Statistical Methods for Design and Analysis of Electrical Machines

Arbaaz Khan

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada
September 2021

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

In engineering, the process of designing a product relies heavily on computer simulations to provide accurate insights into the product performance, under different design requirements and constraints. These computer simulations play a crucial role in virtual prototyping, as one can get accurate details about the product performance, without actually building a physical structure. As such, these analyses are carried out through high-fidelity and time-consuming computer simulations. At times, these simulations can be too expensive, which can limit the efficiency of the analysis and thus the ability to explore new design possibilities. Data-driven statistical learning methods can help reduce the computation burden associated with these simulations, by either finding a cheap-to-evaluate surrogate model to approximate the simulation output or reducing the number of simulations required in the design process. In this thesis, both these two approaches are explored for the design and analysis of electromagnetic devices.

The first few chapters of this thesis investigate the feasibility of novel data-driven deep learning models to predict the solution of Maxwell's equations and performance maps for low-frequency electromagnetic devices. A deep neural network is trained in a supervised manner to learn a mapping for the magnetic field distribution for topologies of different complexities of geometry. Further, a probabilistic model is introduced to improve the accuracy and to quantify the uncertainty in the prediction, based on Monte Carlo dropout. The same methodology along with other architectures are explored for predicting performance maps of electric motor drives. The output of the proposed methods has a good match with that of the finite element solution, indicating a high prediction accuracy as well as low run-time useful for design and optimization problems. Further, this work also explores methods to extend a trained deep neural network for predicting performance maps to work on different motor drive topologies. This procedure reduces the computation cost associated with training Deep Networks by transferring knowledge over similar tasks handled by the deep networks.

The other contribution of this work proposes a new Topology Optimization method which transforms the material distribution problem into a movement sequence search problem for a controller moving in the design space. This method enforces connectivity between the cells in the discretized design domain that contain the same material. This removes the need of any filtering or smoothing of the optimal result to obtain a manufacturable design. This method also provides the ability to leverage the use of different tree search algorithms such as a data-driven Reinforcement Learning based agent which is trained on the proposed Topology Optimization based environment. Compared with conventional optimization methods, such as the evolutionary algorithms, that require high computational cost for iterative electromagnetic analyses for generating optimal designs, once the training process is finished, the deep reinforcement learning based agent is capable of yielding manufacturable solutions without any time-consuming iterative analysis process, thus reducing the number of computer simulations required for product design.

Résumé

En ingénierie, le processus de conception d'un produit repose grandement sur des simulations informatiques afin de procurer des informations précises sur les performances du produit, et ce, sous différentes exigences et contraintes de conception. Ces simulations jouent un rôle crucial dans le prototypage virtuel, car elles permettent d'obtenir des détails précis sur les performances du produit, sans pour autant devoir construire une structure physique. À ce titre, ces analyses sont effectuées par le biais de simulations informatiques haute-fidélité qui prennent beaucoup de temps. Ces simulations peuvent parfois être trop coûteuses, ce qui peut limiter l'efficacité des analyses, et par conséquent la capacité d'explorer de nouvelles possibilités de conception. Les méthodes d'apprentissage statistique basées sur les données peuvent aider à réduire la charge de calcul associée à ces simulations. On peut trouver un modèle de substitution moins coûteux pour approcher le résultat de la simulation, ou bien réduire le nombre de simulations requises dans le processus de conception. Dans cette thèse, ces deux approches sont explorées pour la conception et l'analyse de dispositifs électromagnétiques.

Les premiers chapitres de cette thèse étudient la faisabilité de nouveaux modèles d'apprentissage en profondeur basés sur les données afin de prédire la solution des équations de Maxwell et les cartes de rendement des dispositifs électromagnétiques à basse fréquence. Un réseau neuronal profond est entraîné de manière supervisée pour en connaître davantage sur la cartographie de la distribution du champ magnétique pour des topologies avec différentes complexités de géométrie. De plus, un modèle probabiliste est introduit pour en améliorer la précision et quantifier l'incertitude de la prédiction en se basant sur la perte de Monte Carlo (Monte Carlo dropout). La même méthodologie ainsi que d'autres architectures sont explorées pour prédire les cartes de rendement des entraînements de moteursélectriques. Le résultat des méthodes proposées correspond bien à celui de la solution par éléments finis. Celui-ci indique une précision de prédiction élevée ainsi qu'un temps d'exécution moindre, utile pour les problèmes de conception et d'optimisation. En outre, ce travail explore également des méthodes afin d'étendre un réseau neuronal profond entraînés pour prédire des cartes de performances afin de fonctionner sur différentes topologies d'entraînement de moteurs. Cette procédure réduit le coût de calcul associé pour former des réseaux profonds en transférant les connaissances des tâches similaires gérées par les réseaux profonds.

L'autre contribution de ce travail propose une nouvelle méthode d'optimisation de la topologie qui transforme le problème de distribution des matériaux en un problème de recherche de séquence de mouvement d'un contrôleur se déplaçant dans l'espace de conception. Cette méthode renforce la connectivité entre les cellules du domaine de conception

discrétisé qui contiennent le même matériau. Cela élimine le besoin de filtrer ou d'affiner le résultat optimal afin d'obtenir une conception fabricable. Cette méthode offre également la possibilité d'exploiter l'utilisation de différents algorithmes de recherche arborescente, tels qu'un agent utilisant l'apprentissage par renforcement basé sur les données. Celui-ci est formé sur l'environnement basé sur l'optimisation de la topologie. L'agent basé sur l'apprentissage par renforcement en profondeur est capable de produire des solutions manufacturables qui prennent peu de temps de processus d'analyse itérative, réduisant ainsi le nombre de simulations informatiques nécessaires à la conception du produit. Ce processus est favorable si on le compare aux méthodes d'optimisation conventionnelles, telles que les algorithmes évolutionnaires, demandant un coût de calcul élevé pour les analyses électromagnétiques itératives pour générer des conceptions optimales, une fois le processus de formation terminé.

Acknowledgements

With great pleasure, I would like to express my deep gratitude to my research supervisor Prof. David A. Lowther, for his guidance, patience, and encouragement. I feel honored to have got this opportunity to work under his supervision and learn from him.

I thank God Almighty for blessing me with the strength, knowledge, and opportunity to undertake this research and complete it.

I appreciate the help and support rendered to me by my colleagues Chetan Midha, Dr. Armin Salimi, Dr. Vahid Ghorbanian, Dr. Mohammad Hossain Mohammadi, and all other members of the *COMPEM* lab. The insightful discussions we had throughout this thesis helped me become a better researcher.

I thank my parents and family for their unconditional love and encouragement, which has been a source of inspiration and motivation for me throughout the years. A special thanks to my wife, who showed remarkable patience to bear with me during the most testing times.

Special thanks go to my friends Aditya Kashi, Akash Nandi, Mushfique and Shaunak Sinha for their kind words and company, which made this journey a great experience.

Contents

List of Figures

List of Tables

List of Acronyms

AC Actor-Critic

A2C Advantage Actor-Critic

AI Artificial Intelligence

ANN Artificial Neural Network

ConvNet Convolutional Neural Network

CNN Convolutional Neural Network

DNN Deep Neural Network

DL Deep Learning

EA Evolutionary Algorithms

EM Electromagnetic

FE Finite Element

FNN Feedforward neural networks

FSCW Fractional-slot Concentrated Winding

GA Genetic Algorithm

GHG Green House Gas

GPU Graphics Processing Unit

GRU Gated Recurrent Unit

G_t expected return

IPM Internal Permanent Magnet

$l.r$ learning rate

LSTM Long short-term memory

MC Monte Carlo

MEC Magnetic Equivalent Circuits

MDP Markov Decision Process

ML Machine Learning

MSE mean squared error

NL non-linear

NN Neural Network

PG Policy Gradient

QL Q-Learning

$\mathbf{Q}(s, a)$ Action value function

RNN Recurrent Neural Network

SeqTO Sequence based Topology Optimization

SynRM Synchronous Reluctance Motors

TD Temporal Difference

TL Transfer Learning

RL Reinforcement Learning

TO Topology Optimization

VDC V-design cycle

$\mathbf{V}(s)$ State value function

Chapter 1

Introduction

1.1 Motivation

The most important energy conversion devices nowadays are electric machines which includes motors and generators. They are used with different specifications, structures and ratings in various applications, such as aerospace (Bouzidi, Masmoudi, and Bianchi 2015), transportation (Bilgin et al. 2019), power generation (Saruwatari et al. 2016) and healthcare (Nicolaescu 2012). Depending on the application, different optimal performance parameters, such as efficiency (McCoy and Douglass 2014), noise and vibration (Wang et al. 2016), (Sheikman, Folden, and Francis 2014) and torque ripple (Baek et al. 2014), might be required. This makes electromagnetic analysis and optimization the basic ingredients for the development of an electric machine. The designing of an electric motor is generally a rigorous, highly knowledge- or experienced-based, and time-consuming procedure (Mayr et al. 2018). During the product development procedure, from design and optimization through to manufacturing and test stages, a common flow of product development is followed, which is also known as V-design cycle (VDC) (Sharafi 2013; Rivera et al. 2017). This product development process of electric motors consists of two main stages, namely design and manufacturing. The overview of the proposed automated design and manufacturing process for electric motor drives is illustrated in Figure 1.1, along with the principal components.

Given a machine's specification, the goal is to find a winning candidate in terms of performance and to make sure the candidate satisfies all the design criteria. The 'Design Phase' in the V-Design cycle is a critical stage since the product requirements, i.e. system- and subsystem-level, are specified in this stage to perform a simulation-based engineering design and optimization of the subsystems (motor, inverter, gearbox, battery, and charger, etc), that are then passed to the next stage in order to make sure the implemented system satisfies the initial specifications (Ghorbanian, Salimi, and Lowther 2017). There can be a

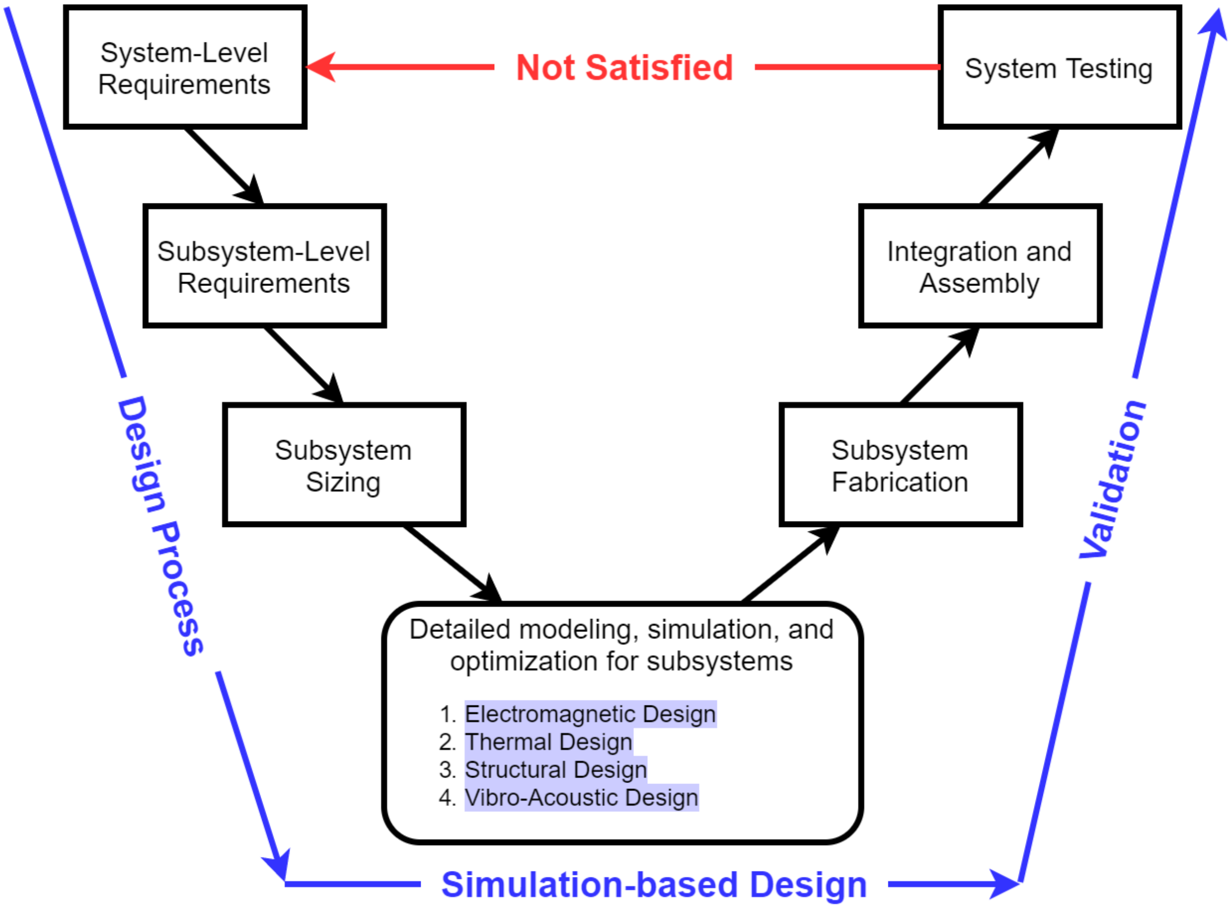


Figure 1.1: The V-Design and Manufacturing Cycle

significantly high number of possible candidates in a design space for a machine, even after applying the design and engineering constraints. Since sophisticated simulation and design optimization tools, i.e. (FE) packages, are used to analyze the multiphysics performances, the wall-clock time for finding an optimal design directly depends on how fast the simulation and optimization processes are (Tüchsen et al. 2018; Silva 2018b). The required resources, including time, money, and high-performance computing services, will increase dramatically if the process is to be repeated for all new designs or applications. In such a scenario, it is infeasible to rank all the possibilities before choosing an optimal solution due to the massive computational burden involved in the analysis through conventional techniques. Hence, only a handful of potential designs obtained through the design cycle undergo a full multi-physics simulation at any stage in the design process. Finally, a prototype is manufactured and its performance is experimentally verified.

In case the design or prototype does not meet the desired requirements, the process has to restart. This cycle can be reduced if the design space can be searched based on a multi-physics solution. However modern computer tools for analysis (at high fidelity), will take too long as there are hundreds or even thousands of design candidates. Hence, a need to formulate alternative strategies to ease the computational cost involved in the process. This has the potential to reduce the research cycle associated with product development. This thesis will look at some novel ideas for the design and analysis of electric devices and machines, ranging from surrogate models for fields and performance maps to intelligent agents for design optimization. All the data-driven methodologies discussed in this work originate from the field of statistical learning.

Machine Learning (ML) algorithms, specifically feedforward neural networks (FNN) have been shown as suitable candidates for function approximation. The application of this has been seen in the usage of neural networks as surrogates for performance evaluation metrics (such as Torque) have become prominent in the past couple of decades (Ghorbanian 2018; Wlas, Krzeminski, and Toliyat 2008). Alternatively, electrical machines can also be designed using an equivalent circuit (or lumped parameter approach). These methods can provide a fast prediction of global performance parameters such as average and ripple torque. However, these models cannot produce information on local effects within the device such as fields, losses, and stresses. Due to these complexities and limitations, big data, Artificial Intelligence (AI), Machine Learning (ML), and data mining tools seem to be promising ways to keep up with the ever-increasing demand of fast and accurate analysis of electric designs (Kißkalt et al. 2018; Khan et al. 2020a; Salimi 2018; Ghorbanian et al. 2019). This work focuses on technologies that broadly include (a) modern machine learning techniques, such as deep learning carried out with the aid of high-performance computing, and (b) intelligent generative design, such as the topology optimization for flexible and complex shape design.

1.2 Overview of Thesis Contribution

The thesis is organized into six chapters and in terms of the research methodology of AI, the work performed here can be broadly classified into two categories: supervised learning and reinforcement learning. However, the algorithms have been extended as per the requirement of the task being considered. As an example, the supervised learning task in Chapter 2 was extended to incorporate Bayesian learning to develop an indication of algorithm confidence in the prediction. Similarly, the usage of certain algorithms such as the attention mechanism was modified (as compared to (Bahdanau, Cho, and Bengio 2014)) to better suit the task of efficiency prediction in Chapter 3.

This section presents an overview of all the chapters of this thesis, along with the author's contribution.

1.2.1 Application of Deep Learning for Electric Machine Performance Analysis

For designing surrogates for the global quantities, (e.g. the torque ripple) which have a limited number of output values, usually one, for each motor design, the task is simple enough to be handled by a simple feedforward neural network. However, fully connected neural networks usually lose their attractions when it comes to many input-many output design and modeling problems, due to the required training complexity, while the same problem can be handled more efficiently using the deep Convolutional Neural Networks (CNNs) and/or Recurrent Neural Networks (RNNs). One example of such kinds of problems is modeling a local quantity of an electric motor, such as the magnetic flux density, where hundreds or thousands of numerical values distributed on the mesh structure of an FE solution should be predicted.

Chapter 2 deals with the problems of many electric devices operating at electrical power frequency (50Hz). The advantages and limitations of such methods for a variety of problems are also discussed. In addition to task compatibility, the change helped improve the speed of training (Khan, Ghorbanian, and Lowther 2019).

The author is responsible for handling the data generation along with development and validation of all DL models associated with the study. The findings of this Chapter are presented in Khan, Ghorbanian, and Lowther 2019, with other authors contributing with guidance on the research methodology, reviewing of manuscript and analysis of the results.

1.2.2 Surrogate Model for Performance Maps

Extending the usage of deep networks such as CNN, and RNN, is used for predicting performance parameters of electric motor drives (Khan et al. 2020a). The RNNs are useful when a variable number of inputs, outputs, or both exist. The efficiency, power factor, losses, and advance angle maps of an electric motor drive are clear examples of this application. Further, the usage and application of such networks are extended to predict higher dimensional output spaces such as performance maps, which will be discussed in Chapter 3. The prediction time of performance maps is reduced considerably to a few seconds for multiple geometries, against hours required for traditional approaches (Mohammadi and Lowther 2017).

Two papers (Khan et al. 2020a & Khan et al. 2020b) were published based on the results obtained in Chapter 3. The author is partially responsible for handling the data generation and solely performed the analysis and validation of components associated with DL models, along with formulating the experiment associated with transfer learning (Section 3.7). Other authors involved in this work contributed in conception of the research work, acquisition of data, interpretation of results and drafting of the article.

1.2.3 Topology Optimization

The most noticeable advantage of the Topology Optimization (TO) is its capability of free-shape optimization, which can outperform template-based design optimization routines in terms of the speed of the process if well-tuned. However, the majority of the TO techniques introduced in the literature result in solutions that are not easily manufacturable, i.e., the optimized design has checkerboard patterns, floating or stray pieces of material. To overcome this difficulty, either different filtering or smoothing methods are employed (Campelo, Watanabe, and Igarashi 2008a) or smoothness of the solution is added as an objective of the optimization process itself (Dupré et al. 2014). A new TO method (sequence-based) is proposed in Chapter 4 which ensures that the final optimized solution does not have any checkerboard patterns, perforations, or floating pieces of material. This method leverages a sequence-based environment to impose connectivity on cells containing the same material. This proposed methodology neither requires any filtering or smoothing technique nor any modification to the optimization objective function for obtaining manufacturable optimal solutions.

Building on the proposed sequence-based TO methodology, in the next chapter (Chapter 5), the concept of an intelligent agent is investigated in the field of TO by means of Deep Reinforcement Learning (DRL) algorithms. The agent is trained in an interactive manner of learning such that the topology of the geometry is basically an iterative process involving continuous interaction between an RL agent and a FE-based environment. The process involves an incentive-based generation of a number of design structures as output that satisfies the design requirements and objectives determined by the designer at the highest level of the design process.

The results from this section were published through Khan, Midha, and Lowther 2020 and Khan and Lowther 2020. Although the author was involved in all aspects of the work, research based on reinforcement learning and sequence-based TO environment formulation was carried out individually. The rest of the authors were involved in data acquisition, methodology validation, benchmarking the results and drafting of the article.

1.3 Challenges

The application of ML or DL algorithms is a budding area of research. Most of the active fields of research associated with modern AI algorithms are applied in the field of computer vision, language/signal processing, robotics, medical science. However, finding suitable applications in the field of electromagnetics, using state-of-the-art ML techniques, is not an easy task. Replication of work performed in other fields without understanding the behavior of every component of the algorithm can lead to inefficient results or misleading results. This generates a responsibility that while checking the feasibility of ML algorithms in different aspects of design and analysis of electric machines, an effort should be made to understand the limitations of the algorithms and perform knowledge extractions, rather than targeting high accuracy predictions. Some of the major issues faced during this study are discussed next.

It is well known that some of the tasks that feel intuitive for a human are very difficult for an AI agent. While analyzing the magnetic field distribution in a geometry, a trained human eye can easily identify regions of high magnetic saturation and can recommend accordingly for improvements. The lack of this intuitive understanding continues to impact the adoption of AI in the industry. Furthermore, the ability to learn from given data, explored in Chapters 2 & 3, often shows the limitation of supervised learning and is related to the limited sample data collected for training. One of the possible issues is that ML (especially deep networks) requires building rich representations of the knowledge needed to understand an environment. A task of design optimization can involve information about material distribution, field distribution, excitations, boundary conditions, and other factors that affect the performance. This results in a high dimensional space representation in the characterization of a learning task. All this requires substantial computational resources both from generating big data based on electromagnetic analysis and proper training of the DL-based learning algorithm. This constrains the ML algorithm from generalizing to the population data representation and often introduces biases in the model performance. Techniques to limit the high computation requirements and improve generalization are discussed through out the work, where deemed necessary. In Chapter 3, knowledge learnt by a neural network on a task is used to improve the performance on a different task. Similarly, in Chapters 4 and 5, a novel technique to allow the AI to interact with the electromagnetic environment opens new domains for improving the computational efficiency of a Topology Optimization task. This ability also opens frontiers to learn directly from an environment without any bias introduced by the pre-computed and collected data. Further, the AI can hold the interaction experiences and replay to learn from its memory and react accordingly to the dynamics of a physics-based environment. This is an interesting field to explore and extend the current state of the art in the field of electromagnetics.

Another important issue is the formulation of an incentive for the learning algorithm to improve its performance. For a supervised learning task (such as in Chapters 2 & 3), this can be easily established as we have access to true labels. However, for the search of an optimal geometry, the formulation is troublesome. The incentive here should be able to come up with novel designs and be able to teach itself patterns that can be easily generalized and at the same time improve its performance. Further, the AI agent should be able to learn the essentials of the new task in a relatively short period of time (through training on a new task or environment) and extend its usefulness to similar electromagnetic designing tasks.

For all the statistical learning experiments in this thesis, ground truth, labels and evaluations is gathered using a Finite element analysis (*MAGNET v2020.2*). There are various sources of errors associated with any numerical method. In case of FEA, errors can be introduced in the solution due to rounding-off, limited modeling capabilities and through insufficient discretization of the mesh. However, due to a lack of better analysis technique for handling massive design space and for the wide usage of FEA in the industry (*FEA Software Market - Global Industry Analysis* 2021), the solution obtained through a FE based simulator is accepted as the ground truth.

The purpose of this work will be to explore statistical algorithms and develop intelligent agents for aiding the design and analysis of electric machines. An emphasis will be placed on developing algorithms that are flexible enough to adapt to similar tasks and environments. This is a grand ambition but with real potential in today's age of big data, compute power, and advancements in the field of AI. With these advancements, the task looks feasible.

Chapter 2

Magnetic Field Prediction

2.1 Introduction

Data-driven methodologies are currently revolutionizing how we model, predict, and control complex systems such as climate (Knüsel and Baumberger 2020), finance (Hilpisch 2018), traffic (Lin et al. 2018), robotics and autonomy (Zhang, Han, and Deng 2018). The most pressing scientific and engineering tasks of the present era are not dependent on empirical models or derivations based on first principles (Brunton and Kutz 2019). Increasingly, researchers are turning to data-driven approaches for characterizing and modeling a diverse range of complex systems with the goal of sensing, prediction, estimation, and control. With modern mathematical methods, enabled by the unprecedented availability of data and computational resources, we are now able to tackle previously unattainable challenge. One such challenge is to analyze the performance of the electric machine in a relatively short period of time. An electromagnetic model is needed to perform the design and analysis of an electric machine (Mousavi-Aghdam et al. 2015). This model will be used to evaluate relevant electromagnetic performance characteristics associated with an electric machine such as electromagnetic force, torque, fields, and losses distribution maps. Further, a model capable of providing an accurate and fast performance evaluation is the backbone of an optimization task for achieving objectives such as minimizing the torque ripple, increasing average torque and efficiency. It also plays a central role in designing the motor drive system.

The very word 'model' implies simplification and idealization. A model is an approximation of reality and hence will not reflect all of reality. In 1976, a renowned British statistician named George Box wrote the famous line, "All models are wrong, some are useful" (Box 1979; Box 2006). The intuition behind this sentence is that every model is wrong because it is a simplification of reality. However, simplifications of reality can be quite useful. They can help us explain, predict and understand the universe and all its various components. It means useful insights can be provided from models which are not a perfect representation of the phenomena they model. The practical question is how wrong do they have to be in order to be not useful. As such, the desirable features of a model can be summarized as follows (Silva 2018a):

- A model should be consistent in its ability to explain past observations and predict future observations.
- Cost of usage: A model should be computationally cheaper than traditional methods available for analysis.
- Level of accuracy in the prediction: A model that gives highly accurate results but takes a lot of computational time may not be useful. On the other hand, a model that is not as accurate but can produce estimates very quickly can be beneficial.

Maxwell's equations form the foundation of classical electromagnetism, classical optics, and electric circuits. We can consider models built on solving physics (such as Maxwell's equations) as the First Principle models (Silva 2018a). Modeling is the mathematical representation of physical phenomenona and simulation is the numerical representation of such models on computing machines. Over the years Computer-Aided Design (CAD) has evolved as the field of "using computers to aid in the creation, modification, analysis, or optimization of a design" (Sarcar, Rao, and Narayan 2008). Accurate modeling and analysis of an electromagnetic problem is both a challenging and time-consuming process due to several reasons. First of all, the system of differential equations for the electromagnetic analysis of electric devices is nonlinear due to the property of saturation in magnetic materials. The non-linear dependence of inductances on rotor-to-stator angles is reflected in the non-linear relationship between current and fluxes. Since the induced voltage and electromagnetic torque are proportional to the state variables, such as flux, current, and speed, it makes it impossible to find an analytic solution to the machine system of differential equations (Ostovic 2012). If we neglect the effects of saturation and non-linear magnetic materials, a model will be suitable for only a fraction of a device's operating capacity. Apart from the difficulty of solving a nonlinear system of equations, finding a field solution on a complex geometry such as that of an electric machine also adds to the complexity of the task.

Having identified the challenges associated with the field of electromagnetic modeling and simulation, we can broadly divide the field into three categories of algorithms namely: Analytical, Numerical, and Magnetic Equivalent Circuits (MEC), as shown in Figure 2.1. Of the three, analytical algorithms are usually the least demanding in terms of computational need and will provide the fastest results. These methods try to find the closed-form

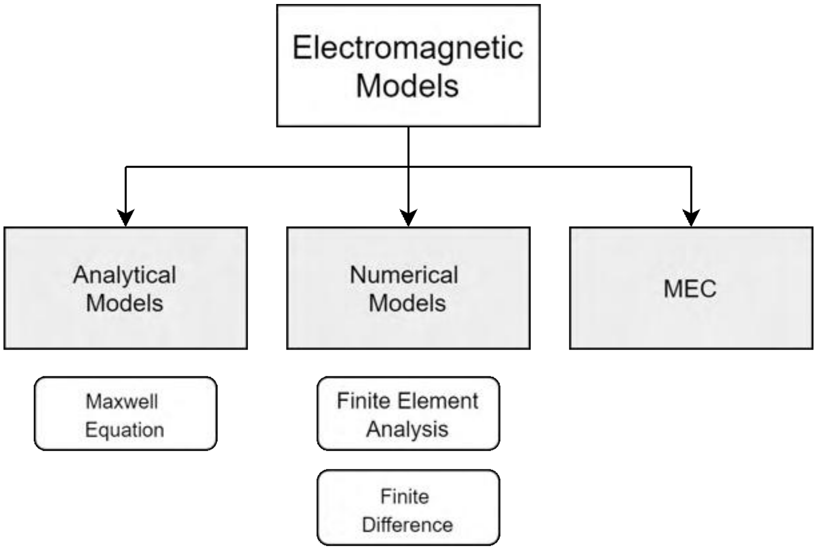


Figure 2.1: Different electromagnetic modeling techniques.

expressions for magnetic fields and losses in a motor. However, they face a major challenge in approximating complex geometries and incorporating the effect of iron saturation in their analysis (Alger 1970; Hamdi 1994), which is significant for estimating magnetic fields and losses of the motors (Sheikh-Ghalavand, Vaez-Zadeh, and Isfahani 2009). Furthermore, they are incapable of modeling skin and proximity effects in the winding and the end winding inductance. Despite their limited capabilities, analytic models are popular because of the low computation power required for their highly simplified expressions and ease of parameterization. As a result, they are employed when fast prototyping is needed and the design is evaluated only on the basis of global performance quantities such as torque and forces (Ray et al. 1984; Krishnan, Arumugan, and Lindsay 1988).

The physics-based models, on the other hand, are founded on established and detailed physical principles and lead to more accurate simulations (Salon 1995). They model the interaction of electromagnetic fields with physical objects and the environment. Two such methods are Magnetic Equivalent Circuits (MEC) and Finite Element Analysis (FEA).

The MEC modeling is a popular method in modeling electric machines (Yilmaz and Krein 2008). It is a special case since it can be considered as an analytical or numerical technique according to how it is applied. It is considered as an analytical technique if it is combined with other analytical modeling techniques such as Maxwell's equations (Uddin and Sozer 2017; Radun 2000). When the nonlinearity of the magnetic materials is considered, the MEC modeling has to be combined with numerical techniques (Ostovic 2012). In Ostovic 2012, the MEC method was applied to calculate the magnetic flux in C-core magnetic devices and also to model magnetic devices with moving parts such as electric machines.

The present-day Finite Element (FE) method based software suite allows for the accurate analysis of electric machines in three dimensions with coupled fields. It is the method of choice for today's machine designers, in comparison to analytical models and MECs. Finite Difference Methods (FDM) are another option to model electric machines. However, FDM has difficulties in modeling complex geometries; therefore, it is not directly applied to model electric machines (Hameyer and Belmans 1999).

Another class of models, surrogate or data-driven models, is not necessarily based on first principles. They approximate the relationship between the input parameters and output characteristics and reduce the repetitive bottom-up approach taken by the first principle models. Machine Learning (ML) and curve-fitting-based techniques are some of the approaches in this category of modeling used to predict the performance of electric machines. They include experimental or Finite Element (FE) model data to develop the model. These methods can provide a fast prediction of global performance parameters such as average and ripple torque (Mohammadi, Ghorbanian, and Lowther 2019; Ibrahim et al. 2020). However, these models, based on current the literature, are not capable of producing information on local effects within the device such as fields, losses, and stresses. Knowledge of the flux distribution inside the motor for different operating conditions is essential, at the design stage, to predict the performance (Watthewaduge et al. 2020). The saturation and the geometry of the motor structure make the analytical methods of field analysis difficult. This chapter will explore the field of Deep Learning (a sub-field of ML) to predict the field distribution for different electromagnetic problems. Since ML models require labeled data for training and without an accurate model, it is impossible to predict a machine's characteristics and performance. As such it is important to identify a reliable source to generate training data for the ML task. Since analytical models are not capable of capturing sufficient information for the analysis of electric machines, the discussion will be limited to only MEC and FEA, to serve as the source of data generation.

2.2 Physics-based Modeling

Before finalizing the source of labeled data, it is important to understand the capabilities and limitations of FE analysis and MEC in reference to electric machine design analysis. Based on the study, we will determine the numerical method suitable to generate the synthetic training data. The literature on FE and MEC based techniques is vast and covers the analysis of a variety of electric devices. This study is limited to only electric machines, as they are the primary motivation of this work.

2.2.1 Finite Element Methods

The FE method is applied to design and analyze electric motors and generators with various stator/rotor topologies (Mousavi-Aghdam et al. 2015; Rosu et al. 2017). It is a numerical model for solving problems of engineering and mathematical physics by formulating the problem results in a system of algebraic equations. FE analysis can be broken down into four broad steps (Jin 2015; Salon 1995):

- Domain Discretization.
- Selection of Interpolation Functions.
- Formulation of the System of Equations.
- Solution of the System of Equations.

A proper discretization optimizes the number of elements and unknowns for the desired solution accuracy because the manner in which the domain of the solution is discretized will affect the memory storage space, computation time, and the accuracy of the numerical solution.

The process of generating a system of equations is to convert a continuous operator problem to a discrete problem. This step is more involved and depends on the specific FEM method used (two popular methods are Ritz and Galerkin (Silvester and Ferrari 1996)). The whole cycle of the FEM model can be shown in Figure 2.2.

The final accuracy of the results reported in a finite element analysis model depends on many factors. One of the primary factors for accuracy is the number of mesh elements (Bianchi and Bolognani 1998). It is necessary to increase the number of elements in the highly saturated regions to improve the model accuracy. Among all numerical modeling techniques, FE is considered the most popular technique for modeling machines that operate in magnetic saturation conditions (Bostanci et al. 2017). Moreover, since the machine airgap is changing with the rotation of the rotor, it will also require a large number of elements.

The field solution can be obtained using various numerical methods, for example, LU Decomposition, Banded Choleski, the Conjugate Gradient Method, and the Preconditioned Conjugate Gradient Method. These methods work by reducing the error between an estimate and the unknown solution at every iteration. Due to the nonlinearity of the magnetic materials, the formulated system of equations is nonlinear. The Newton-Raphson method is used to solve the system of equations (Salon 1995). For electric motors, post-processing is afterward applied to the converged magnetic vector potentials to calculate various electromagnetic quantities such as flux linkage, flux density, and electromagnetic torque.

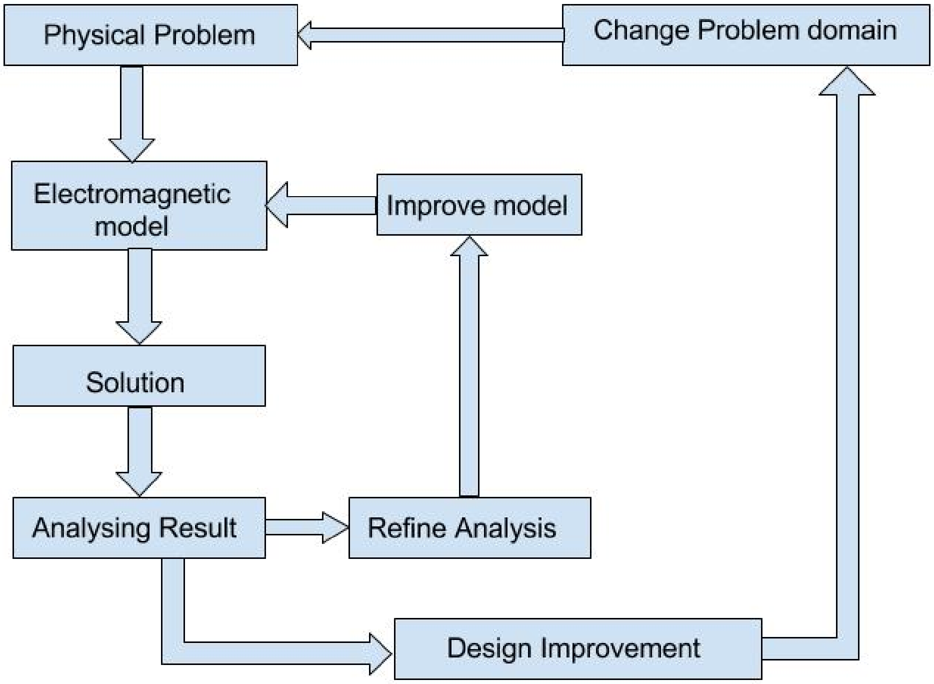


Figure 2.2: Computational Analysis in FEM

2.2.2 Magnetic Equivalent Circuits

MEC models, in general, are seen as a compromise between FEM and lumped parameter models (Sudhoff et al. 2007). They take machine geometry and material characteristics into account and are thus more refined than lumped parameter models (Amrhein and Krein 2007; Moallem and Dawson 1998). The circuit-oriented approach makes them a more intuitive design tool compared with FEM models (Tavana and Dinavahi 2016; Asghari and Dinavahi 2012).

MECs offer an alternative possibility that is based on permeance network models comprising reluctance and mmf sources. In general three types of elements appear in MEC: mmf sources, flux sources, and permeance. Node potential methods (Ostović 1987) solve the MEC. The MEC can be considered as a reduced order FE which translates a geometrical description of a magnetic device consisting of coils of wire, a magnetic conductor such as iron, and permanent magnet material into an electrical circuit description (Krause et al. 2013). By taking into account an approximately accurate machine geometry, stator, and rotor slots effects, skewing, the type of winding connections, stator and rotor leakages, and linear or nonlinear magnetic characteristics of machine cores it provides a comparatively more accurate solution than analytic methods (Carpenter 1968). An electric machine is assumed to be a quasi-stationary device; that is, any change of current that builds the flux is followed by an immediate change of flux (In other words, the time needed for an electromagnetic wave to pass through the machine is negligible compared to the period of the wave). Such a space may be partitioned into flux tubes. The flux tubes are the basis of the magnetic equivalent circuit method. A flux tube is a geometrical space in which all lines of flux are perpendicular to their bases, and no lines of flux cut their sides (Ostovic 2012).

Currents flowing in the windings are the MMF sources in the magnetic equivalent circuit of an electric machine. These MMF sources are placed in the teeth. A reluctance/permeance and an unknown flux are assigned to each yoke part or each tooth. Slots are modeled by their leakage permeances. In the air gap, a flux tube is defined when any of the stator teeth come face to face with a rotor tooth. Permeance is assigned to each air-gap flux tube. Geometries of these flux tubes vary due to the rotation of rotor teeth with respect to stator teeth and also depend on the air-gap length between any two facing stator and rotor teeth. Any air-gap asymmetry will affect the height of these flux tubes. Although some researchers have presented methods to consider motion in motors, the change of reluctance network during motion adds to the complexity of solving equations (Zhao et al. 2011; Zhang et al. 2006)

2.2.3 Comparison between MEC & FEA

The finer modeling and simulation resolution offered by FE analysis enables us to study complex issues such as internal faults or localized magnetic saturation in machines (Wang, Jatskevich, and Dommel 2007). FEM models are computationally very intensive, and the model size and time step constraints make them unsuitable for use in a real-time simulation environment (Tavana and Dinavahi 2016; Liu, Mohammed, and Liu 2009). Hence in the case of limited computation capability, FEA models are used for final design analysis only.

The advantages of the MEC method include reduced model complexity compared to FE, ease of parameterization, and fast computational time (Yilmaz and Krein 2008). MEC has gained popularity in past years mainly since the extension to 3-D does not expand the numerical complexity as quickly as with FEA (Kim et al. 2002; Amrhein and Krein 2007). MEC is also useful for performance analysis and initial device sizing. However, it may be inadequate for the accurate representation of complex geometric shapes and nonlinear material properties. A comparison of reported results between FEA and MEC simulations against experimental measurements is shown in Table 2.1. It can be observed that both FEA and MEC exhibit acceptable accuracy against experimental results.

Ref	Method	Force/Torque Comparison
(Liu and Chiang 2006)	FEA-MEC	6% on force between MEC and test 10% on force between FEA and test
(Hur et al. 1997)	FEA-MEC	8.6% on force between 2D MEC and test 9% on force between 2D FEA and test 1% on force between 3D MEC and test
(Leplat et al. 1996)	FEA-MEC	16.9% on torque between MEC and test 11.1% on torque between FEA and test
(Kim et al. 2004)	FEA-MEC	3-5% on force between FEA and test 5-35% on force between MEC and test

Table 2.1: Experimental and Simulation result comparison

FE analysis has both advantages and disadvantages relative to MEC based analysis. Regarding benefits, FEA generally requires less deployment time for simulation. The direction of flux in each element of the MEC model must be known before the MEC method is applied, whereas, in FEA, one of the results of computation is the flux distribution itself (Ostovic 2012). In addition to that, FEA is generally more accurate than MEC analysis. FEA also offers a comparatively more flexible tool to study designs incorporating new shapes as compared to any other models discussed thus far. Based on this study, it is decided that FE offers the flexibility to model complex geometries and generates the field solution with high accuracy and thus is ideal for populating the training dataset consisting of thousands of simulated designs.

2.2.4 Need for an Improved Surrogate Model

Having discussed the advantages and limitations of the popular numerical methods associated with the analysis of electric machines, it is clear that a computationally cheap and accurate field solution is necessary for the analysis of electromagnetic devices such as actuators, transformers, and electrical machines (Yilmaz and Krein 2008; Silva 2018b; Silva et al. 2017b). This can be a challenging task and usually for generating an optimal design, we need to perform a few thousand analyses as part of optimization. Usually, once a handful of designs are shortlisted using MEC or d-q analysis, testing of these designs with 3-D FEA is performed. There have been studies where FEA is used for design step directly (Bianchi and Bolognani 1998; Salon 1995). However, this dramatically reduces the size of search space and is computationally very expensive due to longer analysis time. Decades of research and engineering have gone into designing fast numerical solvers for design analysis. However, employing FE methods for modeling and simulation purposes remains a computationally expensive process and is still an active field of research. This has also led to the development of alternative surrogate models to accurately evaluate and/or optimize a device (Silva 2018b; Silva et al. 2017b; Wang et al. 2016; Ghorbanian, Salimi, and Lowther 2017; Ghorbanian and Lowther 2017). Unfortunately, these surrogate methods are usually specific to a problem and describe systems with very few parameters, thus limiting the ability of such surrogate models to deal with any arbitrary change in the design of a machine (Silva et al. 2017b). On the other hand, in recent years, rapid developments in the field of machine learning (ML) and big data (Krizhevsky, Sutskever, and Hinton 2012), have opened new venues for pattern recognition and curve fitting in complex problems such as computer vision (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016), dense regression (Badrinarayanan, Handa, and Cipolla 2015). Most of the recent work in this growing field of research is related to improving data-driven statistical methods to supplement simulations to solve engineering problems. These works are shown across applications such as smoke simulation (Chu and Thuerey 2017), liquid splash modeling (Um, Hu, and Thuerey 2018) and modeling liquid behavior with obstacles (Tompson et al. 2017). Another work, which is more related to this discussion, tried solving Poisson's equation with a Deep Learning (DL) algorithm (Tang et al. 2017). The use of deep networks suitable for the task for magnetic field estimation might accelerate the solution of such problems. Hence, the possibility of applying deep networks to magnetic field estimation is investigated in this work, for the domain of low-frequency EM devices, using a bitmap approach.

2.3 Machine Learning

Although DL has gained a lot of importance in recent times, it has been in existence since 1940. The first experiment with neural networks was performed in the 1940s (McCulloch and Pitts 1943) and the algorithms to train deep networks have been present since 1980. The first successful application of Convolutional Neural Networks(CNN/ ConvNets) was developed by Yann LeCun (LeCun et al. 1998) in the 1990s. The most popular of those architectures is known as *LeNet*, which was used to read zip codes, handwritten digits, etc. Utilization of these concepts for producing state of the art results in more evolved fields such as computer vision (He et al. 2015), text analysis (Devlin et al. 2018), forecasting, etc., demands computational memory and speed which was not available on mainframes until a few years back. At one point in time, this field was regarded as an art rather than a science due to the difficulty involved in successfully training these models, but that is not the case anymore. Some other reasons that have contributed to the success of these deep models today are the resources available both regarding training data and software libraries such as Tensorflow(Abadi et al. 2015), Theano (Theano Development Team 2016), PyTorch (Paszke et al. 2017), etc. Also, these algorithms have undergone some major changes in the past decade which have greatly simplified the training process.

Due to the advancement in computational technologies and improvement in DL algorithms(Hinton, Osindero, and Teh 2006; Bengio, LeCun, et al. 2007; Delalleau and Bengio 2011), the number of hidden units in an artificial neural network has doubled in size every 2-3 years (Goodfellow, Bengio, and Courville 2016), since the late 2000s. Some of the major factors that can be attributed to this growth are:

- Computation Power (GPUs, ASICs, TPU (Tensor Processing Unit)).
- Data (Both ability to generate massive datasets and process them).
- Improvement in algorithms (CNN, LSTM (Long short term memory)).
- Infrastructure & Frameworks (Git, Cloud Computing, Tensorflow (Abadi et al. 2015)).

As for defining 'Deep Networks', there is no universally agreed-upon threshold of depth dividing shallow learning from deep learning. One of the earliest deep neural networks had three densely connected hidden layers (Hinton, Osindero, and Teh 2006). In 2014, VGG networks consisted of 16+ hidden layers and were called 'Very Deep' networks (Simonyan and Zisserman 2014). In 2016 the "Extremely Deep" residual networks (He et al. 2016) consisted of 50 to 1,000 hidden layers. For Feedforward Networks, CAP is the number of hidden layers plus one. Formally, CAP (Credit Assignment Path) is the chain of transformations from input to output. Most researchers in the field agree that deep learning has multiple nonlinear layers (CAP ≥ 3) and Schmidhuber (Schmidhuber 2015) considers CAP ≥ 10 to be very deep learning. So, a neural network with more than one hidden layer can be considered a Deep Network.

Why and where should one use a Deep Neural Network? As the number of features increases in our data, it becomes difficult to train a linear model. The interaction between these features can be too complicated for simpler models to achieve satisfactory accuracy. On the other hand, deep neural networks can adapt to more complex datasets and generalize to previously unseen data; thus they can fit more complex data than linear models can. It has been observed that the network's performance degrades if a single layer is removed (Krizhevsky, Sutskever, and Hinton 2012). However, the tradeoff of an additional layer is that the model will take longer to train, will be larger in size, and have lesser interpretability. One of the trickiest things about these networks is to get a configuration of parameters that works. There can be multiple possibilities of these configurations.

The way the model is trained, i.e., the starting position, optimization algorithms, learning rate, etc., plays an essential role in the success and failure of our model. For the DL models trained in this chapter, the outcome/target/label/dependent variable is in the form of a quantitative quality (magnetic field distribution), which is predicted based on the information related to geometric features, excitations and the material properties, all of which will constitute the features/predictors/independent variable. Having identified the training data, the next step is to perform statistical learning that will enable us to predict the field solution of new unseen geometries. This type of learning is a case of supervised learning. The next few sections will dig deep into the complete ML pipeline. The first step (discussed in Section 2.4) in the pipeline is the collection of training data in a format suitable for training a DNN. Once sufficient data is collected different neural network architecture are explored to suit the learning task, This process is discussed in Section 2.6.1, followed by the working of the Neural network in Section 2.6.2 and the decision-making process involved in tuning the network suitable for the task at hand. Results based on this pipeline are shown in Section 2.7.

In addition to magnetic field estimation, identifying when the input does not fit the problem description is also essential and is handled very well by an FEA solution system. Basically, a ML network trained in a supervised manner could predict the solution for any arbitrary input. As such, a DL emulator lacks a physics-based judgment. This is an important issue that will be addressed through an uncertainty analysis over the prediction, to generalize the DL estimator as much as possible.

2.4 Data Collection

Deep Learning models usually have many parameters to be tuned and therefore need a lot of data to generalize. The input data is called a dataset. A ‘dataset’ is defined as a collection of instances that share some common attributes. One of the most significant attributes related to a dataset is the type of task which is also referred to as the ‘target’. The other important attribute is the domain, which is the source of the data. The domain and task characterize a supervised learning problem.

In supervised learning, the two most important types of learning problems can be characterized based on the type of task, which are classification and regression. For a Classification task, the target is a label (also known as class) where the task is to match the samples with existing entities in the form of classes. On the other hand Regression is an example of supervised learning where the task is to predict a number by learning a function relating the input and output.

The performance of any ML algorithm is only as good as the data used to train it. A weak representation of the ML problem insufficiently captures the dynamics necessary to successfully learn the mapping of inputs to outputs.

A weak representation of a problem can be attributed to two main reasons:

1. Missing important information of the domain representation (input).
2. The dataset, which is a sample of a population, does not represent the population in an unbiased manner.

To tackle the above issues, the input data is represented using the CAD geometry along with information related to the material distribution and excitations, at sufficient resolution. Further, all the samples are populated using a Latin Hypercube.

Depending on the complexity of the problem, different network architectures or parameters might be considered in the search for an optimal structure to guarantee a reasonable prediction error. Therefore, to appropriately establish the idea of DL for magnetic field estimation, three problems with different complexities of the geometry, material, and excitation, i.e. a simple coil in air, a transformer (*Description of TEAM Problem: 32 A Test-Case for Validation of Magnetic Field Analysis with Vector Hysteresis*) and an Interior Permanent Magnet (IPM) machine (Wang et al. 2016; Ghorbanian, Salimi, and Lowther 2017; Ghorbanian and Lowther 2017) are incorporated. All the three problems are parameterized and simulated in the FEA software package – MAGNET (*MAGNET v2020.2*). All the materials properties including the non-linear (NL) nature of the B-H relation were used as is mentioned in the material library of MAGNET (*MAGNET v2020.2*).

Given the goal of this work, i.e. predicting the field distribution in electromagnetic devices using DL, a set of inputs and outputs used for training the networks must be generated first. The input of the network is formatted as a uniform structured grid containing information regarding the geometry, material, and excitation at each node in the input (pixel).

2.4.1 Coil

A simple coil made of copper is located inside an airbox, whose dimensions (height = 160mm, width = 160 mm) are fixed. However, the radius of the coil ($3mm \leq \mathrm{R} \leq 15mm$) as well as its center's position change as the parameters of the problem. Constraints are placed to make sure the entire coil remains inside the box and that the radius is determined first and the center is adjusted accordingly. The coil current is also changed from 5A to 15A.

2.4.2 Transformer

A transformer with two coils, one of which is not excited, is considered as the second problem which is more complex than the coil problem in terms of the geometry, material, and field

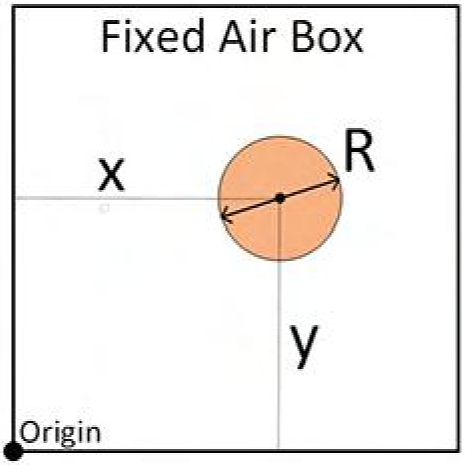


Figure 2.3: Parametrized Geometry - Coil.

distribution. The transformer core is made of M19 silicon steel material(NL). The coils are made up of copper, and the dimensions change as in Table 2.2. The left coil's current is fixed at 1A and there are 90 turns and the right coil's current is zero. The transformer depth is 2.5mm.

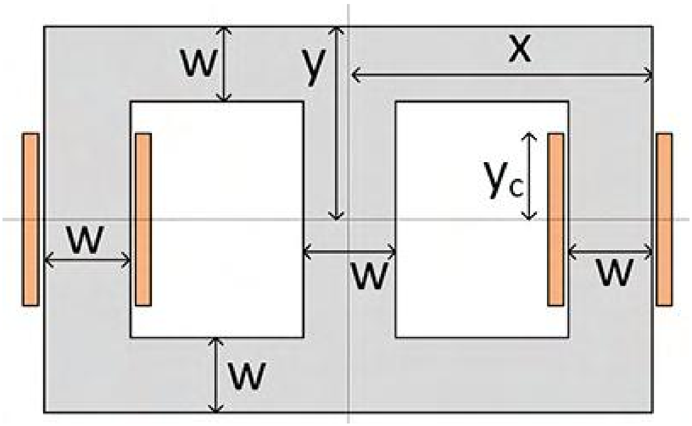


Figure 2.4: Parametrized Geometry-Transformer.

Parameter	Minimum (mm)	Maximum (mm)
x	10	190
y	10	190
w	5	150
y_c	5	50

Table 2.2: Transformer Parameters

2.4.3 IPM Motor

A 4-Pole, 24-slot IPM motor is analyzed as the most complex problem in this work. In addition to M19, and copper, permanent magnets (NdFeBr) are also incorporated. Table 2.3 lists the parameters of the IPM motor. The stator windings have 8 turns and the excitation current was varied from 25 A to 35 A rms. A partial (one pole) model is used for the simulations to reduce the computation time.

In addition to magnetic field estimation, identifying when the input does not fit the problem description is also essential and is handled very well by an FEA solution system. Basically, a ML network trained in a supervised manner could predict the solution for any arbitrary input. As such, a DL emulator lacks a physics-based judgment. This is an important issue that will be addressed employing the uncertainty analysis, to generalize the DL estimator as much as possible.

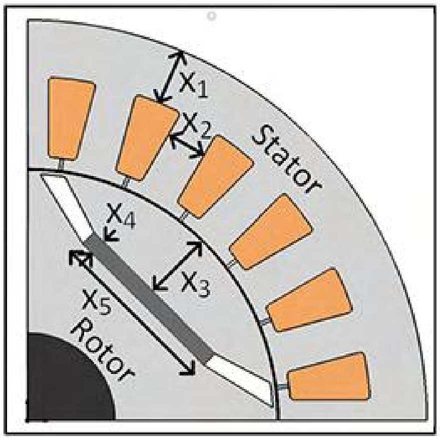


Figure 2.5: Parametrized Geometry - IPM Motor.

Parameter	Minimum (mm)	Maximum (mm)
Stator back iron (x_1)	5	35
Stator tooth width (x_2)	5	12
Magnet inset depth (x_3)	5	25
Magnet thickness (x_4)	1	50
Magnet width (x_5)	20	50

Table 2.3: IPM Motor Parameters

2.5 Neural Networks

Neural Networks consist of collections of neurons that are connected in an acyclic graph. In other words, the outputs of the neurons in a fully connected layer can become inputs to other neurons through pairwise connections. Cycles are not allowed since that could result in an infinite loop during the forward pass of a network. Also, any two neurons in a single hidden layer do not share any connections. The architecture of an Artificial Neural Network (ANN) with two hidden layers is shown in Figure 2.6.

Every node computes a weighted sum of incoming inputs connecting the nodes of the previous layer. This can be visualized as merging all the (weighted) inputs entering a node to generate a single number. Every activation function (or non-linearity) takes a unique value and performs a specific fixed mathematical operation on it. Several activation functions are described in the literature (Goodfellow, Bengio, and Courville 2016).

Unlike all layers in a Neural Network, the output layer neurons most commonly do not have an activation function. This is because the last output layer is usually taken to represent the class scores (e.g., in classification), which are arbitrary real-valued numbers, or some real-valued target (e.g., in regression).

