution of individual parameters and KPIs, whereas off-diagonal scatter plots illustrates the bivariate relationships between different pairings of the parameters and KPIs in the dataset. It can be seen that there is some correlation between KPI y_1 (costs of active parts), y_4 (maximum power of machine), and y_5 (weighted efficiency value).

Pixel resolution study: Before training any machine learning algorithm, it is necessary to transform raw data into a suitable format, such as a matrix of numbers or a vector of continuous values. In this study, a CNN is trained using image-based data. Thus, pre-processing was performed on the raw image data, which involved transforming the CAD model into a pixelated image. Each pixel in the pixelated image represents a unique identifier value (UIV) corresponding to a PMSM component. In this dataset, the pixelated rotor model consists of three UIVs: air (0), metal body (1), and magnet (2). Figure 4.5 illustrates an example from the dataset, depicting both the CAD model and the corresponding pixelated cross-sectional image of a half pole of the rotor. The transformed dataset is then used for the CNN training.

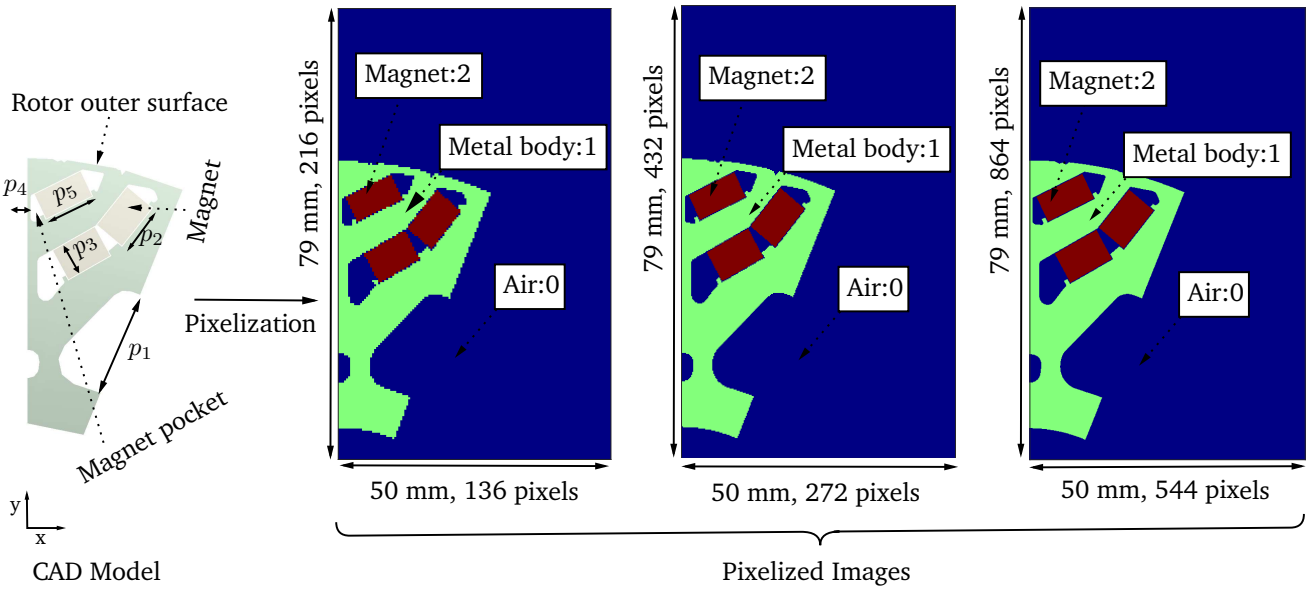


Figure 4.5: Pixelization dataset 1. Figure based on [152, Fig. 5].

An intriguing observation is that the prediction of KPIs in the scalar parameter-based meta-model depends solely on the scalar parameters. Even a minor variation in a single parameter typically results in a different prediction. In contrast, the image-based model relies on the accuracy of the image representation. During the initial study, a resolution of 136×216, pixels is chosen for the bounded geometrical domain of 79 mm$\times 50$ mm. This resolution describes a precision of 0.36 mm/pixel, which shows a minimum change of any input scalar parameter per pixel. It means that approximately 3 pixels are required to represent a variation of 1mm in a geometry parameter. As a result, it impacts the sensitivity of the meta-model. This implies that increasing the pixel precision enhances the interpretation of variations in geometry parameters. Table 4.1 provides an explanation for five rotor parameters. The first column in Table 4.1 describes the range from minimum to maximum, while the next three columns show the precision value in mm per pixel. The last three columns provide information on the number of pixels needed to represent the maximum variation in the given geometry parameter. This is based on parameter ranges and is calculated using

Table 4.1: Pixel Resolution Detail concerning geometry parameter variation with dataset 1. Table based on [152, Tab. 4].

	Range [mm]		Image resolution in pixels, X-direction=50mm, Y-direction=79mm					
	Min.	**Max.**	Precision [mm/pixel]			Pixel value		
	[mm]	[mm]	136×216	272×432	544×864	136×216	272×432	544×864
p_1	6.64	12.97	0.36	0.18	0.09	18	35	70
p_2	7.19	9.48	0.36	0.18	0.09	7	13	25
p_3	5.13	6	0.36	0.18	0.09	3	5	10
p_4	0.8	1.51	0.36	0.18	0.09	2	4	8
p_5	7.66	10.27	0.36	0.18	0.09	8	15	29

the pixels required for a unit length. Total three resolution values are compared: 136×216, 272×432, and 544×864 pixels.

4.3.2 Dataset 2

The dataset 2 has a different parametrization than dataset 1. The geometry representation comprises the rotor full pole and stator cross-section. The cross-sectional image of one sample is described in Figure 4.6. In the pixelated form, the UIVs are assigned as follows: air (0), metal (1), magnet (2), and copper (3). It

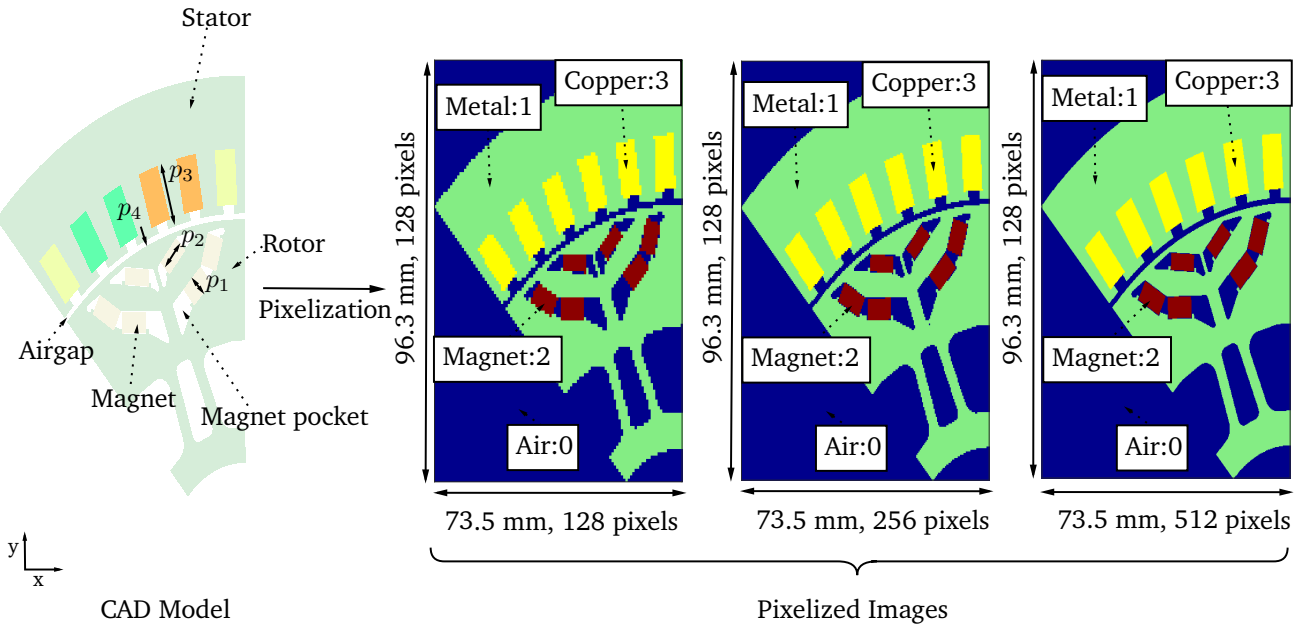
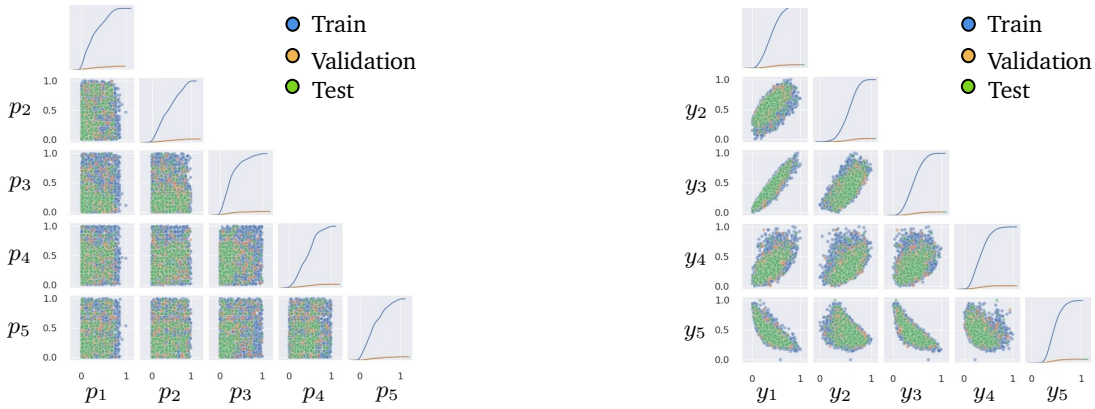


Figure 4.6: Pixelization dataset 2. Figure taken from [152, Fig. 6].

comprises $n_2 = 12$ varying scalar parameters (stator and rotor). All the parameters with their respective ranges are detailed in Table 8.5. Likewise, as in dataset 1, the other constant parameters are described in Table 8.1. After filtering the initial population, the total number of valid samples in dataset 2 is $\mathcal{D}_2 = 7744$.

The joint distribution of a few scalar parameters and KPIs is illustrated in Figure 4.7a and Figure 4.7b, respectively. The joint distribution of input scalar parameters is all but uniform, whereas KPIs are non-uniformly distributed. The KPIs y_1 and y_3 are strongly correlated, whereas y_4 and y_3 show a weaker correlation. The short description of KPIs is given Table 8.6.



(a) Parameters distribution.

(b) KPIs distribution.

Figure 4.7: Dataset 2: parameters and KPIs distribution. Figure taken from [152, Fig. 4].

4.4 Network architecture and training details

Before finalizing and training any DL model, it is necessary to understand the different types of parameters involved in the training process. Most of these parameters have already been explained in Sec. 3.2. In the following subsection, the hyperparameter tuning approach followed throughout this thesis to obtain and train the proposed ANNs is explained. Setting the correct combination of hyperparameters is a challenging task, often viewed as an optimization problem, where the set of parameters are consistently adjusted to minimize the cost (loss) function. Therefore, the objective in this thesis is to obtain ANNs that maximize the test performance with reasonable prediction accuracy while mitigating overfitting and underfitting problems, and improving the generalization and interpretability of the ANN.

4.4.1 Hyperparameter tuning

Any class of ANN broadly comprises two types of parameters: trainable parameters and hyperparameters. Trainable parameters primarily include weights and biases that are tuned during ANN training with respect to the cost function (refer to Sec. 3.2.1.1). They are directly dependent on the dataset used to train the ANN. On the other hand, hyperparameters are a group of parameters that are typically set before the training process. In this thesis, for better understanding, they are broadly classified into two categories:

- Network hyperparameters: They mainly define the structure of the ANN. They are set before the training process begins and mostly remain fixed throughout the training. Examples include the number of layers, their connections, the number of neurons per layer, activation functions, etc.

- Training hyperparameters: They control the training process of the ANN. They influence how the ANN learns from the data and updates its trainable parameters. Training hyperparameters include batch size, loss function,learning rate, number of epochs, optimizer, early stopping criteria, etc. They are usually initialized before training and can be adjusted during the training process to improve the network's performance as needed.

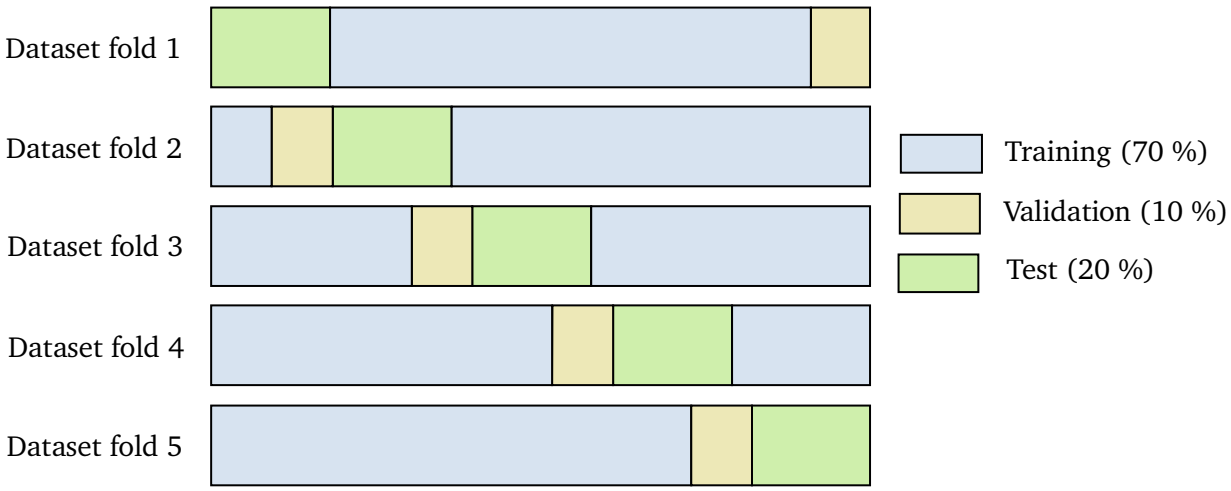


Figure 4.8: Representative image for five fold cross-test and validation.

All these hyperparameters impact model's generalization ability, convergence speed, stability and prediction accuracy. There are several approaches for hyperparameter optimization (HPO) in ANNs, such as random search, grid search, and Bayesian optimization [229]. Each has its own merits and demerits. Grid search exhaustively tests all combinations, making it computationally expensive; random search is quick and versatile, capable of efficiently exploring large hyperparameter spaces, although it doesn't guarantee finding the optimal solution [83, Chapter 1]. Bayesian optimization is a probabilistic method, while population-based training evolves models using concepts from natural selection [15]. However, there is no universally best method for HPO, as the choice heavily depends on available computational resources, datasets, and the complexity of the task. Reviews of many such methods and information about their publicly available ready-to-use implementations with ML frameworks can be found in [83, Chapter 6], [229, Section 4], and [15, Section 6].

Tuning approach: The hyperparameters tuning is performed in two steps for most of the DNN models proposed in this thesis.

- **Manual approach:** Since there are many hyperparameters to consider, finding the best possible combination for a large dataset can be a time-consuming and computationally expensive task. Therefore, the first step is to determine a few key parameters, along with their respective ranges, that may have the potential to yield good network performance. This determination is made through a manual search involving trial and error. During the manual search, multiple trials are conducted with different combinations of these crucial parameters. These parameters include the types of network layers, activation functions, loss functions, batch size, learning rate, etc. By systematically exploring these combinations, the settings that may lead to optimal performance are identified. This allows narrowing the search space and focusing on a subset of hyperparameters showing promising performance. It gives insights and aids in making informed decisions about the most influential

parameters by manually evaluating their impact on the model's performance. These parameters can serve as a starting point for further exploration, enabling a balance to be struck between computational computational efficiency and finding optimal hyperparameter configurations.

- **Random approach:** After determining the key hyperparameter set and their ranges for the HPO of ANNs, a random approach is employed to effectively explore hyperparameter combinations. An in-house optimization tool that incorporates the implementation of the Asynchronous Successive Halving Algorithm [126, Algorithm 2] is used. This algorithm aids in efficiently exploring the hyperparameter space by iteratively eliminating underperforming configurations. It is run on an in-house GPU cluster. Within the defined ranges, a set of hyperparameter values is randomly sampled to form network configurations. Each network configuration is evaluated for the train-test-validation split (e.g., 70-20-10) percentages with a five-fold cross-test and validation; see Figure 4.8. Multiple configurations are executed in parallel, utilizing the available resources. The algorithm progressively eliminates underperforming configurations and allocates more resources to promising ones. This ensures a focus on hyperparameter configurations that demonstrate superior performance, bringing closer to the optimal solution. Consequently, combinations of hyperparameters that lead to enhanced generalization performance with reasonable prediction accuracy can be identified.

The content of the next three subsections section is based on [152].

4.4.2 Network architecture

The type of network structure is finalized based on the specific characteristics of the input data used in the training phase. The datasets employed consist of scalar parameters that provide comprehensive information about the rotor and stator structure of the PMSM cross-section. As a result, the densely connected DNN (see Figure 4.9) is employed for training with the data based on these scalar parameters. The dataset 1 consists of image data depicting only a half pole cross-section of the rotor (see Figure 4.5). On the other hand, dataset 2 comprises of image data illustrating both the full pole stator and rotor (see Figure 4.6). For the training of the image-based data, deep convolutional neural network (DCNN) structures as shown in Figure 4.10 are utilized. Furthermore, a combination of image and scalar parameter-based data is utilized, and for that, a multiple-input DCNN structure is adopted as illustrated in Figure 4.11. It should be noted that in both datasets, the images can only capture structural details and, to some extent, material information. However, describing other parameters such as varying current is not possible in the 2D image representation. These additional parameters can be easily included in the scalar-based model. To determine the final three network configurations that yield reasonable prediction accuracy, along with training hyperparameters for each of these input types, the two-step process is followed as explained in the previous Sec. 4.4.1. Firstly, a manual search is performed, with approximately twenty configurations being evaluated. Subsequently, the random approach is employed on a GPU cluster to evaluate an additional roughly hundred configurations. Details of each final tuned network configuration and training specifications are given in the following subsections.

4.4.2.1 Scalar parameter-based DNN structure

The final configuration is obtained after hyperparameter tuning, as shown in Figure 4.9. It consists of five hidden fully connected dense layers. The final network configuration for dataset 1 can be described as $56 \rightarrow 448 \rightarrow 250 \rightarrow 224 \rightarrow 224 \rightarrow 198 \rightarrow 11$, where the first entry represents the number of scalar

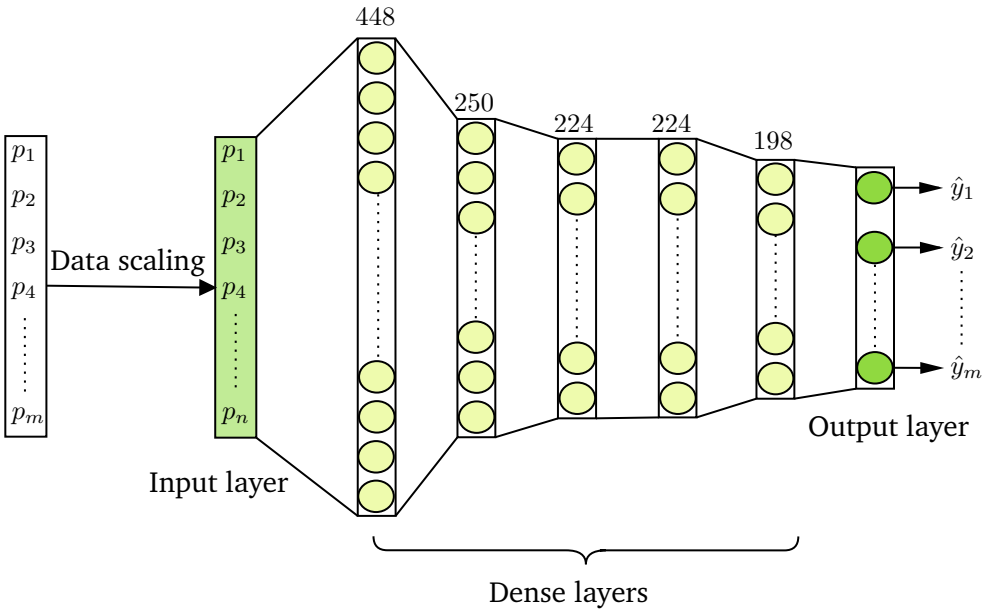


Figure 4.9: Scalar parameter-based DNN. Figure based on [152, Fig. 9].

input parameters, and the last entry represents the number of output KPIs. Similarly, for dataset 2, the tuned network structure can be described as $12 \rightarrow 448 \rightarrow 250 \rightarrow 224 \rightarrow 224 \rightarrow 198 \rightarrow 10$. The elu activation function was obtained during hyperparameter tuning and is applied between the hidden layers.

4.4.2.2 Image-based DCNN structure

As illustrated in Figure 4.10, the tuned DCNN structure can be described in two parts. The first part consists of five 2D convolutional layers that aim to extract spatially correlated features from the visual representation of the geometry cross-section of the PMSM. As described in the Sec. 3.2.2, these layers play a crucial role in identifying key patterns within the data. The second part of the structure comprises fully connected dense layers. These layers utilize the information gathered by the previous convolutional layers and transfer the semantic relevance by mapping the extracted key features to the target KPIs in the output layer. It is important to ensure that the number of dense layers is sufficient to handle complex features effectively. The configuration of the dense layers is intentionally kept identical to that of the tuned scalar DNN structure (see Figure 4.9), allowing for a performance comparison between the scalar-based and image-based meta-models.

The hyperparameters of the DCNN, such as the number of kernels, kernel size, and the number of convolutional layers, are tuned using the random approach. Initially, hyperparameters and their respective ranges are narrowed down based on the details provided in Table 4.2. The inclusion of pooling layers is also explored, but after examination, it is observed that they do not improve prediction performance in these datasets. Moreover, their introduction leads to an unnecessary loss of important spatial information. Therefore, the pooling layers are not used for these datasets. The decision to exclude pooling layers from a DCNN should be made based on the specific requirements of the task at hand, particularly considering factors such as the preservation of fine-grained details, which is crucial for PMSM. The exclusion pooling layers can also lower some computational burden during training by reducing the number

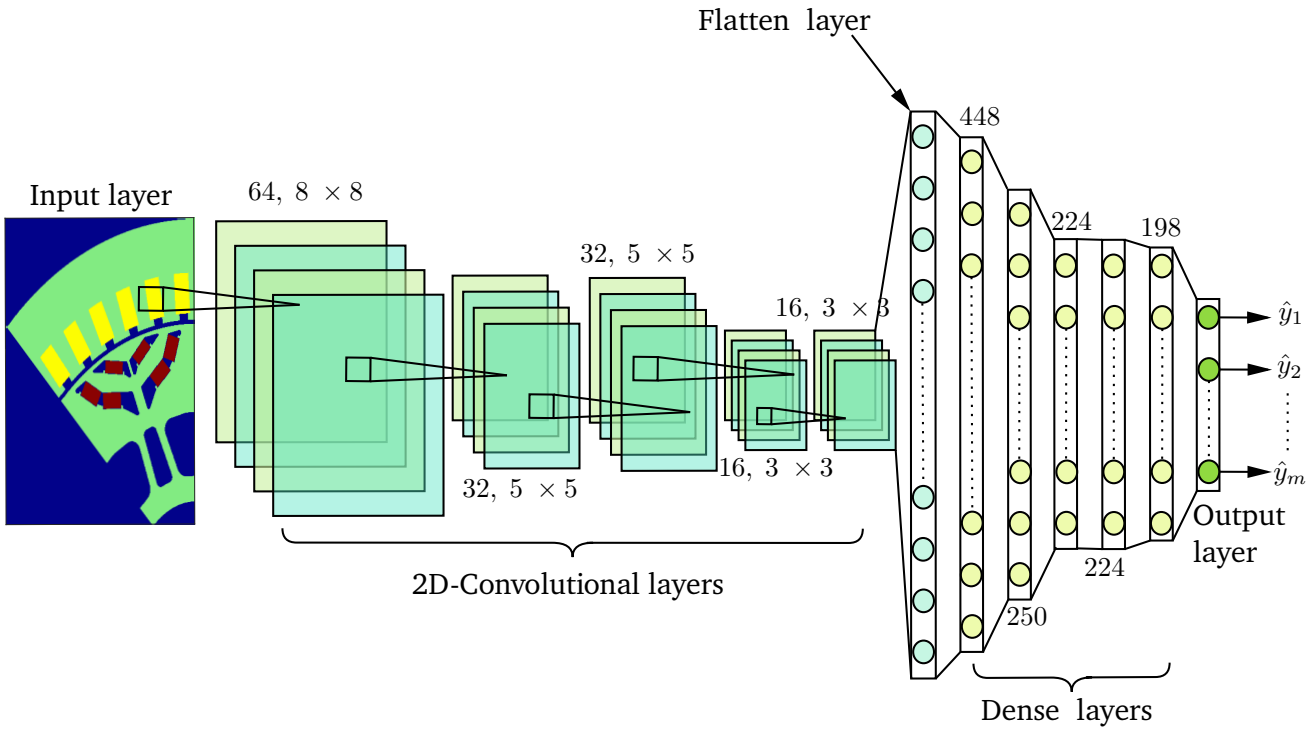


Figure 4.10: Illustration of image-based DCNN for dataset 2. Figure based on [152, Fig. 10].

of trainable parameters. The use of strided convolutions as an alternative down-sampling technique to enhance generalization performance is also explored. The stride is kept at two after experimenting with various stride values listed in Table 4.2. The proposed network structure in Figure 4.10 is designed to be invariant to input layer dimension. For further information on other layers, such as the flatten layer, refer to Sec. 3.2.2.

Table 4.2: Hyperparameter details

Hyperparameter	Value range	Type	Final value
Learning rate	$[10^{-5}, 10^{-4}]$	Continuous	adaptive rate 1000 decay steps
Average number of neuron per hidden layer	$[180, 600]$	Integer	see Figure 4.9
Number of dense layers	$[3, 6]$	Integer	5
Number of 2D convolutional layers	$[4, 7]$	Integer	5
Number of strides	$[1, 3]$	Integer	2
Batch size	$[40, 60]$ with step size 4	Integer	50
Kernel size	$\{3, 5, 8\}$	Integer	see Figure 4.11 and Figure 4.10
Number of filters per convolutional layer	$\{16, 32, 64, 128, 256, 512\}$	Integer	see Figure 4.11 and Figure 4.10
Activation functions	tanh, softplus, relu, elu	Categorical	elu
Optimizer	SGD, Adamax, AdaGrad, Adam	Categorical	Adam
Loss functions	MSE, MAE	Categorical	MSE

4.4.2.3 Multiple-input DCNN structure

The only deviation from the DCNN described in the previous Sec. 4.4.2.2 is the inclusion of an auxiliary input layer. As depicted in Figure 4.11, this layer is combined with the output of the flatten layer. This

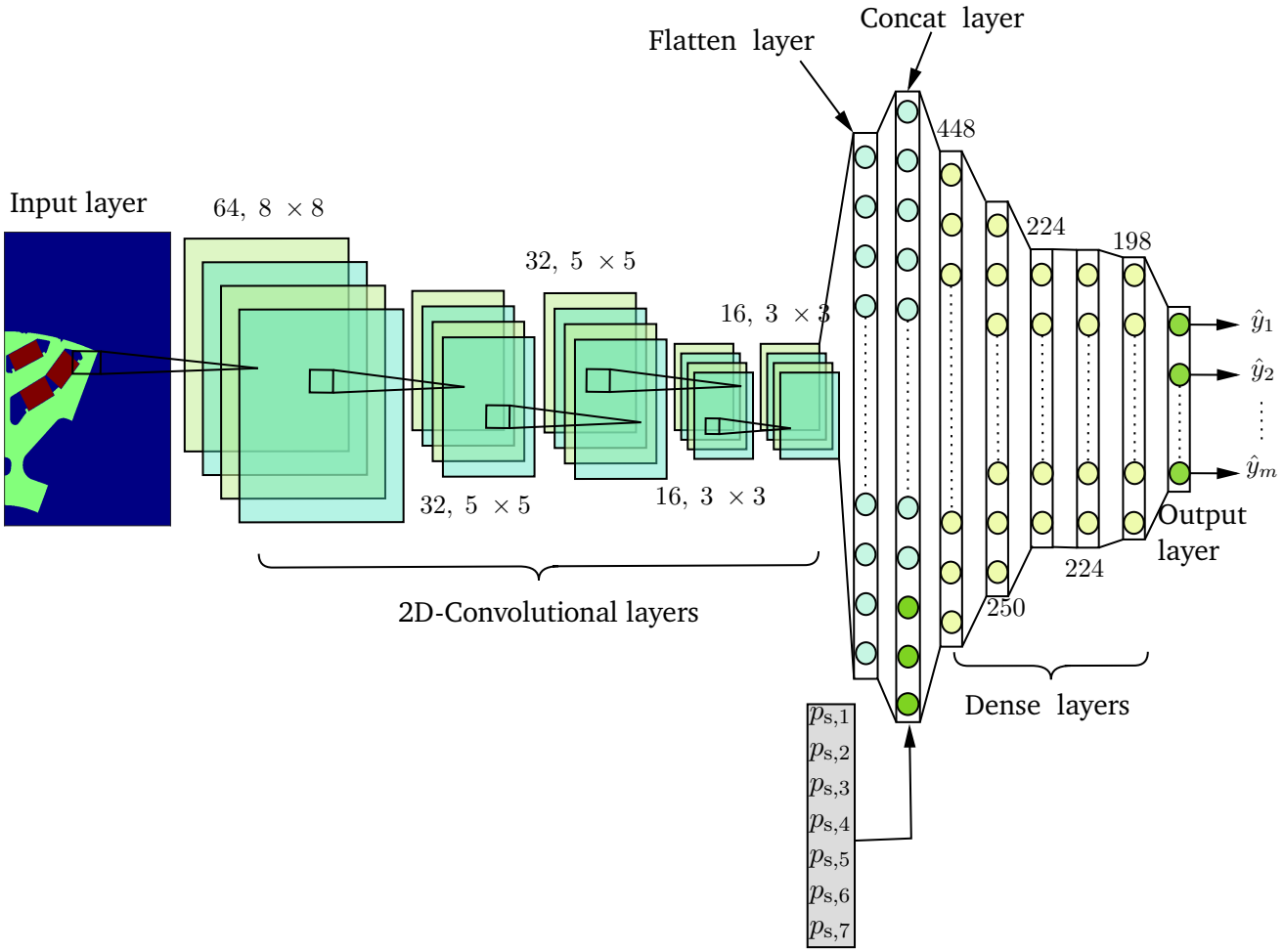


Figure 4.11: Illustration of DCNN with multiple-inputs for dataset 1. Figure based on [152, Fig. 11].

layer is introduced specifically for the dataset 1 to incorporate stator geometry information (see Table 8.2), which is not present in the image input, as it solely represents the cross-section of the rotor. It is important to note that this is not the case with dataset 2 since its image-cross-section already entails both information (stator and rotor).

4.4.3 Training details

It is common practice to partition the dataset before training any ML meta-model. Typically, the dataset is divided into three parts: training, validation, and test sets. In this case, both datasets are divided using a $90 - 5 - 5$ split percentage. Dataset 1, with a total of $\mathcal{D}_1 = 68099$ samples, is divided into $\mathcal{D}_{1,\text{train}} = 61290$ training samples, $\mathcal{D}_{1,\text{validation}} = 3405$ validation samples, and $\mathcal{D}_{1,\text{test}} = 3404$ test samples.

Similarly, dataset 2 consists of a total of $\mathcal{D}_2 = 7744$ samples, divided into $\mathcal{D}_{2,\text{train}} = 6970$ training samples, $\mathcal{D}_{2,\text{validation}} = 387$ validation samples, and $\mathcal{D}_{2,\text{test}} = 387$ test samples. Figure 4.4 and Figure 4.7 depict the training, test, and validation distribution of a few parameters and KPIs for both datasets. Prior to training, all the scalar input parameters and output KPIs are normalized using min-max scalar transformation, ensuring that they fall within the range of [0, 1]. This uniform scaling enhances the prediction performance of the meta-model during training. The general training pseudo-code for all three meta-models is explained in Algorithm 2. The training goal is to minimize the cost or loss function (here, MSE) to tackle the

Algorithm 2 Pseudo-code for meta-model training:

$(\mathbf{P}_{\text{train}}, \mathbf{Y}_{\text{train}})$, $(\mathbf{P}_{\text{validation}}, \mathbf{Y}_{\text{validation}})$, $(\mathbf{P}_{\text{test}}, \mathbf{Y}_{\text{test}})$: Divide the dataset $\mathcal{D}$
▶ Training: 90%, Validation: 5%, Test: 5% of individual dataset $\mathcal{D}$
$\mathbf{n}_{\text{epochs}} \coloneqq 100$, $\mathbf{v}_{\text{p, limit}} \coloneqq 5$, $\mathbf{v}_{\text{p, counter}} \coloneqq 0$
▶ Hyperparameter initialization: no of epochs, validation patience (VP) limit and VP counter
$\mathbf{b}_{\text{size}} \coloneqq 50$ ▶ Hyperparameter initialization: batch size
$\mathbf{l}_{\text{rate}} \coloneqq 10^{-4}$ to 10^{-5}
▶ Hyperparameter initialization: start to end learning rate scheduler with 1000 decay steps
$\mathbf{K}_\gamma$: Initializing meta-model trainable parameters (γ)
▶ Glorot uniform initializer [67]
for $e \coloneqq 1$ to $\mathbf{n}_{\text{epochs}}$ **do**
  $\mathbf{P}_{\text{t,shuffle}}, \mathbf{Y}_{\text{t,shuffle}} \coloneqq \text{Shuffle}(\mathbf{P}_{\text{train}}, \mathbf{Y}_{\text{train}})$ ▶ Shuffle training data randomly at every epoch
  for $i \coloneqq 1$ to $\mathbf{n}_{\text{iter}}$ **do**
    ▶ Compute $\mathbf{n}_{\text{iter}} \coloneqq \left\lceil \frac{n_{\text{train}}}{\mathbf{b}_{\text{size}}} \right\rceil$, where n_{train} is the number of training samples.
    $\mathbf{p}_{\text{batch}}, \mathbf{y}_{\text{batch}} \coloneqq \text{getBatchOfData}(\mathbf{P}_{\text{t,shuffle}}, \mathbf{Y}_{\text{t,shuffle}}, \mathbf{b}_{\text{size}}, i)$
    ▶ Get current batch
    $\hat{\mathbf{y}}_{\text{batch}} \coloneqq \mathbf{K}_\gamma(\mathbf{p}_{\text{batch}})$ ▶ Predict KPIs for the current batch
    $\mathbf{L}_{\text{batch,KPI}} \coloneqq \frac{1}{\mathbf{b}_{\text{size}}} \sum_{j=1}^{\mathbf{b}_{\text{size}}} \|\mathbf{y}_{\text{batch}}^{(j)} - \hat{\mathbf{y}}_{\text{batch}}^{(j)}\|^2$ ▶ MSE loss for all meta models
    $\nabla \mathbf{K}_\gamma \coloneqq \frac{\partial \mathbf{L}_{\text{batch,KPI}}}{\partial \gamma}$ ▶ Compute gradients using backpropagation algorithm [119]
    γ : Update training parameters using gradients $\nabla \mathbf{K}_\gamma$ with Adam [104]
  end for
  Update $\mathbf{v}_{\text{p, counter}}$
  if $\mathbf{v}_{\text{p, counter}} \geq \mathbf{v}_{\text{p, limit}}$ **then**
    Return γ: Network training completed
  end if
  Continue training
end for
Return γ: Network training completed

multiple-output regression problem for both datasets. The loss function is decided during hyperparameter tuning along with other training hyperparameter values such as learning rate, batch size, optimizer, and activation functions that are detailed in Table 4.2. The other hyperparameters which are not mentioned in Table 4.2 and are decided by experience include total number of training epochs (100) and early stopping $(5$ epochs$)$. Early stopping is employed to prevent overfitting and improve the network's generalization performance. If the network's validation error does not decrease continuously for five epochs, training is stopped at that point. Another popular regularization approach, dropout [204], randomly disconnects a fraction of neurons during each training iteration to prevent overfitting. Dropout rates ranging from 0.2 to 0.4 were manually tested, but no performance improvement was observed. Therefore, the choice was

made to use only the early stopping method for these datasets.

The final training was executed on an NVIDIA Quadro M4000 GPU. The training pipeline is implemented using the DL framework TensorFlow2 [1] throughout this thesis. The training time per epoch and the number of trainable parameters for each model are detailed in Table 4.4 and Table 4.5. Generally, the

Table 4.3: Computational details for training on Datasets 1 and 2

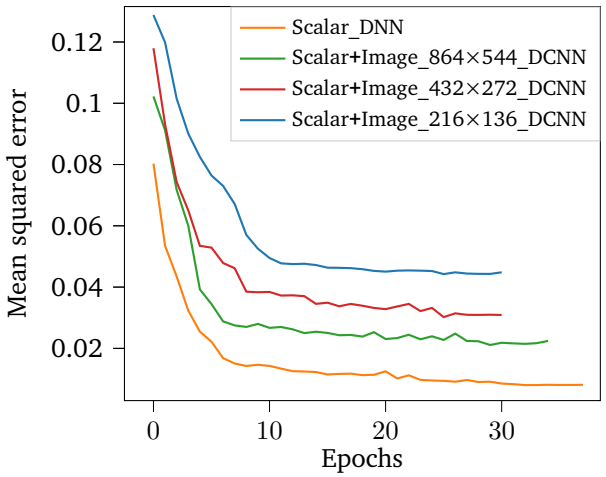
Table 4.4: Dataset 1

Model	**Time per epoch**	**Trainable parameters (million)**
DNN: scalar	$\sim$ 7 seconds	0.29
DCNN:scalar + image 216×136	$\sim$ 1 min	0.43
DCNN:scalar + image 432×272	$\sim$ 4.15 min	0.96
DCNN:scalar + image 864×544	$\sim$ 13 min	3.04

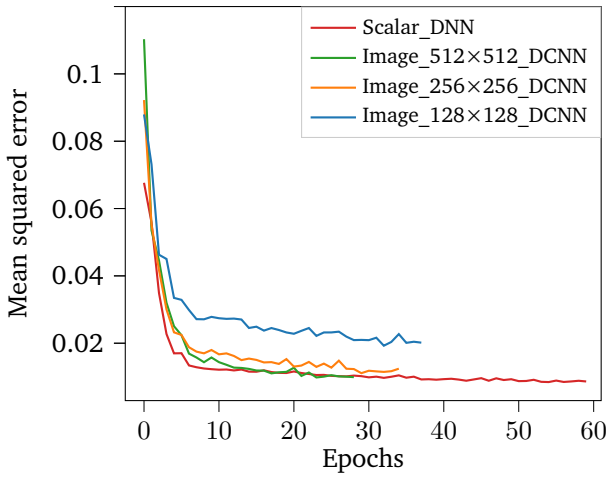
Table 4.5: Dataset 2

Model	**Time per epoch**	**Trainable parameters (million)**
DNN: scalar	$\sim$ 1 s	0.27
DCNN: image 128×128	$\sim$ 4 s	0.38
DCNN: image 256×256	$\sim$ 11 s	0.61
DCNN: image 512×512	$\sim$ 26 s	1.76

training time depends on various factors such as image resolution for image-based data, number of samples for training, batch size, availability of computational resources, number of trainable parameters etc. As to be expected, it is observed that higher-resolution images require more computational time and memory than lower resolution images during training. Scalar parameter-based DNNs have significantly lower training time compared to image-based meta-models. The meta-model training is more memory-bound than compute-bound, as processing high-resolution images and a large number of samples requires more memory. Parallelizing many samples over available computational resources is feasible if the memory requirements are met. This was also reported in [96]. The benefit of parallelization over a GPU during training can be leveraged if memory requirements are met, allowing for faster training of meta-models for the dataset 1. Validation curves during training for all the meta-models are displayed in Figure 4.12. It can be seen that for both datasets, the scalar-based models require a higher number of epochs for convergence but take much less time for the entire training than the image-based models, as shown in Table 4.4 and Table 4.5. At convergence, scalar-based models exhibit a lower validation loss, while image-based models with lower resolution display a higher validation loss. This trend is observed in both datasets. After training, all meta-models make predictions at a speed of around 1 ms/sample for new PMSM designs. The evaluation time for meta-models is very low compared to the conventional FE-based workflow, which takes roughly 3 hours/sample to 5 hours/sample on a single-core CPU. The content of the next section is based on [152].



(a) Dataset 1: validation curves during training.



(b) Dataset 2: validation curves during training.

Figure 4.12: Validation curves during training. Figure taken from [152, Fig. 7 and Fig. 8].

4.5 Numerical analysis

Equation 4.5 characterizes a non-linear multi-target regression problem [26]. In this thesis, various continuous target KPIs with different scales and units are addressed. Consequently, the following statistical measures are primarily employed to quantify the performance of the trained meta-model:

- **Mean relative error (MRE):** MRE normalizes the error for various target KPIs, making it easier to interpret and compare. The MRE provides a measure of the prediction accuracy, which is useful for understanding the model's overall performance [74]. It can be expressed as a percentage:

$$\overline{\varepsilon}_{\mathrm{mre}}(y_j, \hat{y}_j) = \frac{1}{n_\mathrm{s}} \sum_{l=1}^{n_\mathrm{s}} \frac{|y_j^{(l)} - \hat{y}_j^{(l)}|}{|y_j^{(l)}|} \times 100 \tag{4.6}$$

  where n_s is the total number of samples, y_j and $\hat{y}_j$ are true and predicted values of KPIs for the l-th sample from the given dataset with input parameters $\mathbf{p}^{(l)}$.

- **MAE:** MAE is a commonly employed regression metric that measures the average absolute difference between the predicted KPIs and the actual KPIs. It shows less sensitivity to outliers compared to other metrics like root mean squared error (RMSE), making it suitable when the dataset contains such extreme samples [35, 224]. MAE gives an intuitive understanding of the average prediction error in the original units of the target KPIs. It is mathematically described as

$$\overline{\varepsilon}_{\mathrm{mae}}(y_j, \hat{y}_j) = \frac{1}{n_\mathrm{s}} \sum_{l=1}^{n_\mathrm{s}} \left| y_j^{(l)} - \hat{y}_j^{(l)} \right|. \tag{4.7}$$

- **RMSE:** RMSE is another common regression metric that calculates the square root of the mean of the squared differences between the actual KPIs and the predicted KPIs. Compared to MAE, it is more

sensitive to outliers due to its heavy penalization on larger errors. It is written as

$$\bar{\varepsilon}_{\text{rmse}}(y_j, \hat{y}_j) = \sqrt{\frac{1}{n_\text{s}} \sum_{l=1}^{n_\text{s}} \left(y_j^{(l)} - \hat{y}_j^{(l)} \right)^2}. \tag{4.8}$$

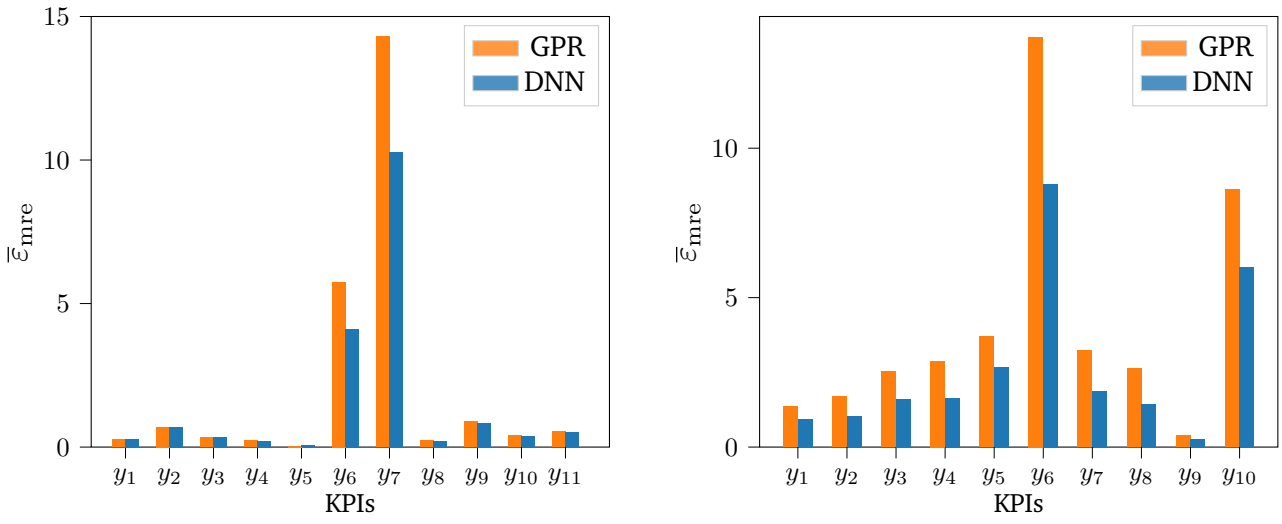
- **Pearson correlation coefficient (PCC):** PCC quantifies the linear relationship between predicted KPIs and true KPIs [109]. A PCC value close to one indicates a strong positive correlation, implying a better meta-model performance. It helps in assessing how well the meta-model captures the overall trend and direction of the relationship between the predictions and the true KPIs. It is mathematically described as

$$\varepsilon_{\text{pcc}}(y_j, \hat{y}_j) = \frac{\sum_{l=1}^{n_\text{s}} (y_j^{(l)} - \bar{y_j})(\hat{y}_j^{(l)} - \bar{\hat{y}}_j)}{\sqrt{\sum_{l=1}^{n_\text{s}} (y_j^{(l)} - \bar{y_j})^2} \sqrt{\sum_{l=1}^{n_\text{s}} (\hat{y}_j^{(l)} - \bar{\hat{y}}_j)^2}}, \tag{4.9}$$

where $\bar{y_j}$ represents the actual mean value and $\bar{\hat{y}}_j$ represents the predicted mean value for the jth KPI.

