5-2023

Generative Neural Network Approach to Designing and Optimizing Dynamic Inductive Power Transfer Systems

Andrew Pond Curtis
Utah State University

GENERATIVE NEURAL NETWORK APPROACH TO DESIGNING AND OPTIMIZING DYNAMIC INDUCTIVE POWER TRANSFER SYSTEMS

by

Andrew Pond Curtis

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Nicholas Flann, Ph.D. Major Professor	Yuan Shuhan, Ph.D. Committee Member
Vladimir Kulyukin, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2023

ABSTRACT

Generative Neural Network Approach to Designing and Optimizing Dynamic Inductive Power Transfer Systems

by

Andrew Pond Curtis, Master of Science

Utah State University, 2023

Major Professor: Nicholas Flann, Ph.D.
Department: Computer Science

Generative Neural Networks (GNN) have demonstrated remarkable power in creating novel graphic design images from text-to-image training, stemming from their ability to be creative yet constrained in a regular domain. This work applies GNNs to design dynamic inductive power transfer systems to make charging electrical vehicles (EVs) more convenient and cheaper. Discovering optimal and safe coil implementations (for EV and road) is challenging because of the combinatorial explosion of possible configurations, along with multiple conflicting objective functions, such as maximizing the output power, while minimizing stray magnetic fields and the volume of windings and magnetic cores. To solve the problem, a differentiable simulator and evaluator define loss functions that train a generative neural network to only produce configurations that satisfy eight given design criteria. Before training, the rate of finding successful designs is 0.005%, but within 500 training epochs, the rate becomes 98% (about 30 seconds total GPU run time). Solution diversity and quality may be further improved by applying loss functions trained on pairwise Pareto-optimal generated examples.

(57 pages)

PUBLIC ABSTRACT

Generative Neural Network Approach to Designing and Optimizing Dynamic Inductive Power Transfer Systems

Andrew Pond Curtis

Electric vehicles (EVs) offer many improvements over traditional combustion engines including increasing efficiency, while decreasing cost of operation and emissions. There is a need for the development of cheap and efficient charging systems for the future success of EVs. Most EVs currently utilize static plug-in charging systems. An alternative charging method of significant interest is dynamic inductive power transfer systems (DIPT). These systems utilize two coils, one placed in the vehicle and one in the roadway to wirelessly charge the vehicle as it passes over. This method removes the current limitations on EVs where they must stop and statically charge for a period of time. However, the physical designs of the charging unit coils depends on many physical parameters, which leads to complexity when determining how to design the unit. A design then needs to be judged for its quality and performance, for which there are eight proposed objective functions. These objective functions represent performance metrics but are conflicting. Some metrics, such as output power are to be maximized, while others such as stray magnetic field and volume of windings and magnetic cores are to be minimized. Different unit designs will trade off performance for these objectives.

In order to address the complex issue of finding near-optimal designs, a machine-learning, Generative Neural Network (GNN) approach is presented for the rapid development of near-optimal DIPT systems. GNNs generate new examples from a trained neural network and have demonstrated remarkable power in creating novel graphic design images from text-to-image training. This stems from their ability to be creative yet constrained in a regular domain. In this case, a simulator network and evaluator generator network are implemented. The simulator is a pre-trained neural network that maps from the physical designs to the objective functions. The generator network is trained

to generate the near-optimal physical designs. A design is considered successful if it passes given thresholds for each of the eight objective functions, which evaluate the quality of a design. Before training, the rate of finding successful designs is 0.005%, but within 500 training epochs the rate becomes 98% (about 30 seconds total GPU run time). Engineers and production professionals are interested in both the best performing designs as well as a diversity of configurations to build. In order to improve on these criteria, several different loss functions were developed that incorporate the objective functions. Loss functions are what the neural networks use to determine how to adjust parameters and produce a better design. The various loss functions presented greatly influence the ability of the GNN to produce diverse and high-performance design solutions.

I dedicate this work to my parents, Chuck and Karen, who inspired me to pursue education and supported me along the way.

ACKNOWLEDGMENTS

I would like to give special thanks to my supervisor Dr. Nicholas Flann, whose mentorship and collaboration was invaluable in this work.

I also extend my thanks to my colleague, collaborator and friend Md Shain Shahid Chowdhury Oni.

Furthermore, thanks to Shuntaro Inoue and to his collaborators and faculty from the electrical engineering department at USU for their preliminary work which was essential for this project.

I would like to express my gratitude to the USU computer science department faculty and administration for working with me and providing for the assistantship that facilitated my graduate education.

Finally, thanks to my parents, Chuck and Karen, as well as my close friends Jaxxton, Sam and Arian for their support during the course of my studies.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

GNN	Generative Neural Network
EV	Electric Vehicle
DIPT	Dynamic Inductive Power Transfer
IPT	Inductive Power Transfer
FEM	Finite Element Method
GAN	Generative Adversarial Network
DS	Design Solution
SM	Surrogate Model
GP	Geometric Parameters
MP	Magnetic Parameters
μT	Microtesla

CHAPTER 1

Introduction

Dynamic inductive power transfer (DIPT) systems transmit energy wirelessly and provide a solution to charge EVs while driving. As a result, the EV range is extended, potentially infinitely, and the need for plug-in-charging is reduced or eliminated. Furthermore, wireless charging technology can lower battery weight and cost because the EV does not need a large battery to operate between charges. A DIPT system is composed of two coils of wire, one positioned within the EV and the other fixed in the roadway, seen in Figure 1.1.

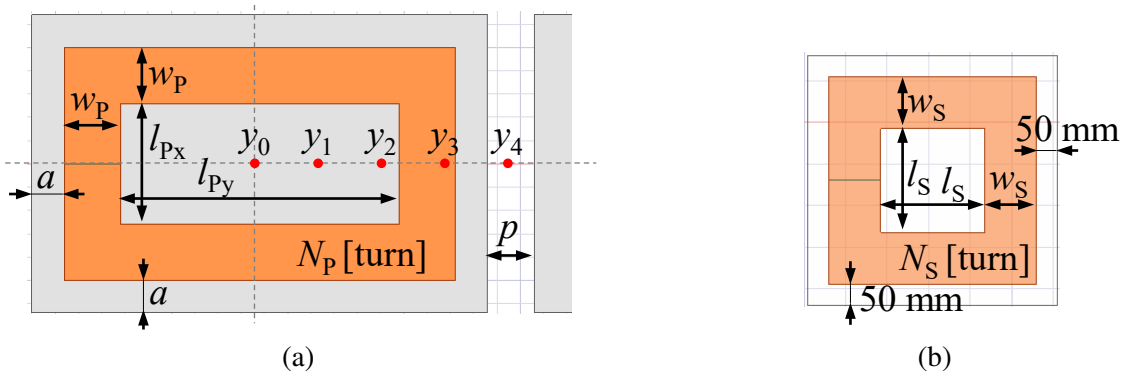

Fig. 1.1: **Coil design parameters**. (a) The primary coil, described by the parameters l_{Px}, l_{Py}, w_P, a, p, which are dimensions in mm and N_p the number of wire turns. (b) The secondary coil, described by the parameters, l_S, w_S, dimensions in mm and N_S, the number of wire turns.

The physics of operation are well understood and computational models exist [1] that enable simulation and evaluation of a specific coil design using conventional numerical analytic methods, such as the finite-element method (FEM). Given a means to simulate the behavior of a potential design, engineers can define specific objective functions that quantify an implementation precisely.

This work presents a machine-learning solution to this design problem where an initial random design generator is trained to produce design solutions that pass specific objective thresholds. A generative neural network is applied to shift a Gaussian distribution, initially independent of design objectives, to one in which most designs generated pass objective thresholds. The method works in

a similar way that a generative adversarial network learns to create images that look like paintings by Monet [2].

Successful generative models in the text-to-image domain [3] rely on fully differentiable loss functions [4], that enable back-propagation of error calculated from image and text data samples. The work presented here utilizes the simulation and evaluation model given in [1] as loss functions because it is fully differentiable. Inoue [1] initially implemented a traditional FEM simulator, which was then used to train a simple feed-forward neural network forming a surrogate model. This surrogate model produced designs that were validated experimentally [1]. Since it is an ordinary neural network, this surrogate model and the objective functions written within Pytorch are all fully differentiable. The work presented here utilizes the fully-differentiable simulation model given in [1] in conjunction with an evaluator network to perform back-propagation of novel loss functions utilizing the objectives.

1.0.1 Problem Definition: Generating Near-Optimal DIPT systems

Given:

(1) A trainable, but initially random design generator that returns a vector of numbers that describe a specific DIPT design configuration, $DS_i = \langle I_p,\ N_p,\ N_s,\ l_{\mathrm{Px}},\ l_{\mathrm{Py}},\ w_{\mathrm{P}},\ a,\ p,\ l_{\mathrm{S}},\ w_{\mathrm{S}} \rangle$ given in Table 3.1.

(2) A simulation model that takes a specific design as input and calculates the expected behavior of the DIPT system under operation. Here the magnetic fields generated and electrical power induced in the secondary coil is determined as the secondary coil passes over the primary coil.

(3) Eight objective functions that take the simulation result and calculate specific design objectives, given in Table 3.3.

Find:

A generator of DIPT designs that quickly learns to create a diversity of coil implementations that satisfy and attempt to optimize all eight criteria given in Table 3.3.

CHAPTER 2

Background and Related Works

Pareto optimization or multi-objective optimization is an area of decision making involving multiple conflicting objectives. These objectives are numerical and can be mathematically computed simultaneously. When this issue is present for a set of objectives, points will appear on a non-dominated Pareto front. This means for a pair of objectives, no improvements can be made for that design in that objective without some trade-off in another. The points that fit into this criteria are called non-dominated or Pareto points. An example of this can be seen in Figure 2.1. The blue points represent design solutions that are dominated, the colored points are Pareto points lying along the Pareto front, visualized by the red line.

The appearance of Pareto optimization problems are common in many areas such as economics, engineering, science and logistics. The field of electrical power systems often deals with these optimization problems and have used a variety of algorithms and viewpoints to address it. This section will cover the common types of algorithms used on these optimization problems, their advantages and disadvantages and introduce some previous work on applying gradient-based approaches to Pareto optimization problems.

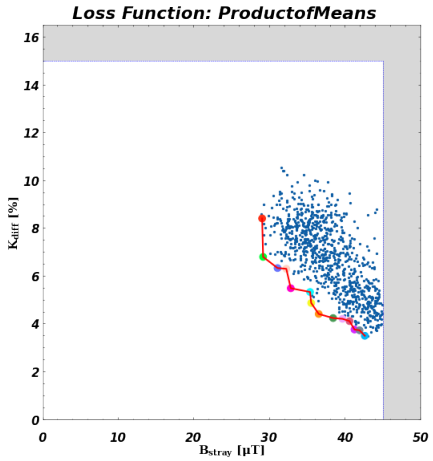

Fig. 2.1: Example of a Pareto Front with two objectives. Each colored point represents a non-dominated Pareto point and the red line is the Pareto front. The blue points are dominated solutions.

The Lexicographic method [5] [6] assigns a weight to objectives denoting their importance or significance. In this case, one objective function can not degrade the performance of its predecessor optimization objective functions. This solves the issues involving which objectives to prioritize. A variation of this method to make it more robust involves adding a relaxation factor, proposed by Marler [7]. This approach relaxes some constraints by a factor δ_i. Another variation is to combine a-priori linear weight assignment with other objective functions, known as a weighted sum approach [5]. The drawback to these methods is that the objectives are not considered equally and revolve around having and applying a-priori domain knowledge to assign the fitness function. These choices can also severely inhibit overall optimization for designs. Since for this application, the objective functions are to be considered equally, and without a-priori weighting, this methodology cannot be directly applied.

The most popular non-gradient-based optimization methods primarily come from evolutionary algorithms. These algorithms imitate the biological evolution of living things or the social interactions among species [8] to perform multi-objective optimization. The three main subgroups of evolutionary algorithms that have been applied for this area are: genetic algorithms (GAs), Memetic algorithms (MAs) and particle swarm optimization (PSO).

In a genetic algorithm, an initial population is created representing a set of design solutions. Individuals in the population are then evaluated based on a fitness function and can either ”survive” or ”die”. A variety of approaches can then be applied to permeate the surviving individuals, often referred to as crossover or mutation, to create new children. This mimics the natural survival of the fittest. The population then evolves over time.

Memetic algorithms [9] have the distinctive feature of allowing all chromosomes and offspring to gather some experience through a local search and improvement phase before participating in the evolutionary process. This allows for the individuals to adapt themselves. This can lead to improvements in performance for some applications.

The particle swarm optimization method [10] was motivated by the group dynamics of a flock of migratory birds attempting to reach an unknown destination. In this method, a swarm (population) of particles (individuals) moves together through the objective space. The movement is motivated

by both the performance of the particle as well as the population. These individuals do not produce new children such as in the GAs and MAs.

Ant-colony optimization algorithms evolve similarly to PSO but instead rely on ants (individuals) to leave behind pheromones. Pheromones are information giving the performance for objectives that an ant found. Later iterations of ants use this information to guide them to better solutions. Dorigo's [11] development of ant-colony optimization was based on the observation that ants can determine the shortest path between their colony and a food source.