A loss function measures the quality of a particular set of parameters based on how well the induced scores agreed with the ground truth target in the training data. The loss

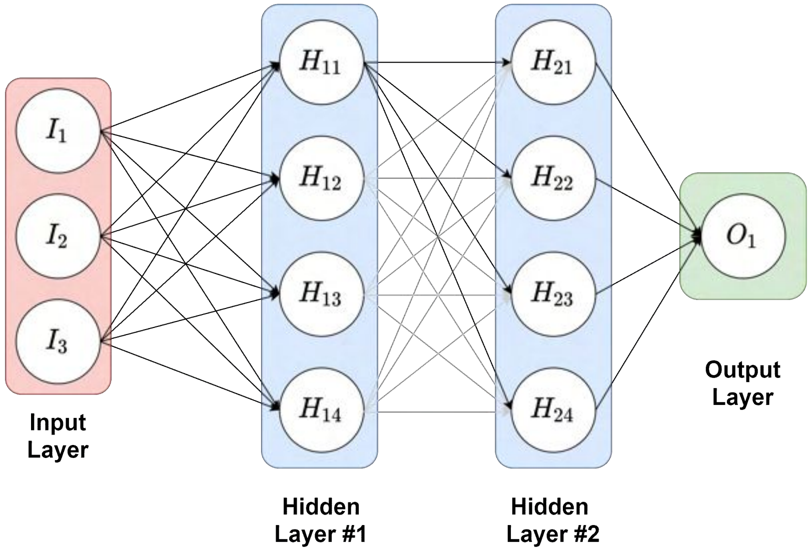


Figure 2.6: Artificial/Feedforward Neural Network with 2 hidden layers.

function is dependent on training data and weights/ biases of the network. Since in Machine Learning, we take the training data as given and fixed, and weights as variables that we can control, we use back backpropagation to compute the gradients for the network parameters and perform a parameter update. Backpropagation is a way of computing gradients of expressions through the recursive application of the chain rule. The terminology of an N-layer neural network is such that we do not count the input layer. Therefore, a single-layer neural network describes a model with no hidden layers (input directly mapped to output). This same terminology will be followed for the CNN and RNN as well.

2.6 Convolutional Neural Networks

Convolution Neural Networks (ConvNets or CNN) are similar to ordinary Neural Networks in the sense that they are made up of learnable weights and biases. Each neuron will receive some inputs, multiplies them with weights (trainable parameters), and operates on it with a non-linear function. But the difference is that CNNs make an explicit assumption that input data follows a structured pattern to store the information (usually in the form of a uniform grid such as images, time-series events, etc.). This introduces certain properties in the architecture which helps in reducing the number of parameters in the network and makes the whole training process easier (Goodfellow, Bengio, and Courville 2016).

A brief discussion on simple Feedforward Neural Networks is covered in Section 2.5. This will help in comparing and relating simple Neural Networks with CNNs.

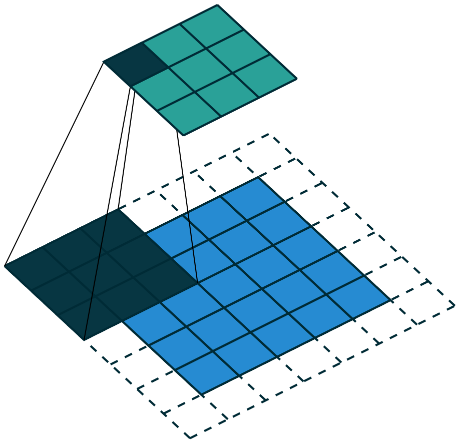


Figure 2.7: Convolution kernel visualization (Dumoulin and Visin 2016) (Blue = Input map, Cyan = Output map).

2.6.1 Architecture of the CNN based model

There are three main types of layers present in CNN architecture:

- **Convolutional layer(CONV)**: is used to compute the connection of each neuron with the local region of the input volume. It consists of a set of learnable filters/ kernels, as shown in Figure 2.7. Each filter is small spatially as compared to the input image but slides across the whole length and width of the image and computes dot products between the entries of the filter and the input at any position creating a 2-D activation map/feature map giving the response of that filter at any position. The algorithm can learn appropriate filters that get activated for certain visual features, each filter looking for something new to learn. When the 2-D activation map/ feature map is stacked together, they result in the output volume of a CONV layer.

- **Pooling layer (POOL)**: is in charge of downsampling the spatial dimension of the input. They are periodically inserted in-between CONV layers. The aim is to reduce the number of parameters in the network, thus helping in avoiding overfitting. The most popular form is a pooling layer with filters of size 2×2 applied with a stride of 2 (skipping a pixel), thus down-sampling every depth slice in the input by two along both width and height. It has two hyperparameters size and stride to control the volume of the output.

- **Fully connected layer (FC)**: Neurons in these layers have a dense connection with all activations in the previous layers, similar to the architecture of regular Neural Networks. FC and CONV layers differ in the structure, as a CONV layer exhibits local

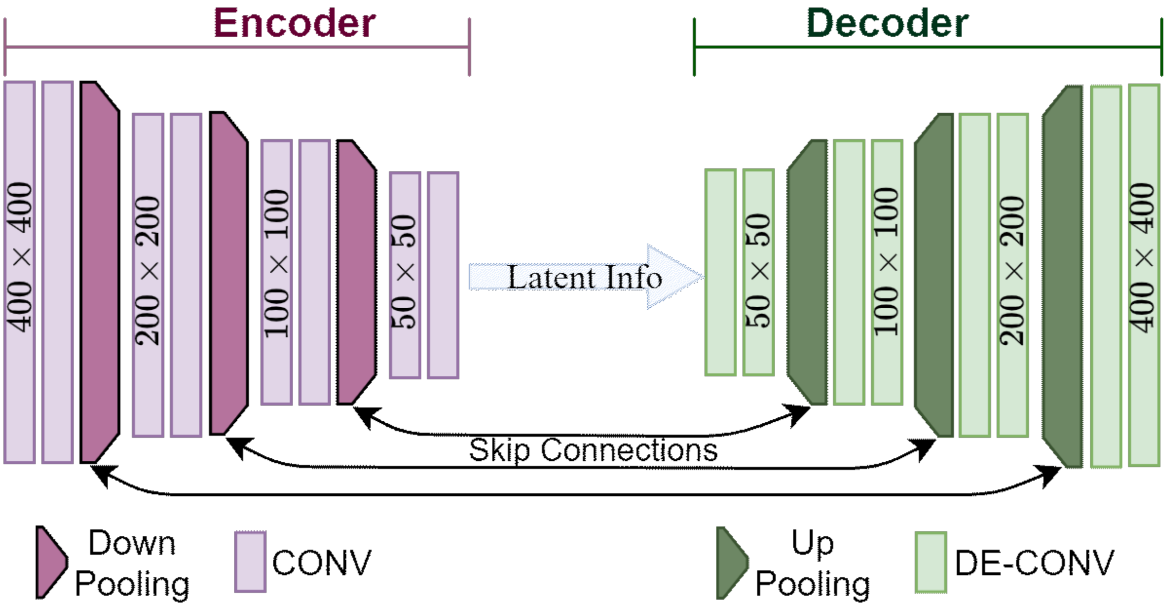


Figure 2.8: Encode-Decoder based Convolutional Neural Network Architecture

connectivity and parameter sharing, but are identical in their functionality since both layers still compute the dot product between their weights and activations, followed by a non-linear activation function. The FC layer aims to balance the locality-context tradeoff. This layer can be thought of as relating all the features learned by the CONV layer to develop patterns that capture global trends.

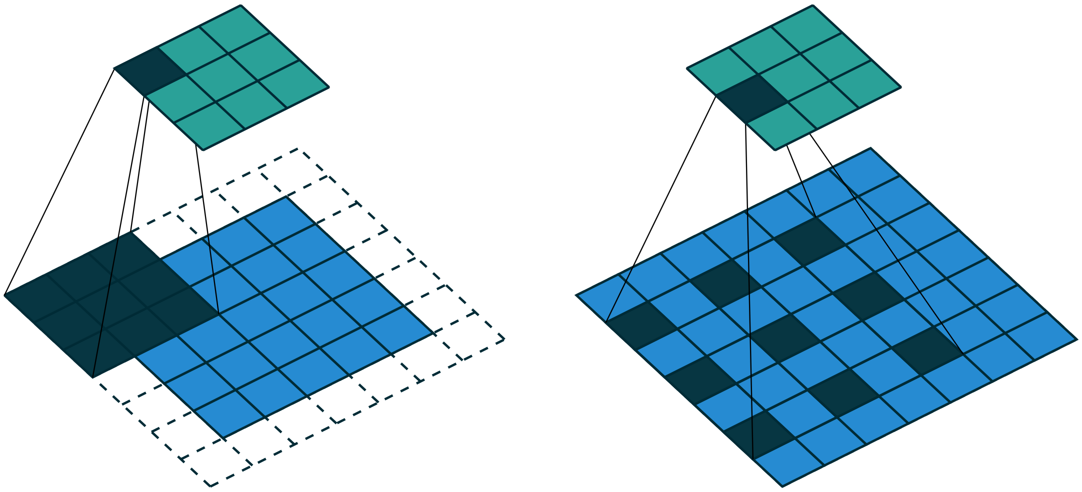
The input to the DL network consists of uniformly placed spatial data where the neighboring pixels are related to each other. Grouping together the pixels that have similar attributes, through a convolutional filter is performed using a dense kernel as shown in Figure 2.7. The traversal of the filter over the geometry results in a local measure of how likely two pixels are related to each other. In addition, a dilated filter is able to capture spatially distant relationships as shown in Figure 2.9b.

The DL model shown in Figure 2.8 and 2.11 consists of a total of 32 layers with 16 trainable convolutional layers, 8 pooling/up-sampling layers and 8 dropout layers. Each block in the architecture consists of two sets of 3×3 convolution layers (including ReLU (Rectified Linear Unit (Goodfellow, Bengio, and Courville 2016)) and a batch-normalization layer (Géron 2019)), followed by a 2 × 2 max-pooling (Goodfellow, Bengio, and Courville 2016) (green arrow)/ up-sampling (blue arrow) layer with a stride of 2. This pattern is not followed for the last two convolutional layers due to a network design constraint, i.e. to fit the field distribution dimensionality. Each convolutional layer has a stride of 1. The network is trained using backpropagation with an aim to minimize the RMS error between the prediction and FEA field distribution. The limits on geometrical and excitation parameters mentioned

Network Architecture		Dataset	
Number of Layers	32	Training Dataset	30000
Trainable Layers	16	Validation Dataset	10000
Number of parameters	2.4 Million	Test Dataset	5000

Table 2.4: Network and Training partition features.

in Section 2.4 are used to populate a design space using a Latin Hypercube. The dataset was then partitioned randomly into training, validation, and test sets. More information on the network and training dataset size can be seen in Table 2.4. Although there are other techniques for partitioning the data, namely, cross-validation and bootstrap-based validation, the high computation cost associated with these techniques prohibited their usage in this case.



(a) Non-dilated Convolutional filter with padding.

(b) Dilated filter with dilation rate of 1.

Figure 2.9: Different types of kernel for a Convolutional layer in a CNN (Dumoulin and Visin 2016) (Blue = Input map, Cyan = Output map)

2.6.2 Working of the DL model

The architecture of such a network resembles that for image segmentation purposes with the focus on dense regression instead of classification (Badrinarayanan, Kendall, and Cipolla 2017). The network architecture can be divided into two parts: encoder and decoder, as seen in Figure 2.8. The encoder section tries to extract spatially related features from the input. This involves creating a bottleneck layer to compress out the most useful information from a high dimensional input, known as ‘latent representation’. This information is then used by the decoder section to semantically project the discriminative features learned by the encoder onto the original input space to predict the field solution. For the network to train successfully, it must have sufficient capacity to satisfactorily capture the complexity of the problem (Ghorbanian, Salimi, and Lowther 2017). This capability is achieved by choosing an appropriate value for the number of layers, kernel size, number of kernels, and proper parameters for regularization. Model selection is usually performed by investigating a set of candidate models and choosing the one which gives the best balance between model fit and complexity. Random search and grid search are two of the common ways to arrive at optimal values of these hyperparameters (Géron 2019). A total of 35 networks with different configurations were trained in the process of model selection. It is observed that the number of layers and the number of kernels/filters in a layer play an important role in the model performance. However, the most significant improvement is achieved by adding dilated filters, shown in Figure 2.9b. This is a significant feature since the performance of the network improved not due to the addition of neurons but by changing the behavior of the constituent's filters. A comparison is shown in Figure 2.10, where a network with a kernel size of 5, number of kernels/filters in a convolutional layers (K) = 64, skip connections (shown in Figure 2.8 and 2.11 (b)) and alternate dilated layers in the encoder outperforms other networks. The normalized RMS error in prediction over the validation shows an improvement for a network with dilation over the same network architecture with non-dilated (dense) kernels. The pattern is consistent for all three problems, with a similar accuracy level achieved for networks with K=32 and K=64 for the coil problem, as is shown in Figure 2.10 (a), (b) and (c).

To gain intuition, we can think that the magnetic field at a point can be affected by regions far away from it (the source of excitation). The presence of dilation enables spatially far relations to be captured without increasing the kernel size. This is important for maintaining the locality-context trade-off of the network.

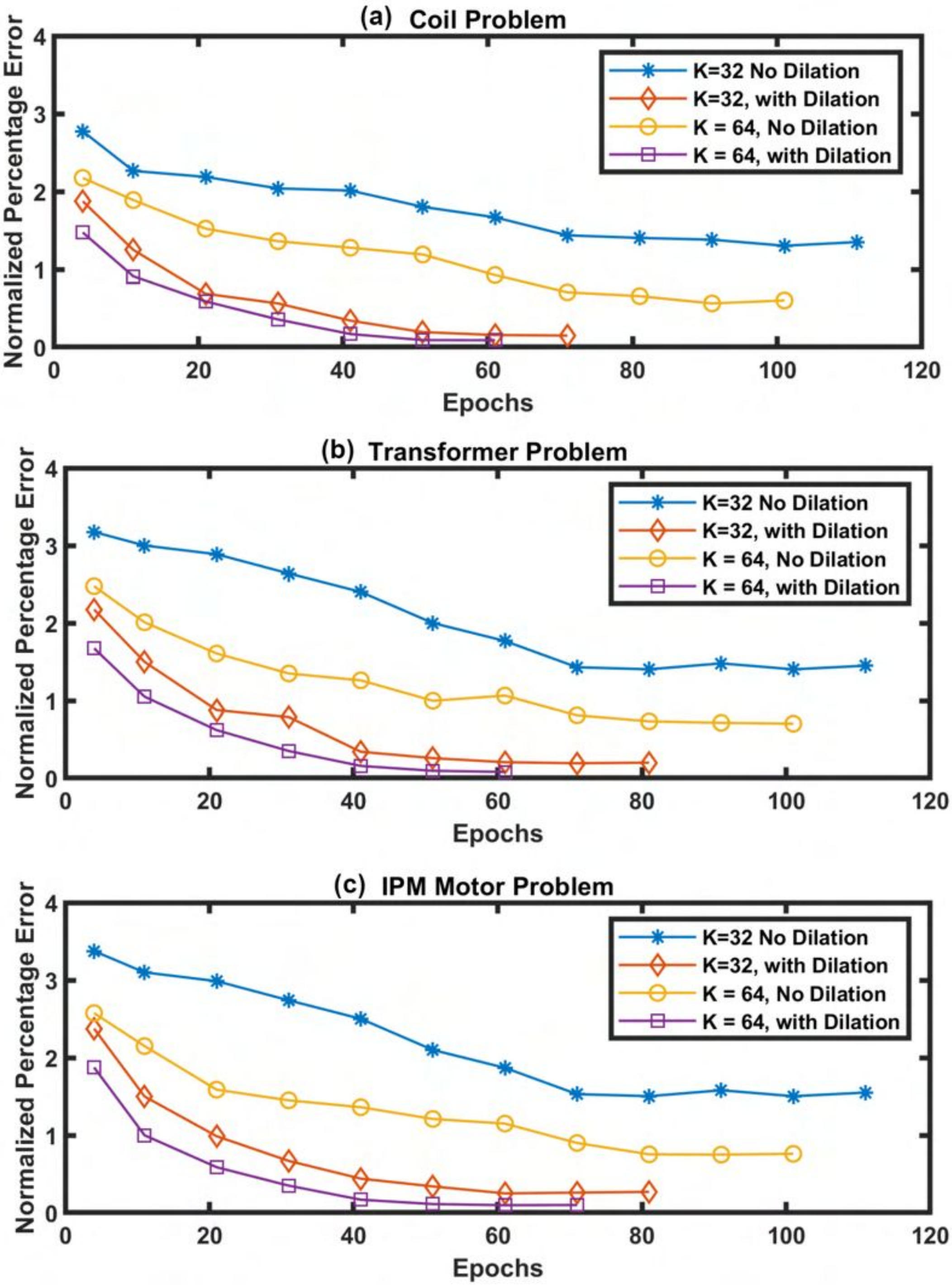


Figure 2.10: Training curve with Normalized Prediction Error (calculated over validation dataset).

2.7 Results & Performance

Figure 2.11 (c) shows the predicted magnetic field distribution in a coil, transformer and an IPM motor. The normalized mean squared error between the ground truth and the

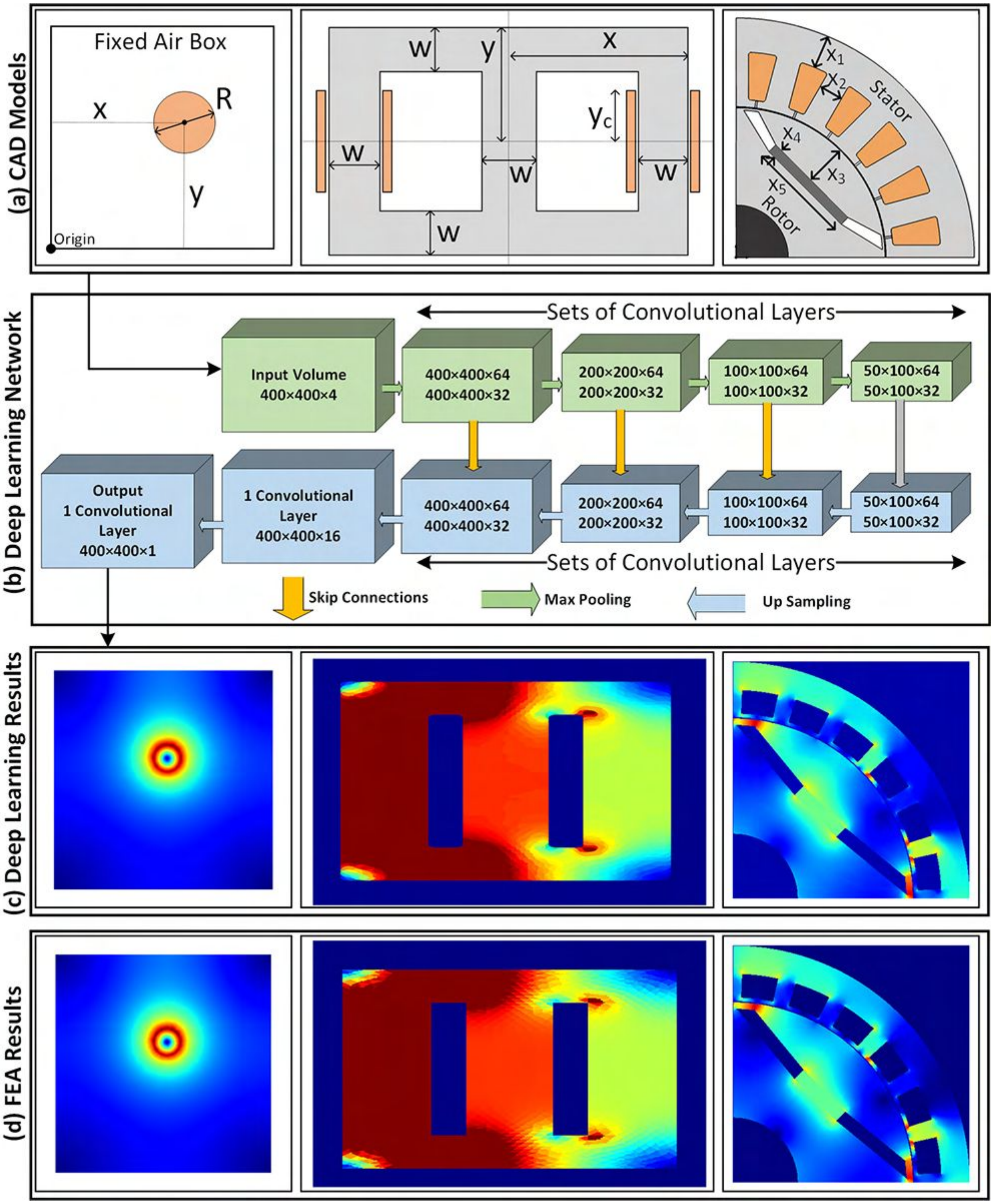


Figure 2.11: (a) Problem definition, (b) Deep network architecture, (c) field predictions, (d) FEA results.

prediction lies in the range of 0.1% - 1% per node/pixel. The error distribution for an IPM motor can be observed in Figure 2.12 and is highest around the corners, specifically in the carrier region of the rotor (boxed in black) and in the section of back iron (circled red). This was expected since a structured grid dense enough to capture these minute details will be computationally too expensive for this exploratory work. Although material exhibiting non-linear relations was included in the study, the performance of the network will deteriorate if the excitation is far different from that used for generating the dataset.

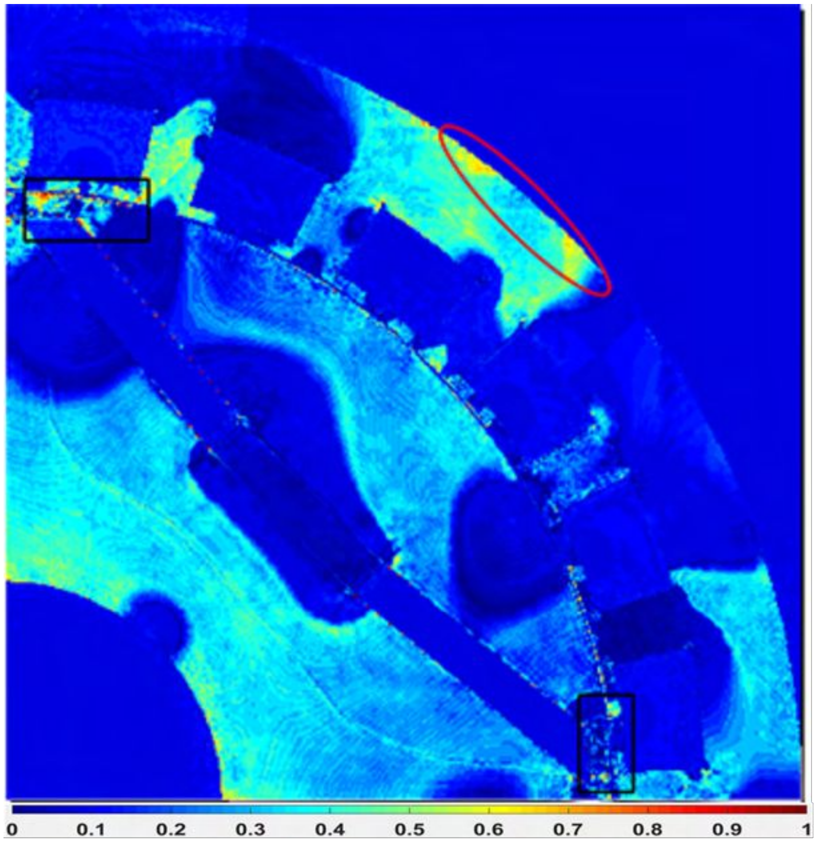


Figure 2.12: Error (Ground Truth - Prediction) distribution.

The network was trained on an NVIDIA 1080 Ti and took around 10 minutes per epoch with a batch size of 16. Convergence is achieved in around 60 - 100 epochs (depending on the network configuration, as shown in Figure 2.10), resulting in a total run time of 12 – 15 hours. The prediction time is in the range of 2-3 seconds for 100 geometries at a time (batch size). The time required to analyze multiple geometries is memory-bound rather than compute-bound, i.e. it is possible to parallelize the training and inference to thousands of samples if the memory requirements are satisfied.

2.8 Uncertainty Prediction

The DL trained in this work and some other works (Tang et al. 2017) employ supervised training and the prediction accuracy depends on how close the input is to the training data. A geometry that is significantly different from the ones in the training dataset will produce substantial errors. To be able to rely on such a predictor (trained in a supervised manner), we need to provide a measure of uncertainty in the results. This knowledge is essential to quantify the confidence one can have in the results. One way to achieve this is by extending the DL model by introducing a probabilistic component in the neuron weights. The technique is based on quantifying uncertainty known as Monte- Carlo (MC) dropout (Gal and Ghahramani 2016). Monte Carlo integration gained popularity during the Manhattan Project. Due to its effectiveness, it was included in the top 10 algorithms of 20^{th} century (*The Beginning of Monte Carlo Method*). Instead of generating a system of the equations to describe a complex problem, we can simulate it multiple times and average out the quantity we are interested in.

2.8.1 Uncertainty Maps

Uncertainty maps are images that indicate the model's uncertainty for a given pixel classification. It may be tempting to interpret the values from the final softmax layer of a CNN as confidence scores, but we must be careful as work on Adversarial Networks have shown that minute perturbations to a real image can change the softmax output to arbitrary values (Goodfellow, Shlens, and Szegedy 2014). This can make the network predict wrong results with high confidence. A network will try to predict even in cases where the input data itself might be irrelevant. This is especially important when dealing with images that were not present during training. (Gal and Ghahramani 2016) showed how we can use dropout in a deep CNN to get uncertainty information from a model's predictions. Instead of having fixed weights, each weight is drawn from a distribution.

Quantifying the uncertainty in the network's prediction would allow us to find regions in an image for which the network is unsure of its prediction. Furthermore, segmentation based on an uncertainty threshold value can sort out the pixels over which the model is confident in the prediction, thus providing us with a partial but highly accurate solution.

2.8.2 Theory

Dropout (Srivastava et al. 2014) was introduced as a regularization technique with an aim to check the bias-variance tradeoff of the network. The technique involves a parameter

(p), which determines the probability of dropping out any neuron from the network for an iteration during the training on the dataset (X, Y). Thus, training a neural network with dropout is equivalent to training a collection of 2^K thinned networks with extensive weight (W) sharing, but a single network approximates the average output at test time, through a weight averaging technique (Badrinarayanan, Handa, and Cipolla 2015). Thus, assuming a point estimate for weights. On the other hand, dropout variational inference is a practical approach for approximate inference in large and complex models (Kendall, Badrinarayanan, and Cipolla 2015).

Even with a small number of parameters, inferring the model posterior distribution $(p(W|X, Y))$ in a Bayesian NN is a difficult task, with the variational inference being a popular approach to approximate the posterior distribution. This is done by minimizing the Kullback-Leibler divergence from the true posterior $(p(W|X, Y))$. However, this approach can be very computationally expensive, almost doubling the number of model parameters (Blundell et al. 2015). On the other hand, it has been shown that dropout can be cast as approximate Bayesian inference in Gaussian processes. This means that uncertainty information can be extracted from models without requiring additional parameters. Usually, training a neural network with dropout can be seen as training a collection of 2^K thinned networks with extensive weight sharing, but a single network approximates the average output at test time. Thus assuming a point estimate for weights.

Instead, performing dropout during the test phase can be viewed as performing Monte Carlo sampling from the posterior distribution over the various dropout models. By applying dropout to all the weight layers, we are essentially drawing weight from a Bernoulli distribution. This technique has been utilized in order to produce uncertainty maps as a measure of prediction uncertainty. This work is in line with Bayesian SegNet (Kendall, Badrinarayanan, and Cipolla 2015). However, this work extends this information to more than a visualization tool and relates uncertainty information and segmentation accuracy.

We are interested in finding the posterior distribution over the convolutional weights, W given the observed training data: input (X) and target (Y).

$$p(W|X, Y)$$

Unfortunately, this posterior distribution is not tractable.

$$p(W|X, Y) = \frac{p(Y|X, W) \times p(W)}{P(X)}$$

The evidence term : $p(X) = \int p(X, W) \times p(W)dW$, is very difficult to compute. We need to approximate the distribution of the weights (W). We can use Variational Inference (VI) to approximate it. Here we learn a distribution q(W), by minimizing the Kullback-Leibler divergence between this approximate distribution & the true posterior:

$$KL\,(q(W)\ ||\ p\,(W|X,Y)).$$

Since dropout is seen as an approximation to a Bayesian Network, the uncertainty in the weights induces a prediction uncertainty.

$$p(y|x,X,Y) = \int p(y|x,W) \times p(W|X,Y)dW$$

Since the true posterior is intractable, we approximate the posterior using Monte Carlo integration:

$$p(y|x,X,Y) = \int p(y|x,W) \times q(W)dW$$

How to implement this? With Dropout, a unit is present with a probability '*p*', and at testing the unit is always present and the weights are multiplied by '*p*'. In practice, we can sample from the distribution by running several forward passes through the network. Dropout can be cast as approximate Bayesian inference over the network's weights. This imposes a Bernoulli distribution over the CNN filter's weights.

2.9 Bayesian Network - Results

Using Monte Carlo (MC) dropout (Gal and Ghahramani 2016), uncertainty maps are computed for all three CNNs by retrieving 10 Monte Carlo samples from the networks and then calculating the standard deviation over the softmax outputs of the samples. In this Chapter, if not stated otherwise, the uncertainty maps are displaying the mean standard deviation over all the classes. Besides the uncertainty maps, Monte Carlo sampling has also been shown to be valuable to increase classification accuracy (Kendall, Badrinarayanan, and Cipolla 2015), as it has been shown to outperform the standard weight averaging technique. A qualitative measure can also be extracted by generating uncertainty maps. A normalized uncertainty map through such implementation is shown in Figure 2.13(b), with regions of high uncertainty highlighted and expanded in Figure 2.13(c). Figure 2.13(a) shows the error distribution between the prediction and ground truth. It can be seen that the standard deviation of the Monte Carlo samples (Figure 2.13(b)), is related to classification accuracy. A

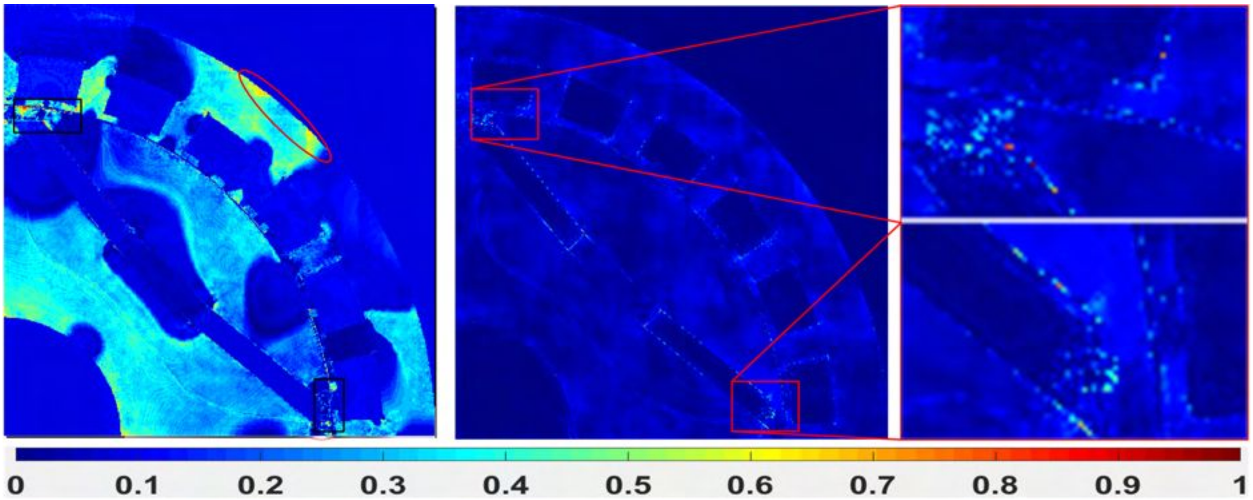


Figure 2.13: (a) Error distribution, (b) Normalized uncertainty map from Bayesian Inference, (c) Region of uncertainty map with high uncertainty.

high value of uncertainty is an indication to not rely on the results and perhaps employ the generalized conventional solvers (FEA, FDM, etc.). Out of the three regions of high error (circled in Figure 2.13 (a)), the Bayesian approach could relate two (black box in Figure 2.13(a)) with high uncertainty. The uncertainty maps are computed by retrieving 10 MC samples from the network and calculating the standard deviation over the samples. Besides the uncertainty maps, MC sampling improved the accuracy (normalized mean squared error) by around 10%.

2.10 Conclusion

By learning the field distribution from the FEA software, our model efficiently approximated the distribution. This enables us to generate the field distribution for new geometries at lower computational costs, with the added advantage of parallelizability over GPUs. A probabilistic way of analyzing the prediction without the ground truth is also presented. Thus, providing the decision-making capacity to either use the estimated field for post-processing or revert to other numerical solvers. Although a structured CNN is not the ideal architecture to handle an unstructured mesh as is employed in FEA, the DL field estimator model can function as an initial guess for the FEA solver. With recent advancements in Graph Convolutional networks, which are more suited to handle unstructured meshes, we hope to extend this work to come up with an improved predictor in the future. Another bottleneck is the amount of simulated (big) data required for training a deep network. An approach involving semi-supervised or unsupervised learning can offer an exciting path with less dependency on labeled data. Nonetheless, this work can provide guidelines for selecting a suitable network design for future work in this field.

Crucially, the proposed approach shows good accuracy in magnetic field prediction and can take advantage of GPUs for fast training. Through this chapter, essential guidelines for setting up deep networks are established, the result/inference is improved by Bayesian techniques and the confidence levels of predictions are evaluated appropriately.

The work presented in this chapter serves as foundation for exploring the usage of DL models for other computationally expensive tasks such as performance maps, namely efficiency and power factor maps. These will be discussed next in Chapter 3.

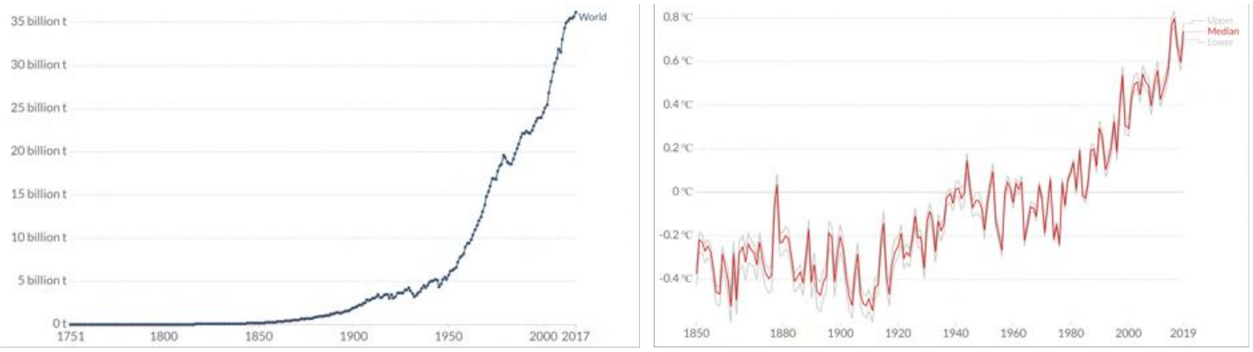
Chapter 3

Performance Map Prediction

3.1 Introduction

The Paris Climate Change Conference (COP 21) (“UNFCCC, Adoption of the Paris agreement. COP” 2015; “Paris agreement” 2015) was tasked to address the greatest challenge ever faced by the world - Global Warming (Figure 3.1). The principal target of this conference was to adopt a new climate agreement to limit global warming to well below 2°C, as compared to pre-industrial levels. All the participating nations submitted national plans to achieve this target in the next few decades and identified options that may help to reduce anthropogenic greenhouse gas (GHG) emissions. One of the largest contributors to GHG emissions is the transportation sector. This has been shown in a study performed in the USA (U.S. Greenhouse Gas Emissions and Sinks 1990–2018) (*Inventory of US Greenhouse Gas Emissions and Sinks: 1990–2016* 2018), under the United Nations Framework Convention on Climate Change (UNFCCC), where it was found that the transportation sector accounted for the largest portion (28%) of total U.S. GHG emissions in 2018, followed by electricity generation as shown in Figure 3.2. The worldwide contribution of the transportation sector towards global CO_2 emissions is at 14%, with 95% of the worlds' transportation energy coming from burning fossil fuels, especially gasoline and diesel.

Cars, trucks, commercial aircraft, and railroads, among other sources, all contribute to transportation sector emissions. In the same study, (*Inventory of US Greenhouse Gas Emissions and Sinks: 1990–2016* 2018), it was also observed that the trend from 1990 to 2018 (for the USA) shows that GHG emissions in the transportation sector increased more in absolute terms than any other sector (i.e. electricity generation, industry, agriculture, residential, commercial), largely due to increased demand for travel and personal ownership of vehicles. Within the transportation sector, light-duty vehicles (including passenger cars and light-duty trucks) were by far the largest category, with 59% of GHG emissions, while medium-



(a) (b)

Figure 3.1: Annual (a) CO_2 emission (b) Global land-sea temperature increase relative to 1961-1990 average temperature. Data from (Friedlingstein et al. 2019).

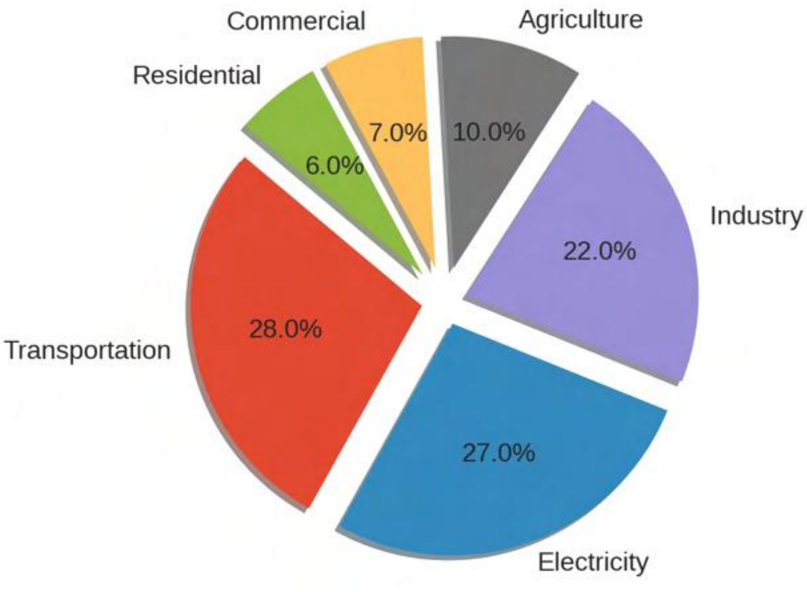


Figure 3.2: US GHG Emissions by Sector

and heavy-duty trucks made up the second-largest category, with 23% of emissions, a combined share of 82% as shown in Figure 3.3. The greenhouse gas pollution that causes global warming comes from numerous sources (CO_2, CH_4, N_2O and various $HFCs$). Although automobiles release most of the GHG, the main contribution comes from the carbon dioxide (CO_2) emitted as the engine burns fuel.

To reduce the GHG emissions, governments and car manufacturers around the world are setting ambitious targets of ending the sale of combustion engine-based cars in the not too distant future. Thus, targeting the 82% share of carbon emissions from the transportation sector, agencies such as the European Environment Agency (EEA) are also designing strict guidelines and placing heavy emission premiums to limit the CO_2 emissions (European En-

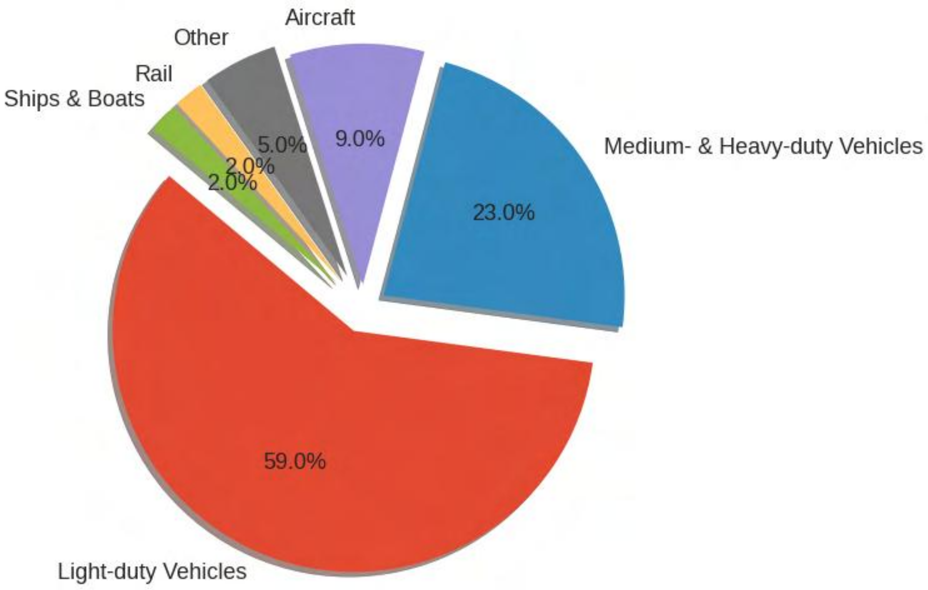


Figure 3.3: US Transportation Sector GHG Emissions by Source

vironment Agency 2016; European Environment Agency 2015). With a similar focus on reducing carbon emissions, the US Department of Energy (Yang et al. 2015) has also encouraged researchers and automotive manufacturers to design highly efficient drivetrains for variable-speed drives such as those for hybrid and electric vehicles (HEVs). HEVs serve as an alternative to fossil fuel-based vehicles and have become popular in recent years. They offer a chance for a more environmentally-friendly mode of transportation and an increasingly viable solution to the global pollution crisis. The popularity of HEVs is due to other factors too. A study from the University of Michigan's Transportation Research Institute (Sivak and Schoettle 2018) found that electric vehicles cost less than half as much to operate as gas-powered cars.

This serves as a massive motivation to develop techniques that can help improve the efficiency of electric vehicles so that they become even more popular and efficient. One of the most effective graphical ways to describe the performance of an electric machine is through a performance map. This chapter is dedicated to finding novel statistical solutions to generate performance maps such as efficiency and power factor maps. Although only two types of electric motors are considered, the techniques discussed in this chapter can easily be adapted for all kinds of electric machines.

3.2 Literature survey

The main component of an electric drivetrain is the electric motor. Electric motors used in the majority of industrial applications operate at one or a few predefined operating points. As such motors designed for such industrial usage are optimized for specific operating points and the performance is well defined. Additionally, many machines were designed to operate at either dc or a fixed frequency and voltage, e.g. 60Hz, 110volts. The development of power electronics; the electrification of many systems; and the need to operate efficiently over an entire performance envelope rather than just at one point; have significantly changed the design targets of an electric motor. At the same time, motors used in the field of Electric Vehicles and Hybrid Electric Vehicles are expected to operate efficiently over the entire torque-speed range. Further, the machine is designed to account for frequent start/stop and high acceleration and deceleration. To analyze a motor's performance, it has to be studied at all possible torque-speed combinations within the motor's operating range.

For a specific drive configuration (battery, inverter, modulation scheme), an electric motor such as the interior permanent magnet (IPM) machine can operate flexibly at different speed and torque points with high-efficiency levels when compared to an internal combustion engine. Typically, these points are represented by an efficiency map as shown in Figure 3.4. Projecting onto the 2-D plane of speed and torque, the efficiency map consists of varying efficiency levels in different regions. The peak torque is constrained by the motor's thermal limit, while the maximum speed depends on the rotor's mechanical structure. The benefit of an efficiency map is for vehicle dynamic simulations, as in (Mahmoudi et al. 2015), to predict an HEV's range (or total energy usage) in an urban or highway setting.

Finding such a map, however, for a given motor design is computationally expensive. The generation of a performance map such as efficiency, power factor, etc., can require significant computation, extending from hours to even days to run finite element (FE) simulations for all operating points and this means that it is difficult to include this information in the early stages of the design process. To tackle this computational burden, related works on designing electric traction motors consider multiple operating points (e.g. rated, peak, or max speed) using surrogate-based optimization to ensure that the designed motor is optimized with respect to different operating conditions (Silva et al. 2017a). While this methodology can provide optimal solutions, it is difficult to determine how changing a motor parameter can affect the location of the high-efficiency regions in an efficiency map. However, the performance map itself could be described as a response surface representing the impact of changes in excitation, geometry, and materials on the particular performance parameter over the entire operational envelope. As such, it could be a candidate for a machine learning

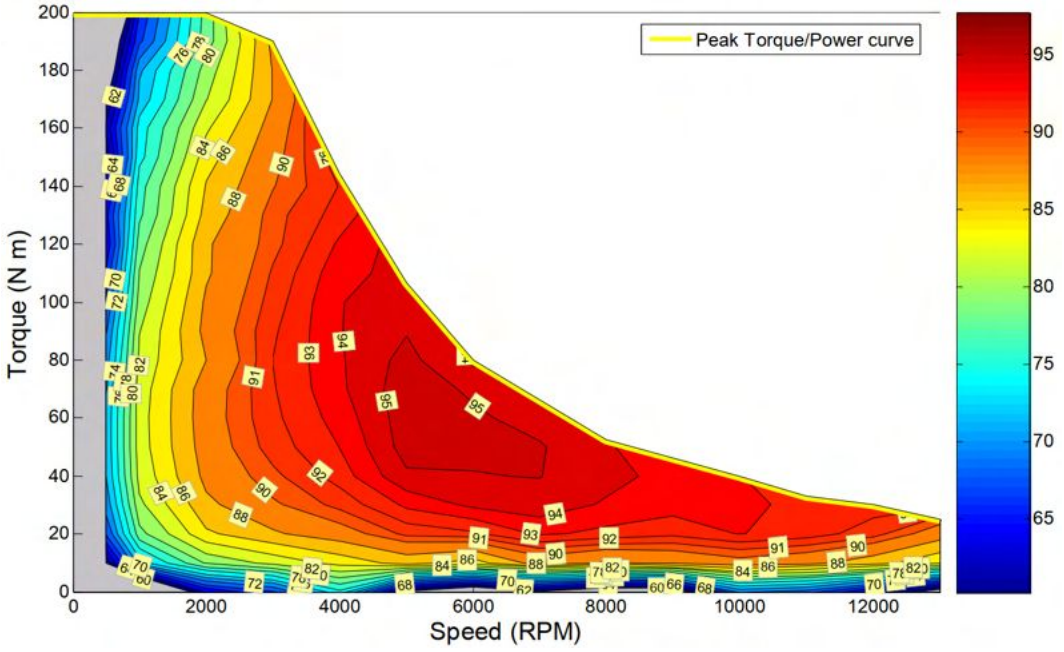


Figure 3.4: Efficiency map of the 2010 Toyota Prius IPM motor (Mahmoudi et al. 2015).

approach provided that issues such as generalization of the knowledge and the ability to transfer learning between scenarios can be resolved. Having the capability to predict this map reasonably quickly, e.g. in seconds, for use in motor design and optimization as well as in sensitivity analysis will be a significant improvement over the present method.

Recently, deep neural networks (DNNs) have gained popularity in the field of computational electromagnetics, since they can provide relatively accurate solutions in a reasonably short time compared to physics-based simulations. Examples include topology optimization of electric motors (Asanuma, Doi, and Igarashi 2020; Barmada et al. 2020) and performance prediction of motor drives (refer Chapter 2). Such approaches employ data-driven, deep neural networks to approximate the output performance that was derived from the solution of FE solvers.

This chapter explores different architectures and techniques to develop a suitable DL surrogate model for predicting a performance map for different motors. Based on the investigations, this work suggests guidelines for designing a Deep Learning (DL) surrogate using supervised learning. Once the optimal architecture has been identified, the focus will shift to reduce the data and computational requirements associated with supervised learning. This chapter aims to extend DL-based surrogate modeling and the content and structure of the chapter are summarized as follows: Section 3.3 starts with the setup of the supervised learning task by describing the data collection process. Next in Section 3.4, the problem description for supervised learning is identified and relevant requirements of the model architecture are discussed. Throughout Section 3.5, the results obtained with different supervised task approaches are compared and discussed, before a Bayesian learning approach is used to address the issue of uncertainty in the trained networks. This will conclude the cold start supervised learning task of a performance map estimation.

From Section 3.7 onwards, the trained networks are extended to train and predict performance maps for a new geometry. This will involve the discussion of a transfer learning strategy and performance analysis in Section 3.8.

3.3 Data Collection

The procedure for efficiency map calculation is given below and shown in Figure 3.5, which is based on Mohammadi and Lowther 2017. For a given motor geometry, material and simulation parameters (drive, excitation, time steps), electromagnetic FE simulations using *Simcenter MotorSolve* are run at different dq currents to obtain the dq flux linkages for characterization in Stage 1. Then, these 9 dq flux linkages points are used to construct flux linkage maps (λ_d, λ_q) as functions of the dq currents using 2nd-degree polynomials. Next in Stage 2, different control strategies, namely the Maximum-Torque-Per-Ampere, Flux Weakening, and Maximum-Torque-Per-Volt, are used to find optimal excitation conditions (I_s^*, γ^*) for every motor speed N^*. In Stage 3, these points are used in FE simulations to compute the values of the electromagnetic torque T_{em}^* and efficiency η^*. The final (N^*, T_{em}^*, η^*) points are then post-processed and interpolated to generate an efficiency map in Stage 4.

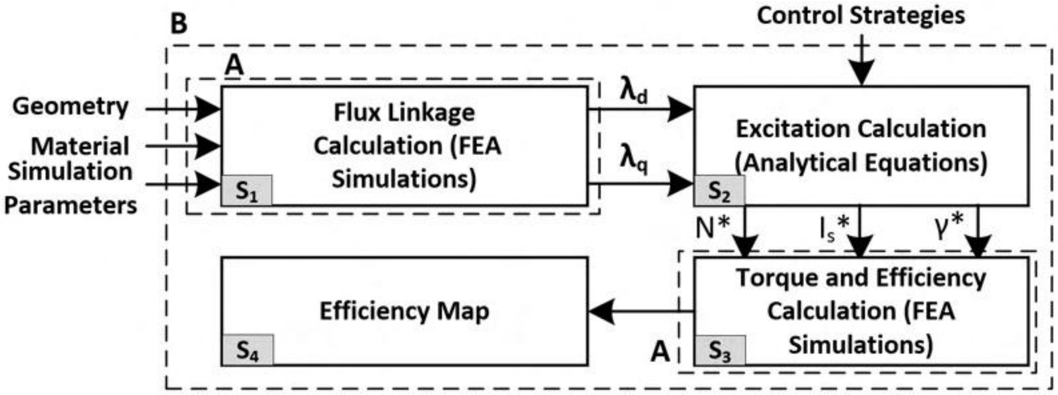

B
Control Strategies
A
Geometry
Material
Simulation
Parameters
Flux Linkage Calculation (FEA Simulations)
S₁
λd
λq
Excitation Calculation (Analytical Equations)
S₂
N*
Is*
γ*
Torque and Efficiency Calculation (FEA Simulations)
S₃
A
Efficiency Map
S₄

Figure 3.5: Flowchart of the process for calculating efficiency maps.

To predict the efficiency map for a given motor geometry, a large dataset of design variations, motor geometric images, and efficiency maps is required. Latin Hypercube sampling (McKay, Beckman, and Conover 2000) was used to populate a design space of 2900 samples of a 24-slot 4-pole parameterized IPM motor displayed in Figure 3.6. A variation of ±50%

was set on each of the reference design parameters (shown in Table 3.1). Of these samples, 2500, 200, and 200 were randomly partitioned for the training, validation, and test sets. For each motor sample around 120 FE simulations were used for Stages 1 and 3.

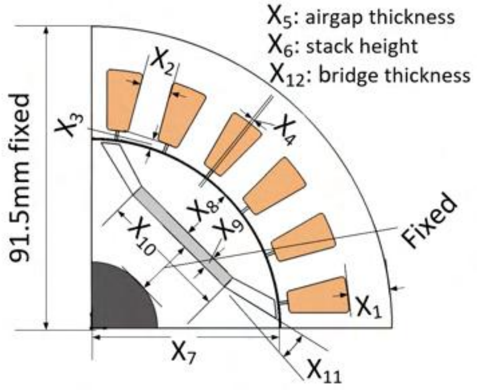


Figure 3.6: Parametrized Motor Geometry - IPM Motor.

Independent Parameters	
24 slots	4 poles
33.33 A$_{rms}$	400V_{dc}

Parameter	Value	Parameter	Value
X_1	18 mm	X_7	50 mm
X_2	5.8 mm	X_8	18 mm
X_3	3 mm	X_9	4 mm
X_4	1 mm	X_{10}	15 mm
X_5	1 mm	X_{11}	45°
X_6	100 mm	X_{12}	1.5 mm

Table 3.1: Dependent & Independent Parameters of the reference design.

3.4 Neural Networks

Due to the high cost of generating an efficiency map from detailed FE computations, Stages 1 and 3 in Figure 3.5 which usually involves FE simulations, are targeted as candidates for DL models. The first model is based on an Artificial Neural Network-Convolutional Neural Network (ANN-CNN) architecture and acts as a surrogate to Stage 1 for 'Flux Linkage Calculation' shown in Figure 3.5. Similarly, a second network discussed later, learns the behavior of the 'Torque and Efficiency Calculation' FE simulations in Stage 3 of Figure 3.5. This model is based on a combined CNN-Recurrent Neural Network (RNN) architecture. Note that the terms feedforward network and ANN are used interchangeably in this work.

3.4.1 Modular network approach

The aim of the model in Figure 3.7 is to predict the values of dq flux linkages (Stage 1, Figure 3.5), necessary for characterizing a motor drive. This model consists of three sub-networks. The first takes the CAD geometry, which is parametrized to change dimensions to build a design space consisting of 2900 different designs (samples), along with the information of material distribution in the form of a pixelated image as input to a CNN. In parallel, the current excitation (I_s, γ) represented as a 1-D vector is fed to the first feedforward network. After the information passes through a series of hidden layers on both sides of the network

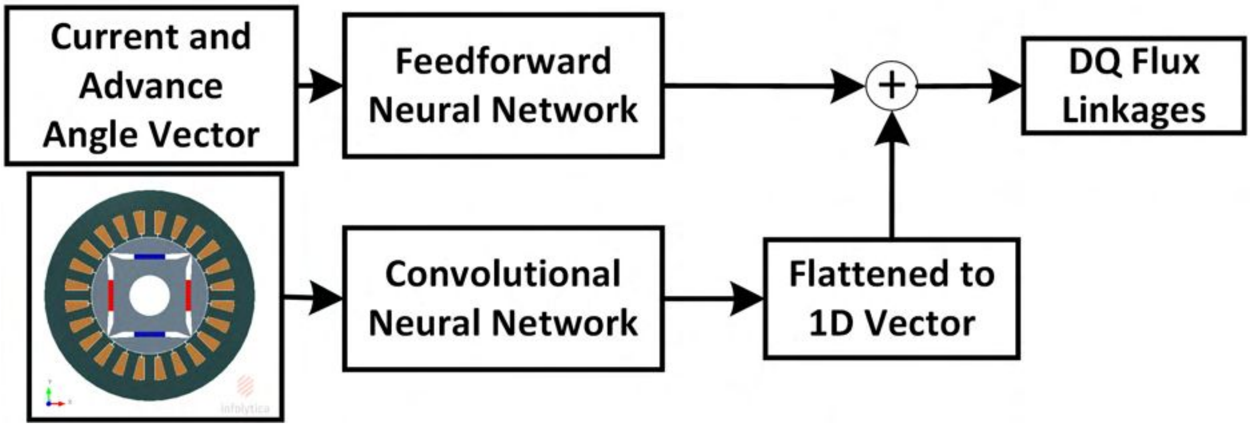

Current and Advance Angle Vector
Feedforward Neural Network
+
DQ Flux Linkages
Convolutional Neural Network
Flattened to 1D Vector
infolytica

Figure 3.7: Neural network-based flux linkage prediction.