4.5.1 Gaussian process regression and DNN for parameter based meta-models

In the beginning of Chapter 3, it is mentioned that DL algorithms perform better compared to other state-of-the-art classical ML approaches in the case of big data and higher dimensional design space. To investigate this at a high-level with these datasets, a parameter-based untuned or base DNN model is compared with Kriging, also known as Gaussian process regression (GPR) [169]. The GPR based meta-model is trained using the scikit-learn library [32, 159] employing its default parameter settings, which include the RBF kernel (1.0) and the default optimizer L-BGFS-B [33, 141]. The RBF kernel is commonly employed in practical applications; however, a more comprehensive numerical comparison would ideally involve



(a) Dataset 1 **(b)** Dataset 2

Figure 4.13: KPIs prediction performance comparison for parameter-based meta-models.

exploring additional hyperparameters and alternative kernels. Nevertheless, investigating these factors is not the main focus of this thesis. The prediction performance of the GPR-based meta-model and the DNN-based meta-model are almost similar, except for torque-related KPIs, where the DNN-based meta-model performs better than the GPR-based meta-model for dataset 1 as described in Figure 4.13a. Due to memory limitations, the GPR-based meta-model was trained in two separate executions since dataset 1 is a much bigger dataset concerning input design space and samples as compared to dataset 2. The results regarding dataset 2 have been presented in Figure 4.13b. The results show that the GPR-based meta-model is outperformed by the DNN-based meta-model for all the KPIs. Additionally, the GPR-based meta-model requires approximately ten times longer training time than the DNN meta-model.

4.5.2 Evaluation of dataset 1

Two types of meta-models are trained: first, the scalar DNN depicted in Figure 4.9, exclusively for the geometry parameters that describe the geometry of both the stator and the rotor, and another type of meta-model, shown in Figure 4.11, trained on multiple inputs. These inputs include a half-pole rotor cross-section image and the scalar parameters describing the stator geometry. The training, validation, test sets, and training hyperparameter settings (e.g., learning rate, validation patience) are kept identical during the training of all the meta-models. Table 4.6 illustrates the numerical evaluation of all the KPIs with their mean values of percentage relative error and PCC on test samples. It can be observed that the

Table 4.6: Dataset 1: evaluation summary. Table taken from [152, Tab. 7].

	DNN		**DCNN** (544×864)		**DCNN** (272×432)		**DCNN** (136×216)	
	$\overline{\varepsilon}_{\mathrm{mre}}$	$\varepsilon_{\mathrm{pcc}}$	$\overline{\varepsilon}_{\mathrm{mre}}$	$\varepsilon_{\mathrm{pcc}}$	$\overline{\varepsilon}_{\mathrm{mre}}$	$\varepsilon_{\mathrm{pcc}}$	$\overline{\varepsilon}_{\mathrm{mre}}$	$\varepsilon_{\mathrm{pcc}}$
y_1	0.12	0.99	0.17	0.99	0.20	0.99	0.22	0.98
y_2	0.44	0.97	0.60	0.96	0.64	0.97	0.70	0.95
y_3	0.12	0.99	0.41	0.98	0.42	0.93	0.42	0.93
y_4	0.05	0.98	0.24	0.98	0.25	0.95	0.27	0.94
y_5	0.01	0.94	0.05	0.92	0.05	0.90	0.06	0.89
y_6	1.28	0.98	2.99	0.97	3.55	0.96	4.77	0.95
y_7	4.22	0.95	9.4	0.94	10.69	0.92	12.34	0.89
y_8	0.13	0.98	0.26	0.98	0.26	0.98	0.28	0.96
y_9	0.29	0.96	0.55	0.95	0.71	0.94	0.76	0.94
y_{10}	0.16	0.98	0.32	0.94	0.35	0.93	0.35	0.91
y_{11}	0.21	0.96	0.76	0.95	0.79	0.95	0.82	0.95

KPIs concerning to the torque behavior of the PMSMs, such as y_6 and y_7, demonstrate poorer prediction performance for the scalar DNN-base meta-model compared to other KPIs, with average $\overline{\varepsilon}_{\mathrm{mre}}$ values of 4.22% and 1.28%, respectively. The same holds true for multiple-input DCNN. The multiple-input DCNN is trained using three different image resolutions: 136×216 pixels, 432×272 pixels, and 864×544 pixels. The meta-model, trained on higher resolution image data (864×544 pixels), has an average $\overline{\varepsilon}_{\mathrm{mre}}$ of 1.43% across all the KPIs. This is approximately 12.05% and 24.96% lower than the input data with image resolutions of 432×272 pixels and 216×136 pixels, respectively. One observation is that the higher the resolution, the longer the training time for the meta-model, but with a gain in prediction performance. This fact can be explained by the training time of two multiple-input DCNNs, i.e., roughly 1 hour for the meta-model with 136×216 pixels versus roughly 2.5 hours for the meta-model 272×472 pixels. The average $\overline{\varepsilon}_{\mathrm{mre}}$ for the latter meta-model is 14.68% lower than the first meta-model across all the KPIs. It is also observed

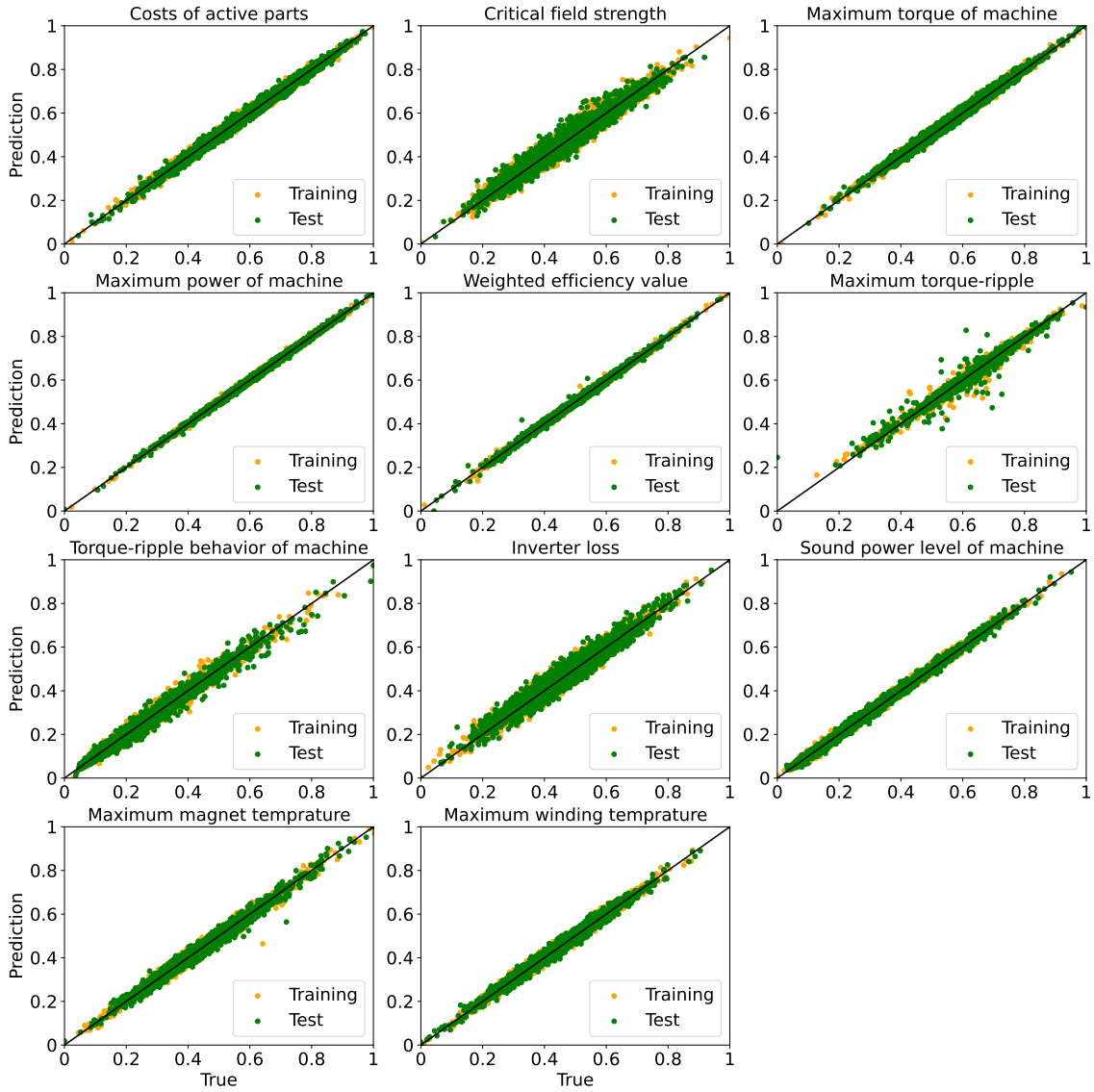


Figure 4.14: Dataset 1: prediction plots over test samples with scalar DNN based meta-model. Figure taken from [152, Fig. 12].

that the scalar DNN-based meta-model takes much lower training time than any image-based model with the average $\overline{\varepsilon}_{\text{mre}}$ of 0.64% that is much lower than the best-performing multiple-input DCNN model with a resolution of 544×864 pixels having an average $\overline{\varepsilon}_{\text{mre}}$ of 1.43%.

Normalized prediction plots for both the test and train samples, are shown in Figure 4.14. The horizontal axis represents the ground truth, while the vertical axis represents the predicted values. A cumulative plot for below 5% relative error for all the KPIs is shown in Figure 4.15 for all the meta-models. The cumulative plot displays the number of samples on the vertical axis and the relative error in $\%$ on the horizontal axis. By plotting the cumulative plot, it can be observed how the error accumulates or changes as the number of samples increases. A cutoff of 5% means that any relative error beyond this threshold is considered significant. By setting a cutoff value, the proportion of samples that fall within this acceptable error range can be analyzed. The distribution of errors can be visualized using this plot. By examining the

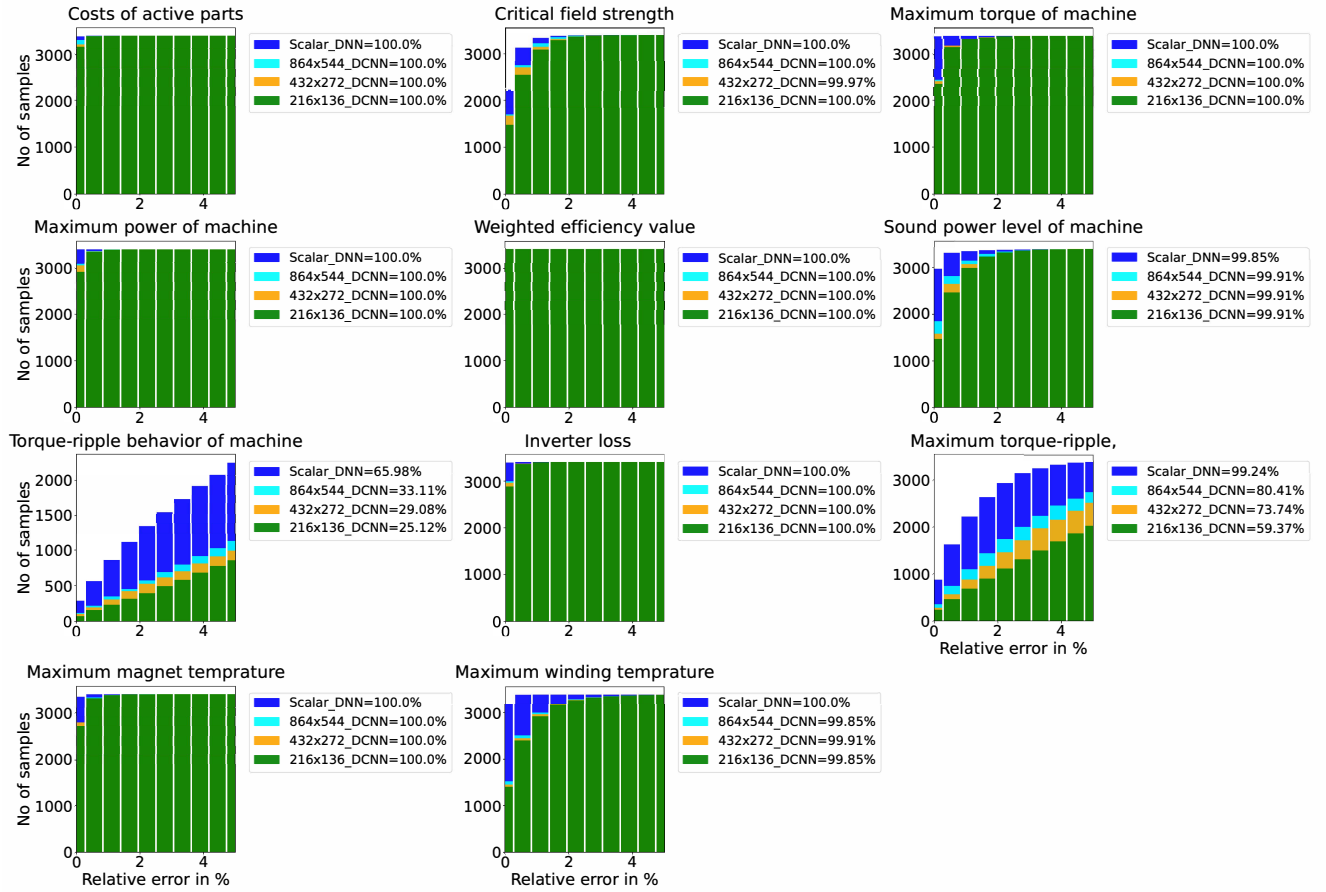


Figure 4.15: Dataset 1: cumulative accuracy plots of the KPIs prediction over test samples with $\overline{\varepsilon}_{\text{mre}} < 5\%$. Figure taken from [152, Fig. 13].

cumulative plot, the percentage of samples that fall within a certain error intervals can be determined. This information is valuable for evaluating the reliability of a meta-model. Additionally, this cumulative plot can help identify outliers or extreme errors that may require further investigation. Thus, the plotting of cumulative plots provides insights into the prediction accuracy of all meta-models. From the cumulative plots, it is evident that the torque-ripple behavior of the machine and the maximum torque-ripple KPIs have a higher number of samples with relative errors exceeding 5%. This may be due to a higher number of outlier samples. It can be seen that some of the training samples are predicted with high errors, as is illustrated in the training prediction plots in Figure 4.14.

4.5.3 Evaluation of dataset 2

Similar to the dataset 1, the dataset 2 is trained with two different forms of meta-models: one only using scalar information of rotor and stator, as shown in Figure 4.9, and another meta-model, as illustrated in Figure 4.10, which uses only pictorial information of one full pole rotor and stator cross-section of the PMSM. The dataset 2 consists of much fewer samples with a different set of KPIs than the dataset 1. The input design space is also more uniformly distributed with a lesser number of scalar parameters (12). Table 4.7 describes numerical assessment of all KPIs with their MRE in percentage and

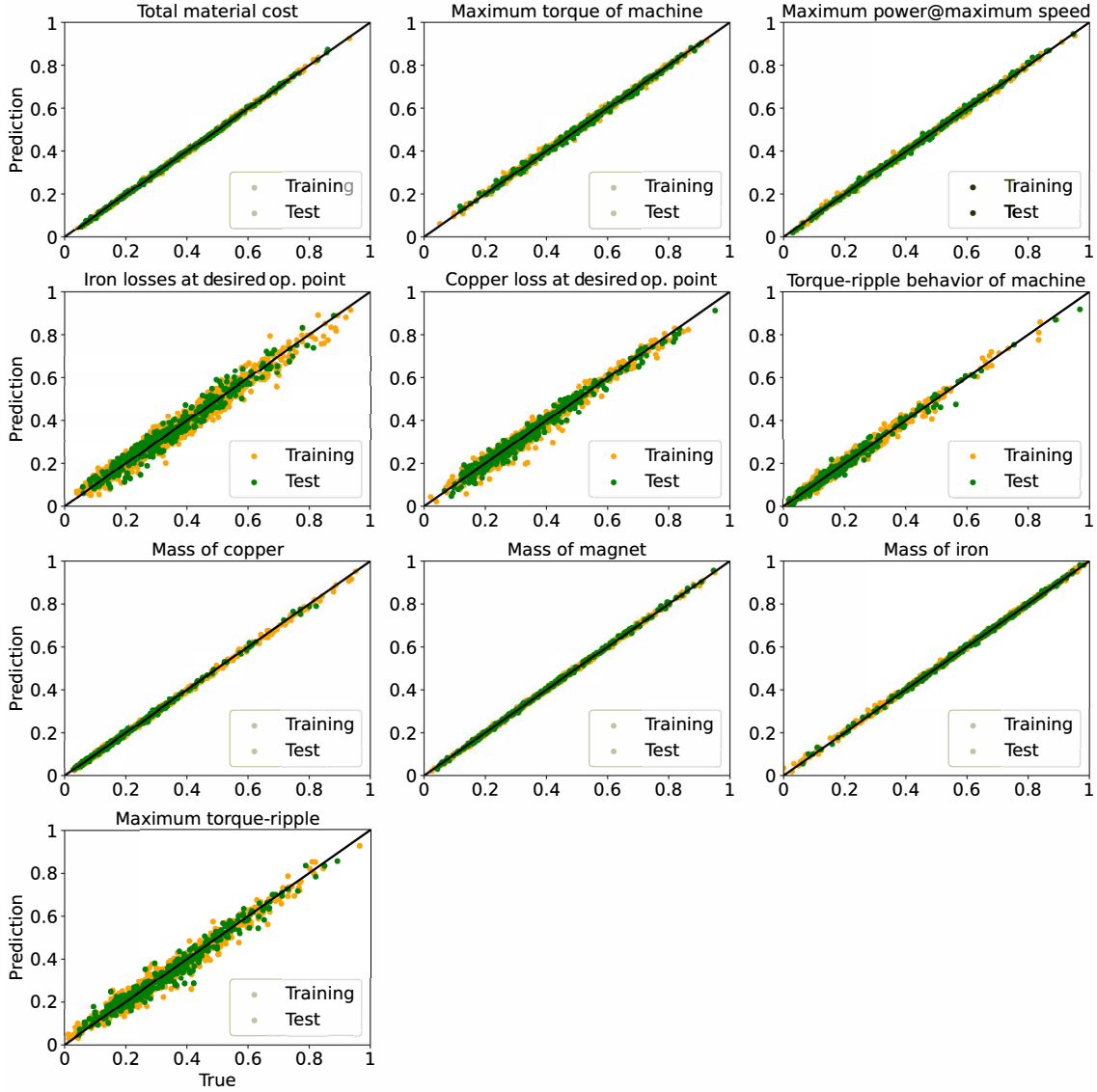


Figure 4.16: Dataset 2: prediction plots over test samples with scalar DNN based meta-model. Figure taken from [152, Fig. 14].

PCC over all test samples. It can be seen that both types of meta-models (scalar DNN and image-based DCNN) show lower prediction accuracy for torque concerning KPIs, i.e., y_6 and y_{10}, compared to other KPIs. Similar to dataset 1, all meta-models trained with dataset 2 also show that the prediction accuracy improves with image precision. The average MRE over all KPIs is 1.8% for the meta-model with 512×512 pixels, 1.97% for the meta-model with 256×256 pixels, and 2.52% for the meta-model with 128×128 pixels. The scalar DNN meta-model has an average MRE of 1.66% over all the KPIs, which is 7.45% lower than the best performing DCNN meta-model with 512×512 pixels. Analogous to dataset 1, the normalized prediction plots and the cumulative plots for the error distribution of all KPIs are displayed in Figure 4.16 and Figure 4.17, respectively. Test samples of both KPIs, torque-ripple behavior of the machine and maximum torque have many samples with a relative error greater than 5% compared to other KPIs. This is observed for both image-based and scalar-based meta-models. However, scalar-based models have fewer samples with high relative errors.

Table 4.7: Dataset 2: evaluation summary. Table taken from [152, Tab. 8].

	DNN		**DCNN** (512×512)		**DCNN** (256×256)		**DCNN** (128×128)	
	$\overline{\varepsilon}_{\text{mre}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$	ε_{pcc}
y_1	0.42	1.00	0.47	1.00	0.56	1.00	0.78	0.99
y_2	0.51	1.00	0.45	1.00	0.46	0.99	0.68	0.98
y_3	0.91	1.00	0.90	1.00	1.16	0.99	1.49	0.99
y_4	1.52	0.98	1.33	0.99	1.26	0.99	1.31	0.96
y_5	1.83	0.99	1.82	0.99	1.64	0.99	2.06	0.97
y_6	5.9	0.98	6.70	0.98	7.99	0.96	9.67	0.97
y_7	1.06	0.99	0.78	0.99	0.62	0.98	0.93	0.94
y_8	0.84	1.00	0.97	1.00	1.15	1.00	1.53	0.98
y_9	0.13	1.00	0.14	1.00	0.14	1.00	0.17	0.99
y_{10}	3.47	0.98	4.36	0.98	4.72	0.97	6.58	0.95

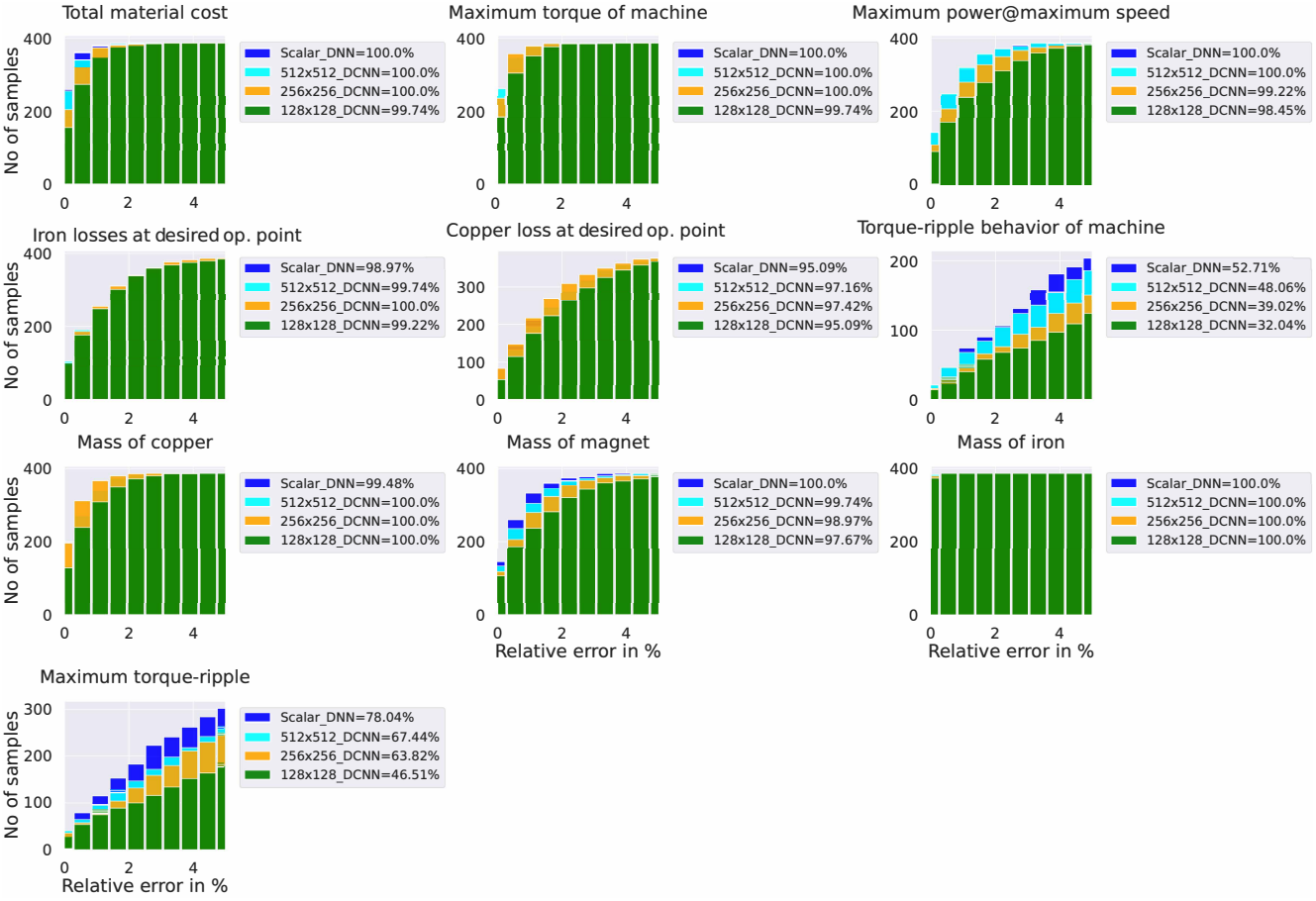


Figure 4.17: Dataset 2: cumulative accuracy plots of the KPIs prediction over test samples with $\overline{\varepsilon}_{\text{mre}} < 5\%$. Figure taken from [152, Fig. 15].

4.6 Summary

In this chapter, a data-driven approach was introduced using supervised learning for predicting a large number of cross-domain KPIs, employing different types of input data seen as geometry representations of PMSMs, namely, parameter- and image-based (2D cross-section) representations. A detailed numerical analysis is carried out on two distinct datasets of different sizes.

To facilitate a comprehensive numerical comparison between both input representations, three input-based meta-models were proposed: only scalar input (DNN), only image input (DCNN), and a combination of both (multiple-input DCNN). It was observed that the entirely scalar parameter-based meta-model considerably outperformed the image-based DCNN and the multiple-input DCNN meta-models in terms of prediction accuracy, while also offering significantly lower computational demands.

On the other hand, the image-based representation can be useful in the context of re-parametrization since the trained DCNN meta-model solely depended on the final image space, thus eliminating the need for new meta-model training, unlike the parameter-based meta-model.

The pixel resolution study for dataset 1 gave insights on the required pixel per mm to accurately describe variation in the parameter for different image sizes. The numerical results show that increasing pixel resolution enhances the prediction accuracy of image-based models, bringing it closer to that of the scalar-based model for some KPIs, but at the expense of increased computational effort.

Several crucial parameters, such as varying voltage, current, and axial machine length, cannot be incorporated with 2D image-based representation. These can be separately fed as scalar inputs to the DCNN. Nevertheless, in this investigation, stator geometry parameters were included in multiple-input DCNN (see Figure 4.11) for dataset 1, but the meta-model based solely on scalar parameters remained superior.

The proposed approach is limited to predicting only single-valued KPIs, which might not suffice for a detailed performance quantification of PMSMs. Additionally, the pure data-driven approach doesn't leverage any physical laws that might be helpful in predicting more accurately highly non-linear KPIs, e.g., torque ripple-related KPIs. These KPIs demonstrated the worst prediction performance across both datasets with all meta-models.

In the next chapter, these limitations will be addressed through the introduction of a hybrid data- and physics-driven approach.

5 Physics and data-driven hybrid model for optimization of electrical machines

Chapter 4 presented a data-driven DL approach for performance quantification of PMSMs concerning various input geometry representations of PMSMs, i.e., scalar parameter and image-based representations. The numerical results showed that the scalar parameter-based input representation has a better functional mapping ability with KPIs and significantly higher computational efficiency compared to the image-based representation. Additionally, it is important to note that the scalar parameter-based representation easily incorporates crucial parameters such as machine length, excitation current, and voltage. These parameters cannot be represented in a 2D cross-section image of PMSMs. Therefore, from this point onward, the scalar parameter-based input representation will be employed.

In this chapter, a hybrid approach combining data- and physics-driven models is presented to enable a more detailed performance analysis of PMSMs. First, a generalized hybrid approach is introduced by discussing a few limitations of the data-driven approach. Subsequently, the procedure related to the hybrid approach is described, and the details about the dataset used are given. Next, network architecture and training specifications are provided. Numerical results and quantitative analysis are then presented and compared to the data-driven approach. In the end, the application of the hybrid approach for the MOO of PMSMs in industrial settings is demonstrated.

The majority of the content and structure of this chapter follows our work presented in [153, 155].

5.1 Introduction of a generalized hybrid approach

Merging physics and data-driven approaches has deep historical roots. As explained in [117], models based on first principles might be termed the Newtonian paradigm [51, p. 56], stemming from Newton's unifying laws of motion. In contrast, the Keplerian paradigm [51, p. 56], named after Johannes Kepler, focuses on fitting mathematical descriptions to observed data. These two methodologies, although distinct, can complement each other; for example, Kepler's laws can be derived from Newtonian principles, and models based upon Newton's laws can be adjusted with data to better represent real-world celestial movements. The authors formally define *hybrid modeling* as a method to produce an improved model, which is more explainable and reliable by integrating data-based models with first principle-based models. To illustrate the concept of hybrid modeling, the paper [117] provides three application examples: characterizing superconducting magnets by merging data with physics, data-driven magnetostatic field simulations, and Bayesian optimization for circuit board design.

In this thesis, the simulation workflow for calculating KPIs for the PMSM is presented in Sec. 2.2.3. The magneto-static FE simulation (refer to Sec. 2.2.2) lies at the core of this workflow. However, performing large-scale FE simulations requires significant computational resources, which becomes a major bottleneck

in the numerical optimization of electrical machines. This limitation restricts the exploration of a large design space during the MOO. The proposed data-driven DL approach in Chapter 4 has several limitations as follows:

- The DL model completely relies on the final KPIs it is trained on using supervised learning. The calculation of final KPIs implicitly incorporates the impact of system parameters (refer to Sec. 2.2.3). As shown in Figure 5.1, the prediction ability of the trained DL-based meta-model is confined by fixed settings of system parameters. For example, if a system parameter, such as the inverter input current is changed, then the calculation of KPIs like maximum shaft power and copper losses (2.38) also alters. A PMSM with the same design parameters produces different shaft power and copper losses at a higher inverter input current than at a lower one, since the calculation of these KPIs implicitly depends on the inverter input current. In this scenario, a meta-model trained with fixed settings for a higher input current will yield inaccurate predictions when the input current varies.
- Thus far, the proposed DL models are limited to predicting single-valued KPIs. However, for a detailed analysis of PMSM design, it is necessary to compute more challenging KPIs such as efficiency maps and various performance curves (see Figure 2.8). Separate DL models can be trained to compute these KPIs; for example, various DL models, such as feed-forward ANN, image-based CNN, and geometry parameters based-RNN, are individually and sequentially trained to predict efficiency maps [94]. Thus, the data-driven meta-model-based simulation workflow remains less flexible and demands more computational resources for individual training of each DL model to include these complex performance measures.
- An entirely data-driven model also lacks physics-based interpretation when making predictions. The reason is that when evaluating the resulting DL meta-model, it does not make use of the physics (encoded by equations) explicitly. For example, currents may only approximately sum up to zero at a circuit node because the corresponding Kirchhoff law is not explicitly enforced and it was not "rediscovered" by the computer. This limitation can impact prediction accuracy, particularly for highly non-linear KPIs. For instance, torque-ripple related KPIs have exhibited poorer accuracy than other KPIs as observed in Sec. 4.5.

The physics-based post-processing, the final step in the simulation workflow for calculating KPIs (see Figure 2.5), only takes 3-5 minutes per design on a single-core CPU. This is significantly faster compared to the FE simulation, which typically requires around 3-5 hours per design. This motivates us to propose a hybrid approach that combines the data-driven DL part with subsequent the utilization of physics-based post-processing. The aim is to train the scalar parameter-based DL model to approximate expensive magneto-static FE calculations instead of training it directly on the final KPIs, as in the data-driven DL approach. The results of the magneto-static FE simulations will be referred as ***intermediate measures*** for the rest of the thesis. These intermediate measures characterize the electromagnetic behavior of the PMSM. They mainly include electromagnetic torque ($\mathcal{T}$) and flux linkages (ψ) over one electrical period, and integrated iron losses ($\mathcal{V}_{\mathrm{fe}}$) over the rotor and the stator regions. The content of the following section is based on [153].

5.2 Procedure and dataset details

Figure 5.1 describes block diagrams of different methods discussed so far for calculating KPIs for the PMSM. In the conventional approach, each magneto-static FE simulation utilizes an $l-$dimensional input vector

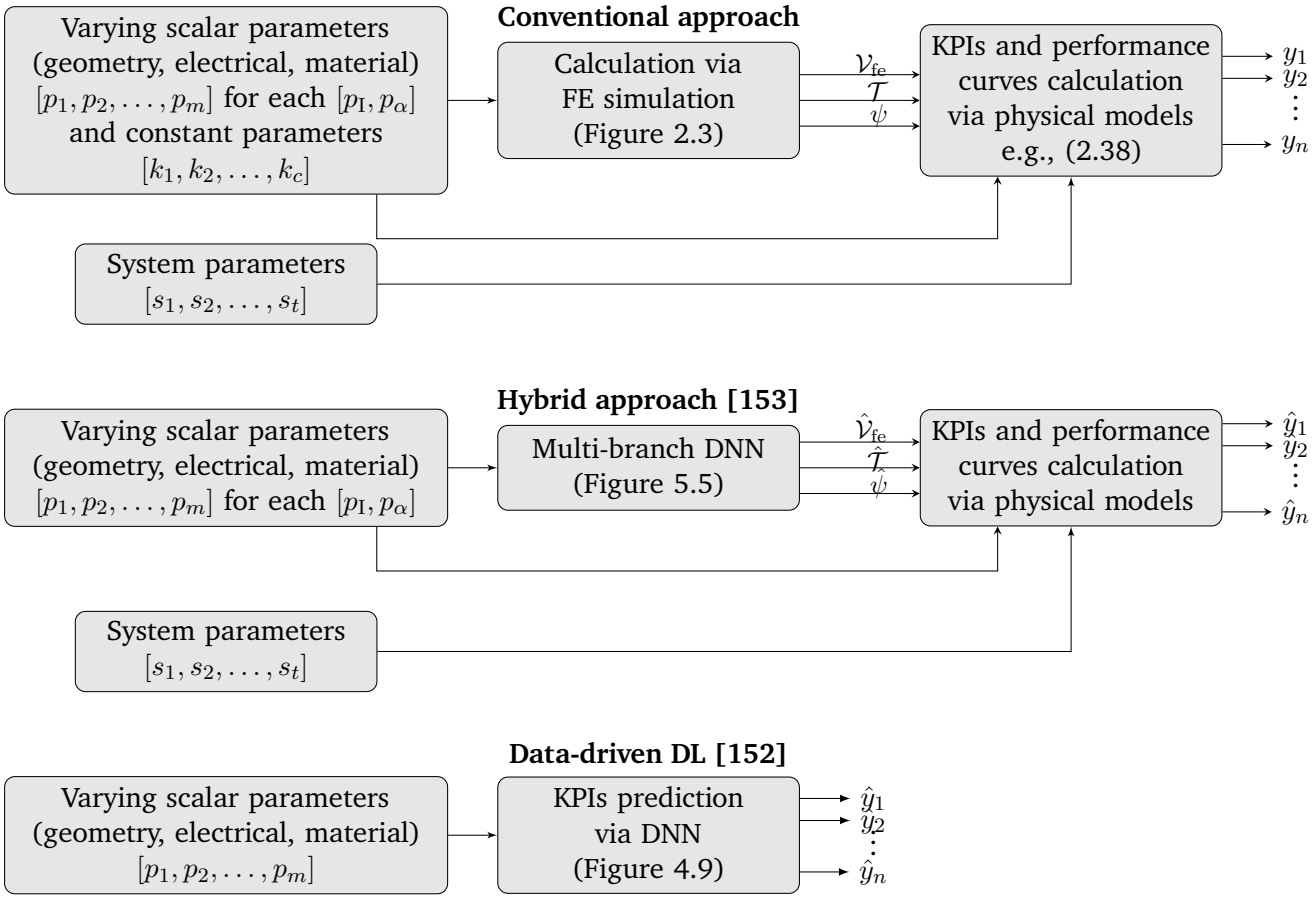


Figure 5.1: Schematic representations of various methods for computing KPIs. Figure based on [153, Fig. 5], © 2022 IEEE.

$\mathbf{x}$ to obtain intermediate measures $\mathbf{z}$, which are subsequently processed to compute the vector of target KPIs ($\mathbf{y}$). Suppose a PMSM simulation dataset is given with

$$\mathcal{D} := \left\{ (\mathbf{x}^{(1)}, \mathbf{z}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(N_\mathrm{total})}, \mathbf{z}^{(N_\mathrm{total})}, \mathbf{y}^{(N_\mathrm{total})}) \right\} \tag{5.1}$$

of N_total samples. The input vector $\mathbf{x}$ for each design in the dataset $\mathcal{D}$ is composed of design parameters $\mathbf{p}$, system parameters $\mathbf{s}$, and constants $\mathbf{k}$. Here, $\mathbf{p}$ is a vector $[p_1, p_2, \ldots, p_m]$ with m design parameters, $\mathbf{s}$ is a vector $[s_1, s_2, \ldots, s_t]$ with t system parameters, and $\mathbf{k}$ is a vector $[k_1, k_2, \ldots, k_c]$ with c constant parameters. A few examples of all these parameters can be found in Table 8.7. As depicted in the first diagram of Figure 5.1, the magneto-static FE model $\mathbf{M}$ calculates the $n-$dimensional vector of intermediate measures $\mathbf{z} = \mathbf{M}(\mathbf{p}, \mathbf{k})$ for each design in the dataset $\mathcal{D}$. In the subsequent step, the intermediate measures $\mathbf{z}$, along with concerning system parameters $\mathbf{k}$ and input parameter $\mathbf{p}$, are processed using physics-based formulas to calculate the target KPIs $\mathbf{y}$. The conventional calculation of the KPIs $\mathbf{y}$ can be expressed mathematically in an abstract manner by

$$\mathbf{y} = \mathbf{K}(\mathbf{p}, \mathbf{M}(\mathbf{p}, \mathbf{k}), \mathbf{s}). \tag{5.2}$$

It should be noted that the system parameters $\mathbf{s}$ are involved solely during the post-processing stage.

In Chapter 4, the data-driven approach is explained, focusing on learning the functional mapping $\mathbf{p} \rightarrow \mathbf{y}$, i.e.

$$\mathbf{y} \approx \hat{\mathbf{K}}_\theta(\mathbf{p}) \tag{5.3}$$

where θ represent the training parameters (weights and biases). The approximation $\hat{\mathbf{K}}_\theta$ serves as a DNN meta-model for the prediction of the KPIs. Please note that here, $\mathbf{p}$ denotes varying scalar parameters used to create the dataset. The training parameters are optimized by minimizing the training loss function, e.g., MAE. Mathematically, this can be described as,

$$\min_\theta L(\theta) := \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}^{(j)} - \hat{\mathbf{K}}_\theta(\mathbf{p}^{(j)})\|_1 \tag{5.4}$$

where N represents the number of training samples. The term $\mathbf{y}^{(j)}$ denotes true KPIs concerning j^{th} sample, while $\hat{\mathbf{K}}_\theta(\mathbf{p}^{(j)})$ represents the predicted KPIs using the DNN meta-model. The expression $\|\cdot\|_1$ denotes the ℓ_1-norm is used to measure MAE between the true values and the predicted values.

In the proposed hybrid approach, the goal is to train the meta-model solely to approximate the computationally expensive function $\mathbf{M} : \mathbf{p} \rightarrow \mathbf{z}$ for the FE calculation. This can be abstractly written as

$$\mathbf{y} \approx \mathbf{K}(\mathbf{p}, \hat{\mathbf{M}}_\varphi(\mathbf{p}), \mathbf{s}), \tag{5.5}$$

where φ represents the DNN training parameters. Similar to the data-driven approach, the training loss function (e.g., MAE) can be written as

$$\min_\varphi L(\varphi) := \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{z}^{(j)} - \hat{\mathbf{M}}_\varphi(\mathbf{p}^{(j)})\|_1, \tag{5.6}$$

where $\mathbf{z}^{(j)}$ are actual reults from FE model $\mathbf{M}(\cdot)$ and $\hat{\mathbf{M}}_\varphi(\mathbf{p}^{(j)})$ represent predicted results.

Once the DNN ($\hat{\mathbf{M}}_\varphi(\cdot)$) is trained, it will serve as a meta-model to predict intermediate measures for new PMSM designs.

5.2.1 Dataset details

The dataset is created for the double-V topology of the PMSM, as can be seen in Figure 5.2, which displays a representative geometry from the dataset. A total of $N_\text{p} = 35$ varying input design parameters (p_i) within their specified lower and upper bounds are considered. The complete list is given in Table 8.11. The dataset generation procedure follows the same approach explained in Figure 4.3, with the only difference being that intermediate measures are being stored for each PMSM design along with KPIs. After filtering out erroneous designs from the initial population, $N_\text{PMSM} = 44877$ valid PMSM designs were obtained. Next, the 2D magneto-static FE simulation is performed for all these PMSM designs at $N_\text{OP} = 37$ operating points (determined by experience). The operating points of the electrical machine are treated as variable electrical excitation inputs, defined by an input phase current I and its associated control angle α. Here, the control angle (α) represents the electrical phase angle between the input phase current (I) and the no-load induced voltage (back-EMF). The magneto-static FE simulation for each operating point is run for fifteen time steps that cover the electrical 60° interval. This choice aligns with the setup of 1 time step = electrical 4° for this

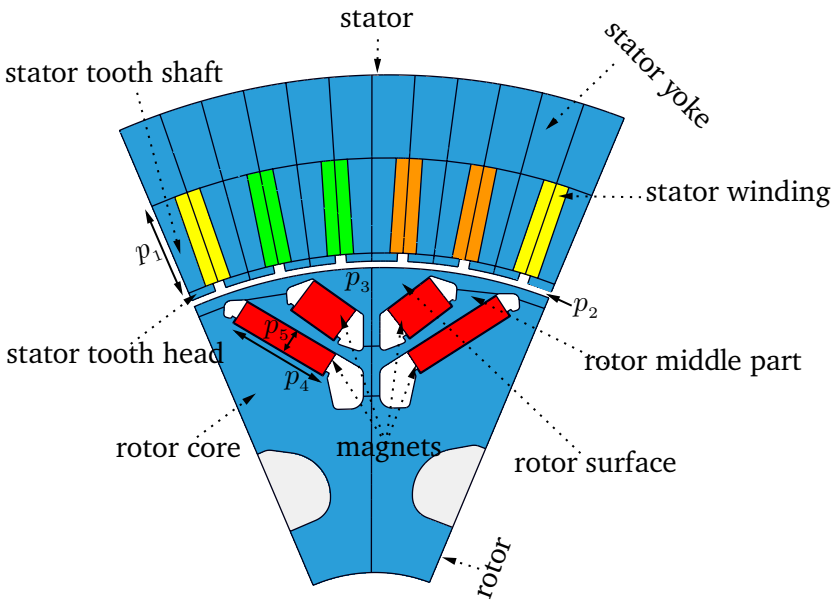


Figure 5.2: Exemplary double-V PMSM geometry. Figure taken from [153, Fig. 1], © 2022 IEEE.

thesis. If necessary, this configuration can be adjusted. To compute the complete 360° electrical period, the magnetic state symmetry is leveraged. Each time step is treated as an independent output quantity, and they are not dependent on one another. It should noted that the time steps are determined at a constant speed of $(n_{\text{rpm\_base}})$ 10000 rpm. Figure 5.3 depicts torque and flux calculations for one operating point during an electrical period for a PMSM sample from dataset $\mathcal{D}$. FE simulations are performed on a high-performance computing (HPC) cluster, where each PMSM design takes approximately three to six hours per simulation on a single-core CPU, depending on the availability of computing resources on the cluster. Ultimately, the dataset $\mathcal{D}$ consists of $N_{\mathcal{D}} = N_{\text{OP}} \times N_{\text{PMSM}} = 1660449$ samples. The intermediate measures ($\mathbf{z}$) are listed in Table 8.8. As listed in Table 8.8, they include the electromagnetic torque $\mathcal{T}$, the non-linear iron losses $\mathcal{V}_{\text{fe}}$, and the fluxes ψ_1, ψ_2, ψ_3 associated with three coils for one electrical period. The iron losses $\mathcal{V}_{\text{fe}}$ are computed within the FE solver employing Steinmetz's models [171, 205]. During the post-processing phase, the calculated losses can be scaled for different speeds for performance analysis. All the intermediate measures are post-processed using different physics-based models to obtain the KPIs; refer to Sec. 2.2.3 for more details. Table 8.9 lists KPIs that are considered for investigating the hybrid approach. The pairwise distribution of five KPIs and varying design parameters is displayed in Figure 5.4. All five parameters appear to follow mostly homogeneous distributions, whereas the target KPIs are inhomogeneous. From the off-diagonal plots, some correlation can be noticed between KPIs y_1, y_2, and y_3. The content of the following section is based on [153].

