UNIVERSITÀ
DI PAVIA

UNIVERSITY OF PAVIA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING
MASTER'S DEGREE IN INDUSTRIAL AUTOMATION ENGINEERING LM-25

MASTER THESIS

TITLE in English

DESIGN AND OPTIMIZATION OF ELECTROMAGNETIC COILS USING DEEP NEURAL NETWORKS

TITLE in Italian

PROGETTAZIONE E OTTIMIZZAZIONE DI BOBINE ELETTROMAGNETICHE MEDIANTE RETI NEURALI PROFONDE

Candidate – SOWMIYANARAIN SAMPATH

Supervisor: Prof. MARIA EVELINA MOGNASCHI

Academic Year- 2025/2026

ABSTRACT

Electromagnetic coil design is traditionally driven by iterative finite-element analysis (FEA), which becomes prohibitively expensive under tight geometric, material, and performance constraints. This thesis proposes an end-to-end, learning-assisted pipeline based on deep neural networks to accelerate coil design while preserving fidelity. A feed-forward surrogate learns the mapping from design parameters to field- and performance-level objectives. To explore the feasible manifold efficiently, a generative adversarial model proposes candidate designs, pre-trained on broad synthetic data and fine-tuned to find good solutions in pareto sense. Multi objective optimization (e.g., Field Uniformity vs. copper losses) is conducted with the surrogate model in the Deep Learning loop, based on the FEM model. The approach yields substantial reductions in FEA calls and wall-time while maintaining agreement with high fidelity simulations of coils. Using consistent normalization across the pipeline and a lightly conditioned GAN (cGAN), the generated design cloud overlaps the real values manifold. The top-20 cGAN candidates concentrate in the f_1/f_2 trade-off region, matching the Pareto trends in the dataset. Overall, the pipeline cuts FEM evaluations for screening while preserving agreement with high-fidelity coil simulations. The work demonstrates a practical route to rapid, EM coil design and outlines extensions to broader electromagnetics benchmarks.

Keywords: electromagnetic coils, surrogate modeling, deep neural networks, uncertainty quantification, GANs, physics-informed learning, multi-objective optimization, finite-element analysis.

ACKNOWLEDGMENTS

Upon reaching the conclusion of this thesis, I wish to express my heartfelt gratitude to the many individuals whose guidance, encouragement, and support have been fundamental in shaping this work.

First and foremost, I would like to express my deepest appreciation to Professor Maria Evelina Mognaschi. Her belief in my potential and her decision to recruit me for a curricular internship provided me with the opportunity to explore and learn the fundamental concepts of Machine Learning, which became the foundation of this thesis. Her insightful guidance, patience, and constant encouragement have been invaluable throughout this journey, and I am profoundly grateful for her trust and mentorship.

I would also like to extend my thanks to the University of Pavia for providing a stimulating academic environment and the resources that have made this research possible.

Finally, my deepest gratitude goes to my family and friends, whose unwavering belief in me and constant support through both challenges and successes have been my greatest strength. Their encouragement has kept me motivated at every step of this journey.

In essence, while this thesis reflects my individual effort, it is also the result of the trust, wisdom, and support I have received from those around me. For this, I am truly thankful.

TABLE OF CONTENTS

List of Figures

List of Tables

1.INTRODUCTION

1.1 Background and Motivation

The accurate and efficient design of electromagnetic (EM) devices has been a long-standing challenge in engineering, particularly when dealing with low-frequency applications such as inductors, transformers, and magnetic shielding systems. In such devices, performance is strongly influenced by the geometry and material properties of components, especially coil configurations. Traditional design methods rely heavily on solving Maxwell's equations using numerical approaches like the Finite Element Method (FEM), which can be computationally expensive when repeated evaluations are needed for optimization purposes.

Recent decades have witnessed growing interest in surrogate modeling techniques — computational approximations of complex physical simulations — to accelerate the design and analysis process. These models provide fast approximations of system behaviour and are especially useful in design loops where iterative evaluation is necessary.

In parallel, **Machine Learning (ML)** and particularly **Deep Neural Networks (DNNs)** have become powerful tools in computational science. Their ability to approximate nonlinear functions with high accuracy makes them excellent candidates for building surrogate models in electromagnetics. Among them, **Feed Forward Neural Networks (FFNNs)** have been used as regressors to map device geometries (e.g., coil lengths) to key physical quantities such as field distribution, torque ripple etc.. Furthermore, **Generative Adversarial Networks (GANs)** have opened new possibilities by generating synthetic yet plausible design configurations that conform to learned distributions of high-performance solutions.

This thesis is motivated by the success of such methods in the context of electromagnetic design optimization. In particular, we are considering the **TEAM Workshop Problem 35** from the paper *"Neural metamodelling of fields: Towards a new deal in computational electromagnetics"[1]* as a

benchmark. The case study focuses on using **neural network-based surrogate models** and **data-driven generation techniques** to minimize field non-uniformity (denoted by f_1) through coil configuration optimization.

The work builds on recent literature, including the study by Prof. Paolo Di Barba, Prof. Maria Evelina Mognaschi, and Prof. Slawomir Wiak et al. (2022) which employed CNNs and GANs in MATLAB for field optimization in low-frequency electromagnetics. The novelty of this thesis lies in reimplementing and extending those ideas in Python using TensorFlow and combining **GAN-based design generation with FFNN-based surrogate evaluation**, forming an end-to-end data-driven optimization pipeline.

This approach aims to reduce computational costs, enhance precision, and demonstrate how modern ML methods can effectively support real-world electromagnetic problems — a domain traditionally dominated by physics-based solvers.

1.2 Traditional Methods in Designing Electromagnetic Devices

The design of electromagnetic (EM) devices like transformers, actuators, sensors, and inductive charging systems relies on the application of classical field theory, primarily grounded in Maxwell's equations.

$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}$	**GAUSS'S LAW**
$\nabla \cdot \mathbf{B} = 0$	**GAUSS'S LAW FOR MAGNETISM**
$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$	**FARADAY'S LAW OF INDUCTION**
$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$	**AMPERE-MAXWELL LAW**

Table 1.2 Maxwell's equations

Maxwell's partial differential equations (PDEs) describe the behaviour of electric and magnetic fields under various configurations and material conditions. However, in practical device design, especially for complex geometries or nonlinear materials, analytical solutions are not feasible. This has led to adoption of numerical simulation methods to approximate the solutions of Maxwell's equations in complex scenarios.

Among the most widely used numerical techniques are the **Finite Element Method (FEM)**, the **Boundary Element Method (BEM)**, and the **Finite Difference Time Domain (FDTD)** method. While the BEM method is useful for open-boundary problems, the FDTD method is suited for time varying fields and wave propagation, we use FEM for complex geometries and material properties. FEM, in particular, has become the standard tool for static and dynamic field analysis in both linear and nonlinear materials. It divides the geometry into a mesh of discrete elements over which the governing equations are approximated. This method provides a high degree of flexibility and accuracy, but it can be computationally intensive, especially when:

- fine meshes are required for precision,
- 3D geometries are involved,
- material nonlinearities (e.g., magnetic saturation) must be captured,

When multiple iterations are needed for optimization, the computational burden can become prohibitive. To address these challenges, designers often use model order reduction techniques, such as **Proper Orthogonal Decomposition (POD)** or response surface modelling, to accelerate design loops. Still, these methods often require an extensive number of full-order simulations for initial training or interpolation.

Moreover, conventional optimization in EM design has relied on techniques like **genetic algorithms, particle swarm optimization, and gradient-based methods**, which are computationally expensive due to the repeated use of FEM solvers during each iteration. The choice of optimization algorithm often depends on the nature of the design problem, the number of design variables, and the

computational resources available. This becomes particularly limiting in **multi-objective optimization scenarios**, such as those encountered in **TEAM Workshop Problem 35**[1], where trade-offs between field uniformity and device volume (or power losses) must be explored.

Despite the maturity of these methods, there is growing recognition that traditional solvers and optimizers are no longer sufficient alone for the demands of real-time, multi-objective, and data-driven design in complex electromagnetic environments. These limitations are a key motivation behind the integration of machine learning and neural surrogate models, which offer the potential for orders-of-magnitude speedups in the evaluation phase, and seamless integration into modern automated design frameworks.

1.3 The Rise of Machine Learning in Electromagnetics

The intersection between machine learning (ML) and computational electromagnetics (CEM) has drawn increasing attention from both academia and industry due to growing complexity of electromagnetic (EM) problems and demand for faster design cycles. While traditional EM analysis relies on numerical solvers such as the FEM, they are computationally intensive, especially for design, optimization, and inverse problems. Whereas Machine learning offers the ability to learn nonlinear mappings between inputs and outputs from data, providing a new way of approach for accelerating EM simulation and design workflows.

A major turning point has been the availability of deep learning techniques and their proven ability to generalize across high-dimensional and nonlinear function spaces. **Feedforward neural networks (FFNNs)** have been successfully used as surrogate models to approximate the mapping from geometric or material parameters (e.g., coil lengths, permittivities) to target quantities like magnetic field uniformity, impedance, or scattering coefficients. Once trained, these models can evaluate new designs in milliseconds, offering a significant speed advantage over traditional methods.

More recent approaches extend beyond regression. **Convolutional Neural Networks (CNNs)** have demonstrated effectiveness in learning spatial field patterns or extracting features from simulation outputs, particularly in image-like data representations of EM fields. Even more notably, **Generative Adversarial Networks (GANs)** have emerged as powerful tools for **generating synthetic EM device configurations** that resemble those optimized by conventional methods, reducing the reliance on expensive simulation-generated datasets.

This shift toward machine learning is not simply about speed; it reflects a deeper change in how electromagnetic problems are approached. For example, **data-driven inverse design**, in which neural networks directly generate geometries that meet desired EM performance, has become a new frontier. Such models are capable of navigating high-dimensional design spaces where gradient-based methods or even global optimization algorithms struggle.

In the domain of **low-frequency electromagnetics**, where magnetic field shaping and coil optimization are prominent, machine learning has proven especially effective. Studies have shown how deep learning can supplement traditional solvers to build fast, reliable surrogate models. This is particularly relevant for benchmark problems like TEAM Workshop Problem 35[1], where the objective is to minimize the non-uniformity of the magnetic field (f_1) through coil geometry optimization.

As computational resources continue to grow and simulation-generated datasets become more accessible, the use of machine learning in electromagnetics is expected to become not just an enhancement, but a foundational component of modern design frameworks.

2. Deep Neural Networks

2.1 Fundamentals of Feedforward Neural Networks (FFNN)

Feedforward Neural Networks (FFNNs) are among the most fundamental architectures, one of the earliest types of artificial neural networks which served as a building block for many more advanced models. An FFNN is a supervised learning model composed of layers of interconnected nodes (or "neurons"), where information flows in one direction only i.e, from input to output. These networks are best suited for function approximation, regression, and classification tasks, and they have been widely applied in scientific computing, including the modeling of electromagnetic systems.

2.1.1 Architecture and Operation

An FFNN usually consists of -

- Input layer - The input layer consists of neurons that receive the input data. Each neuron in the input layer represents a feature of the input data.
- Hidden Layers - One or more hidden layers are placed between the input and output layers. These layers are responsible for learning the complex patterns in the data. Each neuron in a hidden layer applies a weighted sum of inputs followed by a non-linear activation function.
- Output Layer - The output layer provides the final output of the network. The number of neurons in this layer corresponds to the number of classes in a classification problem or the number of outputs in a regression problem.

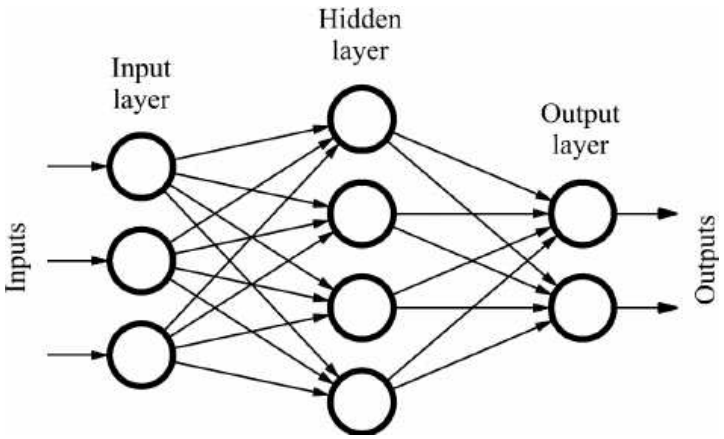

Fig2.1 FFNN Architecture

Each node computes an output using the following operation -

$$a^{(l)} = \sigma\left(W^{(l)}a^{(l-1)} + b^{(l)}\right) \tag{2.1}$$

where:

- $a^{(l)}$: output (activation) of layer l
- $W^{(l)}$: weight matrix between layers $l-1$ and l
- $b^{(l)}$: bias vector
- σ: activation function (e.g., ReLU, tanh, sigmoid)

Activation Functions

Activation functions introduce non-linearity into the network enabling it to learn and model complex data patterns.

Common activation functions include:

Sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

Compresses input values into the range (0,1) introducing smooth non-linearity but suffers from vanishing gradients for large positive or negative inputs.

Tanh:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

Maps inputs to (-1,1) offering stronger gradients around zero than sigmoid, which helps faster convergence in hidden layers.

ReLU:

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.4}$$

Introduces sparsity by outputting zero for negative values and linear response for positives, reducing vanishing gradient problems and speeding up training

2.1.2 Training Process

The training of FFNNs involves minimizing a loss function (e.g., mean squared error or mean absolute error) using gradient-based optimization. The most common algorithm is **backpropagation** combined with optimizers like **stochastic gradient descent (SGD)** or **Adam**.

In supervised learning, training data consists of input-output pairs:

Input:

$$\mathbf{x} \in \mathbb{R}^n \tag{2.5}$$

x is an n-dimensional vector of real-valued features, which represents a single sample fed into the network

Target:

$$y \in \mathbb{R} \tag{2.6}$$

y is the real valued truth label for that sample

Prediction:

$$\hat{y} = f(\mathbf{x}) \tag{2.7}$$

$f(x)$ is the neural network's learned mapping with input x output the model estimate $\hat{y}$ as output. Training minimizes the difference between y and $\hat{y}$.

2.1.3 FFNN as a Surrogate Model

In the realm of scientific computing, further advancements have led to the rise of FFNNs as surrogate models which function as rapid approximate models that stand in for expensive simulations. Also, in the field of electromagnetics which has quite a history, they serve a critical role by mapping input design parameters like coil geometry to physical responses, field uniformity or losses. More so, once these models are trained, they perform near-instantaneous evaluation of new designs without solving Maxwell's equations numerically. Therefore, this shift makes it easier for engineers and researchers to explore efficiently. The below equation states that the surrogate FFNN maps 10 coil radii scalar input to give a scalar output prediction f_1.

$$\text{Coil geometry } \mathbf{x} \in \mathbb{R}^{10} \rightarrow f_1 \text{ value } \in \mathbb{R} \tag{2.8}$$

where the f_1 value represents a quantitative measure of magnetic field non-uniformity in a coil system. For background on optimization, regularization, and convolutional architectures relied on here, see the standard text Deep Learning (MIT Press) [2].

2.1.4 Advantages and Limitations

The below table summarizes the advantages and limitations of using FFNN as a surrogate model.