Evolutionary-based approaches to multi-objective optimization problems are clever and can work in the case of non-differentiable solutions. However, they are relatively slow algorithms and often will not come close to finding global minimums for optimization. Powerful new gradient-based approaches with deep-learning back-propagation allow for greatly increased speed and performance in the case that the solution can be determined in a fully-differentiable manner.

Inoue [1] proposed an optimization method for a dynamic power inductive system using the combination of the finite element method (FEM) and neural networks to find the non-linear relationship between design parameters and optimization methods. Inoue combined this method with a genetic algorithm approach and found a substantial number (26%) of design solutions that met threshold values. A result of this work was the creation a feed-forward neural network trained on FEM simulations that was capable of mapping from design parameters to the spatiotemporal behavior of a design.

The opportunity taken in this research was to utilize Inoue's feed-forward network in a generative adversarial network (GAN) to find and accelerate the production of novel and near-optimal designs. GANs [12] are a powerful approach to generating synthetic data that is novel but similar to a provided data set. GANs are popular in the domains of text-to-image translation, realistic human face generation, photo blending, and many more applications. Examples of recent popular text-to-image generators are DALLE-2 and Stable Diffusion, and for the text-to-text domain ChatGPT has been making waves in mainstream media.

Using GANs directly in optimization problems is a developing field. Some applications of utilizing a coupled-network approach are discussed next. Albuquerque [13] proposed an idea to

solve the optimization problems using a generator and multiple discriminators. They evaluated the performance of their method using multiple gradient descent and hyper-volume maximization [14] on multiple datasets and showed that their method produced an impressive result.

Chen et al. [15] proposed a novel surrogate-assisted deep learning approach to solving the heat source layout optimization problem called neighborhood search-based layout optimization (NSLO) to reduce the complexity of incorporating computationally intensive finite methods inside the training loop. NSLO is integrated with a feature pyramid network (FPN), which functions as a substitute model to resolve discrete heat source layout optimization issues. Experimental findings in this work, show the FPN surrogate model can generate an accurate prediction with enough training data and maintain a good generalization capability.

Li [16] proposed a non-iterative topology optimization algorithm for conductive heat transfer structures that handles pixels and produces a near-optimal solution by using two coupled neural networks and a trained generative adversarial network approach. Their approach consists of predicting near-optimal structure in coarse resolution and obtaining ultimate structure refinement in fine resolution. The drawback of this approach is that it may not accurately anticipate refinement if predictions are made for high-resolution design domains.

One additional method proposed later in this research involves utilizing Pareto points in the loss function calculation. Recent methods utilizing Pareto front calculations with neural networks have either required creating a new network and optimizing it for each point on the Pareto front or using hyper-networks with conditioned preferences to significantly increase the number of trainable parameters. Ruchte [17] suggested augmenting the feature space with these preferences to condition the network on them directly. They also suggested punishing solutions that retain a slight angle to the preference vector to achieve a well-spread Pareto front. Suzuki [18] proposed a multi-objective Bayesian optimization method based on entropy. Existing entropy-based techniques either omit trade-offs between objectives through oversimplification or require complicated approximations to measure entropy.

DIPT systems optimization have multiple objectives with no obvious a-priori domain weighting. Since the solutions to these objective functions are well known and fully differentiable, it provides

the opportunity to apply gradient-based approaches with back-propagation. This allows for a vast increase in speed of training as well as access to new approaches for finding optimal design solutions using GANs. The loss functions presented later in this work enable learning of the generator to produce well-performing designs with complicated interactions of objective functions without the need for a-priori domain knowledge or weighting.

CHAPTER 3

Methods

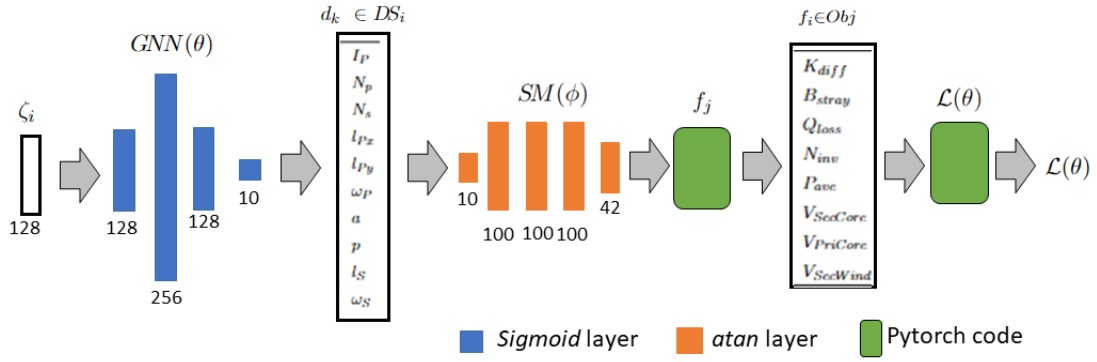

Fig. 3.1: **Generative Model Architecture**. A design $d_k \in DS_i$ (see Figure 1) is generated from Gaussian noise ζ_i passed through a four-layer neural network $GNN(\theta)$. The design is evaluated using a surrogate simulation model $SM(\phi)$ followed by eight objective functions $f_j \in O$. Learning is accomplished by back-propagating gradients of a loss function $\mathcal{L}(\theta)$ formulated to improve all the objectives.

3.1 Overview of Generative Neural Network Approach

Generative neural networks can solve engineering design problems since they demonstrate the ability to be creative yet constrained within a regular domain [12]. In this work, we use a similar approach to a GAN network, but the loss function is based directly on the objectives rather than an error over a data set. Figure 3.1 depicts the learner's architecture. A vector of Gaussian noise ζ_i is generated and input to the Generator Network, $\zeta_i \circ GNN(\theta)$, where θ are the trainable parameters of the generator network. The output is scaled into physical units d_k to represent a design solution, $d_k \in DS_i$ as illustrated in Figure 1.1. The design DS_i is input into the simulator model $SM(\phi)$. $SM(\phi)$ are the parameters of the simulator network and are held constant. The output of the simulator model produces the spatiotemporal behavior of the design under operation. The eight objective functions defined in Table 3.3 are then calculated and formed into a loss function considering the minimization or maximization goals. Finally, both loss $\mathcal{L}(\theta)$ and its gradient $\partial\mathcal{L}(\theta)/\partial\theta$ are calculated and back-propagated into the generator network to update weights θ, using the Adam optimizer with a learning

rate of $3 \cdot 10^{-4}$, which was found to be effective during testing.

The input to $GNN(\theta)$ is a vector ζ_i of 128 samples of Gaussian noise, with a mean of 0.0 and standard deviation of 1.0. The generator consists of an input layer, three hidden layers, and an output layer. Each of the hidden layers are batch normalized with atan activation functions and are fully connected, providing 68,618 trainable parameters in θ. The simulator model is a pre-trained neural network that maps the design parameters d_k to the electromagnetic behavior when operated [1]. The simulator model $SM(\phi)$ is excluded from the network's training process, with the 25,742 parameters ϕ set to non-trainable during back-propagation.

3.2 Generative Network

3.2.1 Noise Input

As a standard, gaussian noise was used for input into the generative neural network. For empirical testing 10^4 10-element vectors of noise were produced for input to the GNN for each epoch of training. The mean for the gaussian noise was set to 0.0 and the standard deviation as 1.0.

3.2.2 Generative Neural Network Architecture

The generative neural network architecture can be seen in figure 3.1, it consists of input and output layers of vector length 10. The output layer uses a sigmoid activation function. There are three hidden layers of sizes 128, 256 and 128. The feed forward network uses batch normalization and atan activation functions on the hidden layers. The 10-element output vector represents a set of design parameters for a DIPT system and are listed in table 3.1. Feeding the 10^4 10-element vectors of noise produces 1000 possible sets of design parameters for each epoch.

3.2.3 Design Parameters

The physical design solution parameters, $DS_i = \langle I_p,\ N_p,\ N_s,\ l_{\mathrm{Px}},\ l_{\mathrm{Py}},\ w_{\mathrm{P}},\ a,\ p,\ l_{\mathrm{S}},\ w_{\mathrm{S}} \rangle$ and ranges for the DIPT coil systems are given in Table 3.1. Additional, fixed design specifications are given in table 3.2.

I_p is the electrical current provided to the primary coil, while N_p and N_s define the number of

coil turns for the primary and secondary coil, respectively. The remaining seven parameters define physical characteristics measured in mm of the two coils, illustrated in Figure 1.1.

Parameter	Variable	Value
Primary Coil RMS current	I_P	50~200 A
Turn number of the primary side	N_p	4-10
Turn number of the secondary side	N_s	4-10
X-direction length of the primary winding	l_{Px}	50~650 mm
Y-direction length of the primary winding	l_{Py}	50~2050 mm
Width of the primary winding	ω_P	25~325 mm
Length of the edge of the primary core	a	0~200 mm
Pitch of the adjacent cores	p	0~200 mm
Length of the secondary winding	l_S	50~450 mm
Width of the secondary winding	ω_S	25~225 mm

Table 3.1: The design parameter definitions and ranges for the DIPT system.

Parameter	Variable	Value
Input DC voltage	V_{dc}	400 V
Output DC voltage	V_{bat}	400 V
Output power at the center	$P_{out,x0,y0}$	50 kW
Switching frequency	f_s	85 kHz
Air Gap	g	200 mm
Length of the edge of the secondary core	b	50 mm

Table 3.2: Design specifications for the fixed parameters of the DIPT system.

3.3 Simulator Model

The design parameter sets are sent though the pre-trained simulator model in order to produce another set of output parameters that represent the spatio-temporal behavior a DIPT design. This feed-forward neural network simulator model was trained using FEM simulations [1]. The weight and bias parameters of the simulator model are held constant since the intention is to observe the resulting behavior of the design parameter sets. The simulator network architecture can be seen in Figure 3.1. The input to the simulator model is 1000 12-element vectors, the increased vector length is from the addition of some intermediary calculations on the design parameters. The output layer from the surrogate model produces 42 output parameters which represent magnetic parameters (MP) that are explored below in in 3.1. There are 3 linear hidden layers with size 100. The first two layers have atan activation functions. The network is fully differentiable and has 25,742 parameters which are held static during training in order to retain the simulated FEM performance of a design.

3.3.1 Simulator model output parameters

The output parameters of the simulator model can be identified with three groups with various values depending on misalignment (x) and position (y). These groups are: coupling coefficients (k), self-inductance of primary coil (L), and magnetic fields (B). Further description is given below:

$$\begin{aligned} \text{MP}: \; & L_{\text{P,x0,yi}},\; L_{\text{P,x1,y0}},\; L_{\text{S,x0,yi}},\; L_{\text{S,x1,y0}},\; k_{\text{x0,yi}}, \\ & k_{\text{x1,y0}},\; \boldsymbol{B}_{\text{P,x0,y0,0deg}},\; \boldsymbol{B}_{\text{P,x0,y0,90deg}},\; \boldsymbol{B}_{\text{S,x0,y0,0deg}}, \\ & \boldsymbol{B}_{\text{S,x0,y0,90deg}},\; \boldsymbol{B}_{\text{P,x1,y0,0deg}},\; \boldsymbol{B}_{\text{P,x1,y0,90deg}}, \\ & \boldsymbol{B}_{\text{S,x1,y0,0deg}},\; \boldsymbol{B}_{\text{S,x1,y0,90deg}}, \end{aligned} \tag{3.1}$$

Where:

$L_{\text{P,x0,yi}}$ self inductance of primary side coil ($i = 0 \sim 4$) when misalignment is 0 mm ($= x_0$)

$L_{\text{S,x0,yi}}$ self inductance of primary side coil ($i = 0 \sim 4$) when misalignment is 0 mm($= x_0$)

$L_{\text{P,x1,yi}}$ self inductance of primary side coil ($i = 0 \sim 4$) when misalignment is 100 mm ($= x_1$)

$L_{\text{S,x1,yi}}$ self inductance of primary side coil ($i = 0 \sim 4$) when misalignment is 100 mm($= x_1$)

$k_{\text{x0,yi}}$ coupling coefficient when misalignment is 0 mm ($= x_0$)

$k_{x1,yi}$ coupling coefficient when misalignment is 100 mm (= x_1)

$B_{P,x0,y0,0deg}$ magnetic fields at 0 in one cycle of the operational frequency

$B_{P,x0,y0,90deg}$ magnetic fields at 90 in one cycle of the operational frequency

$B_{S,x0,y0,0deg}$ magnetic fields at 0 in one cycle of the operational frequency

$B_{S,x0,y0,90deg}$ magnetic fields at 90 in one cycle of the operational frequency

$B_{P,x1,y0,0deg}$ magnetic fields at 0 in one cycle of the operational frequency when misalignment is 100 mm

$B_{P,x1,y0,90deg}$ magnetic fields at 90 in one cycle of the operational frequency when misalignment is 100 mm

$B_{S,x1,y0,0deg}$ magnetic fields at 0 in one cycle of the operational frequency when misalignment is 100 mm

$B_{S,x1,y0,90deg}$ magnetic fields at 90 in one cycle of the operational frequency when misalignment is 100 mm

3.4 Objective Functions

Here the eight objective functions are defined, each calculated from the design parameters and the output of the simulator model. The threshold values are given in table 3.3 and discussed individually below.