(ANN and CNN), the hidden output from the CNN is flattened and merged with the hidden vector of the first ANN, which is fed to the second feedforward network, which then predicts the target dq flux linkages. The flux linkage map, obtained by interpolating the predicted dq flux linkage values, is used for calculating the optimal control values $(N^*, I_s^*, \gamma *)$ in Stage 2. These points along with the pixelated input of CAD geometry and material distribution, are used as input for the Stage 3 surrogate model, as represented in Figure 3.8 to predict the corresponding efficiency (η^*) values. However, due to the different torque-speed range of various designs, the output length depends on the number of inputs. This means that a model architecture suited to handle variable input sequence length is required. Although an efficiency map is represented on a 2D map, the torque-speed points can be represented in the form of a 1D sequence. The conversion of a 2D map to a 1D sequence can be done in many ways - row first, column first, or in any random order. The architecture for performance should be independent of the order of the sequence and should rather be able to identify the spatial relation over a large separation over the sequence of the input. For these requirements, a popular DL algorithm for such sequence-to-sequence modeling is based on using RNNs with an attention mechanism to handle long-term dependencies as in Figure 3.8 (a). This part of the model is called the 'Encoder'. Finally, a dense time distributed layer in Figure 3.8 (b) accumulates information from all the sub-networks and works as a decoder to predict the target in the form of a sequence of efficiency values.

There are five inputs fed to the sequence-based model: the excitation (N^*, I_s^*, γ^*) and torque values (T_{em}^*) along with the base speed (N_{base}^*), for all the points where efficiency values are to be calculated, as displayed in Figure 3.8 (a). An RNN cell consists of hidden memory elements which are responsible for encoding long-term dependencies (Graves, Mohamed, and Hinton 2013) in the input sequence, shown with the dotted arrow in Figure 3.8 (a). Once the RNN has seen all the input sequence, the encoded hidden information is referred to as the

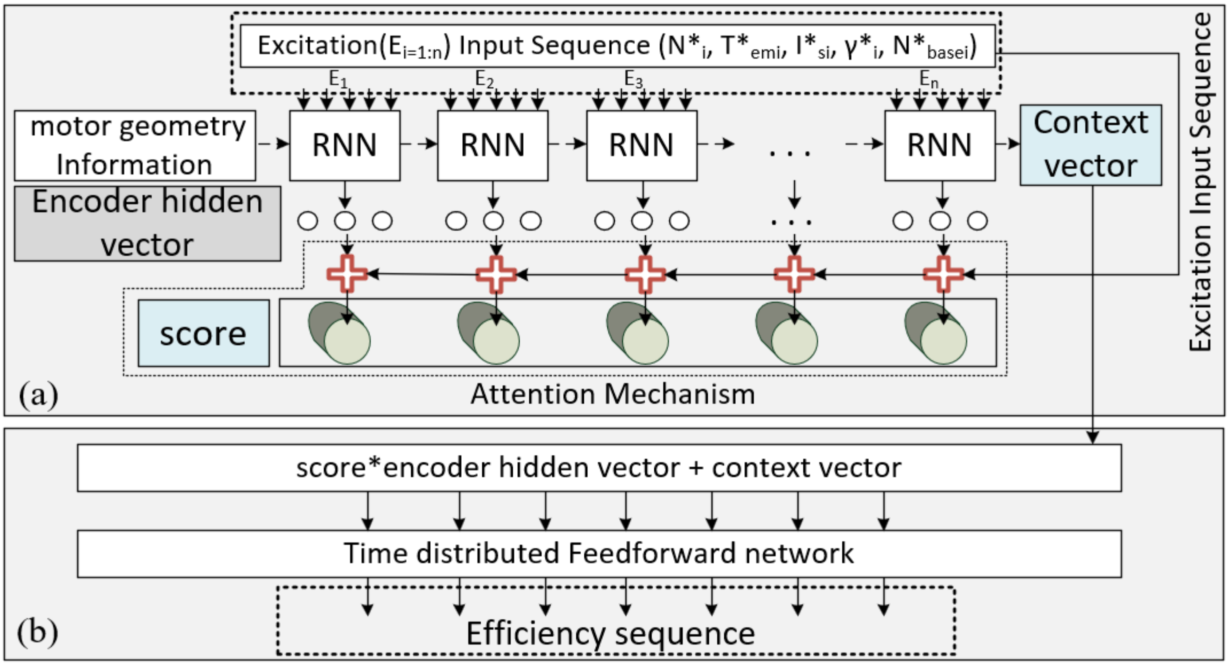

Excitation($E_{i=1:n}$) Input Sequence (N^*_i, T^*_{emi}, I^*_{si}, γ^*_i, N^*_{basei})
E_1
E_2
E_3
E_n
motor geometry Information
RNN
RNN
RNN
. . .
RNN
Context vector
Excitation Input Sequence
Encoder hidden vector
. . .
score
(a)
Attention Mechanism
score*encoder hidden vector + context vector
Time distributed Feedforward network
(b)
Efficiency sequence

Figure 3.8: Network architecture: (a) RNN with attention mechanism, (b) prediction from time distributed feedforward layers.

context. This context vector is initialized with information related to the motor geometry, provided either as geometric parameters or through a CNN as in Figure 3.8 (a), and encodes input information (geometric parameters, excitation, torque) relevant for the prediction. In brief, this vector tries to summarize the global information for a design sample from the inputs in a single abstract representation.

Even though efficiency maps can be predicted with just the context and encoder hidden vector, there is an information loss which leads to poor performance. This occurs when all the input data and motor geometric information is encoded into a potentially long sequence ($\sim 120 - 170$). To mitigate this issue, we introduce an attention mechanism (Vaswani et al. 2017) in Figure 3.8 (a). The importance of this attention mechanism can be attributed to the smooth regions of an efficiency map. For an intuitive understanding, the neighboring regions in the map can contribute to the prediction as they tend to have similar values especially in the regions of high efficiency. Thus, the closer the neighbor is to the point of prediction, the higher the voting power (i.e. attention) it gets. The calculation of attention can be costly as the model needs to compute the attention value for each combination of input and output sequence elements. As for the operation in the attention layer, the input sequence is assembled and is compared with the information captured by the encoder sequence to generate a corresponding score. As an example, assume that an attention score (a_{32}) is large, then the second encoder output state will get a lot of attention while predicting the value of efficiency for sequence step 3. This can be visualized in Figure 3.8 (b), where the encoder output is weighted with the ‘score’ value before being provided to the time distributed feedforward network. Moreover, the time-distributed feedforward network takes global information (the same for all the points) in the form of a ‘context vector’. The output of this ANN sub-network is the predicted efficiency values in a sequence. One of the important reasons for the model to handle variable sequence lengths is the property of ‘parameter sharing’. The RNNs can “roll-on” to any length of sequence as the weights of each RNN cell are shared with all the other cells. Similarly, the time-distributed feedforward network is a copy of the same network offered for the prediction of each element of the sequence.

3.4.2 End-to-end network approach

Another study is performed to find the effectiveness of a single end-to-end DL model instead of the chain of sub-models explained in Section 3.4.1. A different configuration involves data engineering such that the 2D structured input contains information about the CAD geometry and material distribution. The target is a uniform grid of efficiency values interpolated from an FE-based efficiency map. Formulating the data in such a way bypasses the dependency on the generation of the flux linkage map and excitation calculation through the control strategies. The target 2D data is formulated such that the peak value of speed (N^*) and EM torque (T^*_{em}) were normalized to 1 per sample. Thus, the efficiency map relies on a separate network, which is trained to predict the peak values associated with each design. This network for efficiency map prediction is similar to the CNN-based network discussed in Section 2.6. The network architecture is shown in Figure 3.9. The encoding and decoding sections of the network are symmetrical.

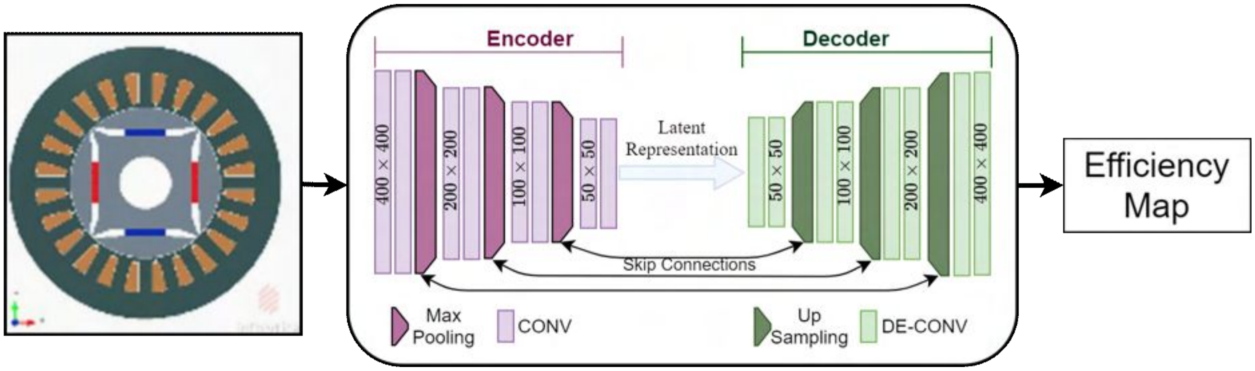


Figure 3.9: Network Architecture for End-to-end efficiency map prediction.

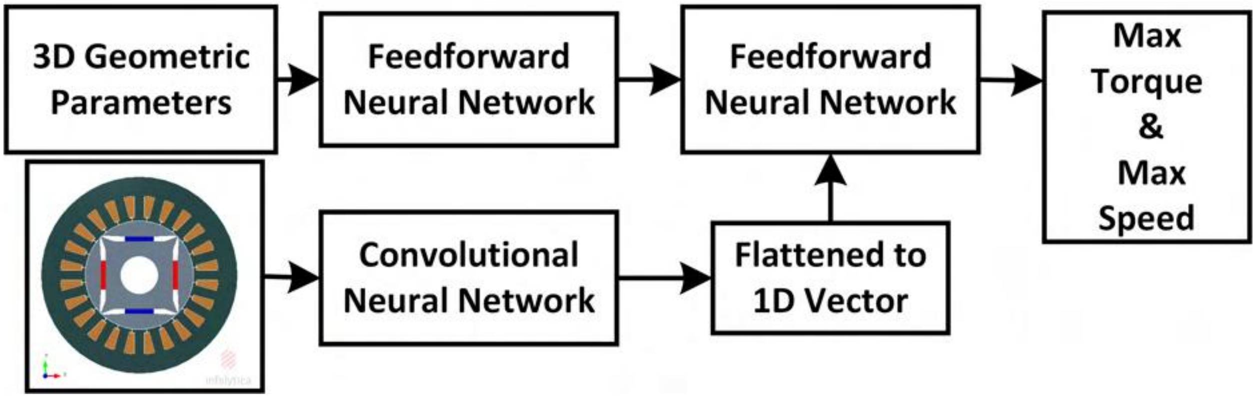

3D Geometric Parameters
Feedforward Neural Network
Feedforward Neural Network
Max Torque & Max Speed
Convolutional Neural Network
Flattened to 1D Vector

Figure 3.10: Supplementary Neural network for maximum Torque and Speed prediction.

However, the value of speed (N^*) and EM torque (T^*_{em}) in the predicted map are normalized values. The other network used to convert the normalized T^*_{em}, N^* to absolute values is shown in Figure 3.10. This supplementary network uses two sets of information as input, one from the CAD geometry and additional 3D geometrical parameters such as the stack length of the motor. Predictions from both the networks are needed to create an efficiency map.

3.5 Results of Supervised Learning

3.5.1 Modular network approach

An example flux linkage map constructed using the predicted dq flux linkage values from the DL model is shown in Figure 3.11 (a), alongside the map obtained using FE analysis in Figure 3.11 (b), and shows a close match. The root mean squared error (RMSE) of the output of the network in Figure 3.11 is compared to the ground truth over the test dataset and lies in the range of 1-2%. The predicted values of dq flux linkages are then used to evaluate the excitations to be fed into the second network for efficiency value prediction.

Two variants of this sub-network were trained. One includes a CNN which feeds the design and material distribution information to the RNN as the initiation of the hidden layer, while the other variant uses the values of the geometric parameters. The performance of both variants was comparable, with the one using data from geometric parameters marginally better than the CNN-based model. In addition, several hyperparameters were tested as part of model selection. While grid search can be employed, as it is computationally demanding and thus a random search was employed to come up with an optimal hyperparameter configuration. The RMSE over the test set was in the range of 0.75-1.5% per point. Finally, the predicted sequence of efficiency values is used to generate an efficiency map. The prediction

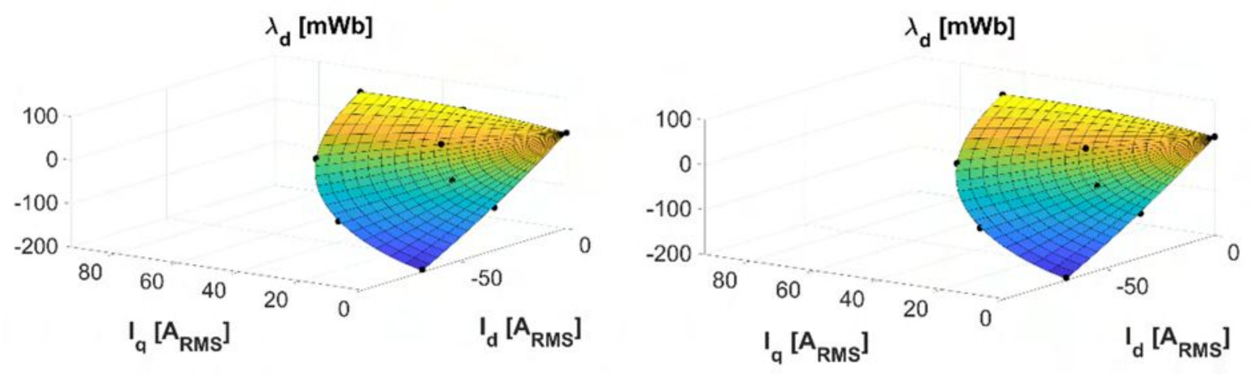


Figure 3.11: Flux linkage (d-axis) map obtained from (a) DL, (b) FE.

from the model illustrated in Figure 3.8 is shown in Figure 3.12 for two design samples.

For a more quantitative measure, the concept of a confusion matrix (Géron 2019) is used as a performance evaluator. The efficiency values are partitioned into two classes based on a threshold chosen here as 85%. The (T^*_{em}, N) points in the efficiency map either belong to class 1 ($\eta^* > 85\%$) or class 0 ($\eta^* \leq 85\%$). Based on this classification, the following metrics defined in Table 3.2, were used where CA is the classification accuracy in eqn. 3.1. The variation of CA over 100 randomly selected designs from the test set is between 90-96% indicating high accuracy in detecting high-efficiency regions in the tested maps. Since class imbalance is not an issue for this analysis, further measures derived from confusion matrix such as precision and recall are not evaluated.

	Prediction: 0	**Prediction: 1**
Target: 0	True Negative (TN)	False Positive (FP)
Target: 1	False Negative (FN)	True Positive (TP)

Table 3.2: Confusion Matrix for high efficiency classification.

$$\text{Classification Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

The results presented were based on a dataset of 2900 samples used here as a proof-of-concept. Several constraints in terms of regularization and early stopping (Caruana, Lawrence, and Giles 2001) were employed to check over-fitting during the training process. The network was trained on an NVIDIA 1050 Ti GPU and took around 5 mins per epoch with a batch size of 32. Convergence was achieved in around 25–35 epochs, resulting in a total run time of 2–3 hours. The prediction time is in the range of 10–12 seconds for 100 geometries at a time (batch size) which demonstrates an improvement over using FE simulations for this

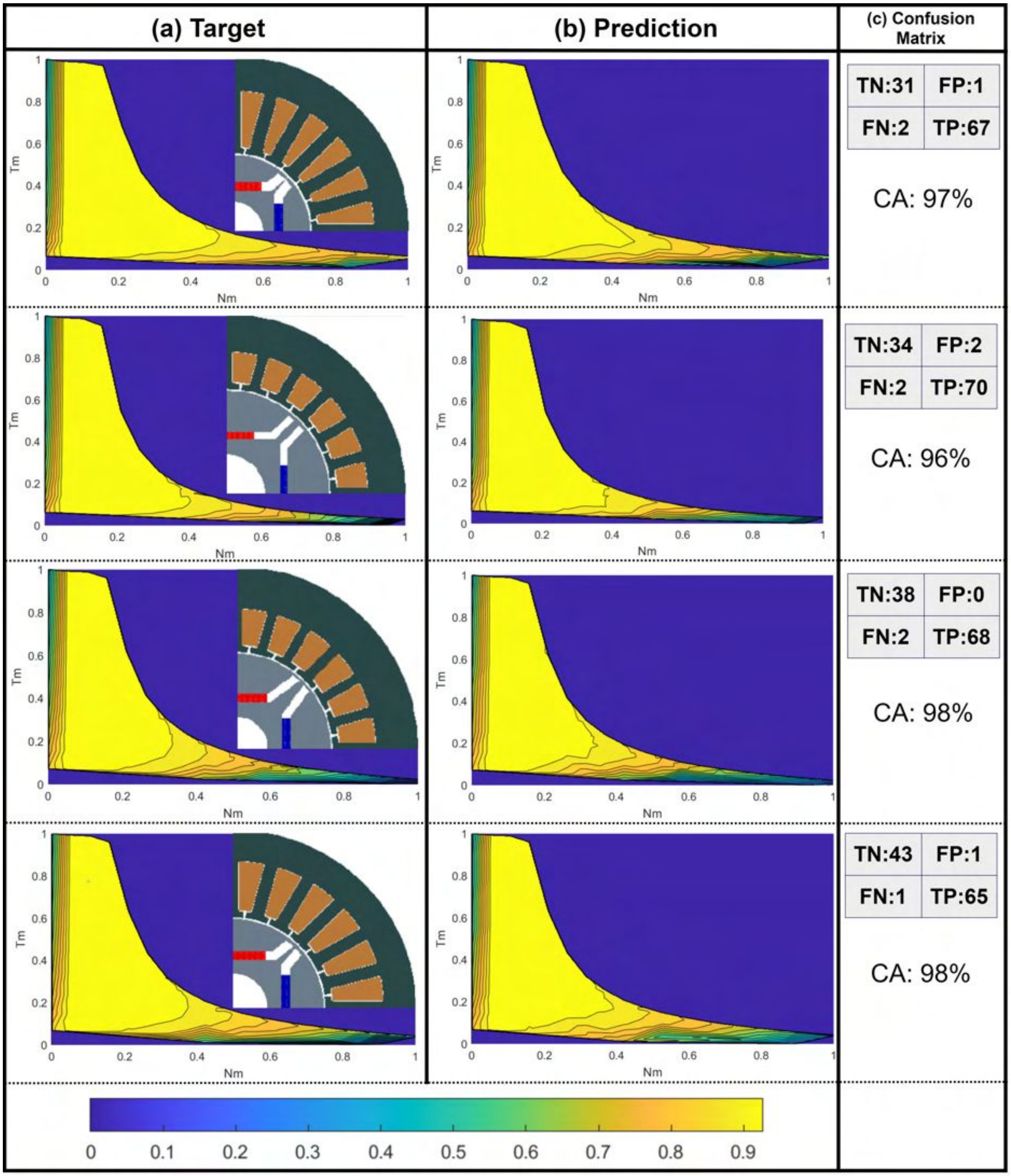


Figure 3.12: Qualitative and quantitative results of efficiency maps for two design samples: (a) Targets, (b) Predictions, (c) Confusion Matrices.

IPM motor. It should be noted that the time required to analyze multiple geometries is memory-bound rather than compute-bound, i.e. it is possible to parallelize the training and inference to thousands of samples if the memory requirements are satisfied. In comparison FE based calculation of an efficiency map can take between 1-2 hours, resulting in months to generate a dataset of this size. Further, the FE calculation is usually compute-bound.

3.5.2 End-to-end network approach

Similar to the above, the end-to-end network was trained on the same GPU and took around 10 mins per epoch with a batch size of 16. The network converged in around 40–50 epochs with a total run time of 6–7 hours. The prediction time is in the range of 2–3 seconds for 100 geometries. Figure 3.14 shows the target and predicted efficiency map for a sample IPM motor. The average mean squared error between the ground truth and the prediction lies in the range of 1-1.5% per node/point in the 2D grid for a sample, similar to the performance obtained with the Modular network approach (Section 3.5.1).

A computationally cheaper surrogate for the time-consuming task of generating an efficiency map can streamline the process of design and analysis. However, a neural network will make a prediction even if the input is completely unrelated to what it has been trained on. A motor geometry significantly different from the ones in the training set can produce substantial errors. This creates a dilemma for the user; whether to trust the prediction or not in the absence of ground truth (FE results). To rely on such a predictor (trained in a supervised manner), it is necessary to provide an uncertainty measure in the results to quantify the confidence level. One way to achieve this is to extend the DL model using a probabilistic component in the neuron weights. The mathematical study on the theory of MC dropout can be found in Section 2.8.2.

Similar to Chapter 2, Monte Carlo (MC) dropout technique is used to quantify the uncertainty in the predictions. A high value of uncertainty is an indicator that the designer should not rely on the prediction and perhaps employ conventional FE solvers instead. The error distribution, normalized for better visualization, over the efficiency map in Figure 3.14 can be observed in Figure 3.15 (a) and is highest around the edges of envelopes. A normalized uncertainty map for the same design sample using MC dropout implementation is shown in Figure 3.15 (b). It can be seen that the Bayesian approach could relate the regions of high error with high uncertainty. These uncertainty maps are computed by retrieving ten-MC samples from the network and calculating the standard deviation over the samples.

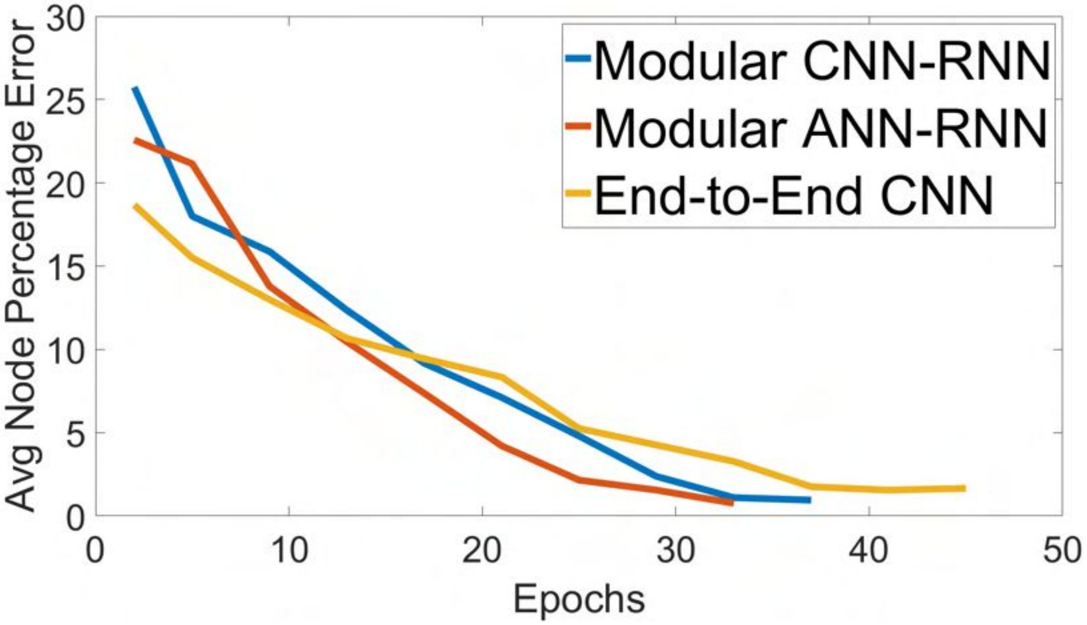


Figure 3.13: Training curves for supervised efficiency map prediction.

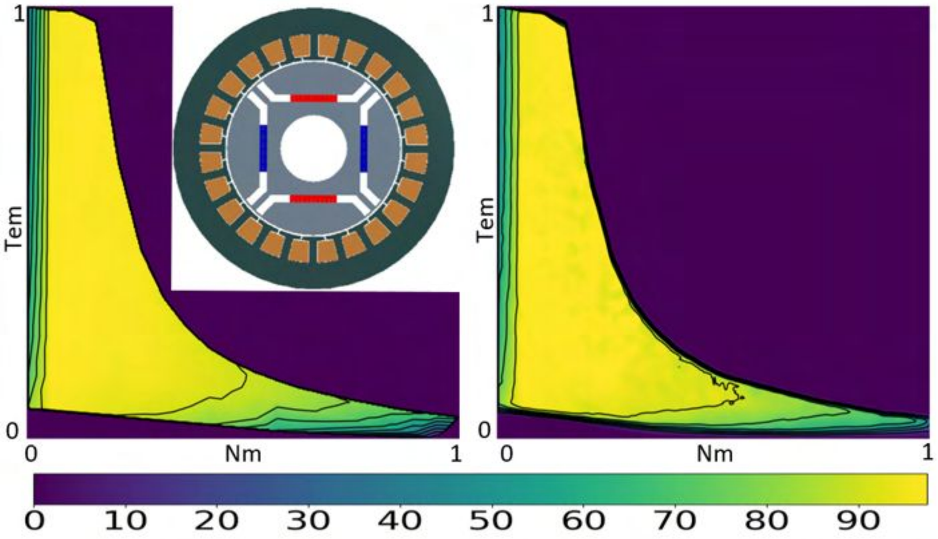


Figure 3.14: End-to-end model: (a) Target, (b) Prediction.

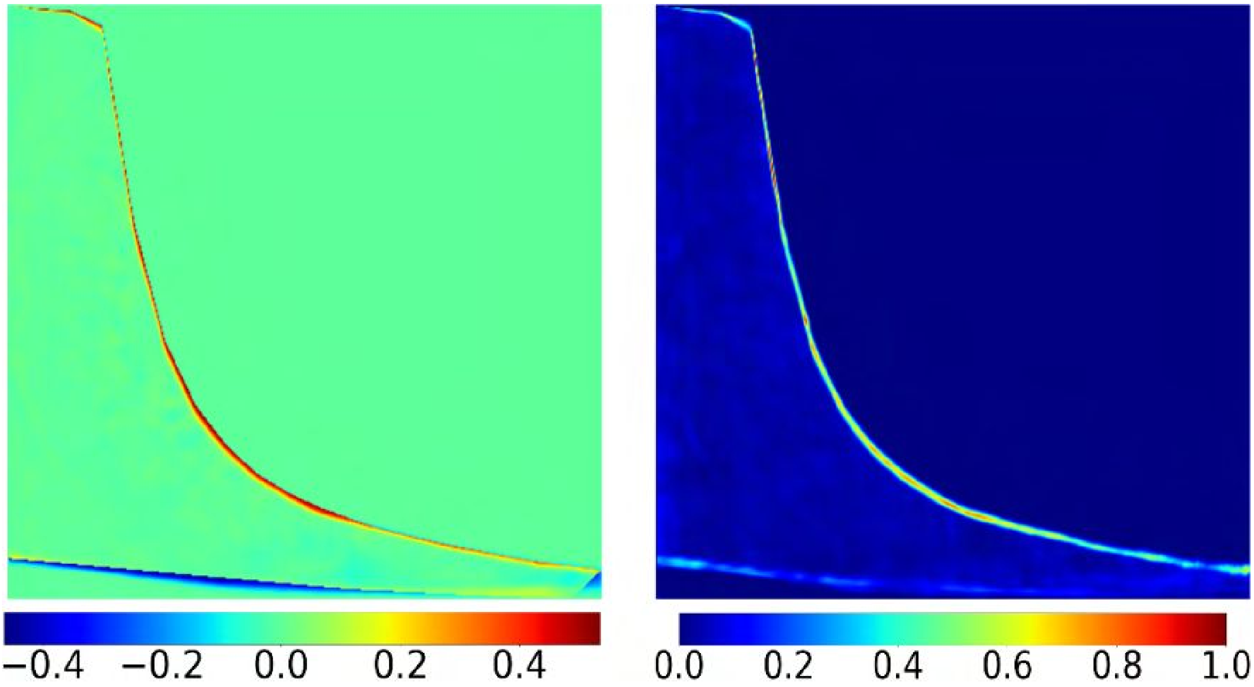


Figure 3.15: (a) Normalized error distribution, (b) Normalized uncertainty map

3.6 Challenges with Supervised Learning

By learning a representation of the efficiency map, the various models in Section 3.4 can act well as a surrogate to FE simulations for geometries similar to the training data. However, a significant bottleneck throughout this work was the computational resources required to generate the training data. The data collection process took a total of more than 2 months, with four instances of motor design space being analyzed at a time using a FE software (*Simcenter MotorSolve*). This is a major drawback of training an entire DNN from scratch (e.g. through random initialization). It is important to find a solution to this issue of the requirement that sufficient data for training is available from the same distribution.

A possible solution to reduce the computational burden on both fronts (data generation and training time) is to use the concept of 'transfer learning' (TL) (Pan and Yang 2009), where useful knowledge is extracted from neural networks trained on similar domains, i.e. motor geometries. The predictor for a new task will start its learning based on the knowledge extracted from a related task that has already been learned (Mou et al. 2016). In other words, a pre-trained network, which is initially ill-prepared for the new task, is selected to make predictions and its network architecture is incrementally modified to fit the needs of the new task. This learning concept is necessary since many different motor configurations are available, and it will be impossible to generate massive datasets for all of them. Also, a model trained on one type of geometry can be used in creating a surrogate for other performances such as power factor and loss maps. Therefore, the following sections investigate transfer learning to reduce the training time of deep neural networks applied to electric motor drives by transferring knowledge over similar tasks. Two types of synchronous AC machines are studied: an interior permanent magnet (IPM) (Khan, Ghorbanian, and Lowther 2019) (the same as one used in Section 3.3) and a fractional-slot concentrated winding permanent magnet (FSCW PM) (Reddy et al. 2012; Rahman et al. 2017). Design spaces of motor geometries are fed as inputs to the DNNs, which then output performance maps of motor drives, such as efficiency and power factor, as described below.

3.7 Transfer Learning Methodology

3.7.1 Dataset Generation

The technique for data generation is similar to that employed in Section 3.3. Different datasets of motor designs and performance maps are needed for the transfer learning problem. Each parameter of the reference designs in Figure 3.16 was varied by ±50% of the nominal value. Next, Latin Hypercube sampling was used to generate a design space of 3200 samples for the parameterized PM motors. From these generated samples, 2500, 200, and 200 were randomly partitioned for the training, validation, and test sets, respectively. The additional (300) samples of the PM motor were not used for the TL process (Section 3.8.1 & 3.8.2) and act as a 'data sufficiency' check for Section 3.8.3.

To calculate the performance maps, i.e. efficiency and power factor, a similar procedure as in 3.3 and (Mohammadi and Lowther 2017; Khan et al. 2020a) is adopted as shown in Figure 3.17. For a given motor geometry and a set of parameters, electromagnetic finite element (FE) simulations (*Simcenter MotorSolve*) are run at various dq currents for motor characterization. Different motor control strategies are then solved to find the optimal excitation conditions for every speed. These points are used to compute optimal electromagnetic quantities, including torque, efficiency, and power factor, using FE simulations. Finally, the desired performance map is generated in the post-processing stage. Here, two DNNs were used to act as surrogates for the FE analysis, namely for Stages 1 and 3 of Figure 3.17.

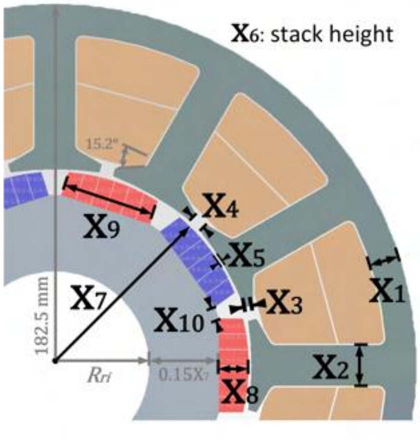


Figure 3.16: Parametrized Motor Geometry of FSCW PM Motor

Independent Parameters	
12 slots	10 poles
300 A_{rms}	$450V_{dc}$

Parameter	Value	Parameter	Value
X_1	13 mm	X_6	115 mm
X_2	25 mm	X_7	122.5 mm
X_3	4.3 mm	X_8	12.5 mm
X_4	7 mm	X_9	60 mm
X_5	1.0 mm	X_{10}	0.25 mm

Table 3.3: Dependent & Independent Parameters of the reference design for a FSCW PM Motor

3.7.2 Problem Definition of Transfer Learning

Transfer Learning is a well-studied topic in the field of machine learning. This work focuses on performing model-based knowledge transfer, which is the most popular methodology for DNNs (Pan and Yang 2009). The main aim is to enhance the 'generalizability' of DNNs, which is necessary to reduce the future costs of training networks on similar tasks and at the same time improve the network's robustness. The magnitude of similarity between two ML tasks is not a well-defined metric but an attempt can be made by relating the input features and the label space of the task. Using the notation defined in Pan and Yang 2009, the task of prediction using a ML model can be seen as a representation of the domain to the target label. The domain ‘D’ is composed of two components: a feature space X and a marginal probability distribution $P(x)$, where each input observation in the dataset belongs to X, i.e. $x \in X$. For a given domain, a label space ‘L’ also has two components Y and $P(y|x)$, where the probabilistic function denoted by $P(y|x)$ can make predictions on instances from X. In the case of (Khan et al. 2020a), the labels were efficiency values for each design point and a design point was represented by the geometrical features (either as CAD image or design vector) and the excitation points. For supervised learning, we require data for a domain and corresponding data for labels. On the other hand, for TL, a model must be trained on a source domain and labels (D_s, L_s) and also on target data for the domain and labels (D_t, L_t) where the original knowledge is to be transferred. This work explores knowledge transfer capability over two scenarios as shown in Figure 3.18. In the first scenario, the source task of efficiency map prediction estimation is replaced with the target task of a power factor map, over the same IPM geometry. In this case the source and target domain remain the same $(D_s = D_t)$, but the label space changes $(L_s \neq L_t)$. For the rest of the Chapter, this task will be referred to as ‘Internal Knowledge Transfer’. On the other hand, the task referred to as ‘External Knowledge Transfer’ involves the prediction of efficiency maps but over a different design space of a FSCW Surface-mounted PM motor. This is a comparatively less similar task since neither the domains nor the labels necessarily overlap ($D_s \neq D_t$ and $L_s \neq L_t$) completely. However, given the domain spaces (D_s, D_t), there can be some resemblance over the labels since both Y_s and Y_t are bounded between 0-100% and the conditional probabilities can also be relatable. Calculating performance maps remains the same based on the domain knowledge. Understanding the source and target task is important as it was observed that very dissimilar tasks fail in the transfer learning process (Mou et al. 2016).

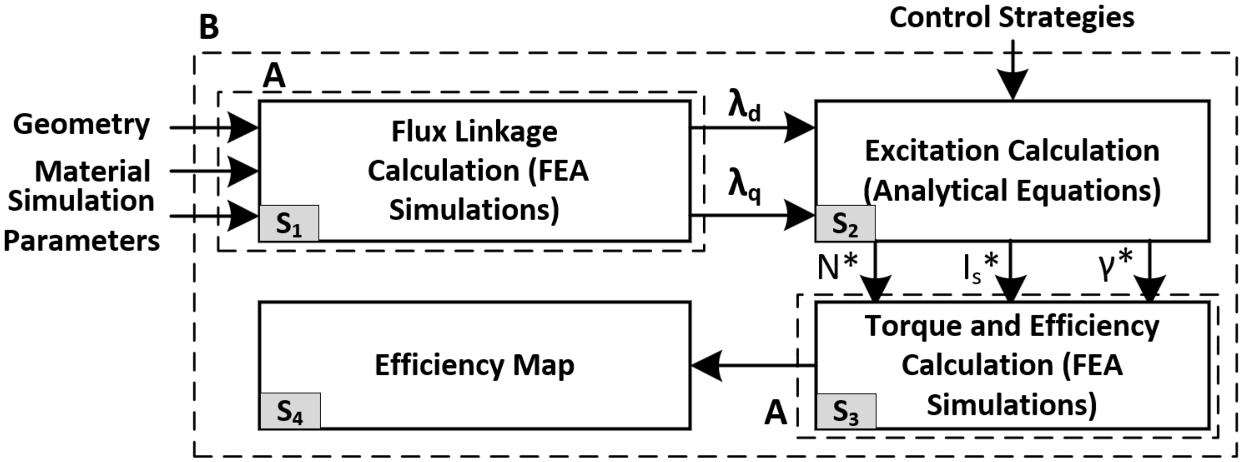

B
Control Strategies
A
Geometry
Material
Simulation
Parameters
Flux Linkage Calculation (FEA Simulations)
S_1
λ_d
λ_q
Excitation Calculation (Analytical Equations)
S_2
N*
I_s*
γ*
Torque and Efficiency Calculation (FEA Simulations)
S_3
A
Efficiency Map
S_4

Figure 3.17: Flowchart of the process for calculating performance maps.

It is also important to distinguish TL from a closely related learning paradigm called multi-task learning. Although both aim to generalize and exploit the common knowledge shared by different tasks, the approaches are different. TL focuses on learning a new task using some source task as auxiliary information. Conversely, multi-task learning tries to jointly improve the general performance of all the constituent tasks without any source or auxiliary information. As such, multi-task learning requires training data for all the tasks to be learned. The goal of multi-task learning is not to reduce the data dependency or the computational load. This also highlights the advantage of TL in cases where the source pre-trained model has millions of weight parameters and is trained on massive datasets that needed significant GPU hours. Access to only the trained weights along with the network architecture is needed for transferring knowledge to a target task.

3.7.3 Knowledge Transfer Procedure

After identifying the source and target tasks, the next step is the procedure of transferring the knowledge from the source to the target. Only Stage 3 of Figure 3.17 is used here for TL, since the data and computation required for Stage 1 was significantly lower than that for Stage 3, i.e. Stage 1 can be trained in a supervised manner. The source training data remains the same, as used in Section 3.3 to train the network in Section 3.4. For Stage 3, the general approach used to achieve successful transfer of knowledge is by combining the techniques of greedy pre-training followed by fine-tuning the overall model. The idea of greedy layer-wise pretraining is widely used in training deep belief networks and autoencoders (Goodfellow et al. 2014). However, the DNN weights, in this case, are trained in a supervised manner. A certain part of the pre-trained network is identified as prior knowledge, which is useful for the new task and can be identified as the common knowledge between the source and target task. Greedy pre-training 'freezes' the prior knowledge part of the pre-trained network and substitutes the rest of the network with corresponding 'new' layers. The terms 'frozen' and 'new' refer to whether the gradient can flow through the layers during training through backpropagation. A 'frozen' layer will not see a change in its weights during the training procedure, whereas a 'new' layer, initialized either randomly or using one of the initialization techniques (Goodfellow et al. 2014), is trained in a supervised manner during the pre-training stage.

Once the new layers are sufficiently trained and are assumed to work in synchronization with the 'frozen' or prior section of the network, the whole network is fine-tuned. This process is similar to supervised learning, where the gradient flows through the whole network, making changes in the weights of all layers and no part of the network is in the 'frozen' state anymore. One distinction in practice is reducing the learning rate during the fine-tuning step, as large

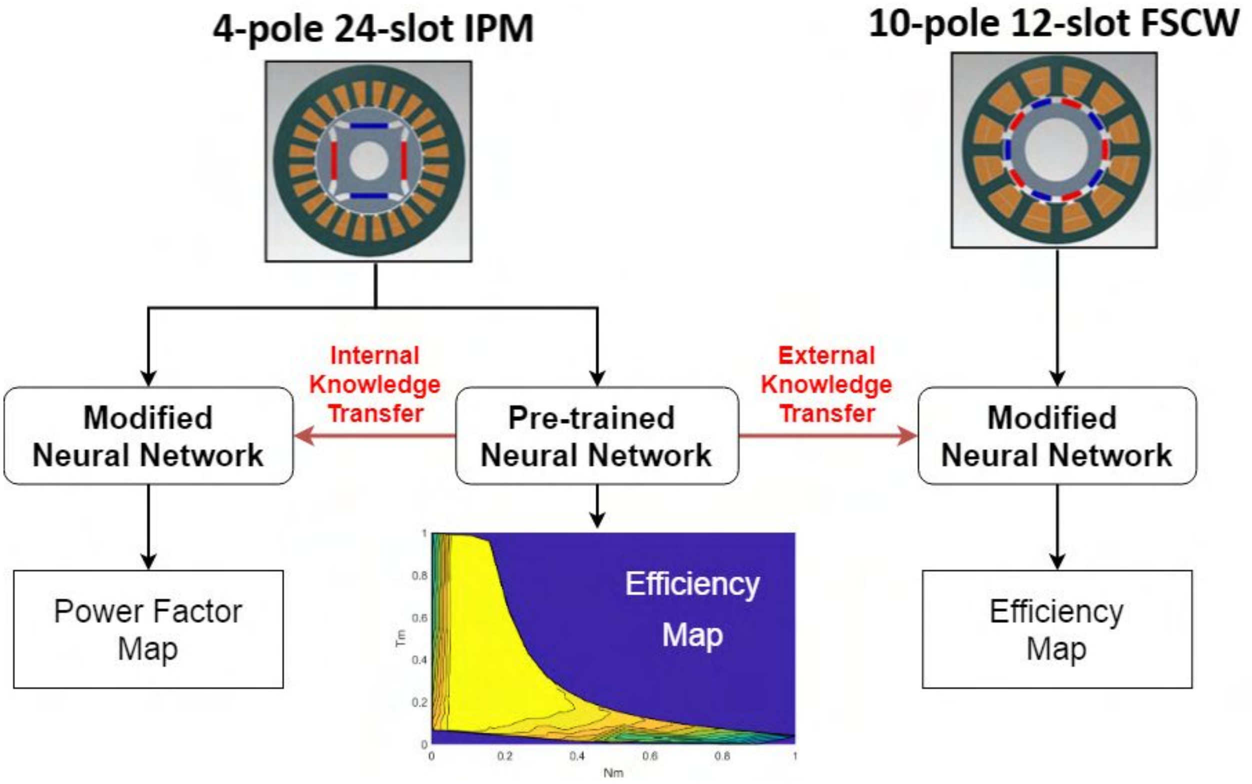


Figure 3.18: Transfer learning tasks: Internal Knowledge Transfer (IPM to IPM) and External Knowledge Transfer (IPM to PM).

changes in the pre-trained part of the network can compromise the highly optimized weights from the source model. Although guidelines for the learning rate are not well defined, in this work the learning rate was reduced by an order of 10 between the greedy pre-training and fine-tuning step. Ideally, the pre-trained network must be retained as much as possible to leverage the powerful expressive ability of deep learning models. The decision to freeze or replace a part of a network is highly task-dependent and certain guidelines that were observed during this work are discussed next.

3.8 Results of Transfer Learning

As explained earlier, transfer learning aims to decrease the computational load by reducing the data requirement and training cost on the target task. To compare the training time, supervised learning on the same target task serves as the benchmark. Once the ideal performance for a task is identified, the training data is systematically reduced until there is a significant drop in model performance and the data requirement threshold for successful knowledge transfer is saved. To quantify the change in weights and to compare the relative change with other layers in the same network, the weight distribution of each layer from the pre-trained network is compared with the target network through a hypothesis test of statistical significance between the two means of the weights:

$$H_0 : \mu_s = \mu_t \tag{3.2}$$

$$H_a : \mu_s \neq \mu_t \tag{3.3}$$

Here, the α-value (coverage) is set to 0.05 or 5%.

Instead of using classical formula based statistical methods for hypothesis testing, a resampling based technique is used (as described in Chapter 5, Bruce, Bruce, and Gedeck 2020). The procedure behind this resampling based hypothesis test is to combine the data from both sets (pre-trained and trained weights) and repeatedly draw samples without replacement from the permuted data. On calculating the quantity of interest (difference in weights between the two sets), one can obtain the distribution due to random variation in the weights of the two NNs. Based on the hypothesis test, if there is no significant difference in the mean values of layers from the source and target model, then these layers are denoted by the 'green' color in Figure 3.19. A p-value failing to cross the threshold results in failing to reject the null hypothesis, i.e. there is no significant difference in the mean values of both layers from source and target models. These insignificant layers are denoted by 'green' color in Figure 3.19. The largest change in weights for a layer in the network is denoted by the color 'red'. The level of change is relative to the change observed in other layers of the same network and all the changes in between the two extremes ('red' 'green') are denoted with 'yellow' color. To quantify these changes, the architecture is kept the same between the source and target neural networks. No additional layers were added to the source network in this study.

3.8.1 Internal Knowledge Transfer

To transfer part of the pre-trained network that was originally trained on efficiency maps to power factor maps, the domain remains the same ($D_s = D_t$), i.e. the same geometric variations of the IPM motor are used. Since the label space for the target is significantly different from that of the source, the output layer (time-distributed dense layer) of the pre-trained network was replaced with a new layer. The process of greedy supervised pre-training was used to train this layer with a learning rate of 1×10^{-3}. After sufficient training, the network was fine-tuned with changes performed in all layers at a learning rate of 1×10^{-4}. Overall, TL managed to save about 50-55% in training time compared to a cold-start supervised learning process as shown in Figure 3.20. All other configurations for using different parts of the network for pre-training and fine-tuning were tested and did not result in a better performance than the one shown in Figure 3.19. The majority of change in weights is observed in layers closer to the output, which matches with the discussion in Section 3.7.2. Similar to 3.5, the normalized RMSE over the test set lies in the range of 0.25-1.0% per point in the *pf* map.

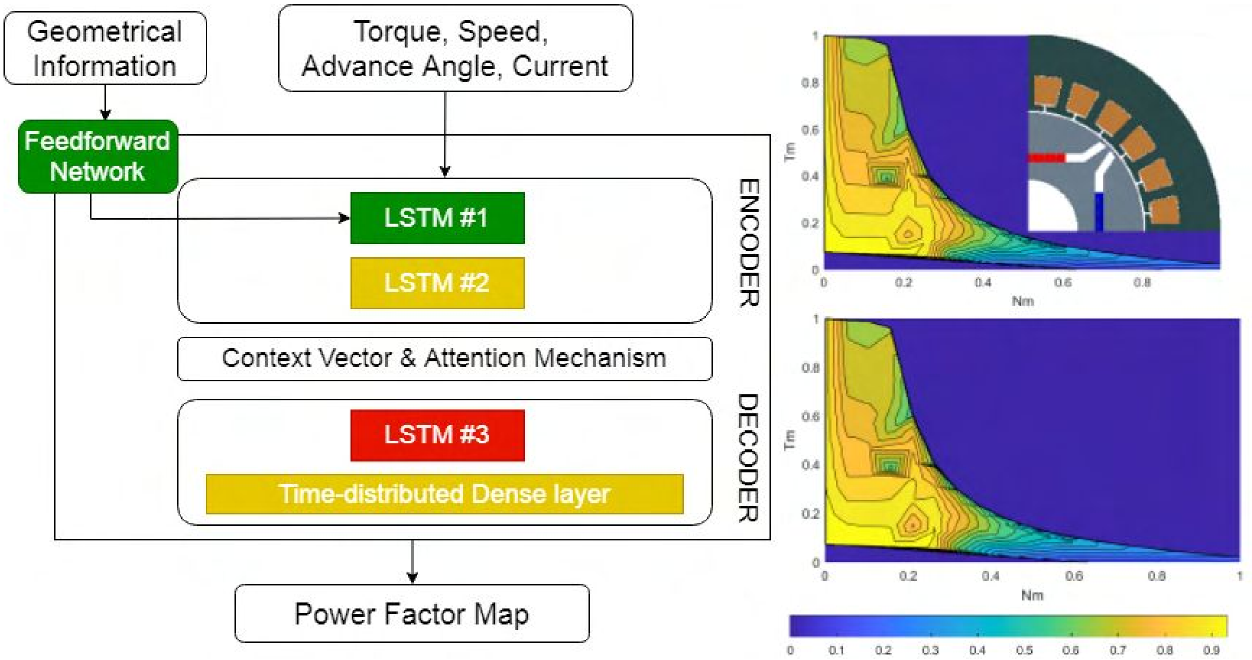


Figure 3.19: Change in weights of each layer: Efficiency map to Power factor map (IPM to IPM)

3.8.2 External Knowledge Transfer

For the External Knowledge Transfer task, the difference mentioned in Section 3.7.2 lies with both the domain and label parts of the task, which differs from the original one. Also, the geometrical features do not match as seen in Figure 3.18. Hence, the feedforward network must be modified completely with new dimensionality to match the input dimensions, but at the same time should work in tandem with the rest of the network. The rest of the network is tuned only after appreciable training is completed with the input part of the network. However as seen in Figure 3.21, the changes in the network are distributed evenly near the input and output layers, signifying the hypothesis in Section 3.7.2. The normalized RMSE value per point lies in the range of 2-3%, which is slightly worse than that obtained for the IPM Motor in Section 3.5 ($\approx 1.5\%$). As observed in the training curves of Figure 3.22 the computation load for training has been reduced by about 40-45% as compared to a

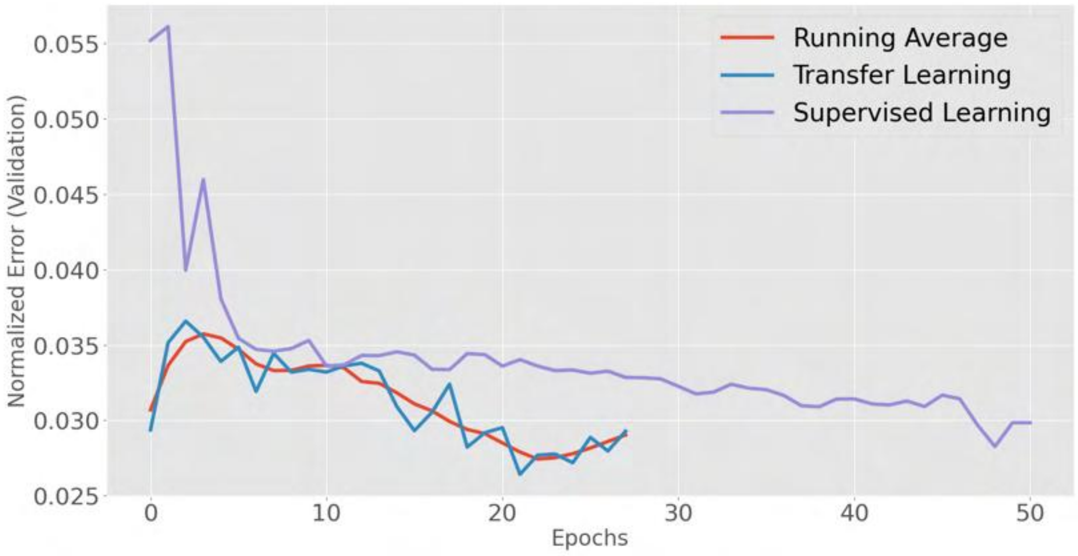


Figure 3.20: Comparison of training curves between supervised learning and transfer learning tasks: Efficiency map to Power factor map (IPM to IPM)

supervised learning approach on the same task.

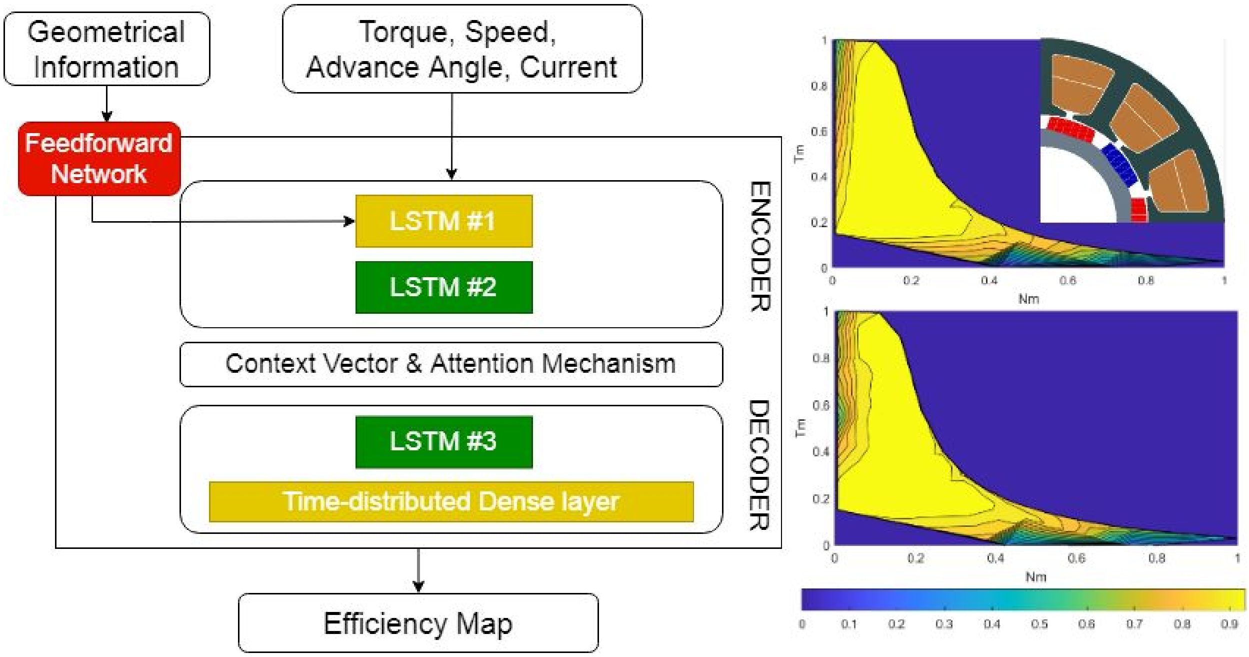


Figure 3.21: Change in weights of each layer: Efficiency map to Efficiency map (IPM to PM).