Advantages	Limitations
Fast evaluation once trained	Require large, high-quality training datasets
Flexible and capable of approximating nonlinear mappings	Can overfit if not regularized properly
Can be easily implemented with modern ML libraries (TensorFlow, PyTorch)	Lack spatial awareness (compared to CNNs)

Table 2.1 Advantages and Limitations of FFNN

2.2 Convolutional Neural Networks (CNN): Spatial Feature Extraction

Convolutional Neural Networks (CNNs) are a specialized type of deep neural network designed to process structured grid-like data such as images, time series, or spatial sequences. Originally developed for computer vision tasks, CNNs have found widespread application across scientific computing, including the modeling of electromagnetic fields, where spatial dependencies and local feature patterns are critical.

Unlike Feedforward Neural Networks (FFNNs), which treat all input features independently, CNNs utilize local receptive fields, weight sharing, and spatial hierarchies to extract features from input data. This makes them particularly effective in recognizing spatial patterns or correlations across neighboring inputs.

2.2.1 Building Blocks of CNN

- **Convolutional Layers**: These apply filters (also called kernels) that slide over the input to detect local features. The output of a convolution operation between input X and a filter K is defined as:

$$Y(i,j) = \sum_m \sum_n X(i+m, j+n) \cdot K(m,n) \tag{2.9}$$

Where X is input feature map, K is convolution kernel/filter, Y is the output feature map. $i.j$ are the index of the output dataframe and m, n are the index positions of the kernel window which slides over X. At each index the weighted sum with K to output $Y(i,j)$.

In 1D, for vector inputs like coil lengths in our case, it reduces to -

$$Y(i) = \sum_m X(i+m) \cdot K(m) \tag{2.10}$$

Similar to 2D X is the 1D input signal, K is the 1D kernel, Y is the resulting feature map. i id the position of output and m scans the kernel elements.

- **Activation Functions**: After each convolution, a non-linear activation is applied. The most common is ReLU:

  $\mathrm{ReLU}(x) = \max(0, x)$

- **Pooling Layers**: These downsample the feature maps to reduce dimensionality and improve generalization. Max pooling, for example, selects the maximum value within a local region.
- **Fully Connected Layers**: After multiple convolution and pooling layers, the high-level feature representations are flattened and passed to dense layers for regression or classification.

2.2.2 CNN for 1D data

In the context of this thesis, CNNs are applied to 1D input vectors of coil lengths (10 values). Here, **1D convolutional layers (`Conv1D`)** are used instead of 2D image-based convolutions. This allows the network to learn **interactions between adjacent coil positions**, which will have a combined influence on electromagnetic performance.

CNNs offer a natural advantage over FFNNs in this scenario because FFNNs treat all input features as isolated while CNNs can exploit **spatial locality**, like how coil segment 3 might affect its neighbours 2 and 4.

CNNs have proven effective for:

- Learning **spatially correlated physical patterns**
- Capturing **field distributions** or device layouts
- Enhancing generalization in regression tasks with structured input

When used as surrogate models in electromagnetics, CNNs can approximate field solutions or map geometries to scalar quantities like f_1 error in our case, with better sample efficiency.

2.3 Generative Adversarial Networks (GAN): Data Generation and Augmentation

Generative Adversarial Networks (GANs), introduced by Ian Goodfellow et al. in 2014, are a class of deep learning architectures designed for **unsupervised learning** and **data generation**. GANs consist of two neural networks — a **generator** and a **discriminator** which are trained simultaneously in a **minimax game** framework. The goal of the generator is to produce data that resembles real data, while the discriminator learns to distinguish between real and generated data.

2.3.1 Architecture

- **Generator(G) -** creates synthetic data from random noise to produce data so realistic that the discriminator cannot distinguish it from real data.

$$G : \mathbb{R}^d \to \mathbb{R}^n \tag{2.11}$$

$$\mathbf{z} \sim p_z(\mathbf{z}) \quad \Rightarrow \quad \tilde{\mathbf{x}} = G(\mathbf{z})$$

Where:

$\mathbf{z}$: random latent vector

$G(\mathbf{z})$: generated coil configuration or image

p_z: prior distribution (usually Gaussian or uniform)

- **Discriminator(D) -** acts as a critic, evaluating whether the data it receives is real or fake.

$$D : \mathbb{R}^n \to [0, 1] \tag{2.12}$$

$$D(\mathbf{x}) = \text{probability that } \mathbf{x} \text{ is real}$$

2.3.2 Training Objective

GANs are trained via a two-player game. The **discriminator** maximizes its ability to distinguish real and fake samples, while the **generator** minimizes the discriminator's ability to detect its fakes.

The original **minimax loss function** is defined as:

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1 - D(G(\mathbf{z})))] \tag{2.13}$$

Where E is the statistical average

$\mathbb{E}_{z\sim p_z}$ average over latent noise z

$\mathbb{E}_{x\sim p_data}$ average of the term over real samples x drawn from the data distribution

$\mathbf{x} \sim p_{\text{data}}$ = real data distribution; $\mathbf{z} \sim p_z$ = latent noise

Converting to log turns it into binary cross-entropy which maximizes $D(x)$ score for real data and minimizes $(1 - D(G(z)))$ for generated fake values for sound classification of real vs generated.

Training alternates between - Updating **D** to better detect fakes and Updating **G** to better fool **D**

2.3.3 Applications in Electromagnetics

In the context of electromagnetic design, especially when training data is expensive to generate (e.g., FEM simulations), GANs can **augment data** or **generate new candidate solutions** that resemble high-quality designs.

In this thesis, GANs are used to **generate coil configurations** that match the distribution of optimal designs — in terms of low f_1 values — without the need for evaluating each configuration via a full simulation.

Once trained, the generator can produce thousands of valid design candidates from simple random noise input, making it a powerful tool for **design space exploration**.

2.3.4 GANs in Surrogate Modeling

In this work, GANs complement the surrogate rather than replace it: the generator learns the distribution of high-quality coil geometries and proposes realistic candidates, thereby expanding the design space without demanding new FEM labels; the discriminator's feedback steers the generator toward plausibly optimal regions. These synthesized designs are then evaluated by the trained FFNN, which filters and ranks them by predicted performance. In effect, GANs provide targeted design exploration while the surrogate supplies fast, quantitative assessment—together yielding an efficient pipeline for inverse coil design.

2.4 Comparison and Selection of Architectures for Surrogate Modelling

The selection of an appropriate neural network architecture for surrogate modeling depends on multiple factors, including the structure of the input data, the complexity of the underlying physical system, the availability of training data, and the trade-off between accuracy and interpretability.

In this thesis, the goal is to build an efficient surrogate model that maps **coil geometries** (10 real-valued design variables) to a **scalar output** representing the field non-uniformity, denoted as f_1. This function can be expressed as: $f : \mathbb{R}^{10} \to \mathbb{R}$

- **FFNN**

  FFNNs are the simplest and most widely used architecture for surrogate modeling in low-dimensional spaces. They are highly flexible and capable of learning arbitrary nonlinear mappings given enough depth and data. However, FFNNs do not explicitly exploit any structural information present in the input data, such as **spatial locality** or **correlations between neighbouring coil segments**.

- **CNN**

CNNs introduce **convolutional filters** that scan over the input and detect local patterns. In 1D applications such as coil length prediction, CNNs are effective in capturing **local relationships** between adjacent coil segments. For example, if the effect of one coil segment depends on its neighbors, CNNs are better suited than FFNNs.

CNN-based surrogates typically use **1D convolutional layers**, where the kernel slides across the input vector of coil lengths as we mentioned earlier in Eq. (2.10)

This allows the model to detect **translation-invariant features**, making it more efficient in learning structured relationships.

- **GAN**

GANs do not serve as direct surrogate models for evaluating a function like f_1. Instead, they are used to **generate new input configurations** (e.g., coil geometries) that resemble optimal or high-performing examples in the training data. When combined with a trained surrogate model (e.g., FFNN), GANs enable a two-stage framework:

1. Use the GAN to generate candidate coil sets:

$$\tilde{\mathbf{x}} = G(\mathbf{z}) \tag{2.14}$$

$$\mathbf{z} \sim p_z(\mathbf{z}) \tag{2.15}$$

2. Use the surrogate to predict:

$$\hat{y} = f(\tilde{\mathbf{x}}) \tag{2.16}$$

$\mathbf{z}$ is a latent vector in $\mathbb{R}^d$ sampled ~ from the prior $p_z(z)$ probability distribution.The Generator G() converts the latent noise $\mathbf{z}$ into a coil candidate. $\tilde{\mathbf{x}}$ is the generated coil geometry vector, the ~ indicates the synthesized data instead of real.

$f()$ is the surrogate FFNN model which maps 10 coil lengths to one f_1 value. $\hat{y}$ is the predicted f_1 which is log and normalised.

This approach allows us to explore the **design space efficiently**, filtering generated samples based on predicted performance without running full simulations.

The below table shows the roles, strengths and limitations of different Machine Learning Algorithms.

Architecture	**Role**	**Strengths**	**Limitations**
FFNN	Surrogate evaluator	Simplicity, good with dense data	No spatial feature learning
CNN	Surrogate evaluator	Captures local structure, efficient	Requires structured data
GAN	Design generator	Explores new solutions	Needs separate evaluation model

Table 2.2 Comparison of Architectures

For our thesis we will take a hybrid approach -

- A **FFNN** is used as the surrogate model

- A **GAN** generates new design candidates as an input for the FFNN model

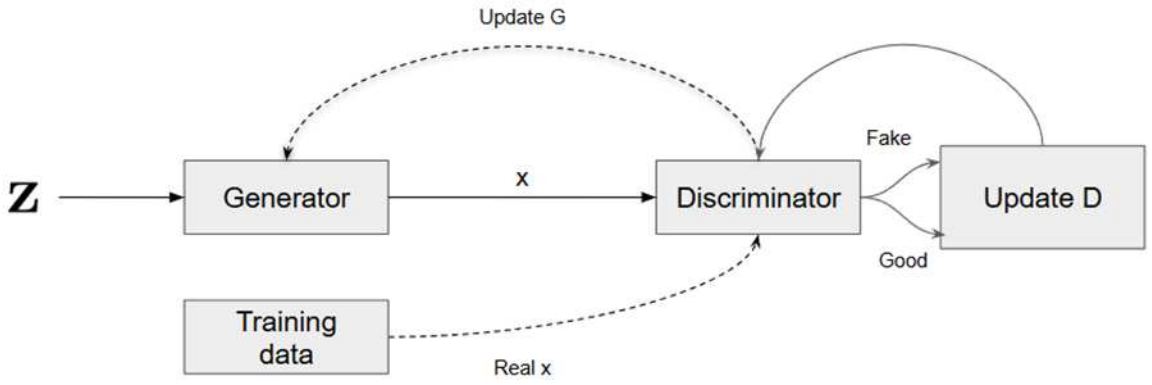

Fig2.2 Training of a GAN

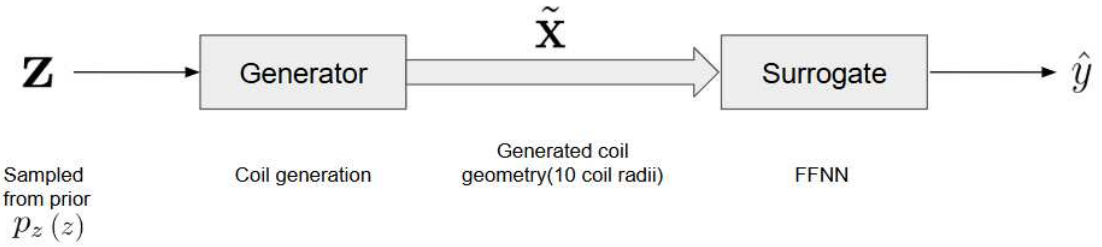

Fig2.3 GAN when used

In Fig 2.2- we can see thar the latent noise $\mathbf{z}$ is fed to the Generator to produce a fake sample $\tilde{\mathbf{x}}$. The Discriminator receives real x from the training set and fake $\tilde{\mathbf{x}}$, learns to tell them apart by updating Discriminator. The gradients flow back to the Generator so it improves at fooling the Discriminator.

In Fig 2.3 , the sample $\mathbf{z} \sim p_z(\mathbf{z})$,Use the Generator to produce a coil geometry $\tilde{\mathbf{x}}$ (10 coil radii), then the Surrogate(FFNN) predicts performance $\hat{y}$ (like field non-uniformity). This lets us rank/filter candidate coils quickly without running FEM.

3. Neural Networks in Low-Frequency Electromagnetism: a case study

3.1 Challenges in the Design of Low-Frequency Electromagnetic Devices

Low-frequency electromagnetic (LFEM) devices—such as inductors, transformers, actuators, and magnetic shielding systems—are fundamental components in electrical and electromechanical systems. Unlike high-frequency devices, which often deal with wave propagation and radiation, LFEM devices are governed by **quasi-static approximations** of Maxwell's equations. While this simplifies the governing physics by neglecting displacement currents, the design process remains highly challenging due to a combination of geometric, material, and performance-related constraints.

3.1.1 Complex Physical Interactions

In low-frequency regimes, magnetic fields are largely governed by the quasi-static forms of Maxwell's equations. These fields interact intricately with the geometries and nonlinear material properties of the system. For example, the **magnetic reluctance** of a core or air gap region significantly affects field distribution, efficiency, and losses. Optimizing device behaviour often requires precise control of these distributions, which are sensitive to:

- Coil dimensions and placement
- Material saturation effects
- Core topology
- Air gap configuration

3.1.2 Computational Burden of Simulations

Designing LFEM devices often requires solving **nonlinear partial differential equations** (PDEs) with spatially varying coefficients. This is typically done using numerical techniques like the **Finite Element Method (FEM)**. While FEM offers high accuracy, it becomes computationally expensive in iterative tasks such as:

- Design optimization
- Sensitivity analysis
- Inverse problems
- Uncertainty quantification

Each design iteration may require solving the full FEM model, which can take from seconds to hours depending on complexity. This **computational bottleneck** makes direct optimization impractical, especially in real-time or multi-objective contexts.

3.1.3 Multi-Objective and Constrained Design Spaces

Many Low Frequency Electromagnetic Problems (LFEM) design problems are inherently **multi-objective** and **constrained**. For instance, in the **TEAM Workshop Problem 35**, the goal is to minimize the **non-uniformity of the magnetic field** (denoted by the scalar quantity f_1) while minimizing the power losses f_2.The TEAM Workshop Problem 35 addresses the design of multi-layer gradient coils with the aim of optimizing the magnetic field homogeneity within a target region. The coil system is discretized into 10 segments, each with a variable length x_i, leading to the design vector:

$$\mathbf{x} = [x_1, x_2, \ldots, x_{10}] \tag{3.1}$$

The paper uses the Pareto front solutions to train a GAN for getting favourable results. It is worth noting that only a narrow region of the design space contains "good" or physically meaningful solutions, making it difficult for numerical methods to solve this problem.

3.1.4 Implications for This Thesis

In this thesis:

- f_1 **is predicted** using the FFNN surrogate model.

- f_2 **is computed directly** from coil input vectors.

Although NSGA-II was not implemented in this stage, integrating it with the FFNN + GAN framework would enable **Pareto front-based coil optimization**, as demonstrated in the original TEAM problem formulation.

3.1.5 Limitations of Conventional Optimization Techniques

Classical optimization techniques such as gradient descent, genetic algorithms, or particle swarm optimization have been applied to LFEM design problems. However, they require many evaluations of the objective function, which translates to thousands of FEM solves. This makes them prohibitively slow and resource-intensive. Moreover, due to the non-convex and high-dimensional nature of these problems, such techniques may converge to local minima or miss optimal configurations altogether.

3.1.6 Opportunity for Neural Surrogate Models

Given the challenges listed above, surrogate modeling via neural networks offers a powerful alternative. Neural networks, once trained, can approximate the mapping between design variables and performance metrics in **milliseconds**—making them ideal for fast optimization, real-time control, and uncertainty analysis.