Design Criteria	Objective function	Threshold
Coupling coefficient difference	K_{diff}	$\leq$ 15%
Stray magnetic fields	B_{stray}	$\leq$ 45 μT
Coil loss	Q_{loss}	$\leq$ 2000 W
Number of inverters	N_{inv}	$\leq$ 1 1/ m
Power average	P_{ave}	$\geq$ 30 KW/ m
Secondary core volume	$V_{SecCore}$	$\leq$ 1500cm^3
Primary core volume	$V_{PriCore}$	$\leq$ 6000cm^3
Secondary side core winding volume	$V_{SecWind}$	$\leq$ 1000cm^3

Table 3.3: Eight Design objective functions $f_i \in O$ definition and constraints

3.4.1 Stray Magnetic field (B_{stray})

Stray magnetic fields, B_{stray}, are measured in μT and are to be minimized below 45 to adhere to the federally regulated SAE standard [19]. The width of the vehicles is assumed to be 1.6 m. The maximum values of the magnetic stray field at a distance of 800 mm in the lateral direction from the center of the secondary coil at the middle of the 200 mm air gap between primary and secondary coils are called the stray field. The magnetic stray field B_{stray} can be obtained for any combination of currents (I_{P}, I_{S}) and any combination of turn numbers (N_{t}, N_{r}) by the following equations,

$$\begin{aligned} B_{\text{x0,y0}} =&\{[\text{Re}(\boldsymbol{B}_{\text{P,x0,y0}})N_{\text{P}}I_{\text{P}} + \text{Re}(\boldsymbol{B}_{\text{S,x0,y0}})N_{\text{S}}I_{\text{S}}]^2 + \\ &[\text{Im}(\boldsymbol{B}_{\text{P,x0,y0}})N_{\text{P}}I_{\text{P}} + \text{Im}(\boldsymbol{B}_{\text{S,x0,y0}})N_{\text{S}}I_{\text{S}}]^2\}^{1/2} \end{aligned} \tag{3.2}$$

$$\begin{aligned} B_{\text{x1,y0}} =&\{[\text{Re}(\boldsymbol{B}_{\text{P,x1,y0}})N_{\text{P}}I_{\text{P}} + \text{Re}(\boldsymbol{B}_{\text{S,x1,y0}})N_{\text{S}}I_{\text{S}}]^2 + \\ &[\text{Im}(\boldsymbol{B}_{\text{P,x1,y0}})N_{\text{P}}I_{\text{P}} + \text{Im}(\boldsymbol{B}_{\text{S,x1,y0}})N_{\text{S}}I_{\text{S}}]^2\}^{1/2} \end{aligned} \tag{3.3}$$

$$B_{\text{stray}} = \max\{B_{\text{x0,y0}}, B_{\text{x1,y0}}\}. \tag{3.4}$$

3.4.2 Coupling coefficient (K_{diff})

The coupling coefficient, K_{diff}, measures the effect of misalignment on the charging of the coil. This is calculated between (x_0, y_0) and (x_1, y_0). K_{diff} is to be minimized and below a threshold value of 15%. A small K_{diff} improves the output power reduction in misalignment conditions in the lateral direction but requires larger transmit coils.

$$k_{\text{diff}} = \frac{|k_{\text{x0,y0}} - k_{\text{x1,y0}}|}{\max\{|k_{\text{x0,y0}}|, |k_{\text{x1,y0}}|\}}. \tag{3.5}$$

3.4.3 Total output power (P_{ave})

Power average P_{ave}, is the average power transferred during dynamic operation and is to be

maximized and above a value of 30 kW but not to exceed 50 kW in order to achieve consistent output. For this DIPT design, a double-sided LCC compensation network capacitor system is assumed[1]. In this system the rms receiver current I_{r} is represented as

$$I_{\mathrm{S}} = \frac{P_{\mathrm{out,x0,y0}}}{\omega I_{\mathrm{P}} k_{\mathrm{x0,y0}} \sqrt{L_{\mathrm{P,x0,y0}} L_{\mathrm{S,x0,y0}}}} \tag{3.6}$$

where I_{P} is the rms current of the transmitter coil and $P_{\mathrm{out,x0,y0}}$ is output power when the receiver coil is above the center of the transmitter coil. When the reciever coil moves to positions $y_{\mathrm{S}} = y_1 \sim y_4$, seen in Figure 1.1, the output power is calculated as

$$P_{\mathrm{out,x0,yi}} = \omega k_{\mathrm{x0,yi}} I_{\mathrm{P}} I_{\mathrm{S}} \sqrt{L_{\mathrm{P,x0,yi}} L_{\mathrm{S,x0,yi}}} \ (i = 1, 2, 3), \tag{3.7}$$

$$P_{\mathrm{out,x0,y4}} = 2\omega k_{\mathrm{x0,y4}} I_{\mathrm{P}} I_{\mathrm{S}} \sqrt{L_{\mathrm{P,x0,y4}} L_{\mathrm{S,x0,y4}}}. \tag{3.8}$$

The output power $P_{\mathrm{out,x0,y4}}$ is doubled in the assumption of a system with an input and an output voltage source because the receiver coil is induced by the two primary coils when the receiver coil is at the edge of the primary coils[]. Using (3.7) and (3.8), the average output power P_{ave} is calculated as

$$P_{\mathrm{ave}} = (P_{\mathrm{out,x0,y0}} + 2P_{\mathrm{out,x0,y1}} + 2P_{\mathrm{out,x0,y2}} + 2P_{\mathrm{out,x0,y3}} + P_{\mathrm{out,x0,y4}})/8. \tag{3.9}$$

3.4.4 Coil loss (Q_{loss})

Coil loss, Q_{loss}, is the power in watts lost to heat and is to be minimized and below 2 kW. The approximate coil loss is calculated by the following,

$$Q_{\mathrm{loss}} = \frac{\omega L_{\mathrm{P,x0,y0}} I_{\mathrm{P}}^2}{Q_{\mathrm{coil,P}}} + \frac{\omega L_{\mathrm{S,x0,y0}} I_{\mathrm{S}}^2}{Q_{\mathrm{coil,S}}}, \tag{3.10}$$

where $Q_{\mathrm{coil,P}} = Q_{\mathrm{coil,S}} = 400\mathrm{W}$ are assumed as a general case.

3.4.5 Number of inverters (N_{inv})

Number of inverters N_{inv}, measured in 1/m and is to be minimized and less than 1. The number of inverters can be defined as follows:

$$N_{inv} = \frac{1}{l_{\mathrm{Py}} + 2 \times w_{\mathrm{P}} + 2 \times\ a +\ p} \tag{3.11}$$

3.4.6 Primary core volume ($V_{PriCore}$)

Primary core volume, $V_{PriCore}$, measures the volume in cm^3 of the magnetic core in the primary coil. This is to be minimized and less than 6000 cm^3 for safety and efficiency.

$$V_{PriCore} = \frac{(l_{\mathrm{Py}} + 2 \times w_{\mathrm{P}} + 2 \times\ a) \times (l_{\mathrm{Px}} + 2 \times w_{\mathrm{P}} + 2 \times\ a)}{10^3} \tag{3.12}$$

3.4.7 Secondary core volume ($V_{SecCore}$)

Secondary core volume, $V_{SecCore}$, measures the volume in cm^3 of the magnetic core in the EV coil. This is to be minimized and less than 1500 cm^3 for safety and efficiency.

$$V_{SecCore} = \frac{(l_s + 2 \times w_S + 2 \times b)^2 \times 5}{10^3} \tag{3.13}$$

3.4.8 Secondary side core winding volume ($V_{SecWind}$)

Secondary windings volume, $V_{SecWind}$, measures the volume in cm^3 of the wire in the EV coil. This is to be minimized and less than 1000 cm^3 for safety, efficiency and cost.

$$V_{SecWind} = \frac{4 \times (l_s + w_S) \times (6.6)^2}{(10^3) \times N_s} \tag{3.14}$$

3.5 Loss Functions and Optimization

The objective functions have a clear preference for minimization or maximization when it comes to determining an optimal design. This lends itself to gradient-based optimization and use of a neural network. The Adam optimizer, a variant of the stochastic gradient descent algorithm, was used with a learning rate of $3x10^{-4}$. When performing gradient descent the network attempts

to adjust the weights and biases of the perceptrons in a layer of the neural network in order to minimize the loss. The objective functions represent convex functions that can be act as the loss. However, the Adam optimizer depends on a single value of the loss to minimize. Thus, the objective functions must be combined in some fashion in order to provide the learner with one combined loss calculation. The construction of the loss functions using the objective functions greatly affects the ability of the network to learn to produce well-performing and diverse DIPT design solutions. Once a loss function is chosen the final loss is back-propagated through the network. Since all of the calculations are fully differentiable, the loss function can follow back through the calculations and the static parameters of the simulator model to the generative neural network. There the weights and biases are adjusted and better designs can be produced on the next epoch.

3.5.1 Product-of-Means Loss

To avoid the introduction of arbitrary hyper-parameters, no scaling is performed on any of the objective functions. The Adam optimizer was used with a learning rate of $3 \cdot 10^{-4}$. A simple product-of-means loss was first explored, where $\mathcal{L}(\theta)$ is the product of the means of all individual objective functions over a solution population generated during an epoch, which is set at $n = 10^3$.

The P_{ave} objective is to be maximized with an upper bound, so we use a $Pavemin$ in the loss calculation instead. P_{avemin} is defined by deducting $Pave$ from a pragmatic maximum of 10,000. This is done in order to preserve network gradients in the case that a design solution for P_{ave} exceeds 50kW.

Let O be the set of objective functions = $\{K_{diff}, B_{stray}, Q_{loss}, N_{inv}, V_{SecCore}, V_{PriCore}, V_{SecWind}, 10^4 - P_{ave}\}$. Then the product-of-means loss function is defined:

$$\mathcal{L}(\theta) = \prod_{f_j \in O} \frac{1}{n} \sum_{i=1}^{i=n} \zeta_i \circ GNN(\theta) \circ SM(\phi) \circ f_j \tag{3.15}$$

Where θ is the vector of 68,618 trainable weights in the generative NN and ϕ is the vector of 25,742 non-trainable weights in the surrogate model NN.

3.5.2 Pairwise Pareto Optimal Loss

The challenge of this application is that there are many objective functions to optimize and trade-offs are present where some objectives can be improved while others are worsened. This issue is also known as Pareto optimality. Pareto optimal designs are so-called non-dominated solutions for which no improvements can be made for that criteria without losing some optimization in another. In order to find further improvements to objective functions and design solution diversity Pareto optimality can be considered when designing loss functions. In order to accomplish this, pairwise sets of the criteria were analyzed during training to determine the Pareto points for the set. These points represent the best solutions for that set of pairwise criteria. These points can then be utilized directly in the loss function in order to focus on the best designs and not just the means. For this loss function, one pairwise Pareto optimal set is used in place of all of the designs for that objective. The remainder of the loss function is still the product of all of the objectives.

Let O_o be the pair of objective functions to be improved, $\hat{O}$ be $O - O_o$ and PO be the set of random vectors that produce designs that are non-dominated w.r.t O_o, then the pairwise Pareto optimal loss function may be defined as:

$$\mathcal{L}_{PO}(\theta) = \prod_{f_j \in \hat{O}} \frac{1}{n} \sum_{i=1}^{i=n} \zeta_i \circ GNN(\theta) \circ SM(\phi) \circ f_j \prod_{f_j \in O_o} \frac{1}{\|PO\|} \sum_{s_i \in PO} s_i \circ GNN(\theta) \circ SM(\phi) \circ f_j \quad (3.16)$$

3.5.3 Mean-of-Products Loss

For the Mean-of-Products loss function, the consecutive element-wise product of the objective functions O, solution matrices is taken before finally taking the mean of the resulting matrix. The Mean-of-Products loss function may be defined as:

$$\mathcal{L}_{mean}(\theta) = \frac{1}{n} \sum_{i=1}^{i=n} \prod_{f_j \in O} \zeta_i \circ GNN(\theta) \circ SM(\phi) \circ f_j \quad (3.17)$$

3.5.4 Minimum-of-Products Loss

Similarly to the Mean-of-Products loss, the Minimum-of-Products loss takes the consecutive

element-wise product of the objective functions O, and then takes the minimum of the resulting matrix. The Minimum-of-Products loss function may be defined as:

$$\mathcal{L}_{min}(\theta) = \min_{i=1}^{i=n} \prod_{f_j \in O} \zeta_i \circ GNN(\theta) \circ SM(\phi) \circ f_j \tag{3.18}$$

CHAPTER 4

Results

Objective functions K_{diff}, B_{stray}, Q_{loss},N_{inv},$V_{SecCore}$,$V_{PriCore}$,$V_{SecWind}$ are to be minimized and have upper bounds given in Table 3.3. The remaining objective function, P_{ave}, is to be maximized and exceed $30kW$ with an upper bound of $50kW$. The bounds on P_{ave} come from domain knowledge of the designs. $30kW$ is the minimum power for the design to be considered effective. Exceeding $50kW$ would indicate spikes in power which causes inconsistencies in charging and safety concerns. In order to consider P_{ave} in the loss function, we use a proxy objective called $Pavemin$ instead. P_{avemin} is defined by deducting P_{ave} from a pragmatic maximum of $10,000kW$. This is done in order to preserve network gradients in the case that a design solution for P_{ave} exceeds $50kW$.