5.3 Network structure and training specifications

As mentioned in Sec.4.4, in addition to input representation, the choice of hyperparameters significantly impacts the prediction accuracy of any DL-based meta-model. The same strategy explained in Sec.4.4 for hyperparameter tuning is employed. Initially, a base network comprising five dense layers is defined. Then, after evaluating approximately twenty network structures through trial and error, a multi-branch structure was determined, which outperformed single-branch structures. The manually designed base multi-branch

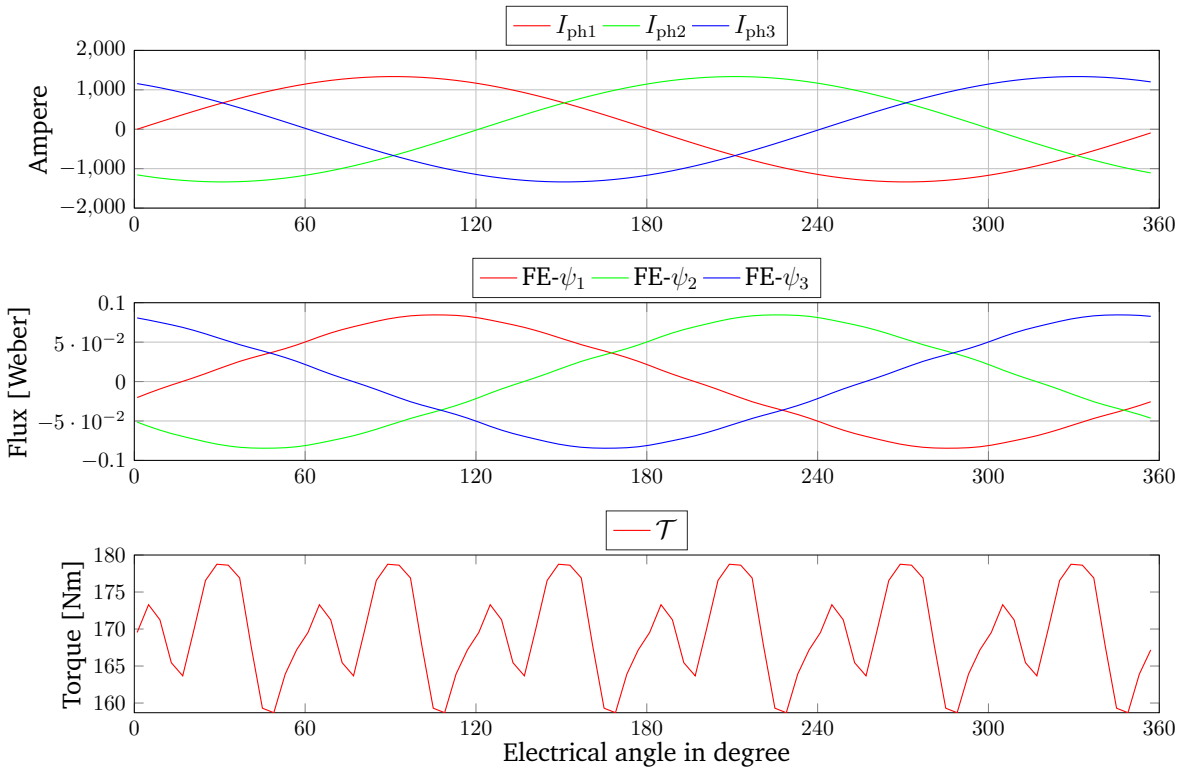
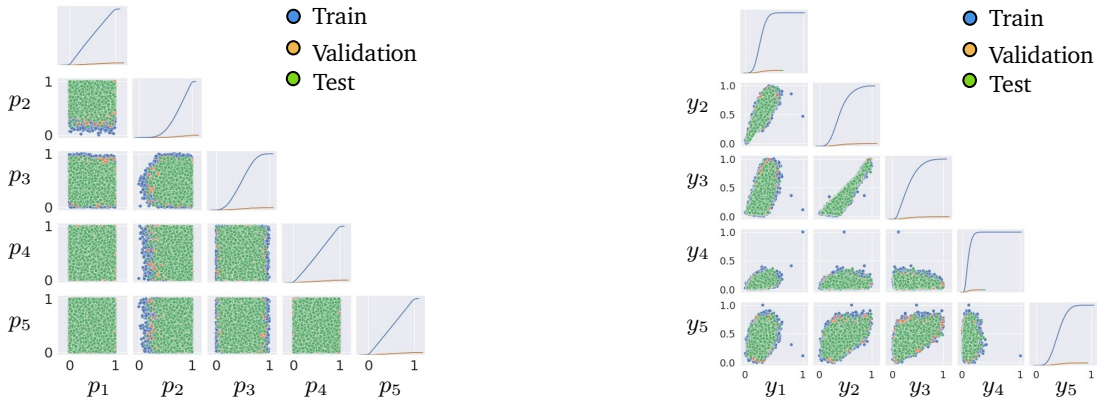


Figure 5.3: Flux and torque illustration for an operating point at maximal current I and $\alpha = 0$ over one electrical period. Figure based on [153, Fig. 3], © 2022 IEEE.



(a) Design parameters.

(b) KPIs.

Figure 5.4: Parameter and KPIs distribution. Figure based on [153, Fig. 4], © 2022 IEEE.

DNN network is constituted of three dense layers for each flux quantity, five layers for electromagnetic torque and non-linear iron losses, and two shared layers. Shared layers in a multi-branch DNN allow the model to learn common representations from the input data, leveraging shared information before branching out to distinct outputs. This architecture reduces the number of trainable parameters, thus reducing the computational burden while improving generalization for the prediction of intermediate measures. Next, the HPO is performed using a random approach with five-fold cross-test and validation

(see Figure 4.8). Around 800 different configurations within the search space specified in Table 5.1 are evaluated, using an in-house optimization tool. The entire HPO process took nearly two days on an in-house GPU cluster. Finally, the multi-branch DNN illustrated in Figure 5.5 is obtained. The size of

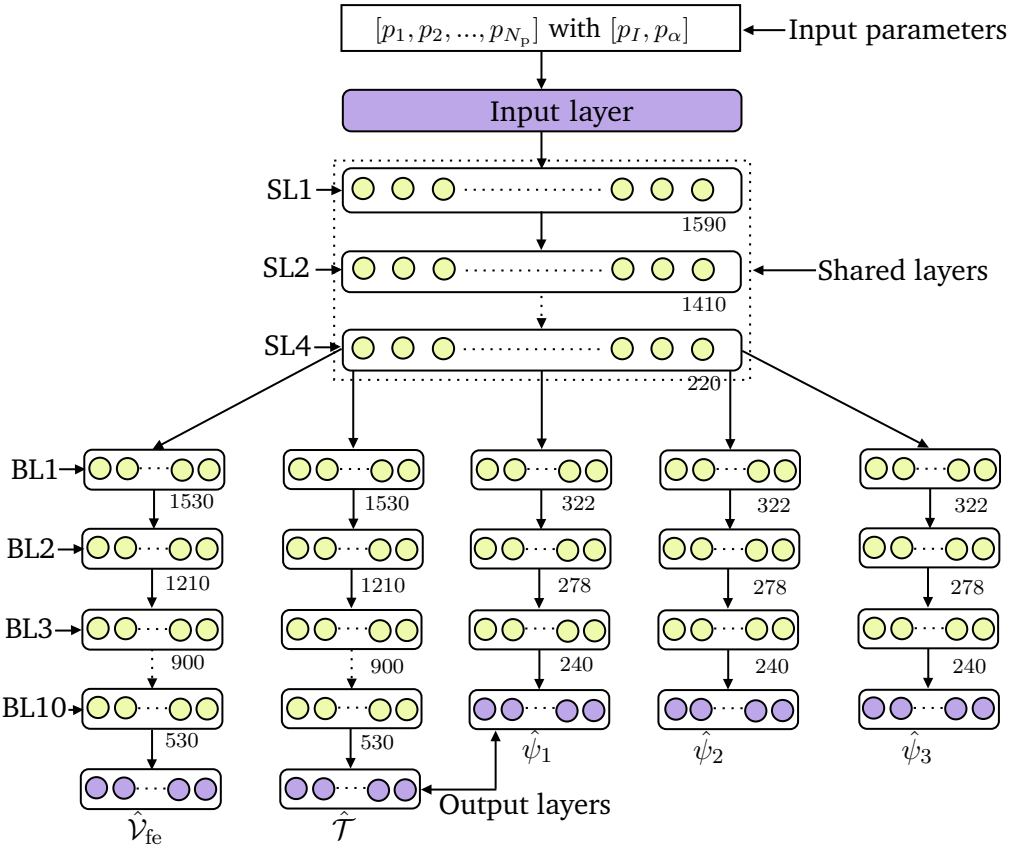


Figure 5.5: Proposed multi-branch network structure. Figure taken from [153, Fig. 6], © 2022 IEEE.

the input layer corresponds to the number of varying input design parameters. The network structure includes a total of five individual branches, each with a different size of branch layers (BL). Two identical branches are dedicated to torque ($\hat{\mathcal{T}}$) and iron loss ($\hat{\mathcal{V}}_{\text{fe}}$) prediction, while remaining three same sized branches do flux ($\hat{\psi}$) prediction. The size of dense layers for the branch of iron loss and torque is as follows: $1530 \rightarrow 1210 \rightarrow 900 \rightarrow 880 \rightarrow 750 \rightarrow 660 \rightarrow 610 \rightarrow 580 \rightarrow 550 \rightarrow 530$. For the flux branches, the layer configuration is $322 \rightarrow 278 \rightarrow 240$. To leverage the correlation among all output intermediate measures, there are four shared layers (SL: $1590 \rightarrow 1410 \rightarrow 810 \rightarrow 220$). As mentioned in the Sec. 5.2.1, the magneto-static FE simulation is run for fifteen time steps for each operating point. Each output neuron represents one time step value in the flux and torque branches. Therefore, the size of the output layers for the torque and flux branches are 15×1. The values of iron losses integrated over both the stator and rotor regions are predicted. Each neuron in the output layer of the iron loss branch predicts a single integrated loss value, resulting in a layer size of 4×1.

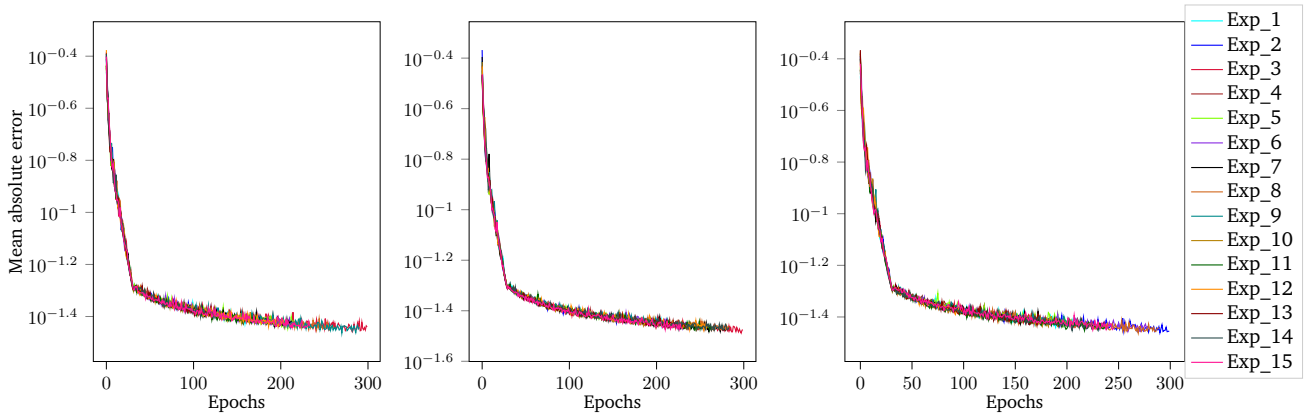
The hybrid approach is compared with the data-driven approach proposed in Chapter 4. The same tuned network structure as shown in Figure 4.9 is utilized. It should be noted that, from a network structure perspective, the two networks are different since they are trained on different input-output data, while the training hyperparameters, including early stopping criteria, training loss function, learning rate, and optimizer, kept identical for both approaches. The content of the next section is based on [153].

Table 5.1: Details of Hyperparameters. Table based on [153, Tab. 4], © 2022 IEEE.

Hyperparameter	Min	Max	Final values
Adaptive learning rate (with 10000 decay steps)	10^{-6}	10^{-3}	10^{-4} to 10^{-5}
Average number of neurons per hidden layer	50	1000	991
Number of common layers	1	6	4
Number of branch layers	3	12	10
Batchsize	80	150	132
Activation functions	relu, elu, tanh, softplus		elu
Optimizers	Adam, adamax, AdaGrad, nadam [47]		Adam
Loss functions	Mean absolute error, mean squared error, huber loss		Mean absolute error

5.4 Numerical analysis

In this section, first, the results of intermediate measures are discussed. Afterward, a quantitative analysis showcasing an empirical comparison of both the hybrid and data-driven DL approaches is presented. The approximation of intermediate measures is a non-linear multiple-output regression problem. Similar to Chapter 4, the prediction accuracy of meta-models is evaluated using multiple evaluation metrics, namely MRE, MAE, and PCC.

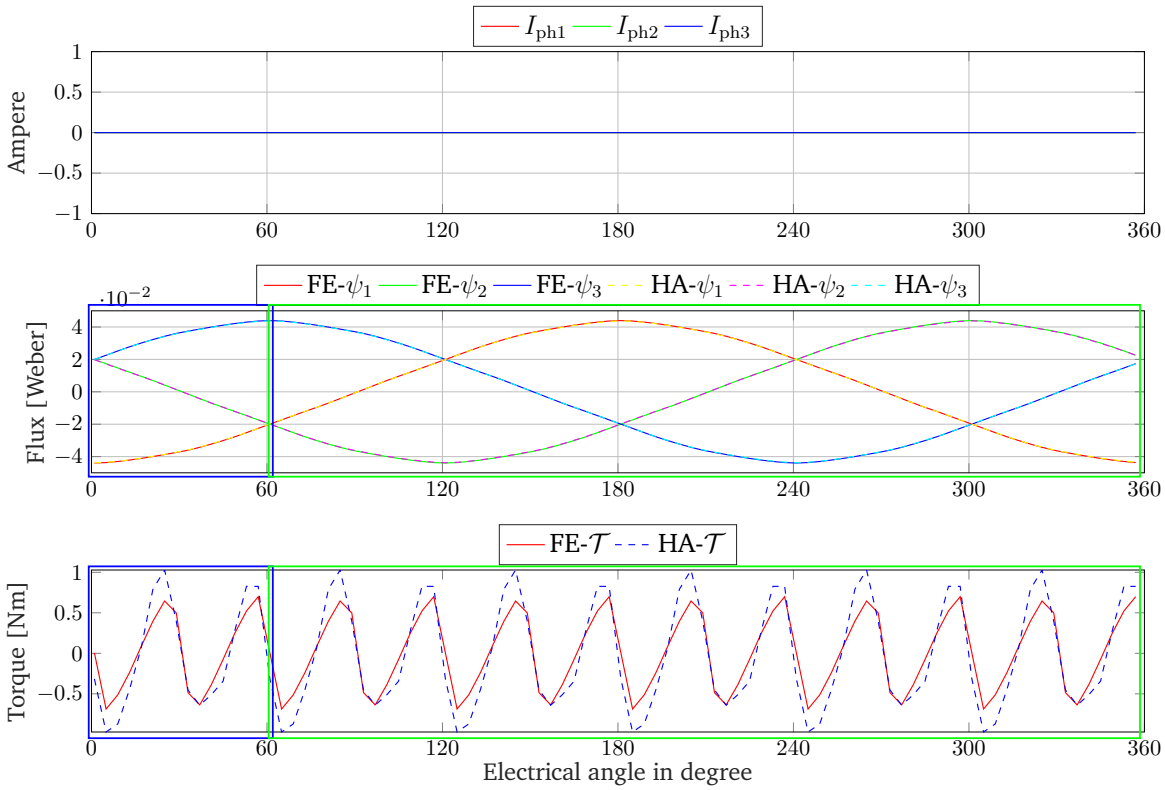


(a) [70-20-10] (%) **(b)** [75-10-15] (%) **(c)** [70-10-20] (%)

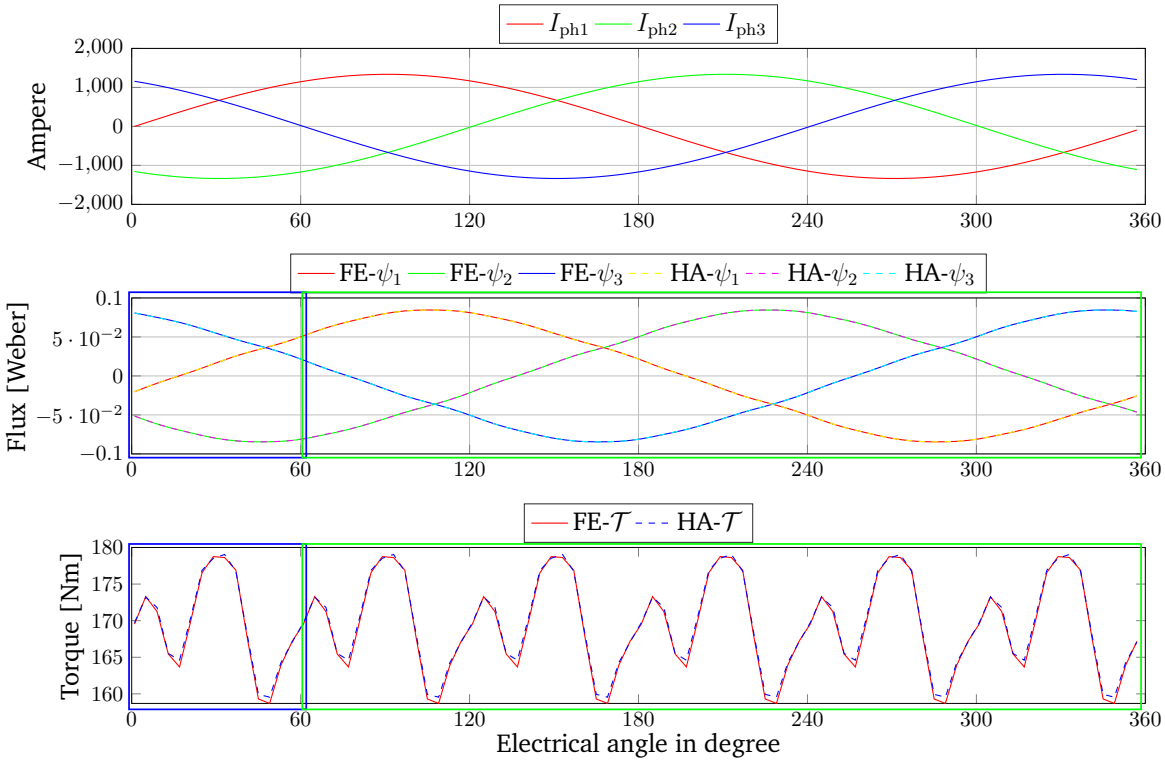
Figure 5.6: Validation curves during training for different train-validation-test split percentages

5.4.1 Analysis on the intermediate measures prediction

After determining the final configuration of the multi-branch DNN, the prediction performance was examined using different training-validation-test split percentages (70-20-10, 75-10-15, and 70-10-20) in fifteen additional experiments, ensuring complete randomization in the dataset. Figure 5.6 shows the validation curves during training for all the experiments. Table 5.2 presents numerical accuracy with mean performance



(a) Operating point: zero current I and $\alpha = 0°$



(b) Operating point: maximal current I and $\alpha = 0°$

Figure 5.7: Plot of flux and torque predictions over single electrical cycle at various operating points. Figure based on [153, Fig. 10], © 2022 IEEE.

Table 5.2: Intermediate measures: mean performance of optimized multi-branch DNN over test samples across the fifteen experiments for different **train-validation-test** split percentages

Measure	**[70-20-10]** (%)		**[75-10-15]** (%)		**[70-10-20]** (%)	
	$\overline{\varepsilon}_{\text{MRE}}$	ε_{PCC}	$\overline{\varepsilon}_{\text{MRE}}$	ε_{PCC}	$\overline{\varepsilon}_{\text{MRE}}$	ε_{PCC}
eddy current loss (rotor)	$1.71 \cdot 10^{0}$	0.98	$1.58 \cdot 10^{0}$	0.98	$1.64 \cdot 10^{0}$	0.98
eddy current loss (stator)	$7.3 \cdot 10^{-1}$	0.99	$6.4 \cdot 10^{-1}$	0.99	$6.7 \cdot 10^{-1}$	0.99
hysteresis loss (rotor)	$1.81 \cdot 10^{0}$	0.98	$1.65 \cdot 10^{0}$	0.99	$1.71 \cdot 10^{0}$	0.98
hysteresis loss (stator)	$6.5 \cdot 10^{-1}$	0.99	$5.5 \cdot 10^{-1}$	0.99	$6.1 \cdot 10^{-1}$	0.99
	$\overline{\varepsilon}_{\text{MAE}}$	ε_{PCC}	$\overline{\varepsilon}_{\text{MAE}}$	ε_{PCC}	$\overline{\varepsilon}_{\text{MAE}}$	ε_{PCC}
Electromagnetic torque $\mathcal{T}$	$0.7 \cdot 10^{0}$	0.98	$0.65 \cdot 10^{0}$	0.99	$0.67 \cdot 10^{0}$	0.99
Flux linkage ψ_1 coil 1	$2.17 \cdot 10^{-4}$	0.99	$1.30 \cdot 10^{-4}$	0.99	$1.64 \cdot 10^{-4}$	0.99
Flux linkage ψ_2 coil 2	$1.87 \cdot 10^{-4}$	0.99	$1.64 \cdot 10^{-4}$	0.99	$1.75 \cdot 10^{-4}$	0.99
Flux linkage ψ_3 coil 3	$1.53 \cdot 10^{-4}$	0.99	$1.4 \cdot 10^{-4}$	0.99	$1.65 \cdot 10^{-4}$	0.99

for optimized multi-branch DNN over test samples. Those results confirm that the network's prediction performance is robust, meaning that independent of the training, test, and validation sets. The iron losses are assessed using the MRE since it encompasses a wide range of values, whereas torque and flux are evaluated using the MAE metric. This choice is made due to the heightened sensitivity of torque $\mathcal{T}$ and fluxes ψ_1, ψ_2, ψ_3. It should be noted that the MAE is calculated over the average of the fifteen predicted time steps for each operating point of all test PMSM designs. At last, the multi-branch DNN is trained with a randomly chosen training set consisting of $N_{\text{train}} = 40390$ PMSM designs, which accounts for approximately 90% of the total $N_{\text{PMSM}} = 44877$. The validation set ($N_{\text{val}} = 2243$) and test set ($N_{\text{test}} = 2244$) each represent around 5% of the dataset. These sets are kept consistent for the data-driven DNN training for the quantitative comparison. Figure 5.4 shows the distribution of training, test, and validation for several parameters and

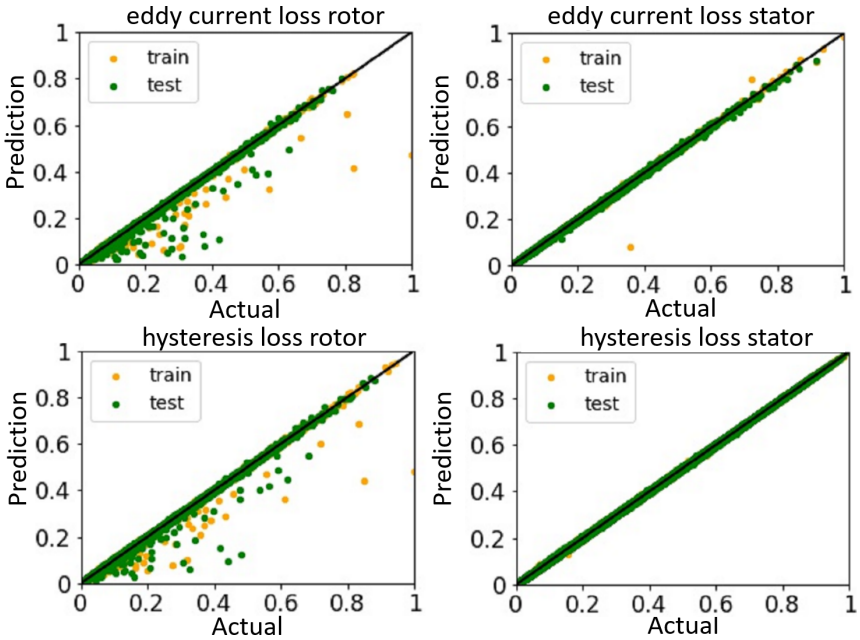


Figure 5.8: Prediction plots of iron-losses over test samples. Figure taken from [153, Fig. 9], © 2022 IEEE.

final KPIs. The validation and training curves for the final training and validation sets are displayed in Figure 8.1. Table 8.10 provides details about the numerical accuracy of intermediate measures over the test samples. The prediction plots for iron losses at each operating point of all test designs are shown in Figure 5.8. The horizontal axis represents the true value, while the vertical axis represents the predicted value. The numerical results show that the multi-branch DNN predicts intermediate measures with high accuracy for most of the test samples. However, a few operating points from both the training and test sets are observed to have poor predictions for iron losses in the rotor region. This might be due to the non-convergence of the FE solver during data generation. The calculation time for a new PMSM design (at all 37 operating points) is approximately 100 ms, which is significantly lower than the magnetostatic FE simulation time of around $3-5$ hours per design. Figure 5.7 and Figure 8.2 present the flux and torque predictions for three operating points from three test designs, demonstrating different input electrical excitation conditions. Note that "HA"is a short abbreviation for the hybrid approach in the figures. In the figures, the blue box represents the time steps of flux-linkages and torque predicted by the multi-branch DNN, while the green box represents the time steps calculated by leveraging the magnetic state symmetry. It can be observed from the analysis presented in Figure 5.7a that the multi-branch DNN exhibits low prediction accuracy for the operating point characterized by a zero input phase current (no load condition) and control angle.

Table 5.3: Hybrid and data-driven DL approach over test samples. Table taken from [153, Tab. 6], © 2022 IEEE.

KPIs	Hybrid approach		Data-driven DL approach	
	$\overline{\varepsilon}_{\text{MRE}}$	ε_{PCC}	$\overline{\varepsilon}_{\text{MRE}}$	ε_{PCC}
y_1	0.35	1.00	0.39	1.00
y_2	0.34	1.00	0.40	1.00
y_3	0.60	0.99	0.55	1.00
y_4	1.62	0.99	3.42	0.98
y_5	0.14	1.00	0.26	1.00
y_6	0.13	1.00	0.16	1.00
y_7	2.93	0.99	6.06	0.98

5.4.2 Quantitative analysis

In this subsection, the hybrid approach is compared with the data-driven DL approach based on scalar parameters discussed in Chapter 4. Figure 5.9 illustrates the evaluation using MRE as the metric, covering an expanding training size ranging from 5% to 100% of the complete training dataset, i.e., $N_{\text{train}} = 2020$ to $N_{\text{train}} = 40390$ designs. It is important to note that the evaluation is conducted on a consistent test set throughout the entire evaluation process. The hybrid approach consistently demonstrates higher prediction accuracy for KPIs $y_1, y_2, y_4, y_5, y_6, y_7$, whereas the data-driven DL approach achieves slightly better accuracy for KPI y_3. The data-driven DL methodology consistently exhibits a trend of steady MRE convergence as the training data increases, which is not observed in the first three KPIs when using the hybrid approach. Nevertheless, both approaches can predict KPIs for PMSM designs that have not been exposed during training. Normalized prediction plots for both meta-models trained on the full training set are depicted in Figure 5.10, showcasing their performance on the test designs. Statistical analysis for the same, using MRE and PCC, is provided in Table 5.3.

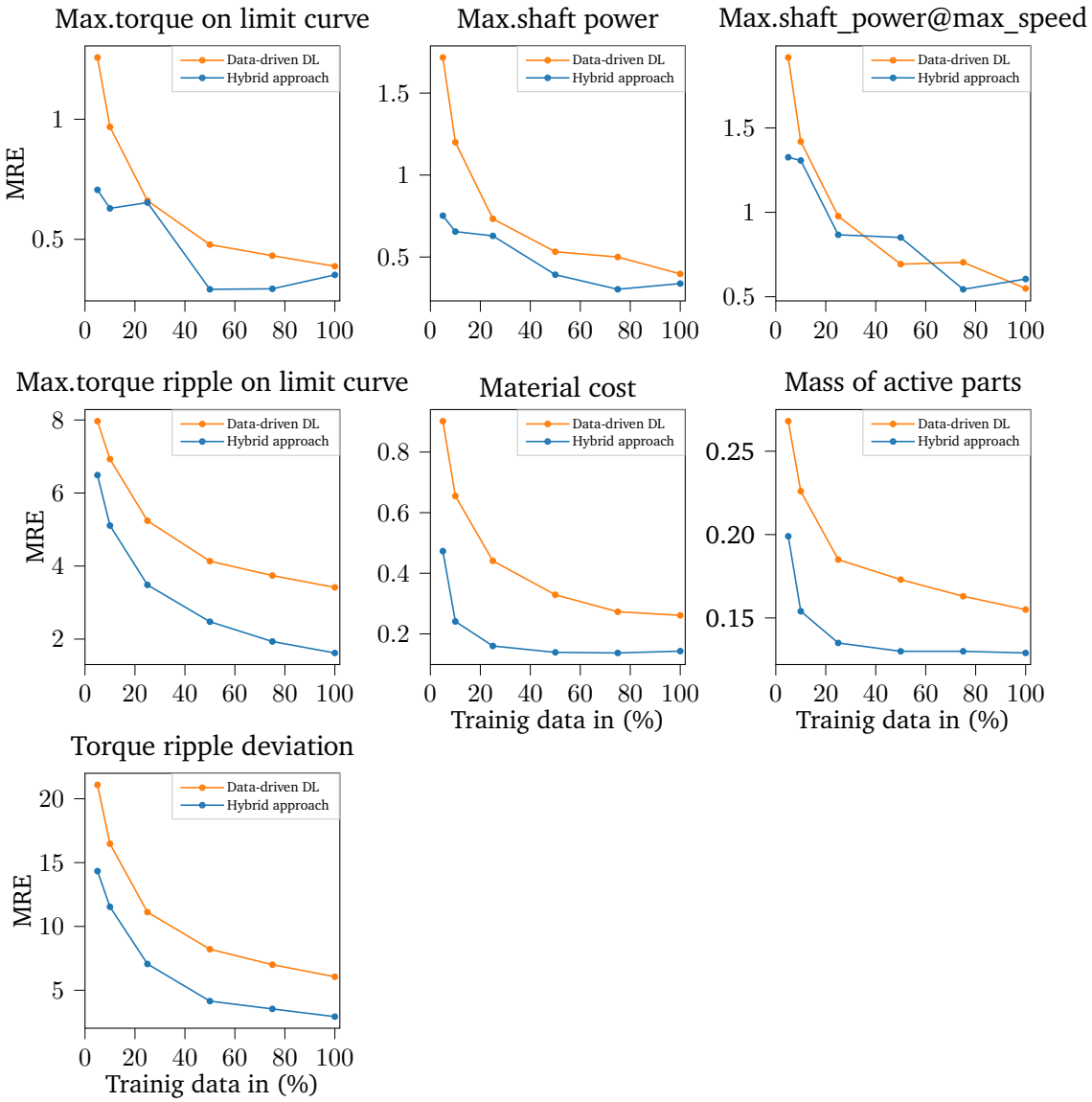


Figure 5.9: KPIs evaluation over varying training set size. Figure taken from [153, Fig. 14], © 2022 IEEE.

It can be seen in the second diagram of Figure 5.1 that the multi-branch DNN only relies on varying input design parameters and is independent of the system parameters. This means that if system parameters are modified, the final KPIs are changing, but there is no need to retrain the multi-branch DNN for such scenarios. However, this is not the case for the data-driven DL approach, where the DNN is directly trained on KPIs. Hence, the hybrid approach offers more flexibility in accommodating variations in parameters. During the post-processing, the physics-based models are fed with results of FE simulation, system parameters, and a few design parameters to compute other challenging measures for detailed analysis of PMSMs as explained in Sec. 2.2.3. Figure 8.3 presents different characteristic curves, including maximum shaft power, maximum torque limit, short circuit current, and open circuit voltage, at various rotor speeds for three PMSM designs from the test set. Efficiency maps can be calculated for the given speed vector. Figure 5.11 depicts efficiency maps for three test designs, each shown in a separate row. The first efficiency map in each row is computed using the conventional approach, while the second one is obtained using the hybrid

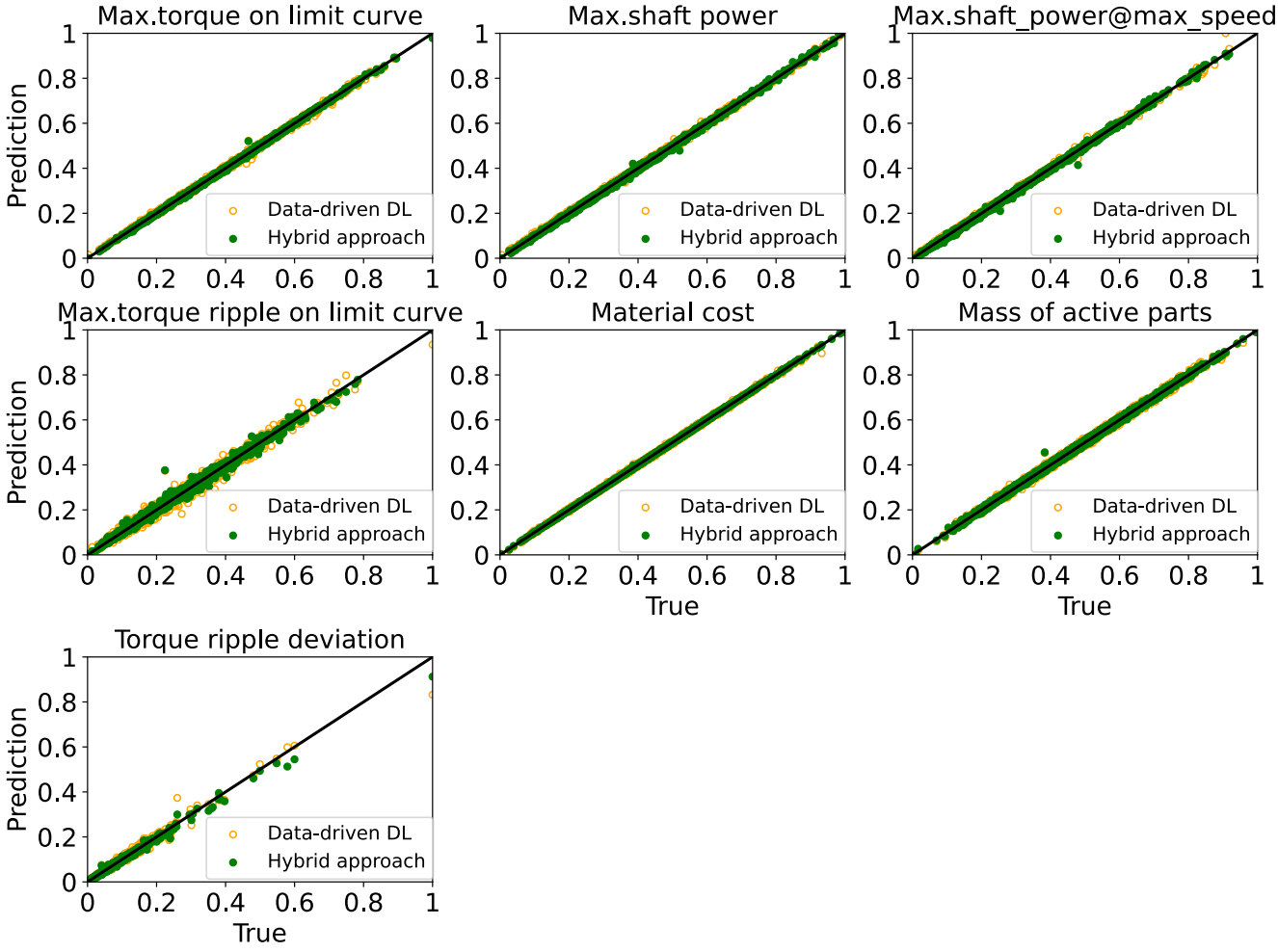


Figure 5.10: KPIs prediction plot over test samples. Figure taken from [153, Fig. 13], © 2022 IEEE.

approach. The third plot illustrates the percentage-based absolute difference between the two approaches. In all three plots, the absolute difference is less than 5% in the high-efficiency regions. A difference of more than 5% is observed in either low-speed or low-torque regions. Nevertheless, the lower prediction accuracy in this region does not significantly impact the calculation of the overall efficiency of the PMSM at other operating points, particularly in regions of high efficiency. This is evident in all three test designs illustrated in Figure 5.11. A possible reason could be that the hybrid approach is not sufficiently accurate for torque prediction at specific operating points. For instance, when considering scenarios such as zero input phase current and zero control angle (open circuit mode), as depicted in Figure 5.7a. Consequently, the hybrid approach may induce high errors in the final calculation of KPIs. Obviously, the precision of the KPI calculations depends on the accuracy of the intermediate measures prediction. The torque values of the PMSM at this operating point range from 10^{-1} to 10^{-3} Nm, which is relatively low compared to other operating regions. In the event that the trained meta-model predicts values within the range of 10^{-2} Nm instead of 10^{-1} Nm, a substantial prediction error arises. Other sensitive post-processing quantities, such as electromagnetic torque ripple, can also be computed with reasonable accuracy. This ripple originates from the interaction between the electromagnetic fields of the rotor and stator. In three-phase PMSMs, the electromagnetic torque ripple in PMSMs can be analyzed in terms of its mechanical harmonic orders, which are usually multiples of $6p$ (such as 24, 48, 72, and 96), using the Fourier coefficients of torque

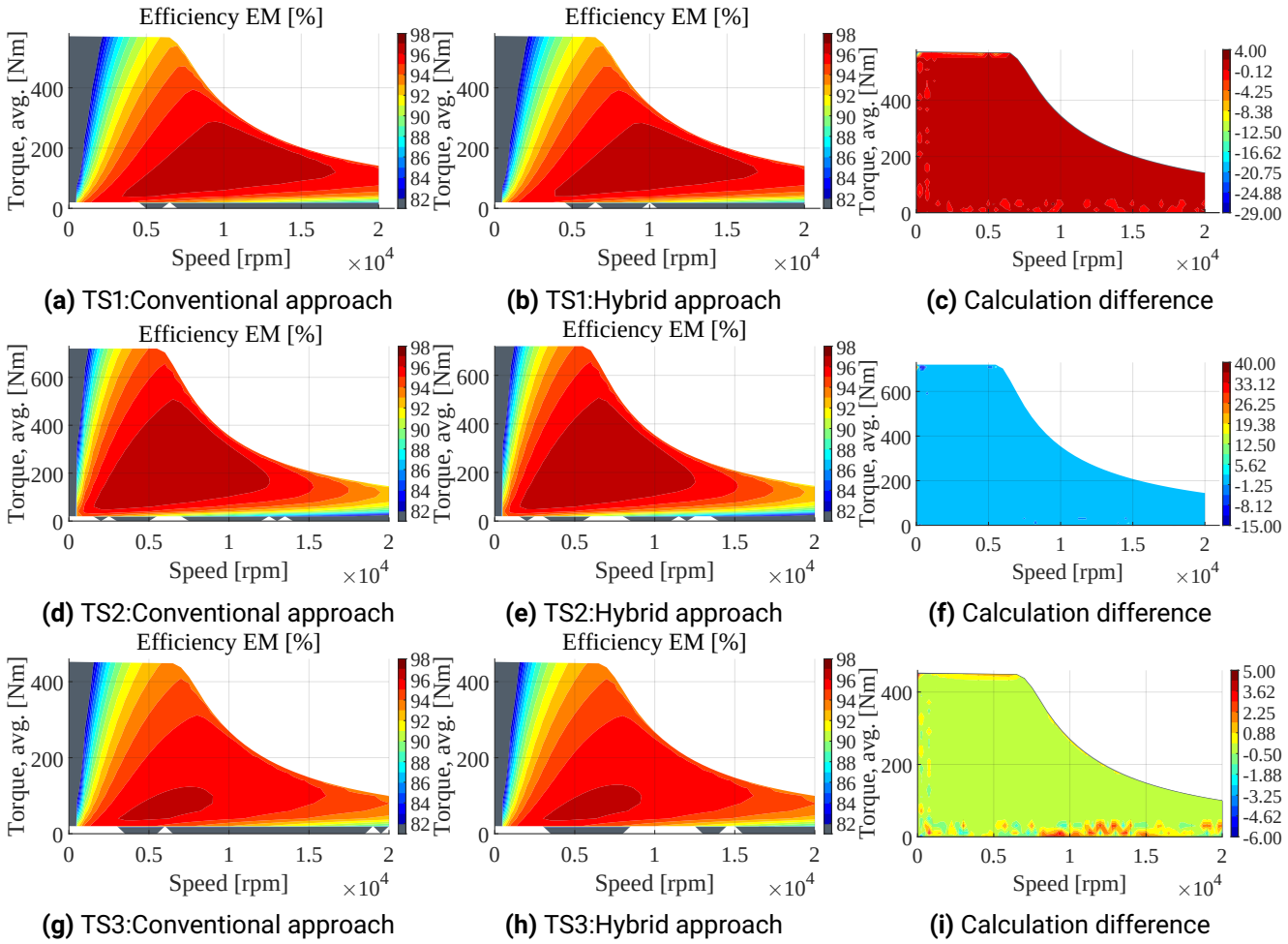


Figure 5.11: Illustration of efficiency map calculation for three test designs (TS stands for test sample. Figure based on [153, Fig. 12], © 2022 IEEE.).

during the post-processing, where p is the number of pole pairs [123]. These harmonic orders represent the frequencies at which periodic variations in torque output occur, and typically, the main order of torque ripple in a three-phase PMSM is given by $6p$ [132]. As shown in Figure 5.12, the electromagnetic torque ripple of order 24 (given that $p = 4$ is fixed in the dataset) is calculated for the same three test designs, with an overall absolute deviation of ≤ 2.2 Nm.

The proposed hybrid approach has a few drawbacks. The training time for the multi-branch DNN is approximately $2 - 2.5$ hours, which is around $6 - 8$ times longer compared to the data-driven approach that takes about $15 - 20$ minutes. This longer training time can be attributed to the larger number of samples required for meta-model training due to the number of operating points per design (37) and the presence of approximately 16 million trainable network parameters (weights and biases). As a result, the hybrid approach demands more computational resources than the data-driven approach. Furthermore, the inclusion of the post-processing tool in the hybrid approach leads to an apparent increase in the computation time for the KPIs calculation, shifting from milliseconds to seconds. Nevertheless, it is worth noting that the post-processing step still requires much less time than the FE simulation. Table 5.4 summarizes high-level comparison between the hybrid approach and the data-driven approach.