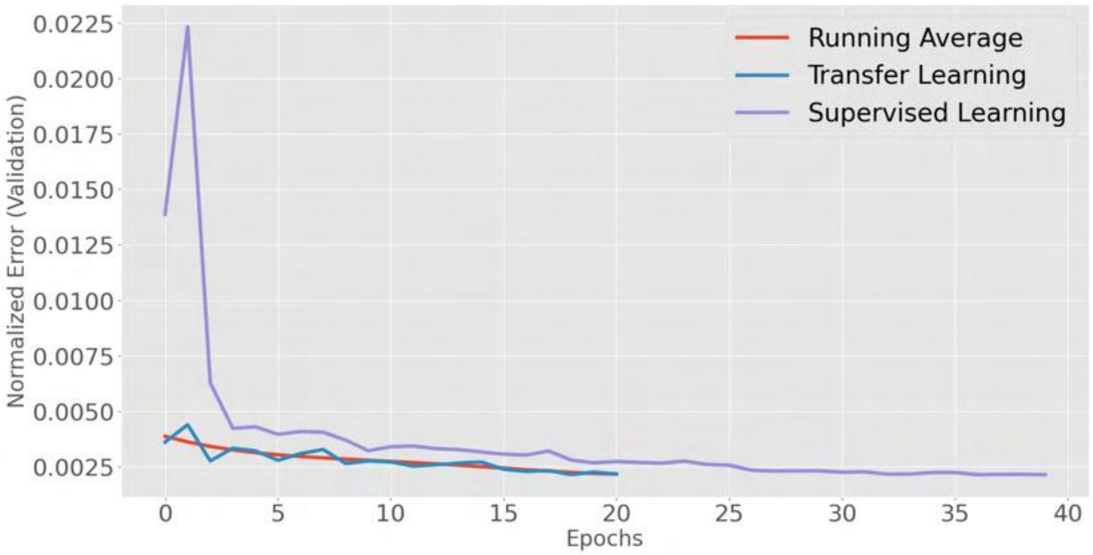


Figure 3.22: Comparison of training curves between supervised learning and transfer learning tasks: Efficiency map to Efficiency map (IPM to PM)

3.8.3 Data Requirements

The other significant contribution of TL is the reduction in the training data needed to successfully train a network. For this reason, part of the overall data is held out to check for ‘data sufficiency’ (300 samples). The remaining data is divided into the usual components of train, validation, and test parts for TL. The training data is continuously decreased until a significant drop in model performance is observed. This performance is observed over the ‘data sufficiency’ part of the dataset. This study yields the threshold of data required to successfully perform transfer learning on this task with this network. A grid search was performed with the percentage of data used for TL starting from 100% and randomly removing 5% of the data at each step of the search. With this study, it was observed that only 60-65% of the data is needed for Internal Knowledge Transfer and about 75-80% of the data needed for External Knowledge Transfer. This demonstrates that the more dissimilar the source and target tasks are in terms of either domain or label space, the more data will be needed for successful TL processes.

Multiple runs for each test (for TL tasks and Data sufficiency check) were performed with the random seed being the only difference.

3.9 Conclusion

This work shows that by learning a representation of the efficiency map, the various models in Section 3.4 can act as a surrogate to FE simulations with high accuracy, for geometries similar to the training data. For geometries significantly different from the training data, transfer learning was employed. Transferring knowledge from one domain to another allows machine learning systems to extend their range of applicability beyond their original creation. Instead of training a neural network from scratch for every problem, transfer learning enables the reuse of models resulting in fewer computational resources being needed. The results demonstrated that for both case studies (i) the training time was reduced by more than 40% and (ii) at most 80% of the training data. This generalization ability also helps to make machine learning models more accessible and more robust in many areas where capabilities or resources such as computing power, data, and hardware might be scarce. Once two domains are known to be close to each other, deep neural network models can be propagated from a well-developed domain to a less-developed domain. Also, in situations that differ from the training data, the trained models need adaptation, before they can be used for the target task.

Finally, it was also found that all the studied models predicted the efficiency maps with high accuracy, and with the advantage of GPUs and probabilistic methods, a parallelizable and robust method to produce efficiency maps is feasible.

Chapter 4

Sequence based Topology Optimization

4.1 Introduction

Topology optimization (TO) is an advanced computer-based design method used for creating efficient topologies via a reasonable material distribution satisfying performance requirements and constraints. Domains such as aerospace (Zhu, Zhang, and Xia 2016), automotive (Jankovics and Barari 2019), civil engineering (Jewett and Carstensen 2019) and mechanical engineering (Guanghui et al. 2020; Zhu et al. 2021) use this sub-field of engineering proactively to create high-performance, lightweight and innovative design solutions that will outperform manual designs. TO algorithms are capable of creating sustainable designs while still being rooted in sound logic. Also, topologically optimized products save resources through weight reduction, by minimizing material usage. A great example is how General Electric used TO to reduce the weight of an engine bracket by 84%, as shown in Figure 4.1. This modification in a small part saved the airlines nearly 31 million dollars by improving the overall energy efficiency (Morgan et al. 2014). Another useful example of a TO solution is the design of the droop nose ribs for Airbus 380, which achieved the required mechanical performance requirements along with reducing weight as compared to the conventional design (Krog, Tucker, Rollema, et al. 2002).

In recent years, the significance of TO has increased in the manufacturing industry with the development of Additive Manufacturing (AM) techniques. Many industries are nowadays looking towards advanced design methods like topology optimization and generative design (Atkinson and Ezell 2019; Coop 2018) for designing future products, including electric machines. Electrical machines are the heart of many appliances, industrial equipment,

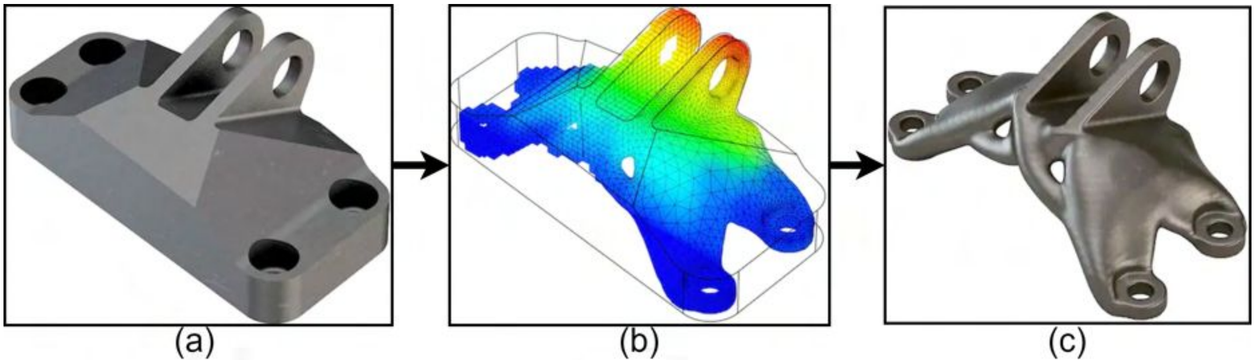


Figure 4.1: GE Aircraft mounted bracket challenge (Morgan et al. 2014) (a) Conventional mounted bracket (b) TO based design analysis (c) Optimized design

and systems. They are the foundation of the power industry and the core components of industrial machinery (Lei et al. 2017). It is estimated that electrical machines use about 43% - 46% of total electricity generated globally (Waide and Brunner 2011), resulting in about 6040 Mega-tonnes of CO_2 emission. This is the largest portion of electricity use among other utilities, as shown in Figure 4.2. Therefore, electric machine energy efficiency is crucial for energy conservation, environment safe-keeping, and global sustainable development. Consequentially, high-efficiency motors will dominate the market development of electrical machines worldwide as discussed in Chapter 3.

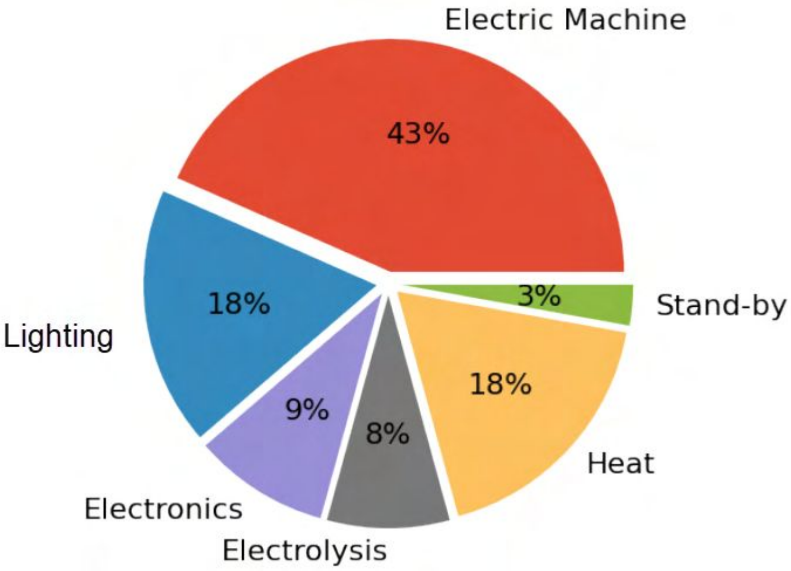


Figure 4.2: Global Electricity consumption (Lei et al. 2017).

In the context of global sustainability, the demand for sustainable alternatives is increasing and electric machines must also fulfill such requirements. As such it is expected that electric machines of the future will be optimized not only physically and technologically but also for their environmental impact. Therefore, their design optimization process becomes more and more complex as more engineering disciplines/domains and constraints are involved, such as electromagnetics, structural, mechanics, and heat transfer (Lei et al. 2017). This will result in more and more companies in the design and manufacturing industry employing TO in search of environment-friendly designs.

4.1.1 Topology Optimization

The theoretical description of topology optimization originates from the mechanical domain, which can be traced back to Michell's optimization paper, published in 1904 (Michell 1904). The decade of 1950-60 saw a renewed interest in this sub-field of engineering, with work focused on the optimal layout design of trusses (Cox 1958; Hemp 1958). According to the documented literature, the surge in layout theory research in the 1960s and 1970s led to many significant publications (Cox 1958; Prager and Taylor 1968; Dobbs and Felton 1969). Topology optimization was first applied to electromagnetic (EM) field by (Dyck and Lowther 1996), which proposed the Optimized Material Distribution method.

Before discussing TO in-depth, it is essential to understand the motivation behind employing TO over other designing techniques. While designers are known to have excellent imagination, they are prone to biases. The way they have worked in the past influences their future decisions when it comes to modelling. In such cases, the initial geometry can either come from the literature or be based on the designer's knowledge. In either case, the proposed designs tend to be similar to the ones already proved to be efficient. In such cases, even if the design is justified, it might not be the most efficient one. Further, the design requirements may be unique in which case previous knowledge may not exist. In cases, such as these, TO is very useful. TO introduces new shapes and sizes that a designer could not have come up with. These shapes and sizes are generally more efficient at accomplishing the product's intended task and free from any priors and biases, as decision-making is solely based on input data.

There is no fixed procedure for the design optimization of a device. However, there are some routine steps to be followed. These are briefly described as follows.

1. **Define the domain**: Select a possible design type or topology, which serves as the design domain. Materials and dimensions are set according to the requirements given by the application and users. The main aim of this stage is to find a feasible scheme for a given application that can be explored for an optimal solution.

2. **Performance and Constraints**: Quantifies the performance/likelihood of the design to be a fit for the requirements. For example, the performance parameters of an electric device can include efficiency, force, torque, etc., while common design constraints are material and manufacturing costs, volume, etc. The main aim of this step is to obtain several design options that may be suitable for the specific application.

3. **Optimization process**: The main target of this stage is to improve the performance of the candidate design proposed in each iteration through some optimization algorithms, to finally converge to an optimal design. This includes implementing multi-physics analysis for each design option generated in the previous step. Due to the multi-physics nature, many engineering disciplines have to be investigated in this step, such as electromagnetics, mechanics, structural, and heat transfer. As an outcome, an optimal design is obtained for a single objective design problem, or some non-dominated designs (also called Pareto optimal solutions) are obtained from a multi-objective design problem.

4.1.2 Design Optimization Methodologies

There are various methodologies designers have employed over the years to find a design which satisfies all the criteria. Some of the techniques popular in literature are:

1. **Sizing (Parametric) Optimization**: Designs are parametrized by a few geometric variables (such as thickness, width, length), as shown in Figure 4.3 (a).

2. **Shape (Geometric) Optimization**: Shape Optimization optimizes the boundary and cross-section of the components of a structure. This results in changes of the size and cross-section of the components/members of the structure, as shown in Figure 4.3 (b).

3. **Topology Optimization**: Both the shape and the topology of the design can vary without any restriction on the design, as seen in Figure 4.3 (c), where the broken lines show holes in the design.

The design variables associated with sizing optimization are simpler than shape design variables. Sizing will usually increase or decrease the geometrical parameters of a design for example the thickness, which can be very limiting in terms of freedom offered for making changes in the design. Size or Parameter optimization is the most constrained design optimization methodology in terms of the freedom of making change where only pre-defined geometrical parameters are to be changed to find an optimal structure (Haftka and Grandhi

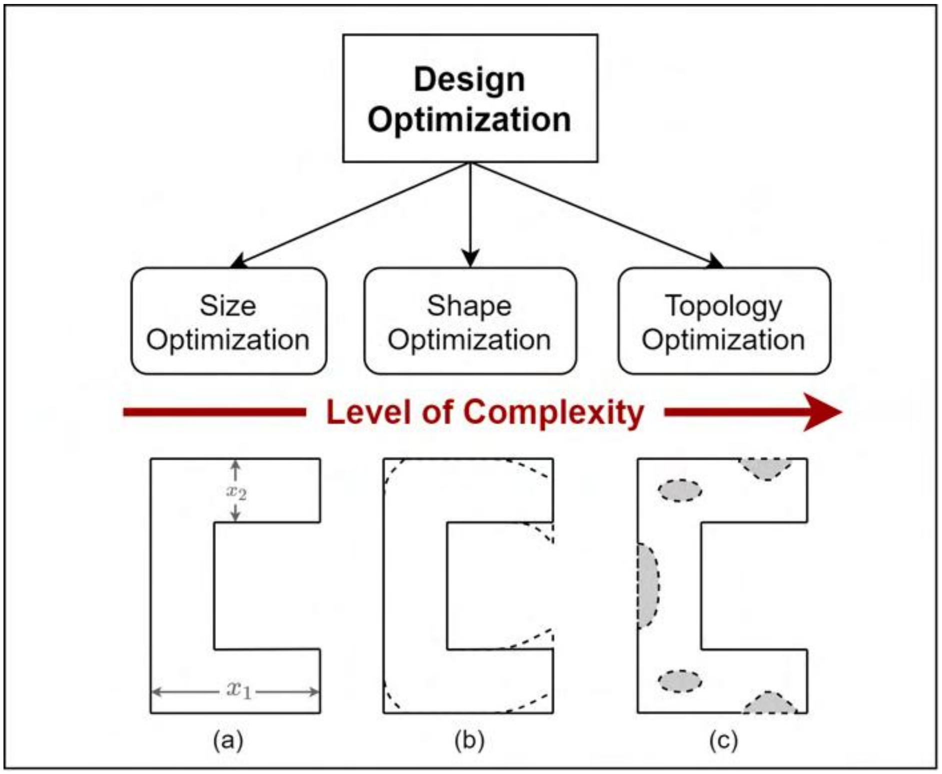


Figure 4.3: Different categories of optimal design problems.

1986). On the other hand, shape optimization offers more freedom than sizing, by modifying the shape of the boundary as shown in Figure 4.3. However, a major restriction in both types of optimization methodology (shape and size) is that the design topology is fixed and pre-defined (Bremicker et al. 1991).

TO is significantly different from the other two approaches since it necessitates transforming the design optimization problem into a material distribution problem and offers the highest freedom in terms of making changes to the design (Lei et al. 2017). It is the most flexible of all the categories of optimization methodology and designing with the advantage of free-shape optimization can outperform other optimization routines in terms of the performance if well-tuned (Garibaldi et al. 2019; Faria et al. 2015). Further, TO is used when any prior bias is to be avoided for a design. Due to this capability to generate novel designs TO has become an integral part of modern engineering design and an asset for designers.

4.1.3 Optimization Algorithms

Unfortunately, transforming the design optimization problem into a material distribution one is also why TO is difficult as the size of search space can be massive. This is related to the number of variables present in the structure to optimize.

The problem of optimal design is defined based on the following constituents (Allaire et al. 2019):

1. A model, to evaluate (or analyze) the performance of a design.

2. Objective function: which is to be either minimized or maximized. Also known as a cost function.

3. Admissible designs/Design domain: defines optimization variables, including constraints.

Before initializing a search for optimal design, it is important to define the design optimization objective, constraints, and parameters. These are usually pre-determined by the user or are derived based on the application requirements and are also dependent on the analysis model used to determine the performance of the designs. As an example, optimization of an electric motor can involve multi-objective requirements to maximize the efficiency based on the constraints of material used and external dimensions.In addition, thermal and structural aspects are also taken into account. In such a case, an electromagnetic analysis based on either FE or MEC has to be performed, along with a thermal network and structural model analysis to calculate winding temperature and resonance frequency.

Once the objectives and design analysis models are defined, the next task is to choose an appropriate optimization method. Optimization methods for solving topology optimization problems can be roughly divided into two categories: methods based on deterministic algorithms and those based on stochastic algorithms. The deterministic methods use the gradient information of the objective function, and have a quick convergence speed. However, their global search ability is poor, and they are prone to stagnate into optimal local solutions, especially non-convex problems. Moreover, the performance deteriorates when dealing with multi-objective optimization problems directly (Li 2019). Such methods are also not suitable for topology optimization problems with a large number of design variables.

On the contrary, stochastic methods are optimization methods that use random search mechanisms. The most popular of such are called Evolutionary Algorithms (EA). They have the following three characteristics:

1. **Population**: EA optimizes the current solutions to generate better solutions. The current set of solutions is called the population.

2. **Fitness**: used to quantify which of the solutions is better than others. The fitness value is calculated for each of the solutions using the fitness function.

3. **Variation**: Since the next generation of the population is an improvement over the current one, several transformations are performed on the present generation.

Stochastic algorithms are well designed to converge to global solutions of an optimization problem. The injected random variations enable the method to escape from a local optimum; therefore, these methods have a stronger affinity for global optimum than the deterministic algorithms. Further, EA are more suited for mathematical optimization problems with many local extrema. However, the main disadvantage of this kind of method is that the computational cost is high and the convergence speed is slow. This issue becomes overwhelming when a high solution accuracy is desired, resulting in finer meshes. Currently, the global optimization methods commonly applied in the electromagnetic field are genetic algorithms, simulated annealing algorithms, the tabu search algorithm, particle swarm optimization algorithm and quantum evolution algorithms. The underlying principle in these algorithms is to eliminate the inferior solutions and advance the superior solutions through the iterative process of mutation and cross-over, and finally finding the optimal solution of the optimization problem (Li 2019).

4.2 Literature Survey

4.2.1 Popular techniques of TO

Distributing materials in a domain to optimize performance is a significant topic in many fields, such as electromagnetics, solid mechanics, heat transfer, acoustics, fluid mechanics, materials design, and various other multiphysics disciplines, as mentioned earlier. There are several TO methods discussed in the literature, such as density-based methods, homogenization, discrete methods or the ON/OFF approach, and boundary variation methods. These have been applied to the design of Electromagnetic (EM) devices with varying degrees of success and popularity. A brief description of some of the popular TO methods discussed in the literature, such as homogenization methods, density-based methods, boundary-based methods, and discrete methods is included in this section.

4.2.2 Homogenization method

Bendsoe and Kikuchi (Bendsøe and Kikuchi 1988) proposed a method based on the homogenization principle in the late 1980s. The method has received a lot of attention in the field of structural engineering as represented by Bendsøe, Díaz, and Kikuchi 1993 and Allaire 1997. This method was first applied in the field of EM in (Yoo, Kikuchi, and Volakis 2000).

In this methodology, the optimal shape of the structure is transformed into an optimization problem for ideal material distribution. This method uses three design variables (in 2D design space) to describe a microstructure, as shown in Figure 4.4. This results in a large number of variables.

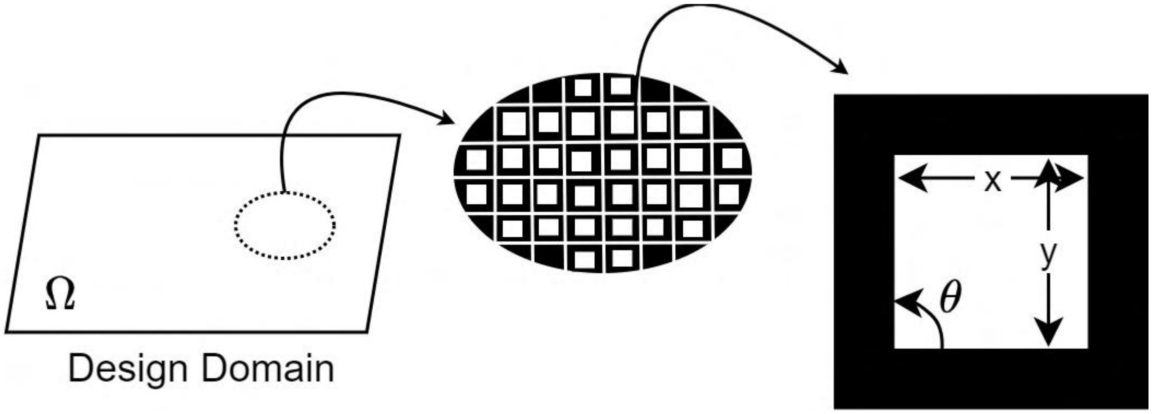


Figure 4.4: Schematic representation of domain discretization used in the homogenization along with a 2-D Unit microstructure with cavity.

In homogenization methods, the optimal shape of a structure is transformed into the optimal material distribution. The homogenization method is based on the concept of relaxation: making ill-posed problems well-posed by enlarging the space of admissible shapes (Allaire et al. 2019). However, it is not suited for large systems due to a large number of variables and adopts a non-smooth estimate of the topology boundary (Midha 2018).

In recent times, the homogenization method has received renewed attention, due to progress in the field of additive manufacturing. The ease of manufacturing microstructure materials and homogenization being the ideal technique to deal with microstructure leads to an ideal combination for TO with 3D printing (Guo and Leu 2013), as shown in Figure 4.5.

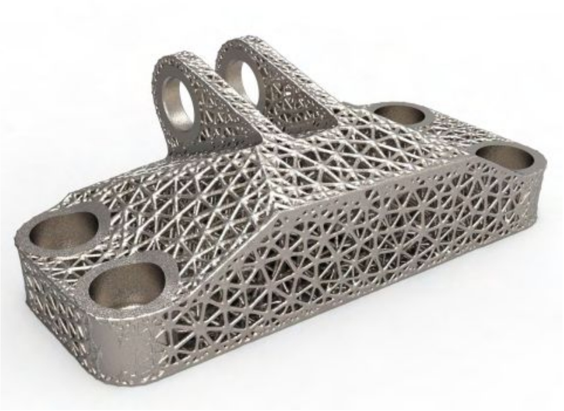


Figure 4.5: Example of a 3D printed lattice structure of a jet engine mounting bracket (Morgan et al. 2014; Allaire et al. 2019).

4.2.3 Density method for TO

Given a fixed domain of finite elements, density based methods optimize the objective function by determining whether each element should be either solid material or void. Thus it poses an extremely large-scale combination optimization problem. By adopting interpolation functions where the material properties are explicitly interpreted as the continuous design variables (usually the density of materials), the discrete variables are transferred to continuous variables and an optimizer is used to iteratively steer the solution towards a discrete solid/void topology. Usually, penalty methods are utilized to force the solution to a crisp "0/1" or "solid/void" topologies. Further, regularization and filter techniques are adopted to alleviate the checkerboard problem and mesh-dependency issue. The typical density based methods, namely the Solid Isotropic Microstructure with Penalization (SIMP) method, was proposed by (Bendsøe 1989). Later (Rozvany and Zhou 1991) studied and further developed the SIMP method. After it gained popularity in structure optimizations, literature of topology optimization in EM domain based on density-based methods have sprouted in electromagnetics (Yoo and Soh 2005; Byun, Park, and Hahn 2002; Wang, Park, and Kang 2004; Okamoto, Akiyama, and Takahashi 2006).

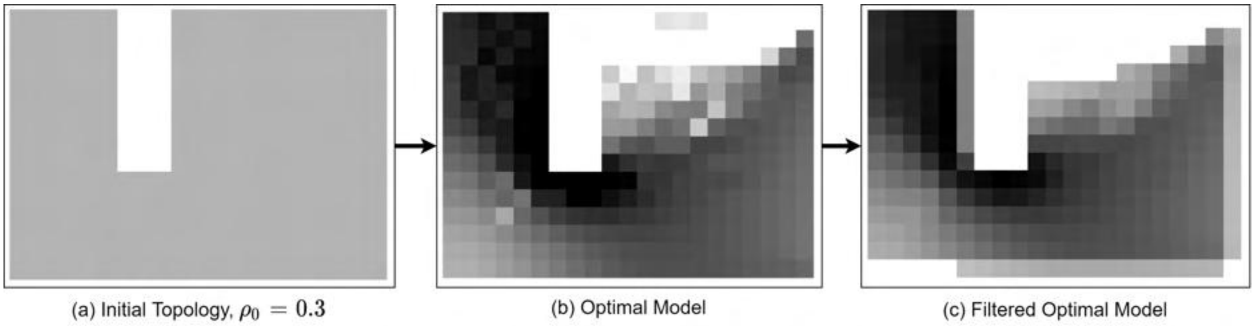

(a) Initial Topology, $\rho_0 = 0.3$
(b) Optimal Model
(c) Filtered Optimal Model

Figure 4.6: (a) Initial Topology, (b) Optimal design from density method, (c) Result with Average filtering (Midha et al. 2019).

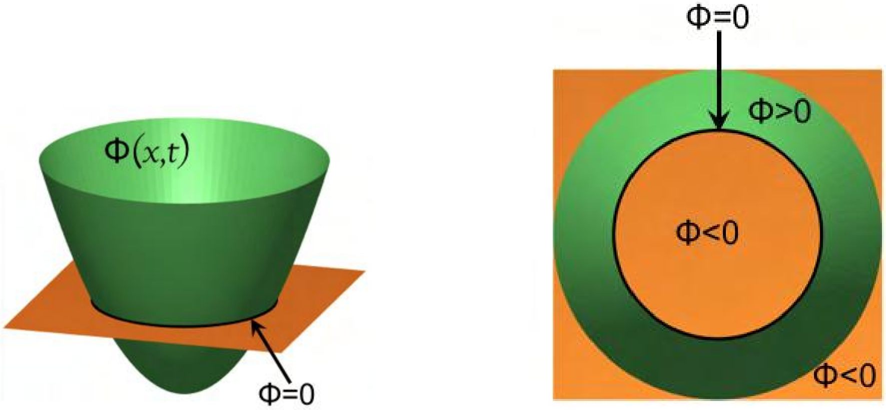
Though the density-based method takes the dominant position in the topology optimization based on its versatility, effectiveness and ease of implementation (compared to the homogenization approach), it also has a few distinct disadvantages. In the mechanical domain, Sigmund and Petersson (Sigmund and Petersson 1998) discussed the difficulties of the SIMP method such as local optimal, mesh-dependent structures and checkerboard patterns. Byun (Byun, Lee, and Park 2004) and Okamoto (Okamoto and Takahashi 2006) observed that the optimized results depend on the initial conditions. Besides, one typical problem with the density-based method is that of intermediate density. This is also referred to as grayscales material, regions of intermediate density that are allowed to exist in the optimal configurations. Although the penalization (regularization) scheme and filtering techniques

(Midha et al. 2019) will eliminate grayscales in a fraction of engineering topology design problems, such filtering schemes crucially depend on artificial parameters that lack rational guidelines for determining appropriate *a priori* parameter values.

4.2.4 Boundary based TO

Apart from homogenization and density-based methods, boundary based methods are a widely adopted approach to topology optimization, especially in recent decades. The methodology was developed by Osher and Sethian 1988. The root of boundary variation methods lies in shape optimization techniques. Different from the density-based methods in which design domain is parameterized in an explicit function, the boundary based methods are based on an implicit function ($\Phi(x,t) = c$) that defines the structural boundary (Deaton and Grandhi 2014). A new dimension is introduced by representing the 2D curve by an intersection between the plane and the surface as shown in Figure 4.7. The circular curve ($\Phi(x,t) = 0$) represents the boundary, that is tracked and is specified as the zero level set of the function Φ. The material distribution in the design domain is determined by the values of the level set function (Φ) in the domain, such as

$$\text{Material} = \begin{cases} 0, & \text{if } \Phi < 0 \\ 1, & \text{if } \Phi \geq 0 \end{cases} \tag{4.1}$$



(a) Intersection of a plane and and a surface

(b) Top View

Figure 4.7: Represention of a curve by a level set.

Boundary variations methods are mainly employed using level-set and phase-field methods. In the level-set method, the obtained boundaries are still represented by a discretized and unsmooth mesh in the analysis domain unless alternative techniques are applied. Another difference between density based methods and boundary based methods lies in the product of optimization. In density based methods, the optimized topology usually contain a lot of intermediate density elements where a post-processing procedure for interpreting topology is needed; however, the latter methods could generate the optimized topology with crisp and smooth edges. Boundary based methods such as level-set method, also generates an irregular and toothed optimized topology and sometimes the resulting solutions are heavily dependent on the initial state.

4.2.5 Discrete Methods

The ON/OFF or binary discrete methods constitute the discrete approach to TO, where each element in the design space is described using binary digits. In this way, each element can either assume a particular material or remain void (or a background material such as air) (Midha 2018) and thus gray cells/materials are absent in discrete TO methods. In this regard, density-based methods can be seen as translating the discrete problem of material distribution to a continuous form, thus enabling gradient-based optimization algorithms. Although the gradient-based approach is usually faster than stochastic optimization based discrete methods, stochastic methods, such as Evolutionary Algorithms (EA) (used in solving a discrete material distribution problem) offer better optimization capability at finding the global solution, stronger robustness and parallel searching capability, as mentioned in Section 4.1.3. This is a distinct advantage over gradient-based methods, even though it comes at a cost of high computation demand by frequent objective evaluation calls. A TO design for an electric machine can easily involve more than a 10,000 FE solution in the process of generating the optimal design. The time frame with a 3D FE solver can easily extend to a week. This is a huge computation burden for most designers.

Another drawback of discrete methods is that the optimal solutions contain floating pieces and checkerboard patterns. To overcome this difficulty, either different filtering or smoothing methods are employed (Campelo, Watanabe, and Igarashi 2008b) or smoothness of the solution is added as an objective of the optimization process itself (Dupré et al. 2014). Various filtering clusters and clan or smoothing techniques are employed to tackle this problem (Campelo, Ramirez, and Igarashi 2010; Campelo, Watanabe, and Igarashi 2008a; Sato, Watanabe, and Igarashi 2014). However, the filtering technique is not reliable and can often lead to an infeasible solution, which can drastically decrease the performance of the optimal design. Further, the filtering technique requires tuning several parameters such as the filter dimensions, thresholds associated with filtering, and the choice of different operating characteristics. Even after tuning, the absence of a single set of parameters of a filtering technique that can give optimal results for all TO problems is missing from the literature.

Even with these shortcomings, the ON/OFF method is popular in the literature due to the convenient implementation of different TO problems, along with easy integration with commercial FEA software. With the help of evolutionary algorithms, it is well designed to exploit global solutions.

4.2.6 Improvement to ON/OFF TO methodology

Considering the above-mentioned shortcomings of ON/OFF based TO, several improvements are proposed in this chapter. The first novel contribution of this chapter is a new TO method which ensures that the final optimized solution does not have any checkerboard patterns, perforations or floating pieces of material. Such designs can be difficult to manufacture using conventional subtractive manufacturing methods. Given that AM is quite flexible in terms of what it can manufacture, it is still necessary to check for manufacturability prior to finalizing the design. The proposed TO methodology leverages a sequence-based environment to impose connectivity on cells containing the same material. This new TO method (sequence-based) neither requires any filtering or smoothing technique nor any modification to the optimization objective function for obtaining manufacturable optimal solutions. Furthermore, this method facilitates the application of heuristic tree search-based algorithms such as Reinforcement Learning, which is another novel proposal of this chapter (discussed in Section 4.5.3). A few test problems proving the validity of this method are also presented, specifically a C-core actuator in Section 4.8 & the rotor of a Synchronous Reluctance Motor in Section 4.6. Further, the TO methodology discussed in this chapter will be extended to Chapter 5 to tackle the issue of generalizing to previously unseen design problems. The ability to generalize will significantly decrease the computation burden associated with an evolutionary algorithm-based discrete TO.

4.3 Sequence based controller

In this proposed sequence based TO method, a group of cells of the discretized design space serves as a pointer which can move about in the design space (Figure 4.8), one step at a time and it is referred to as "the controller". The controller is characterized by its size and the material associated with it. The movement of the controller is decided by choosing from a set of available actions (such as left, right, up or down in the global frame of reference) and the controller leaves behind a trail of material as it moves in the design space. So, the goal is to search for the sequence of actions for the controller such that its movements result in an optimized material distribution in the design space. Figure 4.8 shows the controller moving in a blank grid-based structure. A controller of size 3 fills an area of $3 \times 3 = 9$ cells in the design space. This can be used to modify the problem complexity as a larger size reduces the computation burden of the design optimization problem.

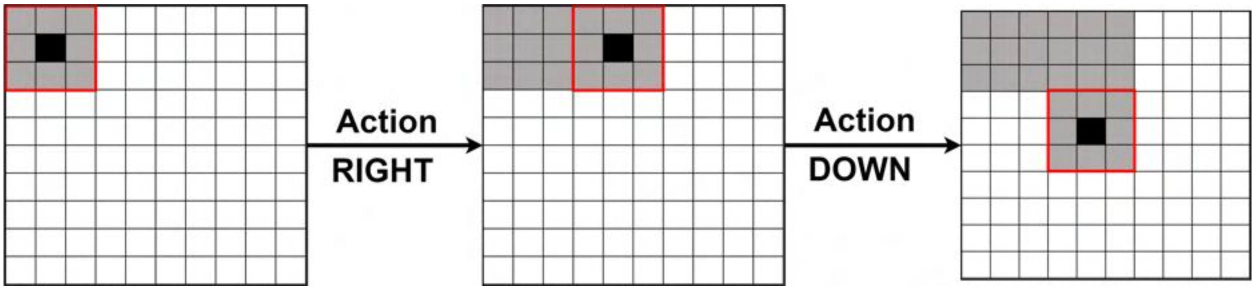


Figure 4.8: The movement of a Controller (size 3 × 3) in a design domain.

To distinguish the current state representation with any changes made in the future, versioning is maintained such that the current formulation of Sequence based TO methodology is named as SeqTO-*v1*. A future modification named SeqTO-*v2*, will be introduced in Chapter 5 Section 5.4.

4.3.1 The Operation of the Controller

The controller can move in the design space freely as long as it does not encounter the boundary of the design space. This boundary can be with the excitation source, air or due to symmetry. Further, the controller can be of different size, which will affect the amount of material it can place at a time. The current position of the controller is characterized by a pointer. This is to distinguish the controller from the rest of the material placed in the trail. In the case where the controller is greater than a single element of the grid, the centermost point of the controller is the pointer as shown in Figure 4.8.

4.4 Topology Optimization with SeqTO-*v1*

4.4.1 Genetic Algorithm

For the ON/OFF method and SeqTO-*v1*, the design parameters or optimization variables under consideration are discrete, and an optimization technique based on evolutionary algorithms, such as a Genetic Algorithm (GA) works well in such circumstances.

GA is one of the most popular metaheuristic approaches for binary optimization problems, and is inspired by the process of natural selection. In GA, a set of solutions are repeatedly modified using the operations such as 'selection', where superior solutions at the current generation are selected for future generations, 'crossover', where the selected solutions are combined to evolve into child solutions sharing the characteristics of the parents, and 'mutation', where the selected solutions randomly change their values with low probability. The GA algorithm used in this study is based on the work in Midha 2018 and the readers are referred to this material for in-depth explanation. MATLAB's "Global Optimization" (MATLAB 2018; The MathWorks, Inc. 2017) is chosen to implement a GA to test the applicability of the proposed environment.

C-core actuator

An electromagnetic actuator is a widely popular device with applications across different industries (Sigmund 2001; Choi and Yoo 2009; Wang and Kang 2002). A 'C' shaped electromagnet consisting of a coil, a magnetic core, and an armature constitutes a C-core actuator, as can be seen in Figure 4.9 In this structure, the current-carrying coil will produce a magnetic field according to Ampere's circuit law (Maxwell 1861). The main purpose of the core is to amplify and concentrate the magnetic flux in a well-defined guided path. In this study, a soft magnetic material (M-19) is used as the ferromagnetic material to form the core. The aim is to guide and maximize the magnetic flux flowing through the magnetic core into the armature. The higher the flux connecting the core and armature, the greater will be the pull/force experienced by the armature. The maximization of the flux through the core is a fundamental issue in the design of many electromagnetic devices including actuators, air compressors, electric motors, and generators. As such, the designing of an actuator is a well-studied topic in the field of EM design (Bíró et al. 2011; Lim et al. 2011; Lee and Wang 2012; Park and Yoo 2012; Okamoto et al. 2015). Some of the work focusing on the design of a C-core is used as a benchmark in this study (Park et al. 2008; Midha et al. 2019).

Applying the Genetic Algorithm (GA) with both ON/OFF and SeqTO-*v1* methodology, to optimize the core of an electromagnetic (EM) actuator so as to maximize the force experienced by the armature. This is done by optimizing the sequence of movements of a controller that consequently results in an optimized material distribution for the core of an EM actuator. The dimensions of the C-core actuator are chosen based on the example considered in Park and Min 2009a and Midha et al. 2019, as shown in Figure 4.9. The optimization problem is formulated in Eqn. 4.2 as

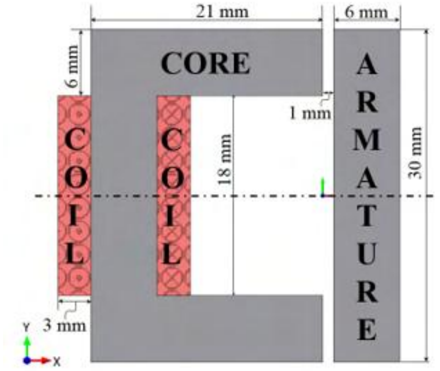


Figure 4.9: The C - core EM actuator

$$
\begin{aligned}
&\underset{\substack{a_i=0,1,2,3\\ i=0,1,2,\ldots,m}}{minimize} \quad f(a)\\
&\text{subject to} \quad K1 \leq \sum_{j=1}^{n}(p_j=1) \leq K2
\end{aligned}
\tag{4.2}
$$

where, $f(a)$ is the negative magnitude of force experienced by the armature (the negative is imposed to formulate a minimization objective), a is a sequence of actions that governs the controller's movements in the design space. The material distribution in the design space is represented by a vector p, which is populated by the action vector a. The vector p contains the material property of each cell in the discretized design domain, m is the pre-defined sequence length (length of vector a) and is a representative of the freedom given to the controller to explore the environment, n is the length of the material distribution vector p, and K1 & K2 is a limit on the minimum and maximum number of switched ($p_j : 0 \rightarrow 1$) cells permitted in the design space.

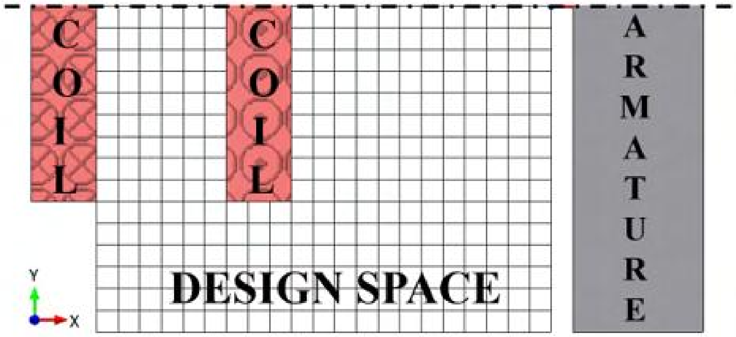


Figure 4.10: C - core discretized design space

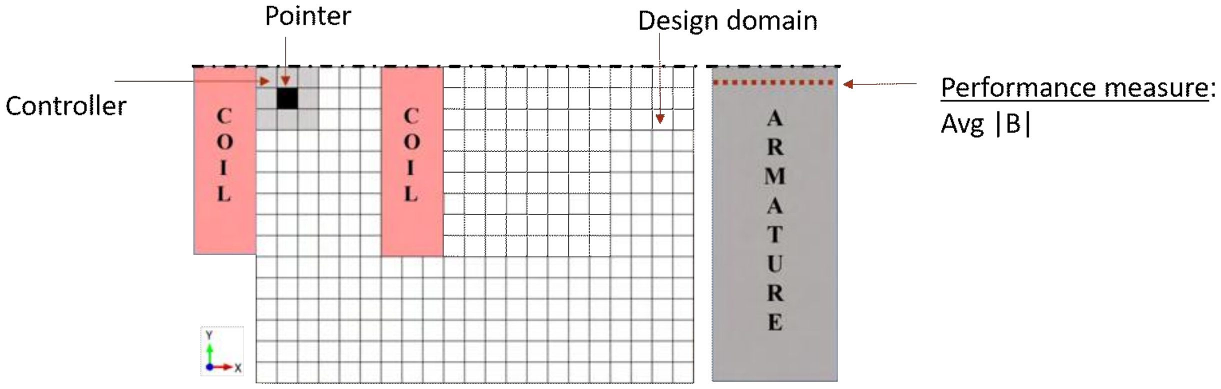


Figure 4.11: C-core controller

The depth of both the armature and the core is fixed to 40 mm (the same as that in (Midha et al. 2019)). The 2-D design space is discretized into 288 square cells of size 1 mm and symmetry about the x-axis is utilized to reduce the size of the design space as shown in Figure 4.10. The material in these cells is permitted to switch between air ($p_i = 0$) and electrical steel ($p_i = 1$), depending on the movements of the controller.

A conventional half-symmetric C-core shape consists of 180 electrical steel cells in the discretized design domain. $K2$ is set to 180 and serves to enforce the volume constraint on the electrical steel. $K1$ is set to 80 to ensure that the controller explores the design space. The value of m (length of a) is an important hyperparameter which must be decided beforehand and it also depends on the selected controller size. It is equal to the maximum number of steps (actions) that the controller is allowed to take when it moves in the design space. This value should be such that the controller is able to take enough steps to fill in at least the allowed number ($K2$) of cells with material. Usually, m is chosen such that the controller is able to fill a little more than $K2$ number of cells in the design space. This is done to account for the steps taken by the controller which do not result in any additional material in the design space. Initially, air ($p_i = 0$) is assigned to all the cells in the design space. A starting location for the controller is specified beforehand and it remains constant throughout the optimization process. This method also allows for the selection of different starting positions of the controller in the design space.

Some of the GA's parameters are directly dependent on the TO controller configuration. The number of design variables for GA is set equal to m. The population size is selected to be a multiple of the number of design variables.

4.4.2 Tree Search Algorithms

Apart from the advantage of generated designs which are free from checkerboard patterns and ensuring connectivity of material in the design domain, sequence-based TO also provides the capability to use algorithms which make decisions as a series of sequences such as tree search algorithms like Temporal difference (TD) learning, Monte Carlo (MC) algorithms and others that are related to Dynamic Programming (DP) (Sutton 1988).

The main advantage of exploring these algorithms is the stark difference they pose in relation to evolutionary algorithms. Table 4.1 lists a comparison of Evolutionary Algorithms and sequence based decision making such as TD-learning. It should be noted that these sequence-based decision making algorithms come under the wide topic of Reinforcement Learning (RL). Reinforcement learning is characterized by the fact that the learning system's actions (movement of the controller) influence its later inputs (states). The learning system does not have direct instructions on the optimal actions to take, and no knowledge of the consequences of actions, including reward signals, and how they play out over extended period of time (Sutton and Barto 2018). Since this work only deploys a TD-learning based algorithm, the general algorithms in RL will not be discussed here.

Reinforcement Learning	**Evolutionary Algorithms**
One agent trying to learn from both positive and negative actions	Starts with many agents, but only strong ones survive
Learns from both positive and negative rewarding actions	Only learns the optimal solutions. Negative or sub-optimal solution information is discarded and lost
Good for problems that can be represented as a Markov Decision Process (MDP) (Section 4.5.1)	Good for problems where loss function is non-differentiable.

Table 4.1: Comparison between Evolutionary Algorithms & Reinforcement Learning

4.5 TD Learning

4.5.1 Markov Decision Process

A Markov Decision Process (MDP) is a discrete-time process that converts a given 'task' to a sequential problem. The key entities of an MDP are the environment, state, action, and reward. The controller interacts with the environment in discrete steps. At each step (denoted by 't' in subscript) the controller receives a 'state' from the environment. The 'state' contains sufficient information regarding the current material distribution in the design domain for the controller to interpret and make a rational decision (action). Based on the action that the agent took in a state (s_t), the environment provides a feedback in the form of a reward (r_t). Contrary to the word 'reward', the agent can also be penalized for an illegal or a bad move with a penalty (through a negative reward). The environment also provides the next transition state (s_{t+1}). Actions influence the reward as well as the future states and through that future rewards. The algorithm revolves around exploring the environment to learn the transition from one state (s_t) to another (s'_t), based on an action (a),($s \rightarrow s'$). This transition will result in a reward signal (r_t). This reward at each step is a scalar quantity $r_t \in \mathbb{R}$.

In an MDP, state transition and reward function are solely dependent on the current state and the chosen action and is independent of the previous states and actions. Some states are set as terminal states, and tasks reaching them are terminated.

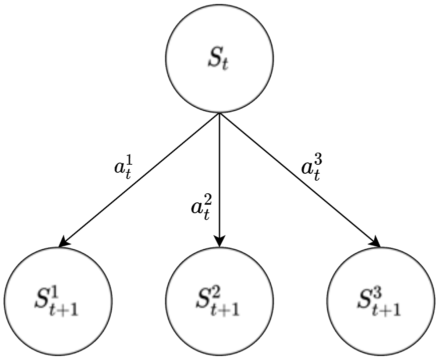


Figure 4.12: State Transition (s_t, a_t, s_{t+1})

Policy

The choice of action has an impact on both the immediate reward and the next state. Policies determine how an agent selects these actions. A 'policy' is a distribution over actions for all possible states. A deterministic policy maps each state to a single action from the action set, thus:

$$\pi(s) = a$$

where π is a deterministic policy and $\pi(s)$ represents the action selected in the state (s) by the policy (π). A deterministic policy here can be represented in a tabular format. The same action can be selected in multiple states, however, some actions may not be favorable and are not selected at all. The distribution of actions is independent of each other, for all possible states. On the other hand, a stochastic policy is one where multiple actions can be selected with a non-zero probability. The set of available actions in each state is the same but the probability distribution varies. In an ideal MDP, a state has all the information for decision making. This chapter deals with deterministic policies and stochastic policies will be discussed in Chapter 5.

Episodic Task

Another important criterion is to determine if the task is episodic or continuous. Since the controller has a limited amount of material (volume constraint) and we want the controller to limit the number of function (FE in this case) calls, the design optimization setup limits the length of an episode to be sufficient to move around and place the given volume of material. It should be checked that the agent has enough leeway in the number of steps in an episode, to be able to explore the vast design space. A guideline, in this case, would be to identify the number of steps needed to come up with the conventional geometry using the MDP formulation and allow the controller to spend double the number of steps in the design domain for each episode. This will give the agent a sufficient amount of steps to explore the design space.

Reward Formulation

The dynamics of the MDP have to be such that they result in the successful creation of an optimal geometry subject to a material (volume) constraint. The goal should be to translate an agent's behavior to maximize the reward. This means maximizing the immediate reward and the cumulative reward in the long run (such as an episode). The agent's learning has to balance the short term and long term goals. The reward function should be seen as a way to communicate with the agent as to what to achieve. In such a case, the obvious reward function will be the performance (objective) criteria that is being maximized/minimized. For a C-core, it will be the force experienced by the armature at any given step of the MDP. However, it was observed that sometimes the force experienced can be noisy and not accurate. This was attributed to circumstances when the design domain contains very little material (usually at the start of an episode), the force value will be small and it will become difficult to distinguish two sparsely populated (with material) design domains. Thus, it was decided that the magnetic field distribution values in the C-core armature (Figure 4.13) is a far more robust indicator of performance. Further, this can work as a substitute for force value as it is directly proportional to the magnetic field lines passing through the armature.

Maximizing the total return at time 't', is simply the sum of rewards the agent will receive

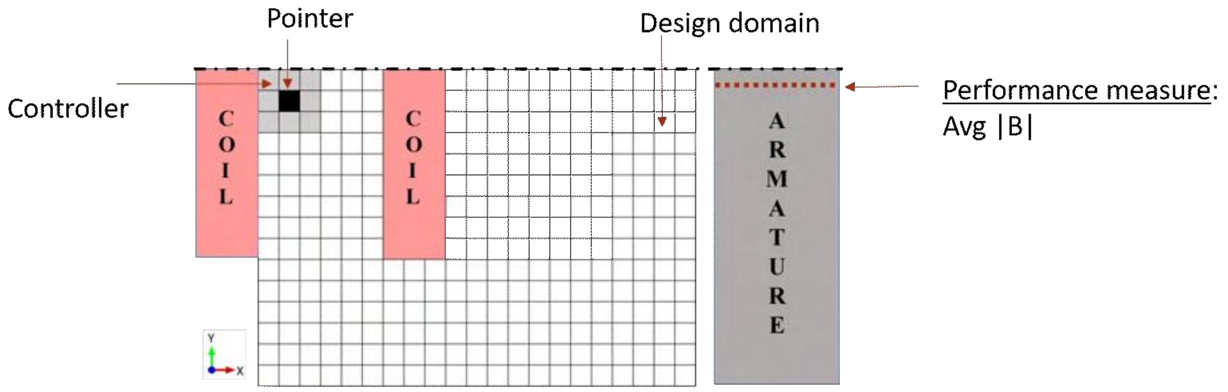


Figure 4.13: C-core controller

after time 't'.

$$G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_N$$

where G_t is called the return.

MDP Dynamics

An important property to remember is that the future states and reward only depend on the current state and the action. This is a Markov property that needs to be satisfied for a proper MDP. For enabling actions that depend on the agent's behavior (action taken) in the previous step, it is important to include this information in the state representation without violating the Markov property. This is achieved through keeping a trace of past distribution of the design domain and material distribution in the state representation. The Markov property means that the present state is sufficient and remembering earlier states would not improve predictions about the future. This is an important property as there can be multiple ways of representing the state for a discrete design domain for TO, as shown in Figure 4.14, 4.15 & 4.16. In this chapter only the state representation shown in Figure 4.14 will be used with the RL and GA algorithms. The other state spaces (Figure 4.15 & 4.16) will be used in Chapter 5 with advanced algorithms in RL.

It should be noted that the MDP formulation can be designed in many different ways. It is very abstract and flexible. Formulating an MDP for any task requires knowledge of system dynamics of the electromagnetic problem, agent training, and the reward formulation. Necessary changes have to be made for adapting this work to other tasks. A number of modifications in the current formulation of SeqTO-*v1* will be discussed in Section 5.4. Further, we have not delved into the process of accounting for the complexity of the problem. The

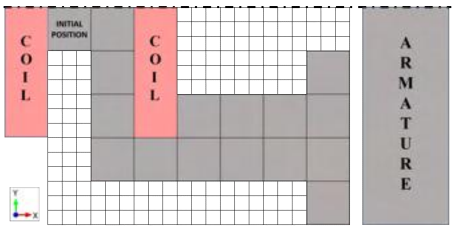


Figure 4.14: Discrete design domain as the state space

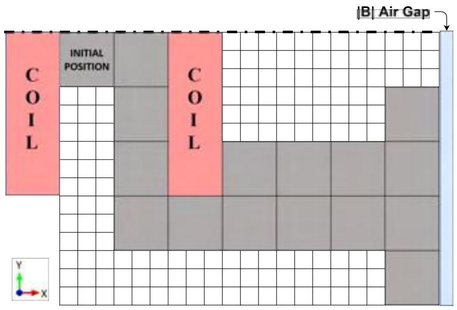


Figure 4.15: Discrete design domain with the $|B|_{airgap}$ as the state space

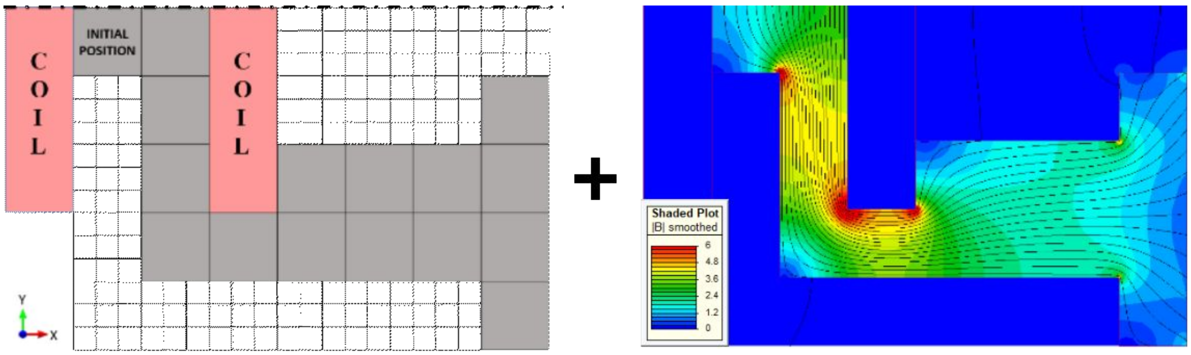


Figure 4.16: Discrete design domain along with the $|B|$ in the design space as the state space

set of states and actions are countable, the value will be very high and is ∞ for the reward set.

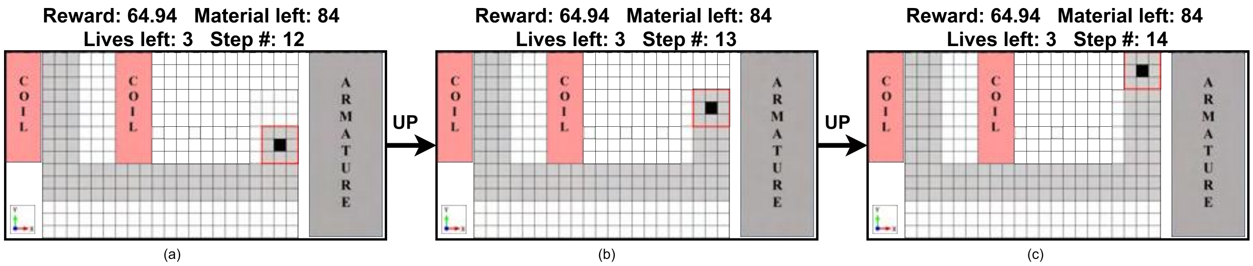


Figure 4.17: Scenario for a negative Reward (R_t)

4.5.2 Value functions

The reward function captures the notion of short term gain. However, the agent should achieve the most reward in all the subsequent state trajectories, i.e, in the long term. The Value function, either $Q(s, a)$ or $V(s)$, formalizes this. The value function can be represented in two forms:

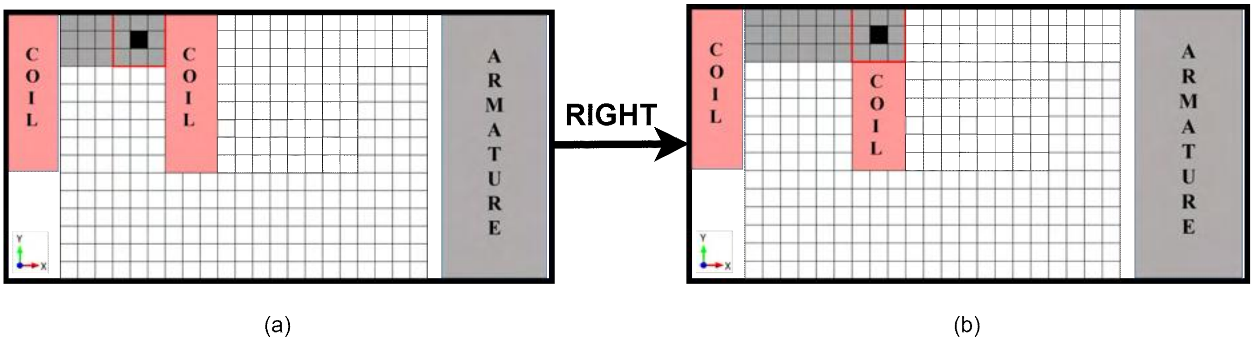


Figure 4.18: Possibility of a design error due to the controller moving out of the designated design space.