For empirical testing of the loss functions' performance, the number of epochs is set to 500 for training and each epoch the generator is fed 10^3 Gaussian noise. The best epoch of training with respect to the loss is saved for the production of the design parameters and testing of objective functions in the results. Another method that was explored was to save the best trained network epoch as the one with the highest number of accepted solutions. However, this approach did not encourage highly performing designs to the same degree.

In the results section, the red lines in the objective plots are the Pareto Optimal fronts for each set O_a of objective functions. The colored dots map the same solutions across the objectives. These solutions were chosen for visualization from the 5 best performing Pareto optimal points for one objective in a Pareto optimal set O_a. In the design parameter plots, a passing design solution is represented by a line. Each point on the radial axis represents the percentage of the design parameter DS_i range. The colored lines map the same solutions as the colored points in the objectives plots.

4.1 Summary of Accepted Solutions Produced by the Loss Functions

Loss Function	Number of Solutions
Random Network	3
Product of Means Loss	948
Pareto B_{stray} and K_{diff}	1000
Pareto N_{inv} and Q_{loss}	484
Pareto $V_{SecCore}$ and P_{ave}	952
Pareto $V_{SecWind}$ and $V_{PriCore}$	978
Mean-of-Products Loss	84
Minimum-of-Products Loss	172

Table 4.1: Number of accepted solutions produced per epoch by each loss function after generator network training.

4.2 Random Network

For validation, the first test involved running a simple random network. The generator was fed Gaussian noise and produced design parameters DS_i. There was no loss being calculated or propagated from the objective functions O. In this case, no learning was performed and the generator produces only three solutions per epoch. The number of accepted solutions per training epoch can be seen in Figure 4.1. Figure 4.2 shows an example of the generated solutions performance for objective functions O and the corresponding design parameters DS_i.

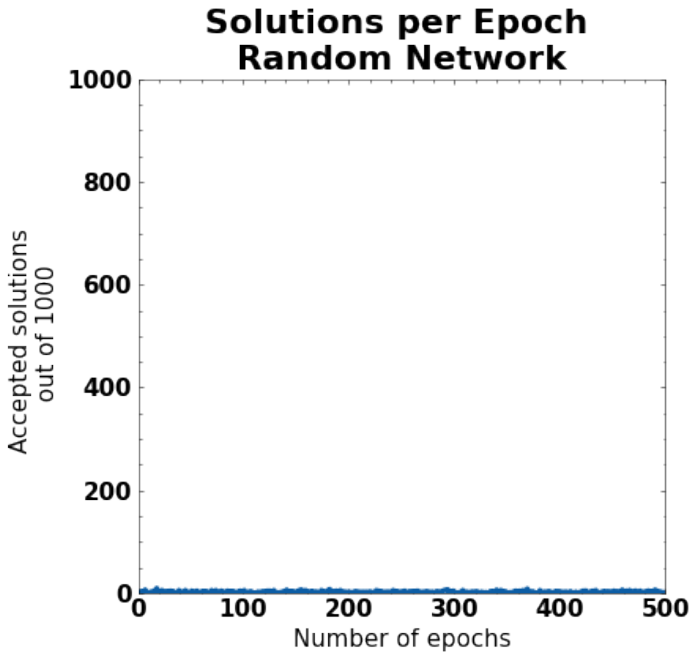

Fig. 4.1: Number of accepted solutions passing thresholds for objective functions O per epoch for the random network.

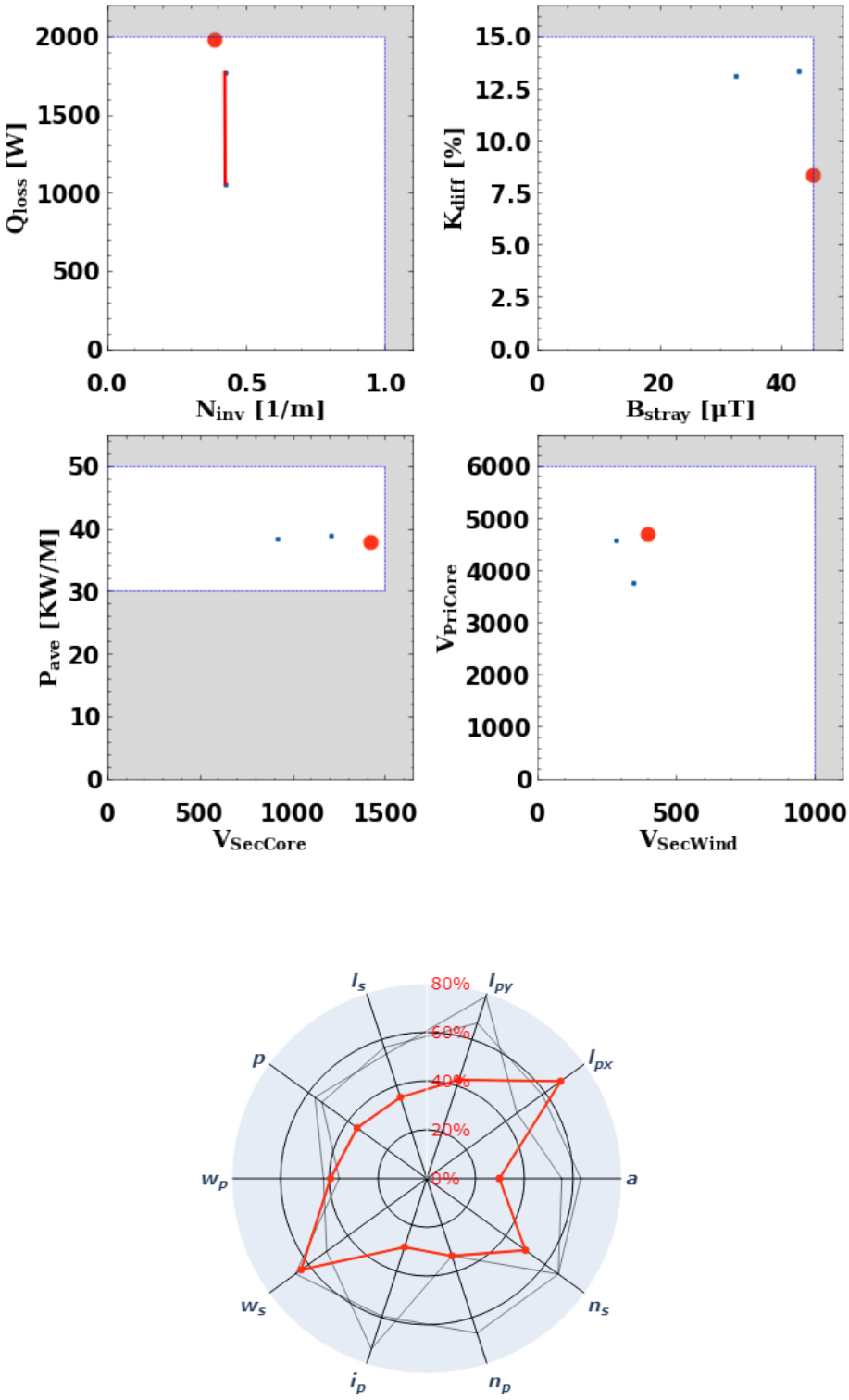

Fig. 4.2: Solutions for objective functions O (top), and design parameters DS_i (bottom). The random network generator produced three solutions.

4.3 Product-of-Means Loss

The Product-of-Means loss performs very well at learning to generate accepted solutions, seen in Figure 4.3. The distribution of the design performance for objective functions tends to be dense and move together over the course of training.

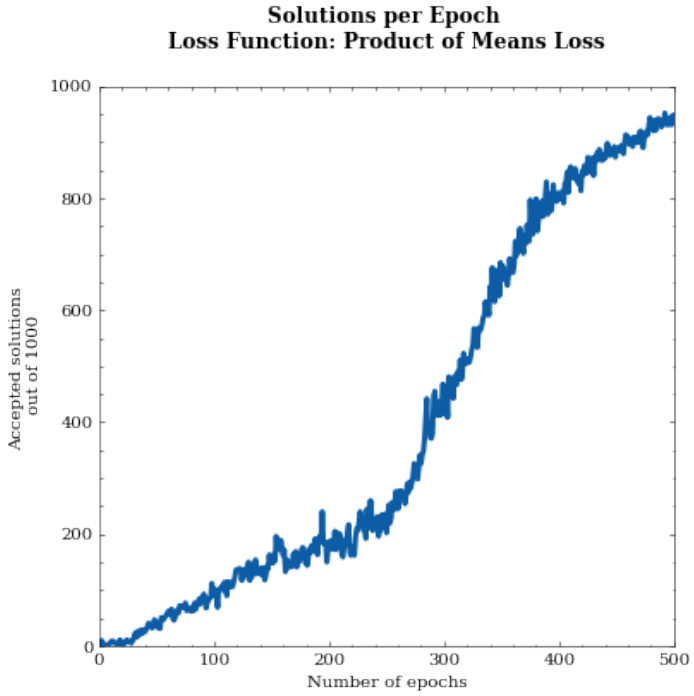

Fig. 4.3: Number of accepted solutions passing thresholds for objective functions O per epoch for the Product-of-Means Loss.

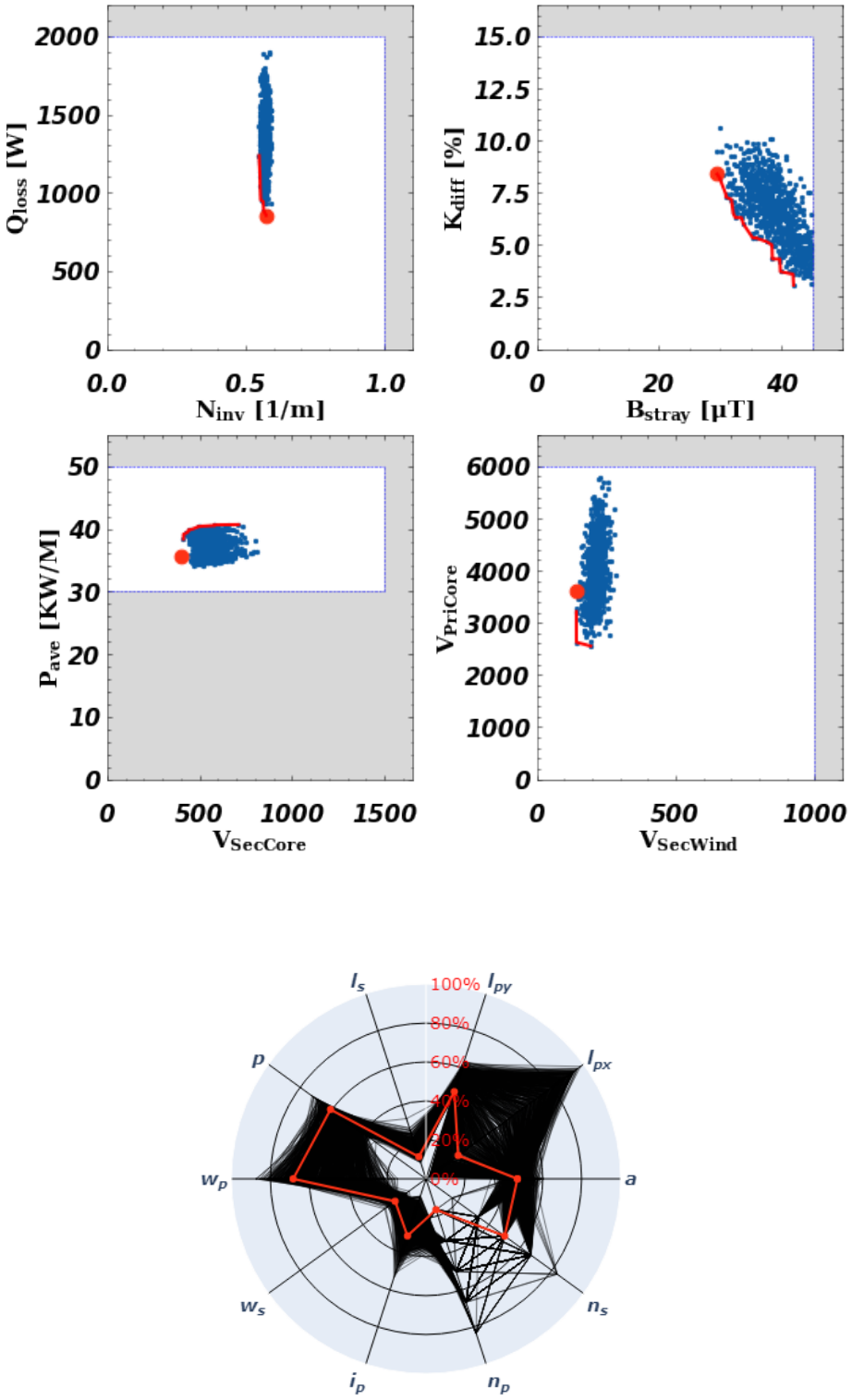

Fig. 4.4: Solutions after training using the Product-of-Means loss function for objective functions O (top), and design parameters DS_i (bottom). The top performing Pareto optimal point, highlighted in red, for B_{stray}, is shown for the set O_a of B_{stray} and K_{diff}.