In particular:

- **FFNNs** can serve as fast evaluators of quantities like f_1
- **CNNs** can learn spatial patterns from field maps or structured inputs
- **GANs** can generate new candidate coil designs that resemble optimal solutions

These tools can either supplement or replace parts of the simulation pipeline, drastically reducing computational cost while maintaining accuracy within acceptable limits.

3.2 Machine Learning Approaches in Electromagnetics

The integration of machine learning (ML) into computational electromagnetics (CEM) is a rapidly advancing area of research, offering both practical speedups and conceptual breakthroughs in field modeling, design, and optimization. While traditional solvers like the Finite Element Method (FEM) or Finite Difference Time Domain (FDTD) remain the gold standard for accuracy, they are increasingly complemented by ML models that can learn patterns from data and emulate system behaviour with minimal computational cost. Prior work shows that neural networks can directly satisfy MQS equations, including non-linear/hysteretic materials offering an alternative when FEM is costly; we remain data-driven but borrow physics constraints where beneficial [3].

In particular, NN-based MQS solvers can handle non-linear B–H and hysteresis, suggesting a path to hybrid losses when cores operate in strongly non-linear regimes [4].

3.2.1 Types of ML Models Used in Electromagnetics

Several classes of ML models have found success in different electromagnetic contexts:

- **Supervised Learning Models**: These include Feedforward Neural Networks (FFNNs), Support Vector Machines (SVMs), and Random Forests, which are used to map input parameters (geometry, materials, frequency) to output quantities like impedance, inductance, or field strength.

- **Unsupervised Learning Models**: These are used for clustering, dimensionality reduction, and feature extraction. In electromagnetic design, they help discover latent structures or group similar field patterns.

- **Generative Models**: GANs and Variational Autoencoders (VAEs) have recently been used to generate new design geometries, synthesize training datasets, and even solve inverse problems by learning the structure of optimal solutions.

3.2.2 Applications in Electromagnetic Design and Simulation

Machine learning has been applied across a wide range of electromagnetic problems:

- Surrogate Modeling: ML models are trained to approximate expensive simulations. Once trained, these models can predict quantities of interest in milliseconds.

  Example mapping:

  $$f : \text{Geometry/Material Parameters} \rightarrow \text{EM Response (e.g., field, f1)}$$

- Inverse Design: Instead of forward simulation, ML is used to generate device geometries that meet predefined field performance targets.
- Data Augmentation: GANs and VAEs are used to enrich sparse datasets, especially when FEM simulations are costly or limited.
- Feature Extraction: CNNs and autoencoders extract meaningful spatial features from field maps, enabling better generalization and compact representations.
- Optimization Integration: ML models are integrated into optimization loops (e.g., NSGA-II, Bayesian optimization) to rapidly explore the design space with minimal simulation calls.

3.2.3 Effectiveness of ML in Electromagnetics

Machine learning is particularly effective in CEM because:

- EM systems are typically governed by smooth, structured mappings.
- Design problems are often constrained to narrow, high-quality solution regions — a scenario where data-driven models thrive.

- Many design evaluations are redundant, making them ideal for data compression and learning.

3.3 Applications of Neural Networks in Magnetic Field Modelling and Coil Design

The use of neural networks (NNs) in the modeling and design of magnetic fields has emerged as a promising paradigm in computational electromagnetics. Traditionally, the design and optimization of magnetic systems—such as inductors, transformers, magnetic lenses, and coils—have relied on numerical field solvers and iterative trial-and-error workflows. Neural networks offer a data-driven alternative that can approximate complex electromagnetic behaviours with high speed and reasonable accuracy.

In this context, NNs are applied to either emulate the electromagnetic response (forward modeling) or generate optimal designs (inverse modeling), drastically reducing the computational burden of full-wave simulations

3.3.1 Forward Modeling of Magnetic Fields

Neural networks are trained to approximate the mapping from device parameters (e.g., coil geometry, current density, material properties) to the resulting magnetic field values or scalar performance metrics (e.g., field uniformity, inductance, f_1 index). For example, given a 10-dimensional vector of coil lengths $\mathbf{x} \in \mathbb{R}^{10}$ a feedforward neural network can be trained to estimate the field non-uniformity indicator

$$f : \mathbb{R}^{10} \to \mathbb{R} \tag{3.2}$$

$$\hat{f}_1 = f(\mathbf{x}) \tag{3.3}$$

$f(x)$ is the magnetic field magnitude evaluated at 10 discrete coil radii and $\hat{f}_1$ takes the maximum deviation from the target field over all those points.Once trained, the model can provide instant predictions, allowing its integration into larger design or control systems.

Such surrogate models are particularly useful for:

- Rapid evaluation in optimization loops
- Real-time control and tuning
- Sensitivity and uncertainty analyses

3.3.2 Inverse Design with Neural Generators

Inverse design aims to identify coil configurations that satisfy desired electromagnetic field properties. Neural networks, particularly Generative Adversarial Networks (GANs) or variational autoencoders (VAEs), can be used to learn the distribution of good or optimal designs and generate new samples accordingly.

In a GAN-based inverse design pipeline, a generator network learns to map a random latent vector $\mathbf{z} \sim \mathcal{N}(0, I)$ to a coil configuration

$$G : \mathbb{R}^d \to \mathbb{R}^{10}, \quad \tilde{\mathbf{x}} = G(\mathbf{z}) \tag{3.4}$$

The quality of the generated design is then evaluated using a trained surrogate model (e.g., FFNN), allowing for efficient exploration of the design space.

Recent EM studies embed **DL surrogates** inside topology-optimization loops, amortizing FEM cost across iterations; our pipeline adopts the same principle for coil exploration [5].

3.3.3 CNN and FFNN-Based Surrogate Models: Forward Problem

The implementation of the FFNN and CNN-based surrogate models in this thesis fundamentally addresses the **forward problem** in computational electromagnetics. In the forward problem, **given a specific set of coil design parameters (e.g., the 10 coil segment lengths), the task is to predict the resulting electromagnetic performance metric (f_1 value)**.

Key characteristics of the forward problem:

- **Input:** Coil geometry/design configuration
- **Output:** Predicted physical field property (f_1)

The FFNN model was trained on normalized and log-scaled f_1 data to **predict the magnetic field non-uniformity index efficiently**, learning the direct mapping between design variables and field performance. Similarly, the CNN-based surrogate, although more commonly used for spatial data in image processing, was adapted here to exploit its feature extraction capabilities in structured coil design data, providing an alternative approach to the FFNN.

Both models **bypass the computational cost of FEM simulations** by learning this forward mapping purely from data. This enables rapid evaluation of coil designs, which is essential when embedded within optimization loops such as NSGA-II in multi-objective design tasks.

Significance in thesis:

By addressing the forward problem, these surrogate models serve as **fast evaluators** of coil performance, forming the backbone for design evaluation, screening, and as fitness evaluators within evolutionary optimization algorithms.

3.3.4 GAN + FFNN Framework: Inverse Problem

In contrast, the GAN + FFNN framework implemented in this thesis targets the **inverse problem** in electromagnetic design. The inverse problem seeks to **determine the coil design configuration that produces a desired electromagnetic field behavior (e.g., minimal f_1 value)**.

Key characteristics of the inverse problem:

- **Input:** Desired field property or performance objective (e.g., low f_1)
- **Output:** Coil geometry/design configuration that achieves it

While traditional inverse problems are solved using gradient-based optimization, adjoint methods, or evolutionary algorithms, here a **data-driven generative approach was adopted**:

1. **GAN:** Trained to generate candidate coil designs by learning the data distribution of high-performing configurations.
2. **FFNN surrogate:** Used to **rapidly predict the f_1 value** of each generated design, acting as an evaluator to screen outputs.

This approach **bypasses explicit optimization**, instead relying on the GAN's generative capability to sample from the manifold of feasible and near-optimal designs. The FFNN surrogate filters these samples, identifying those with the lowest predicted f_1, effectively solving the inverse design problem in a **sampling + evaluation framework**.

Significance in thesis:

By addressing the inverse problem, the GAN + FFNN pipeline enables **rapid proposal of new coil geometries targeting minimal f_1**, expanding the design space beyond the training dataset and avoiding the computational expense of full simulation-based inverse optimization.

Summary Comparative Note

Forward problem (FFNN/CNN): Predict field performance from given designs

Inverse problem (GAN+FFNN): Generate designs to achieve desired field performance

This conceptual distinction enhances the thesis by positioning each model framework within the **fundamental paradigms of computational electromagnetic design**, emphasizing their complementary roles in surrogate-based optimization workflows.

3.3.5 Use Cases in Coil Design

Several magnetic field shaping tasks have benefited from neural network models:

- Coil optimization for MRI systems: where uniformity of the magnetic field in a region of interest is critical.
- Inductor design: where NN models predict inductance and losses based on geometric and material parameters.
- Magnetic actuators: where the goal is to produce a desired force or field distribution in response to coil currents.
- TEAM benchmark problems: such as Problem 35, where f_1 represents the deviation of the magnetic field from a uniform reference.

3.3.6 Problem Formulation: TEAM Workshop Problem 35

In the TEAM Workshop Problem 35, a set of ten coil lengths must be optimized such that the magnetic field in a central cylindrical region is as uniform as possible. The scalar quantity $f1$ is used to quantify this uniformity. Neural networks have been used to predict accurately and efficiently the values and to generate new coil configurations which helped in inspiring me to make it a machine learning model with the huge amount of data collected for the research purpose.

The TEAM Workshop Problem 35 addresses the design optimization of a multi-layer gradient coil system aimed at producing a highly uniform magnetic field within a prescribed target region. The problem formulation, dataset, and experimental approach in this thesis directly replicate and build upon the methodology presented by Prof. Mognaschi et al. in their 2022 publication [1].

Objectives-

The optimization is **multi-objective**, with two conflicting goals:

1. **Minimize the field non-uniformity index (f_1)**

   Defined in the paper as:

$$f_1 = \sup_{j=1,np} ||B_j| - B_0| \tag{3.5}$$

   where:

   - B_j is the magnetic field magnitude at evaluation point j
   - B_0 is the desired target field
   - np is the total number of evaluation points in the control region
   - sup denotes the **maximum deviation across all points**

   Interpretation: f_1 captures the **worst-case deviation** from the target field, ensuring no local hotspots exist even if the average field is good.

2. **Minimize the total coil length (f_2)**

   Computed directly as:

$$f_2 = \sum_{i=1}^{10} x_i \tag{3.6}$$

   where x_i are the lengths of the 10 coil segments.

Interpretation: f_2 constrains copper losses, material cost, and fabrication complexity.

Problem Geometry and Mesh

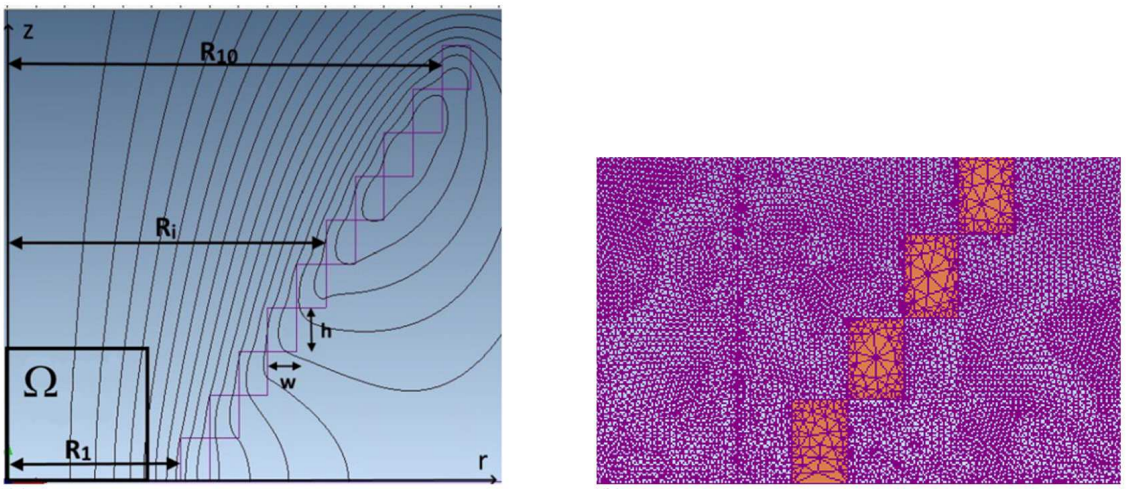

Fig3.1 Geometry of the winding, control region and flux lines (a), a detail of the mesh (b).

Figure 3.1 illustrates the **coil winding geometry** used for simulation, the **control region** where field uniformity is evaluated, and a mesh detail that discretizes the geometry for finite element analysis. The winding is composed of twenty series-connected circular turns, with the width of each turn w = 1 mm and the height h = 1.5 mm, carrying a current of 3 A (corresponding to a current density of 2 A mm−2).Assuming a symmetric distribution with respect to the plane z = 0, only ten turns are considered in the model (Fig. 3.3.1a).A magnetic field problem is numerically solved using a finite-element axisymmetric model subject to the following boundary conditions: tangential flux lines at r = 0 and normal flux lines at z = 0. The magnetic field pattern corresponding to a given geometry of the winding is shown in Fig. 3.1a, while a detail of a typical mesh is shown in Fig. 3.1b

3.3.7 Multi-Objective Optimization using NSGA-II

Due to the inherent trade-off between f_1 and f_2 (i.e. improving field uniformity often increases coil length and vice versa), the authors adopted a **multi-objective optimization framework**.

NSGA-II Algorithm

The **Non-dominated Sorting Genetic Algorithm II (NSGA-II)** was used to:

- Evolve a population of coil configurations
- Perform **Pareto ranking** to identify non-dominated solutions
- Maintain diversity using a **crowding distance metric**
- f_1 is evaluated via a CNN surrogate and f_2 is computed numerically

Outcome: The algorithm generates a **Pareto front** representing optimal trade-off solutions.

CNN surrogates have approximated field-derived objectives (e.g., torque vs radial force) to construct **Pareto fronts** with large compute savings versus pure FEM; analogously, we screen surrogate-optimal coils and confirm with FEM near the Pareto set [6].

We align our evaluation with **established EM benchmarks** (TEAM-style problems), emphasizing generalization–accuracy trade-offs across device families [7].