- **Action value function** $Q(s, a)$: A Q-Value ($Q(s, a)$) is a measure of all the rewards the agent shall receive till the end of the episode, starting from the state (s) and after performing an action (a). This cumulative reward function can be discounted based on the temporal position of the reward from the current step (t).
- **State value function** $V(s)$: On the other hand, the State value function ($V(s)$) is the expected value of all possible future rewards, on all possible actions.

It is preferable to use the Action value function ($Q(s, a)$) for this work. This preference is based on the knowledge that a transition from one state to another can be performed in more than one way. Further $Q(s, a)$ is more useful for comparing the different actions available in a state. The agent-environment interaction generates a trajectory of experience consisting of states, action, and rewards. TD-learning uses the value associated with each state-action pair to utilize all the experience (s, a, s', r) generated by the agent. The value of the state-action pair is the expected return (G_t) when that action is taken

$$Q(s, a) \doteq \mathbb{E}_\pi[G_t|s_t = s, a_t = a] \tag{4.3}$$

For an episodic task, the learning algorithm needs to wait until the end of the episode to work with the absolute/true value of the 'return' for every step in the corresponding episode, since the 'return' (G_t) depends on all the reward obtained from t - step onwards.

$$G_t \doteq R_{t+1} + R_{t+2} \dots R_n \tag{4.4}$$

This approach is the key to Monte Carlo (MC) based methods. In MC, there is no guarantee that the agent will visit all the possible states, along with the other issue of waiting for the end of an episode to update the value function ($Q(s, a)$) (Singh and Sutton 1996). However, there is another approach that uses bootstrapping, which is based on recursive use:

$$G_t \doteq R_{t+1} + G_{t+1} \tag{4.5}$$

In a similar fashion as Eqn. 4.5, the value function (Eqn. 4.3) can also be updated as:

$$\begin{aligned} Q(s_t, a_t) &\doteq \mathbb{E}_\pi[R_{t+1} + G_{t+1} | s_t = s, a_t = a] \\ Q_\pi(s) &= R_{t+1} + Q_\pi(s_{t+1}, a_{t+1}) \end{aligned} \tag{4.6}$$

Compared to MC-based methods that wait until the end of the episode to update all the corresponding value functions, bootstrapping-based methods (such as TD-learning) learn incrementally and throughout the episode. However, the disadvantage of the TD-learning approach is that the incremental updates can have higher variance compared to the true return based approach of MC methods. It has been shown practically that bootstrapping based methods converge faster but may be off the target, or not properly learn the true return values (Sutton and Barto 2018). For faster learning, we want to update towards the return but not wait for the episode to complete, so the empirical (true) value of the return at a time 't' is replaced with the reward and the estimate of the return from the next state. This way the updates to the value function ($Q(s_t, a_t)$, performed by the learning algorithms are faster.

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \Big[G_t - Q_\pi(s_t, a_t)\Big] \tag{4.7}$$

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \Big[R_{t+1} + \max_{a'} Q_\pi(s_{t+1}, a') - Q_\pi(s_t, a_t)\Big] \tag{4.8}$$

TD updates the value of one state ($Q(s_t, a_t)$) towards it's own estimate of the value in the next state ($Q(s_{t+1}, a_{t+1})$). As the estimated value of the next state improves, so does the target.

To overcome the disadvantage of higher variance in the return due to incremental updates in TD learning, the rewards are adjusted by weighing the rewards based on their importance over time. The farther in future a reward is received, the lesser is it's importance while calculating the return at time 't' (G_t). This is controlled using discount factor (γ). The discount factor helps reduce the degree to which future rewards affect the current value function estimates; thus stabilizing the learning. The relation in Eqn. 4.6 modifies to:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a') \tag{4.9}$$

TD-learning and MC-based methods have some similarities too. Both of them learn directly from the experience generated by the agent's interaction with the environment. TD shares some similarities with Dynamic Programming (DP) too. In fact, RL algorithms were developed from DP principles (Bellman 1957). Like DP, TD-learning bootstraps too. In a way, TD-learning combines the ideas from both DP and MC based learning.

4.5.3 Policy function

Intelligent behavior arises from the actions of an individual seeking to maximize the received reward signals in a complex world. The aim is to identify:

- Policy Evaluation: Use the reward signal to understand the behavior of agent-environment interaction.
- Control / Policy Improvement: Develop algorithms that search the space of behaviors to maximize reward signals.

Policy Evaluation is the task of determining the value function for a specific policy. Control on the other hand is the task of finding a policy to obtain as much reward as possible or in other words, finding a policy that maximizes the value function. Thus, the task of policy evaluation and improvement is an iterative process. In Section 4.5.2, it was shown that TD-learning can be used for estimating the value function. However, a value function is always defined with reference to a policy, denoted by π. Action value function (Q_π) describes what happens when the agent selects a particular action. Q_π indicates that the value function is contingent on the agent selecting actions according to the policy (π). The Action value of a state is the expected return if the agent selects an action 'a' and then follows a policy 'π'. If the policy changes, the learning algorithm should reflect the change by adjusting the value function accordingly. As part of policy improvement, the agent uses the information from the value function to make decisions. A policy can be seen as a probability distribution over actions for each state, denoted by $\pi(a|s)$. When a policy maps each state to a single action, it results in a deterministic policy, denoted as:

$$\pi(s) = a \tag{4.10}$$

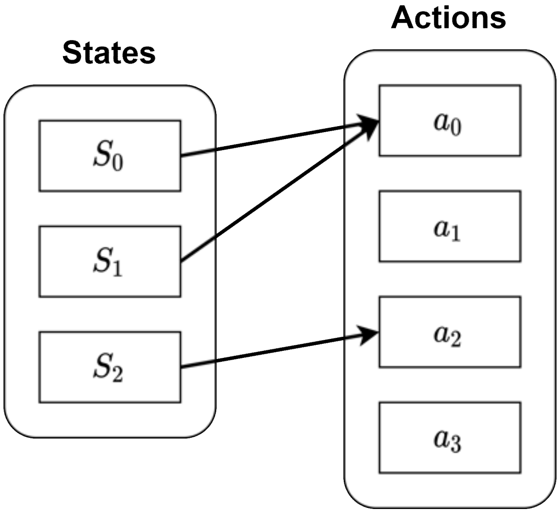


Figure 4.19: A deterministic policy behavior

State	Actions
s_0	a_0
s_1	a_0
s_2	a_2

Table 4.2: A deterministic policy behavior.

	a_0	a_1	a_2	a_3
S_0	8.2	-0.2	4.1	4.4
S_1	6.6	-1.0	-1.5	5.6
S_2	-3.3	-2.6	11.1	-4.7

Table 4.3: Example of a Q-table

Knowing the optimal $Q(s, a)$ function automatically gives us the optimal policy. Specifically, the best policy represents, at every state, choosing the optimal action:

$$\pi(s) = argmax_a \; Q(s, a) \tag{4.11}$$

which provides the agent the optimal action based on the policy starting in the state (s) and all the way until the end of the episode. The policy (π or sometimes $\pi(a|s)$), is a mapping from some state (s) to the probabilities of selecting each possible action given that state, as can be seen in Figure 4.19 and Table 4.3. For example, a greedy policy outputs for every state the action with the highest expected Q-Value.

Q-Learning

An ideal scenario would be that the value function is accurate in it's estimation of the true return from each state. However, a good value estimator needs to have good environment experience to learn from. To generate good experience trajectories, the agent needs to explore the environment and exploit the information it has learned. Even with a branching factor of four (as shown in Figure 4.20), the length of the tree can be massive both due to the length of the episode and the looping characteristics of the MDP. As such the agent needs to identify whether a tree node is good or bad and improve the value function estimate of only those nodes which are potential for optimal geometries. To satisfy both these requirements, we have to use Q-learning. Q-learning (QL) uses the information from the Value function along with a stochastic action to balance the dilemma of exploration and exploitation. This stochastic rule is called the ϵ-greedy approach.

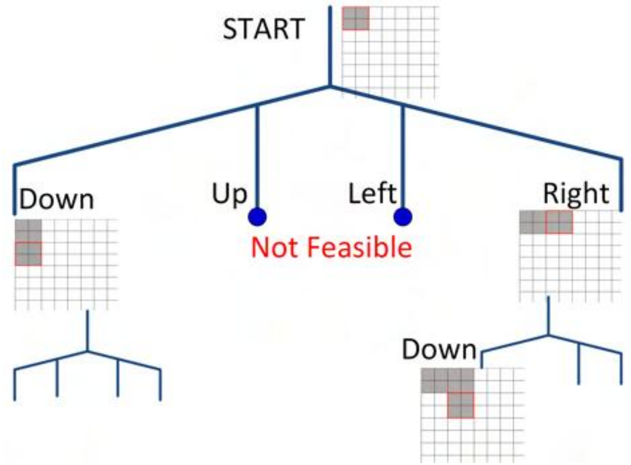


Figure 4.20: The tree diagram for the sequence based controller.

QL was developed in 1989 (Watkins and Dayan 1992). It is one of the major online learning algorithms. The equation shown below describes the TD-learning based update rule for an Action value function (Q-function)

It describes how the action values can be estimated incrementally

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \underbrace{\alpha}_{\text{lr}} \Big[\overbrace{R_{t+1} + \underbrace{\gamma}_{\text{df}} \underbrace{\underset{a'}{max}\, Q(s_{t+1}, a')}_{\text{off-policy}}}^{\text{TD-target}} - Q(s_t, a_t) \Big] \tag{4.12}$$

The error in the estimate is the difference between the old estimate and the new target. Taking a step towards the new target will create a new estimate that reduces the error. The size of the step is determined by the step-size parameter called the ‘learning rate’. The value of step size is in the range of 0 and 1.

ϵ-greedy exploration

ϵ-greedy is a method for balancing the exploration and exploitation performed by the QL-based controller. Exploration allows the agent to improve its knowledge about each action.

By improving the accuracy of the estimated action values, the agent can make more informed decisions in the future. Exploitation on the other hand exploits the agent's current estimated values ($Q(s, a)$). It chooses the greedy action to try to get the most reward. When the agent explores, it gets a more accurate estimate of its value function. On the other hand, when the agent exploits, it has a higher chance of getting more reward. It is not possible to choose both simultaneously. A simple method for choosing between exploration and exploitation is called ϵ-greedy, which is a random process. This approach is based on a parameter ϵ which refers to the agent's probability of choosing to explore over exploit.

$$A_t = \begin{cases} \arg\max_a Q_t(s, a) & \text{with probability } 1 - \epsilon \\ a \sim (a_1, \ldots a_k) & \text{with probability } \epsilon \end{cases} \tag{4.13}$$

The action that is selected at time-step 't' is the greedy action with probability $1 - \epsilon$ or is a random action with probability ϵ. The value of *epsilon* varies from 0 to 1. We usually start with a high ϵ value, close to 1. The higher the value of ϵ, the more exploration the agent will perform as compared to exploitation. This can be justified during the initial stage of learning as the agent is new to the environment and cannot rely on a randomly initialized value function for exploitation. It needs to collect experience to train the value function and as the value function starts gaining capability, we reduce the exploration by decreasing the value of ϵ. By the end of the training, ϵ decreases and goes down to a very low value of say 0.05. This makes sure that at any given point the agent will have some component of exploration. Once the agent converges to a policy that is not changing, the training stops. The starting value, decay rate, and stopping value of the ϵ are treated as hyperparameters in this study. In MDPs, there is always an optimal deterministic policy. In other words, you can always find a deterministic policy that is better than any other policy.

Exploration - exploitation trade-off

Pure greedy action selection can lead to sub-optimal behavior. Since the greedy action is always the same, the agent misses out on learning from the samples of alternative state action pairs. This induces the exploration-exploitation dilemma. It is important that the agent gets the chance to explore most (if not all) of the possible state-action pair that are there in the environment. However, this is an issue if the state space is of very high dimensionality. Further, for an MDP representation which involves the field value in the state representation (Figure 4.15 & 4.16), a tabular method will fail. For a high dimensionality state-space or cases involving continuous values like the magnetic field distribution, one needs a function approximator that can handle this, which will be discussed in Chapter 5.

4.6 C-core - Topology Optimization Results

4.6.1 GA with SeqTO-*v1*

Different environments are set up based on the controller size (1 and 3), the type of material (linear: pure iron (linear B-H characteristics) and nonlinear: electrical steel (M-19 26 Ga)), the number of variables (m) and the population size (in multiples of m). The coil current is fixed at $1A$ and the number of coil turns is set to 500. For this combination of coil turns and current, the force experienced by the armature of a conventional C-core EM actuator is 40 N, as shown in Figure 4.21, while the force (on the armature) obtained with an ON/OFF based TO algorithm in (Midha et al. 2019; Park and Min 2009a) employing electrical steel is 50.90 N. This armature force decreases to 50.60 N on application of post-processing filters to enforce topology smoothness (Midha et al. 2019).

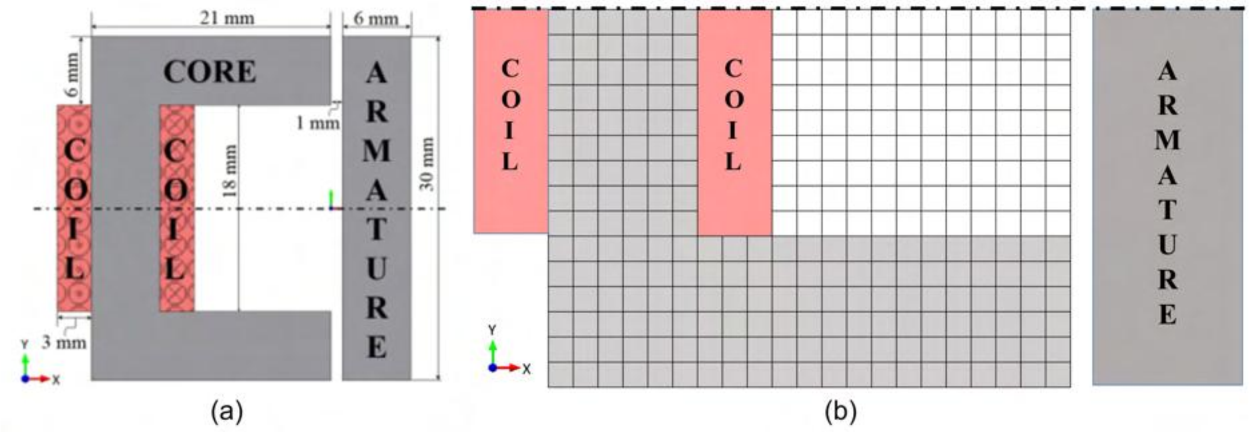


Figure 4.21: (a) Conventional C-core design (b) Only half of the geometry is simulated due to symmetricity.

Environment Type	Design Variables (m)	Pop. Size	Stall Gen.	Force GA SeqTO-*v1*	Force GA On/Off (Midha et al.)
3 × 3 (Linear)	20	20 ||40||60	25	70 N	-
3 × 3 (Linear)	25	25 ||50||75	25	70 N	-
3 × 3 (Non-Linear)	30	20 ||40||60	25	51 N	-
1 × 1 (Linear)	200	200	100	71 N	70 N
1 × 1 (Non-Linear)	200	200	100	55 N	50.6 N

Table 4.4: C-core results obtained from Genetic Algorithm using the SeqTO-*v1* TO environment, along with a comparison of armature force wrt GA optimized results + filters (Midha et al. 2019).

Table 4.4 lists all the different case studies with the dimensions (1×1 & 3×3) referred to the size of the controller, along with the type of material used in the design space. The dimensionality of the controller is reflected in deciding the value of design variables (m) for GA (MATLAB 2018) (The MathWorks, Inc. 2017). A limit on the number of stalled generations is employed to test for convergence. The initial population of the GA is based on a random seeding.

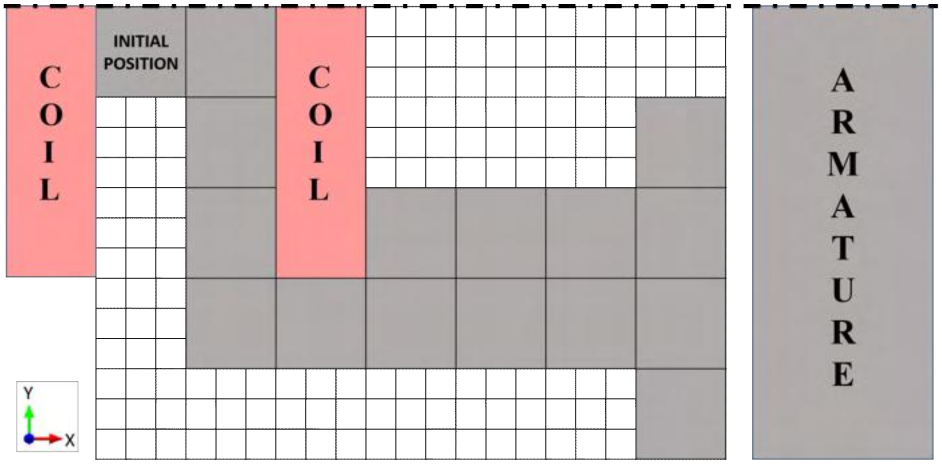


Figure 4.22: Optimized geometry for 3×3 Linear environment (SeqTO-*v1* with GA).

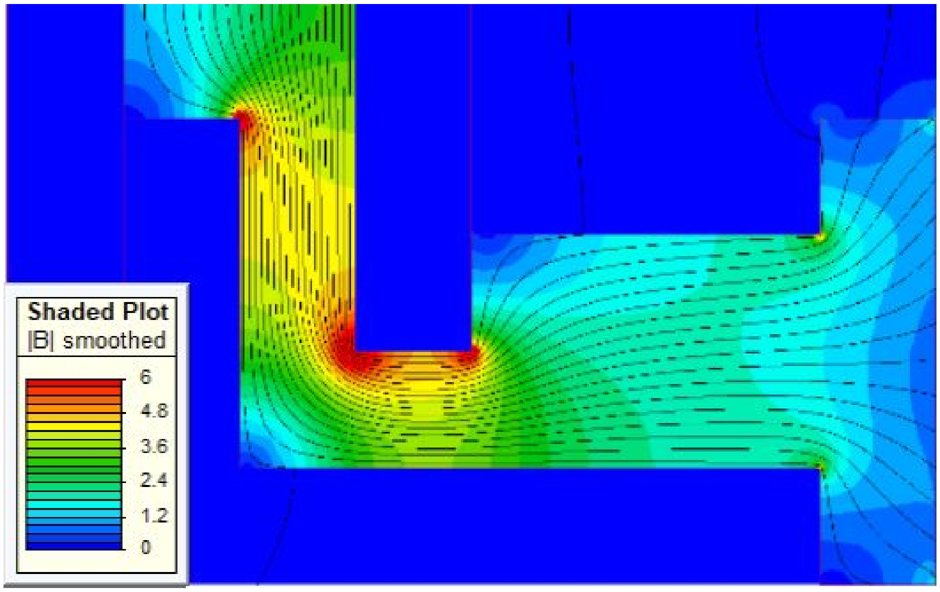


Figure 4.23: Optimized geometry's magnetic field distribution for 3×3 Linear environment (SeqTO-*v1* with GA).

Using the setup discussed in Section 4.4.1, GA is employed on SeqTO-*v1* environment for all the case studies and the optimal force values are listed in Table 4.4. It is observed

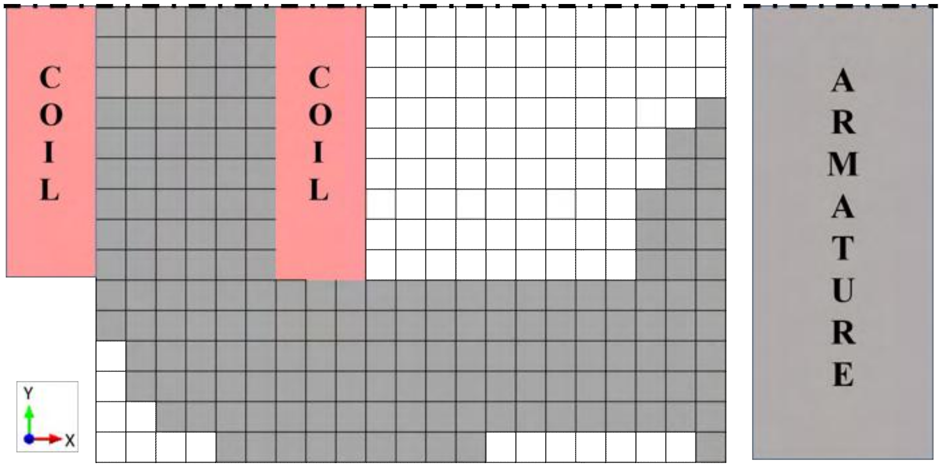


Figure 4.24: Optimized geometry for 1 × 1 Non-linear environment (SeqTO-*v1* with GA).

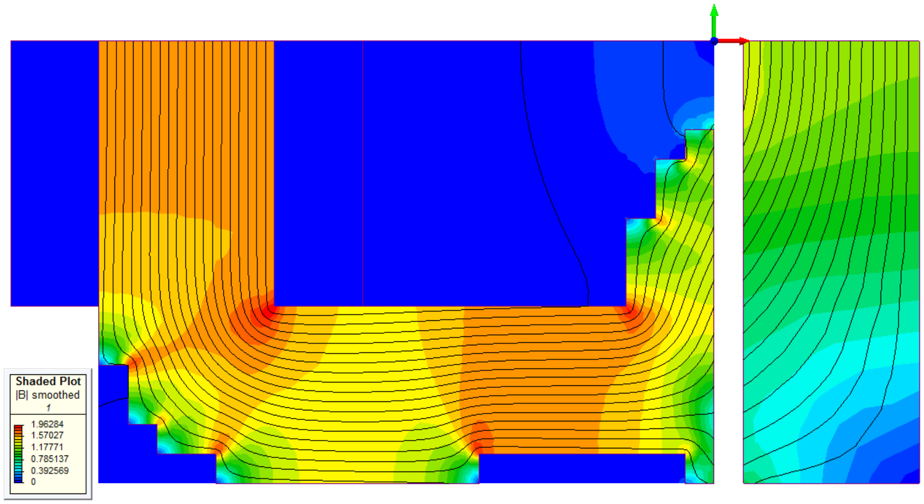


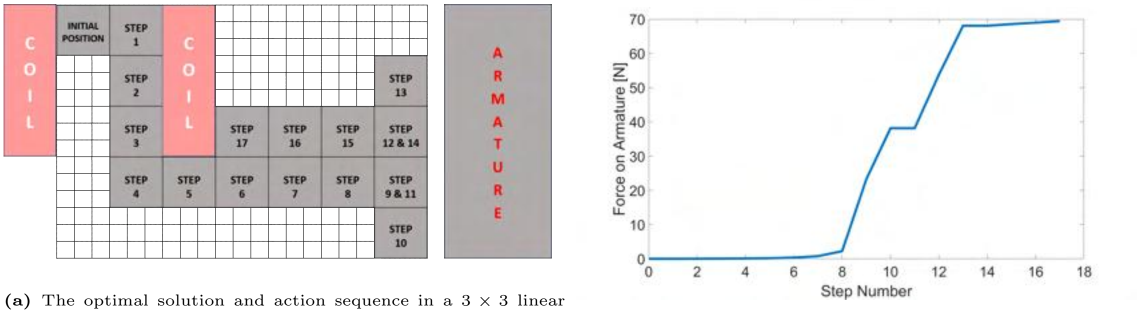
Figure 4.25: Optimized geometry's magnetic field distribution for 1 × 1 Non-linear environment (SeqTO-*v1* with GA).

that the bigger the size of the controller, the fewer the steps required to fill the design space. However, this affects the granularity of the optimal structure obtained, as can be seen through a comparison between a controller of size 3 × 3 (Figure 4.22) and 1 × 1 (Figure 4.24) when the material of the controller is electrical steel. With the finer granularity controller (1 × 1), GA managed to achieve a higher performance value for the optimal design. The optimal design obtained using a 1 × 1 controller on SeqTO-*v1* environment produced a force of 55 N, an increase of about 8% as compared to using a controller of size 3 × 3 (51 N).

Finally, as it is evident from Table 4.4, this new sequence based TO method performs on par with Midha et al. 2019 (for the linear case: 70 N) and achieves a good increase in force when compared for the non-linear cases (8.7% improvement).

4.6.2 C-core: Results of TD-learning

Based on the design setup in Section 4.4.2, a linear material based controller of size three is tested on the SeqTO-*v1*. The optimal geometry for a 3×3 SeqTO-*v1* environment with linear material is shown in Figure 4.26a. This result matches the geometry obtained through the GA algorithm (Figure 4.22). In addition to the optimal geometry, we can get a sequence of actions that are greedy in nature, as proven by the monotonic increase in reward function in the graph (Figure 4.26b) between the force acting on the armature and the step-index. This gives a greedy action that tries to continuously increase the value of the force both for the near future and for the final optimal geometry. The agent needs to account for the long-term impact of its actions. As such the agent creates a channel to feed magnetic flux lines in the armature. This leads to an increase in the magnitude of magnetic flux in the armature, which in turn increases the force value on the armature. However, this results in magnetic field saturation in the iron placed earlier by the agent. This poses a conundrum for the agent as to whether it should place material near the armature to increase the surface area near the airgap and feed more flux or go back and take care of the magnetic saturation in the core of the design. Such information can be helpful in analyzing the importance of each element in the whole topology. Further, a designer can use this knowledge to understand the trade-off of any element with the performance and can further fine-tune the structure without running an optimization algorithm.



(a) The optimal solution and action sequence in a 3×3 linear environment

(b) Force on armature vs steps in an episode

Figure 4.26: Optimal solution 3×3 controller (SeqTO-*v1*) with TD-learning, on a C-core design domain.

4.7 Synchronous Reluctance Motor

Extending the SeqTO-*v1* to more complex electric machines, the rotor of a Synchronous Reluctance Motor (SynRM) is optimized using the proposed TO methodology.

The industrial applications of SynRMs have risen in the past few decades, due to their simple structure and low costs. The SynRM offers a number of advantages. Some of them are listed here:

- High efficiency
- High power density
- Less maintenance
- Reduced inertia
- Very Reliable

Compared to Induction Motors (IMs), which have been the industry standard for many decades, the SynRM can offer high power density for a smaller footprint. This can be almost 2 to 4 times higher than an IM, with a frame size almost 1-2 times smaller (Gyllensten et al. 2016). The SynRM is also one of the most efficient motors available to the industry, with a recent development even reporting an efficiency value as high as 99.05% (full load on a 44 MW, 6 pole, synchronous motor) (*ABB motor sets world record in energy efficiency*). Such highly efficient motors are the future, especially when motors employed by the industries consume about 28-30% of the world's electricity (Kober et al. 2020; *World Energy Outlook 2019 – Analysis - IEA*). Further, a SynRM also operates at a lower temperature as compared to an IM, helping with a longer operational life of insulation, bearing, etc. Internal bearing failure is a major concern with an IM, being responsible for almost 40 % of IM failures (IEEE 1985, IEEE Survey). In addition, with the absence of slip rings, rotor field windings. permanent magnets, commutators, and brushes, SynRMs have a low manufacturing cost, require less maintenance, and offer longer operation life. Some of their practical applications include usage in washing machines, analog electric meters, hard drives disks and electric vehicles.

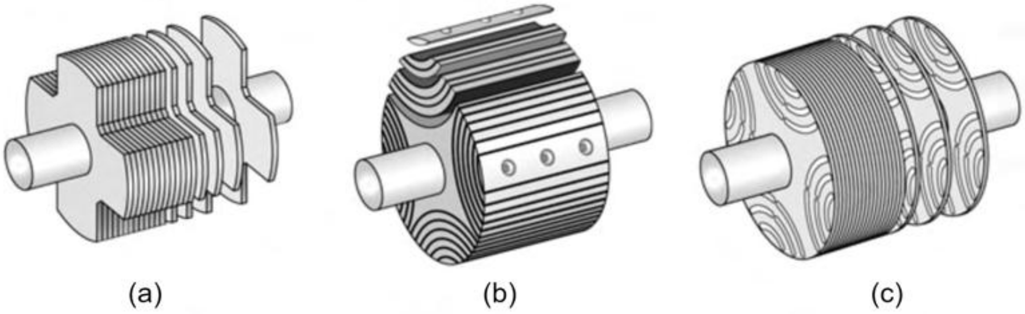
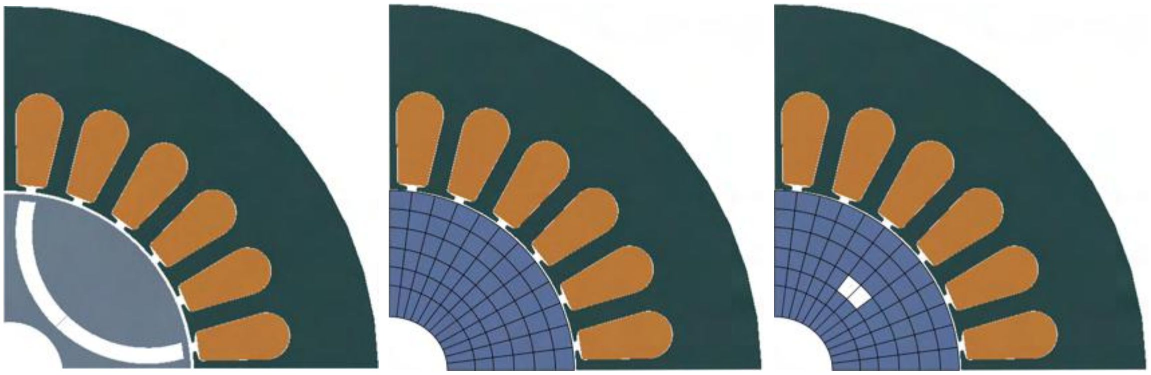


Figure 4.27: Conventional rotor designs for synchronous reluctance motors (a) salient pole type, (b) axial lamination type (c) transverse lamination type (Kolehmainen 2010).

Although reluctance-based machines were invented in the 19^{th} century, they have gained popularity only since the advancement of microelectronics and control systems. SynRM operates at exact - "synchronous" speed by syncing the rotor speed with the stator's rotating magnetic field. The rotating MMF in the stator can be generated by exciting a 3 phase stator winding. The rotor poles exhibit saliency due to internal magnetic barriers (slots, airgaps) or external notches. The presence of notches/barriers/slots in the rotor structure can affect the profile of the reluctance torque experienced by the rotor. The three types of synchronous reluctance rotors, as per the literature, are the simple salient pole rotor, the transverse laminated rotor, and the axially laminated rotor (Kolehmainen 2010), as seen in Figure 4.27. Due to the law of conservation of energy, the rotor will always try to move to a position of least reluctance with respect to the stator. With the help of proper and controlled excitation in the stator poles, the movement of the rotor will be continuous and at a fixed "synchronous" speed. However, the average torque produced by a SynRM is still inferior when compared to other motors such as a permanent magnet motor (Murakami et al. 1999) and is an active field of research (Li, Guimarães, and Lowther 2013; Hidaka and Igarashi 2017; Kim and Park 2010). As a SynRM produces reluctance torque through a salient rotor structure, the rotor flux barriers play a vital role in the design optimization of SynRMs, since they force the main flux through the desired iron paths within a fixed stator geometry. Thus, to improve the performance of SynRM, it is important to optimize the shape of the barrier of the rotor of a SynRM. The objective will be similar to that in Section 4.4.1 and also stated in eqn. 4.14, where $f(a)$ is the negative magnitude of the average torque generated by the motor. The sequence of actions (a) controls the movement of the controller. However, the material distribution controlled by a is air $(p_j : 1 \rightarrow 0)$, resulting in a rotor's flux barrier.

$$\begin{aligned} &\underset{\substack{a_i=0,1,2,3\\ i=0,1,2,\ldots,m}}{minimize} \quad f(a) \\ &\text{subject to} \quad K1 \leq \sum_{j=1}^{n}(p_j = 0) \leq K2 \end{aligned} \tag{4.14}$$

The definition of other parameters $(p, K1, K2)$ associated with defining the problem remains the same as in Section 4.4.1. The geometry for baseline performance was extracted from Siemens's MAGNET (*MAGNET v2020.2*) with the fixed parameters mentioned in Table 4.5 and displayed in Figure 4.28a. The baseline SynRM exhibits an average torque of 2.5 N-m. Taking advantage of the symmetry offered by a 4-pole, 3-phase design, only a quarter of the geometry is considered as the design space. The quarter 2-D design space is discretized into 25 (5×5) cells with the initial design filled with iron $(p_i = 1)$ and can switch



(a) Conventional single barrier SynRM (b) Discretized rotor geometry (c) Controller at start position.

Figure 4.28: SynRM characteristics

to air ($p_i = 0$), based on the movement of the controller. The cross-section of the discretized design space is shown in Figure 4.28b and the starting position of the controller in Figure 4.28c.

Fixed Parameters	Value	Fixed Parameter	Value
Number of stator slots	24	Number of Poles	4
Stator outer diameter	112 mm	Airgap thickness	0.5 mm
Rotor outer diameter	55 mm	Stack height	50 mm
Rotor inner diameter	16 mm	RMS current density	18 A
Core material	M-19 26 Ga	Barrier material	Air

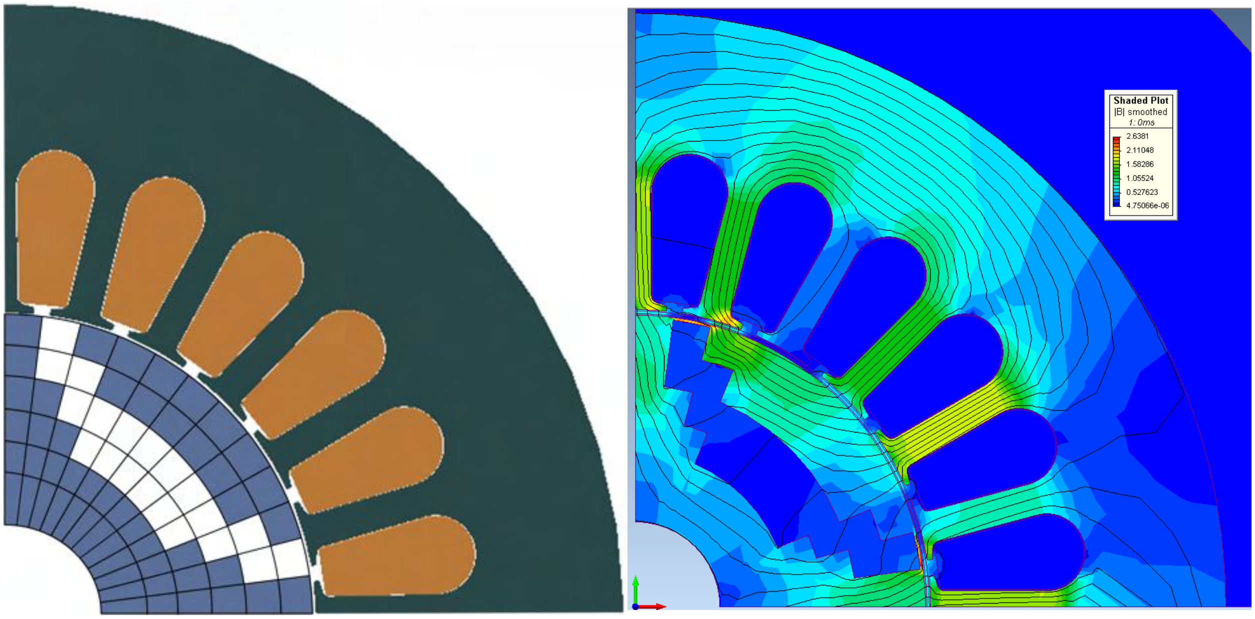
Table 4.5: SynRM fixed parameters

4.8 SynRM - Topology Optimization

4.8.1 GA Results with ***SeqTO-v1***

For the combination of geometric and excitation parameters mentioned in Table 4.5, a sequence-based controller is employed on a 5 × 5 discretized space. The discretized design space (Figure 4.28b) is filled with iron and the controller is responsible for introducing air. A GA based topological optimization is performed to obtain an optimal flux barrier topology. Figure 4.29a shows the optimal topology obtained. The GA algorithm as set in Section 4.4.1 with $m = 25$, population size of 50 and stall generations set to 100. With an average torque of 3.40 N-m, the resultant geometry achieves a significant improvement in the average torque as compared to the torque (2.5 N-m) of the conventional design (Figure 4.28a). To validate the results by the sequence-based controller, the ON/OFF algorithm described in Midha et al. 2019 was also employed, which showed similar results. The average torque ($-f(a)$) is obtained through transient 2D Finite Element simulation (*MAGNET v2020.2*).

The obvious reward function in such a case will be the performance (objective) criteria that is being maximized/minimized. For a C-core, it was the force experienced by the armature at any given step of the MDP. Similarly, for SynRM, it is the average torque generated by the motor at any given step of the MDP.



(a) Optimal geometry

(b) Magnetic $|B|$ field distribution

Figure 4.29: (a) Optimized geometry and the magnetic flux distribution for a discretized SynRM (5×5 design space). SeqTO-*v1* with GA.

Environment Type	Design Variables (m)	Population Size	Stall Generation	Average Torque
5×5 (Non-linear)	25	25‖50	25	3.40 N-m

Table 4.6: TO results for SynRM using GA on the rotor structure; based on *SeqTO-v1*

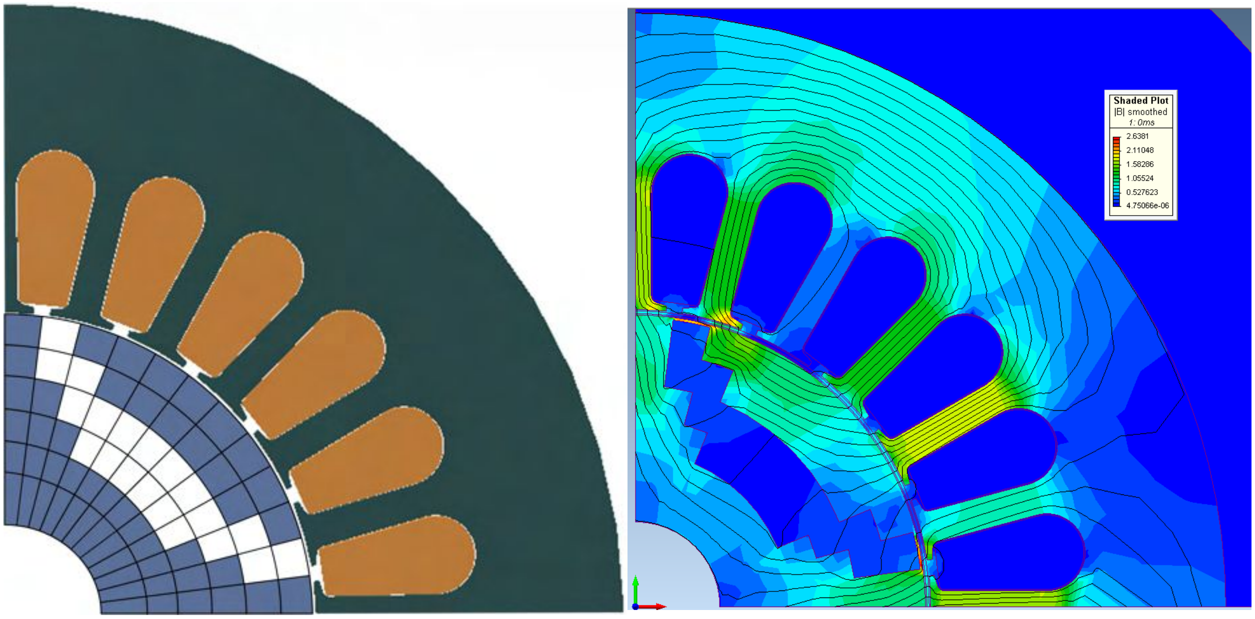
4.8.2 SynRM - TD-Learning results

For the combination of geometric and excitation parameters mentioned in Table 4.5, a sequence-based controller is also employed on a 5x5 discretized space. The discretized design space (Figure 4.28b) is filled with M-19 26 Ga material and the controller is responsible for introducing air (barrier material). The optimal topology of the rotor is shown in Figure

4.30a. With an average torque of 3.40 N-m, the resultant geometry, shown in Figure 4.30a, achieves a significant improvement in the average torque as compared to the torque (2.5 N-m) of the conventional design. The average torque (T_{avg}) is obtained through transient 2D Finite Element simulation (*MAGNET v2020.2*).

Environment Type	Episode Length	Learning Rate	Reward Function	Average Torque
5 × 5 (Non-linear)	25	0.8	∇T	3.40 N-m

Table 4.7: SynRM results using TD-learning with SeqTO-*v1*.



(a) Optimal geometry

(b) Magnetic $|B|$ field distribution

Figure 4.30: (a) Optimized geometry and the magnetic flux distribution for a discretized SynRM (5 × 5 design space). SeqTO-*v1* with TD-learning.

4.9 Comparison between ON/OFF & SeqTO-*v1*

Apart from the advantage of enforcing the connectivity of material, which directly results in manufacturable designs, Sequence-based TO is also efficient in terms of the total number of function calls during the optimization procedure, as seen during experimentation. This improvement can be verified by the data in Table 4.8, where the number of function calls made using Sequence-based TO is reduced by a factor of around 4 - 5, as compared to ON/OFF TO methodology for the same design problem.

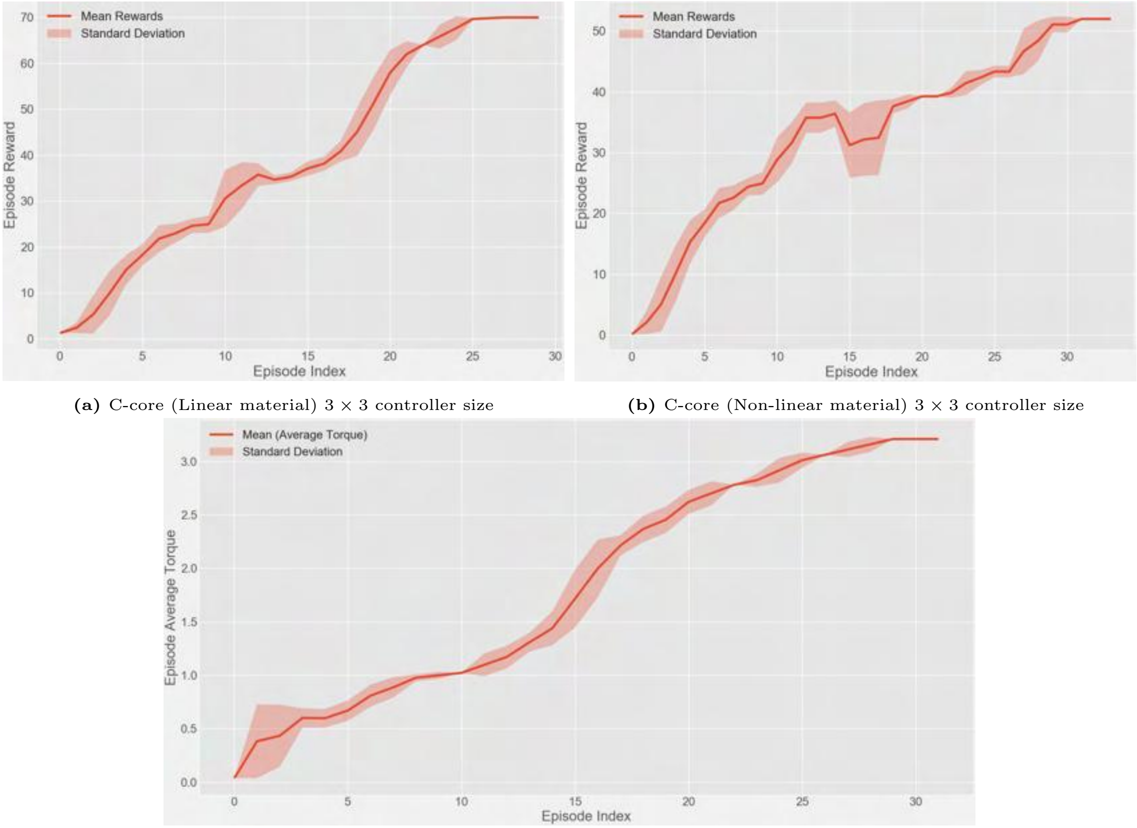
	GA ON/OFF		GA SeqTO-*v1*		TD SeqTO-*v1*	
Scenario	Optimal value	FE Simulations	Optimal value	FE Simulations	Optimal value	FE Simulations
3 × 3 C-core Linear	70 N	20000	70 N	4500	70 N	3750
3 × 3 C-core (NL)	51 N	45000	51 N	7200	51 N	6000
SynRM	3.4 N-m	12000	3.4 N-m	3000	3.4 N-m	2800

Table 4.8: Comparison of On/Off TO with Seq based TO using GA & TD-learning

This computational efficiency can be attributed to the reduced dimensionality of possible candidate designs to explore with SeqTO, due to the inherent requirement of enforced connectivity. This limitation imposed by SeqTO helps avoid the evaluation/generation of candidate designs with floating or isolated material. Further, in a comparison between the GA and TD-based learning when applied on SeqTO, it is found experimentally that TD based learning requires about 15-20% fewer FE evaluations for generating the same optimal designs for linear and non-linear C-core design problem (using a controller of size 3 × 3). However, the number of FE calls for both GA and TD show a smaller difference (around 6.5%) for TO of the SynRM, which can be explained due to the relatively smaller design space of SynRM of $5 \times 5 = 25$ as compared to that of a C-core (288 elements in C-core design space). The computation cost presented in Table 4.8 is calculated by the average performance over many independent runs to make a scientific comparison. The random seed is the only difference between runs of these algorithms.

The transitional progress of learning with the TD-algorithm is shown in Figure 4.31 (a), (b), (c) for C-core (using linear and non-linear material) and SynRM respectively. The graph is plotted by testing the TD algorithm on a fresh episode after a regular interval and no training is performed during this test episode. The solid line in the curves is the running average of the past few test runs and the standard deviation of the local averaging. As can be seen, the TD algorithm continues the learning process until the running average converges to a fixed value and the variance diminishes to a very low value.

Although a FE solver (*MAGNET v2020.2*) is used for all electromagnetic analyses in this study, this is neither a necessary requirement nor a limitation associated with any of the optimization algorithm discussed in this work. Any EM solver able to evaluate the desired performance criteria for the problem geometry and has an interface with the optimization algorithm should be sufficient enough for this work.



(a) C-core (Linear material) 3 × 3 controller size

(b) C-core (Non-linear material) 3 × 3 controller size

(c) SynRM 5 × 5 design space

Figure 4.31: Training Rewards vs Episodes

4.10 Conclusion

The feasibility of a sequence-based (SeqTO-*v1*) controller for Topology Optimization is proved on several geometries of varying complexity. Furthermore, the working applicability of SeqTO with vastly different algorithms such as GA and QL as potential optimization processes is also shown. The structures obtained using the sequence-based controller were free from any island/stray material. The ability of SeqTO to generate such compact designs is useful for multiple engineering domains apart from electromagnetics, as the methodology can be easily imported to other design problems. The ability to work with significantly different algorithms such as GA and TD-learning and resulting in optimal solution proves the efficacy of sequence-based design optimization. In TD-learning, the additional information gained through observing the actions taken by the controller along with the effect on the performance, at different steps of an episode, introduces interpretability in the whole design process. This additional information can be useful for designers to understand the impact of different components in the design. Exploiting this knowledge can result in further hand-tuned designs and offer more flexibility to a designer.

Moreover, breaking the whole geometry into a series of sub-components lays the path to apply more generalizable algorithms such as Deep Reinforcement Learning models, using Deep Neural Networks (DNN). Leveraging the complexity of a DNN to handle problems of high dimensionality has additional advantages, such as the ability to transfer knowledge learned from one environment to another, conditioned on having a similar sub-structure and relatable performance criteria. These possibilities will be explored in Chapter 5.

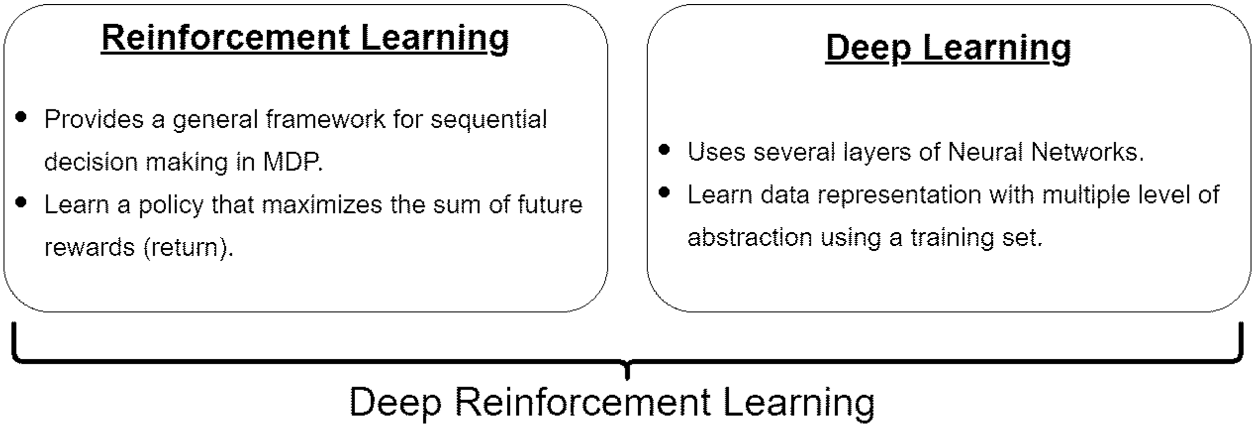
Chapter 5

Intelligent Agents for Topology Optimization

5.1 Introduction

Inventors have always dreamed of machines that can think. When programmable computers were first conceived, people wondered whether machines might become intelligent, a hundred years before one was even built. Building intelligent machines involves the capability to learn from examples, experiences, or even from other machines (Turing 1950). The true challenge for Artificial Intelligence (AI) is solving tasks that provide no supervision and involves its evolution of knowledge from experience. This is significantly different from the supervised learning algorithms discussed in Chapter 2 and 3, where the task was well defined in terms of the labeled data, and the feedback provided to the learning algorithm was in the form of a correct response (ground truth/labeled data). In contrast to supervised learning, a far more complicated task will be to enable a learning technique to improve over time in an interactive environment through trial and error and using feedback from its actions and experiences. This type of learning (RL) when coupled with neural networks is the basis of Deep Reinforcement Learning (DRL), as shown in Figure 5.1.

To some extent, the psychology principle - ‘Law of effects’ (Thorndike 1913), associated with the behaviour of animals, can be related to the foundation of Reinforcement Learning. The notion states that “responses to experiences that produce a satisfying situation become more likely to occur again in that situation, and responses that produce a discomforting effect become less likely” (Gray 2011). This principle was advanced by Edward Thorndike, who was the first to introduce the concept of ‘reinforcement’ in the area of the psychological principle of learning (Thorndike 1898). The terms ‘satisfying’ and ‘dissatisfying’ in the definition were later replaced with the words ‘reinforcing’ and ‘punishing’ respectively (Mazur 2015).



Reinforcement Learning

- Provides a general framework for sequential decision making in MDP.
- Learn a policy that maximizes the sum of future rewards (return).

Deep Learning

- Uses several layers of Neural Networks.
- Learn data representation with multiple level of abstraction using a training set.

Deep Reinforcement Learning

Figure 5.1: RL and DL combine to form DRL paradigm of algorithms.

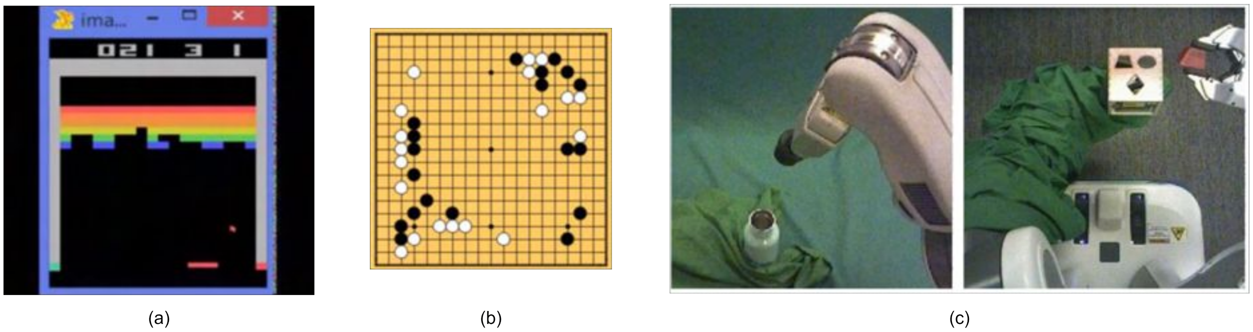


Figure 5.2: Application of DRL in the field of (a) Atari games, (b) Game of Go and (c) Robotics

DRL is expected to revolutionize the field of AI and represents a step towards building autonomous systems with a higher-level understanding of the physical world. The field received a lot of attention for the development of an algorithm that can play a range of Atari games (Silver et al. 2016). This was a breakthrough in artificial intelligence since scientists thought that it would take many more years to make machines win such complex games. Currently, deep NNs are enabling reinforcement learning to scale to previously intractable problems, such as creating a single system that can play multiple video games directly from pixels, named AlphaGo (Silver et al. 2016) and AlphaGo Zero (Silver et al. 2017). DRL algorithms are also applied to robotics (Kober, Bagnell, and Peters 2013), allowing control policies for robots to be learned directly from sensory inputs in the real world (Levine et al. 2016), designing algorithms to automatically learn to allocate and schedule computer resources to waiting jobs, with the objective to minimize the average job slowdown (Mao et al. 2016), design a traffic light controller to solve the congestion problem (Arel et al. 2010), Bidding and Advertisements (Jin et al. 2018). Recently, DRL was tested on predicting the 3D shape that a protein will fold into. The results of the biennial protein-structure prediction challenge called CASP (Critical Assessment of Structure Prediction) is seen as a breakthrough in the application of DRL in the field of drug discovery (Senior et al. 2020), which is seen as a field that is not part of the conventional RL domains. While the present achievements of DRL in the field of robotics and computer games are impressive and required a massive amount of resources to discover, this chapter extends the work in the field of DRL to Topology Optimization.