An example coil diagram, Figure 4.5 was made for the selected red solution from Figure 4.4. This selected design is drawn to scale.

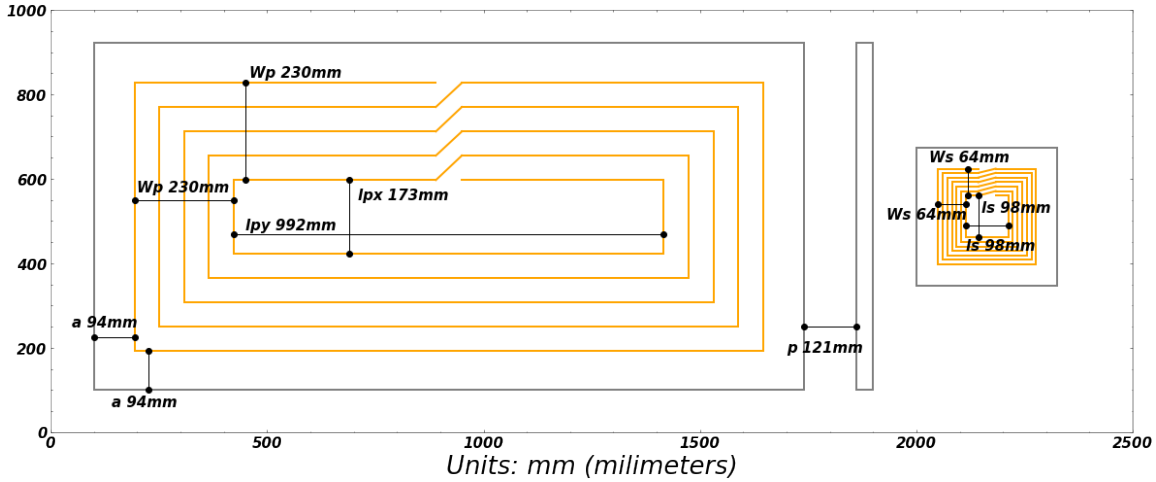

Fig. 4.5: Functional coil design, drawn to scale, of primary coil (left) and secondary coil (right). Dimensions selected from red solution in Figure 4.4

4.4 Pairwise Pareto Optimal Loss

Let O_o be the pair of objective functions to be improved, $\hat{O}$ be $O - O_o$ and PO be the set of random vectors that produce designs that are non-dominated w.r.t O_o. The results for the sets of O_o are given below.

4.4.1 B_{stray} K_{diff}

The loss function with the Pareto set of B_{stray} and K_{diff} learned to produce the most accepted designs out of any loss function, with the trained generator producing 1000 accepted solutions out of 1000. The distribution in the objective space is similar to the Product-of-Means loss function.

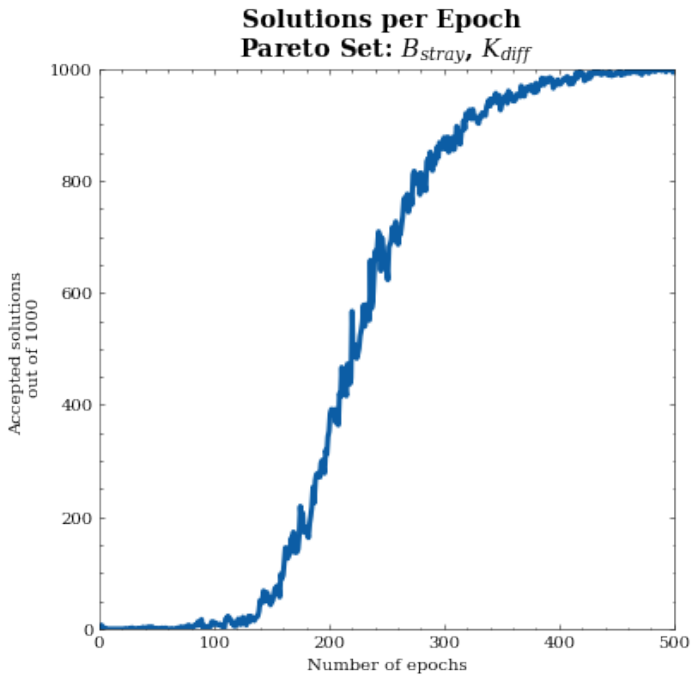

Fig. 4.6: Number of accepted solutions passing thresholds for objective functions O per epoch for the Pareto set of B_{stray} and K_{diff}.

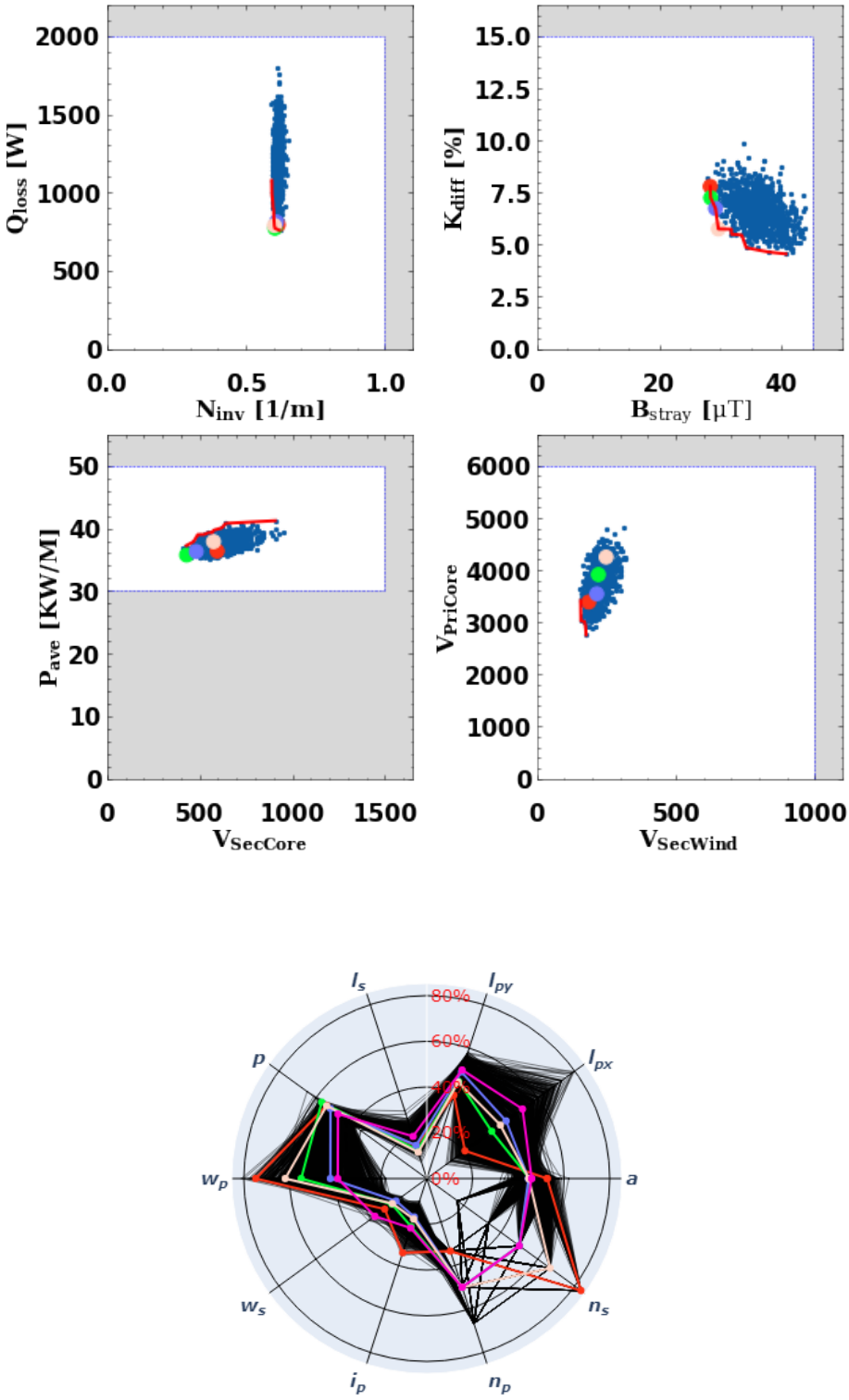

Fig. 4.7: Solutions for objective functions O (top), and design parameters DS_i (bottom). The top 5 performing Pareto optimal points for B_{stray}, are shown for the set O_a of B_{stray} and K_{diff}.

4.4.2 N_{inv} Q_{loss}

The loss function with the Pareto set of N_{inv} and Q_{loss} learned to produce the least accepted designs out of the Pareto optimal set loss functions, with the trained generator producing 484 solutions out of 1000. However, significant improvements are made in the performance of some of the objective functions. These improvements can be seen in much lower values for Q_{loss} and B_{stray} as well as a long extended Pareto front for the set of B_{stray} and K_{diff}. This extended Pareto front can offer a greater diversity of solutions for the design parameters.

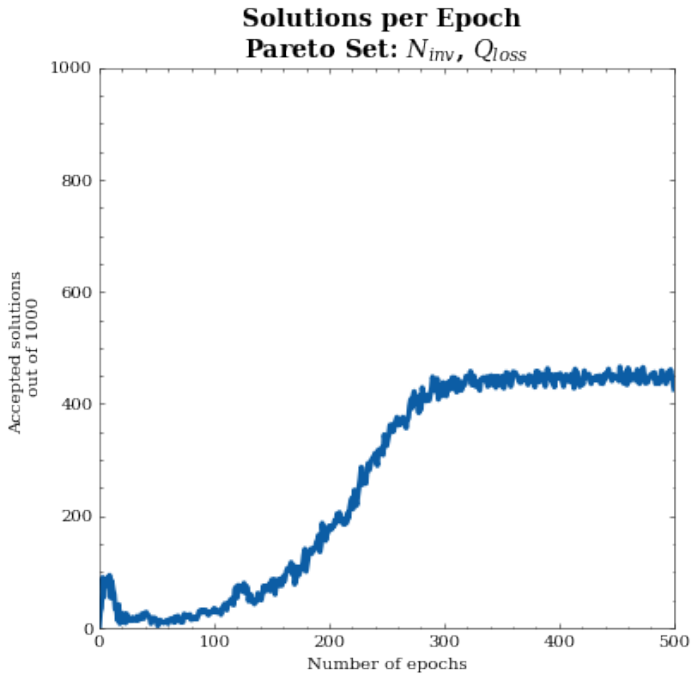

Fig. 4.8: Number of accepted solutions passing thresholds for objective functions O per epoch for the Pareto set of N_{inv} and Q_{loss}.

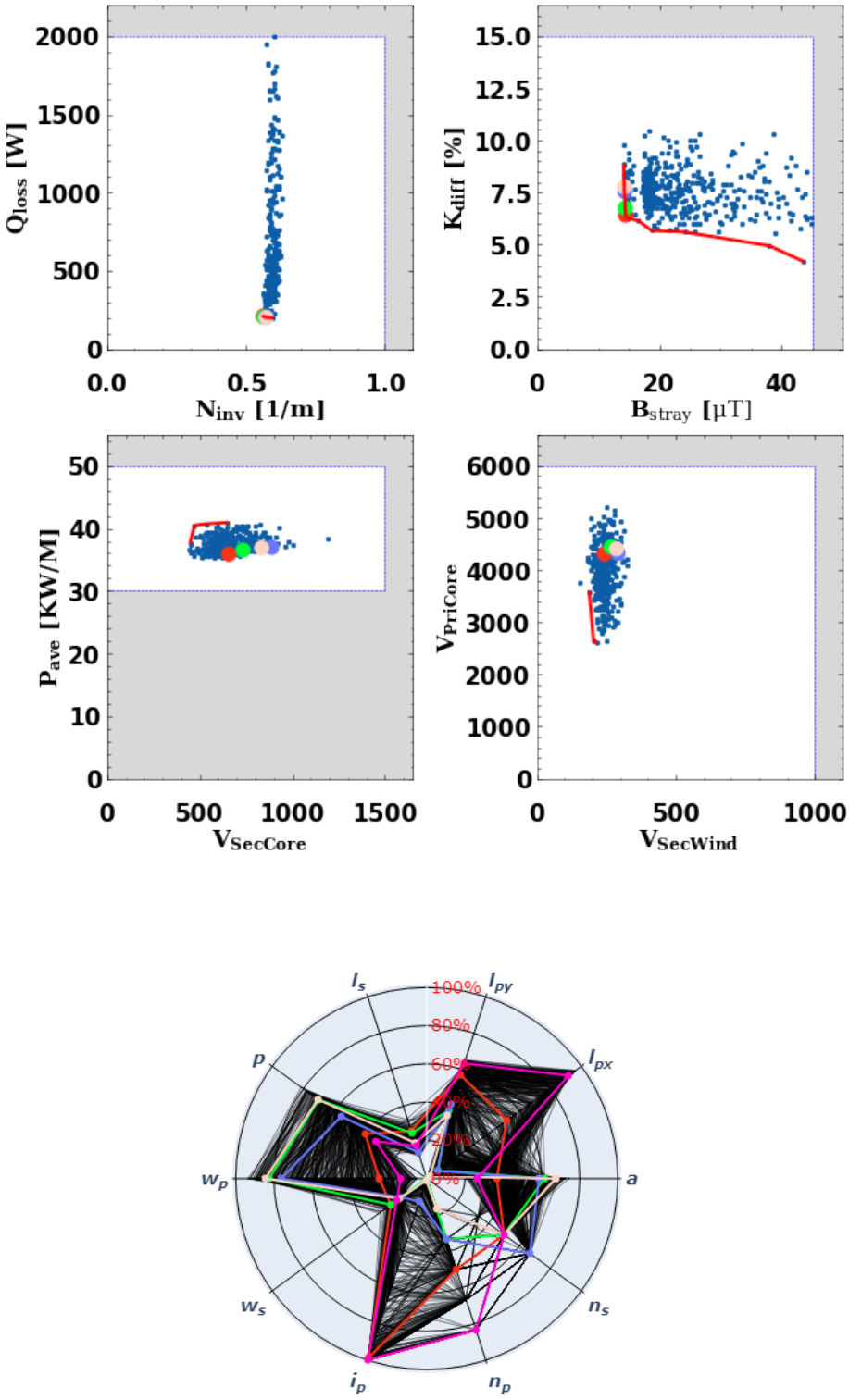

Fig. 4.9: Solutions for objective functions O (top), and design parameters DS_i (bottom). The top 5 performing Pareto optimal points for Q_{loss}, are shown for the set O_o of N_{inv} and Q_{loss}.