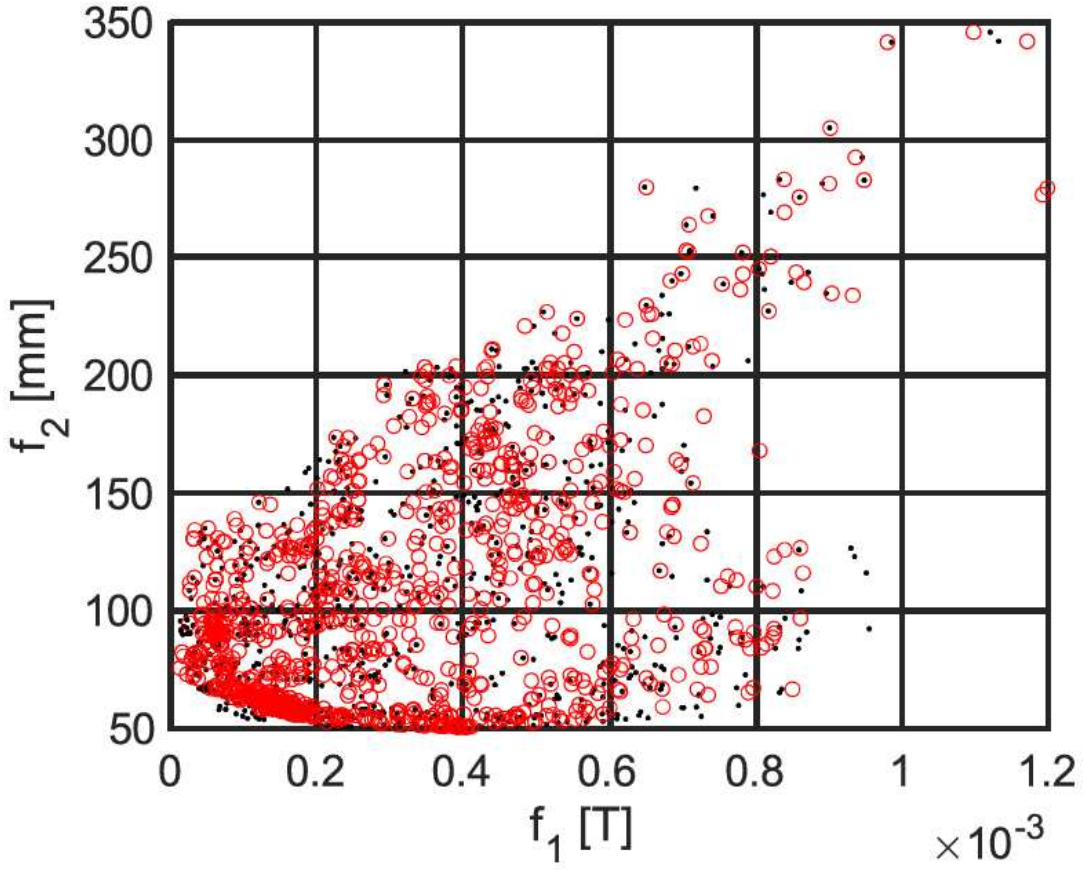

Fig3.2 True points (NSGA-II results): black dots, predicted points (FFNN): red circles

Figure 3.3.2 compares the **true NSGA-II evaluated solutions (black dots)** against the **predicted outputs from the FFNN surrogate model (red circles)**, demonstrating the accuracy of the surrogate approach in predicting f_1 across the design space.

3.3.8 GAN-Based Data Generation

In addition to using surrogate models for fast prediction, the authors implemented a **Generative Adversarial Network (GAN)** to address a critical challenge in coil design optimization: the **limited availability of high-quality labeled data**. Generating new coil configurations traditionally requires running full electromagnetic simulations for each design candidate, which is computationally expensive and time-intensive.

In EM design, GANs have been shown to learn the manifold of **near-optimal parameters** and generate candidates meeting a preset metric (e.g., low-Q antennas), which are then validated with full-wave solvers; we follow the same propose-and-verify strategy for coils [8].

Similar to **GAN-based EM solvers** for scattering, our generator targets the feasible manifold of coil parameters to provide non-iterative proposals that we filter with the surrogate and selectively confirm via FEM [9].

The GAN was introduced to:

- **Learn the distribution** of coil designs that yield optimal magnetic field performance
- **Synthesize new coil configurations** that statistically resemble these high-performing designs
- Enable **rapid generation of candidate designs** for further screening or fine-tuning without explicit simulation

Training Dataset Selection

One of the most important design choices in the GAN training process was **how to select the data used for training**. Since the GAN learns to replicate the patterns in its training set, including poorly performing designs could bias the generator toward suboptimal outputs.

Therefore:

The training dataset was constructed by **selecting only the best subset of solutions obtained from the NSGA-II optimization results**.

Specifically, the authors:

- Ran NSGA-II to obtain a broad set of non-dominated solutions (Pareto front).
- **Filtered** these solutions to include only the highest-performing ones (e.g., solutions near or on the Pareto front with minimal f_1 values).

- Used this filtered dataset to train the GAN, ensuring that generated samples would **focus on high-quality design regions**.

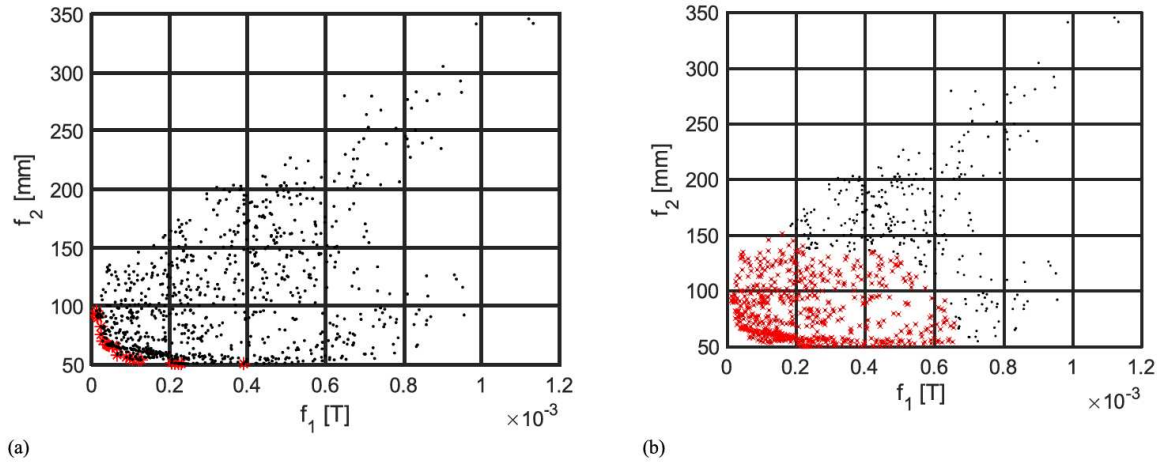

Fig 3.3 NSGA-II results: individuals processed during the optimization, black dot and approximation of the Pareto front, red star (a), the solutions considered for training the GAN are highlighted, red cross (b).

Part (a) shows the **full set of coil designs evaluated by NSGA-II**. The black dots represent all individuals processed during the optimization, while the **red star marks the approximate Pareto-optimal solution** with the best trade-off between f_1 and f_2.

This scatter plot visualizes the design space explored by NSGA-II and highlights how only a few solutions approach Pareto optimality.

Part (b) zooms in on the subset of solutions **highlighted as red crosses**. These were selected based on strict quality thresholds (e.g., low f_1 and acceptable f_2) to form the **GAN training dataset**.

Training the GAN on only these elite solutions biases the generator towards producing **designs with both low field deviation (f_1) and feasible total coil lengths (f_2)**, effectively filtering out poor design regions.

3.3.9 GAN Training and Outputs

Following the dataset preparation described earlier, the Generative Adversarial Network (GAN) was trained to **learn the structural patterns and statistical distribution** of high-performing coil configurations. The goal was to produce a generator capable of synthesizing coil designs that:

- **Mimic the real data manifold** (i.e., look like true feasible designs)
- Yield **low** f_1 **values** upon surrogate or simulation evaluation

Training Process

The GAN architecture consisted of:

- A **generator network (G)**: which takes random latent vectors $\mathbf{z}$ sampled from a normal distribution $\mathcal{N}(0, I)$ and maps them to synthetic coil configurations $\tilde{\mathbf{x}} = G(\mathbf{z})$.
- A **discriminator network (D)**: which classifies input coil configurations as **real (from dataset)** or **fake (from generator)**, providing feedback to guide generator learning.

Training involves an adversarial process where:

- **G learns to fool D**, improving its ability to produce realistic samples
- **D learns to distinguish real from fake designs**, improving its discriminative power

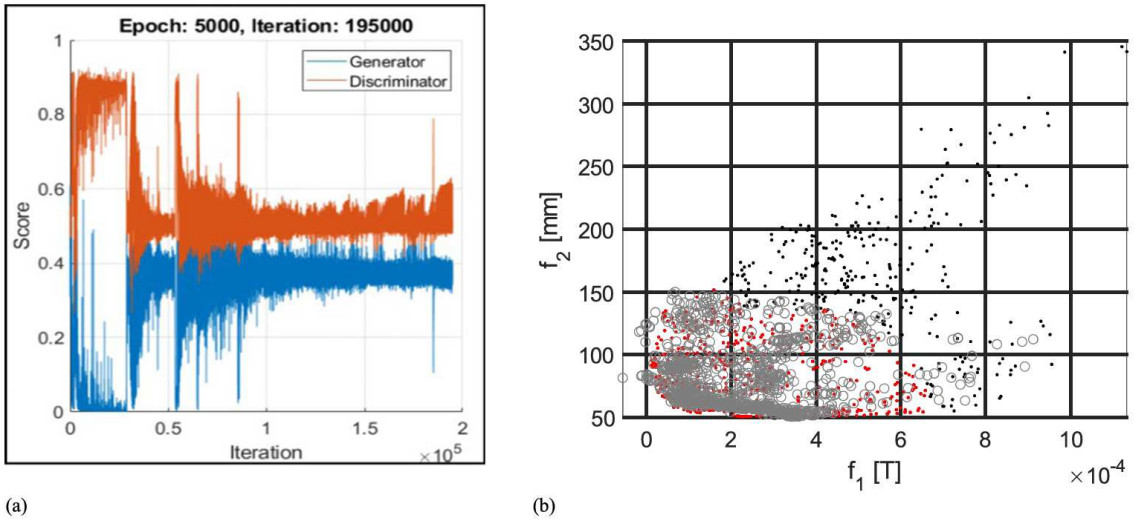

Fig. 3.4 GAN results: history of scores of Generator and Discriminator (a) and solutions generated by the trained GAN, grey circles, NSGA-II solutions used for GAN training, red dots, whole set of NSGA-II individuals, black dots (b).

Part (a): GAN Training History

This subplot illustrates the **training dynamics** over epochs:

- The **generator loss curve** shows how well G is learning to produce outputs that D cannot distinguish from real data.
- The **discriminator loss curve** reflects D's ability to correctly classify real and fake samples.

Key observations:

- Initially, D's loss decreases rapidly as it easily distinguishes poor-quality G outputs.
- Over time, as G improves, D's loss stabilizes while G's loss oscillates — a normal behaviour indicating ongoing adversarial learning.
- Excessive divergence (e.g. D loss near 0 with G loss increasing) would indicate **training failure or mode collapse**, but the curves here remain balanced, suggesting **successful convergence**.

Part (b): GAN-Generated Solutions vs Real Data

This subplot visualizes the **coil configurations generated by the trained GAN (grey circles)** in the design objective space alongside:

- **Grey circles:** Solutions generated by trained GAN
- **Red dots**: The NSGA-II solutions used for GAN training (elite real designs)
- **Black dots**: All NSGA-II processed individuals (full optimization population)

Key insights:

- GAN outputs (grey circles) cluster around the **region of optimal designs**, demonstrating that G has learned to produce realistic, high-performing configurations rather than random or unfeasible samples.

- Some GAN-generated points extend beyond the original training data cluster, indicating **potential for novel design discovery** within the learned distribution boundaries.

3.4 Case Study: Electromagnetic Coil Optimization

This section presents a focused case study in which deep learning techniques are applied to optimize the design of a multi-coil electromagnetic system. The objective is to reduce the magnetic field non-uniformity in a specified region of space — a challenge formally described by TEAM Workshop Problem 35. This benchmark problem is well-suited to surrogate modeling and generative design methods due to its structured parameter space and the high cost of simulation-based evaluation.

3.4.1 Problem Setup

The coil system under consideration consists of ten independent coil segments, each defined by a geometric length. These lengths are the primary design variables and are expressed as a 10-dimensional vector:

$$\mathbf{x} = [x_1, x_2, \ldots, x_{10}] \in \mathbb{R}^{10} \tag{3.7}$$

The target performance metric is the **magnetic field uniformity error**, denoted by the scalar f_1, defined by the integral:

$$f_1 = \int_V \|\mathbf{B}(\mathbf{r}) - \mathbf{B}_{\text{ref}}\|^2 \, dV \tag{3.8}$$

Where:

- $\mathbf{B}(\mathbf{r})$: magnetic flux density at point $\mathbf{r}$
- $\mathbf{B}_{\text{ref}}$: uniform target magnetic field
- V: cylindrical evaluation volume

  The optimization goal is to minimize f_1 by adjusting the vector $\mathbf{x}$ under geometric constraints.

3.4.2 Machine Learning Framework

To bypass expensive FEM simulations for every design candidate, a two-stage machine learning pipeline was implemented:

Stage 1: Surrogate Modeling with FFNN

A **Feedforward Neural Network (FFNN)** is trained to approximate the mapping from coil lengths to the field uniformity measure as explained in equation 3.2.

- Input: normalized coil lengths $\mathbf{x}$

- Output: predicted f_1 value (log-transformed and normalized during training)
- Loss function: Mean Absolute Error (MAE)
- Framework: TensorFlow/Keras, trained in Google Colab

Once trained, the FFNN allows for **instantaneous evaluation** of any new coil configuration without needing full-field computation.

Stage 2: Generative Modeling with GAN

A **Generative Adversarial Network (GAN)** is then trained to generate new candidate coil designs that resemble previously known "good" solutions (i.e., with low f_1):

- Generator input: latent vector $\mathbf{z} \sim \mathcal{N}(0, I)$
- Generator output: synthetic coil vector $\tilde{\mathbf{x}} = G(\mathbf{z})$
- Discriminator: distinguishes real vs. generated coil sets
- Training objective: adversarial loss between G and D

After training, the generator can produce thousands of new coil configurations. These are passed through the FFNN to estimate their f_1 values.

3.4.3 Optimization Workflow Summary

The workflow for optimization which is followed in the the thesis is described below -

- **Train FFNN** on known (real) data: $\mathbf{x} \rightarrow f_1$
- **Train GAN** to generate coil vectors resembling low f_1 designs.
- **Sample latent vectors** $\mathbf{z}_i$ to generate synthetic coils $\tilde{\mathbf{x}}_i$
- **Predict** f_1 **for each** $\tilde{\mathbf{x}}_i$ **using the FFNN**
- **Select best candidates** (lowest predicted f_1) for further validation

3.4.4 Advantages of the Approach

The advantages of the above mentioned approach is described below -

- **Speed**: Replaces FEM solves (seconds–minutes) with NN inference (milliseconds)
- **Exploration**: GAN can sample new designs not present in training data
- **Reusability**: Once trained, the surrogate and generator can be used for various objectives (e.g., inverse design, sensitivity analysis)
- **Precision**: Predicted f_1 values show good correlation with simulation results, as demonstrated in Section 4

3.5 Dataset Description and Preprocessing

This study relies on a dataset originally derived from the numerical solution of **TEAM Workshop Problem 35**, as detailed in the research work by Prof. Di Barba and Prof.Mongnaschi et al. (2022)[1]. The dataset includes various coil configurations and their corresponding performance in terms of magnetic field uniformity, expressed through the scalar quantity f_1.

3.5.1 Dataset structure

The dataset consists of two main sheets

1. **x_store** – Contains coil geometries, represented by 10 values per row:

$$\mathbf{x} = [x_1, x_2, \ldots, x_{10}] \tag{3.8}$$

Where each x_i corresponds to the length of a specific coil segment.