5.2 Literature survey

5.2.1 Application of ML algorithms in TO

In the past decade, machine learning algorithms have been extensively studied in the field of Topology Optimization. This is due to the capability of ML algorithms to approximate highly non-linear functions. A trained ML surrogate model is capable of predicting structures based on the patterns and behaviors learned from the analysis of several examples. Machine Learning (ML) and in particular Deep Learning (DL) have also demonstrated some capabilities in reducing the computational cost associated with topology optimization. A comparison for most of the recent work in the field of TO involving ML algorithms is shown in Table 5.1. As can be seen, most of the work is based on labeled data and involves supervised learning algorithms. After training with optimized solutions from conventional methods for TO- gradient-based or stochastic methods, a neural network can be trained in a supervised manner to predict solutions of the same TO problem under different input conditions. The differing conditions represent the geometries obtained during the process of TO optimization. In the field of solid mechanics, Yu et al. 2019 used 100,000 optimal solutions to a simple mechanical compliance problem (distributing material in a domain so that the structure achieves maximum stiffness for given loading and constraints) with various boundary forces and the optimal mass fractions to train a neural network consisting of a Convolutional Neural Network (CNN) (LeCun et al. 1998) and conditional Generative Adversarial Network (cGAN (Goodfellow et al. 2014)), which can predict near-optimal designs of mass fraction for any given boundary forces. Further, Chandrasekhar and Suresh 2020 used neural networks to represent the density parameter in the Solid, Isotropic Microstructures with Penalization (SIMP) approach (Rozvany, Zhou, and Birker 1992) for a TO process. In another approach, Banga et al. 2018 showed the potential of a 3D Encoder-Decoder based approach to convert the TO problem to an image segmentation (Badrinarayanan, Kendall, and Cipolla 2017) one. The first usage of DL algorithms in the field of TO (Sosnovik and Oseledets 2019) in recent years was to use a Deep CNN Autoencoder to reduce input dimensionality. The latent information (usually extracted from the center-most hidden layer) extracted from a trained CNN-based Autoencoder helped in cutting down the computational time through dimensionality reduction but was not used as an optimization algorithm for TO.

It should be noted that all the researches listed above belong to supervised learning, which is a class of ML algorithms based on static and pre-computed training data, where the input-output (features-labels) pair must be provided. Thus, they suffer from a lack of generalizability and have been criticized in Deng, Qin, and Lu 2020 as not a TO algorithm, since they rely on existing optimal designs as the training data. The predictions are restricted by the coverage of the training dataset and thus are biased on the methodology for collecting the training samples. This can introduce data collection bias to the model, as a surrogate DL model will provide highly accurate results to the design space part of its training data. For any unseen new geometry, the performance of the ML model will be unreliable, if not inaccurate. To consider different domain geometries, these approaches will require new datasets and train new neural networks for any change in the conditions of the TO problem. This issue has been the focus of recent works. (Zhang et al. 2019) used a DL method based on CNN and GANs for TO with generalizability ability where instead of mapping sparse boundary conditions, physical fields were found to allow the Neural Network (NN) to learn a more accurate mapping between the boundary condition and the optimal structure. Thus, the advantage proposed by Zhang et al. 2019 comes from the well-designed input form, enabling the DL model to solve TO problems with different boundary conditions even though the training datasets had only one or a few boundary conditions. Nonetheless, the learning methodology used in Zhang et al. 2019 is supervised and has the same shortcomings as mentioned previously. Smart input encoding does not solve the underlying problem associated with supervised learning methodologies.

Paper	**Learning Architecture**	**Online Learning**	**Design Optimizer**	**Generalizabile**	**Manufacturalibility (part of TO)**
(Chandrasekhar and Suresh 2020)	Feedforward	✘	✘	✘	✘
(Banga et al. 2018)	CNN	✘	✔	✘	✘
(Sosnovik and Oseledets 2019)	CNN	✘	✘	✘	✘
(Lei et al. 2019)	KNN, SVM	✘	✔	✘	✘
(Nie et al. 2020)	GANs	✘	✔	✘	✘
(Yu et al. 2019)	CNN + cGAN	✘	✘	✘	✘
(Deng, Qin, and Lu 2020)	Feedforward	✔	✘	✔	✘
(Doi, Sasaki, and Igarashi 2019)	CNN	✔	✘	✘	✘
(Sasaki and Igarashi 2019)	CNN	✘	✘	✔	✘
(Barmada et al. 2020)	CNN	✘	✘	✔	✘

Table 5.1: Literature survey of ML papers in Topology Optimization

Furthermore, another issue associated with supervised learning is that the material (volume) constraint is treated as a condition to match as closely as possible rather than a true inequality constraint. This is due to the nature of supervised learning where the optimization algorithm tries to minimize a loss function to as low a value as possible. Due to the above-mentioned problems with supervised learning, Deng, Qin, and Lu 2020 limited the use of ML algorithms as a performance estimator (rather than an optimization algorithm) for the compliance minimization problem in mechanics. Deng, Qin, and Lu 2020 trained the DNN in an online fashion, where the training data was selectively collected to capture near-optimal solutions. Although this helped in accelerating the computation by using a DNN as a surrogate of the objective function evaluator, the ML algorithm is not involved in the TO process and thus does not influence the generation of new topologies. Further, as this approach limits the data generation for training only on near-optimal solutions, it will obstruct the ability of the network to distinguish between a good and a bad solution. As a consequence of a network trained with a highly imbalanced dataset, it can overestimate the performance indicator for a poor design and will affect the performance of the stochastic algorithm itself.

Advancements in TO for Electromagnetics

In the field of electromagnetics, Sasaki and Igarashi 2019 introduced TO for EM analysis where a CNN-based performance estimator is used to predict the torque values for the optimization of IPM motors. Further advancements were performed in Doi, Sasaki, and Igarashi 2019 and Barmada et al. 2020 where incremental efforts were made to reduce the computation burden associated with FE analysis. These approaches used DL methods as a filter for removing the inferior population at any iteration of the GA process. The best-performing designs are reevaluated with FEA to reduce the computation burden. This type of supervised learning has been implemented in both an on-line (Doi, Sasaki, and Igarashi 2019) and off-line fashion (Sasaki and Igarashi 2019).

5.2.2 Addressing the Disadvantages of Supervised Learning in TO

Given the advantages associated with involving ML algorithms, there is a motivation to extend the above-mentioned works by pursuing a methodology that focuses on extending generalizability as a part of the optimization process rather than using DL as a surrogate or labeled data-constrained supervised learning for TO. For this purpose, the concept of DRL is introduced for TO in this Chapter. In addition, the concept of manufacturability from Chapter 4 is also an important criterion and should be included from the initial phase of TO with RL.

5.3 Setting up Deep RL

The usage of deep reinforcement learning aims to create a system that can adapt to novel geometries through the use of suitable system dynamics in the form of an MDP and the usage of neural networks as function approximators, which can generalize to new geometries and excitations. This section contains a discussion on the setup of function approximator following a modification of Q-learning from Tabular (Section 4.5.3) to Deep Q-Learning methodology, along with the changes introduced in the MDP, in comparison to the one discussed in Section 4.5.1. As with Chapter 4, the term 'FE calls' is used to refer to the number of simulations performed in a corresponding optimization task. Although a FE solver is used for all electromagnetic analyses in this study, this is neither a necessary requirement nor a limitation associated with any of the optimization algorithm discussed in this work. Any EM solver to be used, as long as it is able to provide an accurate solution and a performance measure required for the operation of the algorithm, such as force magnitude and field distribution.

5.3.1 Advanced Function Approximators

This section builds on the Tabular QL algorithm discussed in Chapter 4 Section 4.5.3, where a greedy policy was obtained for optimal material placement. This involved learning the value function (Q-value) for the states and action pair associated with a TO environment (SeqTo-$v1$). The tabular QL approach will not be able to expand to high dimensional state and action space, due to the curse of dimensionality. It is not possible to compute and preserve the value function (Q-value) for all possible state-action pairs, especially when either of them (state or action) is continuous. Further, a more complex action space adds to the curse of dimensionality problem, making the tabular method either infeasible or too memory intensive. There are other ways to store and represent a value function. This chapter introduces function approximation to approximate the action values. These function approximators can be expressed with a relatively small number of parameters. The ML literature consists of many ways to parametrize an approximate value function (Q or V) such as Linear Regressors, Decision Trees, SVMs, and Neural Networks to name a few classes of available options. In principle, any function that takes the state as an input and produces a (or a set of) real number(s) that represents the value associated with that state, is considered suitable enough to be used as a function approximator for value function (Q or V).

A popular category used for this is parametrized functions where a set of real-valued weights are adjusted to learn a representation from the state space to the corresponding action value. These weights can be tuned to allow for a change in the output of the parameterized function to account for collected experience as and when the experience is generated in a sequential learning format. The aim is to generate an Action value function predictor $\hat{Q}(s, a, w)$ which approximates the true action-value function $Q_\pi(s, a, w)$. This is better than the tabular approach (Chapter 4), for large state and action spaces. Further, the tabular approach worked only with an integer-based encoding of the design domain, as shown in Figure 5.9. For a richer encoding technique as shown in either Figure 5.10 or 5.11, the tabular approach is not an appropriate form of storage. The floating-point data type needed to represent the magnetic field distribution in the air gap (Figure 5.10) and the whole geometry (Figure 5.11), will be either infeasible or will require high memory capacity, as mentioned earlier. Alternatively, an attempt to discretize the continuous magnetic field values will result in a precision trade-off. In such cases (Figure 5.10 & 5.11), a parametrized approximator is ideal for representing a state of the design domain which includes the magnetic field distribution and offers the flexibility of dealing with similar problems including geometries and excitation. Thus taking the advantage of parameter fitting and interpolation.

$$\hat{v}(s, w) \approx v_\pi(s)$$

Another difference between the tabular and parameter-based methodology is that modifying a single entity of the value function table leaves the value functions associated with other states (s) or state-action pair (s, a) unchanged. Now the learning algorithm will modify the weights of a parametrized function and will affect states which are closely related in the weights distribution space. For parametric representation, the learning algorithm will modify the weights of the parametrized function and this change will affect all the values associated with other states (s) or state-actions(s, a) that are closely related to the weight distribution space.

	a_0	a_1	a_2	a_3
S_0	8.2	-0.2	4.1	4.4
S_1	6.6	-1.0	-1.5	5.6
S_2	-3.3	-2.6	11.1	-4.7

Table 5.2: Example of a Q-table

The simplest form of function approximator is a linear function, which is a linear representation of the constituent weights for the Q-value (Table 5.2). These weights are multiplied with the features to get a scalar value for a state Action-value function. The features are some fixed attributes that explain the state. The performance of linear function approximators relies on good features that are capable of discriminating between different state-action pairs. Tabular representation is a special case of linear function approximation, where each state-action pair is a separate feature and thus requires a weight/Q-value for each instance of the state-action pair. This can be stated in the following equation as :

$$\begin{aligned}\hat{Q}(s_i, a_j, w) &\doteq < w, x(s_i, a_j) > \\ &= w_{ij}\end{aligned}$$

where $x(s_i, a_j)$ is one-hot encoding (Géron 2019). One-hot encoding is used since only a single entry in the Q-table will correspond to a state-action pair (s, a), as can be seen from the Table 5.2. One-hot encoding will produce a vector with only a single non-zero element (high) which corresponds to the highest possible value for each row of the Q-table.

So a tabular representation is the same as a linear approximation with the number of weights the same as the number of states, which, as stated above, is suitable for a small integer encoded state-action space. Neural networks, on the other hand, implement a nonlinear function of the states. The state, or features representing the state, is passed to the neural network as an input. All the connections through hidden layers correspond to real-valued weights. When data is passed through a connection the weight gets multiplied by the input. The weights form a connection between any two layers in the neural network. The process transforms the input state through a sequence of layers to finally produce the action value estimate as to the output of the neural network, as shown in Figure 5.3. The field of Deep Reinforcement Learning (Deep RL) deals with using Neural networks in combination with the learning algorithms of RL such as (Q-learning, Monte Carlo, Policy Gradients).

Although there are other possibilities for a function approximator, there are criteria that must be satisfied to produce a satisfactory predictor for the value function. It should possess the property of both generalizability and discrimination. Generalizability is the ability to apply knowledge about a specific scenario to a variety of similar situations. An update to the value estimate of one state-action pair influences the value estimate of other statistically similar state-action pairs. This is also helpful in speeding up the learning by making better use of the experience generated and ultimately help in faster convergence to the optimal solution (design), as not all distinct state-action pairs need to be visited. On the other hand

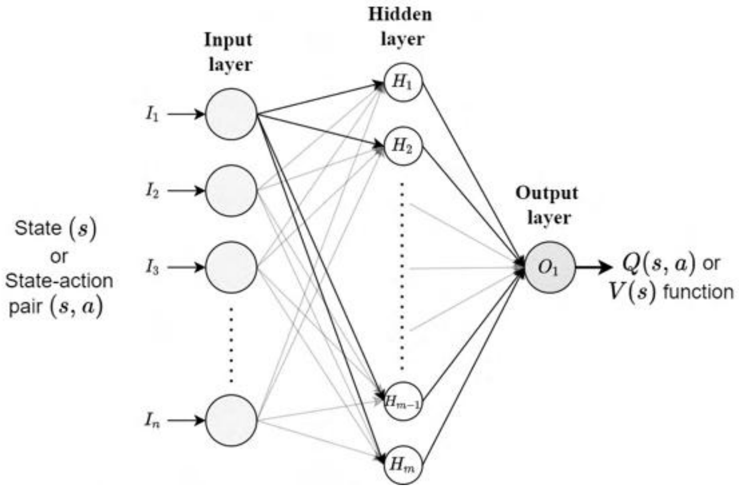


Figure 5.3: Feedforward neural network for Value function ($Q(s, a)$ or $V(s)$).

'Discrimination' is an ability to distinguish two significantly different states, by assigning different values in the form of prediction to significantly different states. There might be some states with similar features (as they have similar geometry) but drastically different returns. An example of this can be seen in Figure 5.4. An ideal candidate for a function approximator should be able to find distinguishable features that can help with the separation. Tabular methods are great at distinguishing two states perfectly but do not generalize at all. Each state-action pair is represented by a distinct entry in the table (Table 5.2). Each entry of the state-action pair (s, a) is independent of each other. Thus, the controller is forced to visit all the state-action pairs in the MDP to be able to find a greedy policy. A neural network on the other hand is good at finding a balance to the generalizability-differentiability trade-off (Hornik, Stinchcombe, and White 1989), conditioned that the features representing the states contain sufficient information. For the scenario shown in Figure 5.4, additional information along with the material distribution such as magnetic field distribution can help the function approximator with balancing the trade-off.

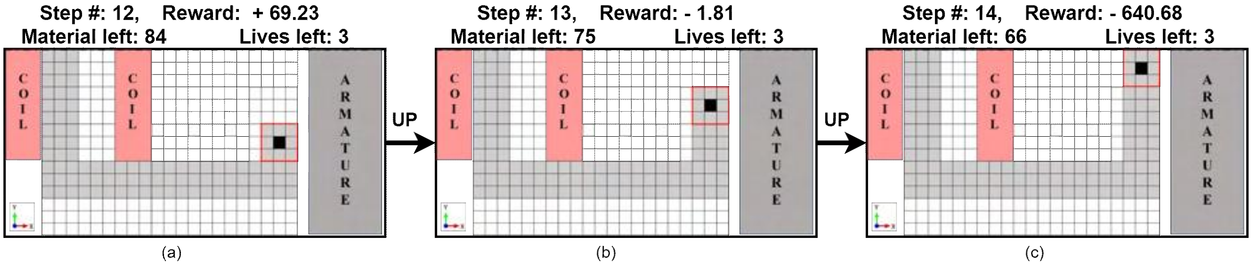

Step #: 12, Reward: + 69.23
Material left: 84 Lives left: 3
COIL
COIL
ARMATURE
(a)
UP
Step #: 13, Reward: - 1.81
Material left: 75 Lives left: 3
COIL
COIL
ARMATURE
(b)
UP
Step #: 14, Reward: - 640.68
Material left: 66 Lives left: 3
COIL
COIL
ARMATURE
(c)

Figure 5.4: Sudden decrease in performance with a slight change in the state representation. Coil Current: 1.5 Amps, No. of Turns: 500.

An additional characteristic that the learning system should have is the ability to learn in a sequential manner. This is in contrast to supervised learning as the data generation process in an RL task is online. The online nature of the data is attributed to the constant interaction of the agent with the environment. On the other hand, the data used in supervised learning is pre-generated and is independent of the current weights of the neural network. This dynamic interaction makes the data generated at any given time dependant on how well the agent understands the environment. This creates a non-stationary distribution of the training data, where previous knowledge gained by the learning system is to be discounted in favor of the new experience. This is one of the reasons why sample efficient algorithms are an active area of research in the field of Deep RL (Henderson et al. 2018). On top of that the ever-present challenge of reducing the computation cost associated with the task of TO. Identifying an ideal learning system that balances all the above-mentioned requirements is a difficult job and is heavily task-dependent. A learning system based on non-linear function approximators such as Neural networks is ideal for generalization, discrimination, and online learning but will require significantly more training examples to train as compared to a tabular method. However, the cost of higher sample requirements can be justified with the ability to generalize over similar optimization tasks.

5.3.2 Objective of the RL Task

The value estimation problem can be setup as a supervised learning task. The first task is to define an objective to optimize. A regression task can be setup such that the error is stated as:

$$\text{Error (MSE)} = \left[Q^*(s,a) - \hat{Q}(s,a,w)\right]^2 \tag{5.1}$$

Although the problem is formulated in the sense of supervised learning, it will be difficult to match the true value ($Q^*(s,a)$) of the action-value function at all states. Also, in most problems, the optimal value function ($Q^*(s,a)$) is unknown at the beginning of the training. Unlike the supervised learning tasks (e.g. the Field & Performance maps estimation), a high accuracy value that has been obtained in Chapter 2 & 3 does not necessarily reflect in a value function being a good predictor for the true action-value function. In the absence of the optimal value function ($Q^*(s,a)$), (as was discussed in Chapter 4, Section 4.5.2, in Temporal Difference (TD) learning), the learning algorithm uses it's own estimate of the value function to update it's own prediction. Another alternative is to use an estimate of the return (G_t), but as was discussed in Chapter 4, Section 4.5.3, the computation cost associated is too expensive.

For making an update to decrease the error on an experience trajectory:

$$(S_1, A_1), Q_\pi(S_1, A_1) \rightarrow (S_2, A_2), Q_\pi(S_2, A_2) \rightarrow (S_3, A_3), Q_\pi(S_3, A_3) \rightarrow \dots \tag{5.2}$$

will be determined by the gradient of the loss function from Eq. 5.1:

$$\alpha \nabla \Big[Q_\pi(s, a) - \hat{Q}(s, a, w) \Big]^2 \tag{5.3}$$

Using the estimation of the return as an approximation for the true Value function, for making updates in the weights (w) of the neural network predicting the action value function ($Q(s, a)$:

$$w \longleftarrow w + \alpha[G_t - \hat{Q}(s_t, a_t w)] \nabla \hat{Q}(s_t, a_T, w)$$

This updates the weights (w) of the neural network such that the action value estimate is closer to the return (G_t) of the sample experience trajectory. The return can be replaced with any other estimate of the action-value function. Similar to Section 4.5.3, instead of the return (G_t), a bootstrap target based on the action-value of the next step resulting in the following error equation:

$$\text{Error (MSE) } = \Big[R_t + \max_{a'} \hat{Q}(S_{t+1}, A_{t+1}, w) - \hat{Q}(S_t, A_t, w) \Big]^2 \tag{5.4}$$

The Q-learning with neural networks is implemented by stacking the actions in one neural network, such that a single network predicts the action value associated with all possible actions in the state as shown in Figure 5.5. The weights in the Q-learning neural network are updated as follows:

$$w \longleftarrow w + \alpha[R_{t+1} + \hat{Q}(s_{t+1}, a_{t+1}, w) - \hat{Q}(s_t, a_t, w)] \nabla \hat{Q}(s_t, a_t, w) \tag{5.5}$$

This method of learning is called on-policy learning. A more complex version of it, uses two neural networks for learning the $Q(s, a)$ value. One is called the behaviour Q-Network and the other is target Q-network, resulting in the following update equation:

$$w \longleftarrow w + \alpha[R_{t+1} + \hat{Q}_{tar}(s_t, a_t, w_{tar}) - \hat{Q}_{beh}(s_t, a_t, w_{beh})] \nabla \hat{Q_{beh}}(s_t, a_T, w_{beh}) \tag{5.6}$$

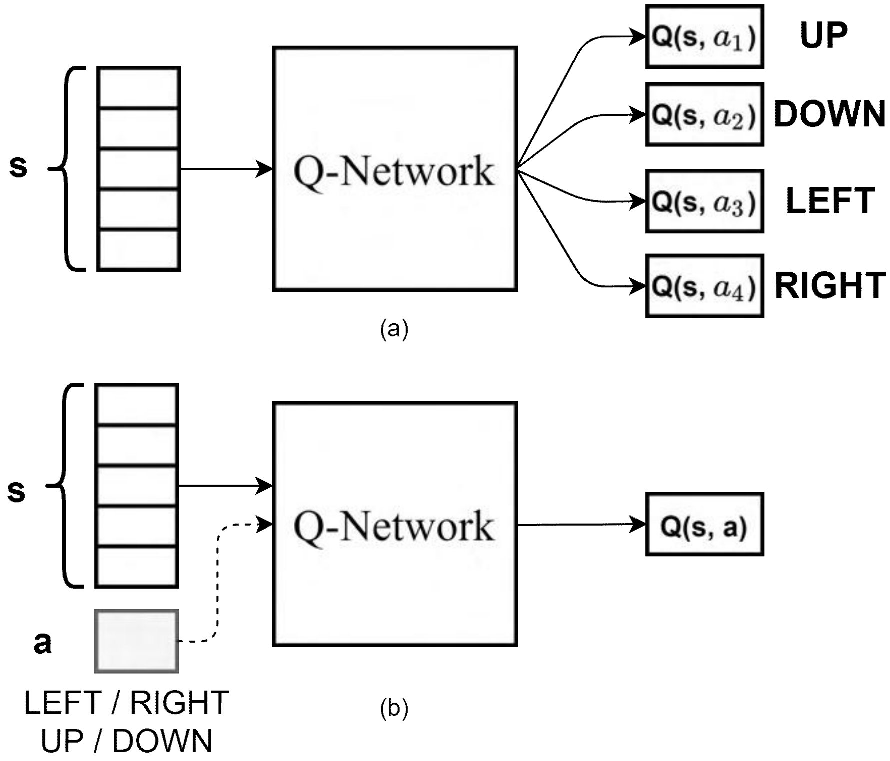


Figure 5.5: Different approaches of setting up the Q-Network

where $\hat{Q}_{tar}(s_t, a_t, w_{tar})$ is the target policy and $\hat{Q}_{beh}(s_t, a_t, w_{beh})$ is the behaviour policy. The behaviour NN is responsible for suggesting actions while collecting the experience. The target policy on the other hand does not interact with the environment directly but is used to train the behaviour policy using the eqn. 5.6. The target policy is obtained from the behaviour policy by the timely transfer of weights from the behaviour policy (shown as 'Knowledge Transfer' in Figure5.6). This method of learning is called off-policy learning, as the policies used for generating experience and training are different. This type of learning has shown improved performance in the training of Q-learning (Hasselt 2010).

$\epsilon-$greedy is used for exploration for both off-policy and on-policy, same as that in Section 4.5.2, where the emphasis is on exploration of the environment at the start of the training and as the agent learns the dynamics of the environment the scale of exploration is gradually decreased.

$$a_t = \begin{cases} \arg\max_a Q_{(beh)}(s, a; w_{(beh)}) & \text{with probability } 1-\epsilon \\ a \sim (a_1, \ldots a_k) & \text{with probability } \epsilon \end{cases} \tag{5.7}$$

Finally, it is also not expected that the approximator is giving great performance on all possible state-action pairs. Importance should be given to samples (state, action) from experience which the policy visits more often than others. Samples that occur more compared to others in a policy should be weighted higher than the ones where the time spent is less.

An agent needs to try many different actions in numerous states to try and learn all the available possibilities and to find the optimal material distribution which will maximize its overall reward; this is known as 'Exploration', as the agent is searching for better prospects in the environment. On the other hand, if all the agent will do is random exploration, it will never maximize the overall reward so it must also use the information it learned from the experience. This is known as 'Exploitation', as the agent exploits its knowledge to maximize the rewards it can receive. The trade-off between the two is one of the most significant challenges of RL problems, as the two must be balanced to allow the agent to both explore the environment enough, but also exploit what it has learned and repeat the most rewarding path it found.

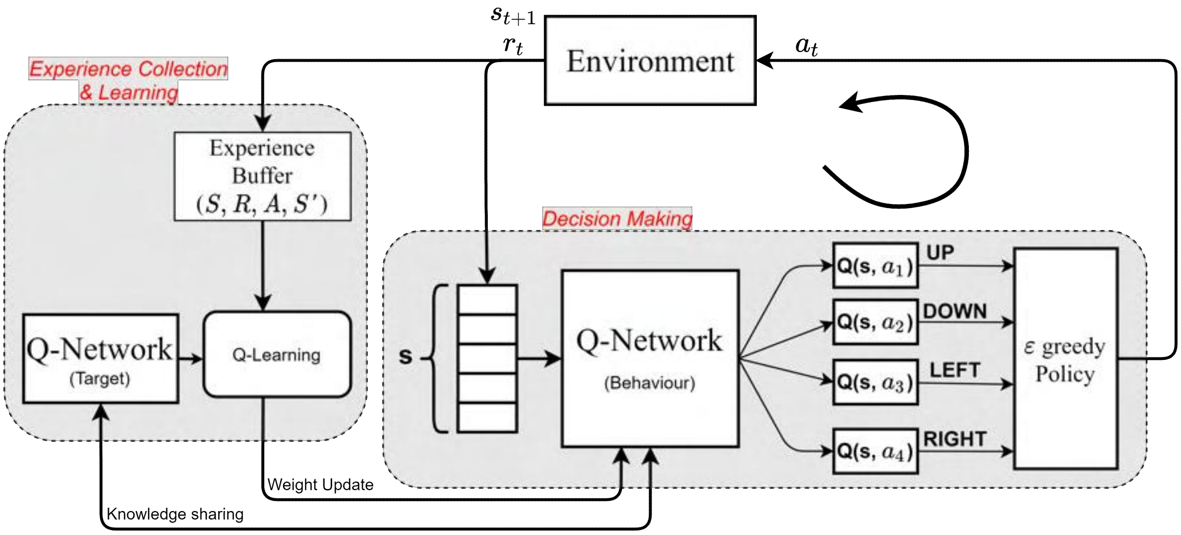


Figure 5.6: Q-Learning Cycle

5.3.3 Experience Replay

As RL tasks have no pre-generated training sets which they can learn from, the agent must keep records of the state transitions it encountered so that it can learn from them later. The memory buffer used to store this is referred to as 'Experience Replay'. There are several types and architectures of these memory buffers - but some very common ones are the cyclic memory buffers, which make sure the agent keeps training over its new behaviour rather than transaction experience that might no longer be relevant. The memory operation of a cyclic memory buffer behaves like that of a queue, where the oldest entry is removed first, once the buffer reaches its capacity. Another type of memory buffer is reservoir-sampling-based, which guarantees each state-transition recorded has an even probability to be inserted into the buffer. In this work, only cyclic memory buffers were used since they limit the amount of memory required for storing the experience buffer.

5.3.4 Deep Q-learning

DeepMind's DQN (Deep Q Network) was a breakthrough success in applying deep learning to RL problems (Mnih et al. 2013). Combining the previously discussed concepts of Experience Replay, ϵ based exploration, and the algorithm of Q-learning the algorithm is used to train a Neural Network (Q-Network, referred as DQN) to find an optimal policy that is greedy on the cumulated reward function. Algorithmically, DQN is related to the tabular Q-learning algorithm discussed in Section 4.5.3. However, the key difference is the involvement of advanced functional approximators for replacing state-action table (from Q-learning; Section 4.5.3) with a neural network to cope with large-scale tasks, where the number of possible state-action pairs can be enormous. This variation introduces two key addition to the Q-learning procedure: Experience Replay (discussed in Section 5.3.2) and the use of a separate target network (Section 5.3.2). The setup of the function approximation-based Q-learning method is shown in Figure 5.6. The core of the C-core (shown in Figure 5.7) is to be designed using TO such that the force experienced by the armature is maximized. The excitation for the current coil is set at 1.0 Amps with 500 turns.

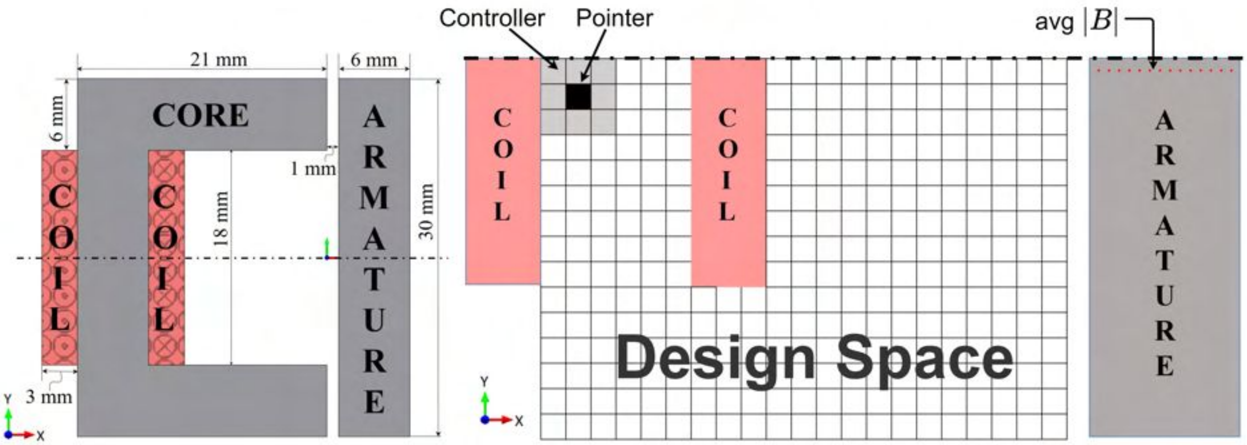
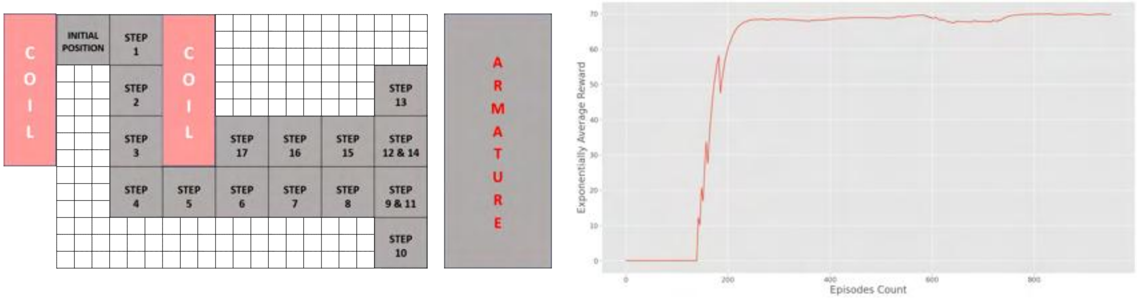


Figure 5.7: (a) Conventional C-core actuator (b) Design domain for half C-core with the controller (3 × 3), pointer and performance measure.

5.3.5 Results of Q-Learning

The optimal geometry for a 3×3 SeqTO environment with a linear magnetic material is shown in Figure 5.8. This result matches the geometry obtained through the GA and tabular TD-learning/Q-Learning algorithm (Figure 4.26a). The network uses the state representation similar to the one used for Tabular QL in Section 4.5.3. The memory buffer is populated with samples collected following a random policy. It was observed that starting with random sampling helps with the exploration on top of the ϵ- greedy policy. Once sufficient random samples have been generated the learning phase of the QL-based agent starts. The results obtained match those obtained with the tabular QL algorithm in Section 4.9. The network takes about 950 episodes to converge to the optimal solution, requiring about 20,000 FE function calls in the process.



(a) Convergence results for 1.0 Amps using Deep Q-learning algorithm.

(b) The optimal solution and action sequence in a 3 × 3 linear environment

Figure 5.8: Optimal geometry of C-core with Deep Q-Learning, using a 3 × 3 controller filling linear iron material.

5.4 Modifications to SeqTo-v1 environment

The RL task needs to be formulated as a Markov decision process (MDP) (Bellman 1957), similar to the one discussed in Chapter 4. Briefly speaking, an MDP comprises of four parts S, A, P, R, where P is the transition probability function. The description of the transition probability functions P is omitted because any state-action pair deterministically leads to a unique next state. The reward is based on the increase or decrease of the magnetic field distribution at the point near the middle of the armature as shown in the armature.

Although the MDP discussed in Chapter 4, Section 4.5.1 is suitable for deploying RL algorithms, the current discussion extends the MDP to account for more complex geometries while keeping the computational cost low. Further, certain modifications are to be made to enable the state to contain more relevant information, that can be useful for the agent to make a decision.

Additionally, up until now only a 3×3 controller has been used with both the tabular and function approximator form of Q-Learning. This limitation does not allow the generation of complex structures. If the same state representation and MDP is to be used with a controller of the size of 1×1, the number of steps needed to achieve an optimal structure with the amount of given material will be overwhelming, resulting in high number of simulations. Additionally, a state should include as much relevant numerical data of the design as possible. The MDP formulated in Section 4.5.1, was ideal for tabular algorithms as it consists of only integer encoding for the state representation.

To mitigate these issues, a state representation that holds more information and an action space that is more complicated and at the same time offers more flexibility to the controller is developed and this modified environment will be referred to as SeqTO-v2. SeqTO-v2 will include different ways of encoding useful numerical information into the state representation. This change is made for both the C-core problem and SynRM geometry. Although for SynRM no changes are made in the action dimensionality (as compared to SeqTO-v1), the techniques mentioned can easily be adopted. These changes are incorporated to balance the granularity of the solution and limit the number of electromagnetic simulations in an episode of training. The state representation used in Chapter 4 is shown in Figure 5.9. In addition to the material distribution and coils, an alternate representation involves additional information of the magnitude of the magnetic field distribution in the airgap, as shown in Figure 5.10. Another possibility is providing information related to the magnetic field distribution over the complete design space. This is shown in Figure 5.11, where the information is stored in channels as the first channel of dimensions $(W \times H)$ consisting of the material distribution along with the spatial location, current magnitude, and the number of turns for the coil. The second channel consists of the magnitude of the magnetic field distribution over the same spatial dimension as the first channel of a multi-channel or 3D representation. This is the most complete state representation of the three.

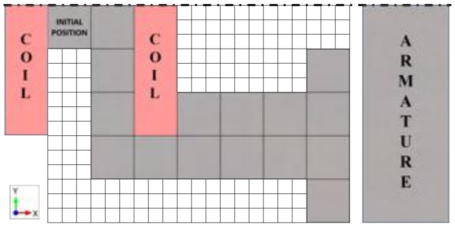


Figure 5.9: Discrete design domain as the state space

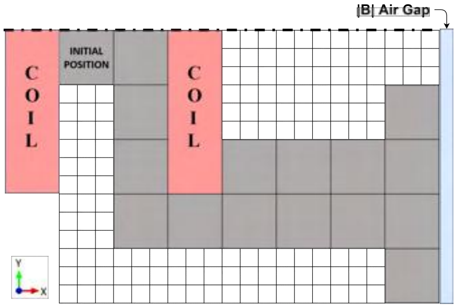


Figure 5.10: Discrete design domain with the $|B|_{airgap}$ as the state space

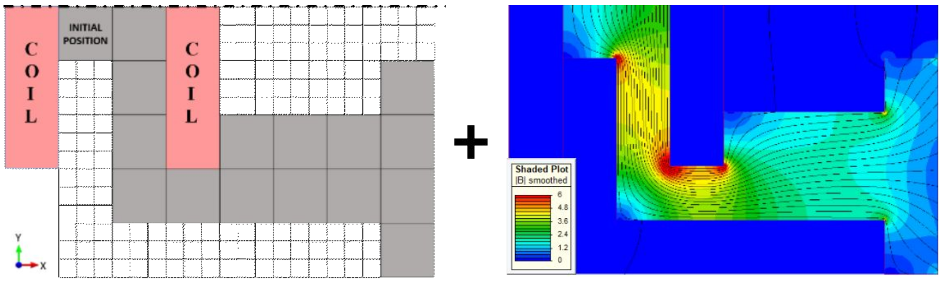


Figure 5.11: Discrete design domain along with the $|B|$ in the design space as the state space

As for the action space, it initially consists of only four options which were discussed in Chapter 4, Section 4.3 and shown in Figure 4.8. The action space for the controller is increased by four to a total of eight such that the controller now has the ability to place not only 3×3 block of data, but additional option including 3×1, 1×3, 3×2, 2×3 along with several other combinations as shown in Figure 5.12. Effectively, the controller now has the capability of placing from a minimum of 3 (1×3) to a maximum of 9 (3×3) in a single step, using an additional set of actions labelled as 'soft' actions. The complete set of actions, available to the controller at any step in an episode are mentioned in Table 5.3 and the operation of these actions is shown in Figure 5.12.

	Action		Action
1	Soft Up	5	Up
2	Soft Down	6	Down
3	Soft Right	7	Right
4	Soft Left	8	Left

Table 5.3: Evolved action space of the controller (SeqTO-v2).

Although the Markov property states that there must not be any useful information in previous states (Markov and Nagornyi 1951). This property can be relaxed, similar to Silver et al. 2017. In the case of using a single design domain information at an instant 't' as our state, we are breaking the Markov property because previous frames could be used to infer the previous changes made by the agent and can help with the differentiation if the agent took any good or bad actions there. An agent trying to enter the coil space is bad and can be differentiated here, as shown in Figure 5.13. This is dealt by bringing some of the history into the state.

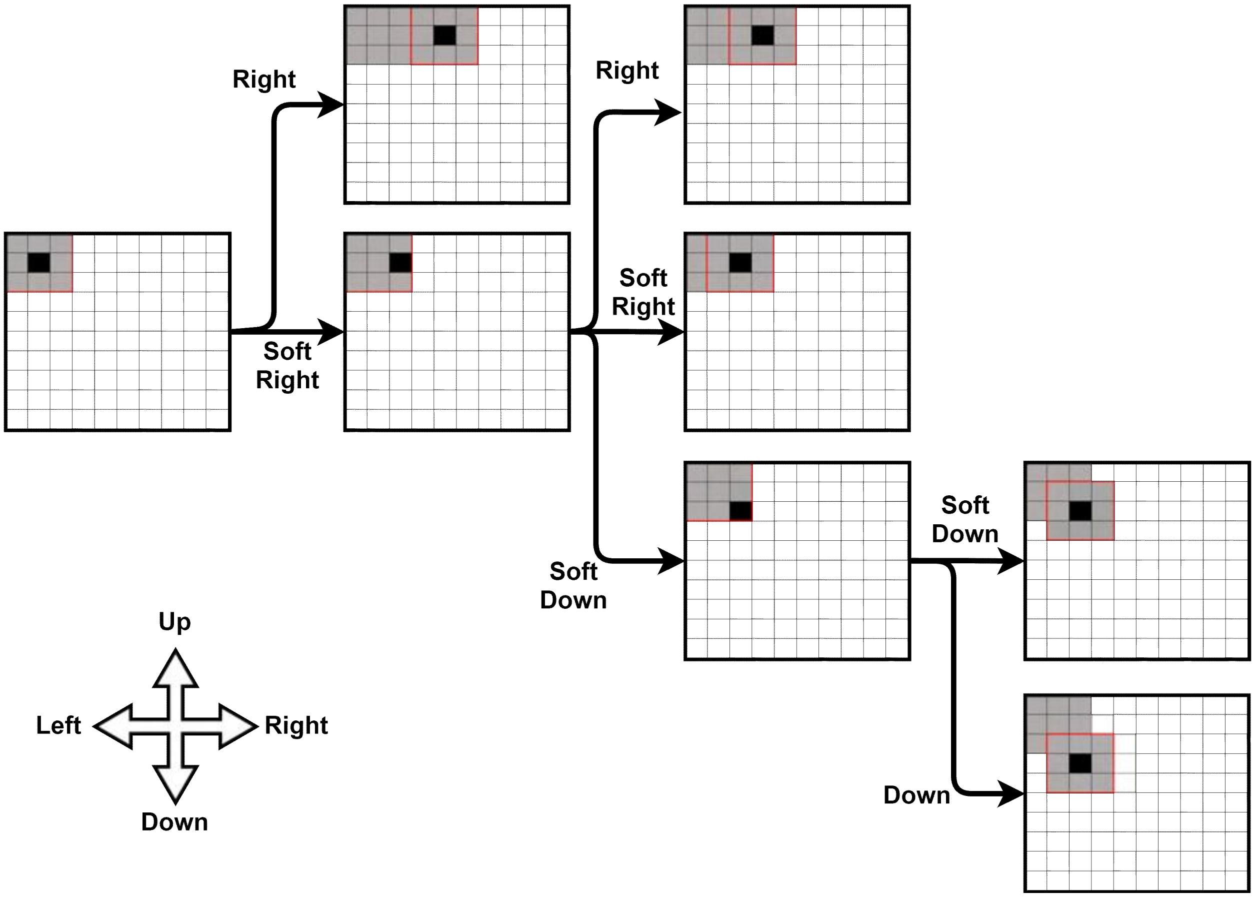


Figure 5.12: Agent Movement with SeqTO-v2 controller.

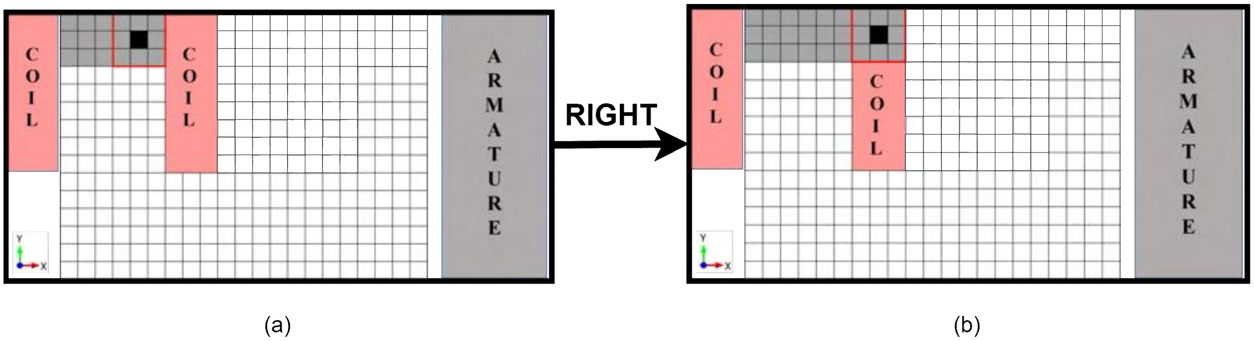


Figure 5.13: Possibility of a design error due to the controller moving out of the designated design space.

5.5 Policy Gradient

A Q-Learning algorithm learns by trying to find each state's action-value function (Q-Value function). Its entire learning procedure is based on the principle of figuring out the quality of each possible action and select the best possible action, according to the learned knowledge.

So Q-Learning tries to have complete and unbiased knowledge of all possible moves, which is also its most major drawback, as this requires a significant number of attempts of each possible state and action combination.

$$\text{Training} \Longrightarrow \theta \longrightarrow Q(s,a) \longrightarrow \pi \tag{5.8}$$

Policy Gradient (PG) is a different paradigm of RL algorithm that learns more robustly, by trying not to evaluate the value of each action, but by simply evaluating which action it should prefer. The learning cycle of a PG algorithm is shorter, skipping the Q-Value function part:

$$\text{Training} \Longrightarrow \theta \longrightarrow \pi \tag{5.9}$$

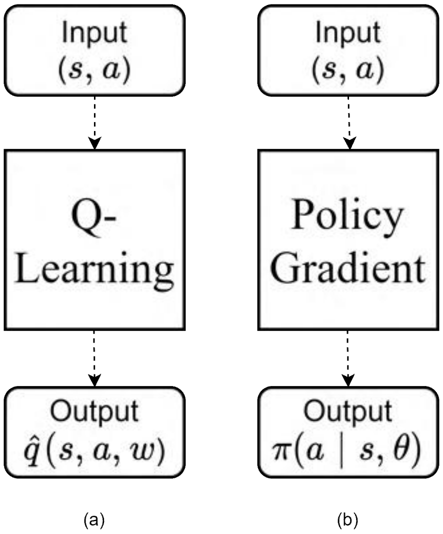


Figure 5.14: Q-learning vs Policy Gradient Method

This means that the network will directly output the probability of selecting each action, skipping the extra calculation. This also allows the PG algorithm to be more robust. The update for the Q-learning algorithm was based on either the return (G_t) or the bootstrapped action value ($R + Q(s', a')$). In PG, a metric called the "performance measure" (represented by the term J) will be responsible for making the changes in the weights of the neural net.

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{5.10}$$

A simple way of analyzing the performance ($J(\theta)$) is to measure all the rewards seen by the agent, similar to the return (G_t). The rewards accumulated in an episode is a good performance measurement since the agent's sole purpose is to maximize its overall collected rewards. This can be calculated as:

$$J(\theta) = G(t) \approx \sum_{a} \Big(Q(s_0, a) \times p(a|s_0) \Big) \tag{5.11}$$

Next, the weights of the neural network are to be optimized with respect to the performance criteria. It should be noted $J(\theta)$ depends on the probability of action selection $p(a)$, which is derived from the policy (π). The gradient of $J(\theta)$ is calculated and the weights are updated by:

$$\nabla_\theta J = G_t \cdot \nabla log(\pi(a_t|s_t;\theta)) \tag{5.12}$$

where G_t is the total accumulated reward from step 't' till the end of the episode. The change in the weights of the policy network is represented as:

$$\theta_{t+1} = \theta_t + \alpha(G_t \cdot \nabla log(\pi(a_t|s_t;\theta))) \tag{5.13}$$

where α is the learning rate and θ is the representation of the parameters (weights) associated with the neural network of the policy (π).

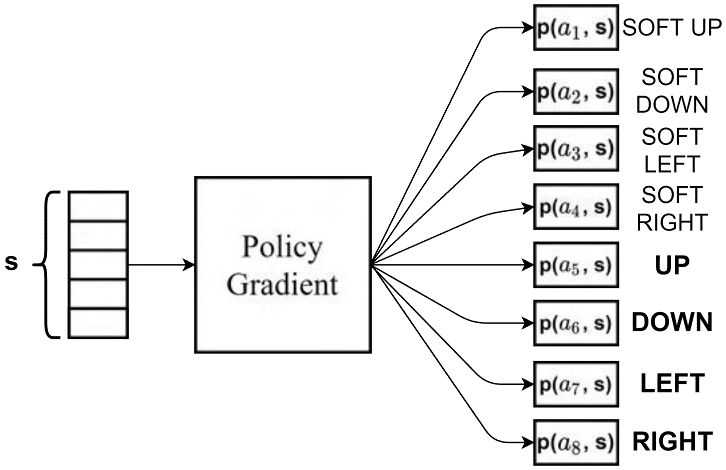


Figure 5.15: Policy Gradient implementation

Exploration & Exploitation

In contrast to QL/DQN, PG trains a stochastic policy, where the actions are sampled based on the probability associated with selecting any action ($p(s|a)$). The amount of randomness in action selection depends on the probability distribution over the action space. If the probability distribution favors one particular action (high $p(a|s)$ value) in a given state, the chances of exploration decreases in that state. On the other hand, if two actions have comparable conditional probability (as seen in Figure 5.15), this will imply that the agent is not very confident in its action selection and should explore between the two high probability actions. This is in contrast with QL, where only the action associated with the highest probability was chosen and the rest were treated the same and explored based on the value of the parameter 'ϵ'.

An example of the same is shown in Figure 5.16 (a), where the Q-Learning algorithm will always choose the action associated with label two (2) based on the greedy policy. On the other hand, A2C will sample all the actions based on the probability associated with each action in Figure 5.16 (b).

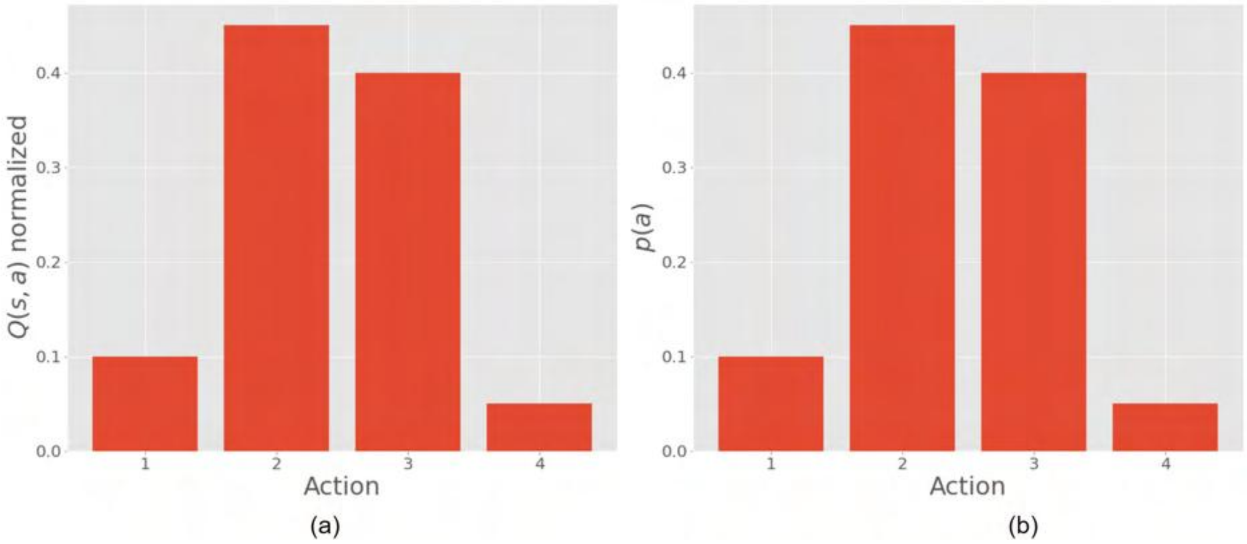


Figure 5.16: (a) Greedy QL policy always chooses action:2, (b) Probabilistic policy with A2C samples all actions

As such action labels 2 & 3, having comparatively high conditional probability will be sampled more as the optimal action in the given state (s). Additionally, there will still be some instances where the rest of the actions (1 & 4) will be sampled. Overall, the probability of occurrence of any of the possibilities from the action space is not zero as was the case with QL.

5.5.1 Improvement to Policy Gradient

Unlike Q-learning, the PG algorithm is an on-policy algorithm, which means it learns only using state-action transitions made by the current active policy. Technically, this means there is no ‘Experience Replay’ memory as in DQN. Once the model is trained, its parameters (θ) change, and therefore its policy. This means that all the experience collected before this training must be discarded, and cannot be used for training at any later stage. So every piece of data collected for training is used only once.

Actor Critic

For PG, agents were designed to learn a policy directly and are trained based on a performance metric, which is to be maximized. As such the chances of selecting an action with a positive impact on the performance further increases. However, the performance metric, in this case, is the total reward expected in an episode. Getting the true value of the return (G_t) requires waiting for the episode to end, delaying the updates to the weights. This can be computationally expensive. In Section 4.5.3, similar issues led to discarding the MC algorithms. To overcome this, it will be better to merge the two techniques of RL that are discussed in this chapter:

1. The agent learns the policy directly based on the Policy Gradient (PG) method.
2. At the same time, learning the state- or action-specific knowledge through the value functions ($Q(s, a)$ or $V(s)$).

The combined method is known as “Actor-Critic” (AC), and is made of two functions/networks learning together: one learns the policy which determines the behaviour /action of the agent (and is therefore known as the Actor), and the other network learns the Q-Value of each state and action pair (and is known as the Critic), where Q-value is also a learned parameter/network. This means that the whole system is divided into two parts or two sub-agents, which yields a more complex network as shown in Figure 5.17.

A policy $\pi_\theta(a|s)$, defined as the probability of an agent taking action ‘a’ in any state ‘s’, is expressed as

$$\pi_\theta(a|s) = \frac{e^{f(s,a,\theta)}}{\sum_{a_i} e^{f(s,a_i,\theta)}} \tag{5.14}$$

where the policy (π) has θ as its parameters and gives a probability distribution for selecting an action ‘a’ over all the possible actions (a_i) conditioned on the current state (s). ‘f’ is the

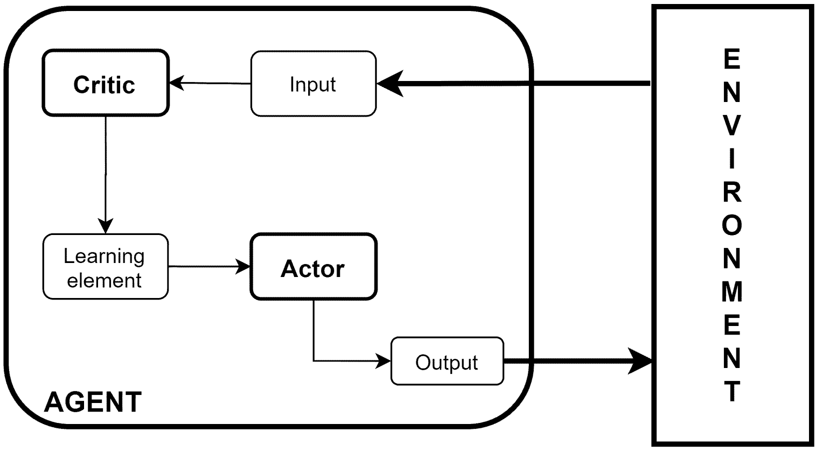


Figure 5.17: Generic AC

output of the function approximator. This approach of policy parametrization is called softmax (Goodfellow, Bengio, and Courville 2016) in action preferences. As mentioned earlier, the function 'f' can be a linear combination of features or a deep neural network, as long as the policy ($\pi_\theta(a|s)$) is differentiable with respect to its parameters.

The update rule mentioned in eqn. 5.13 is modified to be:

$$\theta_{t+1} = \theta_t + \alpha\Big(Q(s_t, a_t) \times \nabla_\theta\Big[\ln \pi(a_t|s_t;\theta_t)\Big]\Big) \tag{5.15}$$

where $\nabla\pi_{\theta_t}(a^*|s)$ is the direction in which to move θ_t so as to increase the estimate of $\pi_{\theta_t}(a^*|s)$ the fastest and thus increase the probability of selecting a^* more often in states similar to s.

In the above setting, just like the PG method, AC is also an on-policy model, which means that after each training all previous data is discarded.