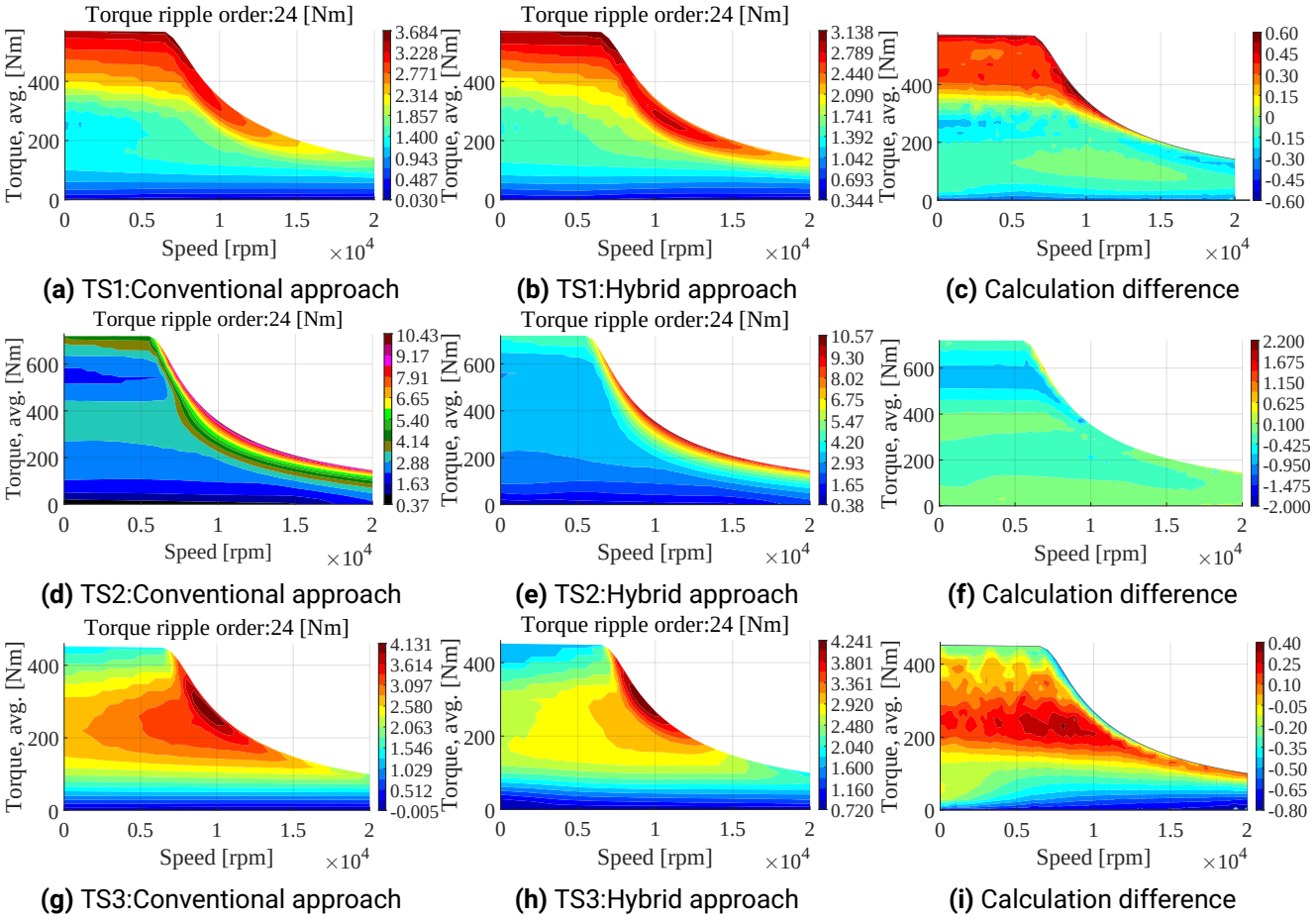


Figure 5.12: Illustration of torque ripple calculation for three test designs, considering order 24.

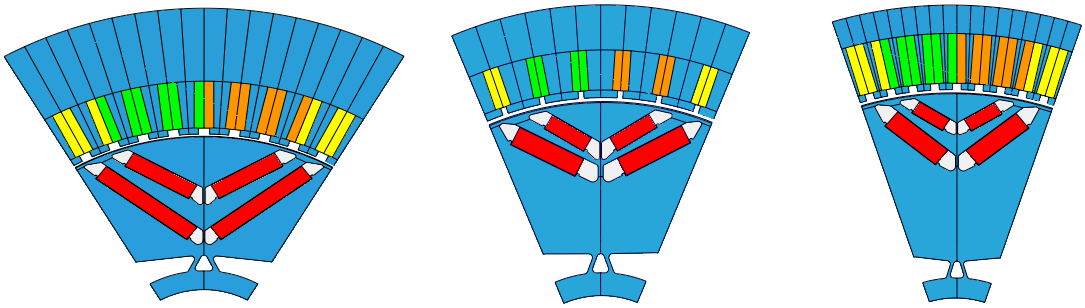
Table 5.4: High-level comparison: Hybrid approach vs Data-driven DL approach

Comparison criteria	Hybrid approach	Data-driven approach
Training computational effort	Moderate	Low
Flexibility (independent of system parameters)	Yes	No (implicit dependency)
KPIs prediction accuracy	High	Sufficient
Inclusion of physics	Yes	No
Calculation of characteristic maps	Yes	No

5.5 Application: MOO using hybrid-approach

In this section, the proposed hybrid approach is applied for the MOO in a real-world industrial setting parallel to the conventional FE-based workflow. A new dataset is created of PMSM double-v topology with a slightly different in-house geometry template to assess the feasibility of incorporating challenging design parameters such as pole pairs, slots per pole per phase, winding connection, and winding scheme for training the multi-branch DNN. These design parameters were constant in the previous dataset, so they were not included in the training of the multi-branch DNN. This section is divided into two parts. In the first part, the details of the new dataset, the training of the multi-branch DNN, and the MOO workflow are briefly presented. In the second part, the numerical results are presented. The content of this section is based on [155].

5.5.1 Dataset, training details, and MOO workflow



pole pairs: 3 pole pairs: 4 pole pairs: 5

Figure 5.13: Representative designs of PMSMs with varying pole pairs.

Table 5.5: PMSM design parameters

Parameters	Unit	Description	Category	Type	Range
p_1	[mm]	Stator outer diameter	Geometry	Continuous	[159-232]
p_{30}	[-]	Number of slots per pole per phase	Geometry	Discrete	[2, 3, 4]
p_{31}	[-]	Pole pairs	Topological	Discrete	[3, 4, 5]
p_{32}	[-]	Stator winding connection $\rightarrow$ 1: Star connection, 3: Delta connection	Electrical	Discrete	[1, 3]
p_{33}	[-]	Winding scheme $\rightarrow$ 0: Full pitch winding, 1: Short pitch winding	Electrical	Discrete	[0, 1]

Dataset: A new dataset is generated using the same procedure as explained in Sec. 4.3 and Sec. 5.2.1. The dataset includes more challenging parameters listed in Table 5.5. The complete list of parameters is provided in Table 8.12. Figure 5.13 demonstrates three representative PMSM designs with varying pole pairs. The total number of varying design parameters is $N_{\mathbf{p}} = 33$, and the filtered initial population yields a total of $N_{\text{PMSM}} = 51532$ PMSM designs. The number of operating points (37), constant parameters, and the system parameters are kept identical to the values given in Table 8.7, excluding those mentioned in Table 5.5.

Table 5.6: Intermediate measures over test samples with optimized multi-branch DNN

Intermediate measures	$\overline{\varepsilon}_{\text{MRE}}$	ε_{PCC}
eddy current loss (rotor)	8.24	0.97
eddy current loss (stator)	1.76	0.99
hysteresis loss (rotor)	8.85	0.97
hysteresis loss (stator)	1.07	0.99
	$\overline{\varepsilon}_{\text{MAE}}$	ε_{PCC}
Electromagnetic torque $\mathcal{T}$	1.7	0.99
Flux linkage ψ_1 coil 1	1.17×10^{-4}	0.99
Flux linkage ψ_2 coil 2	1.15×10^{-4}	0.99
Flux linkage ψ_3 coil 3	1.12×10^{-4}	0.99

Training: The same multi-branch DNN as proposed for the previous dataset shown in Figure 5.5 is used. All training and network hyperparameters are kept identical as described in Sec. 5.3. The multi-branch DNN is trained using a training-validation and test split, with percentages of $80 - 10 - 10$ applied to the total number of N_{PMSM} designs. The evaluation is performed using the MRE and PCC over the test set. The details can be found in Table 5.6. The evaluation is presented similarly to that shown in Table 5.2, where the averages across all operating points are taken for each intermediate measure. The prediction accuracy for all intermediate measures is lower compared to the previous dataset (refer to Sec. 5.2.1), but it remains reasonable overall, with MRE $\leq 10\%$. This could be attributed to the introduction of challenging discrete parameters, a different design space, and many outliers in the dataset. Considering the primary focus on applying the hybrid approach in industrial MOO settings, additional numerical comparisons and experiments are not conducted.

MOO worklow: The meta-model enables faster evaluation of new designs, thereby allowing the MOO execution with effective exploration of a large design space by evaluating more designs. This, in turn, may lead to the generation of designs close to optimum. The MOO of PMSM may involve large number of design variables, conflicting objectives, and constraints. In Sec. 4.2, the generalized MOO problem for designing the PMSM was defined. The broader goal is to solve (4.1) by satisfying conditions of (4.2) and (4.3). Any standard multi-objective optimizer can be utilized to solve (4.1-4.3). The population based nature-inspired evolutionary algorithm (refer to [5] and Sec. 2.3.2.2) is employed for the demonstration purpose. The MOO workflow based on the hybrid approach is presented in Figure 5.14. In the initialization step, objectives ($\mathbf{y}$), constraints ($\mathbf{c}$), and input parameters with their respective ranges are defined. Next, a generation-based optimization loop is triggered, where the initial population is created using the LHS. The objectives $\mathbf{y}$ and the constraints $\mathbf{c}$ are then calculated using the hybrid approach for all designs in the initial population. Afterward, a Pareto front is created by evaluating the fitness of all the samples using any suitable approach (e.g., tournament selection [203]), and the design archive is updated. The convergence condition is subsequently checked, e.g., based on reaching a maximum number of stagnant generations. If the convergence condition is not met, the optimizer generates new designs for the next generation through selection, adaptation, and mutation. This process continues until the convergence condition is met. Finally, the Pareto front is produced, and an optimal design can be selected based on requirements. The same workflow is followed for the conventional optimization to allow the comparison to some extent, where the calculation of objectives ($\mathbf{y}$) and constraints ($\mathbf{c}$) is performed using FE simulation.

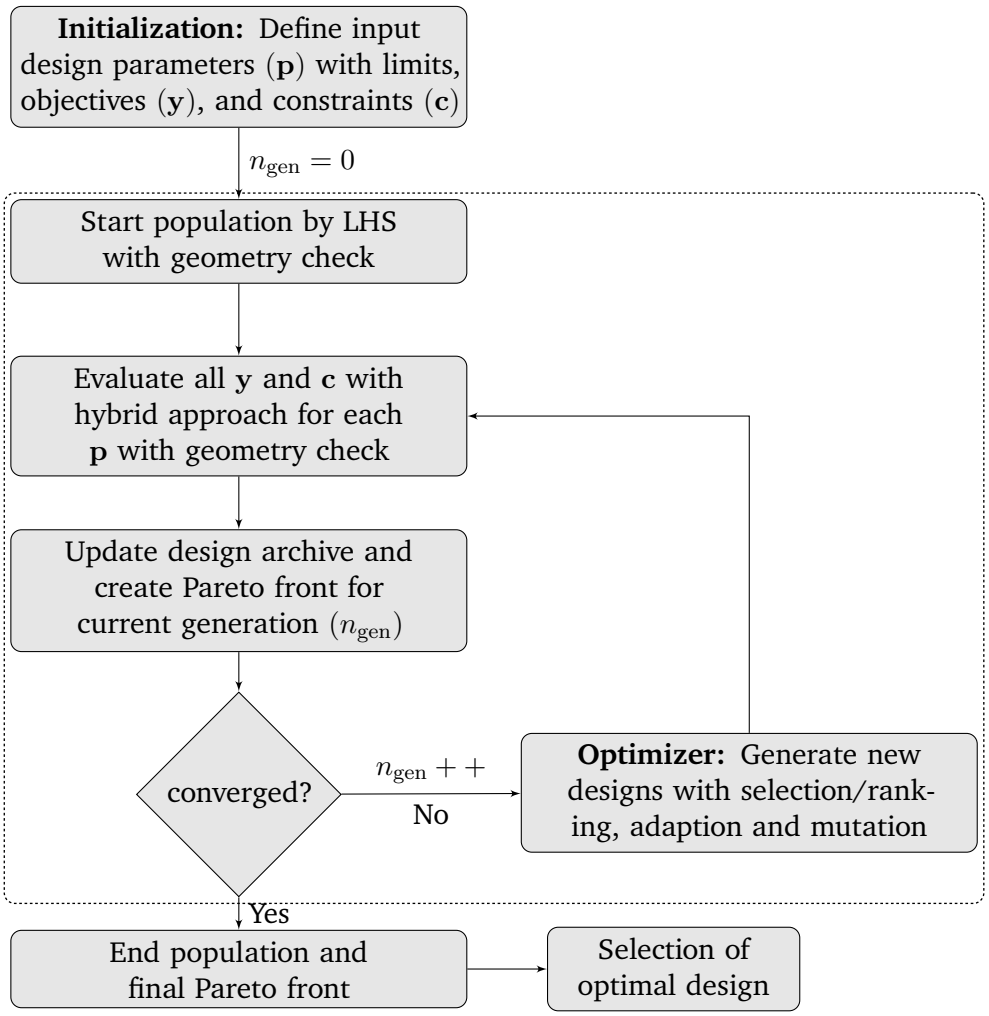


Figure 5.14: Proposed MOO workflow using hybrid approach. Figure based on [155, Fig. 2].

5.5.2 Numerical results

The MOO is demonstrated for two contrasting KPIs: Maximum power and Material cost. The MOO workflow is executed in parallel, employing both the conventional FE-based approach and the hybrid approach with identical hyperparameters settings. These MOO hyperparameters settings include initial designs, mutation rate (5%), convergence criteria such as stagnation generations (20) and a maximum number of generations (30), design selection method (tournament), fitness method (Pareto dominance),

Table 5.7: MOO computational details

Optimization	Successful designs	Pareto designs	Valid Pareto designs	Computation time
HA factor 1	13464/30000	168	92	∼18 hours
HA factor 2	26033/60000	223	122	∼40 hours
FE simulation	13811/30000	170	NA	∼6.5 days

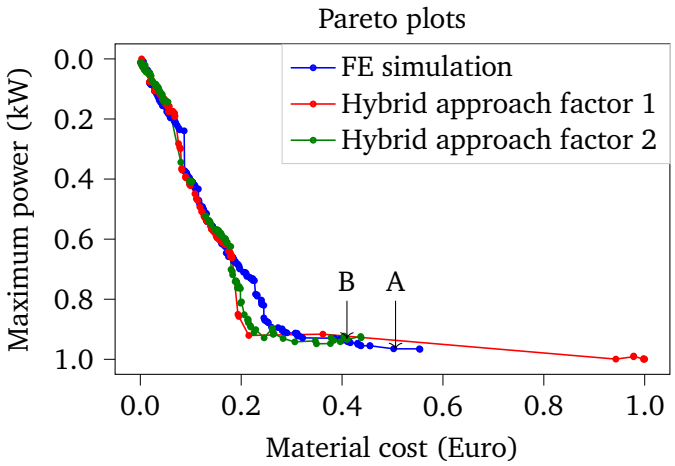


Figure 5.15: Pareto-fronts for Material cost and Maximum power. Figure based on [155, Fig. 3].

crossover probability (50%), population site per generation (1000) and constraints handling. Two additional KPIs are considered as constraints: maximum torque and short circuit current. All these settings are applied by experience.

Table 5.7 illustrates computational details, showing that the hybrid approach lowers the computational cost by approximately eight times. Consequently, twice the number of evaluations can be afforded while still maintaining a reduced computational cost of approximately four times. This is referred as "factor 2". Thus, the optimal region can be explored more precisely. The column "Successful designs"in Table 5.7 refers to geometrically feasible designs that meet all the evaluation criteria by satisfying the constraints and parameter bounds during the MOO run. On the other hand, "Valid Pareto designs"represent designs that fulfill all the conditions through the hybrid approach and are also validated by the FE-based conventional workflow. In other words, these are valid Pareto designs that have been verified through FE simulations. Figure 5.15 illustrates the Pareto fronts for all three MOO runs. The Pareto fronts for the hybrid approach display the true values of the "Valid Pareto designs"determined through FE simulations. The normalized

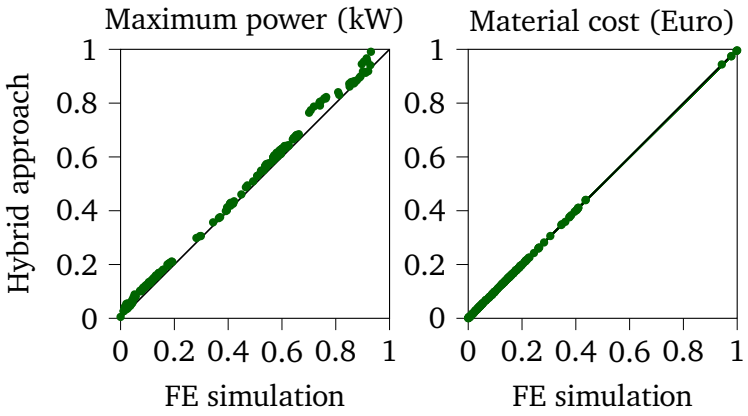
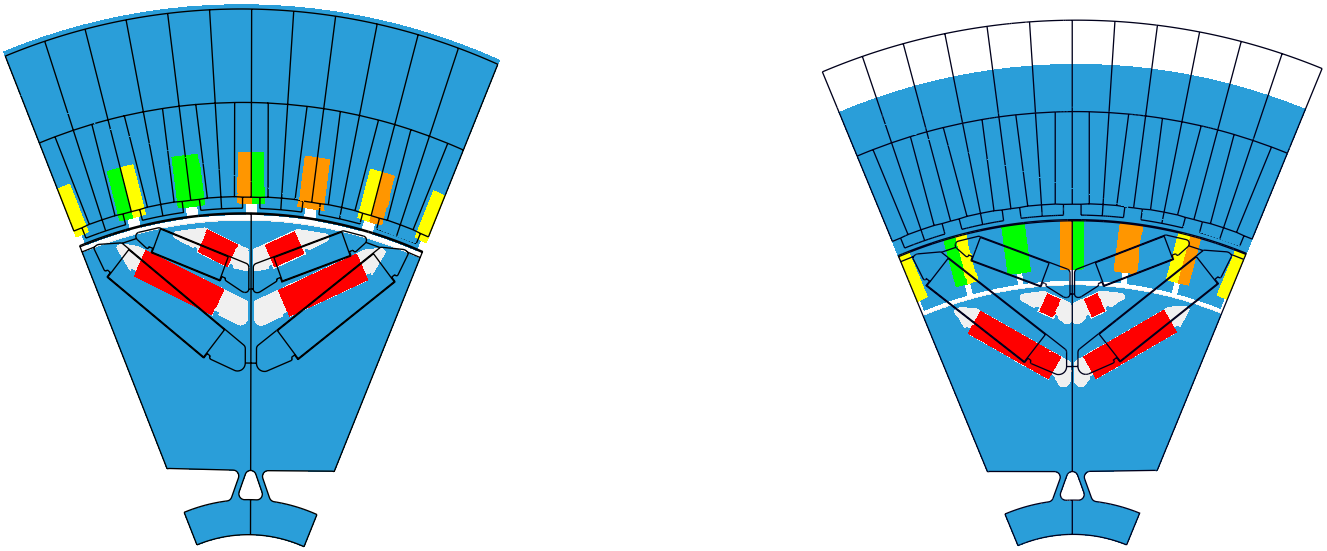


Figure 5.16: Prediction plot of valid Pareto designs for the hybrid approach from Figure 5.15. Figure based on [155, Fig. 4].



(a) FE Pareto design (A)

(b) Hybrid Pareto design (B).

Figure 5.17: The black lines represent the initial design, while the colored faces depict comparable Pareto designs (A and B) selected from Figure 5.15.

prediction plots for all these valid designs are shown in Figure 5.16 for both KPIs. The MRE for Material cost is (0.13%) and for Maximum power is (2.37%) for all these valid Pareto designs. For demonstration purposes, two Pareto designs are selected (designated as A and B in Figure 5.17), one from the hybrid approach's Pareto front and the other from the FE-based Pareto front, both having the same number of pole pairs and are close from the assessment perspective of the maximum power KPI. These designs are then compared with the initial design, which shares the same pole pair configuration. The structural comparison is depicted in Figure 5.17, and the corresponding gains for both KPIs are given in Table 5.8.

Next, the new MOO runs are conducted in parallel while keeping the input parameters listed in Table 5.5 at constant values. The purpose was to observe the behavior of the multi-branch DNN, when the design space

Table 5.8: Analysis of Pareto designs: A and B

Improvement compared to initial design		
KPIs	**FE Pareto design (A)**	**Hybrid Pareto design (B)**
Material cost saving	-40.07%	-51.07%
Maximum power	+19.37%	+15.98%

is reduced by keeping these parameters constant during the MOO process. Figure 5.19 shows prediction plots for both KPIs concerning valid Pareto designs from the hybrid approach. In this setting, the MRE for the material cost is 0.93% and for the maximum power is 6.81%, which is still within reasonable limit but higher than the previous scenario. It shows that the multi-branch DNN can also be employed with next run of optimization with reduced design space, without the need for re-training. In this case, as can be seen in Figure 5.18, FE based Pareto front is more superior than the hybrid approach, however the hybrid Pareto front is not far behind in terms of performance.

The hybrid approach demonstrates comparable MOO results to the FE-based conventional workflow with

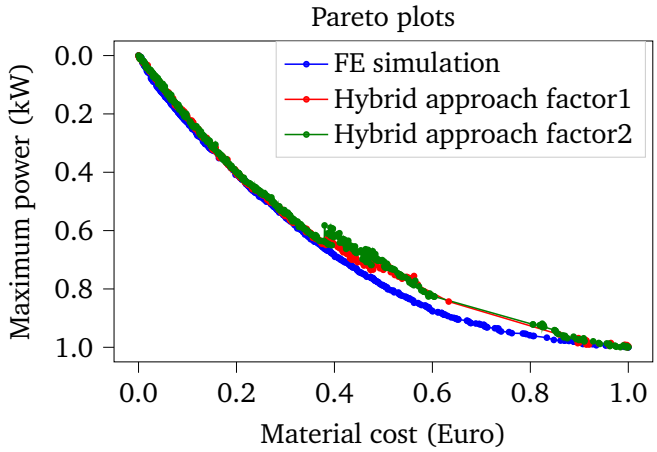


Figure 5.18: Pareto-fronts for the MOO with the constant value of parameters listed in Table 5.5

reasonable prediction accuracy, while being computationally cost-effective. This also allows for efficient exploration of the input design space, enabling more precise investigation of the optimal region at a low computational cost. Consequently, this implies the possibility of producing more designs closer to the optimum within a large and complex input design space.

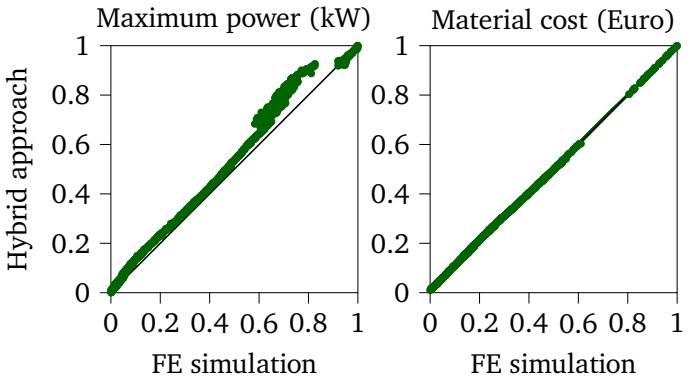


Figure 5.19: Prediction plots of valid Pareto designs for the hybrid approach shown in Figure 5.18

5.6 Summary

In this chapter, a few limitations of the data-driven approach are addressed (Chapter 4). The hybrid approach is proposed that combines both physics and data-driven models to quantify the performance of PMSMs.

The multi-branch DNN (see Figure 5.5) is trained using supervised learning to predict intermediate

measures, which solely rely on varying scalar input design parameters. These predicted intermediate measures, together with system parameters, are then fed into a physics-based post-processing tool to compute the target KPIs. This made the meta-model training independent of the post-processing and system parameters, as opposed to the data-driven approach, which inherently involved fixed system parameter settings for the training (see Figure 5.1). Moreover, from an application standpoint, this decoupling allowed for the inclusion of more intricate performance measures in the analysis of PMSMs, such as computation of efficiency maps and performance curves at the desired speed and torque vector.

An extensive quantitative comparison is shown with the data-driven approach. The results demonstrated that the hybrid approach outperformed the data-driven DL method consistently in terms of KPI estimation accuracy. This advantage can be attributed to two factors. Firstly, it is observed that learning a few time steps of intermediate measures proved less complex than directly learning cross-domain KPIs. Secondly, an in-house physics-based post-processing tool incorporates the laws of physics to ensure calculated KPIs adhered to the appropriate constraints.

However, it is important to note that the hybrid approach required more computational effort than the data-driven approach, both in terms of meta-model training and when evaluating new designs. This was due to a significantly higher number of training samples than the data-driven direct approach and also the fact that the post-processing tool takes a few minutes. Still, the computation time was markedly lower than that of the conventional approach.

The proposed hybrid approach is applied to the industrial MOO loop in a high-dimensional complex design space, running it in parallel with the conventional FE-based workflow. The hybrid method proved to perform comparably to the conventional approach while being eight times more computationally efficient. This enables more effective exploration of the optimal region by evaluating a larger number of designs during the MOO. An additional MOO run was further conducted, with some parameters kept constant to reduce the parameter space and observe the behavior of the meta-model without re-training. Despite a drop in prediction accuracy compared to full space optimization, it remained within an acceptable range, and the final Pareto front was observed to be close to that of the conventional workflow.

In the chapters thus far, both the data-driven and hybrid approaches using the scalar parameter-based input were focused on training a single machine type or rotor topology at a time. In the next chapter, a generative model based approach is introduced to enable concurrent MOO of multiple machines and rotor topologies using the scalar parameter-based input.

6 Concurrent optimization of heterogeneously parameterized electrical machines

In the previous two chapters, both data-driven and hybrid approaches were limited to a single rotor topology or a single machine type (i.e., the PMSM) for the scalar parameter-based representation. In this chapter, a VAE-based workflow is proposed that unifies the design space of differently parameterized rotor topologies and machine types for the scalar parameter-based input. Consequently, this enables concurrent MOO of multiple rotor topologies and rotating machines using the integrated design space.

This chapter begins by motivating for the concurrent MOO of multiple rotor topologies or machine types. A detailed explanation of the proposed VAE-based workflow is then provided in Sec. 6.2. Following that, the presented approach is applied in two scenarios: one involving differently parameterized rotor topologies for PMSM and another involving two machine technologies, ASM and PMSM, which operate on different physical principles. The prediction performance of the VAE-based meta-model with the individually trained DNN meta-model is compared, and the MOO is executed for each scenario using the unified latent space. The content and structure of this chapter are based on our work in [151, 154].

6.1 Motivation

In industry, engineers often face the challenge of selecting the most suitable machine technology or rotor topology for a given application, considering contrasting global KPIs such as shaft power, cost, torque, and more. Generally, this decision relies on classical simulation based optimization, past experiences, and existing databases. However, there are instances where multiple machine technologies or rotor topologies meet all the global KPIs for the target application. For example, the EESM and the IPMSM are two machine types commonly considered for high speed EV applications. A comparison is made between these two machine types based on global KPIs such as cost and efficiency in [37]. Using the FE-based virtual prototyping, a quantitative performance comparison is conducted among IM, SRM, and PMSM over common KPIs such as efficiency and vibration in [230]. In such cases, separate optimization of each technology or rotor type may be required due to distinct parameterization to determine the superior option within the identical boundary parameters and objectives. The conventional FE simulation based approach can be computationally expensive and time-consuming. The data-driven supervised learning approach enables faster MOO; however, it necessitates the individual training of meta-models for each topology of rotor or machine type because of distinct parameterization. Given these challenges, there is a motivation to explore the benefits of concurrent MOO for distinctly parameterized machines. By optimizing different machine technologies or rotor topologies concurrently, the design process can be streamlined. This also reduces computational costs and accelerates the decision-making process. The concurrent MOO offers the potential to efficiently evaluate and compare the performance of multiple machine technologies or rotor types, which may lead to improved design outcomes and more informed engineering choices.

The concurrent MOO of rotor topologies or machine technologies with different parameterizations often involves a high-dimensional and complex scalar input design space. The search for an optimal solution in such a high-dimensional space prohibitively increases the computational cost and results in unrealistic scenarios where MOO becomes infeasible. Hence, it becomes imperative to deal with a high-dimensional complex design space. The generative DL models, such as VAEs [102, 103] and GANs [71], are particularly useful in this context. It is worth noting that the utilization of generative models for optimizing electromagnetic devices is still in the early stages. In the most recent period, a handful of articles have demonstrated the use of generative DL models for optimizing electromagnetic devices. For instance, the VAE and DNN are simultaneously trained for topological optimization of electromagnetic die press [214]. The GAN is trained in conjunction with MLP to predict non-dominated Pareto designs with target KPIs in [46]. To overcome the challenge of generating sufficient data for training ML meta-models, a method is proposed in [197] that employs a deep generative model and a CNN. This method generates a significant amount of data while conducting only a small number of FE simulations. In another study [189], topology optimization of PM motors is demonstrated using the VAE and a multi-layer NN. This approach generates diverse PM motor shapes and predicts the associated motor characteristics within the MOO loop. In a recent study [77], the GAN and the CNN-based meta-models are employed to streamline the image-based optimization process of PMSM by reducing degrees of freedom through lower dimensional latent representation of images. However, this work investigates only a single-rotor topology (double v PMSM) and single-objective optimization. Artifacts and noise in the GAN-generated images are also observed, which could lead to inaccuracies in subsequent KPI predictions. It is also reported that the proposed image-based approach takes much longer training time and processing than the parameter-based model. All these studies have focused on image-based topological optimization of electromagnetic devices since generative models are commonly employed for image-based generative modeling [22, 167, 170]. It should be noted that generative models, for example, GANs, often fail to reconstruct valid image representations due to “mode collapse”[71, 84]. Other models, like VAEs, sometimes fall short of learning a latent space distribution adequately during training, which may lead to the generation of blurry images [22, 188, 212]. The training of GANs poses significant challenges, primarily due to their sensitivity to hyperparameters, their inability to work with explicit density functions, high computational requirements, a lack of inherent evaluation metrics, and convergence issues [84, 90, 170].

In Chapter 4, the data-driven DL approach for predicting KPIs of PMSMs with a specific rotor topology was introduced [152]. Parameter-based and image-based representations were extensively compared. The image-based approach can handle different parameterized topologies and machine types simultaneously, and it can accommodate reparametrization scenarios without requiring meta-model retraining since the image space remains unchanged. However, the primary concern is the generation and processing of high-resolution images because of computational resource limit. On the other hand, the scalar parameter-based approach has demonstrated high prediction precision and significantly lower computational effort compared to the image-based representation. It can also incorporate challenging simulation parameters such as varying voltage, current, and axial machine length, which can not be shown in the 2D image-based representation. Therefore, the scalar parameter-based representation was chosen for proposing the hybrid data- and physics-driven approach[153] in the previous chapter.

This chapter presents a VAE-based approach for the concurrent MOO of multiple machine types and rotor topologies [154, 155]. A vector of scalar parameters is utilized as the input representation. The encoder network encodes the high-dimensional complex input distribution into a lower-dimensional latent distribution. Following simultaneous network training, the DNN uses the latent input for predicting common KPIs, and the decoder reconstructs the corresponding scalar input. The proposed approach is motivated by the use of VAE for generating new chemical structures while predicting their properties [69]. In the following section, first, the concatenation of distinct input design spaces for multiple machine types or

rotor topologies is discussed. Subsequently, the proposed VAE-based training procedure is explained. The content of the next section follows our work presented in [151, 154].

6.2 Methodology

Consider T distinct machines or rotor topologies, denoted as $\mathbf{p}_t \in \mathbb{P}_t \subset \mathbb{R}^{d_t}$, where $t = 1, \ldots, T$.

It is assumed that the KPIs remain the same for each machine type and rotor topology. These KPIs can be computed conventionally (e.g., via FE simulation or analytical calculation) and can be abstractly described as a vector $\mathbf{y} = \mathbf{K}_t(\mathbf{p}_t)$.

A unified design space $\mathbb{P} \subset \mathbb{R}^d$ is created by combining the individual design spaces of each machine and topology. Where, the total dimension is $d = 1 + d_1 + \ldots + d_T$.

Following the integration of design spaces, parameter vector of l-th sample of the t machine type or topology from the complete dataset is described as

$$\mathbf{p}^{(l)} = [t, \mathbf{0}, \ldots, \mathbf{0}, \mathbf{p}_t^{(l)}, \mathbf{0}, \ldots, \mathbf{0}], \tag{6.1}$$

and corresponding KPIs vector as $\mathbf{y}^{(l)} = \mathbf{K}_t(\mathbf{p}_t^{(l)})$.

Consequently, the entire input dataset with a total of M_{tot} samples can be expressed as

$$\mathbf{D} := \left\{ \mathbf{p}^{(l)} \;\middle|\; \text{for } l = 1, \ldots, M_{\text{tot}} \right\} \tag{6.2}$$

that describe the complete set of samples encompassing all topologies and machines.

All parameters in the input $\mathbf{p}_t$ are assumed to be independent in this thesis. As explained in Sec. 3.2.3.2, the encoder network $\mathbf{F}_\Theta$ probabilistically transforms a high-dimensional complex distribution into a lower-dimensional latent distribution $\mathbf{z}$. This transformation enables the VAE to learn a compressed representation of the input data $\mathbf{p}$. In other words, the d-dimensional random variable $\mathbf{p}$ which produces all the designs in $\mathbf{D}$ can be described via hidden variables $\mathbf{z}$ with a latent dimension of n. Notably, in practice, $n \leq d$. To obtain this, the prior distribution of the latent space needs to be defined. It is assumed that the prior distribution is known that serves as a reference for the latent variables in the VAE. Any distribution with determined distribution parameters can be chosen, but the multivariate Gaussian having diagonal covariance is widely used due to its known properties (zero mean and unit variance). It allows sampling for generating new samples and provides a closed-form solution [63]. The standard normal distribution explains that each dimension of the latent space is independent of the others. This enables the VAE to capture diverse and uncorrelated features in the latent representation. The encoder network estimates the conditional distribution $\mathbf{P}(\mathbf{z}|\mathbf{p})$ with the presumption of a standard normal distribution as the prior over $\mathbf{z}$. The encoder $\mathbf{F}_\Theta$ predicts the parameters of the conditional distribution, which include the mean ($\boldsymbol{\upsilon}$) and the diagonal elements of the covariance matrix ($\boldsymbol{\sigma}$) as vectors of n output neurons. If (3.18) is rewritten in reference to input parameters $\mathbf{p}$ as

$$(\boldsymbol{\upsilon}, \boldsymbol{\sigma}) := \mathbf{F}_\Theta(\mathbf{p}), \tag{6.3}$$

where Θ are encoder network training parameters. The latent vector $\mathbf{z}$ is then sampled from the latent distribution $\mathbf{P}(\mathbf{z}|\mathbf{p})$ using the reparametrization trick described by equation (3.19) for effective gradient learning via backpropagation. Afterward, the sampled vector $\mathbf{z}$ is inputted into two ANNs: the decoder network $\mathbf{F}_\Phi$ and the KPI predictor (DNN) $\mathbf{K}_\theta$. The decoder $\mathbf{F}_\Phi$ transforms the latent vector $\mathbf{z}$ into the corresponding

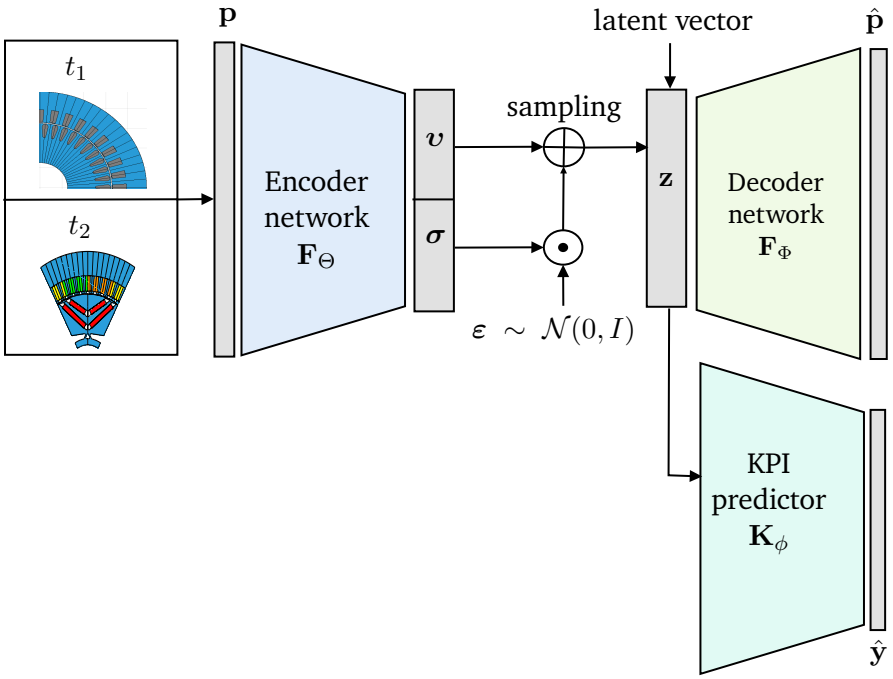


Figure 6.1: VAE-based training workflow. Figure taken from [151, 154, Fig. 2 and Fig.5], © 2022 IEEE.

design vector $\hat{\mathbf{p}}$, estimating the conditional distribution $\mathbf{P}(\mathbf{p}|\mathbf{z})$. The decoder $\mathbf{F}_\Phi$ functions as a design predictor. The KPI predictor $\mathbf{K}_\theta$ can be seen as analogous to the DNN described for the data-driven approach in Sec. 4.4.2.1. It predicts the KPI vector $\hat{\mathbf{y}}$ as output for the given latent input $\mathbf{z}$. The decoder can be mathematically described by rewriting (3.20) in terms of the design parameter $\hat{\mathbf{p}}$ as

$$\hat{\mathbf{p}} := \mathbf{F}_\Phi(\mathbf{z}), \tag{6.4}$$

where Φ represent trainable parameters for the decoder network. Similar to equation (4.5), the KPI predictor $\mathbf{K}_\phi$ can be mathematically expressed for predicting the KPI vector $\hat{\mathbf{y}}$ as

$$\hat{\mathbf{y}} := \mathbf{K}_\phi(\mathbf{z}), \tag{6.5}$$

where ϕ are the trainable network parameters. In the proposed approach, three ANNs are trained simultaneously: the encoder, the decoder, and a KPI predictor. This can be visualized from the schematic representation of the proposed approach shown in Figure 6.1. For the encoder and decoder, the same training loss terms maintained, consisting of MSE and KL-divergence ($\mathcal{D}_{\mathrm{KL}}$), as described in equation (3.21). This is due to the presumption of a standard normal distribution as the prior for the latent space and the need to reconstruct scalar parameter-based input from the latent input. An additional loss term for the KPI predictor needs to be introduced into the training process. Since predicting KPIs is a non-linear multiple-output regression problem, the commonly used MSE function is employed for this downstream regression task [70, Chapter 5]. The modified training loss (3.21) with respect to the network trainable parameters (Θ, Φ, θ), input sample $\mathbf{p}^{(l)}$, and true KPIs $\mathbf{y}^{(l)}$ can be written as

$$\begin{aligned}\mathcal{L}(\Theta, \Phi, \theta; (\mathbf{p}^{(l)}, \mathbf{y}^{(l)})) = &\left\|\mathbf{p}^{(l)} - \hat{\mathbf{p}}^{(l)}\right\|^2 + \left\|\mathbf{y}^{(l)} - \hat{\mathbf{y}}^{(l)}\right\|^2 \\ &+ \mathcal{D}_{\mathrm{KL}}\Big(\mathbf{P}(\mathbf{z}^{(l)}|\mathbf{p}^{(l)}, \theta) \;||\; \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})\Big).\end{aligned} \tag{6.6}$$

The entire process can be considered a combination of supervised and unsupervised learning. The learning of the encoder and decoder networks is an example of unsupervised learning since they are trained solely on input data, which means they do not require explicit actual data or supervision during training. On the other hand, the downstream task of predicting KPIs is an example of supervised learning because the KPI predictor is trained using true KPIs. Thus, the simultaneous training of the VAE and the KPI predictor includes both unsupervised and supervised learning in conjunction.

In the following two sections, the proposed VAE-based training workflow is applied to two different scenarios. In the first scenario, two differently parameterized PMSM topologies, namely Single V (SV) and Double V (DV), are investigated. In the second scenario, two distinct machine technologies, ASM and PMSM, are considered. After meta-model training, the MOO using the latent space for two common KPIs will be demonstrated.

6.3 Scenario 1: Heterogeneous parameterization by rotor topology

In this section, first, details about the datasets for both topologies are provided. Following that, the network architecture and training specifications are explained. Finally, the section is concluded with a numerical analysis, which includes the network's performance over test data, a comparison with individually trained DNNs, and a demonstration of the MOO. The content of this section is based on our work in [154].

6.3.1 Datasets

There are $T = 2$ PMSM topologies (SV and DV; refer Figure 6.2) considered for the datasets. The same process for data generation as described in section 4.3 is followed. After filtering erroneous designs from the initial population, $N_{\mathrm{SV}} = 14854$ SV samples and $N_{\mathrm{DV}} = 13424$ DV samples are obtained. There is no fixed

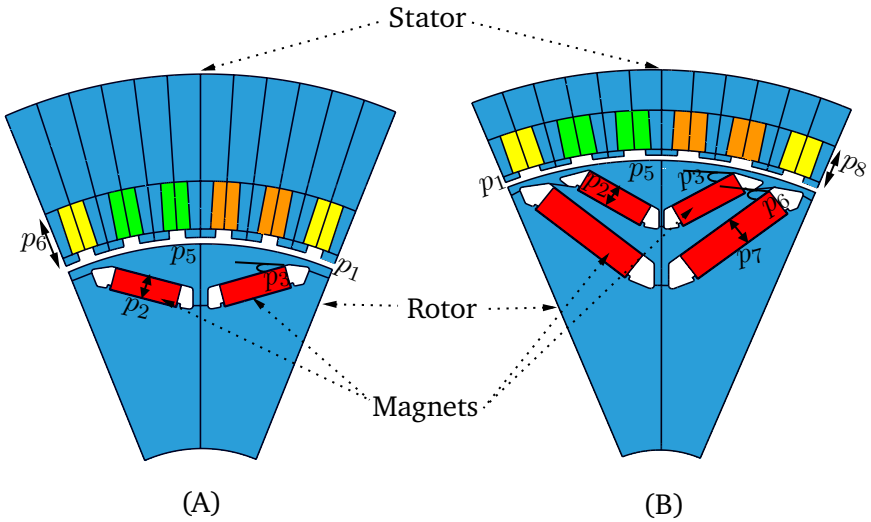
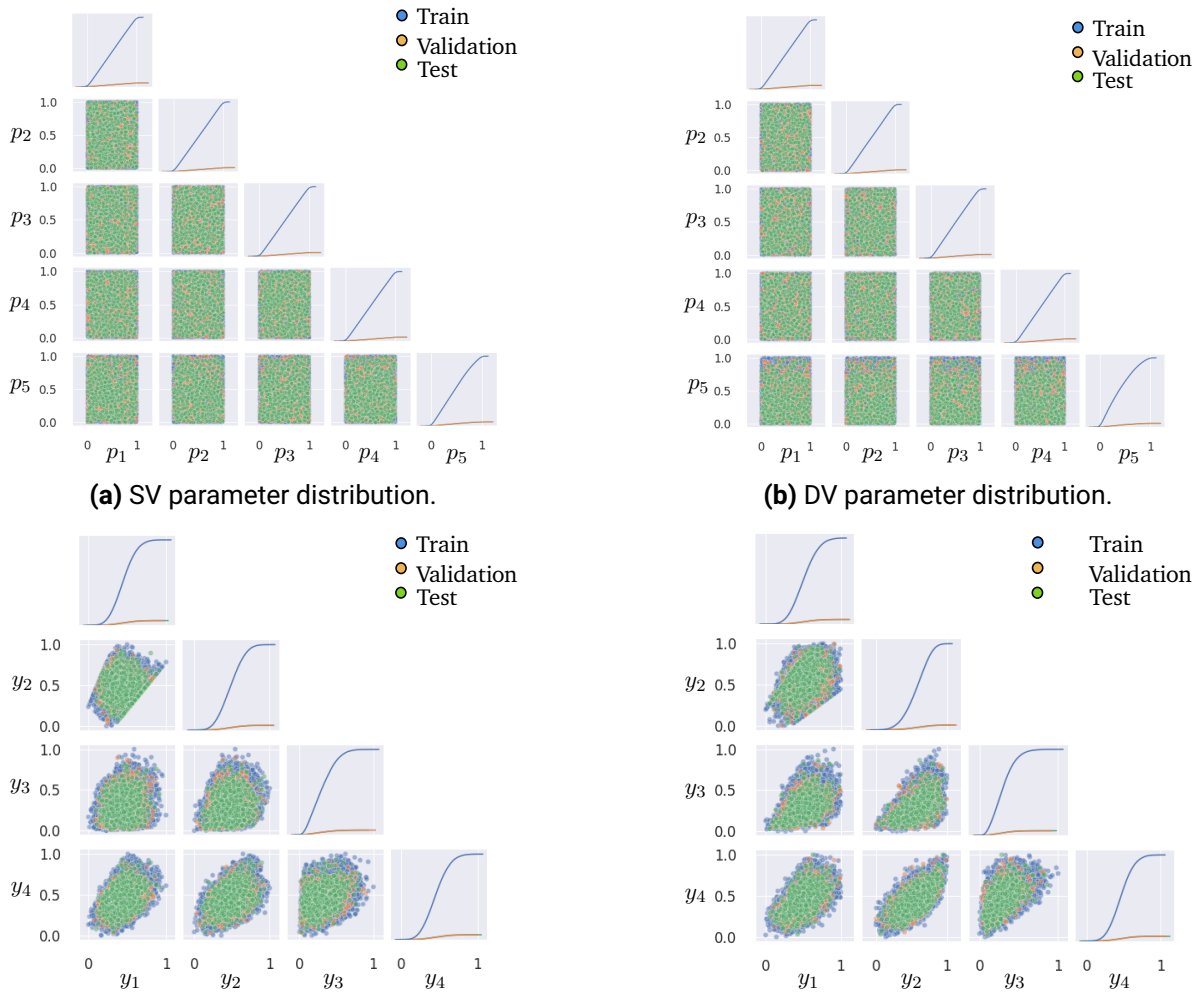


Figure 6.2: A) SV and B) DV representative samples (Pareto samples of Figure 6.10). Figure taken from [154, Fig. 1], © 2022 IEEE.

number for dataset generation, but from experience, the more data for training, the higher the network's approximating ability (refer Figure 5.9). The difference in the number of samples between datasets should be as small as possible (i.e., $\leq 10\%$), ideally zero. Otherwise, the meta-model becomes biased toward samples from one topology during training. The SV and the DV consists $d_1 = 13$ and $d_2 = 18$ varying parameters, respectively. A few of them are depicted in data representative samples in Figure 6.2. The full list is given in Table 8.14 and Table 8.15. The list of common KPIs is given in Table 6.1. The distribution of a few parameters and KPIs is depicted as pair plots in Figure 6.3 for both topologies. It can be seen that all



(a) SV parameter distribution.