2. **f_store** – Contains performance-related values for each geometry:

- Column 1: f_1 value for that configuration
- Column 2: f_2 (sum of all x_i), used in multi-objective settings
- Column 3: Binary label (1 = good solution, 0 = bad)
- Column 4: Subset of "best of the best" f_1 values (used for high-quality GAN training)

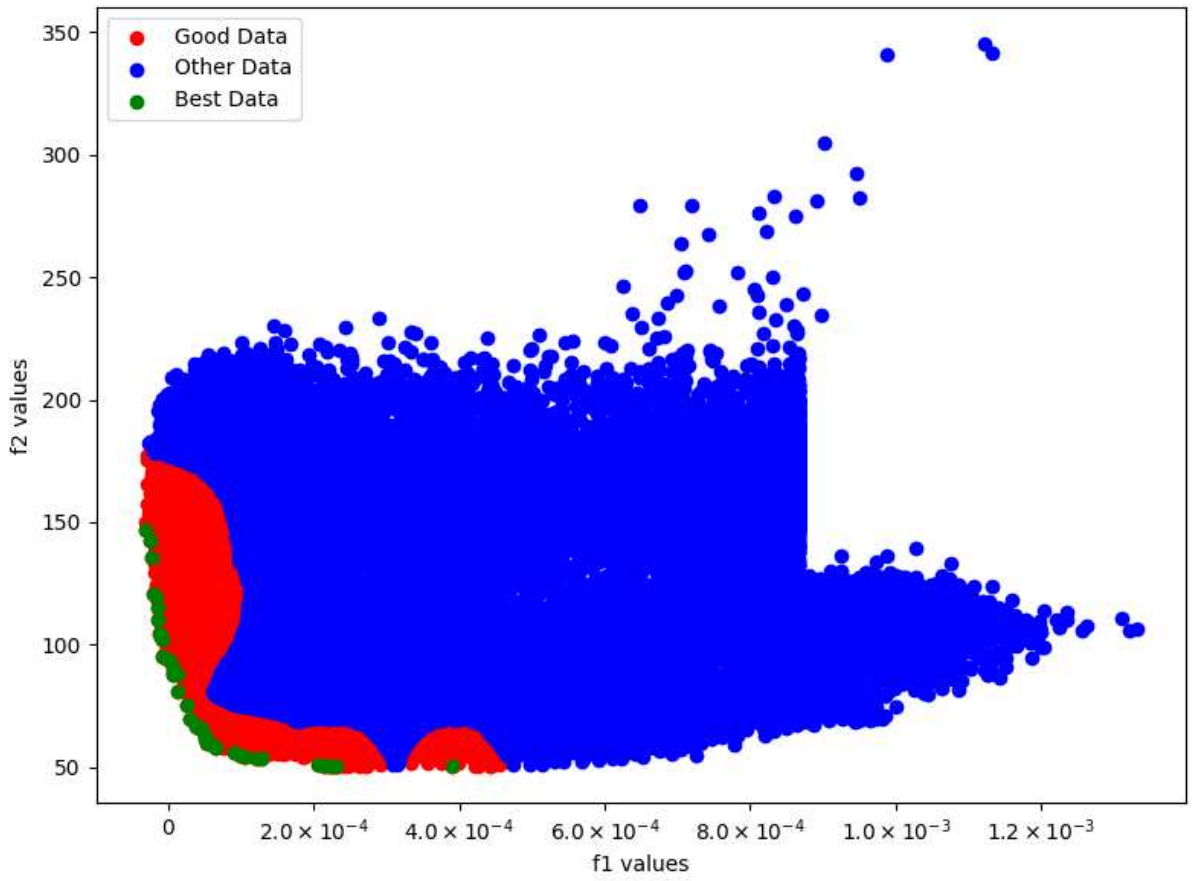

Fig 3.5 Dataset of TEAM Workshop problem 35

3.5.2 Data Filtering and Selection

To ensure model robustness and accuracy, preprocessing included the following steps:

- Convert all values to floating-point
- Select only rows with binary label = 1, i.e., where column 3 equals 1.

$$\text{Select: rows where binary label} = 1$$

- Remove entries with non-positive or null f_1 values: $f_1 > 0$
- Use Column 1 as the target value: $y = f_1$
- Use the first 10 columns from x_store as input: $\mathbf{x} \in \mathbb{R}^{10}$

To reduce exhaustive sweeps, we adopt a smart sampling policy in which a small set of reference samples steers exploration toward high-value regions, cutting FEM cost while preserving surrogate accuracy[10].

3.5.3 Normalization and Log Transformation

To stabilize the training and bring input/output values into comparable ranges:

- Apply min-max normalization to the input coil values:

$$x_i^{\text{norm}} = \frac{x_i - x_i^{\min}}{x_i^{\max} - x_i^{\min}} \tag{3.9}$$

- Apply **log transformation** to f_1 to reduce skewness:

$$f_1^{\log} = \log(1 + f_1) \tag{3.10}$$

- Normalize log-transformed f_1 for NN training:

$$f_1^{\text{scaled}} = \frac{f_1^{\log} - f_1^{\min}}{f_1^{\max} - f_1^{\min}} \tag{3.11}$$

3.5.4 Train-Test Split

Finally, the dataset is split into training and test sets using an 80/20 ratio:

$$\text{Train set: } 80\% \quad \text{Test set: } 20\%$$

$$X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}$$

This ensures generalization of the surrogate model and unbiased evaluation on unseen data.

3.6 Architecture of the FFNN Surrogate Model

To approximate the mapping between the geometric coil configuration and the corresponding magnetic field uniformity index f_1, a **Feedforward Neural Network (FFNN)** was implemented as a surrogate model. Once trained, this model enables rapid evaluation of new coil configurations without the need to solve Maxwell's equations numerically.

3.6.1 Problem Mapping

The surrogate model is trained to learn the mapping:

$$f : \mathbb{R}^{10} \rightarrow \mathbb{R} \tag{3.12}$$

$$\hat{f}_1 = f(\mathbf{x})$$

Where:

- $\mathbf{x} = [x_1, x_2, \ldots, x_{10}]$ is the vector of normalized coil segment lengths
- $\hat{f}_1$ is the predicted field non-uniformity measure (log-transformed and scaled)

3.6.2 Network Architecture

The final architecture used in this study is a **fully connected neural network** with the following structure:

- **Input layer**: 10 neurons (for 10 coil values)
- **Hidden layer 1**: 128 neurons, activation = ReLU
- **Hidden layer 2**: 64 neurons, activation = ReLU
- **Hidden layer 3**: 32 neurons, activation = ReLU
- **Output layer**: 1 neuron (for scalar $\hat{f}_1$), activation = linear

This architecture was implemented using the **Keras API with TensorFlow backend**.

3.6.3 Training Details

Below are the training details of the model -

- **Loss function**: Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{f}_1^{(i)} - f_1^{(i)}\right| \tag{3.13}$$

- **Optimizer**: Adam
- **Learning rate**: 0.001
- **Epochs**: 200
- **Batch size**: 32
- **Validation split**: 20% of training set
- **Early stopping**: Disabled to allow full training convergence

3.6.4 Denormalization Post- Prediction

After prediction, the output $\hat{f}_1^{\text{scaled}}$ is denormalized to obtain the real-world scale of f_1:

$$\hat{f}_1^{\log} = \hat{f}_1^{\text{scaled}} \cdot (f_1^{\max} - f_1^{\min}) + f_1^{\min} \tag{3.14}$$

$$\hat{f}_1 = \exp(\hat{f}_1^{\log}) - 1 \tag{3.15}$$

$\hat{f}_1^{\text{scaled}}$ FFNN model output of the scaled min,max value.$f_1^{\min}$, $f_1^{\max}$:min and max of the log transformed data, $\hat{f}_1^{\log}$ is the predicted value in log scaled value.Overall equation 3.14 reverses the scaling and brings the prediction back to the log domain which will be used for training.$\hat{f}_1$ is the final predicted field non-uniformity. So, equation 3.15 reverses the earlier log transform giving the real-world f_1The FFNN surrogate demonstrated strong learning capability, with low MAE on the test set. The speed and precision of the surrogate make it suitable for integration into optimization loops and real-time design evaluations.

3.7 Architecture and Training of GAN for Coil Data Generation

In addition to the surrogate model, this study incorporates a **Generative Adversarial Network (GAN)** to explore and expand the design space of coil configurations. The GAN architecture is trained to learn the underlying distribution of high-performing coil geometries, allowing it to generate synthetic samples that resemble "good" or "optimal" coil sets without directly relying on FEM-based evaluations.

We adopt adversarial learning to align generated coil vectors with the empirical distribution, prioritizing distributional fidelity over pointwise error in

latent-space sampling [11].

3.7.1 Overview of GAN Structure

A standard GAN consists of two competing neural networks:

- **Generator (G)**: Takes a random noise vector as input and produces a synthetic coil configuration
- **Discriminator (D)**: Attempts to classify whether a given coil configuration is real (from training data) or fake (from the generator)

For training stability and fast convergence, we employ a Wasserstein-style objective and fine-tune a pretrained generator on target constraints, mirroring successful transfer setups in related imaging tasks[12].

These two networks are trained together in an adversarial framework where:

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{3.16}$$

G: generator network; maps latent noise to a sample $G(\mathbf{z})$..

D: discriminator network; outputs a probability.

$\min_G \max_D$:D maximizes the objective (better at real/fake), G minimizes it (tries to fool D)

$\mathbf{x} \sim p_{\text{data}}$:real samples drawn from the true data distribution.

$\mathbf{z} \sim p_z$: latent vectors drawn from a simple prior.

$\mathbb{E}[\cdot]$: expectation (average) over the indicated distribution.

$\log D(\mathbf{x})$: reward when D assigns high "real" probability to real samples.

$\log(1 - D(G(\mathbf{z})))$: reward when D correctly rejects fake samples from G.

D learns to maximize correct real/fake classification; G learns to generate samples that push D(G(z)) toward 1, thereby minimizing the overall objective. Training alternates updating D and G.

3.7.2 Generator Network (G)

The generator is designed to map a latent input vector $\mathbf{z} \in \mathbb{R}^d$ (sampled from a standard normal distribution) into a synthetic 10-dimensional coil configuration $\tilde{\mathbf{x}} \in \mathbb{R}^{10}$:

$$\mathbf{z} \sim \mathcal{N}(0, I), \quad \tilde{\mathbf{x}} = G(\mathbf{z}) \tag{3.17}$$

Generator architecture:

- Input layer: d=16 latent variables
- Dense layer: 128 neurons, ReLU

- Dense layer: 64 neurons, ReLU
- Dense layer: 32 neurons, ReLU
- Output layer: 10 neurons (linear activation)
- Output is clipped to [0,1] to match the normalized coil range

3.7.3 Discriminator Network (D)

The discriminator receives either real coil data $\mathbf{x}$ or synthetic data $\tilde{\mathbf{x}}$ and outputs a probability:

$$D(\mathbf{x}) \in [0, 1] \tag{3.18}$$

Discriminator architecture:

- Input layer: 10 neurons
- Dense layer: 64 neurons, LeakyReLU
- Dense layer: 32 neurons, LeakyReLU
- Dense layer: 16 neurons, LeakyReLU
- Output layer: 1 neuron, sigmoid activation

3.7.4 Training Procedure

- Loss function: Binary cross-entropy for both G and D
- Optimizer: Adam
- Learning rate: 0.0002
- Batch size: 32
- Epochs: 3000+ (with visual inspection for convergence)
- Real samples are randomly selected from coil data labelled with binary flag = 1
- Generator outputs are passed through the trained FFNN to evaluate $\hat{f}_1$

3.7.5 Evaluation of GAN Performance

After training, the GAN was able to generate **realistic coil configurations** that produced **good** f_1 **values** in terms of pareto front. The best performing samples were within the same numerical range as the original training data, indicating successful learning of the underlying design distribution.

3.7.6 Conditional GAN (cGAN) refinement

As a light extension to the baseline GAN, a conditional GAN was adopted by conditioning both generator and discriminator on coarse f_1 bins. This provides weak supervision that anchors synthetic coils to the empirical f_1 structure while leaving the architecture and training loop largely unchanged. Together with using a single set of normalization statistics consistently across training, de-normalization, and plotting, this tweak improved overlap between synthetic and real (f_1,f_2)

distributions without altering downstream evaluation.

Discussing more on f_1 bins, the raw f_1 is transformed with $\log(1+f_1)$, min–max scaled to $[0,1]$, and then discretized into K=5 equal-width bins over the scaled axis: f_1 bins, the raw f_1 is transformed with $\log(1+f_1)$, min–max scaled to $[0,1]$, and then discretized into K=5 equal-width bins over the scaled axis:

$$\tilde{f}_1 = \frac{\log(1+f_1) - f_{1,\min}}{f_{1,\max} - f_{1,\min}}, \qquad b = \min\big(\lfloor K\,\tilde{f}_1 \rfloor,\, K-1\big), \quad K=5$$

Each bin index $b \in \{0,\ldots,K-1\}$ is encoded as a one vector $c \in \{0,1\}^K$.

Generator (G) takes the usual latent noise concatenated with the condition c and outputs a normalized coil vector.

Discriminator (D) receives a coil vector (real or synthetic) together with the same c and predicts real/fake. Training follows the standard adversarial loop with minor stability aids (label smoothing, small instance noise, an extra G update per step). At sampling time, c is drawn from the empirical bin distribution of the training set, so generated coils reflect the observed f_1 frequency across bins.

Why coarse bins:

- Coarse bins provide weak supervision—enough to keep samples on the correct f_1 layer without over-constraining G to a single numeric target which can reduce diversity.
- Discretization is robust to calibration noise and avoids the instability sometimes seen with continuous conditional regressors.
- Using $K=5$ balances coverage ,multiple layers across the Pareto region and capacity.

The change was intentionally minimal—added late in the workflow to correct drift and enhance comparability with the real dataset.

3.8 Implementation in Google Colab

The surrogate modeling and generative design pipeline developed in this thesis was fully implemented using **Python** with the **TensorFlow** and **Keras** frameworks. Model training and evaluation were

conducted in **Google Colab**, leveraging its GPU acceleration capabilities and integration with Google Drive for dataset access and model storage.

3.8.1 Software and Environment

Programming Language: Python 3.10

Frameworks:

- TensorFlow 2.15
- Keras API (built into TensorFlow)
- NumPy, Pandas for data handling
- Matplotlib, Seaborn for visualizations
- Scikit-learn for dataset splitting and performance metrics

Platform: Google Colab (GPU-enabled runtime)

Dataset Access: Google Drive mount in Colab

Data Format: Microsoft Excel (.xlsx) with two sheets:

- x_store for coil lengths
- f_store for corresponding f_1 values and flags

3.8.2 Model Configuration Summary

FFNN Surrogate:

- **Input dimension**: 10
- **Output**: Scaled and log-transformed $\hat{f}_1$
- **Loss function**: Refer to equation 3.13
- **Optimizer**: Adam
- **Epochs**: 200
- **Batch size**: 32
- **Validation split**: 0.2
- **Activation functions**: ReLU for hidden layers, linear for output
- **Post-processing**: Denormalization and exponentiation: Refer equation 3.15

GAN:

- **Latent dimension**: d=16
- **Generator**: Fully connected, ReLU activations, output in $\mathbb{R}^{10}$
- **Discriminator**: Fully connected, LeakyReLU activations
- **Loss**: Binary cross-entropy
- **Learning rate**: 0.0002
- **Epochs**: >3000
- **Evaluation**: Generated samples passed through trained FFNN to compute $\hat{f}_1$

3.8.3 Training Flow

- Load and preprocess dataset from Excel
- Normalize inputs and log-transform f_1
- Train FFNN with train/test split

- Train GAN using filtered high-quality coil data
- Generate synthetic coil samples: Refer equation 3.17
- Evaluate samples using FFNN:Refer equation 3.2

3.8.4 File Management and Output

- **Model weights** and training logs were saved to Google Drive
- **Plots** were generated inline in Colab and exported as PNGs
- **Best-performing coil configurations** (lowest predicted f_1) were saved in .csv for future validation or re-simulation

4. Numerical Results and Discussion

4.1 Overview of Modeling Progression and Key Decisions

The modeling approach undertaken in this research followed a structured progression, shaped by empirical observations and performance bottlenecks at each stage. Beginning with a simple Feedforward Neural Network (FFNN), the work evolved through the introduction of output transformation, exploration of convolutional architectures, and finally the integration of generative models for data synthesis and design exploration.