Advantage Actor Critic

A further extension to AC algorithms is to replace the Q-value (in eqn. 5.15) with another performance metric called 'Advantage'. This is because an absolute value of $Q(s,a)$ is extremely dependent on the reward function and can be noisy during the initial stage. Further, an incorrect initialization of $Q(s,a)$ values should not have a significant effect on the update rule as a very high and noisy value of $Q(s,a)$ can disturb the update rule in a significant way. For this, it is important to quantify the value of the Q-function and a comparison with the other actions is a far better representative of how good certain action is. This is represented as 'advantage', i.e. the advantage of taking a particular action 'a', in comparison to other actions $a \in A$. So we can replace the Q-Value in the Actor-Critic update rule (eqn. 5.15) with the 'advantage' of an action $(A(s,a))$, which is defined by subtracting $V(s)$ from $Q(s,a)$. This function gives an indication to the agent as to how much better or worse taking action 'a' in a state 's' is compared to acting according to the policy (Mnih et al. 2016).

$$A(s,a) = Q(s,a) - V(s) \tag{5.16}$$

where $V(s)$ is the state's Value function (refer to Section 4.5.2). An update to the weights (θ) with $A(s,a)$ is comparatively much more useful for the gradient calculation than $Q(s,a)$, as a positive $A(s,a)$ is good (an improvement over the current policy), and a negative $A(s,a)$ is relatively bad as compared to the action suggested by the current policy. This developed form of the Actor-Critic model is known as the Advantage Actor-Critic, which is abbreviated to A2C (Mnih et al. 2016), as shown in Figure 5.18.

Based on eqn. 5.16, it might seem that the A2C version seems to make learning a little more complex, as the learning system needs to learn both $Q(s,a)$ and $V(s)$. However, that is not needed since a Q-Value is the reward received from state (s) and action (a) and then continue following the greedy policy till the end of the episode and can be written as:

$$Q(s,a) = r(s,a) + \gamma V(s') \tag{5.17}$$

which means that a Q-value is the sum of the immediate reward and the state value (V) of the next state $(s^{'})$. This results in the following equation for the 'Advantage' function $(A(s,a))$:

$$A(s,a) = \Big(r(s,a) + \gamma V(s')\Big) - V(s) \tag{5.18}$$

This means that an agent needs to learn only the State-Value Function, and use it twice - Once for the present state (s) and then for the next state $(s^{'})$. This little tweak makes the A2C easier to implement than the original Actor-Critic.

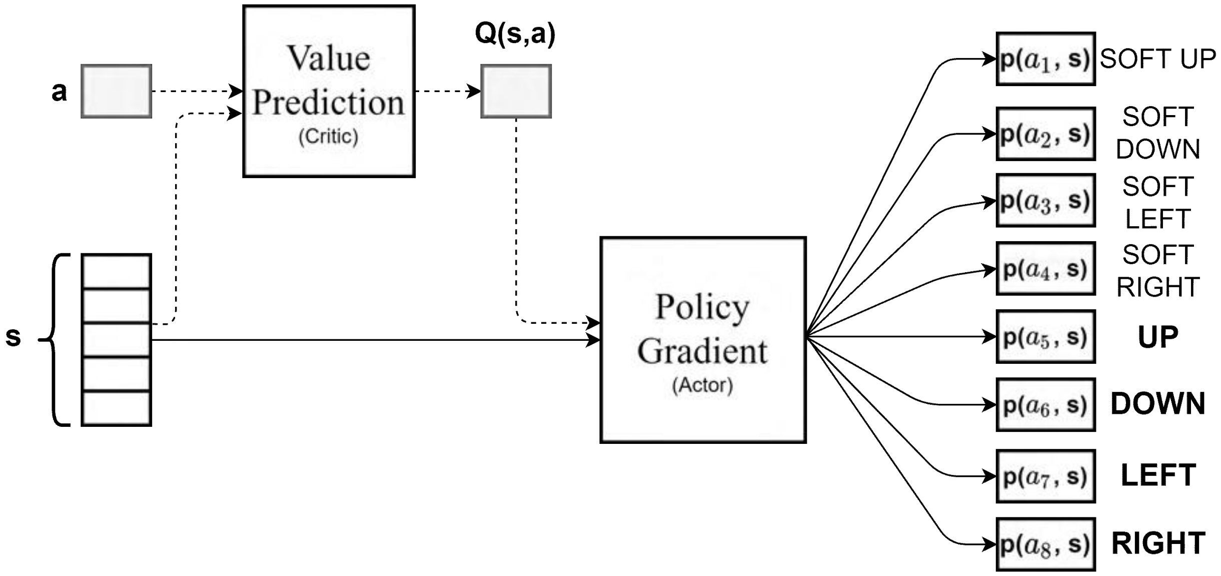


Figure 5.18: Advantage Actor Critic

5.6 Experiment Setup - A2C

As shown in Figure 5.19, the network architecture takes in the state representation and outputs the probability distribution over the possible eight actions available to the agent at any given time (based on *SeqTO-v2*). A few cases with varying coil turns and coil current are considered. For testing the generalizability of the A2C algorithm, it is trained on only three out of the possible eight excitation scenarios for a C-core, as listed in Table 5.4.

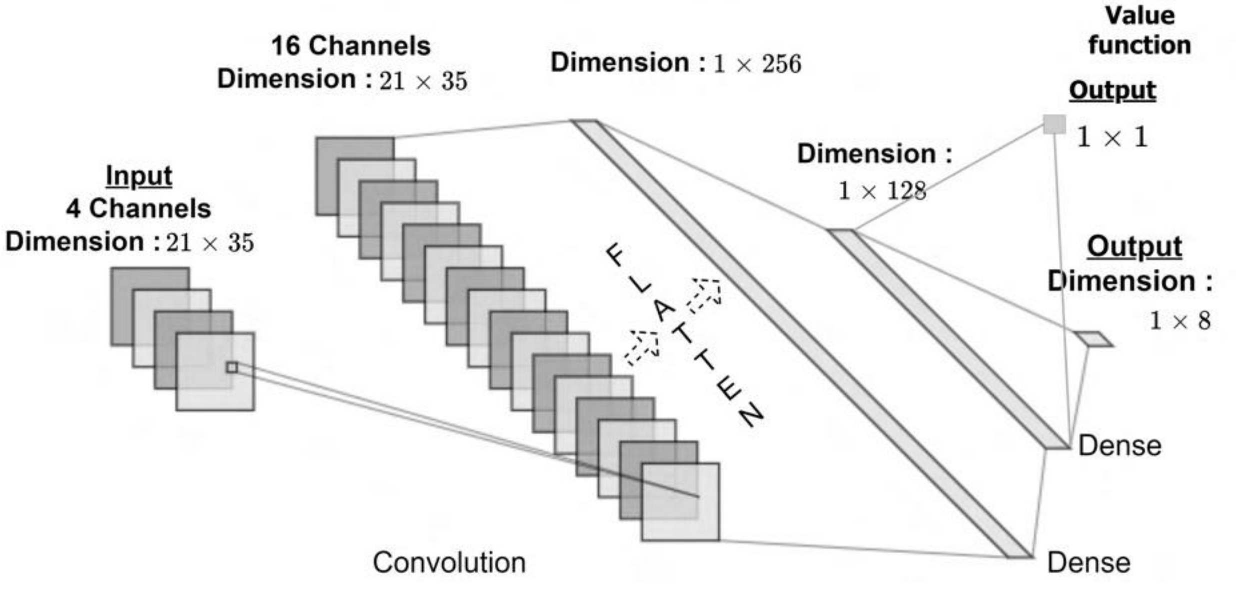


Figure 5.19: Network Architecture for A2C

Type	Scenario	Coil current (A)	No. of turns (T)
TRAIN	1.	1.0	250
	2.	1.0	500
	3.	1.5	500
TEST	4.	0.5	250
	5.	0.75	500
	6.	1.25	500
	7.	1.75	500
	8.	2.0	500

Table 5.4: Different excitation scenarios used for TO of a C-core actuator.

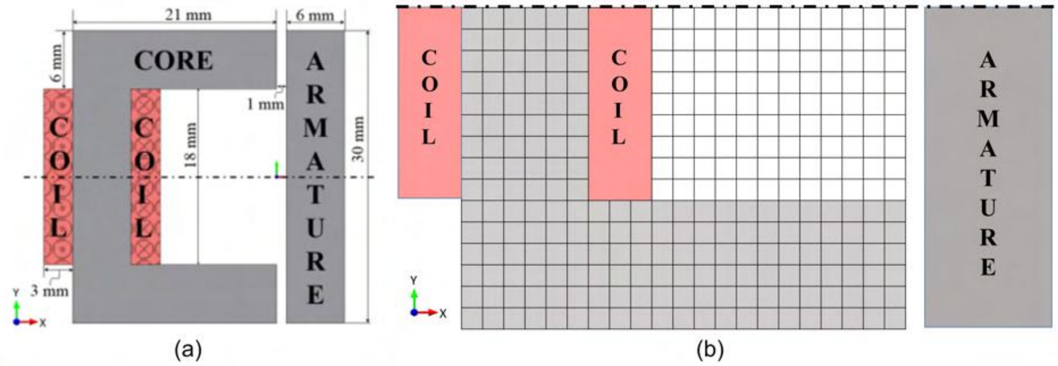


Figure 5.20: (a) Conventional C-core design (b) Only half of the geometry is simulated due to symmetricity.

The scenarios where the agent interacts, collects, and train on data originating from the scenario, is labeled as '*Train*'. The rest of the scenarios are used to check the 'generalizability' of the agent and are referred to as the '*Test*' scenarios. After every few iterations of training, the agent is tested on all the scenarios listed in Table 5.4 to check the performance and monitor if the agent converged to a solution. The optimal design obtained from A2C is compared with that obtained using GA (Park and Min 2009b; Midha et al. 2019). The GA algorithm operates on the On-Off TO methodology. Further, a tabular Q-Learning method is also employed with SeqTO-v1 (pointer size 1×1) to verify the results. The comparison is performed on the value of the performance parameter (force on the armature in C-core) of the optimal design obtained through each method. The value of the torque is also compared with the conventional design of a C-core. The geometry shown is Figure 5.20 is considered as the conventional geometry of the C-core for this study. In addition, a comparison of the number of simulation calls made for each optimization process is also tabulated.

The network used for Actor & Critic is shown in Figure 5.19. The network starts with a convolutional layer to capture the spatial relationship of the material distribution in the design domain and the corresponding material distribution for the current and past 4 states. This is followed by two dense layers and an output layer which correspond to the $Q(s, a)$ for the critic and $\pi(s, a)$ for the Actor network. Apart from the output layer, which is dense (used for regression) for Critic and Softmax (used for distinct classes) for the Actor, the rest of the network architecture and functionality remains the same and can thus be merged to form a single input multi-output network as can be seen in Figure 5.19.

After the training, it is observed that the A2C based agent managed to produce a superior performance on the excitation cases which are part of '*training*' scenarios as compared to the performance on the '*test*' scenarios. However, the performance over the 'test' cases can be further improved by using the technique of transfer learning (TL). Some of the same concepts of TL as discussed in Chapter 3 and (Khan et al. 2020b) can be used here. During the fine-tuning/ re-training, the learning rate is reduced but the network architecture is not altered as the design domain and the MDP encoding was kept the same.

5.7 Results & Discussion

The results obtained over the scenario cases for different excitation for the C-core are tabulated in Table 5.5. This shows a comparison over computational efficiency and optimal accuracy of all RL based algorithms using Sequence-based TO (-v1 & -v2) with that obtained from GA (Park and Min 2009b) and GA + filters (Midha et al. 2019) using ON/OFF TO. The performance improvement can be observed in all five cases, where the previous results from Midha et al. 2019 were available. Some additional scenarios are analyzed for which results from the original work (Midha 2018) were not available, specifically (*1.25 A, 500 Turns*), (*1.75 A, 500 Turns*) & *2.0 A, 500 Turns*). This is performed as part of the inference check of the neural network-based A2C algorithms.

To keep track for convergence, the A2C algorithm is tested at a regular interval during the training phase to check the overall performance of the agent. Along with the value of force for optimal geometry, the number of FE simulations performed in each optimization run is also recorded for comparison and is included in Table 5.5.

For each excitation scenario, the GA and QL results are independent runs where the optimization is performed from scratch. On the other hand, a single A2C based agent is employed to learn (train) from the experience generated on the three train scenarios and generalize to all the other five 'test' scenarios. In such a case, the FE calls for GA and QL are to be summed up if one needs optimal geometry for all the eight scenarios mentioned, since all the runs are independent. In contrast, for A2C, the number of simulations associated with each excitation scenario is simply a milestone during a single optimization/training session.

	Scenario	**Conven -tional**	**GA On/Off** (Park et al.)	**+Filter** (Midha et al.)	**QL SeqTO** (1×1)	**A2C SeqTO -v2**	**A2C (Re-train)**
TRAIN	1.0 A, 250T	9.97	16.80 (200k)	15.90	**19.02** (125k)	16.6 (190k)	-
	1.0A, 500T	38.70	50.90 (210k)	50.60	**54.0** (128k)	51.50 (150k)	-
	1.5A, 500T	78.40	82.10 (195k)	83.30	82.20 (105k)	**85.70** (125k)	-
TEST	0.5A, 250T	2.52	4.56 (159k)	3.88	**4.93** (94k)	4.75 (175k)	4.91 (3500)
	0.75A, 500T	22.20	32.50 (160k)	30.80	**34.80** (120k)	32.70 (175k)	34.60 (7500)
	1.25A, 500T	58.4	-	-	71.30 (108k)	71.60 (150)	72.1, **72.3** (7500)
	1.75A, 500T	93.6	-	-	**97.20** (105k)	96.8, 93.6 (135k)	96.8 (2500)
	2.0A, 500T	105	-	-	**106** (115k)	105 (110k)	**106** (2750)
Total Simulations			(925k)	-	(900k)	(192k)	(24k)
Avg. Simulations/Scenario			(185k)	-	(112k)	(24k)	(4,750)

Table 5.5: Comparison of Force magnitude wrt GA optimized results (Park and Min 2009b), GA+filters (Midha et al. 2019), Tabular QL (using SeqTO-v1) optimized results and with the A2C (RL based; using SeqTO-v2) agent. The number in the brackets in each cell corresponds to the number of FE calls during the particular optimization process.

As such, the number of FE simulations associated with A2C agent in Table 5.5 are not to be summed up, since the number represents the simulations needed for the A2C algorithm to converge to the optimal geometry for the corresponding 'train' or 'test' excitation scenarios.

It is observed that GA requires the most number of FE calls out of all the algorithms tested in the study. Further, the force on armature value of the optimal geometry identified by GA is inferior to that obtained through other algorithms. The possible cause for this is explored later. In terms of FE calls, QL required the fewest function evaluation for seven out of the eight scenarios, when considered independently. This can be attributed to the binding nature of the SeqTO-v1 methodology and the deterministic nature of the QL algorithm. Finally, if the total function evaluations for all the scenarios is to be compared, A2C out performs both QL and GA, due to the nature of SeqTO-v2 methodology and the ability of the function approximator (NN) to extract useful patterns and generalize over similar scenarios. The number of FE simulations required by QL for all (eight) scenarios sums up to 900k. In comparison, A2C requires only a maximum of $190,000$ FE calls to converge to an optimal solution for the three train cases. On average, QL requires about $112,000$ simulations for an excitation scenario. In comparison, the average number of simulations needed for A2C is only $24,000$ FE simulations for an excitation scenario of a C-core. It can be seen that A2C achieves close to optimal geometries for all the scenarios. This is compensated by the significantly fewer number of simulations needed to achieve this solution design. In cases, where only the optimal solution is to be desired, the trained agent can be used as a source domain to further fine-tune an agent for a specific scenario. The resulting design obtained after fine-tuning achieves the same force magnitude as the optimal ones, using only a small amount of computational resources.

The inability of the agent to reach the optimal structure can partially be explained due to the inflexibility of the SeqTO-v2 methodology, where the methodology does not support placing a thin strip close to the air gap, that the QL based agent can achieve with SeqTO-v1 (1×1) environment. However, the number of FE calls needed to use SeqTO-v1 (1×1) will be simply too much to handle as function approximators such as neural networks will require far more training data compared to a deterministic tabular look-up table.

A qualitative discussion for some of the scenarios is presented below, where a comparison of the optimal geometry along with the plot of the magnitude of magnetic flux density is also shown.

C-core Excitation Scenario 1

$\textit{Coil Current} = 1.0A$ *&* $\textit{Coil Turns} = 250$

This excitation scenario is part of the '*train*' cases, where the agent interacted with the environment operating with the excitation settings of *Coil Current = 1.0 A, Coil Turns = 250*, along with two other '*train*' scenarios (*1.0 A, 500 Turns* & *1.5 A, 500 Turns*). The force magnitude generated by the optimal design on the armature, along with the magnetic flux density distribution obtained from the GA, QL, and A2C is shown in Figure 5.21.

The sequence of actions corresponding to the optimal geometry obtained for this excitation using the A2C agent is shown in Figure 5.23. The optimal geometry obtained with a $1 \ \times 1$ controller based on QL generates a higher force ($19.02 \ N$) than the one obtained through an A2C controller ($16.60 \ N$). As mentioned earlier, the reason can be attributed to the ability of a $1 \ \times 1$ controller to place a thin strip of iron close to the air gap, whereas SeqTO-*v2* limits such maneuverability. Since the pointer of the controller cannot move out of the design space, the smallest width of material block is limited to 2 near the boundary of the design space, in the SeqTO-*v2* environment. This can be seen from Step: 12 to 15 in Figure 5.23.

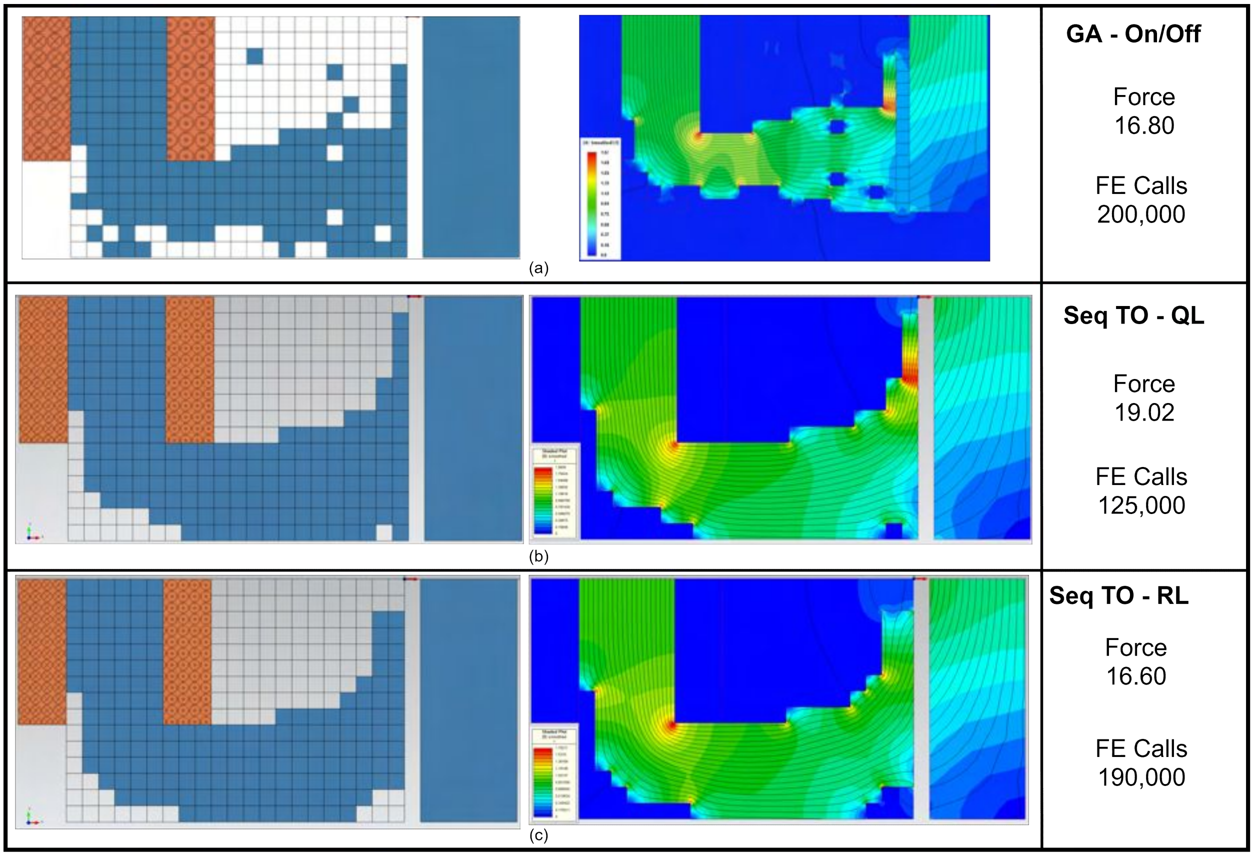


Figure 5.21: Optimal geometry and B distribution for excitation scenario (1.0 A, 250 Turns) for (a) GA (b) QL based on SeqTO-v1 (c) A2C based on SeqTO-v2.

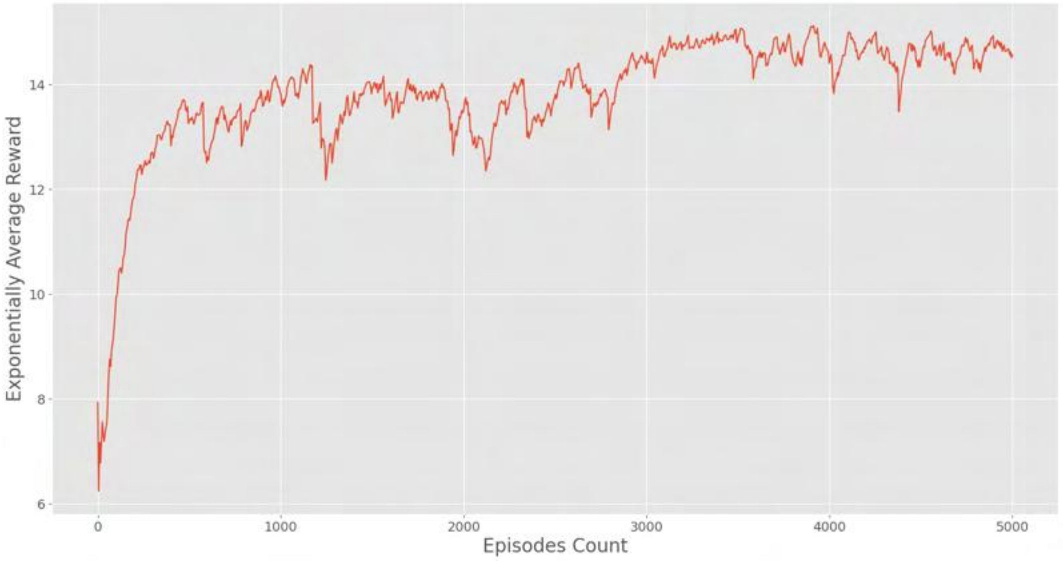


Figure 5.22: Convergence results for 1.0 Amps, 250 Turns

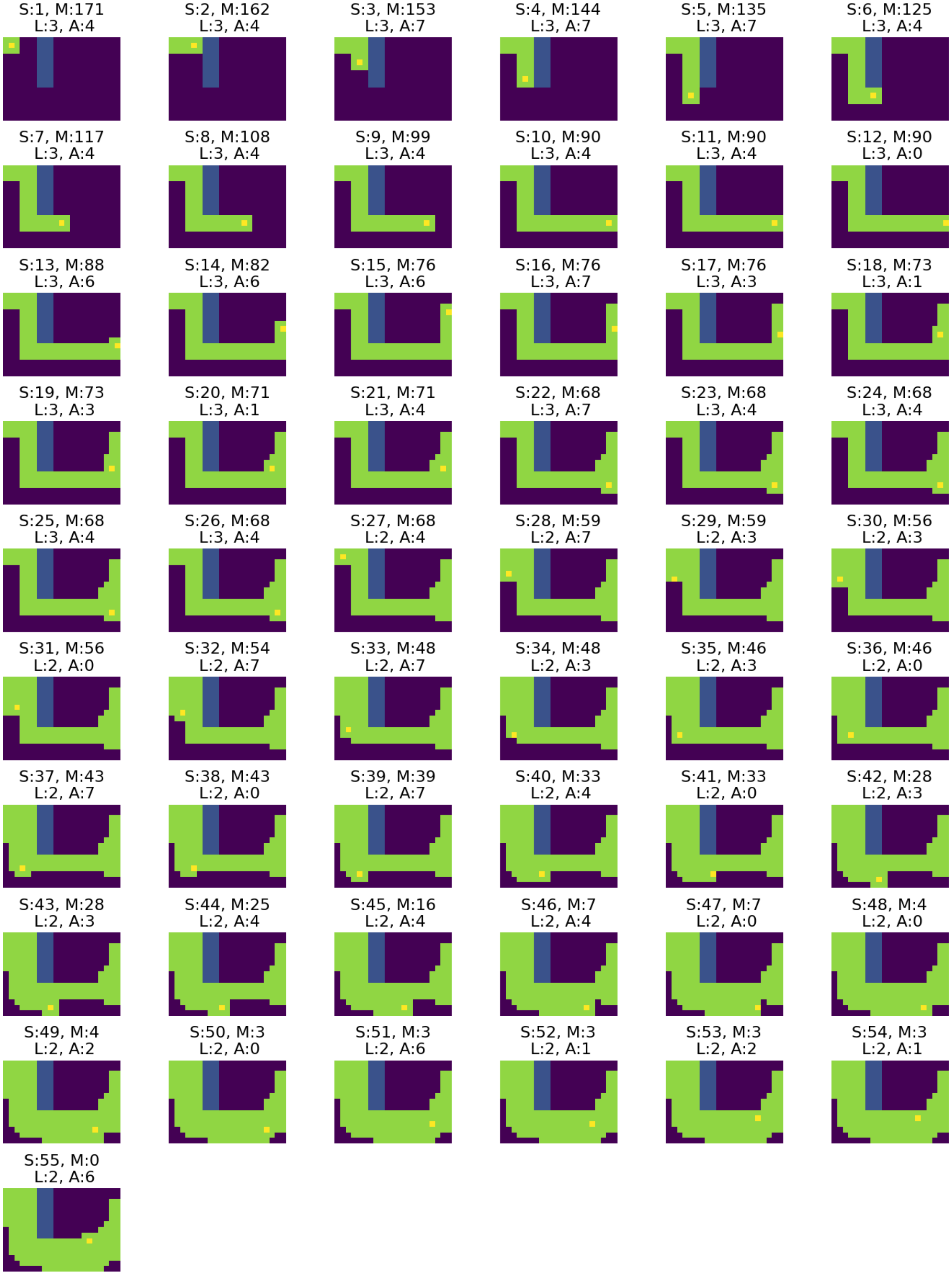


Figure 5.23: Sequence for optimal geometry for excitation: 1.0 A, 250 Turns. Only the design domain is shown. S: Step No., M: Material left, L: Lives left.

C-core Excitation Scenario 2

Coil Current $= 0.75A$ *& Coil Turns* $= 500$

This scenario is part of the '*test*' excitation cases i.e., interaction without learning, where the A2C agent did not learn from an environment operating this excitation scenario during the training phase. The GA algorithm and the QL-based agent were directly optimized using the '*test*' case excitation values and results in the optimal design solution.

Although this '*test*' case was not part of the training for A2C, the agent still managed to obtain a close to optimal (32.70 N; 6% lower) performance when compared to QL (34.80 N) and an improvement over the result obtained using GA (based on On/Off TO) (32.50 N). The agent's performance improves on further tuning of the agent. This re-training is performed as a post-processing step, where the agent is exposed to the excitation scenario (*Coil Current = 0.75 A & Coil Turns = 500*) and it learns from this experience, i.e., interaction with learning. After this re-training, the agent managed to attain a force magnitude (34.60 N) value similar to the QL-based SeqTO-v1 (1 × 1). Again, the main difference here is the granularity level that can be achieved by the SeqTO-v2 as compared to SeqTO-v1 as the QL managed to achieve the same results with less material and higher effective FE Calls. The optimal results from GA, QL, A2C and A2C(re-train) is shown in Figure 5.24 (a), (b), (c) (d) respectively, along with their respective magnetic flux density distribution.

The transitional change in the design space due to interaction between the agent and the environment for the excitation scenario (*Coil Current = 0.75 A, Coil Turns = 500*) is shown in Figure 5.25. Similar agent behaviour as seen earlier for excitation scenario of *1.0 A, 250 Turns* in Section 5.7, as can be seen in Figure 5.25. The ability to obtain a close to optimal geometry (only 6% lower compared to QL) without ever encountering the excitation condition during the training phase, shows the ability of a function approximator-based A2C agent to generalize to the unseen but relevant input. Further, the number of FEA evaluations needed for re-tuning of the agent for this excitation scenario stands at 7500. After this re-tuning, the force magnitude obtained from the optimal design reaches 34.60 N (comparable to QL with 34.80 N). This results in significant computational cost reduction as compared to other learning methodologies like QL $(120,000)$ and GA $(160,000)$ for obtaining similar optimal designs, without a compromise on the performance. An observation that highlights the capability of the agent's maneuverability can be seen in Figure 5.25 (step: 13), where the agent uses the boundary of the design domain to place 2 elements and follows up with 4 elements each in the next couple of steps (step: 14 -15). This is possible as the pointer of the controller stays inside the design domain but part of the 3 × 3 block gets trimmed out as it falls out of the design space. The agent uses this capability to save some material. Later in the episode, it wastes four steps (Step: 23 to 26) trying to move out of the design

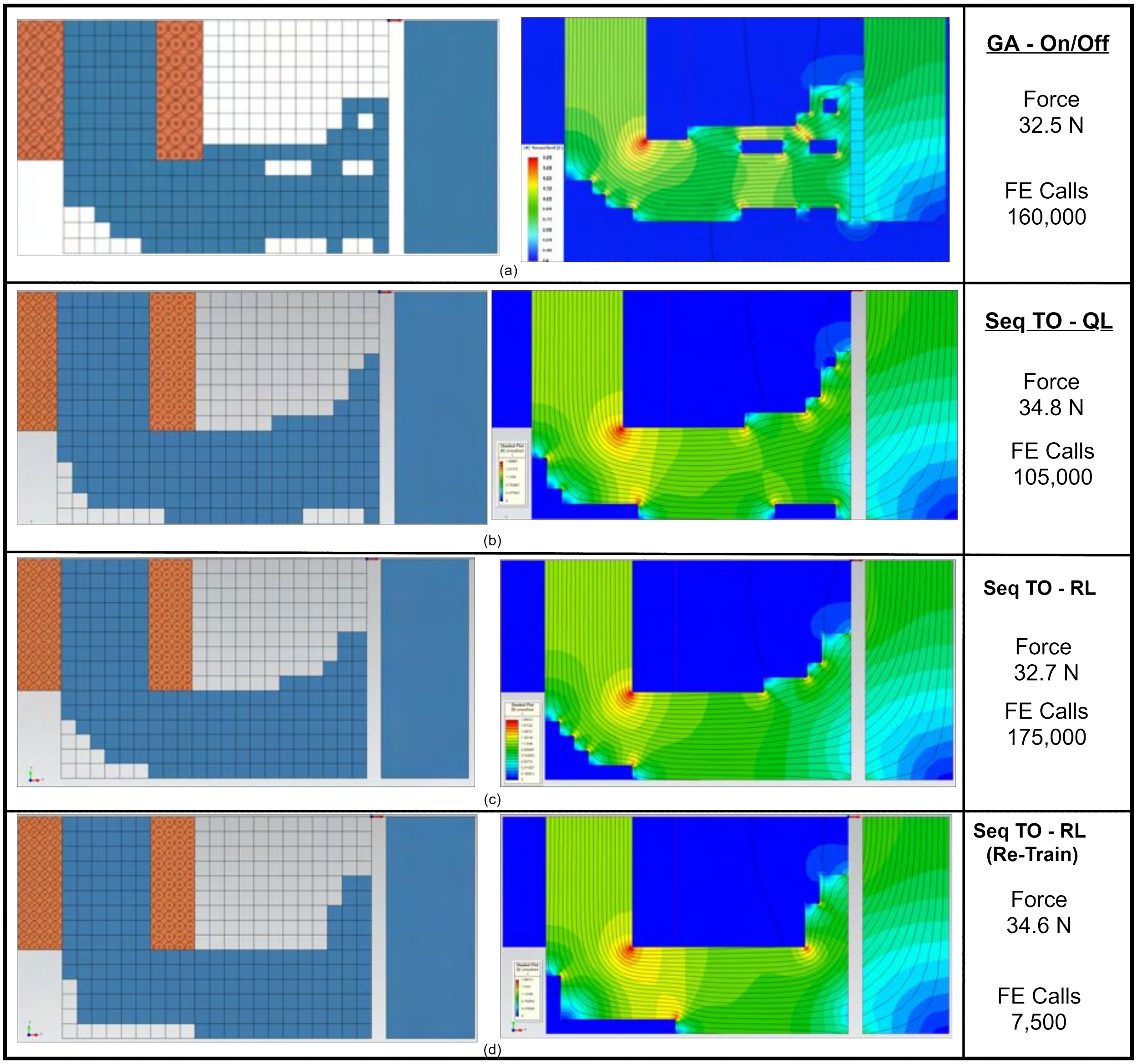


Figure 5.24: Optimal geometry and B distribution for excitation scenario (0.75 A, 500 Turns) for (a) GA (b) QL based on SeqTO-v1 (c) A2C based on SeqTO-v2 (d) fine-tuned A2C agent.

domain and loses a life (L: 3 $\rightarrow$ 2). This places the controller back at the origin and now the agent adds magnetic material where the design is severely affected by magnetic saturation. The agent learns to exploit the MDP dynamics where it receives a penalty for moving out of the domain, but the trade-off is being able to move back to the origin and deal with the saturation in the region of material placed near the coil.

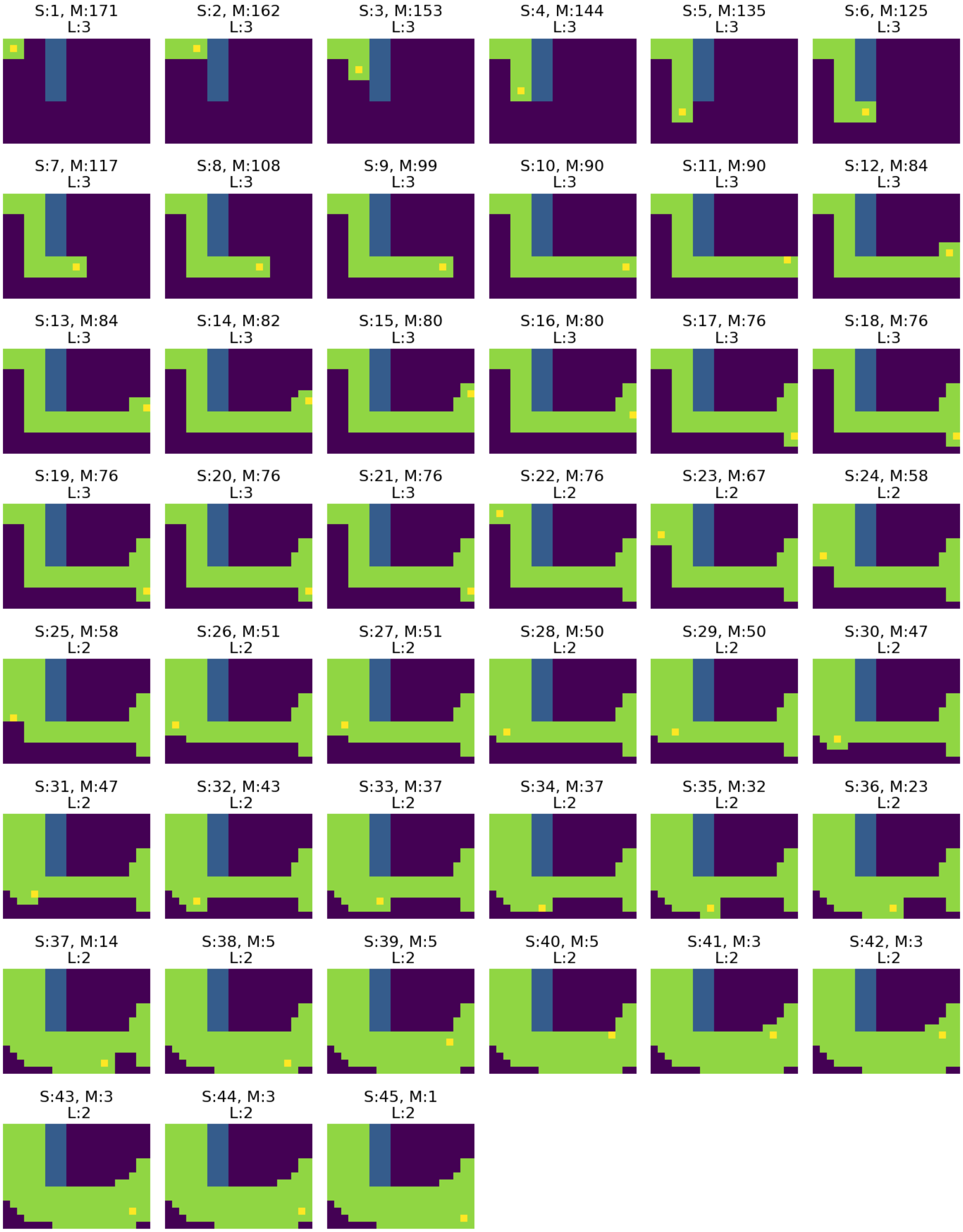


Figure 5.25: Sequence for optimal geometry for excitation: 0.75 A, 500 Turns. S: Step No., M: Material left, L: Lives left.

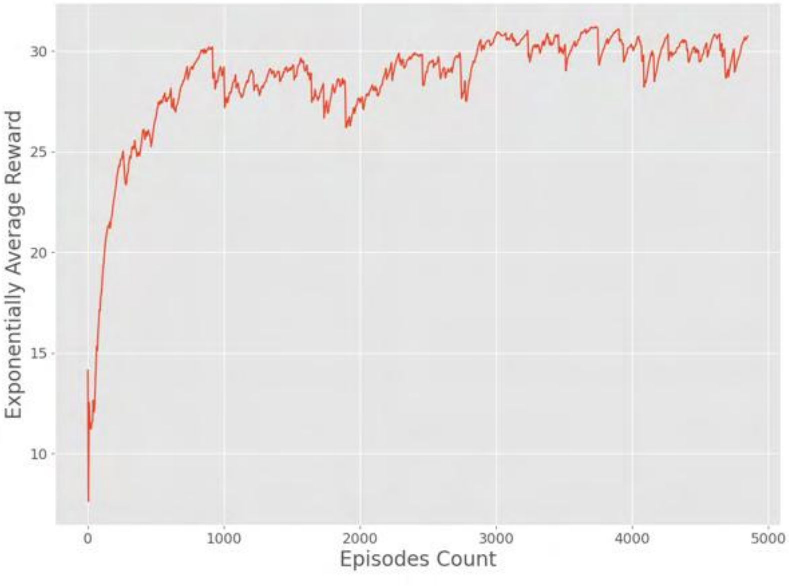


Figure 5.26: Convergence results for 0.75 Amps, 500 Turns.

C-core Excitation Scenario 3

Coil Current = 1.5A
Coil Turns = 500

For *Coil current* = 1.5 *A & Coil Turns* = 500, it is observed that all the three algorithms ended up with the core design that resembles the conventional C-core design geometry. Significantly, for cases with an excitation of *500 Amp-Turns* and beyond, the effect of saturation is dominant, and based on the information present in the flux distribution and the excitation level, the agent can form a policy to tackle the phenomenon of material non-linearity.

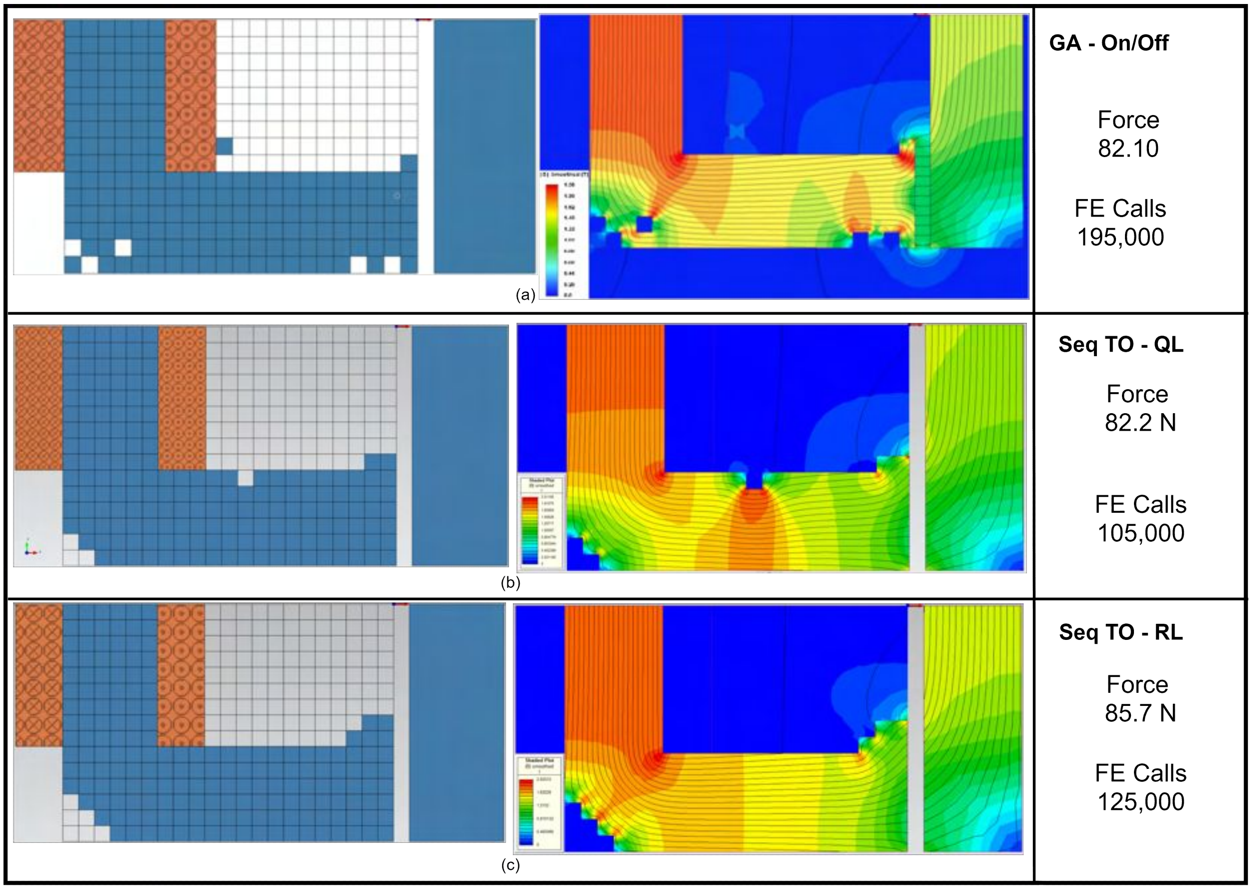


Figure 5.27: Optimal geometry and B distribution for excitation scenario (1.5 A, 500 Turns) for (a) GA (b) QL based on SeqTO-v1 (c) A2C based on SeqTO-v2.

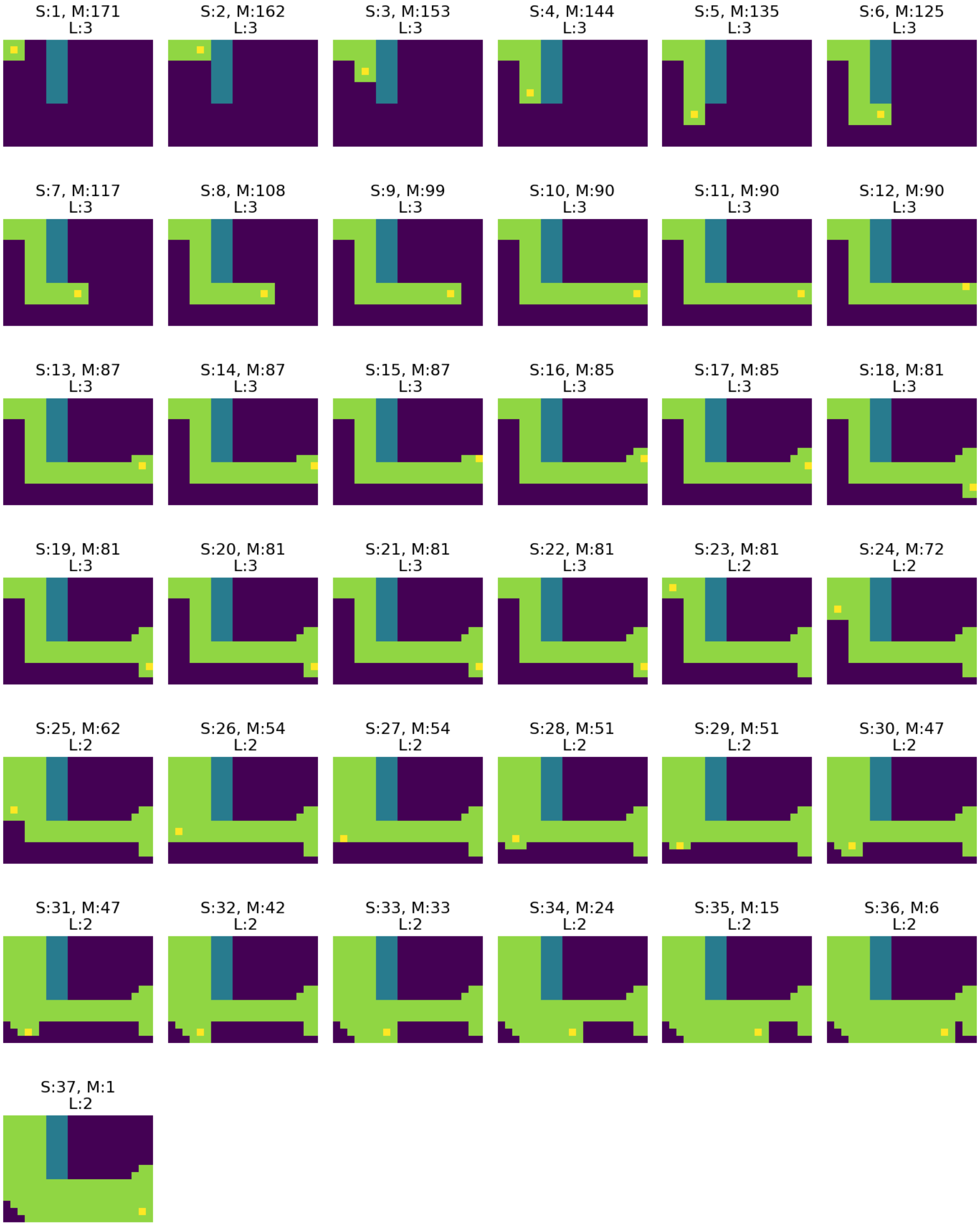


Figure 5.28: Sequence for optimal geometry for excitation: 1.5 A, 500 Turns. S: Step No., M: Material left, L: Lives left.

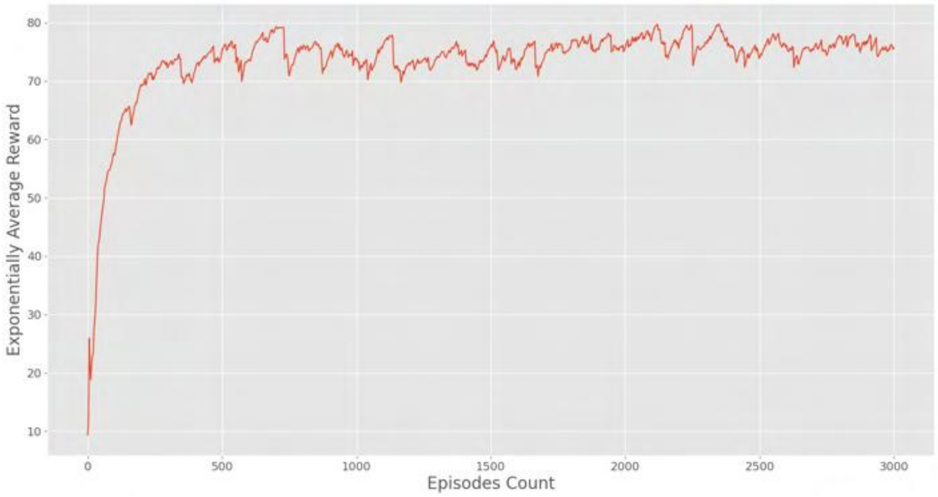


Figure 5.29: Convergence results for 1.5 Amps, 500 Turns.

Other C-core excitation scenarios

The analysis of other scenarios is similar to the three scenarios discussed earlier in this section. For excitation scenarios of (*1.75 A & 500 Turns*) and (*2.0 A & 500 Turns*), there is not much margin for topological change as compared to the conventional geometry of the C-core (shown in Figure 5.20). For both these scenarios, the conventional geometry is one of the solutions provided by the A2C algorithm, as can be seen in Figure 5.31 (b) and 5.30 (b) for 1.75 A and 2.0 A respectively. Based on the agent's action, it can be concluded that the optimal design does not deviate much from the conventional geometry for higher values of excitation (750 Amp-Turns onwards). This is in agreement with the optimal solutions obtained using other optimization algorithms (GA and QL).

The excitation scenario: *1.25 A & 500 Turns* shows resemblance to *1.5 A & 500 Turns*. A comparison of optimal geometries from GA (On/Off), Tabular QL (SeqTO-*v1*) and A2C (SeqTO-*v2*), is shown in Figure 5.32. In contrast, the agent shows a significant change in topology for lower values of excitation as can be seen for the scenario: *0.25A & 500 Turns* in Figure 5.33. The reason being that the magnetic material placed near the coils is no longer easily saturated and the agent can aggressively place more material near the airgap facing the armature, forcing more flux out laterally.

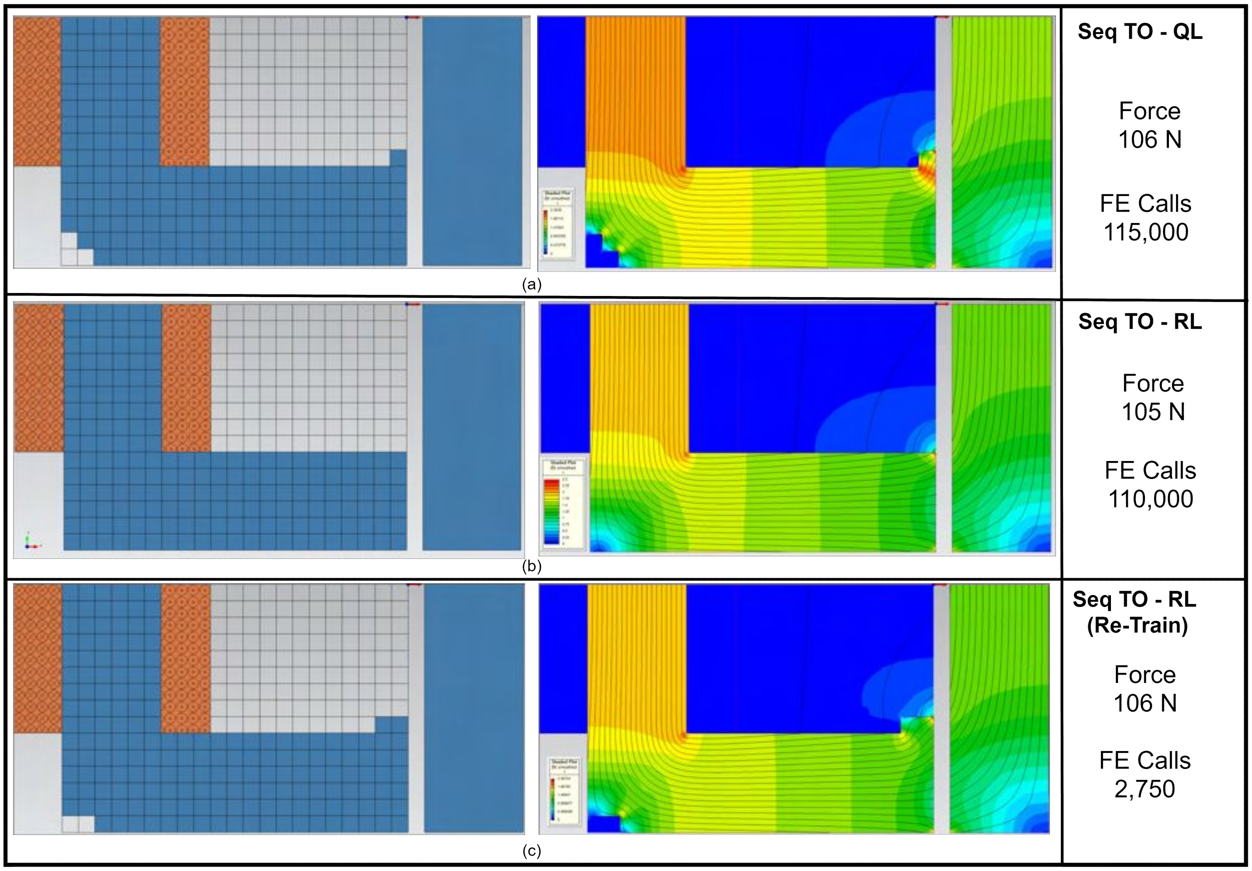


Figure 5.30: Optimal geometry and B distribution for excitation scenario (2.0 A, 500 Turns) for (a) GA (b) QL based on SeqTO-v1 (c) A2C based on SeqTO-v2.

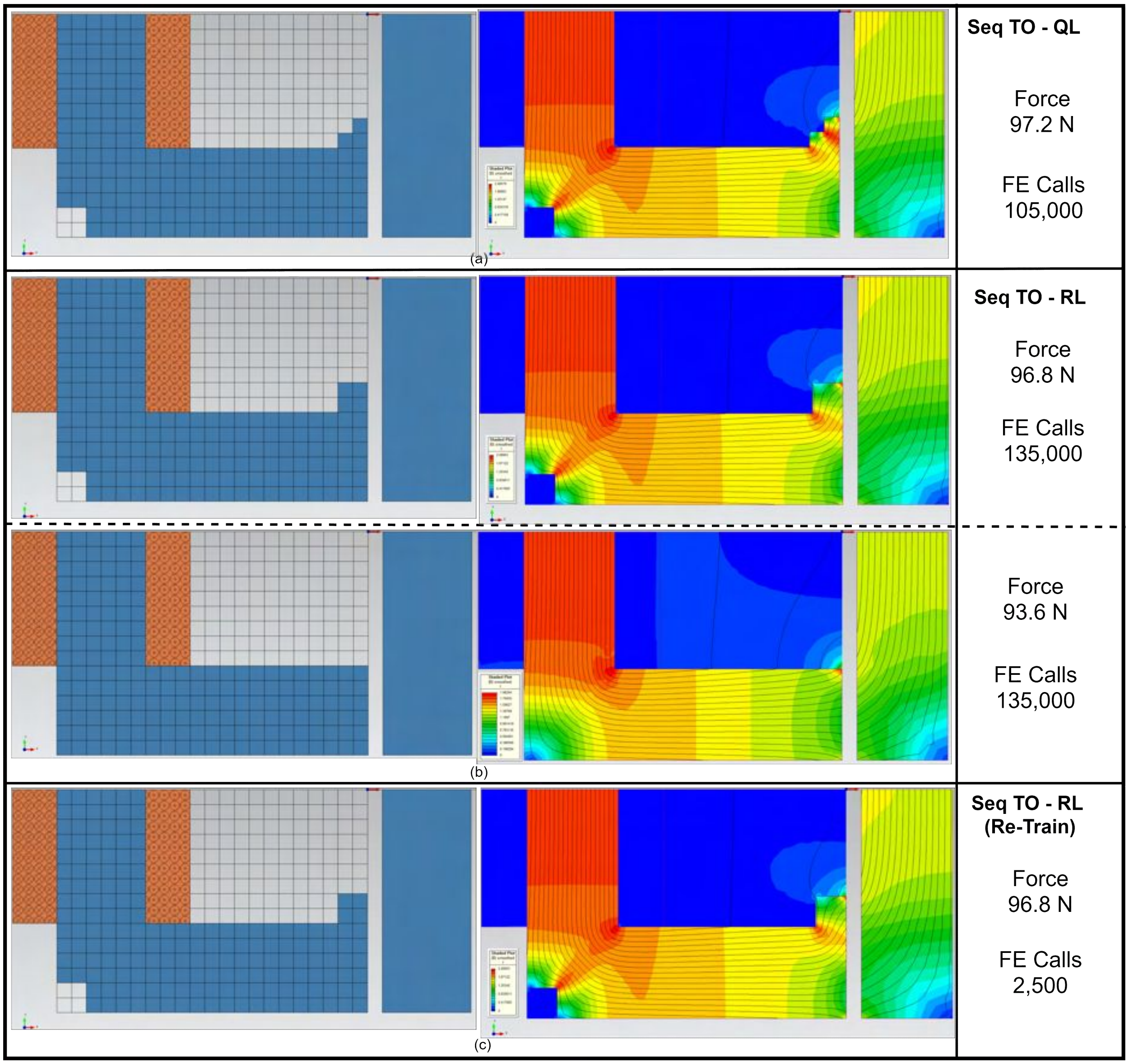


Figure 5.31: Optimal geometry and B distribution for excitation scenario (1.75 A, 500 Turns) for (a) GA (b) QL based on SeqTO-v1 (c) A2C based on SeqTO-v2.