(b) DV parameter distribution.

(c) SV KPIs distribution.

(d) DV KPIs distribution.

Figure 6.3: Visualization parameter and KPIs distribution.

five parameters follow a homogeneous distribution and are independent. On the other hand,all the target KPIs are inhomogeneously distributed. A few of the constant simulation parameters are listed in Table 8.13. To allow comparison to some extent, it is ensured that the system parameters, such as input phase current, voltage, and stator type, are identical. The ranges of common geometry parameters are also kept identical, e.g., iron length and rotor outer diameter. Furthermore, the cost of common materials (e.g., magnets, iron, aluminium, and copper) is assumed to be the same for both topologies.

6.3.2 Network structure and training details

Network structure

Three ANNs are determined for training: encoder $\mathbf{F}_\Theta$, decoder $\mathbf{F}_\Phi$, and KPI predictor $\mathbf{K}_\Phi$. The final configuration is obtained through manual and random approaches by evaluating approximately forty-one configurations, as explained in subsection 4.4.1. It is illustrated in figure 6.5. Information for each of the networks is as follows.

Encoder $\mathbf{F}_\Theta$ and decoder $\mathbf{F}_\Phi$: The encoder functions as an inference network that maps a complex high-dimensional design space into a lower-dimensional latent distribution, the parameters of which are known. The decoder serves as a design predictor, reconstructing the input from the latent distribution. When defining the encoder and decoder networks, there are two important network hyperparameters that significantly affect the accuracy of the VAE. The first one is the use of a 1D-convolutional layers and their corresponding transpose layers. The second one is the number of dimensions in the latent space.

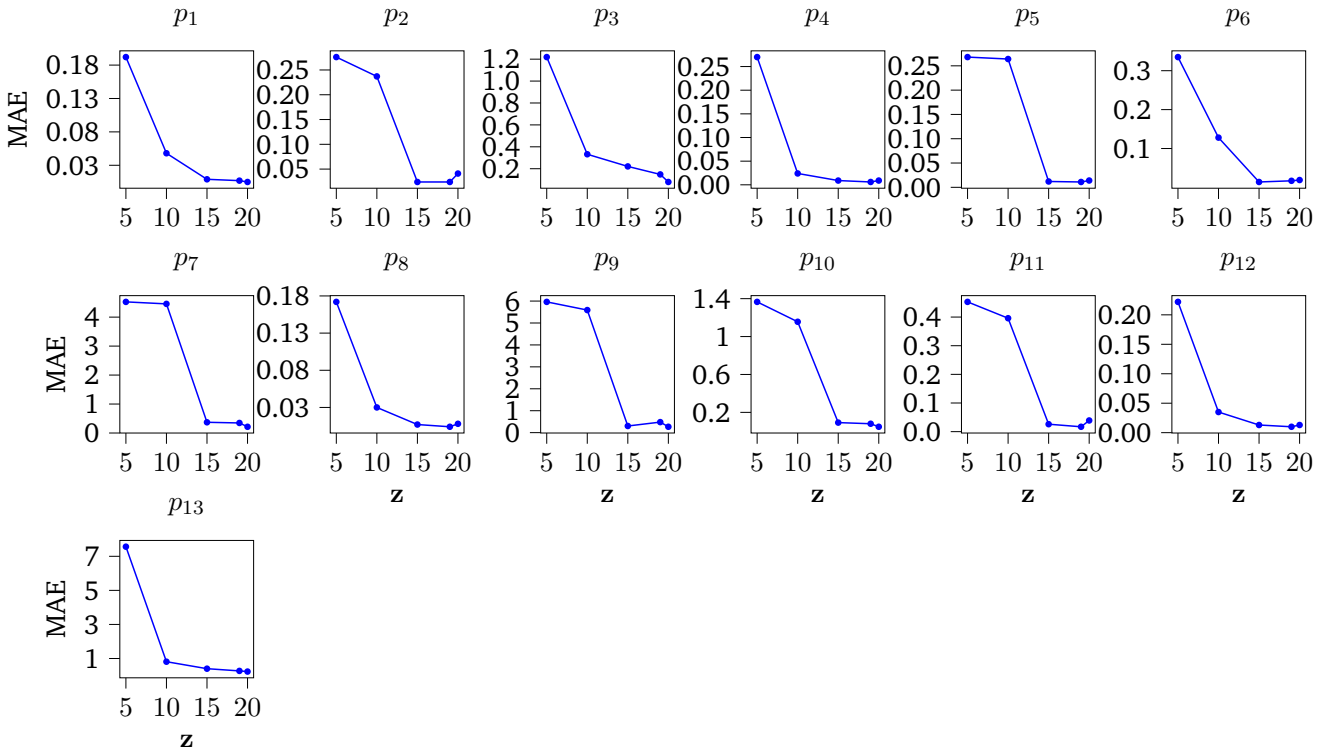
As shown in (6.1), the integrated design vector consists of the topology indicator parameter (which defines SV or DV topology), several zeros, and the actual parameters concerning PMSM topology. Initially, only standard fully connected layers were tried for the encoder and decoder networks. But the network failed to learn parameter reconstruction and the mapping to KPIs in the latent space, possibly due to the presence of sparsity in the input vector. To address this issue, 1D convolutional layers, known for their ability to capture local context and reduce computational effort by utilizing parameter sharing [107], were introduced. The 1D convolutional layers effectively capture the topology indicator parameter and features from the integrated design space. The utility of 1D CNN has been demonstrated in various applications, including real-time electrocardiogram monitoring in the health sector [108], fault diagnosis in rotating electrical machines [92, 179], torque estimation for PMSM [219], and more. An overview and analysis of the use of a 1D CNN in diverse applications are presented in [107].

The number of latent dimensions is another important hyperparameter for the VAE performance. Experiments were conducted by progressively increasing the size of the latent dimension, ranging from five to twenty. The maximum limit of twenty is chosen because the DV PMSM comprises input dimension $d_2 = 18$ in the integrated design space. Figure 6.4 illustrates the evaluation of parameter reconstruction in terms of MAE for increasing latent dimension for both topologies. The results show a consistent decrease in MAE with increasing latent dimension, and the error exhibits convergence around latent dimension $\mathbf{z} = 19$, which is equal to the sum of input dimension of PMSM DV $(d_2 = 18)$ and a topology indicator parameter. Furthermore, it is assumed that the input parameters of each PMSM topology are independent. Hence, for accurate parameter reconstruction, the number of latent dimension $n_\mathbf{z}$ should be set to a value greater than or equal to the maximum input dimension across all PMSM topologies, denoted as $n_\mathbf{z} \geq \max_t(d_t)$.

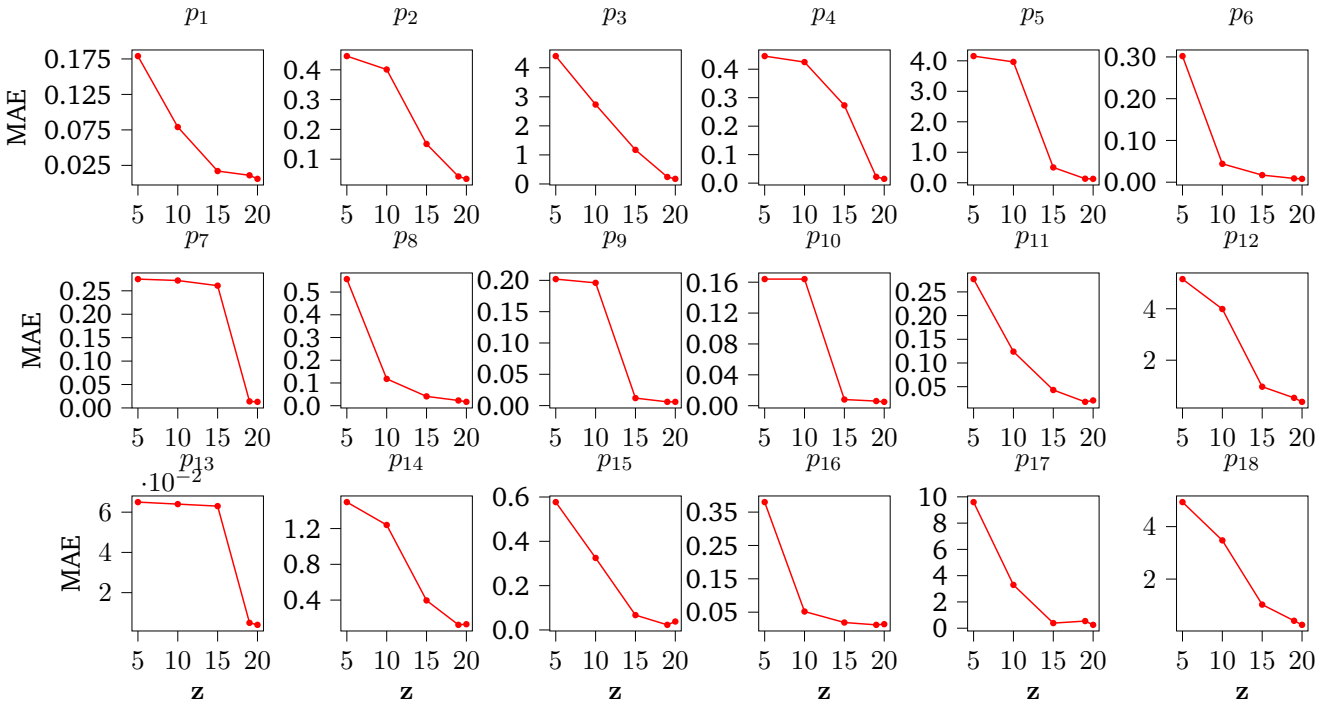
The final encoder network, illustrated in Figure 6.5a, comprises three 1D convolutional layers with specified filter size and stride. These layers are followed by a flatten layer and a fully connected layer. Subsequently, there are three output layers corresponding to the size of the latent space. Among these output layers, two provide the latent distribution parameters: mean ($\boldsymbol{\upsilon}$) and variance ($\boldsymbol{\sigma}$). The sampling layer utilizes these parameters to generate the latent vector ($\mathbf{z}$).

The decoder is the mirror structure of the encoder. It has an input layer with a size of the latent dimension ($\mathbf{z} = 19$) and an output layer with a size of the input parameter dimension ($\hat{\mathbf{p}} = 32$). The intermediate layers contain 1D transposed convolutional layer. The one difference is the use of a linear activation function in the output layer to predict the scalar design parameters of different ranges.

KPIs predictor $\mathbf{K}_\Phi$: The KPI predictor, as shown in Figure 6.5c, is a DNN consisting of five fully connected



(a) SV parameters reconstruction evaluation



(b) DV parameters reconstruction evaluation

Figure 6.4: Evaluation over varying latent dimension $\mathbf{z}$. Figure based on [154, Fig. 5], © 2022 IEEE.

layers with a softplus activation function. The input layer has a size of latent dimension ($\mathbf{z} = 19$), and the output layer has a number of neurons equal to the number of KPIs (4) to be predicted. It should be noted that, for a high-level comparison, equivalent individual DNNs are also trained for the prediction of KPIs. The only difference lies in the input layer size, as each individually trained network has an input dimension d_t specific to the t-th topology.

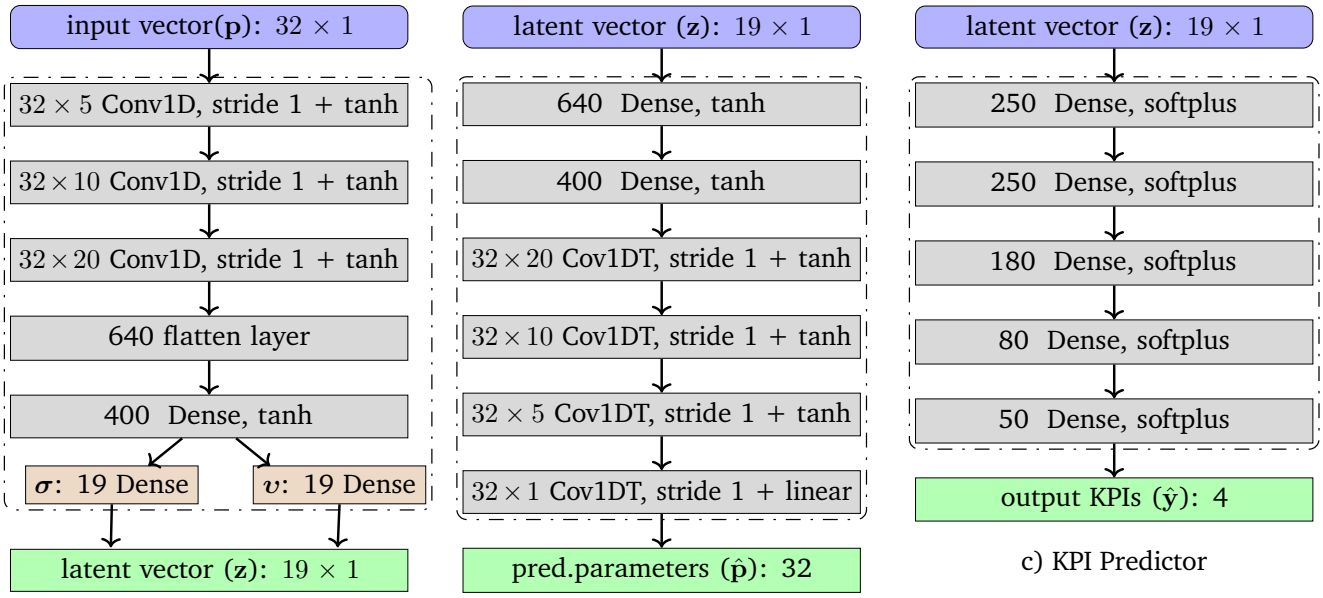


Figure 6.5: Network structure. Figure taken from [154, Fig. 3], © 2022 IEEE.

Training details

The pseudo algorithm for VAE training is described in Algorithm 3. All essential information for training hyperparameters, such as the training-validation-test split, learning rate, validation patience, maximum number of epochs, and batch size, are specified in the algorithm along with their respective values. The distributions for training, test, and validation can be visualized in Figure 6.3. The network training is performed on a NVIDIA Quadro M2000M GPU and takes approximately 12 to 15 minutes. Figure 6.6 illustrates the training and validation curves during the training process. It shows that the network converges around 250 epochs out of a maximum of 300 epochs settings. In the figure, “total loss”refers to the summation of losses from parameter reconstruction and KPIs prediction. Upon completion of training, new samples can be evaluated in approximately 3 to 4 ms per sample, which is significantly faster than FE simulation on a single-core CPU, which takes around 3 to 5 hours per sample.

Algorithm 3 Pseudo-code for training workflow in multi-topology scenario:

$\mathbf{p}_{\text{train}}$, $\mathbf{p}_{\text{validation}}$, $\mathbf{p}_{\text{test}}$: Divide the dataset $\mathbf{D}$
▶ Split the whole dataset $\mathbf{D}$ in training (90%), test (5%), and validation set (5%)
$\mathbf{y}_{\text{train}}$, $\mathbf{y}_{\text{validation}}$, $\mathbf{y}_{\text{test}}$: Get corresponding KPIs ▶ KPIs associated with each design in $\mathbf{D}$
$\mathbf{n}_{\text{epochs}} \coloneqq 300$, $\mathbf{v}_{\text{p, limit}} \coloneqq 10$, $\mathbf{v}_{\text{p, counter}} \coloneqq 0$
▶ Hyperparameter initialization: no of epochs, validation patient (VP) limit and VP counter
$\mathbf{b}_{\text{size}} \coloneqq 40$ ▶ Hyperparameter initialization: batch size
$\mathbf{l}_{\text{rate}} \coloneqq 10^{-3}$ to 10^{-4}
▶ Hyperparameter initialization: start to end learning rate scheduler with 15000 decay steps
$\mathbf{F}_{\Theta}$, $\mathbf{F}_{\Phi}$, $\mathbf{K}_{\theta}$: Initializing trainable parameters in the encoder, decoder, and KPIs predictor.
▶ Glorot uniform initializer [67]
for $e \coloneqq 1$ to $\mathbf{n}_{\text{epochs}}$ **do**
  $\mathbf{p}_{\text{t,shuffle}}, \mathbf{y}_{\text{t,shuffle}} \coloneqq \text{Shuffle}(\mathbf{p}_{\text{train}}, \mathbf{y}_{\text{train}})$ ▶ Shuffle training data randomly at every epoch
  for $i \coloneqq 1$ to $\mathbf{n}_{\text{iter}}$ **do**
    ▶ Compute $\mathbf{n}_{\text{iter}} \coloneqq \left\lceil \frac{\mathbf{n}_{\text{train}}}{\mathbf{b}_{\text{size}}} \right\rceil$, where $\mathbf{n}_{\text{train}}$ is the number of training samples.
    $\mathbf{p}_{\text{batch}}, \mathbf{y}_{\text{batch}} \coloneqq \text{getBatchOfData}(\mathbf{p}_{\text{t,shuffle}}, \mathbf{y}_{\text{t,shuffle}}, \mathbf{b}_{\text{size}}, i)$
    ▶ Get current batch
    $(\boldsymbol{\upsilon}_{\text{batch}}, \boldsymbol{\sigma}_{\text{batch}}) \coloneqq \mathbf{F}_{\Theta}(\mathbf{p}_{\text{batch}})$
    $\boldsymbol{\varepsilon}_{\text{batch}} \sim \mathcal{N}(0, \mathbf{I})$
    $\mathbf{z}_{\text{batch}} \coloneqq \boldsymbol{\upsilon}_{\text{batch}} + \boldsymbol{\sigma}_{\text{batch}} \odot \boldsymbol{\varepsilon}_{\text{batch}}$ ▶ $\mathbf{z} \coloneqq 19$ as $d_2 \coloneqq 18$, and see Figure 6.4)
    $\hat{\mathbf{p}}_{\text{batch}} \coloneqq \mathbf{F}_{\Phi}(\mathbf{z}_{\text{batch}})$ ▶ Predict parameters for the current batch
    $\hat{\mathbf{y}}_{\text{batch}} \coloneqq \mathbf{K}_{\theta}(\mathbf{z}_{\text{batch}})$ ▶ Predict KPIs for the current batch
    $\mathcal{D}_{\text{batch,KL}} \coloneqq -\frac{1}{2} \sum_{j=1}^{\mathbf{b}_{\text{size}}} \left(1 + \log((\boldsymbol{\sigma}_{\text{batch}}^{(j)})^2) - \boldsymbol{\upsilon}_{\text{batch}}^{(j)} - (\boldsymbol{\sigma}_{\text{batch}}^{(j)})^2\right)$
    ▶ KL divergence for the encoder, calculated as described in Appendix B of [103].
    $\mathbf{L}_{\text{batch,rec}} \coloneqq \frac{1}{\mathbf{b}_{\text{size}}} \sum_{j=1}^{\mathbf{b}_{\text{size}}} \|\mathbf{p}_{\text{batch}}^{(j)} - \hat{\mathbf{p}}_{\text{batch}}^{(j)}\|^2$ ▶ MSE loss for decoder
    $\mathbf{L}_{\text{batch,KPI}} \coloneqq \frac{1}{\mathbf{b}_{\text{size}}} \sum_{j=1}^{\mathbf{b}_{\text{size}}} \|\mathbf{y}_{\text{batch}}^{(j)} - \hat{\mathbf{y}}_{\text{batch}}^{(j)}\|^2$ ▶ MSE loss for KPI predictor
    $\nabla \mathbf{F}_{\Theta} \coloneqq \frac{\partial \mathcal{D}_{\text{batch,KL}}}{\partial \phi}$ ▶ Compute gradients using backpropagation algorithm [119]
    $\nabla \mathbf{F}_{\Phi} \coloneqq \frac{\partial \mathbf{L}_{\text{batch,rec}}}{\partial \theta}$ ▶ Compute gradients using backpropagation algorithm [119]
    $\nabla \mathbf{K}_{\theta} \coloneqq \frac{\partial \mathbf{L}_{\text{batch,KPI}}}{\partial \beta}$ ▶ Compute gradients using backpropagation algorithm [119]
    Θ, Φ, θ : Update training parameters using gradients $\nabla \mathbf{F}_{\Theta}, \nabla \mathbf{F}_{\Phi}, \nabla \mathbf{K}_{\theta}$ with Adam [104]
  end for
  Update $\mathbf{v}_{\text{p, counter}}$
  if $\mathbf{v}_{\text{p, counter}} \geq \mathbf{v}_{\text{p, limit}}$ **then**
    Return Θ, Φ, θ: Network training completed
  end if
  Continue training
end for
Return Θ, Φ, θ: Network training completed

6.3.3 Numerical results

The performance of the meta-model for KPI prediction and parameter reconstruction is evaluated using four evaluation metrics: MAE, RMSE, PCC, and MRE. Table 6.1 provides evaluation details of four common

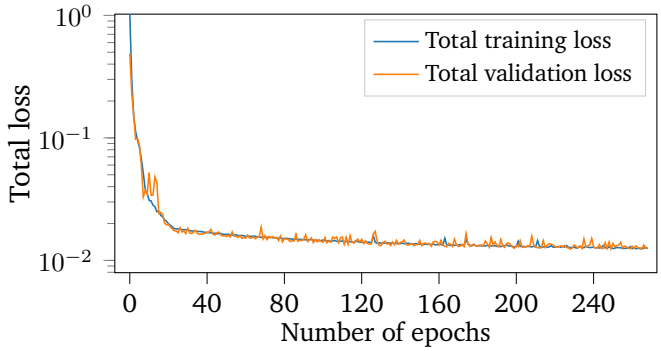


Figure 6.6: Training and validation loss curves. Figure taken from [154, Fig. 4], © 2022 IEEE.

KPIs over the test samples, while Figure 6.7 depicts the normalized prediction plots for the same. It can be observed that the meta-model performs poorly in predicting the KPI maximum torque ripple (y_3) compared to other KPIs, with the MAE of 3.52 Nm and the MRE of 4.05%. The average MAE, RMSE, and MRE across all four KPIs are 2.26, 4, and 1.43%, respectively, which can be considered reasonable.

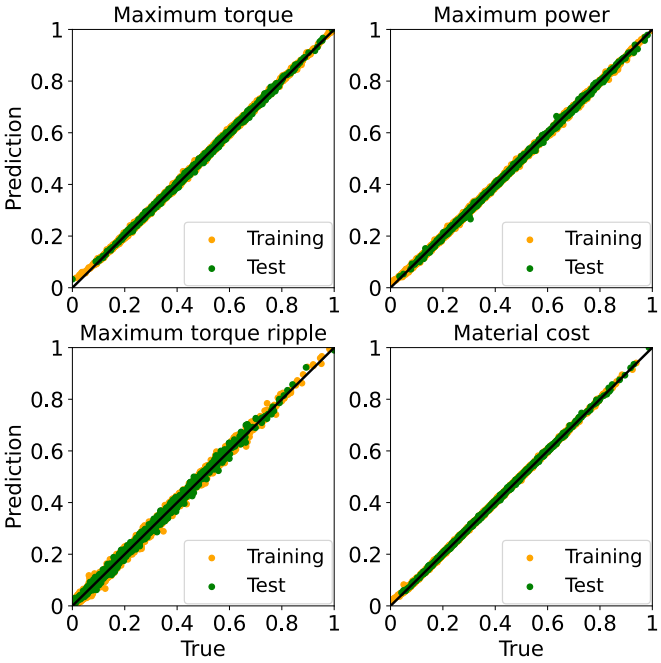


Figure 6.7: KPIs prediction plots for test samples. Figure taken from [154, Fig. 6], © 2022 IEEE.

Similar to KPIs, the evaluation for a few SV and DV parameters reconstruction is illustrated in Table 6.2 and Table 6.3, respectively. Figure 8.4 displays prediction plots for these parameters for training and test samples. It is reported that the reconstruction of the remaining parameters is also achieved with high precision. The prediction plots for them are depicted in Figure 8.5. The average MAE, RMSE, and MRE with all parameters is reported $\leq 1\%$.

Individual DNNs for SV and DV topology datasets were also trained with identical network structures

Table 6.1: Evaluation of KPIs. Table taken from [154, Tab. 1], © 2022 IEEE.

	KPIs	Unit	Prediction accuracy			
			$\overline{\varepsilon}_{\text{mae}}$	$\overline{\varepsilon}_{\text{rmse}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$
y_1	Maximum torque	Nm	2.45	3.15	0.99	0.55
y_2	Maximum power	kW	1.64	2.2	0.99	0.49
y_3	Maximum torque ripple	Nm	3.52	5.01	0.99	4.05
y_4	Material cost	Euro	1.43	1.8	0.99	0.64

(Figure 6.5c), training hyperparameters, and train-validation-test sets. The only difference is the input layer, which corresponds to the number of input dimensions of SV ($d_1 = 13$) and DV ($d_2 = 18$) for each individual DNN. The training for each DNN was followed as explained in Algorithm 2. The idea was to compare the prediction performance of the VAE-based and scalar DNN-based meta-models at a shallow level. The VAE showed better prediction performance for all four KPIs in this train-validation-test set; see Figure 6.8. One possible reason for this could be the larger number of training samples available for the VAE-based meta-model compared to the separately trained DNN for each topology. Overall, both meta-models, the VAE-based and individually trained DNN, exhibit similar performance. For a more rigorous comparison, considerations such as varying training sizes (as shown in Figure 5.9) and performing hyperparameter tuning for individually trained DNN should be taken into account. However, since this is not the main aim of this study, the analysis is limited to a preliminary comparison.

Table 6.2: Evaluation of SV parameters reconstruction. Table taken from [154, Tab. 2], © 2022 IEEE.

	Parameters	Reconstruction accuracy			
		$\overline{\varepsilon}_{\text{mae}}$	$\overline{\varepsilon}_{\text{rmse}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$
p_1	Air gap	0.009	0.01	0.99	0.65
p_2	Height of magnet	0.01	0.012	1	0.18
p_3	Angle of magnet	0.146	0.175	0.99	0.71
p_4	Iron length	0.036	0.047	1	0.23
p_5	Rotor outer diameter	0.017	0.019	1	0.28
p_6	Stator tooth height	0.006	0.006	0.99	0.39

Table 6.3: Evaluation of DV parameters reconstruction. Table taken from [154, Tab. 3], © 2022 IEEE.

	Parameters	Reconstruction accuracy			
		$\overline{\varepsilon}_{\text{mae}}$	$\overline{\varepsilon}_{\text{rmse}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$
p_1	Air gap	0.004	0.006	1	0.37
p_2	Height of magnet 2	0.009	0.011	1	0.19
p_3	Angle of magnet layer 2	0.173	0.188	0.99	0.67
p_4	Iron length	0.017	0.021	1	0.31
p_5	Rotor outer diameter	0.01	0.012	1	0.53
p_6	Angle of magnet layer 1	0.167	0.191	0.99	0.56
p_7	Height of magnet 1	0.022	0.024	0.99	0.39
p_8	Stator tooth height	0.002	0.003	0.99	0.19

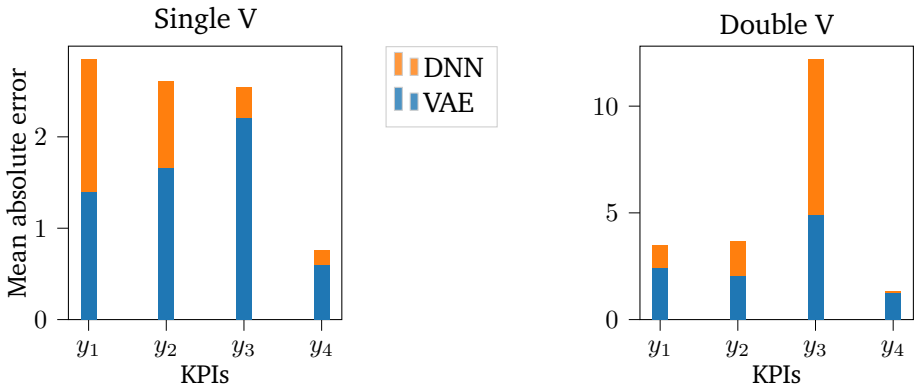


Figure 6.8: Performance comparison between VAE and individually trained DNNs. Figure taken from [154, Fig. 8], © 2022 IEEE.

Multi-objective optimization

The proposed approach is demonstrated by applying it to the MOO problem (refer to Sec. 4.2) for $T = 2$ topologies. The general diagram of the population based evolutionary algorithm is explained in Sec. 2.3.2.2. Equations (4.1-4.3) are solved using the multi-objective genetic algorithm NSGA-II [44]. The proposed MOO workflow can be visualized in Figure 6.9. The KPI predictor is used to evaluate samples generated by the latent distribution. The decoder functions as a design predictor. The MOO is conducted for two

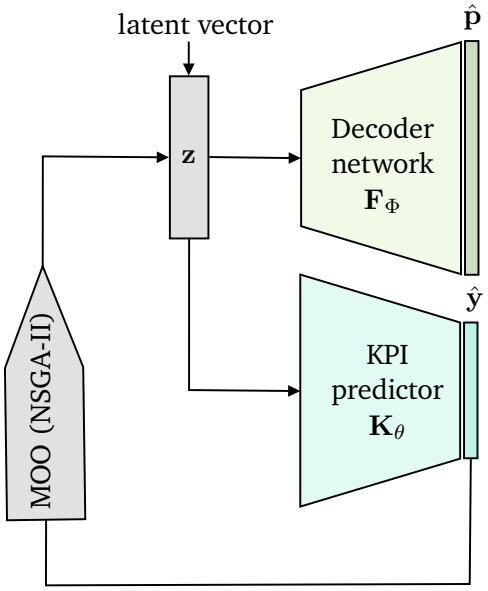


Figure 6.9: MOO Workflow.

contrasting KPIs: material cost and maximum power. The optimization hyperparameters are set based on experience, as provided in Table 6.4. The MOO is constrained by parametric upper and lower bounds defined for data generation to reduce the number of geometric infeasible designs. The latent input bound is decided from the mean values (υ) of the network's training samples. This restriction helps to confine the search within the learned latent distribution, which is also significant for producing valid designs.

Table 6.4: MOO hyperparameter settings. Table based on [151, Tab. 9].

Settings	Value
Sampling approach	random initialization
Population per generation	1000
Stopping point	maximum 100 generations
Stagnation generations	20
Number of objectives	2
Crossover, mutation probability	0.9

The optimization finishes roughly in 2 to 2.5 hours. The final Pareto front, along with network training samples, is depicted in Figure 6.10. The red circles represent SV samples, and the blue circles represent DV samples. By simultaneously plotting the Pareto front and network training samples, it is observed that the Pareto front comprises new designs that are not present in the training dataset. For demonstration purposes, two designs are randomly chosen, one for SV (A) and another for DV (B) from Figure 6.10. The geometries of both designs can be seen in Figure 6.2, and the evaluation of all three KPIs is presented in Table 6.5. The KPI y_3 has the lowest prediction accuracy among all KPIs for both designs, with relative errors (REs) of 8.79% and 6.8%, respectively. However, the average MRE for all KPIs is less than 5% for both samples. The design validation percentage is reported to be approximately 40% after re-simulating twenty designs for each topology from the Pareto front. It is observed that there are some feasible designs from a simulation perspective, for which the KPI predictions did not match the actual values. This discrepancy was mainly due to a non-synchronization issue between the KPI predictor and the decoder. The problem arises because the actual design vector contains additional zeros, whereas the decoder predicts continuous values and does not precisely predict exact zero values but rather values around the range of 10^{-3} to 10^{-4}. This discrepancy may have an adverse impact on KPI predictions. To address this issue, a new optimization procedure is proposed, which will be explained in the next section when dealing with the multi-technology scenario.

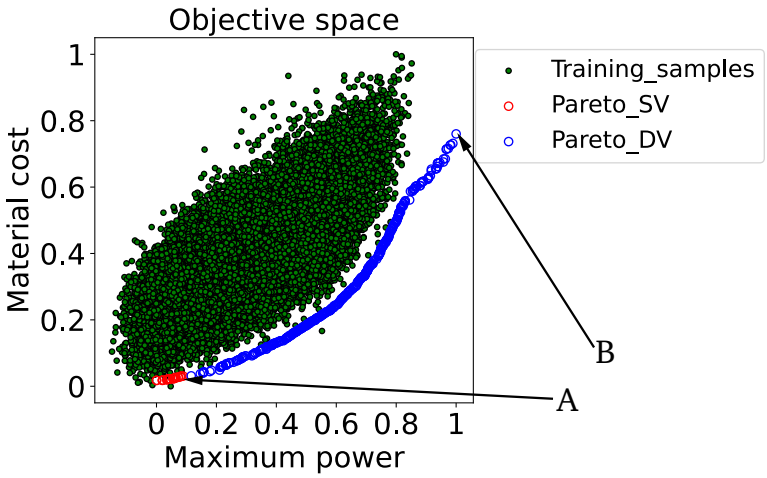


Figure 6.10: Pareto-front: Maximum Power and Material cost for SV (Red) and DV (Blue) topologies. Figure taken from [154, Fig. 9], © 2022 IEEE.

Table 6.5: Evaluation of two Pareto designs (see Figure 6.10 and Figure 6.2). Table taken from [154, Tab. 4], © 2022 IEEE.

KPIs	Design A (SV)			Design B (DV)		
	FE simulation	Prediction	RE(%)	FE simulation	Prediction	RE(%)
y_1	351.86	346.79	1.44	489.36	470.99	3.75
y_2	284.34	280.97	1.18	578.87	600.96	3.8
y_3	29.34	31.92	8.79	232.95	216.97	6.8
y_4	131.8	133.98	1.65	301.31	308.71	2.4

6.4 Scenario 2: Heterogeneous parameterization by machine technology

In this section, the proposed approach is applied to two distinct machine technologies, namely ASM and PMSM. Both machines operate on different working principles. The ASM, which is a type of IM, uses the principle of electromagnetic induction. In this operation, stator windings supplied with AC power generate a rotating magnetic field. This induces an electrical current in the rotor windings, creating a magnetic field that causes the rotor to rotate. In PMSM, the stator windings generate a magnetic field through AC power supply, interacting with the permanent magnets on the rotor to cause the rotation. The primary distinction between ASM and PMSM lies in their methods of creating the rotor's magnetic field. This may impact the machines' efficiency, losses, performance, and cost in different applications. For instance, in EV and HEV applications, the PMSM shows higher efficiency at low speed but incurs higher copper losses at high speed than the IM. The PMSM demonstrates superior power density to the IM. However, the production cost of PMSM is also higher due to the expenses associated with permanent magnets [230].

The remainder section follows the same structure as the previous section and the content is based on [151].

6.4.1 Datasets

Datasets for $T = 2$ technologies are created using the same data generation workflow explained in section 4.3. For the PMSM design, the KPIs are obtained through the magneto-static FE simulation [184], while the ASM design is evaluated using an in-house tool based on analytical calculations (refer to [21, 61]). It should be noted that the time taken for a single ASM design is around $5 - 7$ minutes, whereas for the PMSM, it takes $3 - 5$ hours on a single-core CPU on the HPC cluster. After filtering erroneous designs from the initial population, the number of valid ASM designs remains $N_{\text{ASM}} = 50387$ with $d_1 = 18$, and for the PMSM, the number of valid designs is $N_{\text{PMSM}} = 51532$ with $d_2 = 33$. The difference in the total number of samples between the two datasets is $\leq 3\%$, which is sought to prevent bias toward one machine type during the training process. In this dataset, varying topological parameters, including slots per pole per phase, pole pairs, stator outer diameter, and electrical parameters such as winding connection and winding scheme are considered. The details of all parameters are provided in Table 8.17 and Table 8.12. Representative geometries with annotations of a few parameters are shown in Figure 6.12. The common KPIs under investigation are listed in Table 6.7. Analogous to the previous scenario, the pair wise distribution of a few parameters and target KPIs is plotted in Figure 8.7. Similar to the previous scenario, it is assumed that the

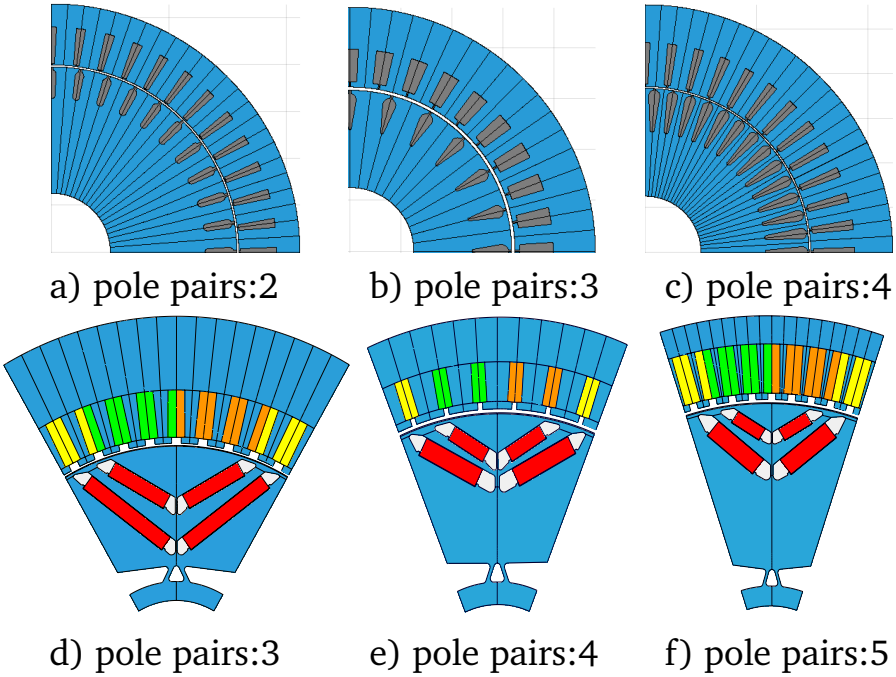


Figure 6.11: Exemplary geometries of ASM (a-c) and PMSM (d-f) with varying pole pairs.

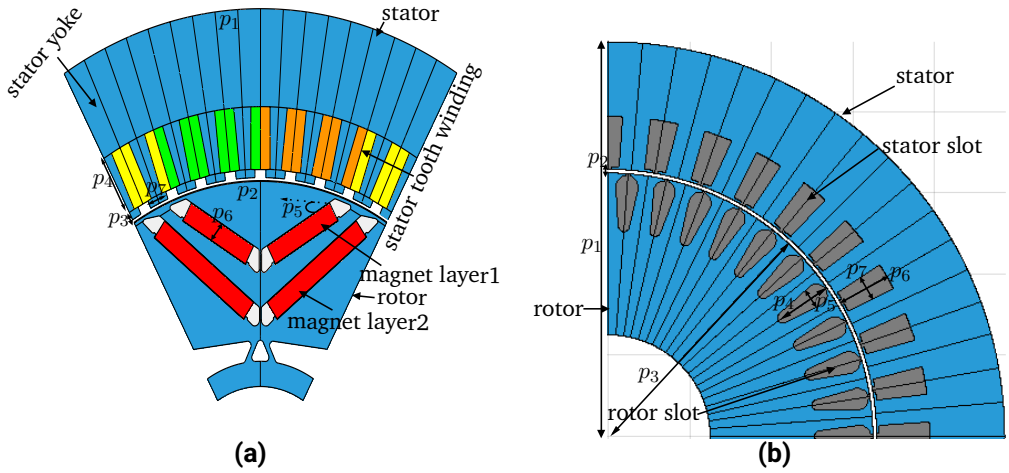


Figure 6.12: Representative samples (a) ASM (b) PMSM. Figure taken from [151, Fig. 1].

system parameters (see Table 8.16) are identical and the cost of common materials, such as copper, iron, and aluminium, is considered to be the same. The primary cost difference arises from the magnets used in the PMSM.

6.4.2 Network architecture and training details

The network architecture and training hyperparameters are determined using a trial and error approach, where roughly twenty configurations are evaluated starting with the base network from Figure 6.5. The final

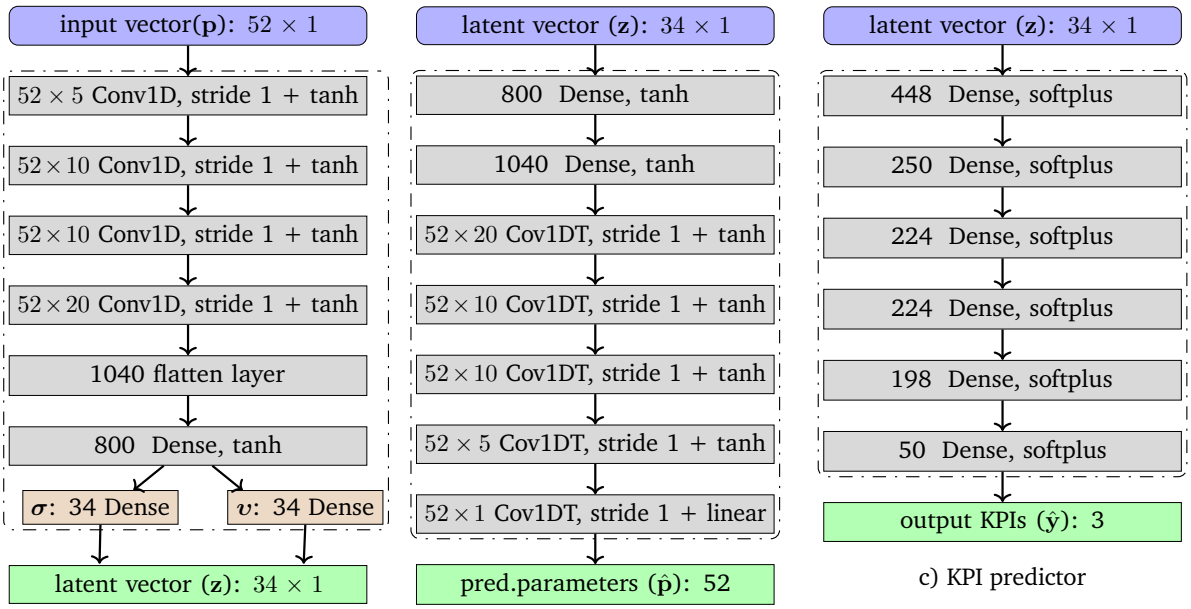


Figure 6.13: Network structure. Figure taken from [151, Fig. 8].

network configuration is illustrated in Figure 6.13. The encoder $\mathbf{F}_\Theta$ consists of four 1D convolutional layers, followed by a flatten layer, a dense layer and three output layers. As explained in Sec. 6.3.2, to capture all necessary features from the integrated design vector, the number of latent dimensions should be set as $n_\mathbf{z} \geq \max_t(d_t)$ from all the machine types. In the dataset, the PMSM has the maximum number of input dimensions, $d_2 = 33$. Therefore, the number of latent dimension $n_\mathbf{z}$ is set to 34, including an additional technology indicator parameter. The decoder (design predictor), similar to the previous scenario, maintains the mirror structure of the encoder with an output linear layer. The KPI predictor comprises an input layer with the number of neurons equal to $n_\mathbf{z}$, five dense layers with softplus activation function, and an output layer with the same number of neurons as the number of common KPIs.