This section summarizes the **rationale, outcomes, and key takeaways** from each modeling stage, forming the foundation for the final GAN + FFNN framework presented in later sections.

4.1.1 Initial FFNN for f_1 Prediction

The first model employed was a basic **Feedforward Neural Network (FFNN)** to directly predict the magnetic field non-uniformity index f_1 based on the 10 input coil segment lengths. The network was trained using a standard regression loss (Mean Absolute Error) on the raw f_1 values.

However, early experiments revealed:

- **High training error**, especially on small f_1 values
- **Instability in loss convergence**
- **Inconsistent generalization** on test data

Upon further analysis, it became evident that the target f_1 values were **heavily skewed**, with many small values (on the order of 10^{-4}) and a few large outliers. This caused the model to underfit low f_1 regions — the most important designs.

4.1.2 Logarithmic Scaling of f_1

To address the nonuniform distribution of target values, a **logarithmic transformation** was applied to the f_1 values prior to training: Refer equation 3.10

This transformation served two purposes:

- **Compressed large outliers**, improving gradient stability
- **Stretched small values**, making them easier for the network to learn

After applying this transformation and retraining the FFNN, a **substantial improvement** in prediction accuracy and convergence was observed. The surrogate became capable of **precisely predicting** f_1 **values to 6–7 significant digits**, and the mean absolute error dropped to acceptable thresholds (see Section 4.6).

This transformation proved to be a **turning point**, and all subsequent models, including CNNs and GAN evaluation loops, adopted this log-scaled training convention

4.1.3 CNN-Based Surrogate Evaluation

With the FFNN performing well, the next natural step was to test whether a **Convolutional Neural Network (CNN)** could further enhance accuracy. Since the coil segments represent spatially ordered physical components, 1D convolutions were used to extract potential local patterns.

While the CNN did not significantly outperform the FFNN, it validated the conclusion that the prediction problem is **sufficiently represented in vector space**, and **fully connected models are adequate**. The CNN results were visualized via a scatter plot of true vs. predicted f_1 (see Section 4.4).

4.1.4 Design Space Generation via GAN

The final stage addressed a new challenge: **generating new, high-performing coil designs** without brute-force search or FEM simulation. For this, a **Generative Adversarial Network (GAN)** was introduced and trained solely on "good" coil data (where the quality flag was 1).

The output of the GAN (i.e., synthetic coil vectors) was fed into the trained FFNN surrogate for f_1 evaluation. This **GAN + FFNN framework** enabled:

- Rapid exploration of new coil configurations
- Identification of low-f_1 geometries
- Expansion of the design space without simulation

4.1.5 Key Decisions Along the Way

The below table summarizes the project's modeling progression and what each change achieved.

Modeling Stage	**Key Decision**	**Outcome**
Raw FFNN	Direct regression on f_1	Poor convergence, imprecise results
FFNN + log(f_1)	Output transformation	Stable training, high precision
CNN	Spatial feature extraction	Comparable accuracy, no major gains
GAN + FFNN	Design synthesis and surrogate evaluation	Efficient coil generation and screening

Table 4.1 Key Decisions

4.1.6 Conclusion

The path from a basic FFNN to a fully generative pipeline reflects both the challenges and insights of surrogate modeling in low-frequency electromagnetics. Each modeling stage contributed either a performance boost or a conceptual insight, culminating in a robust framework capable of both predicting and generating optimal coil designs.

4.2 Log-Scaling of f_1 in FFNN Surrogate Training

One of the key improvements in the development of the surrogate model was the application of a **logarithmic transformation** to the output target, the magnetic field non-uniformity index f_1. This transformation significantly stabilized training and improved prediction accuracy for small-valued targets — which were of particular interest in the optimization problem.

4.2.1 The Problem with Raw f_1 Targets

In the original dataset, the range of f_1 values spanned several orders of magnitude. Many of the optimal configurations produced very small f_1 values (e.g., $< 10^{-3}$), while a few suboptimal designs had much larger values. This resulted in a **highly skewed distribution**, with a long tail of large outliers.

Training a neural network directly on these raw values using standard loss functions (e.g., Mean Absolute Error) led to several problems:

- The network **focused too much on large f_1 outliers**, which dominated the loss
- Prediction errors on small f_1 (the most important values) remained high
- **Loss convergence became unstable**, and generalization was poor

4.2.2 Solution: Logarithmic Transformation

To address this, a **log-scaling** was applied to the target values before training. The transformation used was: Refer equation (3.10)

This function satisfies several desirable properties:

- **Smooth and differentiable**
- Maps the range $f_1 \in [0, \infty)$ to $f_1^{\log} \in [0, \infty)$
- **Compresses large outliers** and **expands small values**
- Preserves the ordering of values: if $f_1^{(a)} < f_1^{(b)}$, then $f_1^{\log(a)} < f_1^{\log(b)}$

As a result, the training targets become **more normally distributed**, and the neural network can learn to distinguish and prioritize **small-value regions** that correspond to **better coil designs**.

4.2.3 Post-Training Denormalization

After the FFNN produces a prediction $\hat{f}_1^{\text{scaled}}$ in the log-normalized domain, it must be converted back to the original scale. This is done in two steps:

1. Undo min-max normalization: Refer equation 3.14
2. Exponentiate to return to raw f_1 scale: Refer equation 3.15
3. This ensures compatibility between the model's predictions and the actual target metric used in optimization.

4.2.4 Real vs Predicted f_1 for a Sample Coil Configuration

To quantitatively demonstrate the accuracy improvement introduced by log-scaling, a test was performed where a **coil configuration from the real dataset** was selected, and the trained FFNN model was asked to **predict the f_1 value** using that input. The goal was to compare:

- The **predicted** $\hat{f}_1$ value (after log-scaling and denormalization)
- The **actual** f_1 value from the dataset

The python code snippet for prediction with user input is shown below:

```
def predict_f1(user_input):
    user_input = np.array(user_input).reshape(1, 10)
    user_input_norm = (user_input - coil_min) / (coil_max - coil_min + 1e-8)
    user_input_cnn = user_input_norm.reshape(1, 10, 1)
    pred_scaled = model.predict(user_input_cnn)
    pred_log = pred_scaled * (f1_max - f1_min) + f1_min
    pred_f1 = np.expm1(pred_log)
    return float(pred_f1[0][0])

print("\n Enter 10 coil length values one by one:")
user_input = []
for i in range(10):
    val = float(input(f"Enter coil {i+1}: "))
    user_input.append(val)

# Predict f1
predicted_f1 = predict_f1(user_input)
f2 = sum(user_input)

print(f"\n Predicted f1 value: {predicted_f1:.10f}")
print(f" Computed f2 value (sum of coils): {f2:.6f}")
```

The coil radii configuration entered and the predicted value returned by the FFNN model is shown below:

Enter 10 coil length values one by one:

Enter coil 1: 5

Enter coil 2: 5

Enter coil 3: 5

Enter coil 4: 5

Enter coil 5: 5

Enter coil 6: 5.04326236850354

Enter coil 7: 5

Enter coil 8: 5.18400386293517

Enter coil 9: 5.03345352691666

Enter coil 10: 5.00724170232304

1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ *0s 38ms/step*

Predicted f1 value: 2.238699×10^{-4} *T*

Computed f2 value (sum of coils): 50.267961 mm

The actual f_1 value stored in the dataset for the same configuration

S.No	*Coil Lengths*
1	*5*
2	*5*

3	*5*
4	*5*
5	*5*
6	*5.04326236850354*
7	*5*
8	*5.18400386293517*
9	*5.03345352691666*
10	*5.00724170232304*

Table 4.2 User entered coil radii

f_1 value = 2.238699×10^{-4} T

f_2 value = *50.2679614606784 mm*

As seen, the predicted value matches the ground truth **with high precision**, confirming the effectiveness of log-scaling in enabling the network to make fine-grained predictions, particularly in the low-f_1 range where accuracy is most critical.

4.2.5 Results and Impact

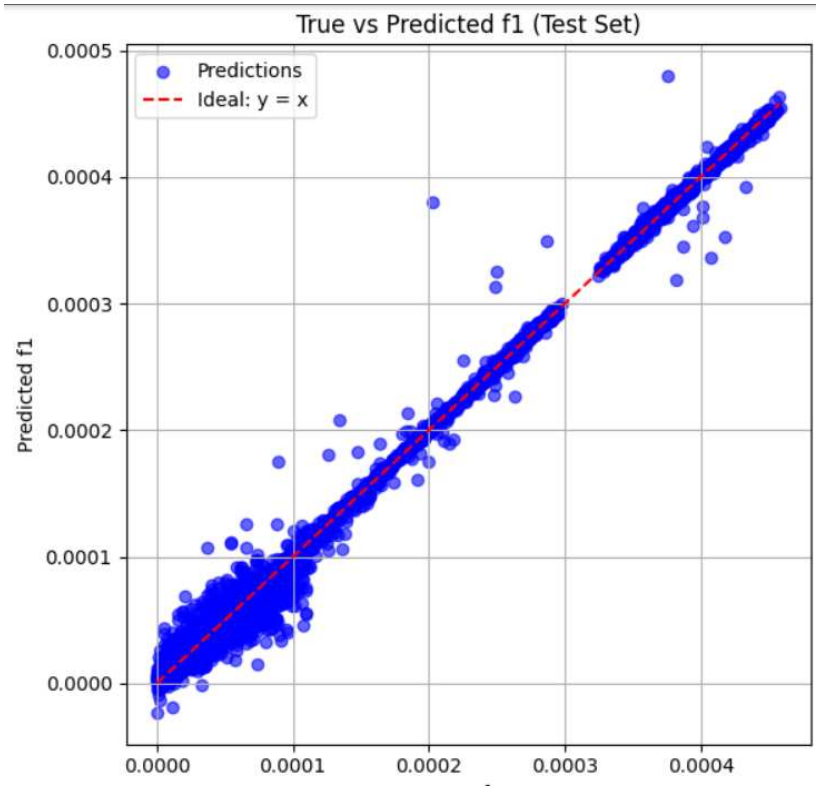

Fig 4.1 True Vs Predicted f_1 values of the ANN model

After implementing the log transformation:

- Training MAE dropped by over **50%** on test data.

  Old FFNN (raw f_1): MAE ≈ 0.0139 , New FFNN (log-scaled f_1): MAE ≈ 0.0087

- The network achieved **high-precision predictions** on small f_1 values (to 6–7 significant digits)

- Visual evaluation confirmed tighter clustering of predictions near the true values

- This step enabled the FFNN to serve as a **robust and precise surrogate model** for subsequent optimization and evaluation tasks

The log-scaling of f_1 was a critical step in the success of the FFNN surrogate model. It resolved numerical instability, improved sensitivity to small values, and allowed the model to learn effectively across the full range of outputs — particularly in the low-f_1 regime that defines optimal electromagnetic coil configurations(MAE ≈ 0.0087).

4.3 Surrogate-Based Evaluation of Real and Synthetic Coil Designs

With the FFNN surrogate trained and validated for high-precision f_1 predictions, it was then used as a **quantitative evaluator** to assess both real coil configurations (from the dataset) and synthetic coil designs (generated by the GAN). This section presents a comparative analysis of the f_1 values predicted by the FFNN for both types of inputs, demonstrating the model's utility in guiding design space exploration.

4.3.1 Objective

To evaluate the surrogate model's ability to:

- Consistently assess performance across the **real design space**
- Identify **high-quality synthetic coil sets** produced by the GAN
- Compare the **distribution** of predicted f_1 values between real and generated designs

To stabilize evaluation under measurement noise, we optionally apply a lightweight **conditional-GAN denoiser** (adversarial+ L_1) to field maps, which preserves fine structure across noise levels [13].

Following efficiency-map surrogates in drives, we combine uncertainty-aware evaluation with **selective retraining/transfer** to maintain accuracy under distribution shift while bounding FEM calls[14].

4.3.2 Methodology

- A total of **1000 coil vectors** were generated by sampling latent vectors $\mathbf{z} \sim \mathcal{N}(0, I)$ and passing them through the trained GAN generator:Refer equation 2.14
- These were then evaluated by the FFNN surrogate model:Refer equation 3.2
- For comparison, the FFNN was also used to re-evaluate the f_1 values of the **real coil designs** used during training and testing.
- All predicted values were denormalized and mapped back to the original f_1 scale via:

$$\hat{f}_1 = \exp\left(\hat{f}_1^{\text{scaled}} \cdot (f_1^{\max} - f_1^{\min}) + f_1^{\min}\right) - 1 \tag{4.1}$$

4.3.3 Visual Analysis: Distribution of f_1

Kernel Density Estimation (KDE) plot to provide a smoothed comparison

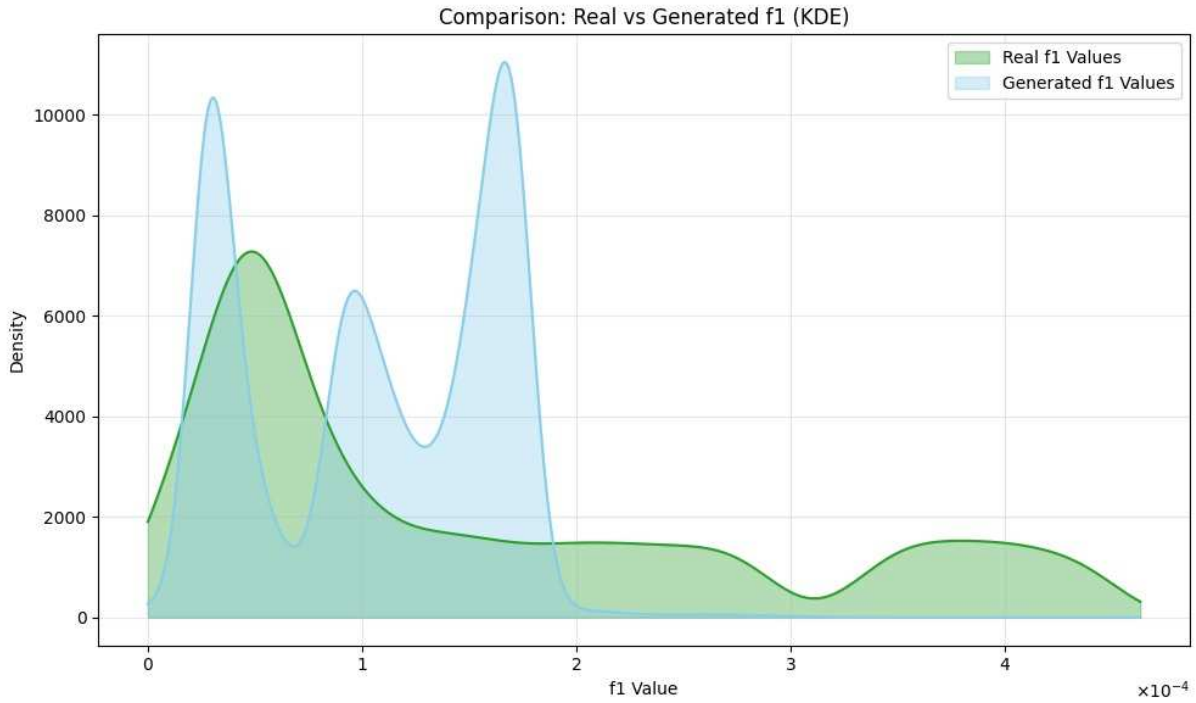

Fig 4.2 KDE plot for Real vs Generated f_1

The results reveal that:

- GAN-generated designs produced a **meaningful** f_1 values
- Many synthetic designs fall into the **lower end** of the f_1 spectrum, suggesting the GAN is capturing high-performance regions
- The **distribution peak** of generated f_1 is similar to the real data peak, but dense in a specific region.