4.4.3 $V_{SecCore}$ P_{ave}

The loss function with the Pareto set of $V_{SecCore}$ and P_{ave}. The trained generator produced 952 solutions out of 1000. This loss function performed similarly in the objective space to the Product-of-Means loss function.

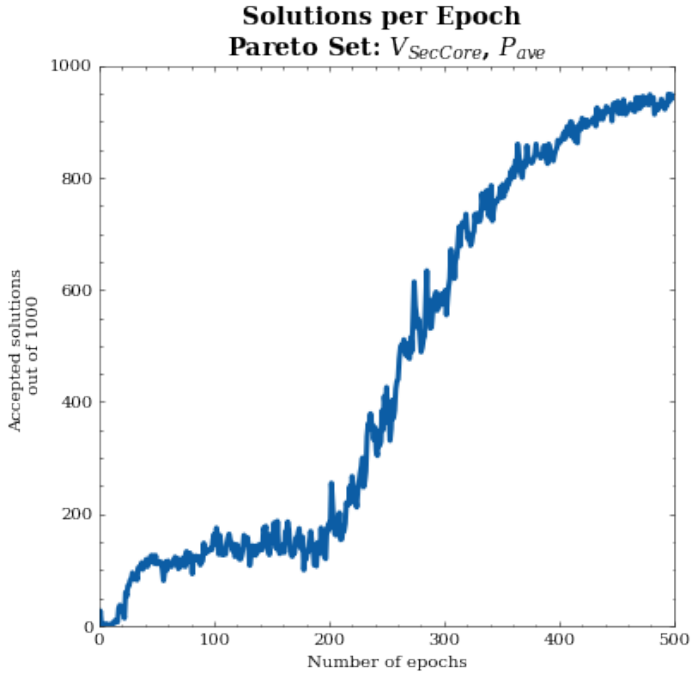

Fig. 4.10: Number of accepted solutions passing thresholds for objective functions O per epoch for the Pareto set of $V_{SecCore}$ and P_{ave}.

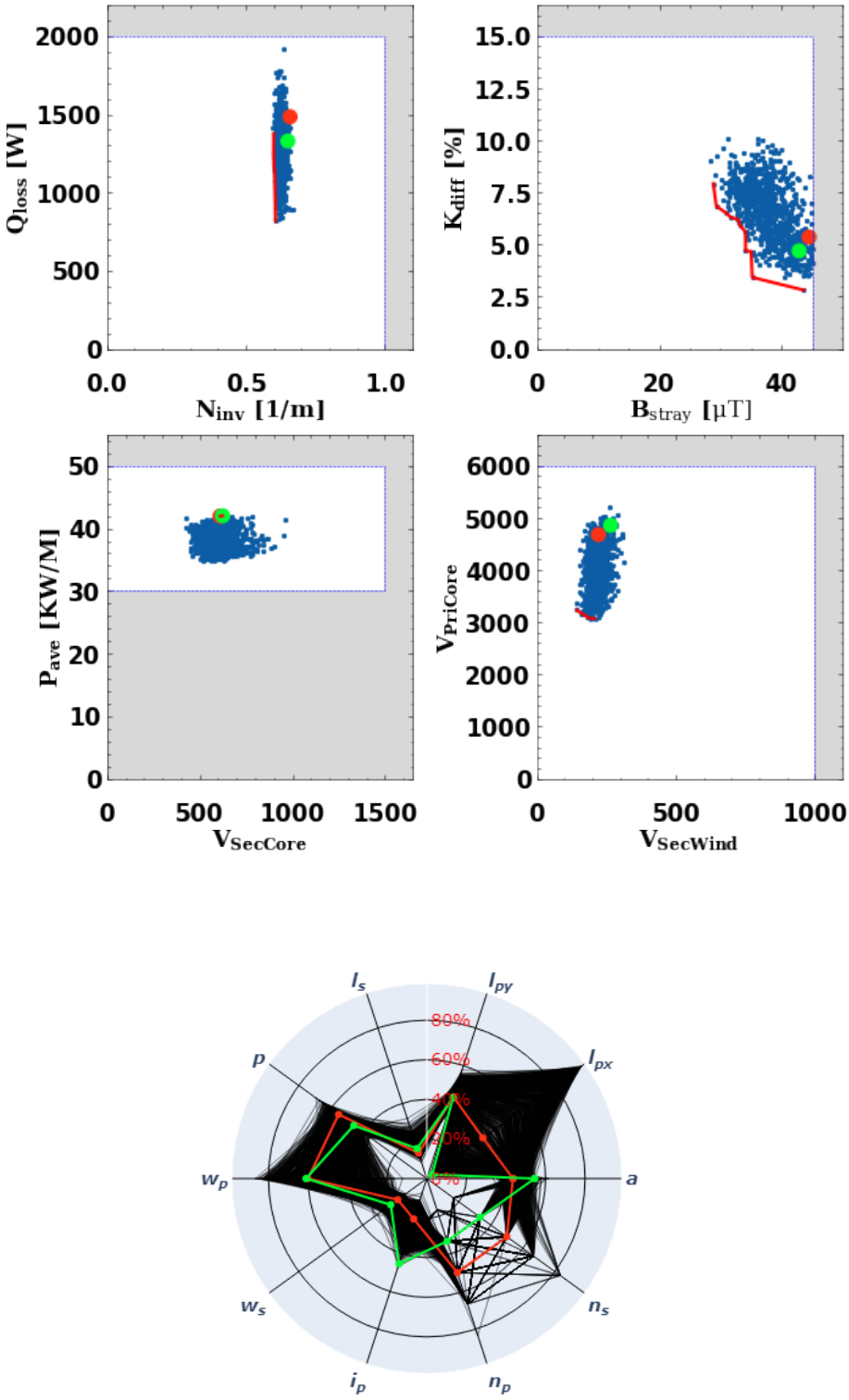

Fig. 4.11: Solutions for objective functions O (top), and design parameters DS_i (bottom). The generator only produced two points in the set O_o of $V_{SecCore}$ and P_{ave}.

4.4.4 $V_{SecWind}$ $V_{PriCore}$

The loss function with the Pareto set of $V_{SecWind}$ and $V_{PriCore}$. The trained generator produced 978 solutions out of 1000. This loss function had perhaps some slight improvements for the objective $V_{SecWind}$ compared to the Product-of-Means loss function.

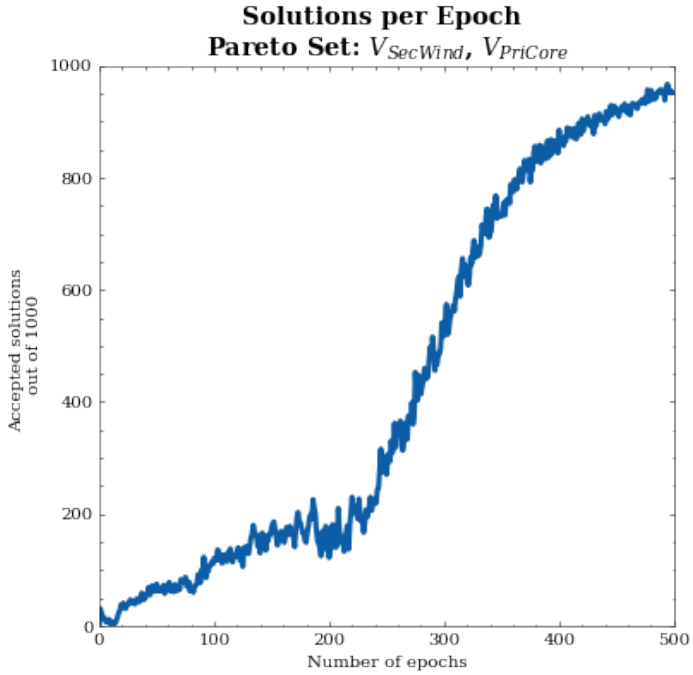

Fig. 4.12: Number of accepted solutions passing thresholds for objective functions O per epoch for the Pareto set of $V_{SecWind}$ and $V_{PriCore}$.

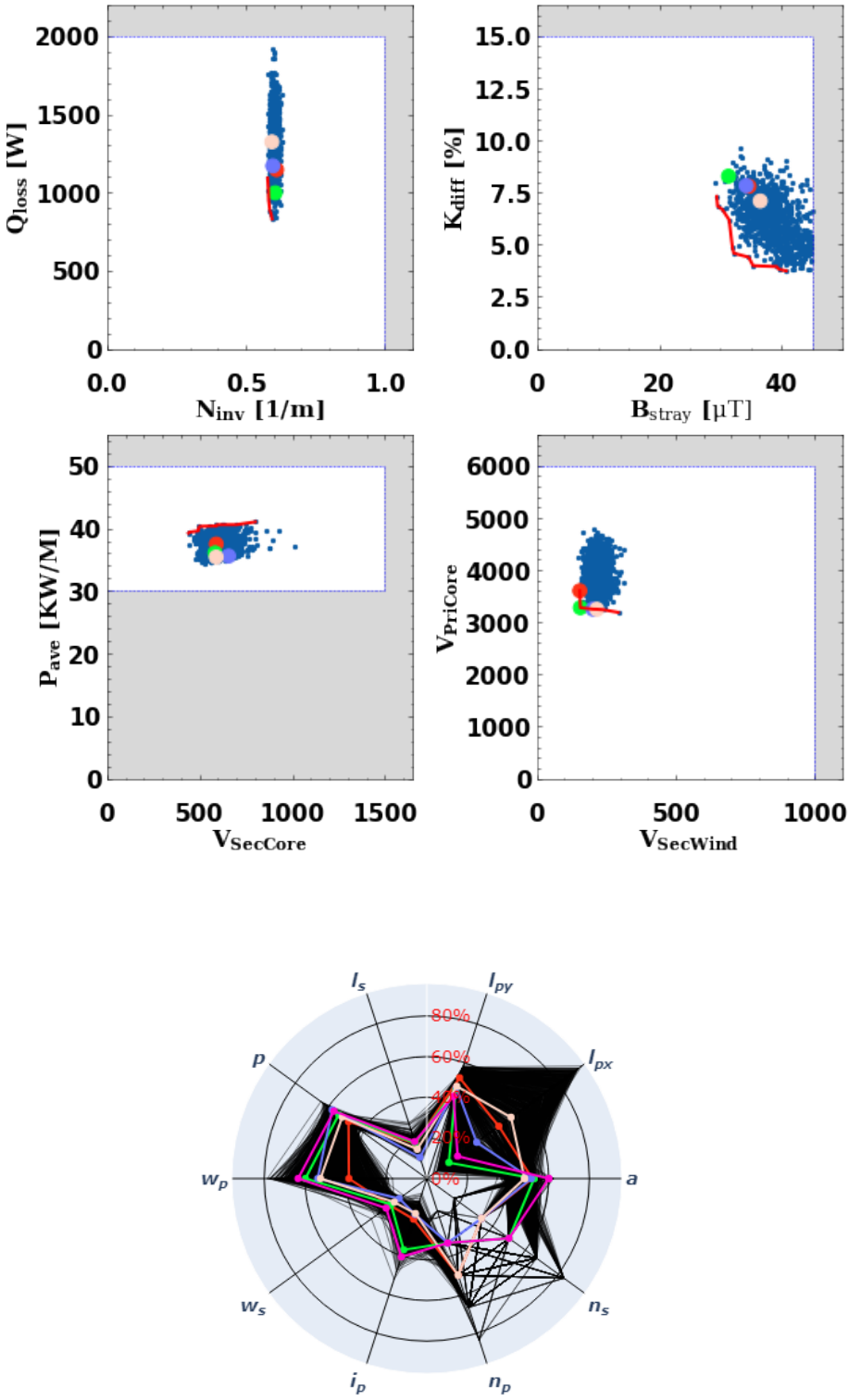

Fig. 4.13: Solutions for objective functions O (top), and design parameters DS_i (bottom). The top 5 performing Pareto optimal points for $V_{SecWind}$, are shown for the set O_o of $V_{SecWind}$ and $V_{PriCore}$.