Table 6.6: Training hyperparameters detail. Table taken from [151, Tab. 5].

Parameters	Value
Adaptive learning rate	10^{-4}-10^{-5}
Activation functions	tanh, softplus
Maximum number of training epochs	300
Validation patience	20
Optimizer	Adam [104]
Latent space dimension	34 (as $d_{\text{PMSM}} := 33$)
Loss functions	KL-divergence and MSE (see equation 6.6)
Batch size	50

Training details

The final training, test, and validation sets contain 80%, 10%, and 10% of the total combined dataset, respectively. The pairwise distribution of all final sets is depicted in Figure 8.7 for a few parameters and all target KPIs.The meta-model training was run on an NVIDIA Quadro M2000M GPU and took roughly 1.5 hours. The same training steps were followed as described in Algorithm 3. The training hyperparameter details are given in Table 6.6. Training and validation loss curves, combining losses of parameter reconstruction and KPIs prediction, are depicted in Figure 8.6. Additionally, two separate DNNs are trained, one for each technology dataset, using identical network structures (except input layer; see Figure 6.13c, training hyperparameters and training-validation-test sets, following the training loop explained in Algorithm 2. The training time for each individual DNN was around 15 minutes.

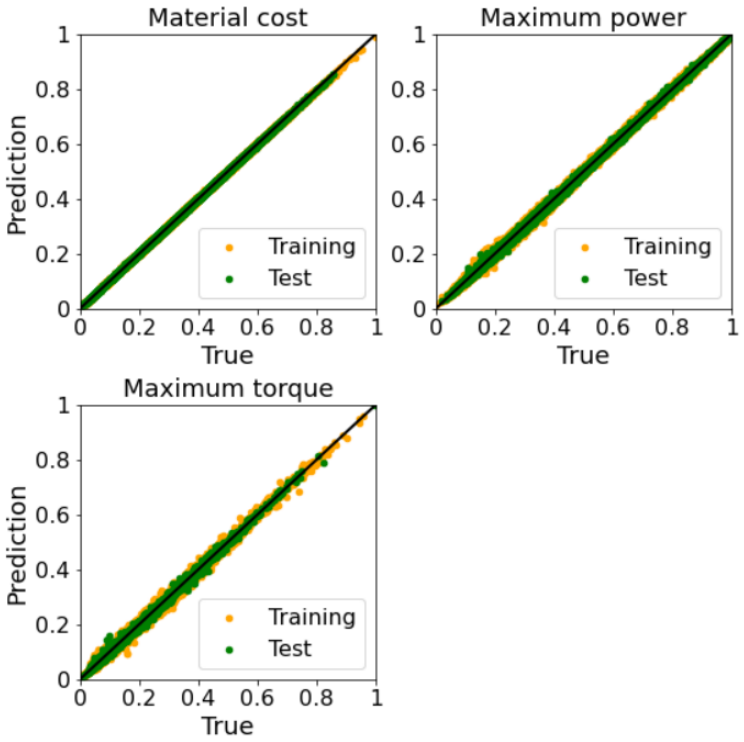


Figure 6.14: KPIs prediction plots. Figure taken from [151, Fig. 10].

Table 6.7: Evaluation of KPIs. Table taken from [151, Tab. 6].

	KPIs	Unit	Prediction accuracy			
			$\overline{\varepsilon}_{\text{mae}}$	$\overline{\varepsilon}_{\text{rmse}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$
y_1	Material cost	Euro	0.43	0.53	1	0.71
y_2	Maximum power	kW	1.90	2.54	1	1.31
y_3	Maximum torque	Nm	3.96	6.52	0.99	1.76

6.4.3 Numerical results

The evaluation of common KPIs with the VAE-based meta-model over the test set is presented in Table 6.7, using MAE, PCC, and MRE evaluation metrics. Among all KPIs, the maximum torque exhibits the worst prediction accuracy, with the MRE of 1.76%, the RMSE of 6.52, and the MAE of 3.96 Nm. The normalized prediction plots for test and training samples are depicted in Figure 6.14. Figure 8.8a and Figure 8.8b show

Table 6.8: Evaluation of ASM parameters reconstruction. Table taken from [151, Tab. 7].

	Parameters	**Reconstruction accuracy**			
		$\overline{\varepsilon}_{\text{mae}}$	$\overline{\varepsilon}_{\text{rmse}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$
p_1	Stator outer diameter	0.39	0.483	0.99	0.19
p_2	Air gap	0.002	0.005	1	0.37
p_5	Rotor outer diameter	0.55	0.636	0.99	0.41
p_4	Rotor slot height	0.05	0.069	0.99	0.37
p_5	Rotor slot width	0.004	0.005	1	0.4
p_6	Stator slot height	0.05	0.004	1	0.32

Table 6.9: Evaluation of PMSM parameters reconstruction. Table taken from [151, Tab. 8].

	Parameters	**Reconstruction accuracy**			
		$\overline{\varepsilon}_{\text{mae}}$	$\overline{\varepsilon}_{\text{rmse}}$	ε_{pcc}	$\overline{\varepsilon}_{\text{mre}}$
p_1	Stator outer diameter	0.51	0.63	0.99	0.26
p_2	Rotor outer diameter	0.39	0.5	0.99	0.27
p_3	Air gap	0.006	0.007	1	0.41
p_4	Stator tooth height	0.054	0.072	0.99	0.37
p_5	Angle magnet layer 1	0.11	0.14	0.99	0.48
p_6	Height of magnet layer 1	0.012	0.015	1	0.27

prediction plots for parameter reconstruction over the training and test sets for each machine technology for a few parameters. The statistical evaluation for these parameters is given in Table 6.8 and Table 6.9. It can be observed that parameters reconstruction is achieved with high accuracy. The average MRE, RMSE, and MAE for all parameters are reported to be less than 1%. The remaining parameters listed in Table 8.17 and Table 8.12 are also considered in this analysis. The prediction plots for them can be seen in Figure 8.10 and Figure 8.9.

Figure 6.15 illustrates a shallow comparison between separately trained DNN and VAE-based KPI predictor over the test samples. The VAE and DNNs have almost the same MAE for KPI y_1 for both machine types. The VAE has a lower MAE for KPIs y_2 and y_3 than DNNs. Overall, it can be seen that the VAE has slightly better KPI prediction accuracy.

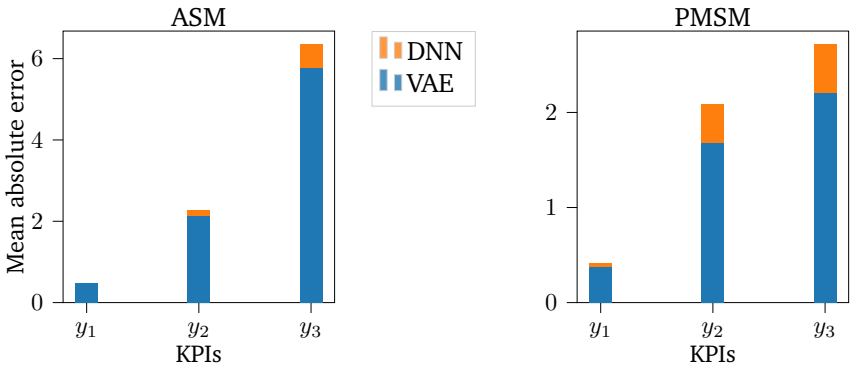


Figure 6.15: Comparison of VAE and individually trained DNN. Figure taken from [151, Fig. 13].

Multi-objective optimization

Similar to the previous scenario in Sec. 6.3, the trained meta-models are applied for the MOO to solve equations (4.1-4.3). However, in this scenario, challenging discrete parameters such as pole pairs, winding connection, slots per pole per phase, etc., are addressed. In the multi-topology scenario, non-synchronization was observed between the KPI predictor and the decoder during the MOO. This issue may be caused by the combined design vector's sparsity (additional zeros). To prevent this problem and handle discrete parameters, a new optimization workflow as displayed in Figure 6.16 is proposed. In the proposed workflow,

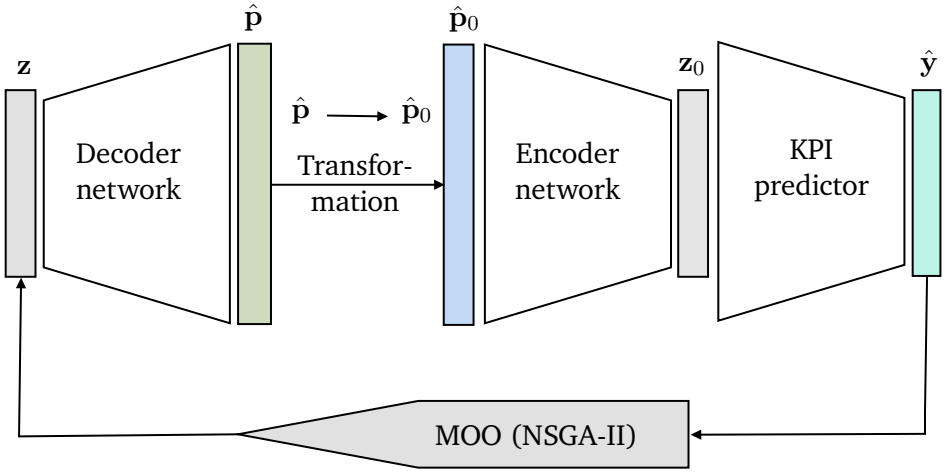


Figure 6.16: Proposed VAE-based optimization workflow. Figure taken from [151, Fig. 6].

first, design parameters are predicted using the decoder with the input of a randomly generated latent vector $\mathbf{z}$ from the latent bounds. Then, the predicted continuous values are replaced with zeros and actual discrete parameter values at positions in the design vector where they should be zero or known discrete values from prior knowledge during the pre-processing of the combined dataset. This transforms predicted design vector $\hat{\mathbf{p}}$ to $\hat{\mathbf{p}}_0$. Afterward, the processed vector $\hat{\mathbf{p}}_0$ is passed to the encoder network, which maps it to a latent vector $\mathbf{z}_0$. This latent vector is then fed as input to the KPI predictor to predict KPI vector $\mathbf{y}$. This procedure ensures synchronization between the KPI predictor and the decoder.

Similar to the multi-topology scenario, the concurrent multi-technology MOO is conducted for two contrasting KPIs: maximum power and material cost, using the genetic algorithm NSGA-II[44]. The rest of the MOO

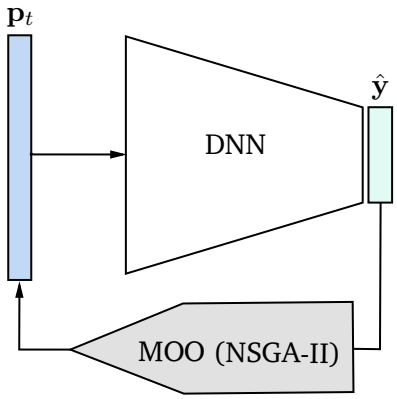


Figure 6.17: Individual DNN based MOO workflow. Figure based on [151, Fig. 7].

procedure, which includes the settings of the MOO hyperparameters (refer to Table 6.4) and the way for determining the parametric and latent bounds, is kept identical to the multi-topology scenario. The MOO is also executed by utilizing separately trained DNNs with each machine technology dataset. The MOO workflow for that is illustrated in Figure 6.17. For this MOO, the input is the actual parameter vector $\mathbf{p}_t$ concerning t technology. Since $T = 2$, the MOO is run for two individually trained scalar DNN-based meta-models. It should be noted that the optimization settings were kept identical during all three MOO runs: one for the VAE-based concurrent MOO and two for the individual machine technology MOO with separately trained DNNs. The VAE-based concurrent MOO takes approximately 2.5 hours, whereas individual machine technology MOO requires roughly 40-50 minutes. The various Pareto fronts concerning the VAE-based concurrent approach, the individually trained DNN, and the training Pareto fronts are displayed in Figure 6.18, along with training samples. It can be seen that both the VAE- and DNN-based Pareto fronts exhibit cost- and power-efficient designs, which are not present in the training data. Two Pareto designs

Table 6.10: Design evaluation from VAE Pareto front. Table taken from [151, Tab. 10].

KPIs	Design A (ASM)			Design B (PMSM)		
	FE simulation	Prediction	RE(%)	FE simulation	Prediction	RE(%)
y_1	45.47	46.27	1.75	153.53	153.49	0.026
y_2	241.7	237.82	1.60	402.98	406.70	0.92
y_3	195.46	187.52	4.06	294.61	286.63	2.70

Table 6.11: Pareto designs from individually trained DNN. Table taken from [151, Tab. 11].

KPIs	Design C (ASM)			Design D (PMSM)		
	FE simulation	Prediction	RE(%)	FE simulation	Prediction	RE(%)
y_1	46.68	46.73	0.53	130.81	129.51	0.99
y_2	227.48	235	3.3	401.8	404	0.54
y_3	158	200.36	26.81	313.05	316.27	1.02

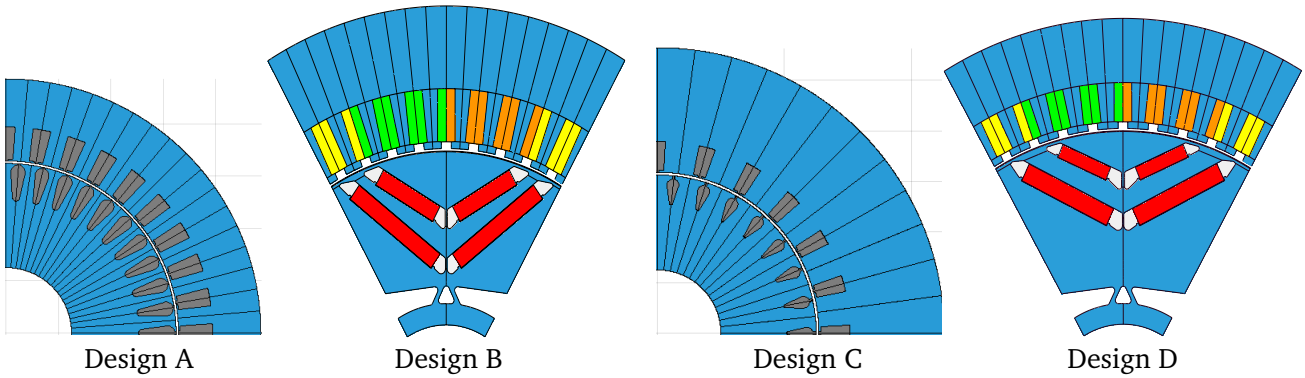


Figure 6.18: Pareto designs. Figure taken from [151, Fig. 14].

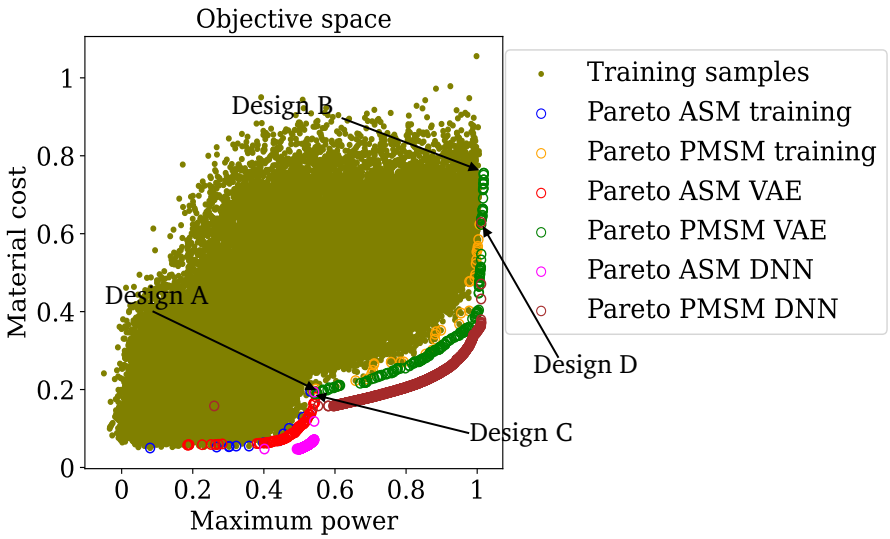


Figure 6.19: Pareto fronts for Material cost and Maximum power are presented. The Pareto front of ASM training samples is depicted in blue, the Pareto front of PMSM training samples is shown in orange, and meta-model training samples are represented in olive. Pareto fronts for the VAE-based method are displayed in green (ASM) and red (PMSM), while Pareto fronts for the individually trained DNNs are shown in brown (PMSM) and magenta (ASM). Figure taken from [151, Fig. 15].

from each Pareto front, concerning each technology, are randomly chosen for demonstration and depicted in Figure 6.18. In Figure 6.19, arrows highlight four designs on their respective Pareto fronts. The first two designs, Design A (ASM) and Design B (PMSM), belong to the VAE-based MOO Pareto front, whereas the remaining two designs, Design C (ASM) and Design D (PMSM), are from the separately trained DNN models. All four designs are also simulated by the FE simulation to measure the proximity of predicted KPIs to the true KPIs. The evaluation of Design A and Design B concerning all three KPIs is provided in Table 6.10. Design A (ASM) exhibits poor prediction accuracy for the KPI maximum torque with the RE of 4.06%. Overall, the meta-model predictions are reasonably accurate for both designs. The evaluation of Design C (ASM) and D (PMSM) is provided in Table 6.11. It can be seen that Design C has a poor prediction for the

KPI maximum torque with a RE of 26.81%, possibly due to a weak functional mapping between the input and the maximum torque output for that sample. The individual DNN-based approach displays a Pareto front with a greater number of power and cost-efficient designs, as illustrated in Figure 6.19. However, around twenty designs from both the VAE-based and DNN-based Pareto fronts were manually inspected. It was observed that the Pareto front obtained with the DNN-based approach produces a higher number of geometrically infeasible designs, including the region consisting of power and cost-efficient designs. Approximately $\sim 60\%$ more invalid designs are reported compared to the VAE-based Pareto region. Even if a design is valid, there may be a poor functional mapping between the input vector and actual KPIs (e.g., Design D with the RE of 26.81% for the maximum torque KPI). This issue arises when the design does not belong to the training distribution. To improve this situation, considering the same geometry checks during the MOO process that were used for data generation would be beneficial. Doing so may implicitly force the optimizer to sample designs from the region of the training distribution. In the MOO process, the same parametric bounds of the data generation are applied as constraints. Hence, designs are sampled from these parametric bounds, but not from the actual training distribution. If the design comes from the training distribution, then there is a better functional mapping between input and true KPIs, while if the design falls outside the training distribution, the DNN (trained with supervised learning) cannot predict reasonably accurate KPIs for that design due to its inability to extrapolate beyond the training distribution [95]. This problem is less present with the VAE-based concurrent approach since the latent bounds are determined from the mean values of training samples. As a result, the design search space is automatically restricted to the input training distribution. Consequently, during the MOO, more valid output samples are produced, and their predictions are closer to true KPIs.

6.5 Summary

In this chapter, the VAE-based approach is introduced to create a unified parameterization for two different machine technologies and PMSM rotor topologies. The KPI predictor (DNN) is trained with a common set of KPIs, using the latent input in conjunction with the encoder and the decoder networks. The prediction accuracy of the proposed network configuration depends on two network hyperparameters: 1D convolutional layers, which are important for learning essential features from the high-dimensional sparse input vector, and the number of latent dimensions, which must be equal to or greater than the maximum input dimension topology ($n_{\mathbf{z}} \geq \max_t(d_t)$), for precise parameter reconstruction. The numerical results display high accuracy in the prediction of KPIs and parameter reconstruction across a complex design space over the test samples, thereby enabling the possibility of concurrent optimization of multiple electric machine technologies and rotor topologies via a single meta-model training.

The proposed VAE-based approach is then demonstrated for MOO in multi-topology and multi-technology scenarios. An enhanced optimization workflow is presented (Figure 6.16) to handle sparsity and discrete parameters in the input design vector while maintaining synchronization between the KPI predictor and the decoder network. This setup implicitly leads the optimizer to sample designs from the input latent distribution during MOO.

A high-level comparative analysis is conducted between the VAE- and individually-trained DNN-based approaches. The VAE-based meta-model demonstrates high prediction accuracy. The numerical results for MOO show that individually trained DNNs yield a Pareto region with power- and cost-efficient designs, albeit with many being invalid. However, the VAE-based workflow produces more valid, geometrically consistent designs, maintaining synchronization between the KPI predictor and the decoder. Both VAE-based latent space optimization and individually trained DNN-based optimization improve upon the training data and

generate new designs that are absent in the training set. A proportional rise in computational time during optimization for multiple machine types can be anticipated when the DNN-based models are trained individually. In contrast, only a slight increase in computational time is anticipated with the VAE-based approach.

In conclusion, the proposed VAE-based approach lays a foundation for concurrent parametric multi-technology and topology optimization using a unified latent space in the domain of rotating electrical machines.

7 Conclusion and Future work

7.1 Conclusion

In this research, the primary objective was to investigate the application of modern deep learning algorithms to expedite performance analysis and, consequently, parametric optimization of electrical machines during the design phase. Both supervised and unsupervised learning methodologies were employed.

The challenge in multi-objective optimization arises from its intense computational demands, especially when dealing with a large number of multi-domain KPIs and input parameters. The conventional FEM-based approach for navigating such high-dimensional design spaces is notably time-consuming. Therefore, the initial goal was to analyze the accuracy of data-driven models in approximating a large number of cross-domain KPIs for different datasets of PMSMs within a high-dimensional space. Three meta-models were proposed (Chapter 4), each with different input representations: a scalar parameter-based, an image-based, and a combination of both. All meta-models were trained using supervised learning. Numerical results indicated that the scalar parameter-based meta-model was more accurate and computationally efficient compared to its image-based and combined input counterparts. However, image-based models are more flexible in scenarios like reparameterization. With these models, no new training is required because the image space remains unchanged, which is not the case with scalar parameter-based models. On the other hand, scalar parameter-based models can more easily incorporate parameters (e.g., stack length, current) that are not visible in 2D images. During the pixel resolution study with image-based models, it was observed that higher resolution resulted in better prediction accuracy, albeit at the cost of computational effort. For some KPIs, image-based models performed comparably to scalar parameter-based models when the image resolution was sufficiently large and the number of training samples was low. One significant limitation of data-driven models is their dependency on final KPIs, which implicitly rely on fixed values of electrical drive system parameters. Consequently, any changes in these parameters necessitate the retraining of the data-driven models. Furthermore, data-driven models do not incorporate any knowledge of physical laws while making predictions.

To address the limitations mentioned above with data-driven models, the generalized hybrid approach was introduced that integrates physics-based models with data-driven models to quantify the performance of PMSMs (Chapter 5). The hybrid approach demonstrated superior prediction accuracy at the expense of computational efficiency but still takes much less time than the conventional FEM for evaluating new designs. The superior performance can be attributed primarily to two factors. Firstly, approximating the functional relationship between input design parameters and a few independent time steps of intermediate measures is anticipated to be more effective and easier as compared to dealing with many cross-domain KPIs. Secondly, the post-processing allows the use of physical laws, and thus, it confirms that KPIs are calculated with proper constraints. With the hybrid approach, complex performance measures such as efficiency maps and various performance curves (e.g., maximum torque limit curve) can be calculated with sufficient accuracy without separate meta-model training, thereby enabling their inclusion in the MOO.

The application of the hybrid approach for MOO was demonstrated in an industrial workflow, in parallel to the conventional FE-based optimization using a commercially implemented population-based evolutionary algorithm. The numerical results showed that the hybrid approach yielded Pareto solutions close to those obtained through the FE-based optimization, while being computationally eight times faster. This computational efficiency also allows for a more precise exploration of the optimum region by evaluating more designs without incurring additional costs. It thus can lead to the possibility of finding better design options closer to the optimum. It was shown that the hybrid model can be applied in the subsequent optimization run in a reduced design space. Although the numerical results indicated a slight decline in prediction performance, it remained within an acceptable range, making the hybrid approach flexible for further optimization without the need for additional meta-model training.

For each machine technology or rotor topology, a separate data-driven model based on scalar parameter input is required due to distinct parameterization. Training these separate models can become both time-consuming and computationally demanding when multiple machines or rotor topologies are involved. To address this problem, a new approach that employs a generative model, specifically the VAE, is proposed to simultaneously optimize heterogeneously parameterized electrical machines (Chapter 6). Numerical results demonstrated that multiple electrical machine technologies (ASM and PMSM) and rotor topologies (SV and DV) could be concurrently optimized using a single meta-model training scheme with a unified latent representation of combined scalar parameter-based input. This was achieved with reasonable prediction accuracy and illustrated the generation of new designs that were not present in the training data.

Despite these advancements, there are limitations and challenges to consider, particularly concerning data generation and model training. Data-driven DL algorithms require a well-distributed dataset that adequately spans the entire design space to prevent issues such as underfitting, overfitting, or biased inference. In the electromagnetic domain, where measurement data collection is both costly and resource-intensive, synthetic data generated through FE solvers was relied upon as the ground truth for supervised DNN training. However, the quality of this synthetic data depends on several factors, such as geometry meshing, convergence criteria, choice of basis and test functions, and boundary conditions. This can potentially lead to biased predictions if not carefully checked, for example, through the higher presence of outliers. Furthermore, data-driven models with supervised learning are inherently limited in their ability to extrapolate beyond their training distribution, resulting in errors when applied to new data points outside of this distribution. Additional challenges include the selection of suitable machine learning algorithms, hyperparameter tuning, and managing complex, high-dimensional design spaces.

In conclusion, this thesis proposes modern deep learning based approaches to accelerate the optimization of electrical machines. The proposed methods enable efficient exploration of the high-dimensional design space for identifying regions of interest while drastically reducing computational costs with reasonable accuracy. All the proposed approaches in this treatise can be applied to other domains, such as structural engineering, aerospace, and renewable energy, where time-consuming computer simulations are at the core of replicating and understanding complex physical behaviors.

7.2 Future work

Several interesting directions for future research could be considered. The datasets examined for the hybrid approach comprise designs with magnetic state symmetry and do not include additional harmonic losses. Future research could explore the effectiveness of the hybrid approach using more complex datasets that account for both asymmetry in the magnetic state and additional harmonic losses.

A large amount of simulation data is required for the data-driven part in the hybrid approach. However, there may be real-world situations where simulation data are scarce. In such cases, physics-informed neural networks or their variants can potentially be employed for solving magneto-static problems in a parameterized form, accommodating varying geometries of electrical machines. While this approach has been explored for simple 2D parameterized magnetostatic problems in [9], extending it to more complex geometries, such as rotating electrical machines, remains an open question.

The proposed VAE-based framework is investigated for concurrent optimization of two machine technologies and topologies and employs a standard Gaussian prior in the latent space. Future work could examine the scalability of this approach for more machine types and the utility of other priors, such as Gaussian mixture for multi-modal distributions.

Since the VAE-based approach is data-driven, it lacks utilizing knowledge of physical laws for KPIs estimation. Therefore, examining the combination of the VAE-based approach with the hybrid method could be a possible research direction for improving prediction accuracy and generalization.

The investigation with other generative models in the literature, such as GANs, will be interesting future work to see how they perform or can complement the current VAE-based approach. In a recent study [77], the GAN-based workflow was already proposed for electrical machine optimization using image-based models, inspired by our work presented in [152, 154].

Exploring these avenues may make the DL-based approaches more adaptive and generalizable, further bridging the gap between academic research and industrial application.

8 Appendix

8.1 Software details

In this thesis:

- The training pipeline and HPO of all metamodels for Chapter 4 and Chapter 5 were carried out using TensorFlow [1] version 2.3.0. The Python version employed was 3.7.9.
- For Chapter 6, the training pipeline and HPO were performed using TensorFlow version 2.5.0 and Python version 3.8.2. This version was specifically chosen because 1D convolutional layers were not available in earlier versions of TensorFlow.
- All datasets were generated using an in-house multi-domain objective toolchain (developed by Robert Bosch). This toolchain was created in MATLAB R2019b and Python 3.

8.2 Datasets detail and numerical results

In this section, we present the full details of the datasets and some numerical results corresponding to Chapters 4 to 6. This is divided into three subsections, with each subsection providing specific dataset details and a few numerical results related to the respective chapters.

Table 8.1: Constant parameters. Table taken from [152, Tab. 2].

Parameter	Dataset 1	Dataset 2	Unit
No of pole pairs	4	4	-
Stator type	Asymmetric	Asymmetric	-
Rotor type	VC-Design	VC-Design	-
No of slots (stator)	48	48	-
Max. phase voltage	640	640	V
Max. phase current	480	600	A
Slots per pole per phase	2	2	-

Table 8.2: Dataset 1: stator parameter details. Table taken from [152, Tab. 1].

	Parameter	Min.	Max.	Unit
$p_{s,1}$	Tooth head overhang 1	0.76	1.19	mm
$p_{s,2}$	Height of tooth head	12.41	18.91	mm
$p_{s,3}$	Tangential groove width	4.23	6.37	mm
$p_{s,4}$	Stator inner diameter	143.41	158.34	mm
$p_{s,5}$	Tooth head overhang 2	1.20	1.64	mm
$p_{s,6}$	Tooth width near air gap	5.05	8.60	mm
$p_{s,7}$	Iron length	160.49	168.00	mm

8.2.1 Chapter 4: datasets detail

All the scalar stator and rotor parameters for dataset 1 are listed in Table 8.2 and Table 8.3, respectively. Table 8.4 provides a brief description of all the target KPIs. Similar to dataset 1, scalar parameters and target KPI details for dataset 2 are given in Table 8.5 and Table 8.6, respectively. The details of some constant parameters for both datasets are provided in Table 8.1.

8.2.2 Chapter 5: datasets detail and numerical results

Table 8.7 describes the range of input operating points for which each PMSM design is simulated. Additionally, it provides information about a few constant simulation and system parameters. Table 8.8 lists intermediate FE measures, which are the targets for the multi-branch DNN. Table 8.9 details the target KPIs over which data-driven and hybrid approaches are numerically compared. Figure 8.1 shows the validation curve during training for the final training-validation sets of the multi-branch DNN. Table 8.10 shows the statistical analysis over the test samples of the final test set. Figure 8.2 demonstrates the torque and flux prediction for one operating point of the test design. Figure 8.3 illustrates the prediction performance of the hybrid approach compared to the conventional FE-based simulation workflow across different performance curves for three test designs. Table 8.12 provides details of scalar parameters for which multi-objective optimization is demonstrated using the hybrid approach.

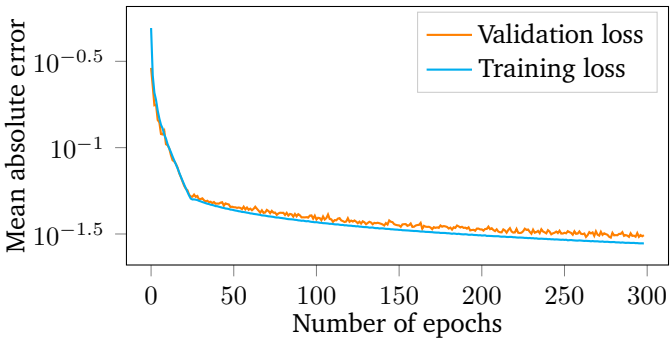


Figure 8.1: Training and validation curves. Figure taken from [153, Fig. 8], © 2022 IEEE.

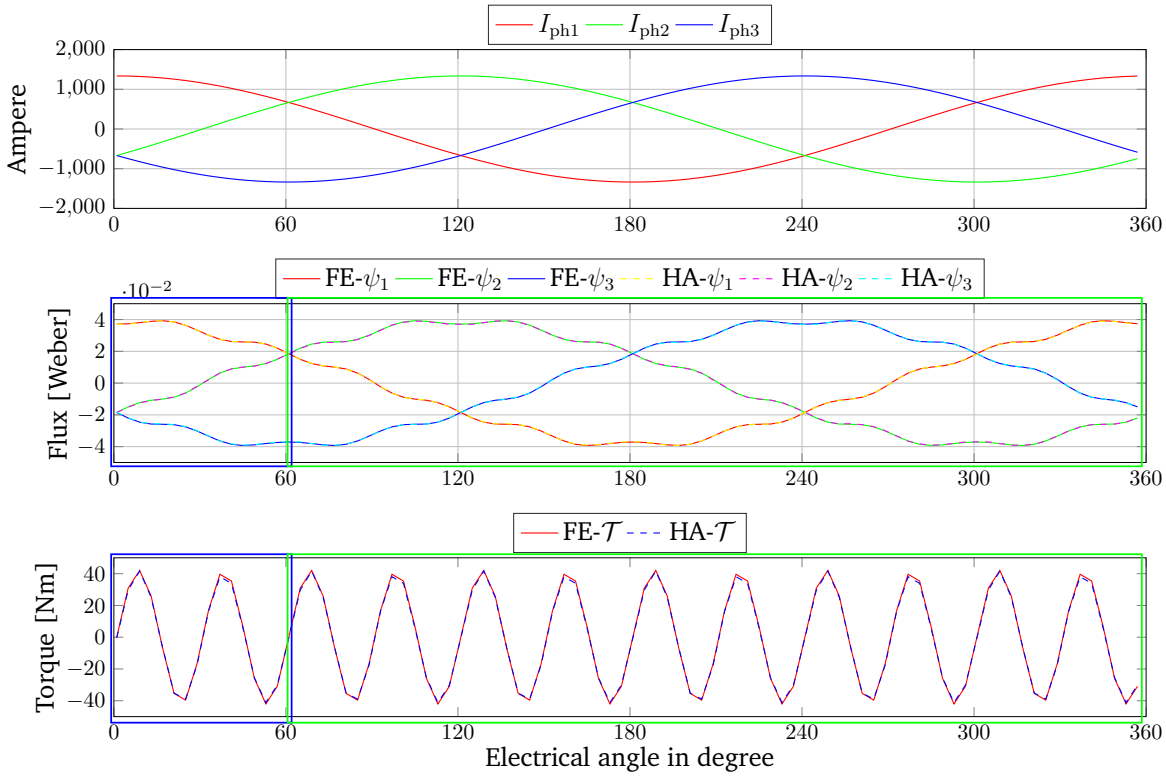
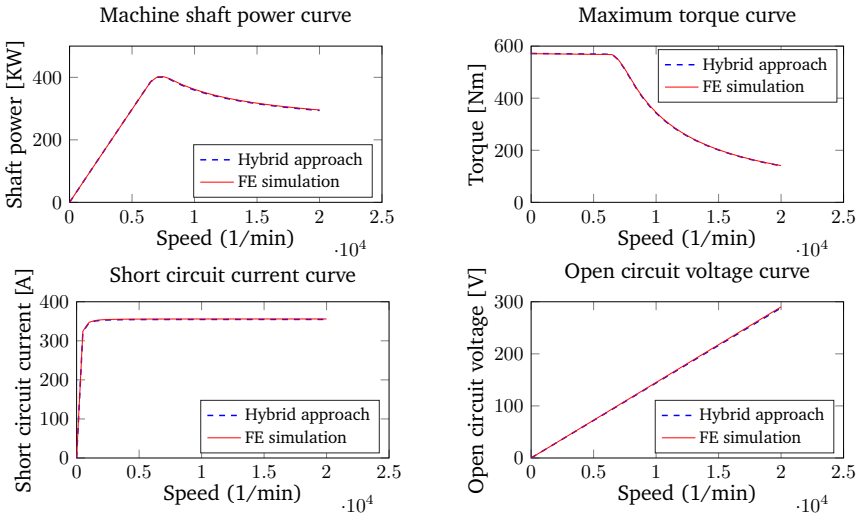
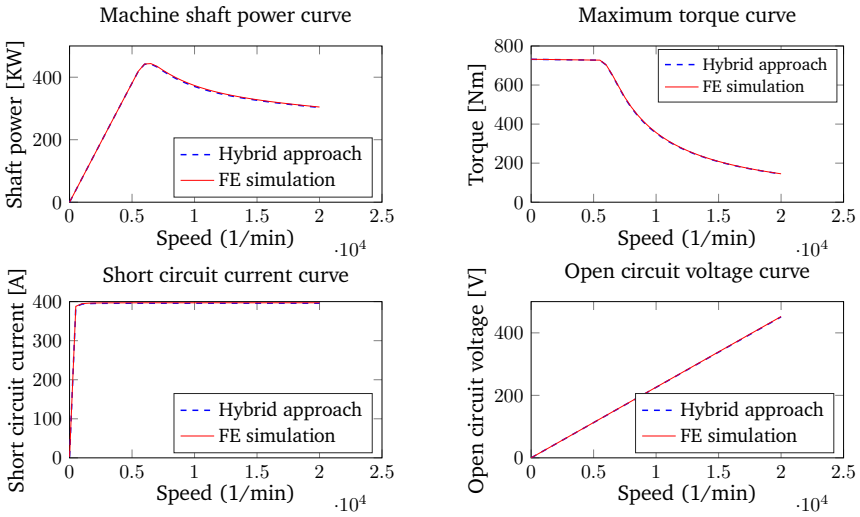


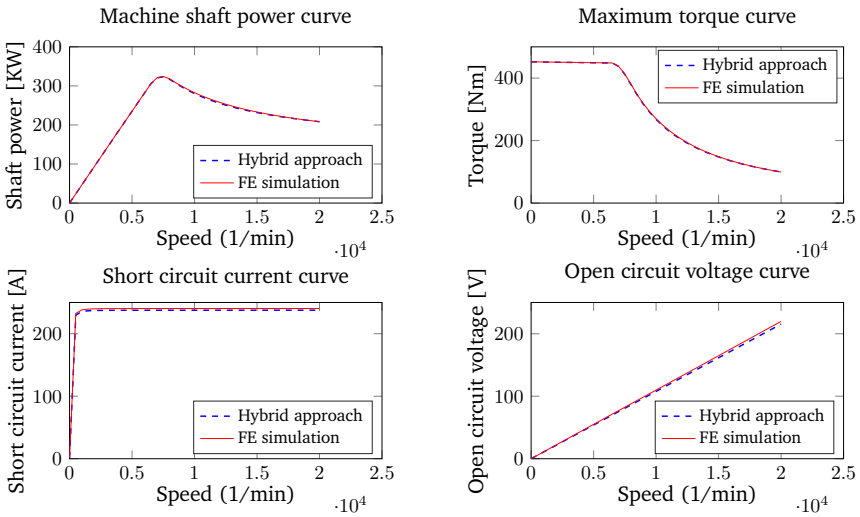
Figure 8.2: Plot of flux and torque predictions over single electrical cycle at operating point: maximal current I and $\alpha = 90°$.



(a) Test design 1



(b) Test design 2



(c) Test design 3

Figure 8.3: Different performance curves for three designs from the test set. Figure based on [153, Fig. 11], © 2022 IEEE.

Table 8.3: List of rotor parameters for dataset 1.

Parameters	Unit	Description	Part	Type	Min	Max
p_1	[mm]	Height adjustment of the inner magnetic pocket	Rotor	Continuous	6.64	12.97
p_2	[mm]	Width of magnets (inner)	Rotor	Continuous	7.19	9.48
p_3	[mm]	Height of magnets (inner)	Rotor	Continuous	5.13	6.00
p_4	[mm]	Bridge width between the two outer magnets	Rotor	Continuous	0.80	1.51
p_5	[mm]	Width of magnets (outer)	Rotor	Continuous	7.66	10.27
p_6	[mm]	Height of magnets (outer)	Rotor	Continuous	3.24	5.70
p_7	[mm]	Distance of the inner magnetic pocket to the rotor outer contour	Rotor	Continuous	1.66	2.71
p_8	[mm]	Bridge width between the two inner magnets	Rotor	Continuous	3.15	4.24
p_9	[mm]	Distance of the magnet pocket of the inner magnet to the rotor contour	Rotor	Continuous	3.36	5.64
p_{10}	[deg]	Inclination angle of the inner magnet 1	Rotor	Continuous	38.32	57.93
p_{11}	[deg]	Inclination angle of the inner magnet 2	Rotor	Continuous	19.50	38.36
p_{12}	[deg]	Pole cover angle of the inner magnet	Rotor	Continuous	112.20	124.12
p_{13}	[mm]	Radius (corner upper right) of the inner magnetic pocket	Rotor	Continuous	1.00	1.80
p_{14}	[mm]	Radius (corner lower left) of the inner magnetic pocket	Rotor	Continuous	0.56	2.32
p_{15}	[mm]	Distance of the inner magnet to the upper corner of the pocket	Rotor	Continuous	0.02	1.44
p_{16}	[mm]	Distance of the inner magnet pocket to the rotor outer contour	Rotor	Continuous	0.93	2.30
p_{17}	[mm]	Distance of the outer magnet to the upper corner of the pocket	Rotor	Continuous	0.06	0.98
p_{18}	[mm]	Distance of the outer magnet pocket to the rotor outer contour	Rotor	Continuous	0.92	1.72
p_{19}	[mm]	Distance of the outer magnetic pocket to the rotor outer contour	Rotor	Continuous	0.81	1.79
p_{20}	[mm]	Air gap	Rotor	Continuous	0.85	1.49
p_{21}	[mm]	Height adjustment of the outer magnetic pocket	Rotor	Continuous	3.26	6.50
p_{22}	[deg]	Inclination angle of the outer magnet	Rotor	Continuous	15.48	36.05
p_{23}	[deg]	Angle to the corner point of the outer magnetic pocket	Rotor	Continuous	80.70	96.64
p_{24}	[deg]	Pole angle of the outer magnet	Rotor	Continuous	57.85	71.54
p_{25}	[mm]	Radius (corner upper right) of the outer magnetic pocket	Rotor	Continuous	0.30	0.66
p_{26}	[mm]	Radius (corner lower left) of the outer magnetic pocket	Rotor	Continuous	0.30	1.17
p_{27}	[mm]	distance between upper corners of inner magnets	Rotor	Continuous	0.32	0.80
p_{28}	[mm]	Spoke height 1	Rotor	Continuous	16.50	20.00
p_{29}	[mm]	Spoke height 2	Rotor	Continuous	3.66	9.34
p_{30}	[mm]	Spoke width	Rotor	Continuous	3.00	5.48
p_{31}	[deg]	Spoke design mechanical angle 1	Rotor	Continuous	2.00	7.06
p_{32}	[deg]	Spoke design mechanical angle 2	Rotor	Continuous	3.00	8.99
p_{33}	[deg]	Spoke design mechanical angle 3	Rotor	Continuous	15.23	20.00
p_{34}	[degel]	Concavity angle 1	Rotor	Continuous	10.95	15.99
p_{35}	[degel]	Concavity angle 2	Rotor	Continuous	14.79	18.42
p_{36}	[degel]	Concavity angle 3	Rotor	Continuous	17.22	20.59
p_{37}	[degel]	Concavity angle 4	Rotor	Continuous	19.39	23.86
p_{38}	[degel]	Concavity angle 5	Rotor	Continuous	21.63	26.96
p_{39}	[mm]	Concavity diameter 2	Rotor	Continuous	0.10	0.60
p_{40}	[mm]	Concavity diameter 3	Rotor	Continuous	0.10	0.60
p_{41}	[mm]	Concavity diameter 4	Rotor	Continuous	0.10	0.60
p_{42}	[degel]	Concavity angle 1	Rotor	Continuous	45.67	50.91
p_{43}	[degel]	Concavity angle 2	Rotor	Continuous	49.32	52.84
p_{44}	[degel]	Concavity angle 3	Rotor	Continuous	51.91	54.78
p_{45}	[degel]	Concavity angle 4	Rotor	Continuous	53.94	56.98
p_{46}	[degel]	Concavity angle 5	Rotor	Continuous	55.68	61.55
p_{47}	[mm]	Concavity diameter 2	Rotor	Continuous	0.10	1.00
p_{48}	[mm]	Concavity diameter 3	Rotor	Continuous	0.10	1.00
p_{49}	[mm]	Concavity diameter 4	Rotor	Continuous	0.10	1.00

8.2.3 Chapter 6: datasets detail and numerical results

This subsection is divided into two parts. Each part provides details about datasets and some numerical results for the given scenario.

Table 8.4: Dataset 1: KPIs information. Table taken from [152, Tab. 3].

	KPI	Unit
y_1	Costs of active parts	Euro
y_2	Critical field strength	kA/m
y_3	Maximum torque of machine	Nm
y_4	Maximum power of machine	W
y_5	Weighted efficiency value	%
y_6	Maximum torque-ripple	Nmp
y_7	Torque-ripple behavior of machine	-
y_8	Inverter loss at desired operation point	W
y_9	Sound power level of machine	dBA
y_{10}	Maximum magnet temperature	K
y_{11}	Maximum winding temperature	K

Table 8.5: Dataset 2: parameter detail. Table taken from [152, Tab. 6].