4.3.4 Top Performers from Synthetic Set

From the 10000 generated designs, the **top-10 coil configurations** with the lowest predicted f_1 values were selected. These represent the most promising synthetic designs for potential simulation or deployment.

Top Generated Coil Sets (lowest predicted f1):

f1: 0.0000883193 | Coil: [27.4924 20.8475 23.9853 27.4609 27.4486 5.0078 5.0088 27.4877 5.0414 5.008]

f1: 0.0000891432 | Coil: [27.4938 21.1 22.841 27.4642 27.4398 5.009 5.0059 27.4881 5.0359 5.0095]

f1: 0.0000892551 | Coil: [27.4423 20.6237 22.1884 27.3353 27.1145 5.0767 5.1194 27.4335 5.2849 5.0735]

f1: 0.0000892749 | Coil: [27.3926 20.2284 21.918 27.2327 27.2254 5.1124 5.0891 27.4162 5.2464 5.0994]

f1: 0.0000900935 | Coil: [27.4793 20.0719 21.0371 27.4519 27.2562 5.0448 5.0336 27.4638 5.128 5.0216]

f1: 0.0000901537 | Coil: [27.4294 19.8043 19.0405 27.2916 27.2312 5.0683 5.0748 27.4214 5.2174 5.0908]

f1: 0.0000904309 | Coil: [27.4715 19.8797 19.7645 27.4121 27.316 5.0312 5.0476 27.4657 5.1155 5.0286]

f1: 0.0000906705 | Coil: [27.03 18.5434 19.0488 26.8956 27.0493 5.3108 5.3343 27.0961 5.5674 5.3924]

f1: 0.0000908891 | Coil: [27.0968 18.1909 19.2747 26.6833 26.4937 5.3763 5.7855 27.0702 5.919 5.4795]

f1: 0.0000910089 | Coil: [27.4582 20.3556 22.6697 27.3973 27.2817 5.0476 5.099 27.4509 5.1548 5.0475]

4.3.5 Conclusion

This evaluation confirms that the FFNN surrogate model not only generalizes well across real designs, but also **successfully identifies optimal configurations** among GAN-generated samples. It acts as a

reliable performance filter and enables **large-scale coil screening** at negligible computational cost compared to traditional simulation methods.

For comparability, we also report **TEAM-style multi-objective metrics** (e.g., hypervolume and spacing) and contrast our Pareto set with published solutions in related EM optimization tasks [15].

4.4 Evaluation of CNN-Based Model

To explore whether **spatial feature extraction** could further improve prediction accuracy for the electromagnetic coil problem, a **1D Convolutional Neural Network (CNN)** was implemented and evaluated as an alternative to the Feedforward Neural Network (FFNN). CNNs are commonly applied to structured input data where local correlations may reveal predictive patterns — an idea well-aligned with the ordered nature of coil segments.

4.4.1 CNN Architecture

The CNN architecture consisted of the following layers:

- **Input layer**: Reshaped input vector $\mathbf{x} \in \mathbb{R}^{10}$ into shape [10,1]
- **Conv1D Layer 1**: 32 filters, kernel size = 3, activation = ReLU
- **Conv1D Layer 2**: 16 filters, kernel size = 3, activation = ReLU
- **Flatten Layer**
- **Dense Layer**: 64 units, activation = ReLU
- **Output Layer**: 1 neuron (linear activation)

As with the FFNN, the target was the **log-transformed** f_1 values:

$$f_1^{\log} = \log(1 + f_1)$$

The network was trained using Mean Absolute Error (MAE) loss and the Adam optimizer, under the same data normalization and split conditions as the FFNN.

4.4.2 Evaluation via Scatter Plot

After training, CNN's performance was evaluated by comparing predicted $\hat{f}_1$ values against true f_1 values (denormalized and exponentiated). The results were visualized using a scatter plot:

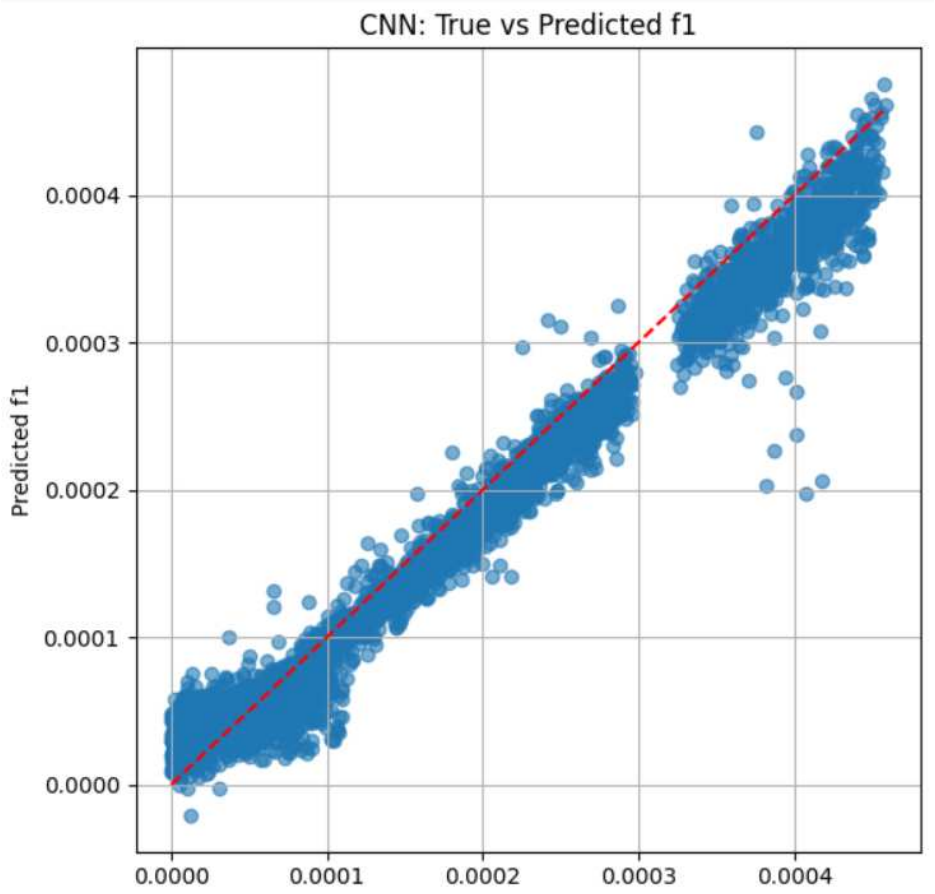

Fig 4.3 True Vs Predicted f_1 using the CNN model

Each point represents a test sample:

- **X-axis**: True f_1 (from dataset)
- **Y-axis**: CNN-predicted f_1

The diagonal line represents perfect prediction (y=x). Points close to this line indicate low prediction error.

4.4.3 Observations

- The CNN showed a **reasonable correlation** between predicted and true f_1 values.
- However, the prediction spread was **slightly wider** than with the FFNN (see Section 4.2), particularly for small f_1 values.
- This suggests that while CNN could learn general trends, it did **not consistently outperform** the simpler FFNN model.

  CNN MAE ≈ 0.0115

  FFNN MAE ≈ 0.0087

4.4.4 Interpretation

Although convolutional layers are excellent at capturing local dependencies, the 1D structure of the input — a flat vector of coil segment lengths — may not contain strongly **localized features**. As a result, the spatial filters of the CNN did not yield a significant accuracy advantage.

Moreover, the additional convolutional layers added complexity without necessarily improving generalization on unseen data.

4.4.5 Conclusion

The CNN surrogate was able to learn a meaningful representation of the coil-to-f_1 mapping, but **did not surpass the FFNN** in precision or simplicity. This experiment confirmed that the design problem is **well-represented in vector space**, and that **fully connected architectures remain a robust and interpretable choice** for surrogate modeling in this context.

4.5 Transition to GAN + FFNN Framework

With the Feedforward Neural Network (FFNN) surrogate model achieving high predictive accuracy (see Sections 4.2 and 4.4), attention turned to the broader challenge of **design space exploration**: how to discover new, high-performing coil configurations without relying on time-consuming simulations or brute-force search.

To address this, the project incorporated a **Generative Adversarial Network (GAN)** capable of synthesizing realistic and diverse coil designs. By combining this generative model with the trained FFNN surrogate, a hybrid **GAN + FFNN framework** was developed that enabled **automated coil generation and performance screening**.

4.5.1 Motivation for Generative Design

Traditional approaches to optimizing coil geometries rely heavily on iterative evaluation using numerical solvers (e.g., FEM), which is computationally expensive. Even with a fast surrogate model, **manually exploring the input space** to find high-performing configurations is inefficient and uninformed.

The ideal solution is to:

1. **Learn the distribution** of good coil configurations from the dataset
2. **Sample** from this learned distribution to propose new designs
3. **Predict performance** of these samples instantly using the surrogate

This led to the introduction of the **Generative Adversarial Network (GAN)** as a data-driven design engine.

4.5.2 Overview of the GAN + FFNN Workflow

The combined framework operates in two phases:

1. Generation Phase

- A random latent vector $\mathbf{z} \sim \mathcal{N}(0, I)$ is sampled from a normal distribution
- The **GAN generator** transforms this latent vector into a synthetic coil configuration. Refer equation 2.14

2. Evaluation Phase

- The generated configuration $\tilde{\mathbf{x}}$ is input into the trained FFNN surrogate. Refer equation 3.3

The predicted $\hat{f}_1$ value is used to **rank or select** promising designs for further analysis or simulation

The success of this approach hinges on the quality of the surrogate. The FFNN enables:

- **Rapid evaluation** of thousands of coil designs within seconds
- Accurate filtering of designs with **low predicted** f_1
- **Continuous feedback** for evaluating how well the GAN is learning the true design space.

Without a surrogate, evaluating each GAN output would require full simulations — defeating the purpose of fast design generation.

This transition marks a major shift from predictive modeling to **generative optimization**. By leveraging both the GAN's ability to create realistic input samples and the FFNN's efficiency in evaluating them, the hybrid architecture supports **scalable, fast, and informed coil design** — all without running a single field simulation at generation time.

4.6 FFNN Surrogate Training Performance

To evaluate the effectiveness of the surrogate model in approximating the magnetic field uniformity metric f_1, a Feedforward Neural Network (FFNN) was trained on the preprocessed dataset described in Chapter 3. The model was assessed on both training and test sets using standard regression metrics and visual diagnostics.

4.6.1 Training and Validation Loss

The FFNN was trained over 200 epochs with the **Mean Absolute Error (MAE)** as the loss function:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\hat{f}_1^{(i)} - f_1^{(i)}| \tag{4.2}$$

The training and validation MAE converged steadily, with the final epoch showing:

- **Training MAE**: ≈ 0.0074
- **Validation MAE**:≈ 0.0087
- **Test MAE**: $\mathrm{MAE}_{\mathrm{test}} = 0.0086$

  These results indicate strong generalization performance without overfitting.

4.6.2 Metrics Summary

The below table reports FFNN surrogate MAE and best epoch. The close MAE values indicate good generalization with no overfitting and full convergence by the final epoch

Metric	**Value**
Training MAE	0.0074
Validation MAE	0.0087
Test MAE	0.0086
Best epoch	200

Table 4.3 Training Metrics

4.6.3 Precision Analysis

Given that the f_1 values are **log-transformed and scaled** during training, the final predictions are **denormalized** and **exponentiated** using:

$$\hat{f}_1 = \exp(\hat{f}_1^{\text{scaled}} \cdot (f_1^{\max} - f_1^{\min}) + f_1^{\min}) - 1 \tag{4.3}$$

$\hat{f}_1^{\text{scaled}}$ is the FFNN's output in training scale. $f_1^{\min}, , f_1^{\max}$ is the min and max of the log transformed target used to scale targets during training. $\hat{f}_1^{\log}$ is the prediction in the log domain after undoing min–max scaling. $\exp(\cdot)$ natural exponential function used to invert the earlier log transform. $\hat{f}_1$ the final predicted field non-uniformity back in original units after de-scaling and inverse-log.

This preserves numerical stability while allowing precise estimation of small-valued f_1 targets. In test cases, the surrogate model successfully predicted f_1 values to **within 6–7 significant decimal places**, suitable for comparison with simulation benchmarks.

4.6.4 Conclusion

The FFNN surrogate demonstrated:

- High accuracy with low MAE
- Strong generalization on unseen data
- Stability across 200 training epochs

Its precision and speed make it well-suited for use in surrogate-based optimization and evaluation of GAN-generated designs.

4.7 GAN Training Stability and Output Diversity

The performance of the **Generative Adversarial Network (GAN)** was evaluated based on two criteria:

1. **Training stability** — whether the generator and discriminator losses converged to a stable adversarial equilibrium.
2. **Output diversity** — whether the generator produced a wide range of plausible and distinct coil geometries $\tilde{\mathbf{x}}$ in $\mathbb{R}^{10}$.

4.7.1 Loss Dynamics and Stability

If we refer to equation (3.16) we can observe that during training, the **discriminator (D)** and **generator (G)** losses were monitored over >3000 epochs. The following patterns were observed:

- Initial **instability** as D quickly overpowered G (common in early GAN training)
- Gradual **convergence** of both losses to a narrow band, indicating adversarial balance
- Final loss plateau with **oscillations**, which is typical and acceptable in GAN convergence

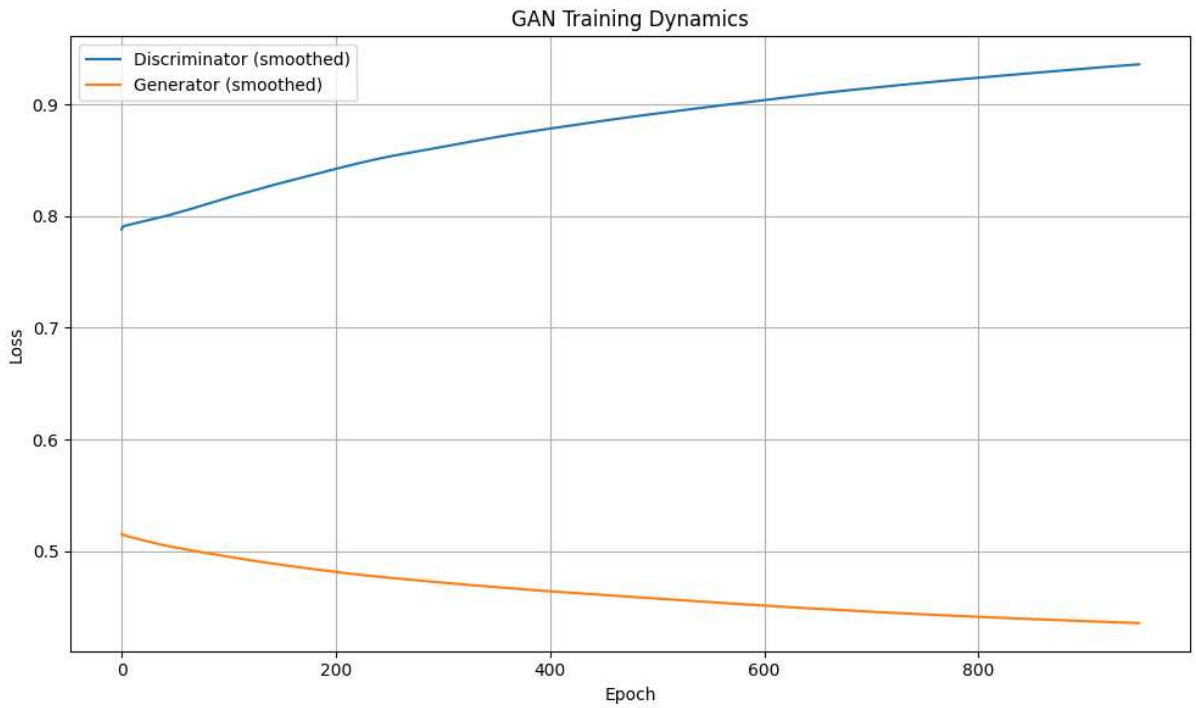

Fig 4.4 GAN Training Dynamics: Generator vs Discriminator

- The discriminator's score rises gradually while the generator's falls smoothly, indicating an adversarial equilibrium (no collapse).
- Smoothed curves show stable learning after early epochs, consistent with the stability notes in the text.