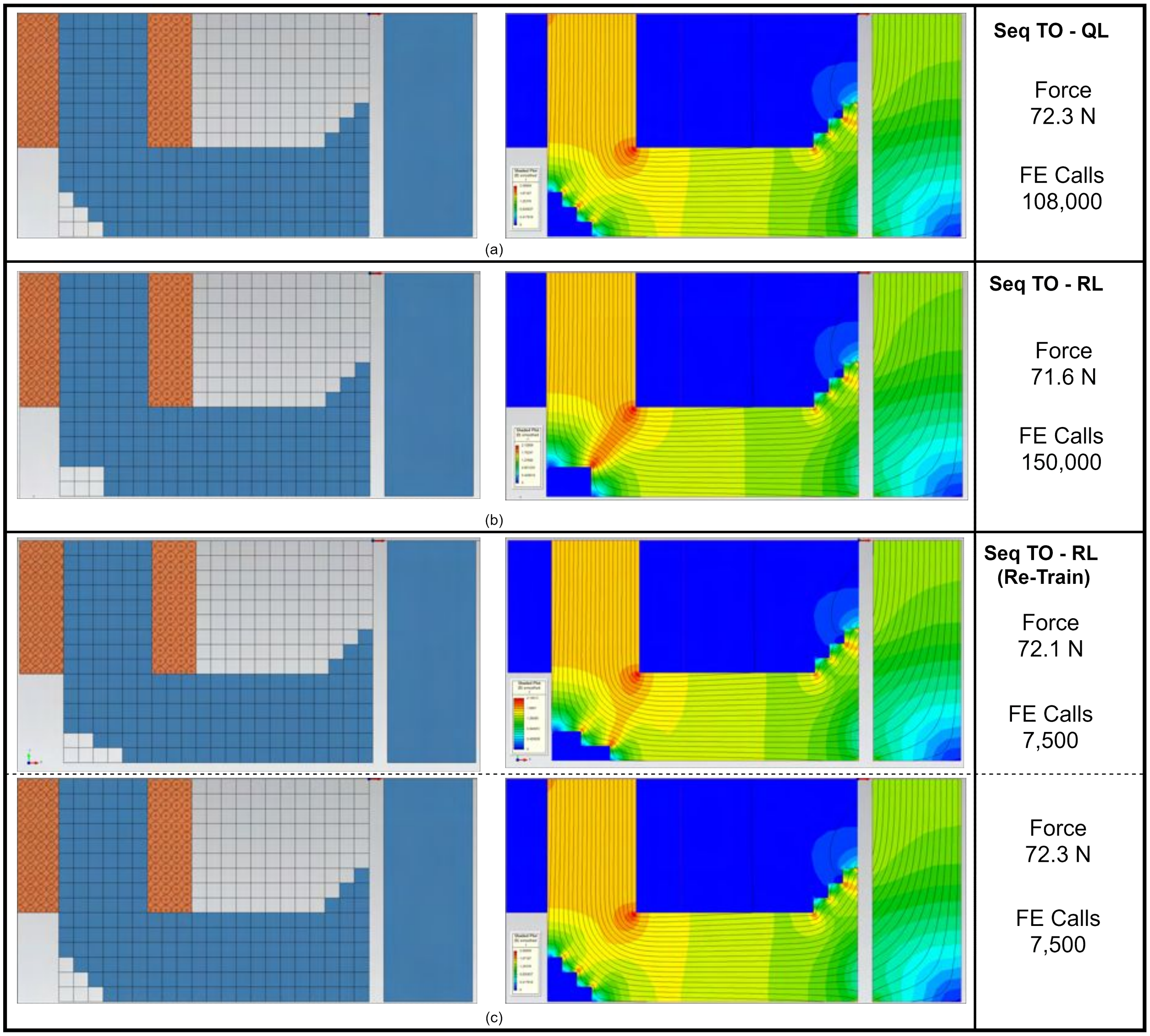


Figure 5.32: Optimal geometry and B distribution for excitation scenario (1.25 A, 500 Turns) for (a) GA (b) QL based on SeqTO-v1 (c) A2C based on SeqTO-v2.

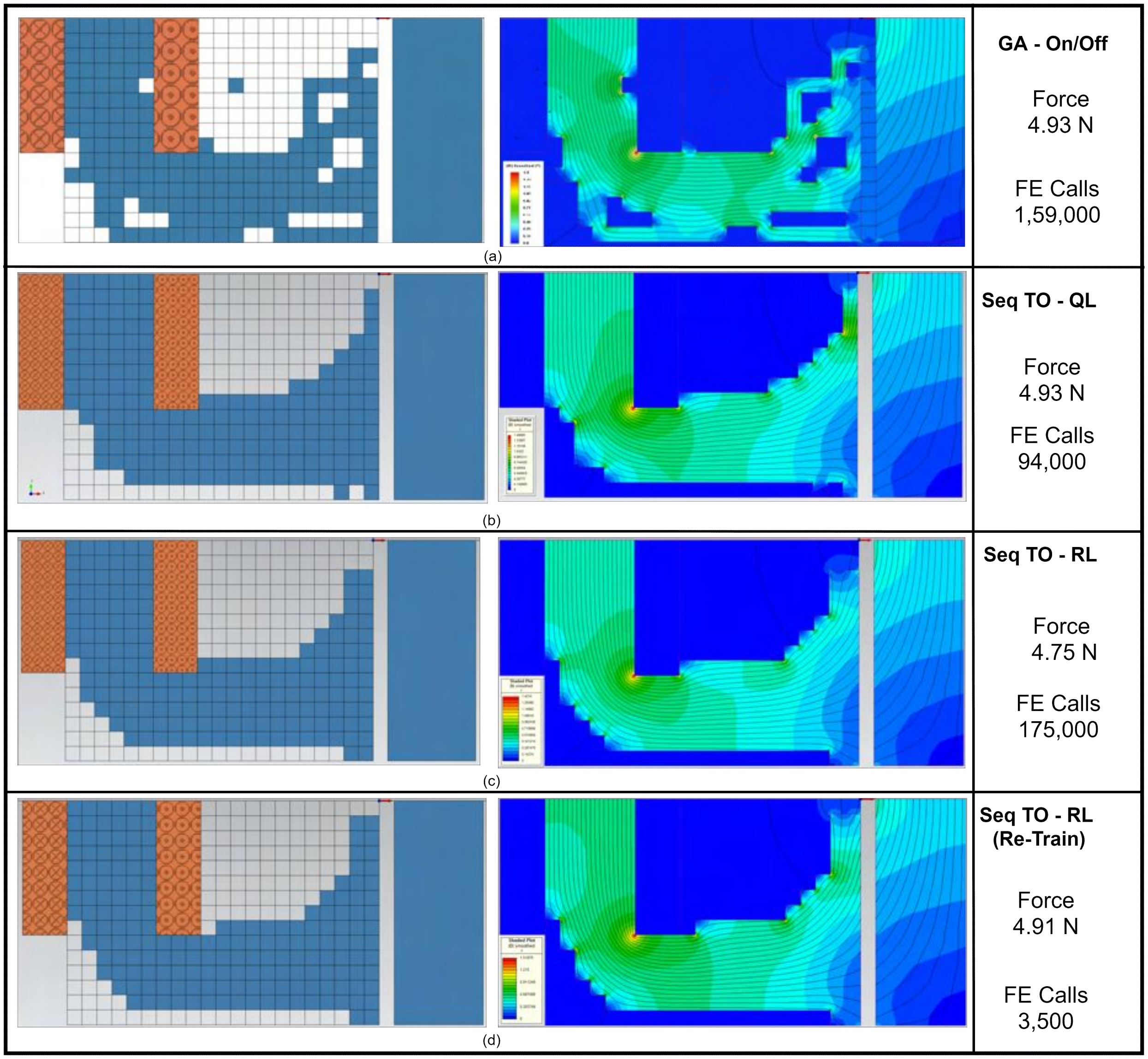


Figure 5.33: Optimal geometry and B distribution for excitation scenario (0.25 A, 500 Turns) for (a) GA (b) QL based on SeqTO-v1 (c) A2C based on SeqTO-v2.

The probabilistic nature of A2C (or PG algorithms), as explained in Section 5.5 can be seen by two optimal designs obtained for excitation of *1.75 A & 500 Turns* in Figure 5.31 (b) and for *1.25 A & 500 Turns* in Figure 5.32. An A2C based agent derives it's action (a) from a stochastic policy $\pi_\theta(a|s)$, defined as the probability of an agent taking action 'a' in any state 's', and is expressed as

$$\pi_\theta(a|s) = \frac{e^{f(s,a)}}{\sum_{a_i} e^{f(s,a_i)}} \tag{5.19}$$

where the policy (π) has θ as its parameters and gives a probability distribution for selecting an action 'a' over all the possible actions (a_i) conditioned on the current state (s).

This is different for Q-Learning which follows a greedy policy, where the algorithm chooses the same (high valued) action every time.

$$a^* = \arg\max_a Q(s,a) \tag{5.20}$$

The different nature of deterministic and stochastic policy is the sole reason for multiple optimal geometries obtained for scenarios: *1.25 A & 500 Turns* and *1.75 A & 500 Turns.*

5.8 Conclusion

It is verified through numerical optimization that the trained A2C agent acquired a policy to optimize the performance of the C-core actuator while strictly satisfying the volume constraint. Compared with conventional optimization methods such as the evolutionary algorithms, which result in high computational cost for iterative electromagnetic analyses for generating optimal designs, a trained deep learning-based RL method is capable of yielding feasible solutions without any time-consuming iterative analysis process. Although the time required for training is long, the agent once trained requires a significantly lower computational cost compared to GA or Tabular QL for previously unseen excitation conditions, thus showing that the trained agent is generalizable. This capability was demonstrated through different excitation scenarios and the RL agent managed to achieve the optimal solution in most of the training scenarios and close of optimal in the rest. These close to optimal topologies (5-10 % lower performance than optimal) can be further improved upon using retrainings requiring very little computation cost. Alternatively, these sub-optimal solutions can serve as a starting seed for a different optimization scheme. The inability of the agent to reach the optimal structure can partially be explained due to the inflexibility of the SeqTO-v2 methodology, where the methodology does not support placing a thin strip close to the air gap, as QL can achieve with SeqTO-v1 (1×1). However, the number of EM simulations (FE calls) needed to use SeqTO-v1 (1×1) will be simply too much to handle as function approximators, such as neural networks, will require far more training data compared to a deterministic tabular look-up table. An improved version of the SeqTO environment will be explored in the future along with the applicability of other RL algorithms. Overall, this is a notable display that the agent possesses generalizability over similar topology optimization tasks.

Although the proposed methodology of Sequence-based TO and sequential decision making using Deep Reinforcement learning algorithms is shown to be capable of providing "instant" optimal or near-optimal solutions for an electromagnetic device, the proposed TO technique and learning methodology can be easily extended to other fields of designing and engineering where TO is used.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

This research aimed to explore the feasibility of modern statistical learning techniques for aiding the design optimization and analysis of electric devices and machines. A number of statistical learning methods, including supervised, bayesian and reinforcement learning have been used in the process.

The first objective of the thesis was to use modern Deep Learning models to reduce the time spent in analyzing field and performance maps for low-frequency electromagnetic devices as compared to conventional high fidelity solvers based on first principles such as the Finite Element method. The high computation demands of high fidelity FE solutions can limit the effectiveness of these analyses and thus the ability to explore new design possibilities. As an example, for estimating the efficiency map of a single motor drive topology, many operating points need to be simulated using finite element analysis. Thus incorporating the whole efficiency map into the design optimization process is an overwhelmingly time-consuming task and may be impossible, depending on the availability of computational resources. For specific applications in the field of computational electromagnetics, the first section of this thesis (Chapter 2 & 3) focuses on designing deep learning strategies for computationally inexpensive and accurate predictions of field distribution and performance maps of electric machines. It is demonstrated that DL algorithms can serve as an ideal venue for fast and precise solutions with sufficient training data. In the process, it was observed that this approach faced the issue of non-reliability and extended time spent in generating data. At times, creating massive datasets using Finite Elements simulations can be too expensive and can take weeks and at times months. For handling the enormous requirements of big data, an information transfer strategy between similar problems referred to as Transfer Learning is also implemented. This is verified with application to two test cases - η map to pf map prediction of same motor topology and η map to η map knowledge transfer for different motor topologies. Overall, a reduction of about 40-60% in overall data requirement was experimentally observed for the two test cases. In scenarios where the NN is not sure about the prediction, a study involving uncertainty analysis in the predictions of DNN is also performed using Bayesian learning.

The other objective of this thesis involved design optimization, where the excessive time spent in the optimization and manufacturability aspect of the optimized design is addressed. A new TO method (SeqTO) was introduced, which inherently generates manufacturable designs, free from checkerboard patterns, perforations or floating pieces of material and requires significantly less computational resources (around 80% lower) as compared to using the same optimization algorithm with the ON/OFF discrete method. The SeqTO environment also serves as the platform for exploring tree search-based heuristic algorithms such as Monte Carlo Tree Search and TD-learning, which further reduces the computation burden by about 5-15 %, compared to using a GA with SeqTO on the same design problem. Leveraging the sequential nature of the controller offered by the SeqTO environment and the pattern learning ability of a NN, a Reinforcement learning agent is also trained, which can generalize to different design parameters, even when such design parameters were unseen during the training phase. It is verified through numerical optimization that the trained agent reduces the number of finite element analysis-based electromagnetic analyses significantly (about 10 times) when deployed for many excitation patterns compared to conventional optimization methods such as the evolutionary algorithms.

Although all the proposed methodologies in this work are capable of providing “instant” solutions for electromagnetic devices, they can be easily extended to other applications where computer simulations play a prominent role in imitating physical laws across various engineering domains.

Overall the main contributions of the thesis revolve around advancing the usage of high fidelity analysis and designing techniques at an early stage of an electric machine development cycle (V-cycle) as envisioned in Chapter 1 (Introduction) of this thesis. It is also aimed to provide practical guidelines and recommendations for successfully deploying modern statistical methods in this field.

6.2 Future Work

A few promising avenues for future work are as follows:

1. The work presented in this thesis takes exploratory steps toward using modern statistical learning methodologies in the field. Generalizability is the key for all the techniques.

The experiment should be set up such that it can be extended to other similar problems with ease. One such approach for EM analysis can be directly solving the governing partial differential equations (PDEs) using physics-informed deep learning methods. Incorporating the prior knowledge of physical laws in the loss function and training the NN in such a fashion should represent the solution of the PDE. Such an approach is intended to be pursued in the future.

2. Data collection for training DL networks and FE-based performance evaluation for RL agents is still computationally expensive.

   (a) For Supervised Learning tasks, the concept of knowledge transfer is explored in Chapter 3. Only two test cases were subjected to the analysis in this work to test knowledge transfer. Future work that investigates different motor designs and observes the reduction of simulation data needed for training as more knowledge gets accumulated in a multi-task DL network could be fruitful.

   (b) On the other hand, for RL based problems, it will be interesting to use multiple independent agents which can interact with different instances of the environment in parallel and thus explore an extensive part of the state-action space in much less wall-clock time.

3. For TO, only single objective tasks were studied in this work. A multi-physics/multi-objective TO problem can be set up using the techniques used in this work with minor changes to the problem setup. Therefore, studying multi-physics and multi-objective problems will be a possible fruitful research direction.

4. It is observed that although SeqTO-*v2* significantly increased the generalizability and reduced the computation burden associated with SeqTO-*v1*, it still has limitations in terms of the granularity of the structure that is produced. Therefore, an improved version of the SeqTO environment should be explored in the future, which can balance the requirements of generalizability, computation load, and the necessary fineness of the structure.

5. In this thesis, only two Deep RL algorithms were explored. However, other algorithms and learning methodologies in the literature, such as PPO (Proximal Policy Optimization Algorithms), SAC (Soft Actor-Critic), and curiosity-based learning, are comparatively faster and can be more robust.

List of References

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

ABB motor sets world record in energy efficiency. URL: `https://new.abb.com/news/detail/1789/ABB-motor-sets-world-record-in-energy-efficiency-saves-half-a-million-dollars` (visited on 06/20/2021).

Alger, Philip Langdon (1970). *Induction machines: their behavior and uses*. Gordon and Breach.

Allaire, G (1997). "The homogenization method for topology and shape optimization". In: *Topology optimization in structural mechanics*. Springer, pp. 101–133.

Allaire, Grégoire et al. (2019). "The homogenization method for topology optimization of structures: old and new". In: *Interdisciplinary Information Sciences* 25.2, pp. 75–146.

Amrhein, Marco and Philip T Krein (2007). "Magnetic equivalent circuit modeling of induction machines design-oriented approach with extension to 3-D". In: *Electric Machines & Drives Conference, 2007. IEMDC'07. IEEE International*. Vol. 2. IEEE, pp. 1557–1563.

Arel, Itamar et al. (2010). "Reinforcement learning-based multi-agent system for network traffic signal control". In: *IET Intelligent Transport Systems* 4.2, pp. 128–135.

Asanuma, Jo, Shuhei Doi, and Hajime Igarashi (2020). "Transfer Learning Through Deep Learning: Application to Topology Optimization of Electric Motor". In: *IEEE Transactions on Magnetics* 56.3, pp. 1–4.

Asghari, Babak and Venkata Dinavahi (2012). "Experimental validation of a geometrical nonlinear permeance network based real-time induction machine model". In: *Power and Energy Society General Meeting, 2012 IEEE*. IEEE, pp. 1–15.

Atkinson, Robert D and Stephen J Ezell (2019). "The Manufacturing Evolution". In:

Badrinarayanan, Vijay, Ankur Handa, and Roberto Cipolla (2015). "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling". In: *arXiv preprint arXiv:1505.07293*.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12, pp. 2481–2495.

Baek, Jeihoon et al. (2014). "Optimal design of five-phase permanent magnet assisted synchronous reluctance motor for low output torque ripple". In: *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, pp. 2418–2424.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Banga, Saurabh et al. (2018). "3d topology optimization using convolutional neural networks". In: *arXiv preprint arXiv:1808.07440*.

Barmada, Sami et al. (2020). "Deep Learning and Reduced Models for Fast Optimization in Electromagnetics". In: *IEEE Transactions on Magnetics* 56.3, pp. 1–4.

Bellman, Richard (1957). "A Markovian decision process". In: *Journal of mathematics and mechanics*, pp. 679–684.

Bendsøe, Martin P (1989). "Optimal shape design as a material distribution problem". In: *Structural and multidisciplinary optimization* 1.4, pp. 193–202.

Bendsøe, Martin Philip, Alejandro Díaz, and Noboru Kikuchi (1993). "Topology and generalized layout optimization of elastic structures". In: *Topology design of structures.* Springer, pp. 159–205.

Bendsøe, Martin Philip and Noboru Kikuchi (1988). "Generating optimal topologies in structural design using a homogenization method". In: *Computer methods in applied mechanics and engineering* 71.2, pp. 197–224.

Bengio, Yoshua, Yann LeCun, et al. (2007). "Scaling learning algorithms towards AI". In: *Large-scale kernel machines* 34.5, pp. 1–41.

Bianchi, N and S Bolognani (1998). "Design optimisation of electric motors by genetic algorithms". In: *IEE Proceedings-Electric Power Applications* 145.5, pp. 475–483.

Bilgin, Berker et al. (2019). "Modeling and analysis of electric motors: state-of-the-art review". In: *IEEE Transactions on Transportation Electrification* 5.3, pp. 602–617.

Bíró, Oszkár et al. (2011). "Multi-domain topology optimization with ant colony systems". In: *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering.*

Blundell, Charles et al. (2015). "Weight uncertainty in neural networks". In: *arXiv preprint arXiv:1505.05424*.

Bostanci, Emine et al. (2017). "Opportunities and challenges of switched reluctance motor drives for electric propulsion: A comparative study". In: *IEEE transactions on transportation electrification* 3.1, pp. 58–75.

Bottauscio, O et al. *Description of TEAM Problem: 32 A Test-Case for Validation of Magnetic Field Analysis with Vector Hysteresis.* Tech. rep. URL: `www.cadema.polito.it/team32/home.html`.

Bouzidi, Ikhlas, Ahmed Masmoudi, and Nicola Bianchi (2015). "Electromagnetic/thermal design procedure of an aerospace electric propeller". In: *IEEE Transactions on Industry Applications* 51.6, pp. 4364–4371.

Box, George (Feb. 2006). "Statistics for Experimenters: Design, Innovation, and Discovery by George E. P. Box; J. Stuart Hunter; William G. Hunter". In: 101, pp. 1720–1721.

Box, George EP (1979). "All models are wrong, but some are useful". In: *Robustness in Statistics* 202.1979, p. 549.

Bremicker, Michael et al. (1991). "Integrated topology and shape optimization in structural design". In: *Journal of Structural Mechanics* 19.4, pp. 551–587.

Bruce, Peter, Andrew Bruce, and Peter Gedeck (2020). *Practical statistics for data scientists: 50+ essential concepts using R and Python.* O'Reilly Media.

Brunton, Steven L and J Nathan Kutz (2019). *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press.

Byun, Jin-Kyu, Joon-Ho Lee, and Il-Han Park (2004). "Node-based distribution of material properties for topology optimization of electromagnetic devices". In: *IEEE transactions on magnetics* 40.2, pp. 1212–1215.

Byun, Jin-kyu, Il-han Park, and Song-yop Hahn (2002). "Topology optimization of electrostatic actuator using design sensitivity". In: *IEEE Transactions on Magnetics* 38.2, pp. 1053–1056.

Campelo, F, J Ramirez, and H Igarashi (2010). *A survey of topology optimization in electromagnetics: considerations and current trends.*

Campelo, Felipe, Kota Watanabe, and Hajime Igarashi (2008a). "Topology optimization with smoothness considerations". In: *International Journal of Applied Electromagnetics and Mechanics* 28.1, 2, pp. 187–192.

— (2008b). "Topology optimization with smoothness considerations". In: *International Journal of Applied Electromagnetics and Mechanics* 28.1-2, pp. 187–192.

Carpenter, CJ (1968). "Magnetic equivalent circuits". In: *Proceedings of the Institution of Electrical Engineers*. Vol. 115. 10. IET, pp. 1503–1511.

Caruana, Rich, Steve Lawrence, and C Lee Giles (2001). "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping". In: *Advances in neural information processing systems*, pp. 402–408.

Chandrasekhar, A. and K. Suresh (2020). "TOuNN: Direct Topology Optimization using Neural Networks". In:

Choi, Jae Seok and Jeonghoon Yoo (2009). "Simultaneous structural topology optimization of electromagnetic sources and ferromagnetic materials". In: *Computer Methods in Applied Mechanics and Engineering* 198.27-29, pp. 2111–2121.

Chu, Mengyu and Nils Thuerey (2017). "Data-driven synthesis of smoke flows with CNN-based feature descriptors". In: *ACM Transactions on Graphics (TOG)* 36.4, pp. 1–14.

Coop, Robert (2018). *Robert Coop, Ph.D. — Head of AI and ML @ Stanley BD - "Audio Event Detection w/Deep Learning"*. URL: https://www.youtube.com/watch?v=9X66iwEQSyI (visited on 06/10/2021).

Cox, HL (1958). *Structures of minimum weight: the basic theory of design applied to the beam under pure bending*. ARC.

Deaton, Joshua D and Ramana V Grandhi (2014). "A survey of structural and multidisciplinary continuum topology optimization: post 2000". In: *Structural and Multidisciplinary Optimization* 49.1, pp. 1–38.

Delalleau, Olivier and Yoshua Bengio (2011). "Shallow vs. deep sum-product networks". In: *Advances in Neural Information Processing Systems*, pp. 666–674.

Deng, Changyu, Can Qin, and Wei Lu (2020). "Deep-Learning-Enabled Simulated Annealing for Topology Optimization". In: *arXiv preprint arXiv:2002.01927*.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Dobbs, Michael W and Lewis P Felton (1969). "Optimization of truss geometry". In: *Journal of the Structural Division* 95.10, pp. 2105–2118.

Doi, Shuhei, Hidenori Sasaki, and Hajime Igarashi (2019). "Multi-objective topology optimization of rotating machines using deep learning". In: *IEEE transactions on magnetics* 55.6, pp. 1–5.

Dumoulin, Vincent and Francesco Visin (2016). “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285*.

Dupré, Luc et al. (2014). “Ant colony optimization for the topological design of interior permanent magnet (IPM) machines”. In: *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*.

Dyck, Derek N and David A Lowther (1996). “Automated design of magnetic devices by optimizing material distribution”. In: *IEEE Transactions on Magnetics* 32.3, pp. 1188–1193.

European Environment Agency (2015). “Evaluating 15 years of transport and environmental policy integration”. In:

— (2016). “Monitoring CO2 emissions from new passenger cars and vans 2015”. In:

Faria, Cassio T et al. (2015). “Design Process of Advanced Reluctance Machines for Electric Vehicle Applications: On Target Setting, Optimization of Different Reluctance Motors Technologies and Assessment of the Most Promising Propulsion Technology for Electric Vehicle Applications”. In: *2015 IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE, pp. 1–6.

FEA Software Market - Global Industry Analysis (2021). URL: https://www.reportlinker.com/p06104732/Finite-Element-Analysis-FEA-Software-Market-Global-Industry-Analysis-Size-Share-Growth-Trends-and-Forecast.html?utm_source=GNW.

Friedlingstein, Pierre et al. (2019). “Global carbon budget 2019”. In: *Earth System Science Data* 11.4, pp. 1783–1838.

Gal, Yarin and Zoubin Ghahramani (2016). “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*, pp. 1050–1059.

Garibaldi, Michele et al. (2019). “Free-form design of electrical machine rotor cores for production using additive manufacturing”. In: *Journal of Mechanical Design* 141.7.

Géron, Aurélien (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media.

Ghorbanian, Vahid (2018). *An HPC-based Data-driven Approach to System-level Design Process for Integrated Motor-Drive Systems*. McGill University (Canada).

Ghorbanian, Vahid and David A Lowther (2017). “A statistical solution to efficiently optimize the design of an inverter-fed permanent-magnet motor”. In: *IEEE Transactions on Industry Applications* 53.6, pp. 5315–5326.

Ghorbanian, Vahid, Armin Salimi, and David Alister Lowther (2017). “A computer-aided design process for optimizing the size of inverter-fed permanent magnet motors”. In: *IEEE Transactions on Industrial Electronics* 65.2, pp. 1819–1827.

Ghorbanian, Vahid et al. (2019). “An HPC-based data-driven process for design exploration and optimization of motor drives”. In: *2019 IEEE International Electric Machines & Drives Conference (IEMDC)*. IEEE, pp. 597–602.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Goodfellow, Ian et al. (2014). “Generative adversarial nets”. In: *Advances in neural information processing systems*, pp. 2672–2680.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2014). “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572*.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, pp. 6645–6649.

Gray, Peter (2011). *Psychology*. Worth Publishers.

Guanghui, SHI et al. (2020). “An aerospace bracket designed by thermo-elastic topology optimization and manufactured by additive manufacturing”. In: *Chinese Journal of Aeronautics* 33.4, pp. 1252–1259.

Guo, Nannan and Ming C Leu (2013). “Additive manufacturing: technology, applications and research needs”. In: *Frontiers of Mechanical Engineering* 8.3, pp. 215–243.

Gyllensten, Freddy et al. (Jan. 2016). “Driving force: Rare-earth-free electric motors with ultrahigh efficiencies deliver sustainable and reliable solutions”. In: *ABB Review* 1, pp. 34–40.

Haftka, Raphael T and Ramana V Grandhi (1986). “Structural shape optimization—a survey”. In: *Computer methods in applied mechanics and engineering* 57.1, pp. 91–106.

Hamdi, Essam S (1994). *Design of small electrical machines*. John Wiley & Sons, Inc.

Hameyer, Kay and Ronnie Belmans (1999). *Numerical modelling and design of electrical machines and devices*. Vol. 1. WIT press.

Hasselt, Hado (2010). “Double Q-learning”. In: *Advances in neural information processing systems* 23, pp. 2613–2621.

He, Kaiming et al. (2015). “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

— (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hemp, William S (1958). “Theory of structural design”. In:

Henderson, Peter et al. (2018). “Deep reinforcement learning that matters”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

Hidaka, Yuki and Hajime Igarashi (2017). “Topology optimization of synchronous reluctance motors considering localized magnetic degradation caused by punching”. In: *IEEE Transactions on Magnetics* 53.6, pp. 1–4.

Hilpisch, Yves (2018). *Python for finance: mastering data-driven finance*. O’Reilly Media.

Hinton, Geoffrey E, Simon Osindero, and Yee-Whye Teh (2006). “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7, pp. 1527–1554.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5, pp. 359–366.

Hur, Jin et al. (1997). “Analysis of PMLSM using three dimensional equivalent magnetic circuit network method”. In: *IEEE Transactions on Magnetics* 33.5, pp. 4143–4145.

Ibrahim, Issah et al. (2020). “Surrogate-Based Acoustic Noise Prediction of Electric Motors”. In: *IEEE Transactions on Magnetics* 56.2, pp. 1–4.

IEEE (1985). “Report of Large Motor Reliability Survey of Industrial and Commercial Installations, Part I”. In: *IEEE Transactions on Industry Applications* IA-21.4, pp. 853–864. DOI: 10.1109/TIA.1985.349532.

Inventory of US Greenhouse Gas Emissions and Sinks: 1990–2016 (2018). Tech. rep. EPA 430-R-18-003.

Jankovics, Davin and Ahmad Barari (2019). “Customization of automotive structural components using additive manufacturing and topology optimization”. In: *IFAC-PapersOnLine* 52.10, pp. 212–217.

Jewett, Jackson L and Josephine V Carstensen (2019). “Topology-optimized design, construction and experimental evaluation of concrete beams”. In: *Automation in Construction* 102, pp. 59–67.

Jin, Jian-Ming (2015). *The finite element method in electromagnetics*. John Wiley & Sons.

Jin, Junqi et al. (2018). “Real-time bidding with multi-agent reinforcement learning in display advertising”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2193–2201.

Kendall, Alex, Vijay Badrinarayanan, and Roberto Cipolla (2015). “Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding”. In: *arXiv preprint arXiv:1511.02680*.

Khan, Arbaaz, Vahid Ghorbanian, and David Lowther (2019). “Deep learning for magnetic field estimation”. In: *IEEE Transactions on Magnetics* 55.6, pp. 1–4.

Khan, Arbaaz and David A Lowther (2020). “Machine Learning applied to the Design and Analysis of Low Frequency Electromagnetic Devices”. In: *2020 21st International Symposium on Electrical Apparatus & Technologies (SIELA)*. IEEE, pp. 1–4.

Khan, Arbaaz, Chetan Midha, and David Alister Lowther (2020). “Sequence-Based Environment for Topology Optimization”. In: *IEEE Transactions on Magnetics* 56.3, pp. 1–4.

Khan, Arbaaz et al. (2020a). “Efficiency Map Prediction of Motor Drives Using Deep Learning”. In: *IEEE Transactions on Magnetics* 56.3, pp. 1–4.

— (2020b). “Transfer Learning for Efficiency Map Prediction”. In: *2020 IEEE 19th Biennial Conference on Electromagnetic Field Computation (CEFC)*, pp. 1–4. DOI: 10.1109/CEFC46938.2020.9451362.

Kim, JK et al. (2004). “Static characteristics of linear BLDC motor using equivalent magnetic circuit and finite element method”. In: *IEEE transactions on magnetics* 40.2, pp. 742–745.

Kim, Youn Hyun et al. (2002). “Analysis of hybrid stepping motor using 3D equivalent magnetic circuit network method based on trapezoidal element”. In: *Journal of applied physics* 91.10, pp. 8311–8313.

Kim, Young Sun and Il Han Park (2010). “Topology optimization of rotor in synchronous reluctance motor using level set method and shape design sensitivity”. In: *IEEE Transactions on Applied Superconductivity* 20.3, pp. 1093–1096.

Kißkalt, Dominik et al. (2018). “Towards a data-driven process monitoring for machining operations using the example of electric drive production”. In: *2018 8th International electric drives production conference (EDPC)*. IEEE, pp. 1–6.

Knüsel, Benedikt and Christoph Baumberger (2020). “Understanding climate phenomena with data-driven models”. In: *Studies in History and Philosophy of Science Part A* 84, pp. 46–56.

Kober, Jens, J Andrew Bagnell, and Jan Peters (2013). “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274.

Kober, T et al. (2020). “Global energy perspectives to 2060–WEC’s World Energy Scenarios 2019”. In: *Energy Strategy Reviews* 31, p. 100523.

Kolehmainen, Jere (2010). “Synchronous reluctance motor with form blocked rotor”. In: *IEEE Transactions on Energy Conversion* 25.2, pp. 450–456.

Krause, Paul et al. (2013). *Analysis of electric machinery and drive systems*. Vol. 75. John Wiley & Sons.

Krishnan, R, R Arumugan, and James F Lindsay (1988). “Design procedure for switched-reluctance motors”. In: *IEEE Transactions on Industry Applications* 24.3, pp. 456–461.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.

Krog, Lars, Alastair Tucker, Gerrit Rollema, et al. (2002). “Application of topology, sizing and shape optimization methods to optimal design of aircraft components”. In: *Proc. 3rd Altair UK HyperWorks users conference*.

LeCun, Yann et al. (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lee, Jangwon and Semyung Wang (2012). “Topological shape optimization of permanent magnet in voice coil motor using level set method”. In: *IEEE transactions on magnetics* 48.2, pp. 931–934.

Lei, Gang et al. (2017). “A review of design optimization methods for electrical machines”. In: *Energies* 10.12, p. 1962.

Lei, Xin et al. (2019). “Machine learning-driven real-time topology optimization under moving morphable component-based framework”. In: *Journal of Applied Mechanics* 86.1.

Leplat, PM et al. (1996). “Comparison between finite element method and magnetic equivalent scheme to model an induction machine”. In: *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering* 15.4, pp. 82–87.

Levine, Sergey et al. (2016). “End-to-end training of deep visuomotor policies”. In: *The Journal of Machine Learning Research* 17.1, pp. 1334–1373.

Li, Min, Frederico G Guimarães, and David A Lowther (2013). “A multiobjective approach for designing the rotor of brushless motors”. In: *IEEE transactions on magnetics* 49.5, pp. 2279–2282.

Li, Yilun (2019). “Numerical methodologies for topology optimization of electromagnetic devices”. PhD thesis. Sorbonne université.

Lim, Sunghoon et al. (2011). “Topology optimization of a magnetic actuator based on a level set and phase-field approach”. In: *IEEE transactions on magnetics* 47.5, pp. 1318–1321.

Lin, Kaixiang et al. (2018). “Efficient Collaborative Multi-Agent Deep Reinforcement Learning for Large-Scale Fleet Management”. In: *arXiv preprint arXiv:1802.06444*.

Liu, Cheng-Tsung and Yi-Hsuan Chiang (2006). “Closed-loop electromagnetic forces calculations and controls of a noncontacting industrial steel plate conveyance system”. In: *IEEE transactions on magnetics* 42.4, pp. 1311–1314.

Liu, Z, Osama A Mohammed, and Shuo Liu (2009). “Equivalent hardware representation of PM synchronous motors from the physics-based phase variable model obtained through FE computation”. In: *IEEE Transactions on Magnetics* 45.3, pp. 1450–1453.

MAGNET v2020.2. Mentor Infolytica, Montreal QC Canada.

Mahmoudi, Amin et al. (2015). “Efficiency maps of electrical machines”. In: *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, pp. 2791–2799.

Mao, Hongzi et al. (2016). “Resource management with deep reinforcement learning”. In: *Proceedings of the 15th ACM workshop on hot topics in networks*, pp. 50–56.

Markov, Andreĭ A and NM Nagornyi (1951). “Teoriya algorifmov (Theory of algorithms)”. In: *Trudy MIAN SSSR* 38, pp. 176–189.

MATLAB (2018). *R2018b (Version 9.5)*. The MathWorks, Inc.

Maxwell, James Clerk (1861). “Xxv. on physical lines of force: Part i.–the theory of molecular vortices applied to magnetic phenomena”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 21.139, pp. 161–175.

Mayr, Andreas et al. (2018). “Electric motor production 4.0–application potentials of industry 4.0 technologies in the manufacturing of electric motors”. In: *2018 8th International Electric Drives Production Conference (EDPC)*. IEEE, pp. 1–13.

Mazur, James E (2015). *Learning and Behavior: Instructor’s Review Copy*. Psychology Press.

McCoy, Gilbert A and John G Douglass (2014). *Premium efficiency motor selection and application guide–a handbook for industry*. Tech. rep. Washington State University Energy Program.

McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

McKay, Michael D, Richard J Beckman, and William J Conover (2000). “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code”. In: *Technometrics* 42.1, pp. 55–61.

Metropolis, N. *The Beginning of Monte Carlo Method*. Tech. rep.

Michell, A.G (1904). “The limits of economy of material in frame-structures”. In: *The London Edinburgh, and Dublin Philos Mag and J Sci* 8.47, pp. 589–597.

Midha, Chetan (2018). “A Study of Topology Optimization Methods for the Design of Electromagnetic Devices”. MA thesis. ECE Dept, McGill University.

Midha, Chetan et al. (2019). “Selection of spatial filters for ON/OFF based topology optimization of a C-core electromagnetic actuator”. In: *IEEE Transactions on Magnetics* 55.10, pp. 1–4.

Mnih, Volodymyr et al. (2013). “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602*.

Mnih, Volodymyr et al. (2016). “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*, pp. 1928–1937.

Moallem, M and GE Dawson (1998). “An improved magnetic equivalent circuit method for predicting the characteristics of highly saturated electromagnetic devices”. In: *IEEE Transactions on magnetics* 34.5, pp. 3632–3635.

Mohammadi, Mohammad Hossain, Vahid Ghorbanian, and David Lowther (2019). “A data-driven approach for design knowledge extraction of synchronous reluctance machines using multi-physical analysis”. In: *IEEE Transactions on Industry Applications* 55.4, pp. 3707–3715.

Mohammadi, Mohammad Hossain and David Alister Lowther (2017). “A computational study of efficiency map calculation for synchronous AC motor drives including cross-coupling and saturation effects”. In: *IEEE Transactions on Magnetics* 53.6, pp. 1–4.

Morgan, HD et al. (2014). “GE Jet engine bracket challenge: a case study in sustainable design”. In: *Sustainable design and manufacturing*, pp. 95–107.

Mou, Lili et al. (2016). “How transferable are neural networks in nlp applications?” In: *arXiv preprint arXiv:1603.06111*.

Mousavi-Aghdam, Seyed Reza et al. (2015). “Design and analysis of a novel high-torque stator-segmented SRM”. In: *IEEE Transactions on Industrial Electronics* 63.3, pp. 1458–1466.

Murakami, Hiroshi et al. (1999). “The performance comparison of SPMSM, IPMSM and SynRM in use as air-conditioning compressor”. In: *Conference Record of the 1999 IEEE Industry Applications Conference. Thirty-Forth IAS Annual Meeting (Cat. No. 99CH36370)*. Vol. 2. IEEE, pp. 840–845.

Nicolaescu, Marius-Andrei (2012). “Rotary electrical motors used in medical industry relevant design considerations and actual advances”. In: *2012 International Conference and Exposition on Electrical and Power Engineering*. IEEE, pp. 664–667.

Nie, Zhenguo et al. (2020). “TopologyGAN: Topology Optimization Using Generative Adversarial Networks Based on Physical Fields Over the Initial Domain”. In: *arXiv preprint arXiv:2003.04685*.

Okamoto, Yoshifumi, Koji Akiyama, and Norio Takahashi (2006). “3-D topology optimization of single-pole-type head by using design sensitivity analysis”. In: *IEEE Transactions on Magnetics* 42.4, pp. 1087–1090.

Okamoto, Yoshifumi and Norio Takahashi (2006). “Investigation of topology optimization of magnetic circuit using density method”. In: *Electrical Engineering in Japan* 155.2, pp. 53–63.

Okamoto, Yoshifumi et al. (2015). “Material-density-based topology optimization with magnetic nonlinearity by means of stabilized sequential linear programming: SLPSTAB”. In: *IEEE Transactions on Magnetics* 51.3, pp. 1–4.

Osher, Stanley and James A Sethian (1988). “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations”. In: *Journal of computational physics* 79.1, pp. 12–49.

Ostović, V (1987). “Magnetic equivalent circuit presentation of electric machines”. In: *ELECTRIC MACHINES AND ELECTROMECHANICS* 12.6, pp. 407–432.

Ostovic, Vlado (2012). *Dynamics of saturated electric machines*. Springer Science & Business Media.

Pan, Sinno Jialin and Qiang Yang (2009). “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359.

“Paris agreement” (2015). In: *Report of the Conference of the Parties to the United Nations Framework Convention on Climate Change (21st Session, 2015: Paris). Retrived December*. Vol. 4. HeinOnline, p. 2017.

Park, Sang-In and Seungjae Min (2009a). “Magnetic actuator design for maximizing force using level set based topology optimization”. In: *IEEE transactions on magnetics* 45.5, pp. 2336–2339.

— (2009b). “Magnetic actuator design for maximizing force using level set based topology optimization”. In: *IEEE transactions on magnetics* 45.5, pp. 2336–2339.

Park, Sang-in et al. (2008). “Magnetic actuator design using level set based topology optimization”. In: *IEEE Transactions on magnetics* 44.11, pp. 4037–4040.

Park, Soonok and Jeonghoon Yoo (2012). “Structural optimization of a multi-physics problem considering thermal and magnetic effects”. In: *IEEE transactions on magnetics* 48.11, pp. 3883–3886.

Paszke, Adam et al. (2017). “Automatic differentiation in PyTorch”. In:

Prager, William and John E Taylor (1968). “Problems of optimal structural design”. In:

Radun, Arthur (2000). “Analytically computing the flux linked by a switched reluctance motor phase when the stator and rotor poles overlap”. In: *IEEE Transactions on magnetics* 36.4, pp. 1996–2003.

Rahman, Tanvir et al. (2017). “Comparison of fractional-slot concentrated winding and PM-assisted synchronous reluctance motors for class IV electric vehicles”. In: *2017 IEEE International Electric Machines and Drives Conference (IEMDC)*. IEEE, pp. 1–7.

Ray, WF et al. (1984). “Switched reluctance motor drives for rail traction: a second view”. In: *IEE Proceedings B (Electric Power Applications)*. Vol. 131. 5. IET, pp. 220–225.

Reddy, Patel B et al. (2012). “Comparison of interior and surface PM machines equipped with fractional-slot concentrated windings for hybrid traction applications”. In: *IEEE Transactions on Energy Conversion* 27.3, pp. 593–602.

Rivera, Christian A et al. (2017). “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)*. IEEE, pp. 1–6.

Rosu, Marius et al. (2017). *Multiphysics simulation by design for electrical machines, power electronics and drives*. John Wiley & Sons.

Rozvany, George IN, Ming Zhou, and Torben Birker (1992). “Generalized shape optimization without homogenization”. In: *Structural and Multidisciplinary Optimization* 4.3, pp. 250–252.

Rozvany, GIN and Ming Zhou (1991). “The COC algorithm, part I: cross-section optimization or sizing”. In: *Computer Methods in Applied Mechanics and Engineering* 89.1-3, pp. 281–308.

Salimi, Armin (2018). “Computer-aided design of electrical machines: the role of robustness and application of statistical analysis techniques in multi-objective design optimization”. PhD thesis. McGill University Libraries.

Salon, Sheppard Joel (1995). *Finite element analysis of electrical machines*. Vol. 101. Kluwer academic publishers Boston USA.

Sarcar, MMM, K Mallikarjuna Rao, and K Lalit Narayan (2008). *Computer aided design and manufacturing*. PHI Learning Pvt. Ltd.

Saruwatari, M et al. (2016). “Design study of 15-MW fully superconducting generators for offshore wind turbine”. In: *IEEE transactions on applied superconductivity* 26.4, pp. 1–5.

Sasaki, Hidenori and Hajime Igarashi (2019). “Topology optimization accelerated by deep learning”. In: *IEEE Transactions on Magnetics* 55.6, pp. 1–5.

Sato, Takahiro, Kota Watanabe, and Hajime Igarashi (2014). “A modified immune algorithm with spatial filtering for multiobjective topology optimisation of electromagnetic devices”. In: *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* 33.3, pp. 821–833.

Schmidhuber, Jürgen (2015). “Deep learning in neural networks: An overview”. In: *Neural networks* 61, pp. 85–117.

Senior, Andrew W et al. (2020). “Improved protein structure prediction using potentials from deep learning”. In: *Nature* 577.7792, pp. 706–710.

Sharafi, Armin (2013). “Knowledge discovery in databases”. In: *Knowledge Discovery in Databases*. Springer, pp. 51–108.

Sheikh-Ghalavand, B, S Vaez-Zadeh, and A Hassanpour Isfahani (2009). “An improved magnetic equivalent circuit model for iron-core linear permanent-magnet synchronous motors”. In: *IEEE Transactions on Magnetics* 46.1, pp. 112–120.

Sheikman, Boris Leonid, Dwayne Andrew Folden, and Samuel Thomas Walter Francis (2014). *Methods and systems for monitoring stator winding vibration*. US Patent 8,812,254.

Sigmund, O and J Petersson (1998). “Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima”. In: *Structural optimization* 16.1, pp. 68–75. ISSN: 1615-1488. DOI: 10.1007/BF01214002. URL: https://doi.org/10.1007/BF01214002.

Sigmund, Ole (2001). “Design of multiphysics actuators using topology optimization–Part I: One-material structures”. In: *Computer methods in applied mechanics and engineering* 190.49-50, pp. 6577–6604.

Silva, Rodrigo (2018a). “Surrogate Problem Evaluation and Selection for Optimization with Expensive Function Evaluations”.

Silva, Rodrigo Cesar Pedrosa (2018b). “Surrogate problem evaluation and selection for optimization with expensive function evaluations”. PhD thesis. McGill University Libraries.

Silva, Rodrigo César Pedrosa et al. (2017a). “Multiple operating points based optimization: Application to fractional slot concentrated winding electric motors”. In: *IEEE Transactions on Industrial Electronics* 65.2, pp. 1719–1727.

Silva, Rodrigo CP et al. (2017b). “Surrogate-based MOEA/D for electric motor design with scarce function evaluations”. In: *IEEE Transactions on Magnetics* 53.6, pp. 1–4.

Silver, David et al. (2016). “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587, pp. 484–489.

Silver, David et al. (2017). “Mastering the game of go without human knowledge”. In: *nature* 550.7676, pp. 354–359.

Silvester, Peter P and Ronald L Ferrari (1996). *Finite elements for electrical engineers*. Cambridge university press.

Simcenter MotorSolve. URL: http://www.infolytica.com/.

Simonyan, Karen and Andrew Zisserman (2014). “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556*.

Singh, Satinder P and Richard S Sutton (1996). “Reinforcement learning with replacing eligibility traces”. In: *Machine learning* 22.1-3, pp. 123–158.

Sivak, Michael and Brandon Schoettle (2018). “Relative Costs of Driving Electric and Gasoline Vehicles in the Individual US States”. In:

Sosnovik, Ivan and Ivan Oseledets (2019). “Neural networks for topology optimization”. In: *Russian Journal of Numerical Analysis and Mathematical Modelling* 34.4, pp. 215–223.

Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Sudhoff, Scott D et al. (2007). “Magnetic equivalent circuit modeling of induction motors”. In: *IEEE Transactions on Energy Conversion* 22.2, pp. 259–270.

Sutton, Richard S (1988). "Learning to predict by the methods of temporal differences". In: *Machine learning* 3.1, pp. 9–44.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Tang, Wei et al. (2017). "Study on a Poisson's equation solver based on deep learning technique". In: *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*. IEEE, pp. 1–3.

Tavana, Nariman Roshandel and Venkata Dinavahi (2016). "Real-time nonlinear magnetic equivalent circuit model of induction machine on FPGA for hardware-in-the-loop simulation". In: *IEEE Transactions on Energy Conversion* 31.2, pp. 520–530.

The MathWorks, Inc. (2017). *Global Optimization Toolbox User's Guide*. `https://nl.mathworks.com/help/pdf_doc/gads/gads_tb.pdf`.

Theano Development Team (May 2016). "Theano: A Python framework for fast computation of mathematical expressions". In: *arXiv e-prints* abs/1605.02688. URL: `http://arxiv.org/abs/1605.02688`.

Thorndike, Edward L (1898). "Animal intelligence: an experimental study of the associative processes in animals." In: *The Psychological Review: Monograph Supplements* 2.4, p. i.

Thorndike, Edward Lee (1913). *The psychology of learning*. Vol. 2. Teachers College, Columbia University.

Tompson, Jonathan et al. (2017). "Accelerating eulerian fluid simulation with convolutional networks". In: *International Conference on Machine Learning*. PMLR, pp. 3424–3433.

Tüchsen, Johann et al. (2018). "Data driven design selection and generation-an industrial case study on electric motors". In: *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, pp. 1709–1720.

Turing, AM (1950). "Mind". In: *Mind* 59.236, pp. 433–460.

Uddin, Wasi and Yilmaz Sozer (2017). "Analytical modeling of mutually coupled switched reluctance machines under saturation based on design geometry". In: *IEEE Transactions on Industry Applications* 53.5, pp. 4431–4440.

Um, Kiwon, Xiangyu Hu, and Nils Thuerey (2018). "Liquid splash modeling with neural networks". In: *Computer Graphics Forum*. Vol. 37. 8. Wiley Online Library, pp. 171–182.

"UNFCCC, Adoption of the Paris agreement. COP" (2015). In: *25th session Paris* 30.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30, pp. 5998–6008.

Waide, Paul and Conrad U Brunner (2011). "Energy-efficiency policy opportunities for electric motor-driven systems". In:

Wang, Bofan et al. (2016). "A neural network based surrogate model for predicting noise in synchronous reluctance motors". In: *2016 IEEE Conference on Electromagnetic Field Computation (CEFC)*. IEEE, pp. 1–1.

Wang, Liwei, Juri Jatskevich, and Hermann W Dommel (2007). "Re-examination of synchronous machine modeling techniques for electromagnetic transient simulations". In: *IEEE Transactions on Power Systems* 22.3, pp. 1221–1230.

Wang, S and J Kang (2002). "Topology optimization of nonlinear magnetostatics". In: *IEEE Transactions on Magnetics* 38.2, pp. 1029–1032.

Wang, Semyung, Seungkyu Park, and Jenam Kang (2004). “Multi-domain topology optimization of electromagnetic systems”. In: *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering.*

Watkins, Christopher JCH and Peter Dayan (1992). “Q-learning”. In: *Machine learning* 8.3-4, pp. 279–292.

Watthewaduge, Gayan et al. (2020). “Electromagnetic Modeling Techniques for Switched Reluctance Machines: State-of-the-Art Review”. In: *IEEE Open Journal of the Industrial Electronics Society* 1, pp. 218–234.

Wlas, Miroslaw, Zbigniew Krzeminski, and Hamid A Toliyat (2008). “Neural-network-based parameter estimations of induction motors”. In: *IEEE Transactions on Industrial Electronics* 55.4, pp. 1783–1794.

World Energy Outlook 2019 – Analysis - IEA. URL: `https://www.iea.org/reports/world-energy-outlook-2019` (visited on 06/20/2021).

Yang, Zhi et al. (2015). “Comparative study of interior permanent magnet, induction, and switched reluctance motor drives for EV and HEV applications”. In: *IEEE Transactions on Transportation Electrification* 1.3, pp. 245–254.

Yilmaz, Murat and Philip T Krein (2008). “Capabilities of finite element analysis and magnetic equivalent circuits for electrical machine analysis and design”. In: *2008 IEEE Power Electronics Specialists Conference*. IEEE, pp. 4027–4033.

Yoo, J and H-J Soh (2005). “An optimal design of magnetic actuators using topology optimization and the response surface method”. In: *Microsystem Technologies* 11.12, pp. 1252–1261.

Yoo, Jeonghoon, Noboru Kikuchi, and John L Volakis (2000). “Structural optimization in magnetic devices by the homogenization design method”. In: *IEEE Transactions on Magnetics* 36.3, pp. 574–580.

Yu, Yonggyun et al. (2019). “Deep learning for determining a near-optimal topological design without any iteration”. In: *Structural and Multidisciplinary Optimization* 59.3, pp. 787–799.

Zhang, Dongxia, Xiaoqing Han, and Chunyu Deng (2018). “Review on the research and practice of deep learning and reinforcement learning in smart grids”. In: *CSEE Journal of Power and Energy Systems* 4.3, pp. 362–370.

Zhang, Yiquan et al. (2019). “A deep Convolutional Neural Network for topology optimization with strong generalization ability”. In: *arXiv preprint arXiv:1901.07761.*

Zhang, Yuejin et al. (2006). “A finite element–analytical method for electromagnetic field analysis of electric machines with free rotation”. In: *IEEE transactions on magnetics* 42.10, pp. 3392–3394.

Zhao, SW et al. (2011). “Survey of modeling methods for flux linkage of switched reluctance motor”. In: *2011 4th International Conference on Power Electronics Systems and Applications*. IEEE, pp. 1–4.

Zhu, Ji-Hong, Wei-Hong Zhang, and Liang Xia (2016). “Topology optimization in aircraft and aerospace structures design”. In: *Archives of Computational Methods in Engineering* 23.4, pp. 595–622.

Zhu, Jihong et al. (2021). “A review of topology optimization for additive manufacturing: Status and challenges”. In: *Chinese Journal of Aeronautics* 34.1, pp. 91–110. ISSN: 1000-

9361. DOI: https://doi.org/10.1016/j.cja.2020.09.020. URL: https://www.sciencedirect.com/science/article/pii/S1000936120304520.