4.5 Mean-of-Products Loss

The Mean-of-Products Loss function trained generator produced the least amount of accepted designs after training, with the exception of the random network, at 84 out of 1000. However, it does outperform some of the other loss functions for the objectives of K_{diff}, P_{ave} and $V_{PriCore}$. This shows that some trade-offs in the ability of the generator to produce accepted designs can be made for performance in certain objectives.

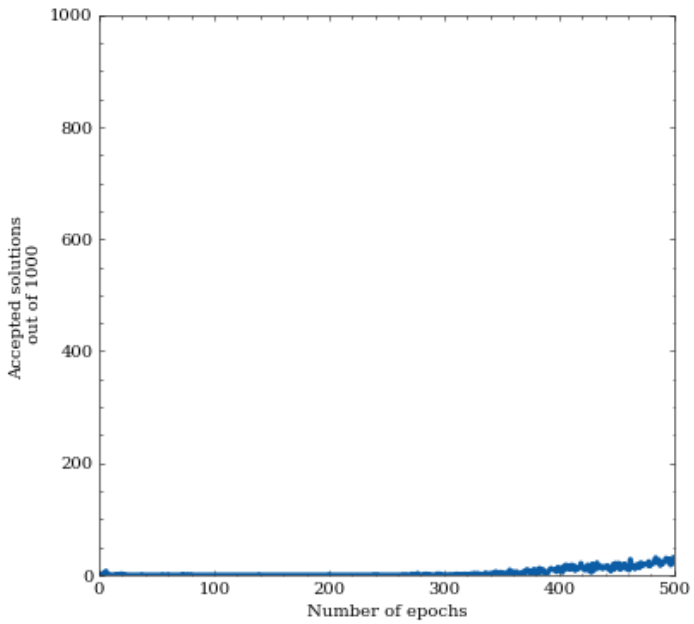

Fig. 4.14: Number of accepted solutions passing thresholds for objective functions O per epoch for the Mean-of-Products loss.

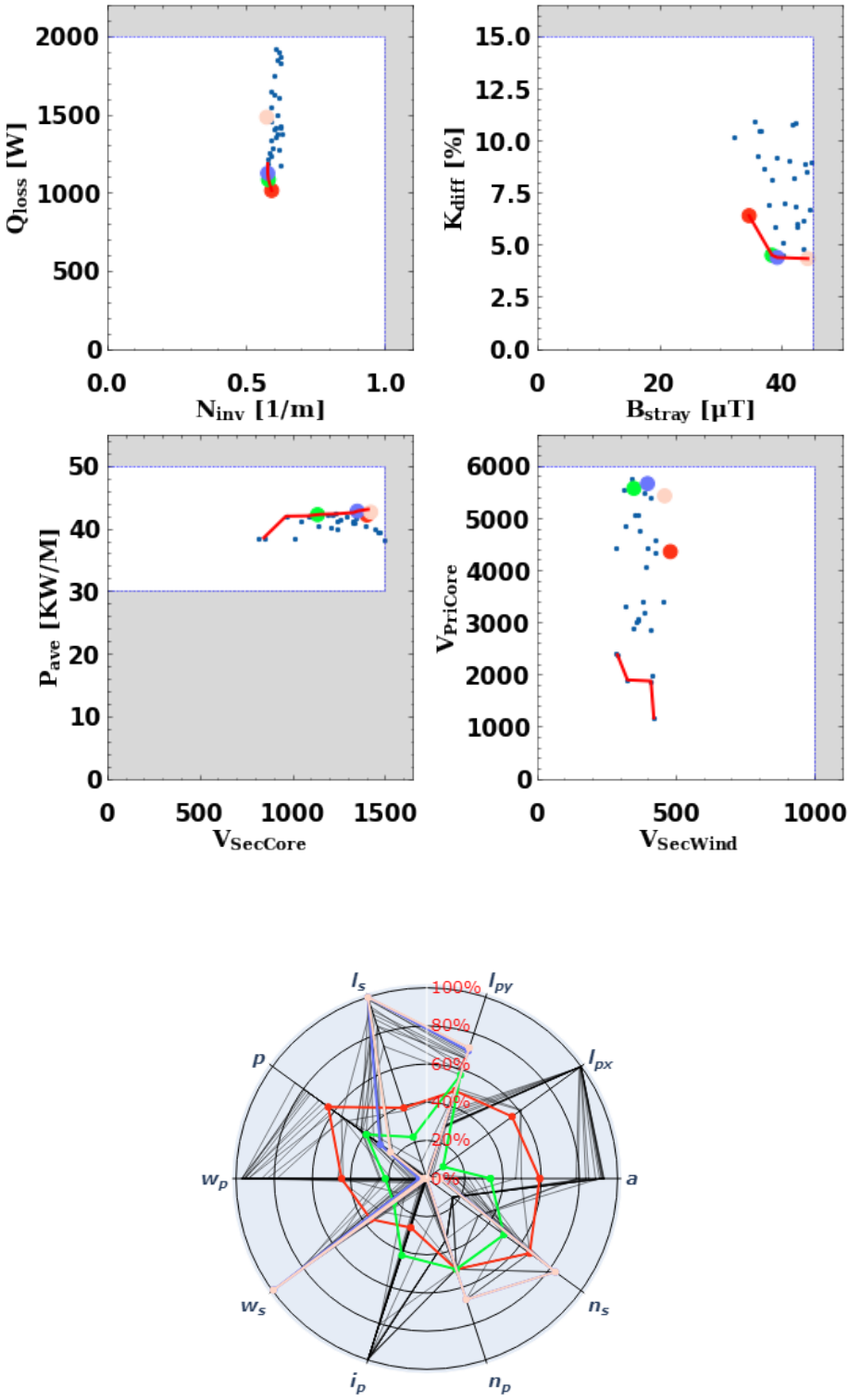

Fig. 4.15: Solutions after training with the Mean-of-Products loss function for objective functions O (top), and design parameters DS_i (bottom). The top 5 performing Pareto optimal points for B_{stray}, are shown for the set O_a of B_{stray} and K_{diff}.

4.6 Minimum-of-Products Loss

The Minimum-of-Products loss function trained generator produced 121 solutions out of 1000. It has significant improvements over other loss functions for the objectives of B_{stray}, Q_{loss}, N_{inv}, $V_{SecCore}$, $V_{SecWind}$ and $V_{PriCore}$.

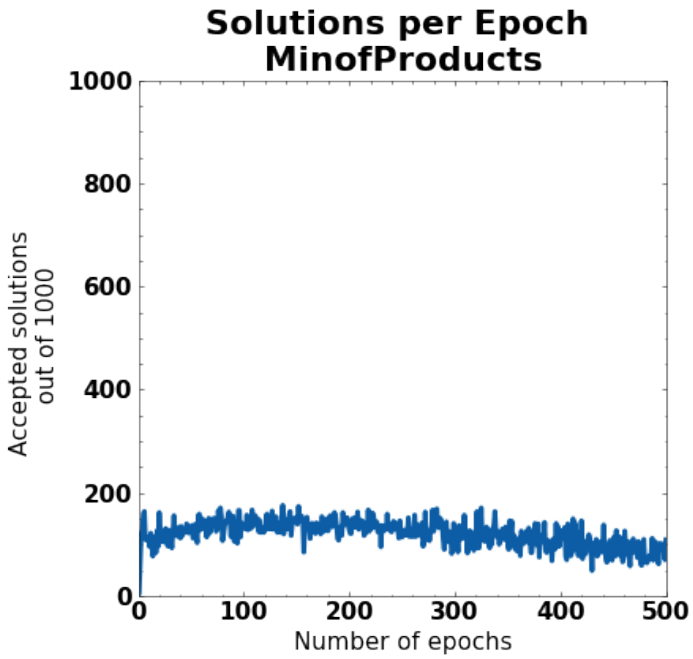

Fig. 4.16: Number of accepted solutions passing thresholds for objective functions O per epoch for the Minimum-of-Products Loss.

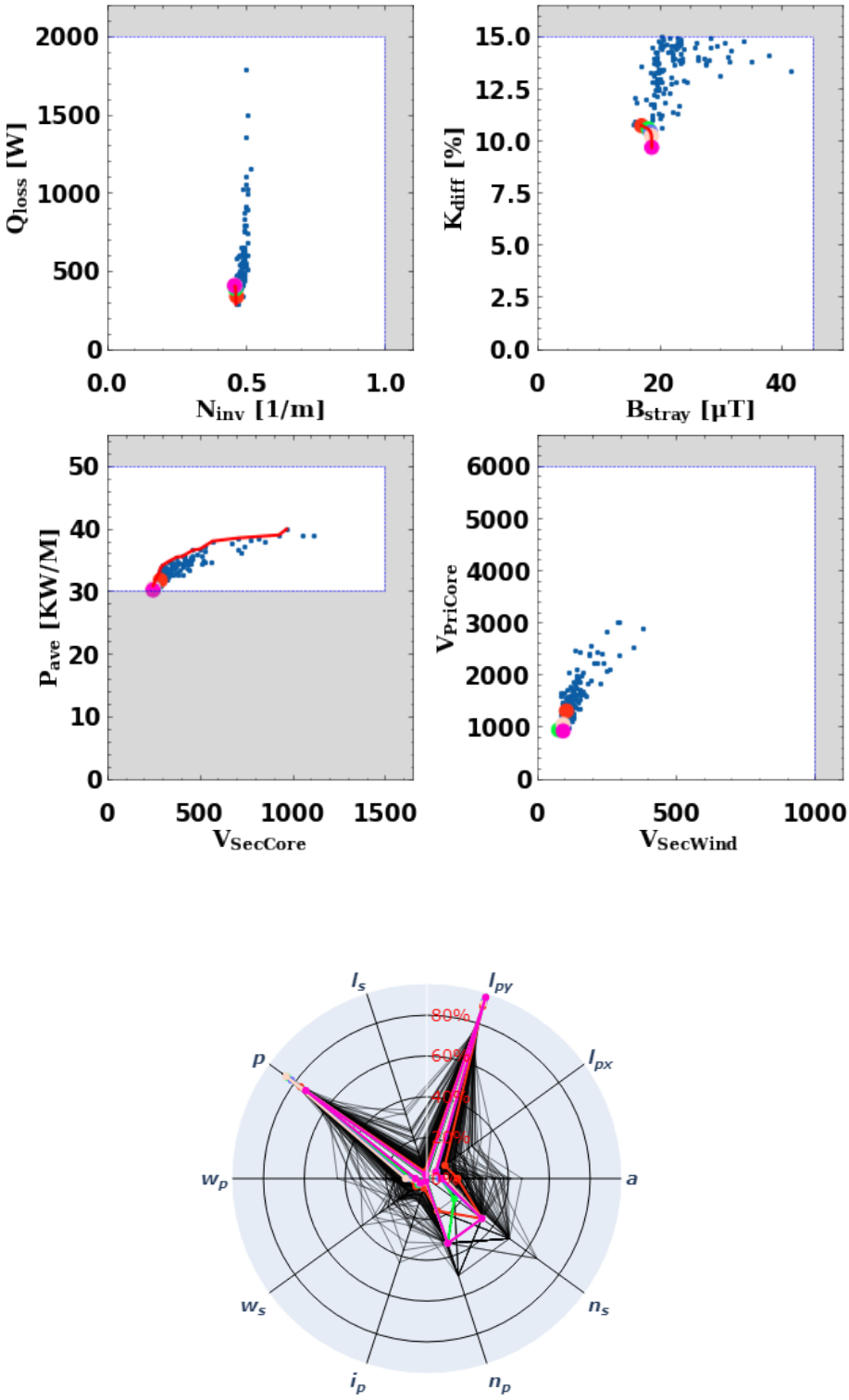

Fig. 4.17: Solutions after training with the Minimum-of-Products Loss function for objective functions O (top), and design parameters DS_i (bottom). The top 5 performing Pareto optimal points for B_{stray}, are shown for the set O_a of B_{stray} and K_{diff}.

4.7 Performance and Design Selection

The included loss functions offer a variety of performance across the objectives and their corresponding design parameters. This diversity of solutions can help engineers choose specific designs for their application. The generator trained with the Minimum-of-Products loss function offers the best performance for many of the objectives. For this reason, it was selected and a design solution was chosen for visualization, seen in red in Figure 4.19. For this solution, an example design configuration was drawn to scale for the primary and secondary coils, shown in Figure 4.20. Additionally, the scatter plot matrix of objectives after training with the Minimum-of-Products Loss is shown below in Figure 4.18. This visualization can be helpful in identifying possible pairwise objectives of interest that can be improved on well together.