	Parameter	Min.	Max.	Unit
p_1	Height of inner magnets	4.0	7.0	mm
p_2	Width of outer magnet	7.0	12.0	mm
p_3	Height of tooth head	12.0	17.0	mm
p_4	Rotor outer diameter	160.0	170.0	mm
p_5	Height of tooth head	4.0	7.0	mm
p_6	Angle of inner magnets	15.0	40.0	degree
p_7	Width of inner magnets	7.0	11.5	mm
p_8	Angle of inner magnets	28.0	58.0	degree
p_9	Height of outer magnet	3.0	7.0	mm
p_{10}	Angle of outer magnet	15.0	38.0	degree
p_{11}	Pole angle of outer magnet	45.0	80.0	degree
p_{12}	Tooth head width	5.0	9.0	mm

8.2.3.1 Scenario 1: Heterogeneous parameterization by rotor topology

Details of some constant simulation and system parameters are given in Table 8.13 for both SV and DV topologies. Table 8.14 and Table 8.15 provide full information on SV and DV scalar parameters, respectively. Figure 8.4a and Figure 8.5a display parameter reconstruction plots for training and test samples of SV parameters. Similarly, Figure 8.4b and Figure 8.5b illustrate parameter reconstruction plots for DV parameters.

8.2.3.2 Scenario 2: Heterogeneous parameterization by machine technology

Similar to the previous scenario, Table 8.16 lists system parameters that are kept constant for both machine types to allow comparison to some extent. Table 8.17 and Table 8.12 provide full details of ASM and PMSM parameters, respectively. Figure 8.7 for ASM and PMSM shows the pairwise distribution for a few

Table 8.6: Dataset 2: KPIs information. Table taken from [152, Tab. 5].

	KPI	Unit
y_1	Total cost	€
y_2	Maximum torque of machine	Nm
y_3	Maximum power at maximum rpm	KW
y_4	Iron losses at desired operation point	W
y_5	Copper loss at desired operation point	W
y_6	Maximum torque ripple	Nmp
y_7	Torque-ripple behavior of machine	-
y_8	Mass of copper	Kg
y_9	Mass of magnet	Kg
y_{10}	Mass of iron	Kg

Table 8.7: PMSM model parameters.

	Input operating points	Min	Max	Unit
p_{I}	Input phase current I (step size: $I_{\max}/6$)	0.00	1336.40	A
p_α	Control angle α (step size: 18°)	0	360	degree
	Constant parameter	**Value**		**Unit**
k_1	Pole pairs	4		-
k_2	Slots per pole per phase	2		-
k_3	Stator type	symmetric		-
	System parameter	**Value**		**Unit**
s_1	Inverter input DC current	900		A
s_2	Inverter input DC voltage	640		V
s_3	Rotational speed	$[1, 20000]$ (step size: 500)		rpm

Table 8.8: Details of outputs (intermediate measures). Table taken from [153, Tab. 2], © 2022 IEEE.

	Measure	Unit
z_1	Nonlinear iron losses $\mathcal{V}_{\mathrm{fe}}$	J
z_2	Electromagnetic torque $\mathcal{T}$	Nm
z_3	Flux linkage ψ_1 coil 1	Wb
z_4	Flux linkage ψ_2 coil 2	Wb
z_5	Flux linkage ψ_3 coil 3	Wb

parameters and target KPIs for both machine types. Figure 8.6 displays the combined validation curve during training, which includes parameter reconstruction loss and KPIs prediction loss. Parameter reconstruction prediction plots for ASM training and test samples are depicted in Figure 8.8a and Figure 8.9. Similar to ASM, prediction plots for PMSM are shown in Figure 8.8b and Figure 8.10.

Table 8.9: KPIs information. Table taken from [153, Tab. 3], © 2022 IEEE.

	KPIs	**Unit**
y_1	Maximum torque on limit curve	Nm
y_2	Max. shaft power	W
y_3	Max. shaft power at max. speed	W
y_4	Max. torque ripple on limit curve	Nm
y_5	Material cost	Euro
y_6	Mass of active parts	Kg
y_7	Torque ripple deviation	Nm

Table 8.10: Intermediate measures over test samples with optimized multi-branch DNN. Tabel taken from [153, Tab. 5], © 2022 IEEE.

Intermediate measures	$\overline{\varepsilon}_{\text{MRE}}$	ε_{PCC}
eddy current loss (rotor)	1.5×10^{0}	0.98
eddy current loss (stator)	5.9×10^{-1}	0.99
hysteresis loss (rotor)	1.6×10^{0}	0.98
hysteresis loss (stator)	4.8×10^{-1}	0.99
	$\overline{\varepsilon}_{\text{MAE}}$	ε_{PCC}
Electromagnetic torque $\mathcal{T}$	6.5×10^{-1}	0.99
Flux linkage ψ_1 coil 1	1.0×10^{-4}	0.99
Flux linkage ψ_2 coil 2	1.0×10^{-4}	0.99
Flux linkage ψ_3 coil 3	1.0×10^{-4}	0.99

Table 8.11: List of varying scalar parameters for the double V PMSM.

Parameters	Unit	Description	Category	Min	Max
p_1	[mm]	Stator tooth height	Geometry	12.00	20.00
p_2	[mm]	Air gap	Geometry	0.50	2.00
p_3	[mm]	Rotor outer diameter	Geometry	159.00	165.00
p_4	[mm]	Width of inner magnet	Geometry	8.00	25.00
p_5	[mm]	Height of inner magnet	Geometry	2.23	7.00
p_6	[mm]	Distance of the magnet pocket of the inner magnet to the rotor contour	Geometry	3.50	5.50
p_7	[mm]	Geometry parameter as per the node plan	Geometry	0.60	1.50
p_8	[mm]	Distance of the inner magnet to the upper corner of the pocket	Geometry	0.30	1.20
p_9	[mm]	Distance 1 of the inner magnet pocket to the rotor outer contour	Geometry	0.80	2.80
p_{10}	[mm]	Distance 2 of the inner magnetic pocket to the rotor outer contour	Geometry	0.80	2.80
p_{11}	[mm]	Bridge width between the two inner magnets	Geometry	0.80	5.00
p_{12}	[mm]	Height adjustment 1 of the inner magnetic pocket	Geometry	4.00	9.00
p_{13}	[deg]	Inclination angle of the inner magnet 1	Geometry	25.00	51.53
p_{14}	[deg]	Pole cover angle of the inner magnet	Geometry	112.00	132.00
p_{15}	[mm]	Radius 1 of the inner magnetic pocket	Geometry	0.50	1.80
p_{16}	[mm]	Radius 2 of the inner magnetic pocket	Geometry	1.20	2.50
p_{17}	[mm]	Radius 3 of the inner magnetic pocket	Geometry	0.50	1.50
p_{18}	[mm]	distance between radii 1 of the inner magnetic pocket	Geometry	2.00	4.50
p_{19}	[mm]	Height of outer magnet	Geometry	2.00	6.50
p_{20}	[mm]	Width of outer magnet	Geometry	6.00	10.50
p_{21}	[mm]	Distance of the outer magnet to the upper corner of the pocket	Geometry	0.20	0.80
p_{22}	[mm]	Distance 1 of the outer magnet pocket to the rotor outer contour	Geometry	0.80	2.00
p_{23}	[mm]	Distance 2 of the outer magnetic pocket to the rotor outer contour	Geometry	0.80	2.00
p_{24}	[mm]	Bridge width between the two outer magnets	Geometry	0.80	3.00
p_{25}	[mm]	Height adjustment 1 of the outer magnetic pocket	Geometry	3.00	8.00
p_{26}	[deg]	Inclination angle of the outer magnet	Geometry	10.00	40.00
p_{27}	[deg]	Angle 1 to the corner point of the outer magnetic pocket	Geometry	82.00	115.00
p_{28}	[deg]	Pole angle of the outer magnet	Geometry	45.00	70.00
p_{29}	[mm]	Radius 1 of the outer magnetic pocket	Geometry	0.30	1.30
p_{30}	[mm]	Radius 2 of the outer magnetic pocket	Geometry	0.30	1.00
p_{31}	[mm]	Tooth head overhang 1	Geometry	0.80	2.00
p_{32}	[mm]	Iron length	Geometry	180.00	200.00
p_{33}	[-]	Tangential groove width at groove base	Material	4.0	7.00
p_{34}	[mm]	Height of tooth head slot	Geometry	0.80	2.50
p_{35}	[-]	Remenance factor	Material	0.71	1.35
p_I	A	Input phase current	Electrical	0.00	1336.40
p_α	degree	Control angle	Electrical	0.00	360.00

Table 8.12: Details of PMSM Parameters including discrete parameters (Chapter 5 and Chapter 6).

Parameters	Unit	Description	Category	Type	Min	Max	Discrete Values
p_1	[mm]	Stator outer diameter	Geometry	Continuous	159.00	232.00	-
p_2	[mm]	Rotor outer diameter	Geometry	Continuous	99.91	196.89	-
p_3	[mm]	Air gap	Geometry	Continuous	0.80	2.20	-
p_4	[mm]	Stator tooth height	Geometry	Continuous	10.00	20.00	-
p_5	[deg]	Angle of magnet layer 1	Geometry	Continuous	17.00	32.00	-
p_6	[mm]	Height of magnet layer 1	Geometry	Continuous	3.50	5.50	-
p_7	[mm]	Tooth head overhang	Geometry	Continuous	1.18	1.45	-
p_8	[mm]	Distance of the magnet pocket of the magnet layer 1 to the rotor contour	Geometry	Continuous	0.96	6.00	-
p_9	[mm]	Relative parameter 1 in the magnet layer 1	Geometry	Continuous	0.09	0.11	-
p_{10}	[mm]	Height adjustment (corner lower left) of the outer magnetic pocket in the magnet layer 1	Geometry	Continuous	0.00	2.00	-
p_{11}	[mm]	Distance of the outer magnetic pocket to the rotor outer contour in the magnet layer 1	Geometry	Continuous	0.72	0.88	-
p_{12}	[mm]	Relative parameter 2 in the magnet layer 1	Geometry	Continuous	0.73	0.89	-
p_{13}	[mm]	Relative parameter 3 in the magnet layer 1	Geometry	Continuous	0.09	0.11	-
p_{14}	[mm]	Bridge width between the two magnets in the magnet layer 1	Geometry	Continuous	0.20	1.20	-
p_{15}	[-]	Relative width of magnet layer 1 to the left maximum	Geometry	Continuous	0.70	1.00	-
p_{16}	[-]	Relative width of magnet layer 1 to the right maximum	Geometry	Continuous	0.70	1.00	-
p_{17}	[deg]	Angle of magnet layer 2	Geometry	Continuous	22.00	40.00	-
p_{18}	[mm]	Distance of the magnet pocket of the magnet layer 2 to the rotor contour	Geometry	Continuous	0.96	6.00	-
p_{19}	[mm]	Relative parameter 1 in the magnet layer 2	Geometry	Continuous	0.09	0.11	-
p_{20}	[mm]	Height adjustment (corner lower left) of the outer magnetic pocket in the magnet layer 2	Geometry	Continuous	0.90	1.10	-
p_{21}	[mm]	Distance of the outer magnetic pocket to the rotor outer contour in the magnet layer 2	Geometry	Continuous	1.62	1.98	-
p_{22}	[mm]	Relative parameter 2 in the magnet layer 2	Geometry	Continuous	0.63	0.77	-
p_{23}	[mm]	Relative parameter 2 in the magnet layer 2	Geometry	Continuous	0.09	0.11	-
p_{24}	[mm]	Bridge width between the two magnets in the magnet layer 2	Geometry	Continuous	0.20	2.20	-
p_{25}	[mm]	Height of magnet layer 2	Geometry	Continuous	3.50	6.00	-
p_{26}	[-]	Relative width of magnet layer 2 to the left maximum	Geometry	Continuous	0.70	1.00	-
p_{27}	[-]	Relative width of magnet layer 2 to the right maximum	Geometry	Continuous	0.70	1.00	-
p_{28}	[-]	Remanence factor	Material	Continuous	0.69	1.20	-
p_{29}	[mm]	Tangential groove width at groove base	Geometry	Continuous	4.00	6.00	-
p_{30}	[-]	Number of slots per pole per phase	Geometry	Discrete	-	-	[2, 3, 4]
p_{31}	[-]	Pole pairs	Topological	Discrete	-	-	[3,4,5]
p_{32}	[-]	Stator winding connection→ 1: Star connection, 3 : Delta connection	Electrical	Discrete	-	-	[1, 3]
p_{33}	[-]	Winding scheme→ 1: Short pitch winding, 4: Full pitch winding	Electrical	Discrete	-	-	[1, 4]

Table 8.13: Constant parameters

Parameter	Single V	Double V	Unit
No of pole pairs	4	4	-
Stator type	Symmetric	Symmetric	-
Winding connection	Star	Star	-
Winding scheme	Full pitch winding	Full pitch winding	-
Max. phase voltage	640	640	V
Max. phase current	900	900	A
Slots per pole per phase	2	2	-
Stator outer diameter	230	220	mm

Table 8.14: SV parameter details.

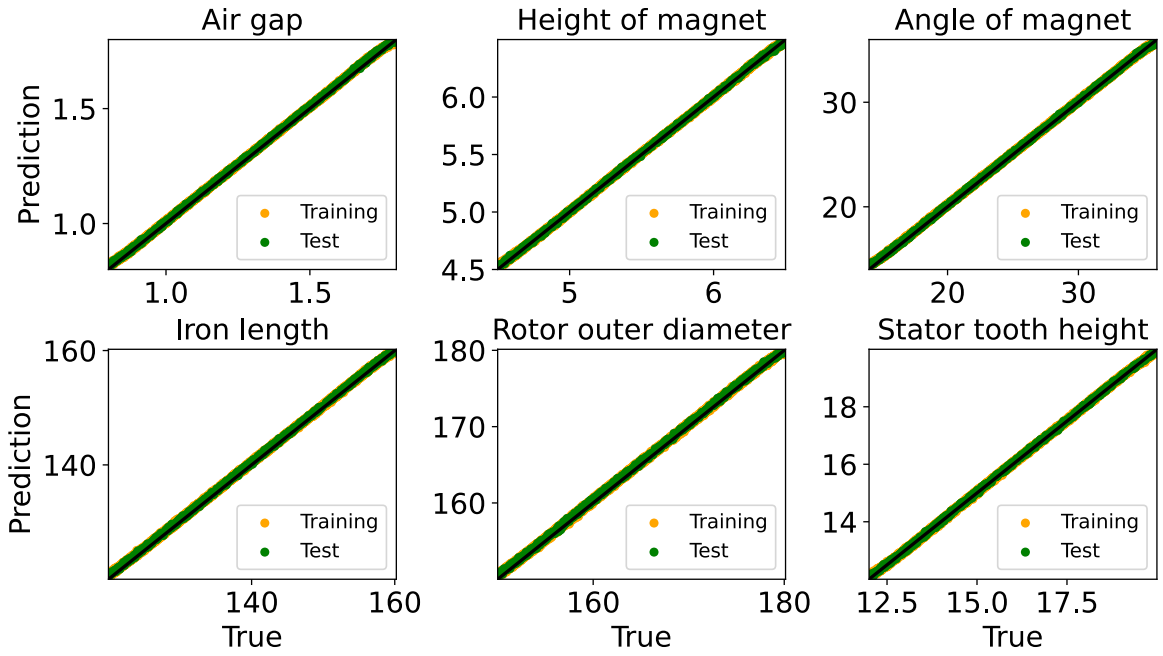
Parameter	Unit	Description	Part	Type	Range	
					Min	Max
p_1	[mm]	Air gap	Rotor/Stator	Continuous	0.80	1.80
p_2	[mm]	Height of magnet	Rotor	Continuous	4.50	6.50
p_3	[deg]	Inclination angle of magnet	Rotor	Continuous	14.00	36.00
p_4	[mm]	Iron length	Rotor	Continuous	120.00	160.00
p_5	[mm]	Rotor outer diameter	Rotor	Continuous	150.00	180.00
p_6	[mm]	Stator tooth height	Stator	Continuous	12.00	20.00
p_7	[mm]	Distance of the magnet pocket (corner upper right) to the rotor outer contour	Rotor	Continuous	1.00	2.25
p_8	[mm]	Tooth head overhang	Stator	Continuous	1.10	1.80
p_9	[deg]	Bridge width between the two magnets	Rotor	Continuous	3.10	4.20
p_{10}	[mm]	Distance of the magnet pocket of the inner magnet to the rotor contour	Rotor	Continuous	1.50	5.00
p_{11}	[mm]	Tangential groove width at groove base	Stator	Continuous	4.50	7.50
p_{12}	[mm]	Height of tooth head	Stator	Continuous	1.30	2.20
p_{13}	[mm]	Rotor inner diameter	Rotor	Continuous	40.00	70.00

Table 8.15: DV parameter details.

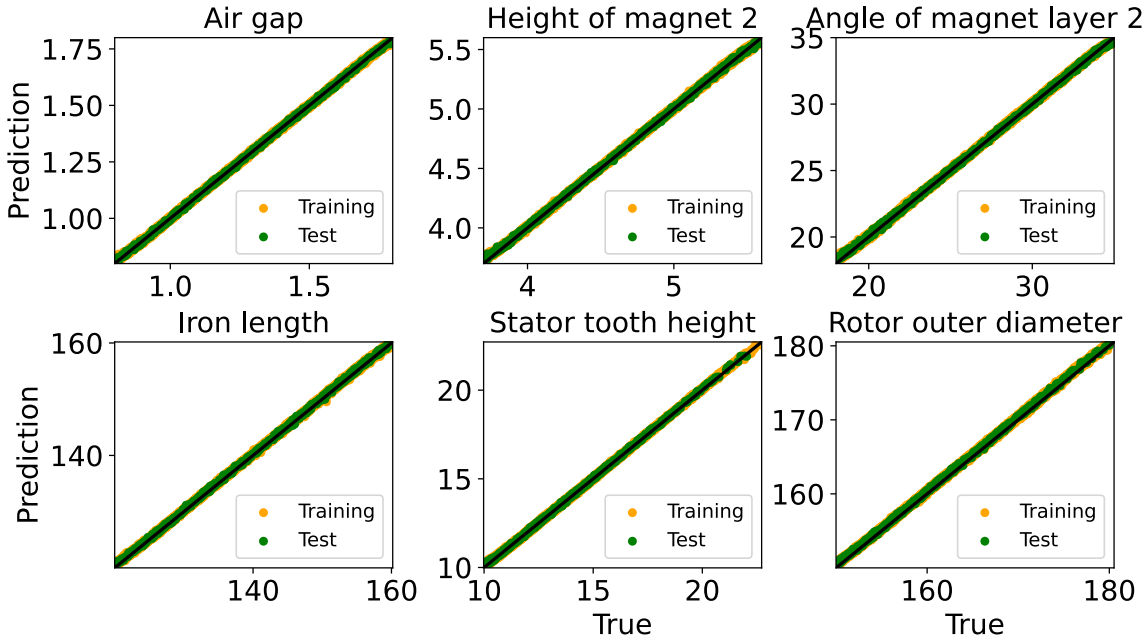
Parameter	Unit	Description	Part	Type	Range	
					Min	Max
p_1	[mm]	Air gap	Rotor/Stator	Continuous	0.80	1.80
p_2	[mm]	Height of magnet 2 (layer 2)	Rotor	Continuous	3.70	5.60
p_3	[deg]	Angle of magnet layer 2	Rotor	Continuous	18.00	35.00
p_4	[mm]	Iron length	Rotor	Continuous	120.00	160
p_5	[mm]	Rotor outer diameter	Rotor	Continuous	150.00	180
p_6	[deg]	Angle of magnet layer 1	Rotor	Continuous	20.00	40.00
p_7	[mm]	Height of magnet 1 (layer 1)	Rotor	Continuous	4.50	6.50
p_8	[mm]	Stator tooth height	Stator	Continuous	10.00	23
p_9	[mm]	Distance of the magnet pocket in layer 2 to the rotor outer contour	Rotor	Continuous	0.90	1.70
p_{10}	[mm]	Bridge width between the two magnets of layer 2	Rotor	Continuous	0.80	1.47
p_{11}	[mm]	Distance of the magnet pocket of the magnet in layer 2 to the rotor contour	Rotor	Continuous	1.00	5.00
p_{12}	[mm]	Bridge width between the two magnets of layer 1	Rotor	Continuous	3.10	4.20
p_{13}	[mm]	Tooth head overhang	Stator	Continuous	1.10	1.35
p_{14}	[mm]	Distance of the magnet pocket of the magnet in layer 1 to the rotor contour	Rotor	Continuous	1.50	5.00
p_{15}	[mm]	Tangential groove width at groove base	Stator	Continuous	3.50	7.50
p_{16}	[mm]	Height of tooth head	Stator	Continuous	1.00	3.00
p_{17}	[mm]	Rotor inner diameter	Rotor	Continuous	38.00	76.00
p_{18}	[mm]	Distance of the magnet pocket in layer 1 to the rotor outer contour	Rotor	Continuous	1.00	2.25

Table 8.16: System parameters. Table taken from [151, Tab. 2].

	System parameter	Value	Unit
c_1	Inverter input DC voltage	650	V
c_2	Inverter input DC current	400	A
c_3	Rotational speed	$[1, 16000]$ (step size: 1000)	rpm

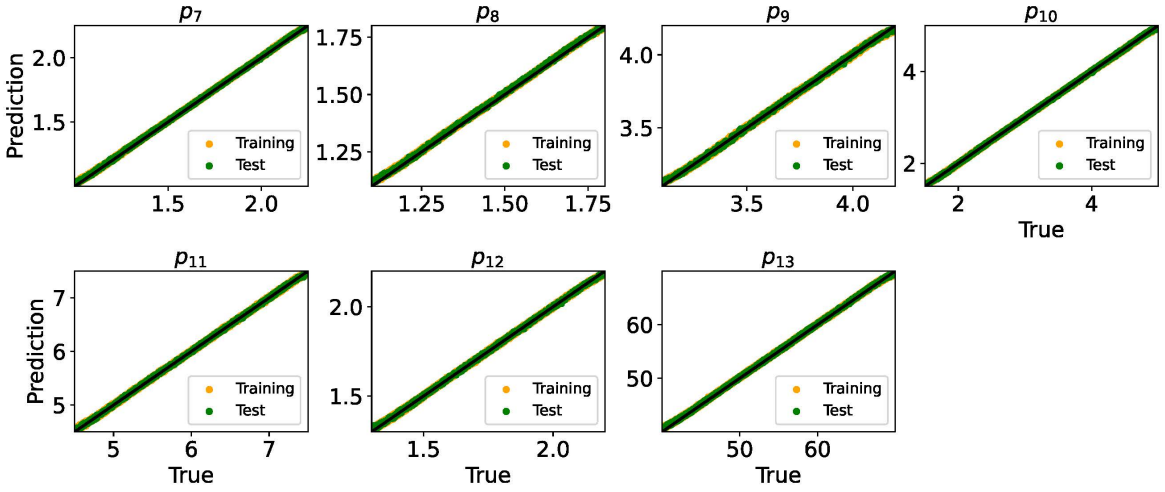


(a) SV parameters

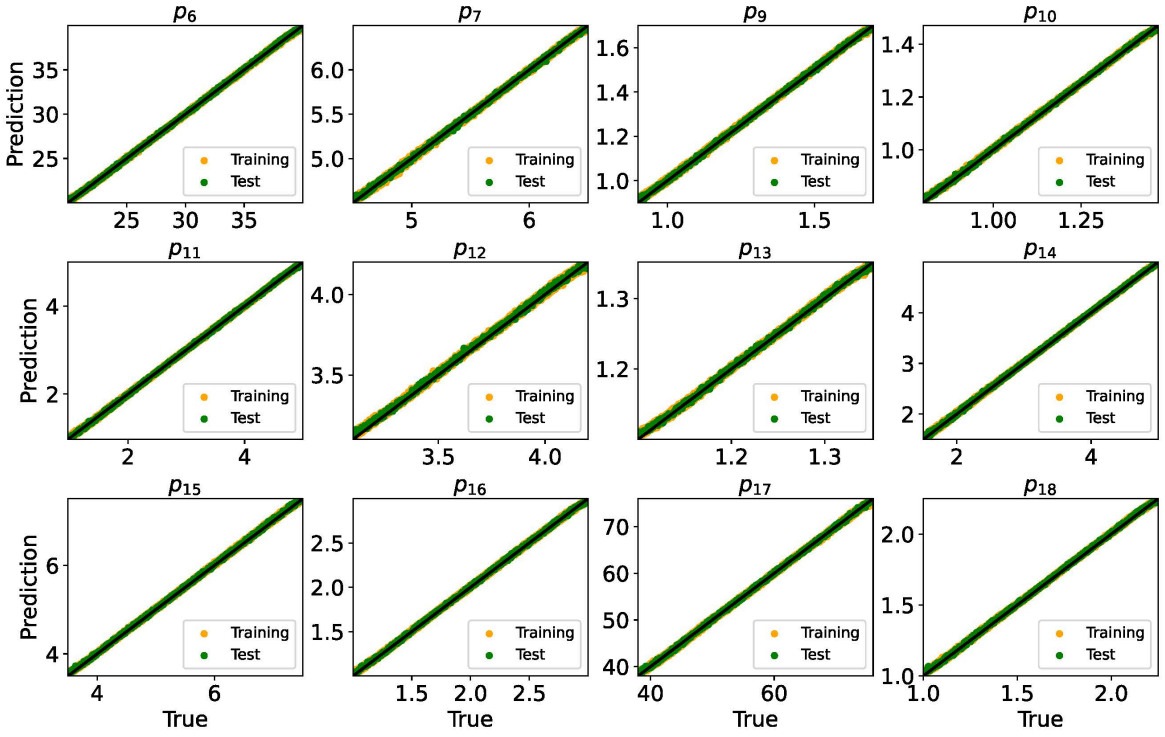


(b) DV parameters.

Figure 8.4: SV and DV parameters reconstruction prediction plots over test samples. Figures based on [154, Fig. 7], © 2022 IEEE.



(a) SV parameters



(b) DV parameters.

Figure 8.5: SV and DV parameters reconstruction prediction plots over test samples.

Table 8.17: Details of ASM Parameters.

Parameters	Unit	Description	Category	Type	Min	Max	Discrete Values
p_1	[mm]	Stator outer diameter	Geometry	Continuous	159.00	232.00	-
p_2	[mm]	Air gap	Geometry	Continuous	0.65	1.70	-
p_3	[mm]	Rotor outer diameter	Geometry	Continuous	85.00	190.00	-
p_4	[mm]	Rotor slot height (winding space)	Topological	Continuous	10.00	21.00	-
p_5	[mm]	Rotor slot width (level 1)	Geometry	Continuous	0.60	1.50	-
p_6	[mm]	Stator slot height (winding space)	Geometry	Continuous	10.00	21.00	-
p_7	[mm]	Stator slot width level 1	Geometry	Continuous	3.55	7.00	-
p_8	[mm]	Radius of rotor slot level 2	Geometry	Continuous	1.50	3.00	-
p_9	[mm]	Radius of rotor slot level 3	Geometry	Continuous	0.50	1.50	-
p_{10}	[mm]	Stator slot width level 3	Geometry	Continuous	0.60	1.50	-
p_{11}	[mm]	Stator slot height (level 1)	Geometry	Continuous	0.50	2.00	-
p_{12}	[-]	Number of rotor slots deviated from Stator slots	Geometry	Discrete	-	-	[3, 4, 6, 9, 12, 18]
p_{13}	[mm]	Inner diameter of short-circuit ring (rotor cage)	Geometry	Continuous	15.00	21.00	-
p_{14}	[mm]	Rotor slot height (level 1)	Geometry	Continuous	0.50	2.00	-
p_{15}	[-]	Number of slots per pole per phase	Geometry	Discrete	-	-	[2, 3, 4]
p_{16}	[-]	Pole pairs	Topological	Discrete	-	-	[2, 3, 4]
p_{17}	[-]	Stator winding connection → 1: Star connection, 3: Delta connection	Electrical	Discrete	-	-	[1, 3]
p_{18}	[-]	Winding scheme → 4: Full pitch winding, 1: Short pitch winding	Electrical	Discrete	-	-	[1, 4]

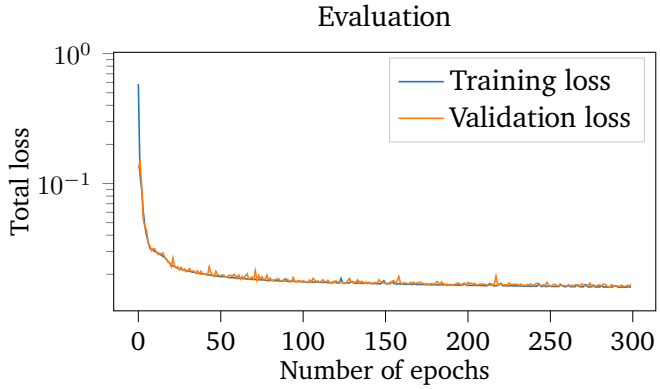
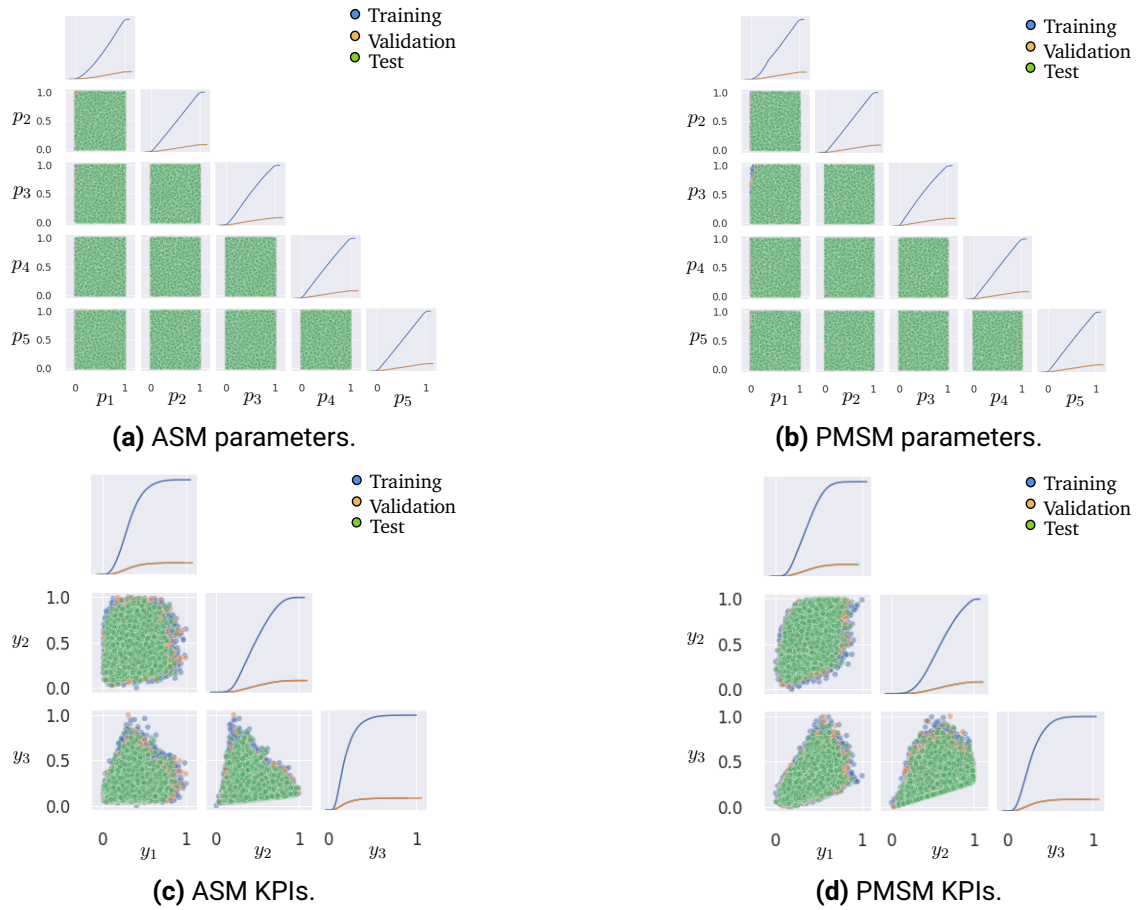


Figure 8.6: Curves depicting the training and validation losses. Figure taken from [151, Fig. 9].



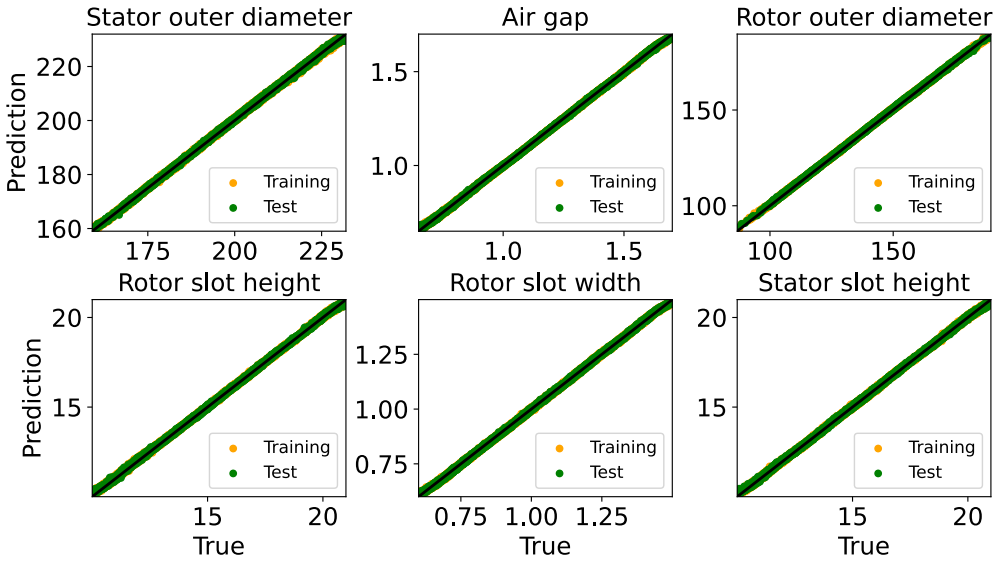
(a) ASM parameters.

(b) PMSM parameters.

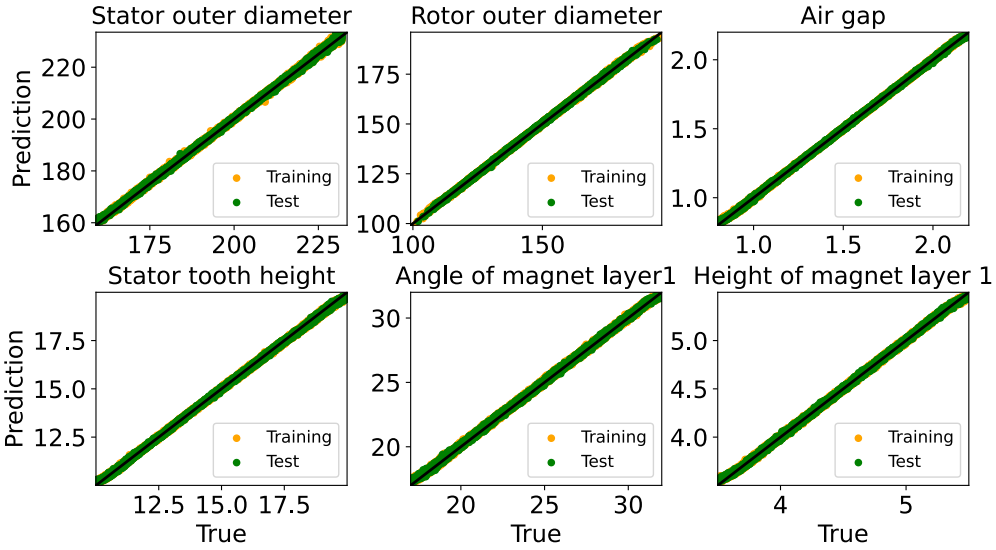
(c) ASM KPIs.

(d) PMSM KPIs.

Figure 8.7: Visualization parameter and KPIs distribution. Figure taken from [151, Fig. 4].



(a) ASM parameters



(b) PMSM parameters.

Figure 8.8: ASM and PMSM parameters reconstruction prediction plots over test samples. Figure taken from [151, Fig. 11 and Fig. 12].

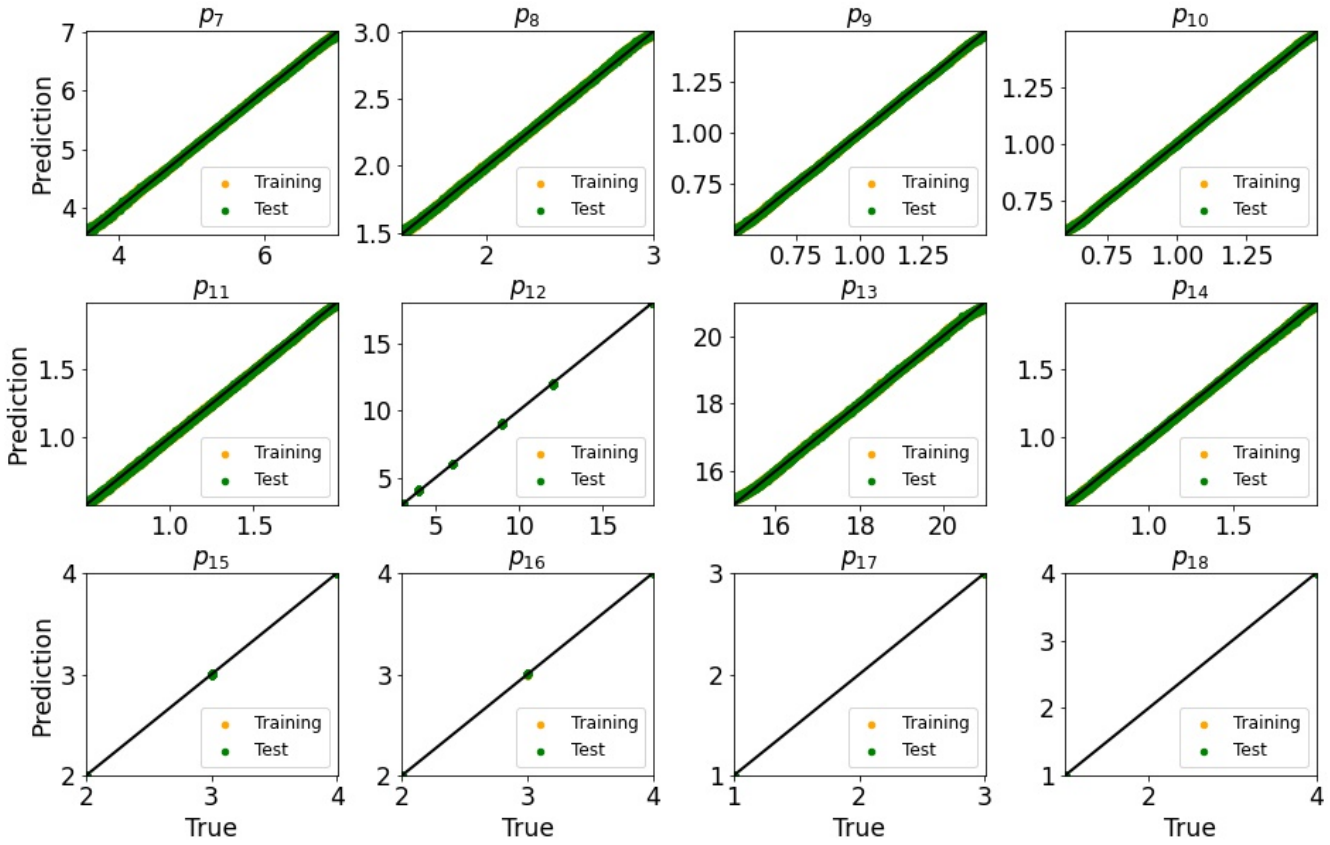


Figure 8.9: ASM parameters reconstruction prediction plots over test samples.

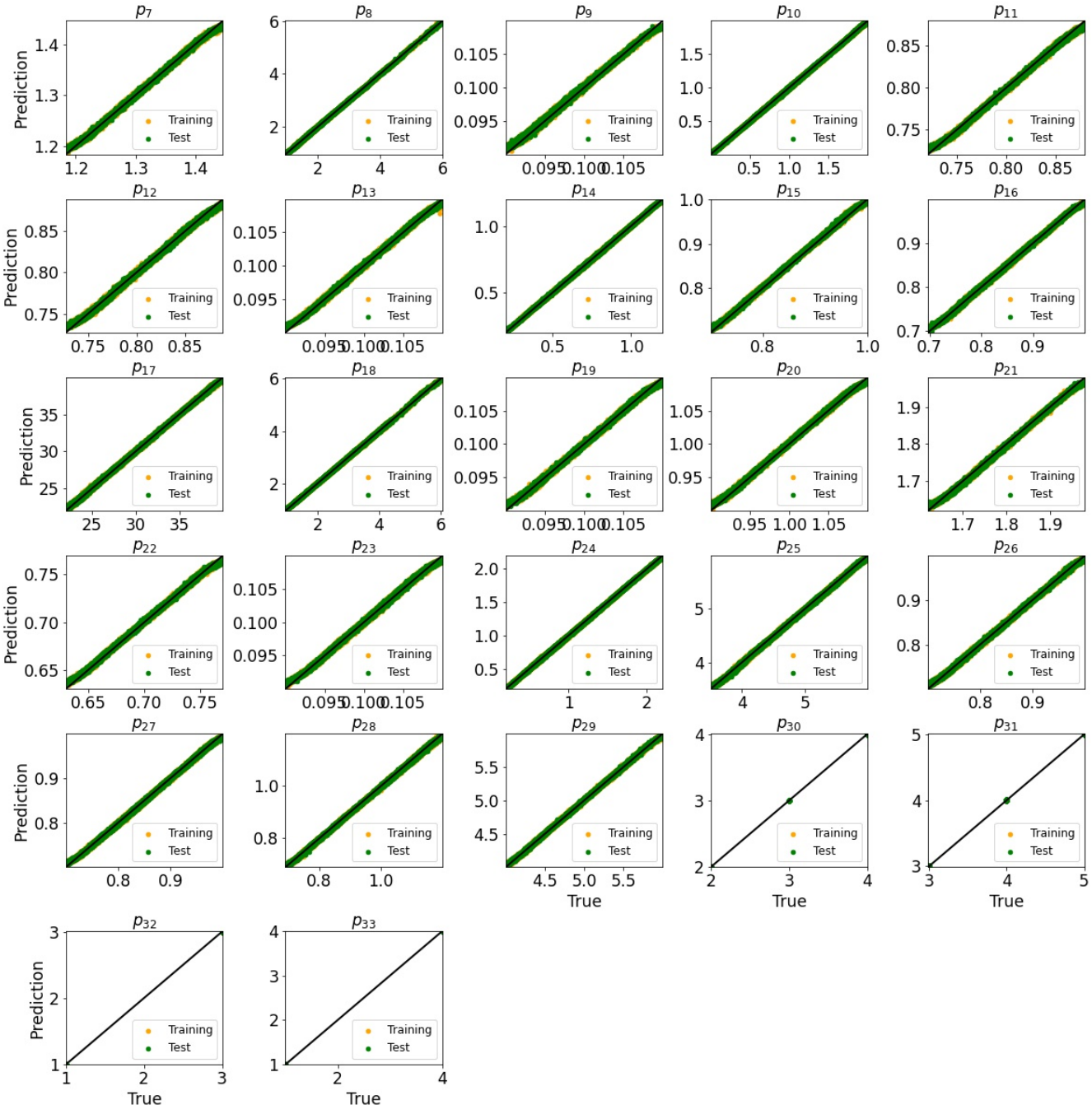


Figure 8.10: PMSM parameters reconstruction prediction plots over test samples.

List of acronyms

AI	Artificial Intelligence
AC	alternating current
Adam	Adaptive Moment Estimation
AdaGrad	adaptive gradient algorithm
ASM	asynchronous machine
ANNs	artificial neural networks
CAD	computer-aided design
CNN	convolutional neural network
DCNN	deep convolutional neural network
EMC	electromagnetic compatibility
DL	deep learning
DV	Double V
EMF	electromotive force
DNNs	deep neural networks
EVs	electric vehicles
EESM	electrically excited synchronous machine
ELU	Exponential Linear Unit
FE	finite element
FEM	finite element method
FSM	flux switching machine
FFSO	free-form structural optimization
GPR	Gaussian process regression
GPUs	graphical processing units
GANs	generative adversarial networks
HPC	high-performance computing
HEVs	hybrid electric vehicles
IMs	induction machines
IPMSM	interior permanent magnet synchronous motor

HPO hyperparameter optimization
KPIs key performance indicators
KL Kullback–Leibler
LHS Latin hypercube sampling
ML machine learning
MDP Markov Decision Process
MVP magnetic vector potential
MRE Mean relative error
MSE Mean squared error
MOO multi-objective optimization
MLP multi-layer perceptron
NSGA non-dominated sorting genetic algorithm
PDEs partial differential equations
PCC Pearson correlation coefficient
PINN physics-informed neural network
PM permanent magnet
PMSMs Permanent magnet synchronous machines
RBF Radial Basis Function
ReLU Rectified Linear Unit
RL Reinforcement learning
RNN recurrent neural network
RMSE root mean squared error
SV Single V
SynRM synchronous reluctance machine
SGD stochastic gradient descent
SRM switched reluctance machine
UIV unique identifier value
VAE variational autoencoder
1D one-dimensional
2D two-dimensional

Bibliography

[1] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[2] A. Amini and A. Amini. *Mit 6.s191: Introduction to deep learning*. Lecture note, Accessed 14 March 2023, Massachusetts Institute of Technology. (2023), [Online]. Available: `https://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf`.

[3] M. Amrhein, T. C. O' Connell, and J. R. Wells, *An integrated design process for optimized high-performance electrical machines*, in *2013 International Electric Machines & Drives Conference*, 2013, pp. 847–854. DOI: 10.1109/IEMDC.2013.6556197.

[4] M. Andrychowicz *et al.*, *Learning to learn by gradient descent by gradient descent*, *Advances in neural information processing systems*, vol. 29, 2016.

[5] ANSYS.Inc, *Methods for multi-disciplinary optimization and robustness analysis*, version 2022R2, Canonsburg (PA) USA: ANSYS.Inc, July, 2022, 90 pp.

[6] C. Audet and W. Hare, *Derivative-free and blackbox optimization*, 2017.

[7] R. Bargallo, *Finite elements for electrical engineering*, *Universitat Politecnica De Catalunya*, 2006.

[8] S. Barmada, N. Fontana, L. Sani, D. Thomopulos, and M. Tucci, *Deep learning and reduced models for fast optimization in electromagnetics*, *IEEE Transactions on Magnetics*, vol. 56, no. 3, pp. 1–4, 2020. DOI: 10.1109/TMAG.2019.2957197.

[9] A. Beltrán-Pulido, I. Bilionis, and D. Aliprantis, *Physics-informed neural networks for solving parametric magnetostatic problems*, *IEEE Transactions on Energy Conversion*, vol. 37, no. 4, pp. 2678–2689, 2022. DOI: 10.1109/TEC.2022.3180295.

[10] Y. Bengio, A. Courville, and P. Vincent, *Representation learning: A review and new perspectives*, *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] Y. Bengio, Y. LeCun, *et al.*, *Scaling learning algorithms towards ai*, *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[12] F. Benvenuto, M. Piana, C. Campi, and A. M. Massone, *A hybrid supervised/unsupervised machine learning approach to solar flare prediction*, *The Astrophysical Journal*, vol. 853, no. 1, p. 90, 2018.

[13] H.-G. Beyer and H.-P. Schwefel, *Evolution strategies–a comprehensive introduction*, *Natural computing*, vol. 1, pp. 3–52, 2002.

[14] N. Bianchi and S. Bolognani, *Design optimisation of electric motors by genetic algorithms*, *IEE Proceedings-Electric Power Applications*, vol. 145, no. 5, pp. 475–483, 1998.

[15] B. Bischl *et al.*, *Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges*, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 2, e1484, 2023.

[16] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[17] F. Bittner and I. Hahn, *Kriging-assisted multi-objective particle swarm optimization of permanent magnet synchronous machine for hybrid and electric cars*, in *2013 International Electric Machines Drives Conference*, 2013, pp. 15–22. DOI: 10.1109/IEMDC.2013.6556123.

[18] A. Boglietti, A. Cavagnino, A. Tenconi, S. Vaschetto, and P. di Torino, *The safety critical electric machines and drives in the more electric aircraft: A survey*, in *2009 35th Annual Conference of IEEE Industrial Electronics*, 2009, pp. 2587–2594. DOI: 10.1109/IECON.2009.5415238.

[19] A. Boglietti and M. Pastorelli, *Induction and synchronous reluctance motors comparison*, in *2008 34th Annual Conference of IEEE Industrial Electronics*, 2008, pp. 2041–2044. DOI: 10.1109/IECON.2008.4758270.

[20] R. Bojoi, S. Rubino, A. Tenconi, and S. Vaschetto, *Multiphase electrical machines and drives: A viable solution for energy generation and transportation electrification*, in *2016 International Conference and Exposition on Electrical and Power Engineering (EPE)*, 2016, pp. 632–639. DOI: 10.1109/ICEPE.2016.7781416.

[21] I. Boldea and S. A. Nasar, *The induction machines design handbook(2nd ed.)* CRC press, 2018.

[22] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, *Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models*, *IEEE Transactions on Pattern Analysis and amp; Machine Intelligence*, vol. 44, no. 11, pp. 7327–7347, 2022, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3116668.

[23] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, *Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models*, *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[24] Z. Bontinck, *Simulation and robust optimization for electric devices with uncertainties*, en, Ph.D. dissertation, Technische Universität, Darmstadt, 2018.

[25] Z. Bontinck, H. De Gersem, and S. Schöps, *Response surface models for the uncertainty quantification of eccentric permanent magnet synchronous machines*, *IEEE Transactions on Magnetics*, vol. 52, no. 3, 2016, Article #7203404, ISSN: 0018-9464. DOI: 10.1109/TMAG.2015.2491607.

[26] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, *A survey on multi-output regression*, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, 2015.

[27] B. K. Bose, *Neural network applications in power electronics and motor drives—an introduction and perspective*, *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 14–33, 2007. DOI: 10.1109/TIE.2006.888683.

[28] G. Bramerdorfer, J. A. Tapia, J. J. Pyrhönen, and A. Cavagnino, *Modern electrical machine design optimization: Techniques, trends, and best practices*, *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 7672–7684, 2018. DOI: 10.1109/TIE.2018.2801805.

[29] L. Breiman, *Random forests*, *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[30] T. Brown *et al.*, *Language models are few-shot learners*, *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[31] D. Bucherl, R. Nuscheler, W. Meyer, and H.-G. Herzog, *Comparison of electrical machine types in hybrid drive trains: Induction machine vs. permanent magnet synchronous machine*, in *2008 18th International Conference on Electrical Machines*, 2008, pp. 1–6. DOI: 10.1109/ICELMACH.2008.4800155.

[32] L. Buitinck *et al.*, *API design for machine learning software: Experiences from the scikit-learn project*, in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[33] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995. DOI: 10.1137/0916069.

[34] W. Caesarendra, T. Wijaya, and T. Pappachan Bobby K.and Tjahjowidodo, *Adaptation to industry 4.0 using machine learning and cloud computing to improve the conventional method of deburring in aerospace manufacturing industry*, in *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, 2019, pp. 120–124. DOI: 10.1109/ICTS.2019.8850990.

[35] T. Chai and R. R. Draxler, *Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature*, *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[36] J.-P. Chilès and N. Desassis, *Fifty years of kriging*, *Handbook of mathematical geosciences: Fifty years of IAMG*, pp. 589–612, 2018.

[37] W. Q. Chu *et al.*, *Comparison of electrically excited and interior permanent magnet machines for hybrid electric vehicle application*, in *2014 17th International Conference on Electrical Machines and Systems (ICEMS)*, 2014, pp. 401–407. DOI: 10.1109/ICEMS.2014.7013504.

[38] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, *Flexible, high performance convolutional neural networks for image classification*, in *Twenty-second international joint conference on artificial intelligence*, Citeseer, 2011.

[39] S. Clénet, *Uncertainty quantification in computational electromagnetics: The stochastic approach*, *International Compumag Society Newsletter*, vol. 13, pp. 3–13, 2013, ISSN: 1026-0854.

[40] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus)*, in *ICLR 2016 : International Conference on Learning Representations 2016*, 2016.

[41] I. Cortes Garcia, S. Schöps, H. De Gersem, and S. Baumanns, *Systems of differential algebraic equations in computational electromagnetics*, *Applications of Differential-Algebraic Equations: Examples and Benchmarks*, pp. 123–169, 2019.

[42] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, *Scientific machine learning through physics–informed neural networks: Where we are and what's next*, *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022.

[43] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, *A survey of deep learning and its applications: A new paradigm to machine learning*, *Archives of Computational Methods in Engineering*, vol. 27, pp. 1071–1092, 2020.

[44] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. DOI: 10.1109/4235.996017.

[45] K. Deb, *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[46] P. Di Barba, *Future trends in optimal design in electromagnetics*, *IEEE Transactions on Magnetics*, vol. 58, no. 9, pp. 1–4, 2022. DOI: 10.1109/TMAG.2022.3164204.

[47] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, *A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks*, in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, 2018, pp. 92–99. DOI: 10.1109/CTEMS.2018.8769211.

[48] S. Doi, H. Sasaki, and H. Igarashi, *Multi-objective topology optimization of rotating machines using deep learning*, *IEEE Transactions on Magnetics*, vol. 55, no. 6, pp. 1–5, 2019. DOI: 10.1109/TMAG.2019.2899934.

[49] S. Dong, P. Wang, and K. Abbas, *A survey on deep learning and its applications*, *Computer Science Review*, vol. 40, p. 100 379, 2021, ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2021.100379.

[50] M. Dorigo, M. Birattari, and T. Stutzle, *Ant colony optimization*, *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[51] W. E, *Machine learning: Mathematical theory and scientific applications*, https://web.math.princeton.edu/~weinan/ICIAM.pdf, Presentation at ICIAM—International Congress on Industrial and Applied Mathematics. Accessed 12 Sep. 2023, 2019.

[52] O. Eluyode and D. T. Akomolafe, *Comparative study of biological and artificial neural networks*, *European Journal of Applied Engineering and Scientific Research*, vol. 2, no. 1, pp. 36–46, 2013.

[53] A. Fischer and C. Igel, *An introduction to restricted boltzmann machines*, in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings 17*, Springer, 2012, pp. 14–36.

[54] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[55] J. N. Fuhg, A. Fau, and U. Nackenhorst, *State-of-the-art and comparative review of adaptive sampling methods for kriging*, *Archives of Computational Methods in Engineering*, vol. 28, pp. 2689–2747, 2021.

[56] K. Fukushima, *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*, *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[57] N. Gabdullin, S. Madanzadeh, and A. Vilkin, *Towards end-to-end deep learning performance analysis of electric motors*, *Actuators*, vol. 10, no. 2, 10.3390/act10020028, 2021.

[58] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, in *international conference on machine learning*, PMLR, 2016, pp. 1050–1059.

[59] X. Gandibleux and M. Ehrgott, *1984-2004–20 years of multiobjective metaheuristics. but what about the solution of combinatorial problems with multiple objectives?*, in *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Proceedings 3*, Springer, 2005, pp. 33–46.

[60] N. Georg, *Surrogate modeling and uncertainty quantification for radio frequency and optical applications*, en, Ph.D. dissertation, Technische Universität, Darmstadt, 2022, xiii, 136 Seiten. DOI: https://doi.org/10.26083/tuprints-00021149.

[61] D. Gerling, *Induction machines*, in *Electrical Machines: Mathematical Fundamentals of Machine Topologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 135–188, ISBN: 978-3-642-17584-8. DOI: 10.1007/978-3-642-17584-8_4.

[62] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities*, *International journal for numerical methods in engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[63] B. Ghojogh, M. Crowley, F. Karray, and A. Ghodsi, *Variational autoencoders*, in *Elements of Dimensionality Reduction and Manifold Learning*. Cham: Springer International Publishing, 2023, pp. 563–576, ISBN: 978-3-031-10602-6. DOI: 10.1007/978-3-031-10602-6_20.

[64] J. Gieras, *PERMANENT MAGNET MOTOR TECHNOLOGY: DESIGN AND APPLICATIONS*. 2010, ISBN: 978-1-4200-6440-7.

[65] J. F. Gieras, *Electric motors for medical and clinical applications*, in *Advancements in Electric Machines*. Dordrecht: Springer Netherlands, 2008, pp. 135–156, ISBN: 978-1-4020-9007-3. DOI: 10.1007/978-1-4020-9007-3_6.

[66] C. Gletter, A. Mayer, J. Kallo, T. Winsel, and O. Nelles, *A novel approach for development of neural network based electrical machine models for hev system-level design optimization.*, in *VEHITS*, 2019, pp. 17–24. DOI: 10.5220/0007570300170024.

[67] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[68] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[69] R. Gómez-Bombarelli *et al.*, *Automatic chemical design using a data-driven continuous representation of molecules*, *ACS Central Science*, vol. 4, no. 2, pp. 268–276, 2018. DOI: 10.1021/acscentsci.7b00572.

[70] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[71] I. Goodfellow *et al.*, *Generative adversarial nets*, *Advances in neural information processing systems*, vol. 27, 2014.

[72] J. Gu, W. Hua, W. Yu, Z. Zhang, and H. Zhang, *Surrogate models-based multi-objective optimization of high-speed pm synchronous machine: Construction and comparison*, *IEEE Transactions on Transportation Electrification*, pp. 1–1, 2022. DOI: 10.1109/TTE.2022.3173940.

[73] S. Gu, E. Holly, T. Lillicrap, and S. Levine, *Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates*, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3389–3396. DOI: 10.1109/ICRA.2017.7989385.

[74] I. Guide *et al.*, *Guide to the expression of uncertainty in measurement*, *International Standard Organisation, Geneva*, 1993.

[75] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[76] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[77] M. Heroth, H. C. Schmid, R. Herrler, and W. Hofmann, *Image-based optimization of electrical machines using generative adversarial networks*, in *2023 IEEE International Electric Machines & Drives Conference (IEMDC)*, 2023, pp. 1–5. DOI: 10.1109/IEMDC55163.2023.10239041.

[78] G. E. Hinton, S. Osindero, and Y.-W. Teh, *A fast learning algorithm for deep belief nets*, *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[79] D.-K. Hong, W. Hwang, J.-Y. Lee, and B.-C. Woo, *Design, analysis, and experimental validation of a permanent magnet synchronous motor for articulated robot applications*, *IEEE Transactions on Magnetics*, vol. 54, no. 3, pp. 1–4, 2018. DOI: 10.1109/TMAG.2017.2752080.

[80] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

[81] L. Huang, Z. Zhu, and W. Chu, *Optimization of electrically excited synchronous machine for electrical vehicle applications*, in *8th IET International Conference on Power Electronics, Machines and Drives (PEMD 2016)*, 2016, pp. 1–6. DOI: 10.1049/cp.2016.0204.

[82] Z. Huang and J. Fang, *Multiphysics design and optimization of high-speed permanent-magnet electrical machines for air blower applications*, *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 2766–2774, 2016. DOI: 10.1109/TIE.2016.2518121.

[83] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[84] A. Jabbar, X. Li, and B. Omar, *A survey on generative adversarial networks: Variants, applications, and training*, *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–49, 2021.

[85] C. Jacobina, M. de Rossiter Correa, E. da Silva, and A. Lima, *Induction motor drive system for low-power applications*, *IEEE Transactions on Industry Applications*, vol. 35, no. 1, pp. 52–61, 1999. DOI: 10.1109/28.740845.

[86] J. Jegan and I. Karuppasamy, *Simulation and validation of permanent magnet synchronous motor drives using reinforcement learning*, in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, IEEE, 2023, pp. 1–5.

[87] L. Jin, F. Wang, and Q. Yang, *Performance analysis and optimization of permanent magnet synchronous motor based on deep learning*, in *2017 20th International Conference on Electrical Machines and Systems (ICEMS)*, 2017, pp. 1–5. DOI: 10.1109/ICEMS.2017.8056321.

[88] R. Jin, W. Chen, and T. W. Simpson, *Comparative studies of metamodelling techniques under multiple modelling criteria*, *Structural and multidisciplinary optimization*, vol. 23, pp. 1–13, 2001.

[89] T. Jokinen, V. Hrabovcova, and J. Pyrhonen, *Design of rotating electrical machines*. John Wiley & Sons, 2013.

[90] S. Jozdani, D. Chen, D. Pouliot, and B. Alan Johnson, *A review and meta-analysis of generative adversarial networks and their applications in remote sensing*, *International Journal of Applied Earth Observation and Geoinformation*, vol. 108, p. 102 734, 2022, ISSN: 1569-8432. DOI: https://doi.org/10.1016/j.jag.2022.102734.

[91] S. Kalt, J. Erhard, and M. Lienkamp, *Electric machine design tool for permanent magnet synchronous machines and induction machines*, *Machines*, vol. 8, no. 1, 2020, ISSN: 2075-1702.

[92] I.-H. Kao, W.-J. Wang, Y.-H. Lai, and J.-W. Perng, *Analysis of permanent magnet synchronous motor fault diagnosis based on learning*, *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 2, pp. 310–324, 2019. DOI: 10.1109/TIM.2018.2847800.

[93] J. Kennedy and R. Eberhart, *Particle swarm optimization*, in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[94] 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. DOI: 10.1109/TMAG.2019.2957162.

[95] A. Khan, *Statistical methods for design and analysis of electrical machines*, en, Ph.D. dissertation, McGill University, 2022.

[96] A. Khan, V. Ghorbanian, and D. Lowther, *Deep learning for magnetic field estimation*, *IEEE Transactions on Magnetics*, vol. 55, no. 6, pp. 1–4, 2019. DOI: 10.1109/TMAG.2019.2899304.

[97] A. Khan and D. A. Lowther, *Physics informed neural networks for electromagnetic analysis*, *IEEE Transactions on Magnetics*, vol. 58, no. 9, pp. 1–4, 2022.

[98] A. Khan and D. A. Lowther, *Machine learning applied to the design and analysis of low frequency electromagnetic devices*, in *2020 21st International Symposium on Electrical Apparatus and Technologies (SIELA)*, 2020, pp. 1–4. DOI: 10.1109/SIELA49118.2020.9167158.

[99] A. Khan, C. Midha, and D. Lowther, *Reinforcement learning for topology optimization of a synchronous reluctance motor*, *IEEE Transactions on Magnetics*, vol. 58, no. 9, pp. 1–4, 2022. DOI: 10.1109/TMAG.2022.3184246.

[100] A. Khan, M. H. Mohammadi, V. Ghorbanian, and D. Lowther, *Transfer learning for efficiency map prediction*, in *2020 IEEE 19th Biennial Conference on Electromagnetic Field Computation (CEFC)*, 2020, pp. 1–4. DOI: 10.1109/CEFC46938.2020.9451362.

[101] W. Kim, A. Kanezaki, and M. Tanaka, *Unsupervised learning of image segmentation based on differentiable feature clustering*, *IEEE Transactions on Image Processing*, vol. 29, pp. 8055–8068, 2020. DOI: 10.1109/TIP.2020.3011269.

[102] D. P. Kingma, M. Welling, *et al.*, *An introduction to variational autoencoders*, *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[103] D. P. Kingma and M. Welling, *Auto-encoding variational Bayes*, in *International Conference on Learning Representations*, 2014.

[104] D. P. Kingma and J. L. Ba, *Adam: A method for stochastic optimization*, in *International Conference on Learning Representations*, 2015.

[105] D. P. Kingma and M. Welling, *An introduction to variational autoencoders*, *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019. DOI: 10.1561/2200000056.

[106] B. R. Kiran *et al.*, *Deep reinforcement learning for autonomous driving: A survey*, *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022. DOI: 10.1109/TITS.2021.3054625.

[107] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, *1d convolutional neural networks and applications: A survey*, *Mechanical systems and signal processing*, vol. 151, p. 107 398, 2021.

[108] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, *1-d convolutional neural networks for signal processing applications*, in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8360–8364. DOI: 10.1109/ICASSP.2019.8682194.

[109] *Pearson's correlation coefficient*, in *Encyclopedia of Public Health*, W. Kirch, Ed. Dordrecht: Springer Netherlands, 2008, pp. 1090–1091, ISBN: 978-1-4020-5614-7. DOI: 10.1007/978-1-4020-5614-7_2569.

[110] W. Kirchgässner, O. Wallscheid, and J. Böcker, *Deep residual convolutional and recurrent neural networks for temperature estimation in permanent magnet synchronous motors*, in *2019 IEEE International Electric Machines Drives Conference (IEMDC)*, 2019, pp. 1439–1446. DOI: 10.1109/IEMDC.2019.8785109.

[111] S. Koziel and L. Leifsson, *Surrogate-Based Modeling and Optimization: Applications in Engineering*. Springer Science & Business Media, 2013.

[112] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017, ISSN: 0001-0782. DOI: 10.1145/3065386.

[113] I. Kulchytska-Ruchka, *Parallel-in-time simulation of electromagnetic energy converters*, en, Ph.D. dissertation, Technische Universität Darmstadt, Darmstadt, 2021, xii, 137 Seiten. DOI: https://doi.org/10.26083/tuprints-00019280.

[114] N. Kumar, S. Sonowal, and Nishant, *Email spam detection using machine learning algorithms*, in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2020, pp. 108–113. DOI: 10.1109/ICIRCA48905.2020.9183098.

[115] Y. Kumar, K. Kaur, and G. Singh, *Machine learning aspects and its applications towards different research areas*, in *2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, 2020, pp. 150–156. DOI: 10.1109/ICCAKM46823.2020.9051502.

[116] H. Kurtović and I. Hahn, *Neural network meta-modeling and optimization of flux switching machines*, in *2019 IEEE International Electric Machines & Drives Conference (IEMDC)*, 2019, pp. 629–636. DOI: 10.1109/IEMDC.2019.8785344.

[117] S. Kurz *et al.*, *Hybrid modeling: Towards the next level of scientific computing in engineering*, *Journal of Mathematics in Industry*, vol. 12, no. 1, pp. 1–12, 2022.

[118] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[119] Y. LeCun *et al.*, *Backpropagation applied to handwritten zip code recognition*, *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989. DOI: 10.1162/neco.1989.1.4.541.

[120] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, *nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: https://doi.org/10.1038/nature14539.

[121] C. Lee and W. Ha, *Optimal design of ipm rotor shape using generative adversarial networks*, in *2021 24th International Conference on Electrical Machines and Systems (ICEMS)*, IEEE, 2021, pp. 2440–2444.

[122] J. Lee, Y.-J. Jeon, D.-c. Choi, S. Kim, and S. W. Kim, *Demagnetization fault diagnosis method for pmsm of electric vehicle*, in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 2013, pp. 2709–2713. DOI: 10.1109/IECON.2013.6699559.

[123] S. Lee, Y.-J. Kim, and S.-Y. Jung, *Numerical investigation on torque harmonics reduction of interior pm synchronous motor with concentrated winding*, *IEEE Transactions on Magnetics*, vol. 48, no. 2, pp. 927–930, 2012. DOI: 10.1109/TMAG.2011.2174346.

[124] G. Lei, J. Zhu, Y. Guo, C. Liu, and B. Ma, *A review of design optimization methods for electrical machines*, *Energies*, vol. 10, no. 12, 2017, ISSN: 1996-1073. DOI: 10.3390/en10121962.

[125] F.-F. Li, Y. Li, and R. Gao, *Cs231n: Convolutional neural networks for visual recognition*, https://cs231n.github.io/convolutional-networks/, Accessed: 6th October 2023, Stanford University, 2023.

[126] L. Li *et al.*, *A system for massively parallel hyperparameter tuning*, in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 230–246.

[127] M. Li, F. Gabriel, M. Alkadri, and D. A. Lowther, *Kriging-assisted multi-objective design of permanent magnet motor for position sensorless control*, *IEEE Transactions on Magnetics*, vol. 52, no. 3, pp. 1–4, 2016. DOI: 10.1109/TMAG.2015.2491301.

[128] Y. Li, T. Sun, W. Zhang, S. Li, J. Liang, and Z. Wang, *A torque observer for ipmsm drives based on deep neural network*, in *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2019, pp. 1530–1535. DOI: 10.1109/ICIEA.2019.8834195.

[129] Y. Li, S. Ng, M. Xie, and T. Goh, *A systematic comparison of metamodeling techniques for simulation optimization in decision support systems*, *Applied Soft Computing*, vol. 10, no. 4, pp. 1257–1273, 2010, Optimisation Methods & Applications in Decision-Making Processes, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2009.11.034.

[130] S. Liang and R. Srikant, *Why deep neural networks for function approximation*, in *ICLR 2017 : International Conference on Learning Representations 2017*, 2016.

[131] X.-Q. Liu, H.-Y. Zhang, J. Liu, and J. Yang, *Fault detection and diagnosis of permanent-magnet dc motor based on parameter estimation and neural network*, *IEEE Transactions on Industrial Electronics*, vol. 47, no. 5, pp. 1021–1030, 2000. DOI: 10.1109/41.873210.

[132] X. Liu, J. Du, and D. Liang, *Analysis and speed ripple mitigation of a space vector pulse width modulation-based permanent magnet synchronous motor with a particle swarm optimization algorithm*, *Energies*, vol. 9, no. 11, p. 923, 2016.

[133] B. Macukow, *Neural networks–state of art, brief history, basic models and architecture*, in *Computer Information Systems and Industrial Management: 15th IFIP TC8 International Conference, CISIM 2016, Vilnius, Lithuania, September 14-16, 2016, Proceedings 15*, Springer, 2016, pp. 3–14.

[134] G. Mademlis, Y. Liu, J. Tang, L. Boscaglia, and N. Sharma, *Performance evaluation of electrically excited synchronous machine compared to pmsm for high-power traction drives*, in *2020 International Conference on Electrical Machines (ICEM)*, vol. 1, 2020, pp. 1793–1799. DOI: 10.1109/ICEM49940.2020.9270852.

[135] A. Mahmoudi, W. L. Soong, G. Pellegrino, and E. Armando, *Efficiency maps of electrical machines*, in *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2015, pp. 2791–2799. DOI: 10.1109/ECCE.2015.7310051.

[136] W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943. DOI: 10.1007/BF02478259.

[137] M. Mckay, R. Beckkman, and W. Conover, *Comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, *Technometrics*, vol. 21, pp. 266–294, 2000. DOI: 10.1080/00401706.2000.10485979.

[138] M. Merkel, P. Gangl, and S. Schöps, *Shape optimization of rotating electric machines using isogeometric analysis*, *IEEE Transactions on Energy Conversion*, vol. 36, no. 4, pp. 2683–2690, 2021. DOI: 10.1109/TEC.2021.3061271.

[139] D. Mishra and P. Joshi, *A comprehensive study on weather forecasting using machine learning*, in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2021, pp. 1–5. DOI: 10.1109/ICRITO51393.2021.9596117.

[140] M. H. Mohammadi and D. A. Lowther, *A computational study of efficiency map calculation for synchronous ac motor drives including cross-coupling and saturation effects*, *IEEE Transactions on Magnetics*, vol. 53, no. 6, pp. 1–4, 2017. DOI: 10.1109/TMAG.2017.2661994.

[141] J. L. Morales and J. Nocedal, *Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization"*, *ACM Trans. Math. Softw.*, vol. 38, no. 7, 2011, ISSN: 0098-3500. DOI: 10.1145/2049662.2049669.

[142] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[143] A. Nakahara, K. Deguchi, S. Kikuchi, and Y. Enomoto, *Comparative electrical design of radial- and axial-flux permanent magnet synchronous machines under space limitation*, in *2014 International Conference on Electrical Machines (ICEM)*, 2014, pp. 422–428. DOI: 10.1109/ICELMACH.2014.6960215.

[144] A. Nawrocka, A. Kot, and M. Nawrocki, *Application of machine learning in recommendation systems*, in *2018 19th International Carpathian Control Conference (ICCC)*, 2018, pp. 328–331. DOI: 10.1109/CarpathianCC.2018.8399650.

[145] M. Nell, J. Lenz, and K. Hameyer, *Scaling laws for the fe solutions of induction machines*, *Archives of electrical engineering*, vol. 68, no. 3, pp. 677–695, 2019.

[146] F. Nishanth and B. Wang, *Topology optimization of electric machines: A review*, in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2022, pp. 1–8. DOI: 10.1109/ECCE50734.2022.9948073.

[147] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[148] P. Offermann, H. Mac, T. T. Nguyen, S. Clénet, H. De Gersem, and K. Hameyer, *Uncertainty quantification and sensitivity analysis in electrical machines with stochastically varying machine parameters*, *IEEE Transactions on Magnetics*, vol. 51, no. 3, pp. 1–4, 2015. DOI: 10.1109/TMAG.2014.2354511.

[149] F. Oliveira and A. Ukil, *Comparative performance analysis of induction and synchronous reluctance motors in chiller systems for energy efficient buildings*, *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4384–4393, 2019. DOI: 10.1109/TII.2018.2890270.

[150] M. Onsal, Y. Demir, and M. Aydin, *A new nine-phase permanent magnet synchronous motor with consequent pole rotor for high-power traction applications*, *IEEE Transactions on Magnetics*, vol. 53, no. 11, pp. 1–6, 2017. DOI: 10.1109/TMAG.2017.2709788.

[151] V. Parekh, D. Flore, and S. Schöps, *Deep learning-based meta-modeling for multi-objective technology optimization of electrical machines*, *IEEE Access*, vol. 11, pp. 93 420–93 430, 2023. DOI: 10.1109/ACCESS.2023.3307499.

[152] V. Parekh, D. Flore, and S. Schöps, *Deep learning-based prediction of key performance indicators for electrical machines*, *IEEE Access*, vol. 9, pp. 21 786–21 797, 2021. DOI: 10.1109/ACCESS.2021.3053856.

[153] V. Parekh, D. Flore, and S. Schöps, *Performance analysis of electrical machines using a hybrid data- and physics-driven model*, *IEEE Transactions on Energy Conversion*, pp. 1–10, 2022. DOI: 10.1109/TEC.2022.3209103.

[154] V. Parekh, 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. DOI: 10.1109/TMAG.2022.3163972.

[155] V. Parekh, D. Flore, S. Schöps, and P. Theisinger, *Multi-objective optimization of electrical machines using a hybrid data-and physics-driven approach*, *arXiv preprint arXiv:2306.09096*, 2023.

[156] R. H. Park, *Two-reaction theory of synchronous machines generalized method of analysis-part i*, *Transactions of the American Institute of Electrical Engineers*, vol. 48, no. 3, pp. 716–727, 1929. DOI: 10.1109/T-AIEE.1929.5055275.

[157] V. L. Parsons, *Stratified sampling*, *Wiley StatsRef: Statistics Reference Online*, pp. 1–11, 2014.

[158] A. Paszke *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.

[159] F. Pedregosa *et al.*, *Scikit-learn: Machine learning in python*, *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[160] R. Pedrosa Silva, *Surrogate problem evaluation and selection for optimization with expensive function evaluations*, en, Ph.D. dissertation, McGill University, 2018.

[161] G. Pellegrino, A. Vagati, B. Boazzo, and P. Guglielmi, *Comparison of induction and pm synchronous motor drives for ev application including design examples*, *IEEE Transactions on Industry Applications*, vol. 48, no. 6, pp. 2322–2332, 2012. DOI: 10.1109/TIA.2012.2227092.

[162] G. Pellegrino, A. Vagati, P. Guglielmi, and B. Boazzo, *Performance comparison between surface-mounted and interior pm motor drives for electric vehicle application*, *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 803–811, 2012. DOI: 10.1109/TIE.2011.2151825.

[163] P. Pietrzak, M. Wolkiewicz, and T. Orlowska-Kowalska, *Pmsm stator winding fault detection and classification based on bispectrum analysis and convolutional neural network*, *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 5192–5202, 2023. DOI: 10.1109/TIE.2022.3189076.

[164] A. Poornima and K. S. Priya, *A comparative sentiment analysis of sentence embedding using machine learning techniques*, in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 493–496. DOI: 10.1109/ICACCS48705.2020.9074312.

[165] M. Popescu, *Prediction of the electromagnetic torque in synchronous machines through maxwell stress harmonic filter (hft) method*, *Electrical Engineering*, vol. 89, pp. 117–125, 2006.

[166] S. Pouyanfar *et al.*, *A survey on deep learning: Algorithms, techniques, and applications*, *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[167] Y. Pu *et al.*, *Variational autoencoder for deep learning of images, labels and captions*, in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016, pp. 2352–2360.

[168] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[169] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006, pp. 7–30, ISBN: 026218253X.

[170] L. Regenwetter, A. H. Nobari, and F. Ahmed, *Deep generative models in engineering design: A review*, *Journal of Mechanical Design*, vol. 144, no. 7, p. 071 704, 2022.

[171] M. Reinlein, T. Hubert, A. Hoffmann, and A. Kremser, *Optimization of analytical iron loss approaches for electrical machines*, in *2013 3rd International Electric Drives Production Conference (EDPC)*, 2013, pp. 1–7. DOI: 10.1109/EDPC.2013.6689759.

[172] C. A. Rivera, J. Poza, G. Ugalde, and G. Almandoz, *A knowledge based system architecture to manage and automate the electrical machine design process*, in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, 2017, pp. 1–6. DOI: 10.1109/ECMSM.2017.7945875.

[173] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.*, *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[174] M. Rosu *et al.*, *Basics of Electrical Machines Design and Manufacturing Tolerances*. John Wiley & Sons, Ltd, 2018, ch. 1, pp. 1–43, ISBN: 9781119103462. DOI: https://doi.org/10.1002/9781119103462.ch1.

[175] S. Ruder, *An overview of gradient descent optimization algorithms, 2016*, *arXiv preprint arXiv:1609.04747*, 2016.

[176] D. E. Rumelhart, *Learning internal representations by error propagation, in parallel distributed processing*, *Explorations in the Microstructure of Cognition*, pp. 318–362, 1986.

[177] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, *nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0.

[178] Y. Saad and M. H. Schultz, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[179] R. Sabir, D. Rosato, S. Hartmann, and C. Gühmann, *Signal generation using 1d deep convolutional generative adversarial networks for fault diagnosis of electrical machines*, in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 3907–3914. DOI: 10.1109/ICPR48806.2021.9413119.

[180] M. Sadiku and A. Peterson, *A comparison of numerical methods for computing electromagnetic fields*, in *IEEE Proceedings on Southeastcon*, 1990, 42–47 vol.1. DOI: 10.1109/SECON.1990.117766.

[181] C. Sain, A. Banerjee, P. K. Biswas, T. S. Babu, and K. Balasubramanian, *Different control mechanisms of a pmsm drive for electrified transportation—a survey*, in *Communication and Control for Robotic Systems*, J. Gu, R. Dey, and N. Adhikary, Eds. Singapore: Springer Singapore, 2022, pp. 395–405, ISBN: 978-981-16-1777-5. DOI: 10.1007/978-981-16-1777-5_25.

[182] S. Salon and J. D'Angelo, *Applications of the hybrid finite element-boundary element method in electromagnetics*, *IEEE Transactions on Magnetics*, vol. 24, no. 1, pp. 80–85, 1988. DOI: 10.1109/20.43861.

[183] S. Salon and M. Chari, *Numerical methods in electromagnetism*. Elsevier, 1999.

[184] S. J. Salon, *Finite Element Analysis of Electrical Machines*. Kluwer, 1995.

[185] I. H. Sarker, *Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions*, *SN Computer Science*, vol. 2, no. 6, p. 420, 2021.

[186] H. Sasaki and H. Igarashi, *Topology optimization accelerated by deep learning*, *IEEE Transactions on Magnetics*, vol. 55, no. 6, pp. 1–5, 2019. DOI: 10.1109/TMAG.2019.2901906.

[187] H. Sasaki and H. Igarashi, *Topology optimization of ipm motor with aid of deep learning*, *International Journal of Applied Electromagnetics and Mechanics*, vol. 59, no. 1, pp. 87–96, 2019. DOI: 10.3233/JAE-171164.

[188] A. Saseendran, K. Skubch, S. Falkner, and M. Keuper, *Shape your space: A gaussian mixture regularization approach to deterministic autoencoders*, *Advances in Neural Information Processing Systems*, vol. 34, pp. 7319–7332, 2021.

[189] 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, 2023. DOI: 10.1109/TMAG.2023.3242288.

[190] H. Sato and H. Igarashi, *Visual interpretation of topology optimization results based on deep learning*, *IEEE Transactions on Magnetics*, 2023.

[191] S. Schöps, H. De Gersem, and T. Weiland, *Winding functions in transient magnetoquasistatic field-circuit coupled simulations*, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 32, no. 6, pp. 2063–2083, 2013. DOI: 10.1108/COMPEL-01-2013-0004.

[192] A. Shapiro, *Monte carlo sampling methods*, in *Stochastic Programming*, ser. Handbooks in Operations Research and Management Science, vol. 10, Elsevier, 2003, pp. 353–425. DOI: https://doi.org/10.1016/S0927-0507(03)10006-0.

[193] M. El-Sharkawi, A. El-Samahy, and M. El-Sayed, *High performance drive of dc brushless motors using neural network*, *IEEE Transactions on Energy Conversion*, vol. 9, no. 2, pp. 317–322, 1994. DOI: 10.1109/60.300142.

[194] J. Shen, X. Qin, and Y. Wang, *High-speed permanent magnet electrical machines — applications, key issues and challenges*, *CES Transactions on Electrical Machines and Systems*, vol. 2, no. 1, pp. 23–33, 2018. DOI: 10.23919/TEMS.2018.8326449.

[195] T. Shen, A. Kilic, C. Thulfaut, and H.-C. Reuss, *An intelligent diagnostic method for permanent magnet synchronous motors (pmsm) in the electric drive of autonomous vehicles*, in *2019 21st European Conference on Power Electronics and Applications (EPE '19 ECCE Europe)*, 2019, P.1–P.10. DOI: 10.23919/EPE.2019.8915161.

[196] M. D. Shields and J. Zhang, *The generalization of latin hypercube sampling*, *Reliability Engineering & System Safety*, vol. 148, pp. 96–108, 2016.

[197] 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. DOI: 10.1109/TEC.2022.3208129.

[198] D. Silver *et al.*, *A general reinforcement learning algorithm that masters chess, shogi, and go through self-play*, *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[199] D. Silver *et al.*, *Mastering the game of go with deep neural networks and tree search*, *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[200] G. D. Smith and G. D. Smith, *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

[201] S. Son, H. Lee, D. Jeong, K.-Y. Oh, and K. H. Sun, *A novel physics-informed neural network for modeling electromagnetism of a permanent magnet synchronous motor*, *Advanced Engineering Informatics*, vol. 57, p. 102 035, 2023.

[202] D. Specht, *A general regression neural network*, *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991. DOI: 10.1109/72.97934.

[203] N. Srinivas and K. Deb, *Muiltiobjective optimization using nondominated sorting in genetic algorithms*, *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[204] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[205] C. P. Steinmetz, *On the law of hysteresis (originally published 1892)*, *Proceedings of the IEEE*, vol. 72, pp. 197–221, 1984.

[206] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, *Policy gradient methods for reinforcement learning with function approximation*, in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[207] J. K. Sykulski, *New trends in optimization in electromagnetics*, in *2008 IET 7th International Conference on Computation in Electromagnetics*, 2008, pp. 44–49. DOI: 10.1049/cp:20080215.

[208] T. Szandała, *Review and comparison of commonly used activation functions for deep neural networks*, *Bio-inspired neurocomputing*, pp. 203–224, 2021.

[209] C. Szegedy *et al.*, *Going deeper with convolutions*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[210] N. Tajbakhsh *et al.*, *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. DOI: 10.1109/TMI.2016.2535302.

[211] A. R. Tariq, C. E. Nino-Baron, and E. G. Strangas, *Consideration of magnet materials in the design of pmsms for hevs application*, in *2011 IEEE Power and Energy Society General Meeting*, 2011, pp. 1–6. DOI: 10.1109/PES.2011.6039824.

[212] J. Tomczak and M. Welling, *Vae with a vampprior*, in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2018, pp. 1214–1223.

[213] J. Treboux and D. Genoud, *Improved machine learning methodology for high precision agriculture*, in *2018 Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6. DOI: 10.1109/GIOTS.2018.8534558.

[214] M. Tucci, S. Barmada, A. Formisano, and D. Thomopulos, *A regularized procedure to generate a deep learning model for topology optimization of electromagnetic devices*, *Electronics*, vol. 10, no. 18, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10182185.

[215] E. S. Tumpa and K. Dey, *A review on applications of machine learning in healthcare*, in *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2022, pp. 1388–1392. DOI: 10.1109/ICOEI53556.2022.9776844.

[216] A. M. Turing, *Computing machinery and intelligence*, *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[217] M. Usama *et al.*, *Unsupervised machine learning for networking: Techniques, applications and research challenges*, *IEEE Access*, vol. 7, pp. 65 579–65 615, 2019. DOI: 10.1109/ACCESS.2019.2916648.

[218] N. Uzhegov, A. Smirnov, C. H. Park, J. H. Ahn, J. Heikkinen, and J. Pyrhönen, *Design aspects of high-speed electrical machines with active magnetic bearings for compressor applications*, *IEEE Transactions on Industrial Electronics*, vol. 64, no. 11, pp. 8427–8436, 2017. DOI: 10.1109/TIE.2017.2698408.

[219] S. Vishwakarma, A. Kumar, and A. Vishwakarma, *Torque estimation of permanent magnet synchronous motor (pmsm) using 1d convolutional neural network*, in *2022 IEEE 6th Conference on Information and Communication Technology (CICT)*, 2022, pp. 1–5. DOI: 10.1109/CICT56698.2022.9997927.

[220] M.-S. Wang, M.-F. Hsieh, and H.-Y. Lin, *Operational improvement of interior permanent magnet synchronous motor using fuzzy field-weakening control*, *Electronics*, vol. 7, no. 12, p. 452, 2018, ISSN: 2079-9292. DOI: 10.3390/electronics7120452.

[221] S. Wang, Y. Teng, and P. Perdikaris, *Understanding and mitigating gradient flow pathologies in physics-informed neural networks*, *SIAM Journal on Scientific Computing*, vol. 43, no. 5, A3055–A3081, 2021.

[222] S. Weerasooriya and M. El-Sharkawi, *Identification and control of a dc motor using back-propagation neural networks*, *IEEE Transactions on Energy Conversion*, vol. 6, no. 4, pp. 663–669, 1991. DOI: 10.1109/60.103639.

[223] B. Wilamowski, B. Wu, and J. Korniak, *Big data and deep learning*, in *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)*, IEEE, 2016, pp. 11–16.

[224] C. J. Willmott and K. Matsuura, *Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance*, *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[225] M. Wlas, Z. Krzeminski, and H. A. Toliyat, *Neural-network-based parameter estimations of induction motors*, *IEEE Transactions on Industrial Electronics*, vol. 55, no. 4, pp. 1783–1794, 2008. DOI: 10.1109/TIE.2008.918615.

[226] S. Xiao, R. Mihai, and J. Sykulski, *Exploration versus exploitation using kriging surrogate modelling in electromagnetic design*, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 31, pp. 1541–1551, 2012. DOI: 10.1108/03321641211248291.

[227] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010, ISBN: 978-0-691-14212-8.

[228] Y.-B. Yan, J.-N. Liang, T.-F. Sun, J.-P. Geng, Gang-Xie, and D.-J. Pan, *Torque estimation and control of pmsm based on deep learning*, in *2019 22nd International Conference on Electrical Machines and Systems (ICEMS)*, 2019, pp. 1–6. DOI: 10.1109/ICEMS.2019.8921886.

[229] L. Yang and A. Shami, *On hyperparameter optimization of machine learning algorithms: Theory and practice*, *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[230] Z. Yang, F. Shang, I. P. Brown, and M. Krishnamurthy, *Comparative study of interior permanent magnet, induction, and switched reluctance motor drives for ev and hev applications*, *IEEE Transactions on Transportation Electrification*, vol. 1, no. 3, pp. 245–254, 2015. DOI: 10.1109/TTE.2015.2470092.

[231] Y. Yongmin, *Multi-objective optimal design of permanent magnet synchronous motor for electric vehicle based on deep learning*, *Applied Sciences*, vol. 10, p. 482, 2020. DOI: 10.3390/app10020482.

[232] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, *A survey on deep learning for big data*, *Information Fusion*, vol. 42, pp. 146–157, 2018.

[233] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, *Improving deep neural networks using softplus units*, in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–4. DOI: 10.1109/IJCNN.2015.7280459.

Acknowledgements

First and foremost, I sincerely thank, Prof. Dr. rer. nat. Sebastian Schöps, for his continuous support, mentorship, and guidance throughout my doctoral research. His expertise and insights have been valuable to my scientific development and the completion of this thesis. I have also greatly appreciated his insights into the broader scientific community and his critiques of my work.

I am also very grateful to my supervisor, Dr. Dominik Flore, whose advice, mentorship, encouragement, and constructive feedback have been invaluable to this research.

To my parents, Chandrika and Mukesh, I offer my heartfelt thanks for your love, unwavering support, and for instilling in me the values and work ethic that have brought me this far. Your sacrifices and belief in me have served as the bedrock upon which my academic journey has been built.

To my wife, Srushti, your love, patience, and constant encouragement have been my source of strength. Thank you for standing by me through the highs and lows of this challenging endeavor.

I would also like to extend my gratitude to my elder sister, Ankur, nephew, Mihit, and brother-in-law, Akshay, for their warm wishes and support. Additionally, I thank my brother, Kamlesh, for his warm wishes and support. I also thank my sisters-in-law, Smruti and Shreya, nephew Vivaan, and my brother-in-law, Purvesh, for their warm wishes.

Finally, I want to sincerely thank my friends Ashvin, Vishnu, Dhruv, Gaurang, Parth, Prithviraj, Neha, Abin, Teja, Isaac, and Shashank for their support and unique contributions to this rewarding experience. Your friendship has made the journey through my doctoral studies a memorable experience.