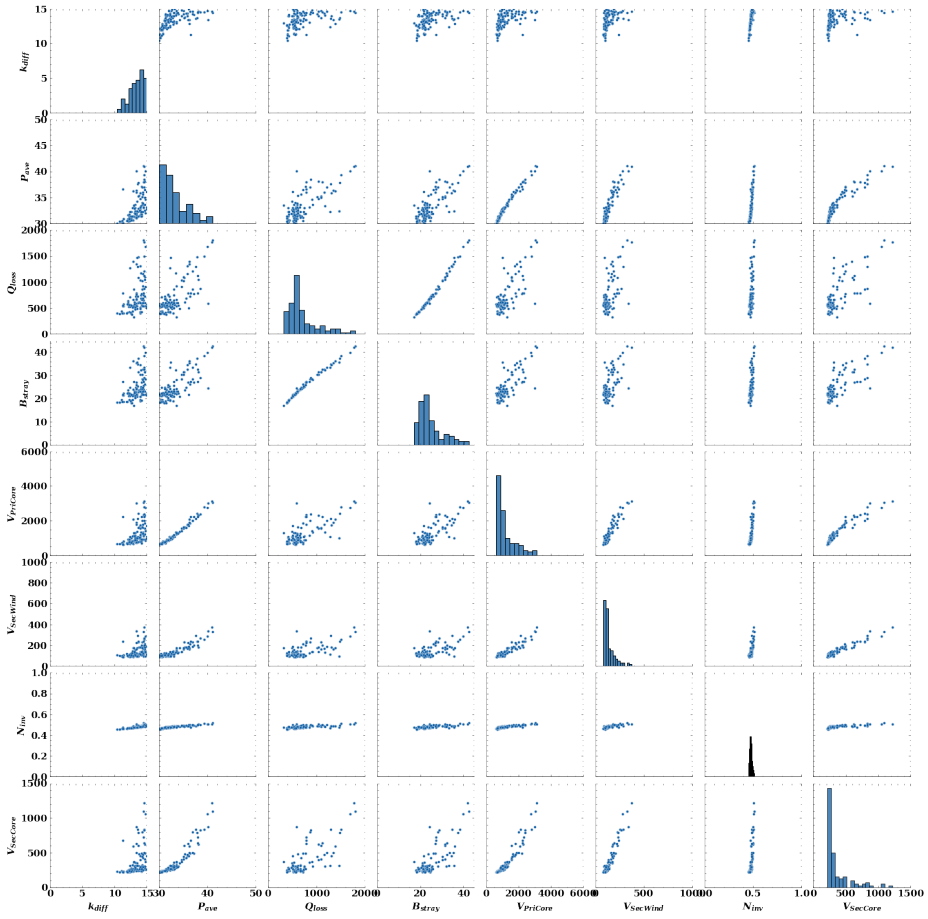

Fig. 4.18: Scatterplot Matrix of the objective functions O_o after training with Minimum-of-products loss function.

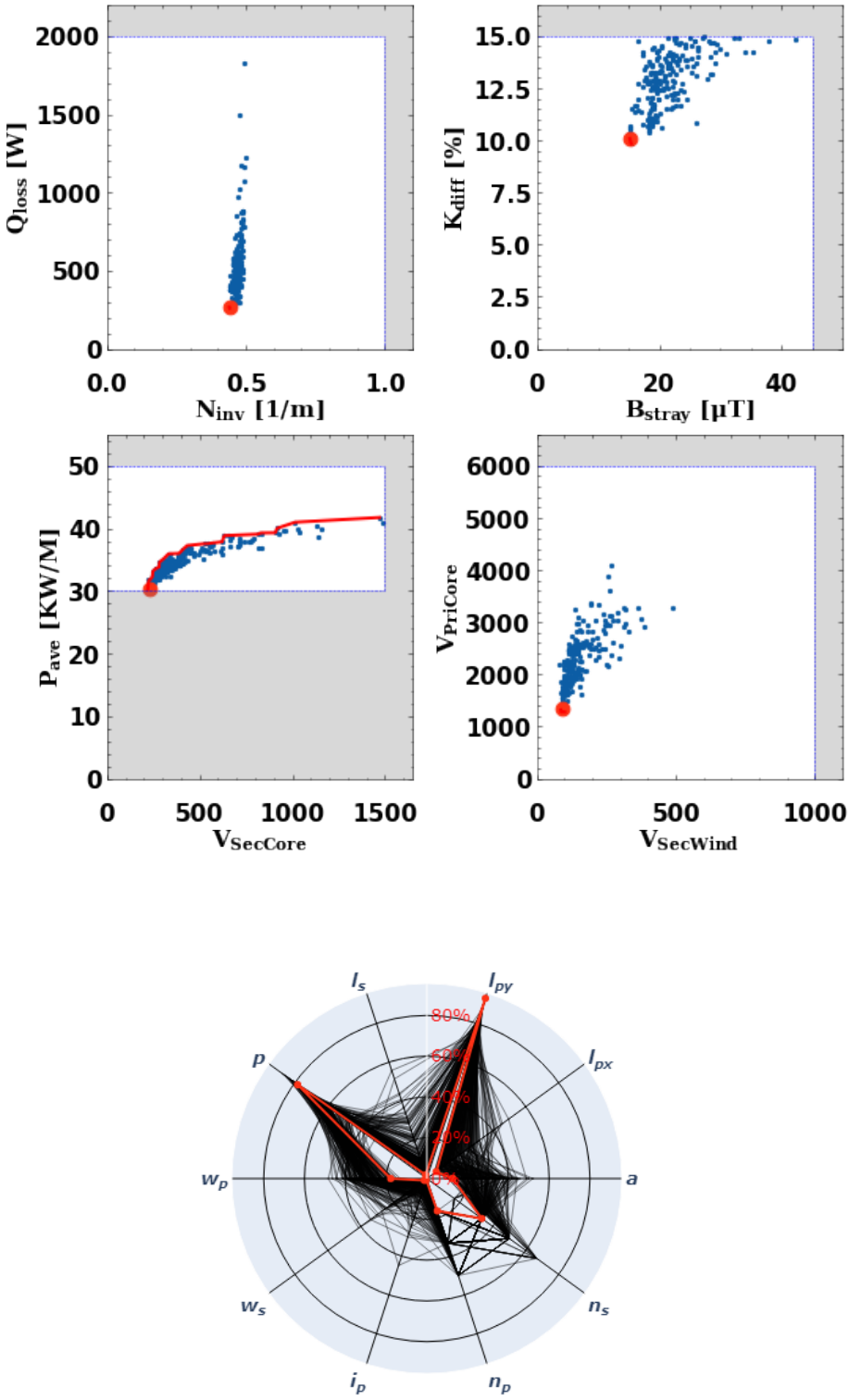

Fig. 4.19: The red solution is the selected design for representation and is within the Pareto set of Q_{loss} and N_{inv}. Solutions after training with the Minimum-of-Products Loss function for objective functions O (top), and design parameters DS_i (bottom).

An example coil diagram, Figure 4.20 was made for the selected red solution from Figure 4.19. This selected design is drawn to scale and has design parameters near the end ranges. This result for the design parameters is significantly different from training involving other loss functions, such as Product-of-Means. This gives engineers an opportunity to select from a more diverse selection of coil designs for performance and diversity in the case that manufacturing constraints are a consideration.

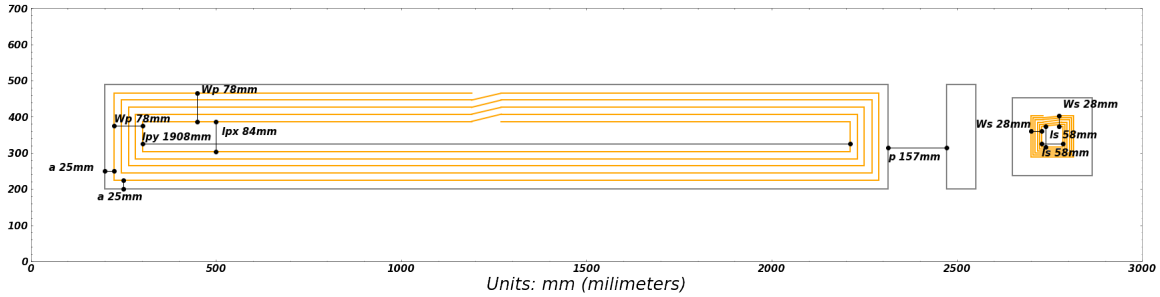

Fig. 4.20: Functional coil design to scale of primary coil (left) and secondary coil (right). Dimensions selected from red solution in Figure 4.19.

CHAPTER 5

Conclusion and Future Work

Dynamic inductive power transfer systems allow for wireless energy transfer and can be applied to charge electric vehicles in a more convenient and efficient manner. A DIPT system is comprised of two coils of wire, the secondary coil within the EV and the primary coil fixed in the roadway. Previous methods of finding well-performing designs relied upon slower optimization and testing methods. The application of generative neural networks was able to greatly expedite the process of generating near-optimal designs. At first a generator network is able to take random Gaussian noise and produce sets of design parameters. These designs can then be fed to a simulator network which was pre-trained using conventional numerical analytical methods (FEM simulations) and is fully differentiable for back-propagation. This simulator network produces the spatiotemporal behavior of DIPT designs under operation and can be used to calculate objective functions.

Objective functions represent analytical performance and safety metrics with thresholds to pass to be considered an accepted solution and can be used to evaluate the performance of designs. Objective functions are then further optimized for increases in performance and diversity of solutions. However, objectives can not all be improved simultaneously and trade-offs are made for performance in one objective versus another.

In order to enable learning and improve optimization during training of the GNN, various novel loss functions were applied. The loss functions can change the distribution of solution performance for the objectives and create a diversity of sets of associated design parameters for engineers to choose from. The methodologies presented utilized several approaches, including directly utilizing generated Pareto points of objective sets. The presented simulator and evaluator GNN in combination with the loss functions represent a new method to significantly improve diversity, optimization and reduce computational time when searching for new design solutions, when a fully-differentiable set of calculations for an engineering application is present.

Several avenues for future work are present. The effect of different architecture and depth of

the hidden layers of the GNN are of interest for increasing performance and reducing computational time. Different loss functions have a large effect on learning and testing new methods may lead to new features and improved learning of the network. Potential improvements could be made with testing of other optimizers and methods to potentially reach more global minima for objective functions. Finally, this method can be applied for use in many other generative model and optimization applications. The code for this application contains some private data and has not currently been made public.

REFERENCES

[1] S. Inoue, R. Nimri, A. Kamineni, and R. Zane, "A new design optimization method for dynamic inductive power transfer systems utilizing a neural network," in *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2021, pp. 1496–1501.

[2] I. Santos, L. Castro, N. Rodriguez-Fernandez, Torrente-Patiño, and A. Carballal, "Artificial Neural Networks and Deep Learning in the Visual Arts: a review," *Neural Computing and Applications*, vol. 33, no. 1, pp. 121–157, Jan. 2021. [Online]. Available: https://doi.org/10.1007/s00521-020-05565-4

[3] A. Oussidi and A. Elhassouny, "Deep generative models: Survey," in *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Apr. 2018, pp. 1–8.

[4] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, "DiffTaichi: Differentiable Programming for Physical Simulation," Feb. 2020, arXiv:1910.00935 [physics, stat]. [Online]. Available: http://arxiv.org/abs/1910.00935

[5] J. Andersson, "A survey of multiobjective optimization in engineering design," *Department of Mechanical Engineering, Linktjping University. Sweden*, 2000.

[6] A. V. Zykina, "A lexicographic optimization algorithm," *Automation and Remote Control*, vol. 65, no. 3, pp. 363–368, 2004.

[7] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[8] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[9] P. Moscato, C. Cotta, and A. Mendes, "Memetic algorithms," in *New optimization techniques in engineering*. Springer, 2004, pp. 53–85.

[10] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[11] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[13] I. Albuquerque, J. Monteiro, T. Doan, B. Considine, T. Falk, and I. Mitliagkas, "Multi-objective training of generative adversarial networks with multiple discriminators," in *International Conference on Machine Learning*. PMLR, 2019, pp. 202–211.

[14] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications," *Theoretical Computer Science*, vol. 425, pp. 75–103, 2012.

[15] X. Chen, X. Chen, W. Zhou, J. Zhang, and W. Yao, "The heat source layout optimization using deep learning surrogate modeling," *Structural and Multidisciplinary Optimization*, vol. 62, no. 6, pp. 3127–3148, 2020.

[16] B. Li, C. Huang, X. Li, S. Zheng, and J. Hong, "Non-iterative structural topology optimization using deep learning," *Computer-Aided Design*, vol. 115, pp. 172–180, 2019.

[17] M. Ruchte and J. Grabocka, "Scalable pareto front approximation for deep multi-objective learning," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 1306–1311.

[18] S. Suzuki, S. Takeno, T. Tamura, K. Shitara, and M. Karasuyama, "Multi-objective bayesian optimization using pareto-frontier entropy," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9279–9288.

[19] "Wireless power transfer for light-duty plug-in/electric vehicles and alignment methodology," *SAE-J2954 standard industry-wide specification*, 2022.