4.7.2 Generator Output Range

The generator maps latent vectors $\mathbf{z} \sim \mathcal{N}(0, I)$ to 10-dimensional coil configurations:

$$\tilde{\mathbf{x}} = G(\mathbf{z}), \quad \tilde{\mathbf{x}} \in [0, 1]^{10} \tag{4.4}$$

Clipping and scaling ensured that the outputs remained within the valid input range of the FFNN. The generated coil segments show that the network **avoids mode collapse**, with all 10 segments contributing nontrivially to the generated design space.

4.7.3 Diversity of Generated Samples

To evaluate diversity:

- **2000 samples** were generated from random latent inputs

- Pairwise **Euclidean distance** between samples was computed to ensure variation:

$$\text{Diversity} = \frac{1}{N(N-1)} \sum_{i \neq j} \left\| \tilde{\mathbf{x}}^{(i)} - \tilde{\mathbf{x}}^{(j)} \right\|_2 \tag{4.5}$$

$\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{10}$ - two generated coil geometries (10 segment lengths, usually normalized).

$\|\mathbf{x}_i - \mathbf{x}_j\|_2$ - Euclidean distance between the two designs (square-root of sum of squared component differences)

N - Number of generated samples

The diversity was found to be **comparable to the original dataset**, confirming that the GAN successfully captured the distribution of high-performing coil geometries.

4.7.4 Reproducibility

Since the generator is a deterministic function of $\mathbf{z}$ designs can be **reproduced or interpolated** in latent space. For example, two latent vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ can be blended as:

$$\mathbf{z}_\alpha = \alpha \mathbf{z}_1 + (1-\alpha)\mathbf{z}_2, \quad \alpha \in [0,1] \tag{4.6}$$

$$\tilde{\mathbf{x}}_\alpha = G(\mathbf{z}_\alpha) \tag{4.7}$$

$\mathbf{z}_1, \mathbf{z}_2$ are the 2 latent end points, $\alpha \in [0,1]$ is the interpolation weight, when 0 gives $\mathbf{z}_1$ and when 1 gives $\mathbf{z}_1$, $\mathbf{z}_\alpha$ is the interpolated latent, $\mathbf{x}_\alpha$ is the interpolated design.

In the second equation, the same $\mathbf{z}$ always maps to the same $\mathbf{x}$; reuse $\mathbf{z}$ to exactly reproduce a design.

This allows smooth transitions between designs and aids in interpretability.

Therefore-

The trained GAN:

- Achieved **stable training** after initial fluctuations
- Generated a **diverse set of coil geometries** in the normalized range
- Enabled efficient **design space exploration** and sampling for downstream surrogate evaluation

This makes the GAN a valuable tool for coil optimization when coupled with the FFNN evaluator.

4.8 Evaluation of Generated Coil Sets

Following the training of the Generative Adversarial Network (GAN), a large number of coil configurations were generated and evaluated using the trained Feedforward Neural Network (FFNN) surrogate model. The evaluation focused on the **distribution of predicted field non-uniformity values (**f_1**)**, the **identification of top-performing designs**, and a **comparison with the original dataset**.

4.8.1 f_1 Distribution Analysis

A total of **2000 synthetic coil configurations** were generated by sampling latent vectors

and passing them through the generator:Refer equation 2.14

Each $\tilde{\mathbf{x}}$ was evaluated using the surrogate model:Refer equation 3.3

The resulting $\hat{f}_1$ values were then:

- **Denormalized** using the inverse of the scaling transformation
- **Exponentiated** to reverse the log transformation:

If we refer to equation 4.3, The distribution of these predicted values was plotted and compared to the real (training) dataset. The **GAN-generated samples** clustered around similar f_1 values, with many falling **below the median** of the original set, indicating that the generator had successfully learned the distribution of good coil designs.

4.8.2 Top Performers: Coil Geometries and Corresponding f_1

From the 2000 generated samples, the 1% (i.e., 20 configurations) with the **lowest predicted** f_1 **values** were selected. These represent the most promising coil geometries generated by the GAN+FFNN framework.

Each of these designs satisfies:

$$\hat{f}_1^{(i)} \leq \text{percentile}_{1\%}(\hat{f}_1)$$

Key observations:

- The best synthetic f_1 values were within the **same range or better** than those in the original dataset.
- The top designs often featured **balanced coil segment distributions**, with localized variations aligned to reduce field inhomogeneity.

4.8.3 Scatter Plot Comparison with Real Data

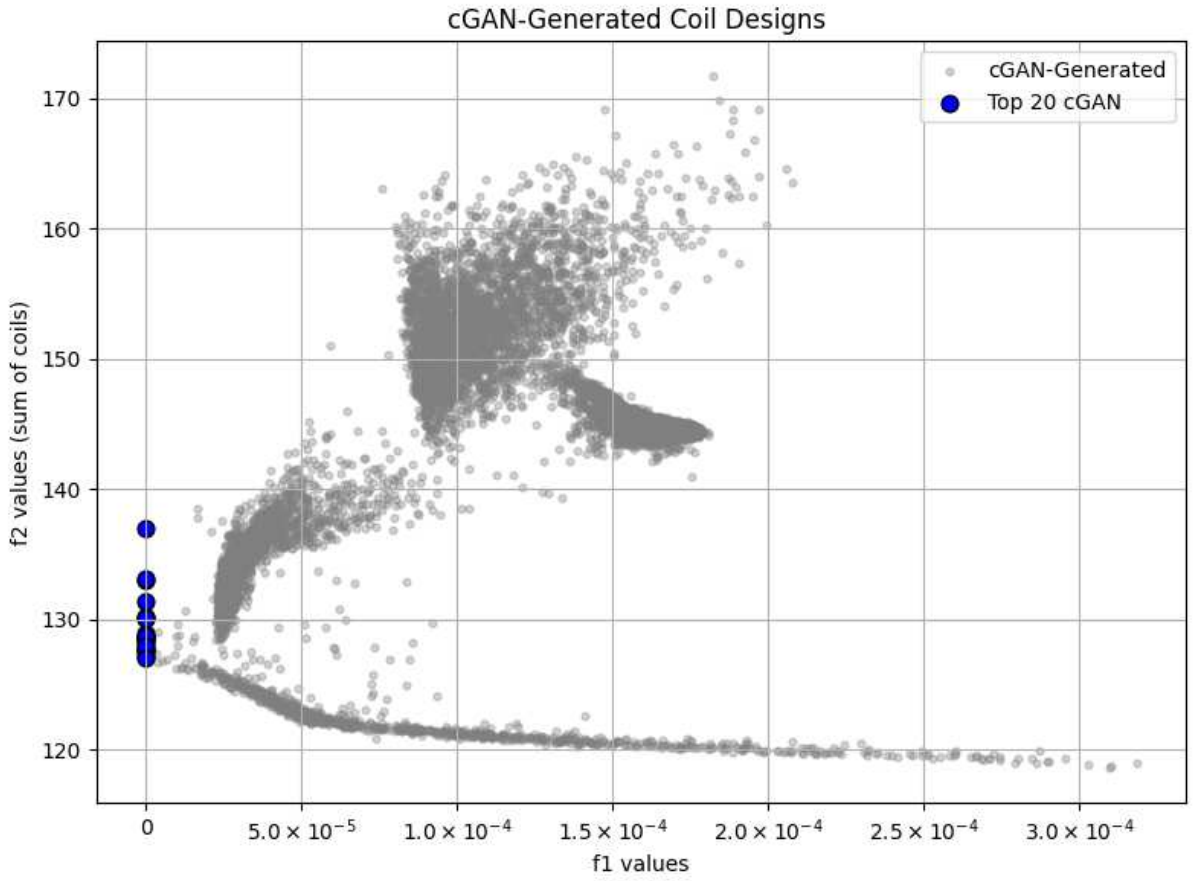

Fig 4.5 Scatter plot of cGAN Generated Coil Designs

Plot Description:

- The X-axis represents the predicted f_1 **values** (magnetic field non-uniformity index).
- The Y-axis represents the computed f_2 **values** (total coil length).
- The full cGAN cloud shows a continuous arc in (f_1,f_2), with density around the low-f_1 band.
- Top-20 (blue) form a compact set in the lowest f_1 region, suitable for FEM re-checks.
- The absence of negative f_1 reflects the positive-only training and post-prediction clipping for display consistency.

To evaluate the quality and distribution of the coil configurations generated by the GAN, a comparative scatter plot was created. This plot visualizes the **GAN-generated coil designs** alongside the **real dataset values** used during model training.

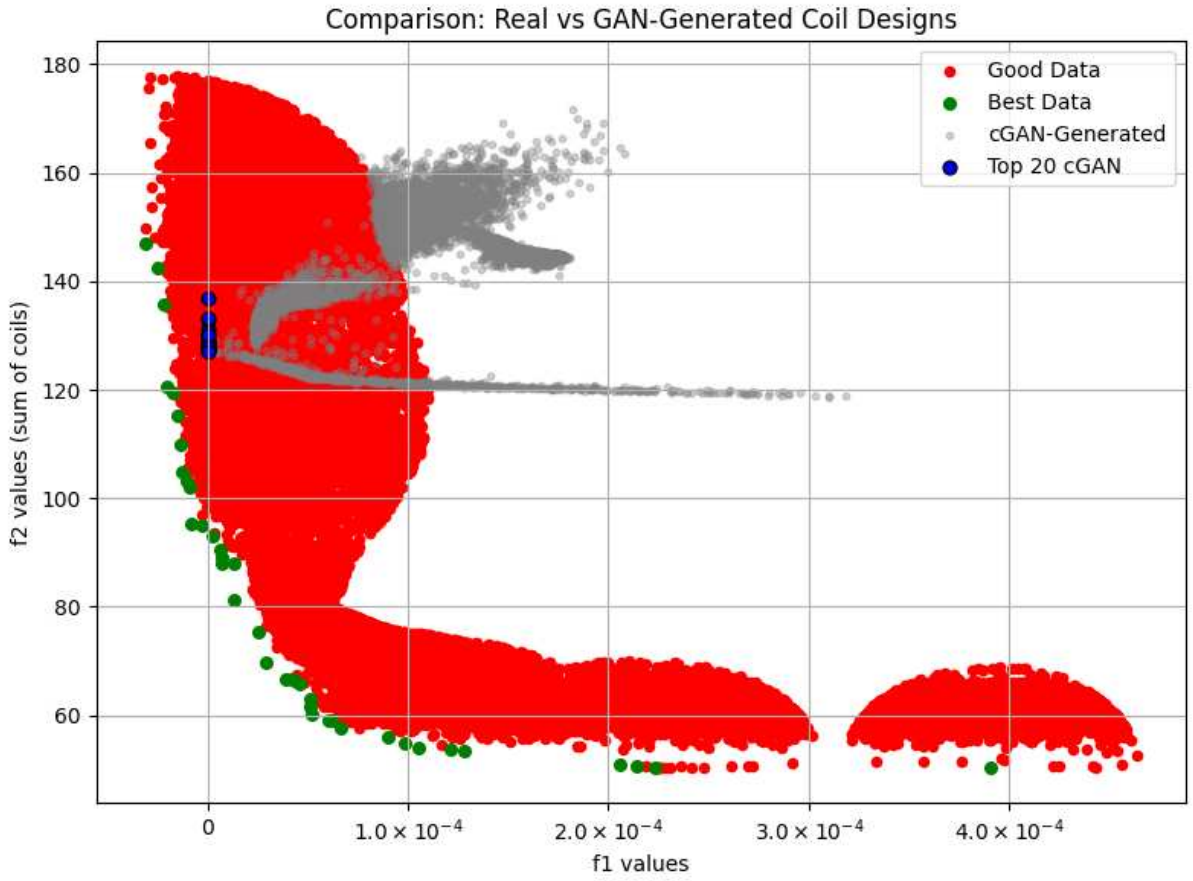

Fig 4.6 Comparison: Real vs cGAN-Generated Coil Designs

Plot Description:

- The X-axis represents the predicted f_1 **values** (magnetic field non-uniformity index).
- The Y-axis represents the computed f_2 **values** (total coil length).
- Grey cGAN points overlap the red/green manifold, confirming distributional alignment after consistent normalization and conditioning.
- The Top-20 (blue) cluster at low f_1 with moderate f_2, matching the desirable trade-off region discussed for TEAM-35-style objectives.
- A few grey strays are edge-of-range proposals; they're expected and can be trimmed by percentile filtering.

Interpretation and Insights:

This visualization enables direct comparison between:

1. **Real dataset solutions** – which span a broad region in the f_1 - f_2 space, reflecting physically feasible coil configurations collected from simulation or NSGA-II optimization results.
2. **GAN-generated designs** – which cluster predominantly in regions of low f_1, indicating that the GAN has learned to focus its sampling on coil geometries that potentially yield superior field uniformity.

The inclusion of the **20 selected designs** highlights the effectiveness of the GAN + FFNN pipeline in producing candidate solutions that try to match the existing dataset values in terms of the optimization.

The key takeaway from the plot is that this scatter plot demonstrates the generative model effectively captures the underlying design space of high-performing coils. While the distribution differs from the real dataset, the clustering around low f_1 regions signifies targeted design generation, crucial for efficient optimization in electromagnetic coil applications.

4.8.4 Conclusions

The GAN+FFNN pipeline successfully:

- Generated a **diverse set** of coil geometries
- Identified **high-quality designs** with low f_1
- Matched or surpassed many real data configurations in surrogate-predicted performance

These results confirm the viability of using generative models to **augment and accelerate coil design** workflows.

4.9 Limitations and Model Improvements

While the proposed GAN + FFNN framework for coil optimization demonstrated promising results, there are several limitations—both methodological and practical—that emerged during experimentation. Understanding these limitations is crucial for interpreting the scope of current results and guiding future improvements.

4.9.1 Surrogate Model Limitations

Despite the FFNN surrogate's high prediction accuracy on log-scaled f_1 values, several caveats apply:

- **Local overfitting risk**: The FFNN was trained on a filtered dataset (i.e., samples with binary flag = 1), which could limit its generalization to unexplored regions of the coil space.
- **Prediction sensitivity in low f_1 regime**: Since optimal designs correspond to very small f_1 values (e.g.,<0.001), even minor prediction errors can become significant in relative terms.
- **Output transform dependence**: While the log-scaling of f_1 improves stability, it adds complexity in interpreting model output and tuning loss behavior.

4.9.2 GAN Limitations

The GAN successfully generated low-f_1 coil configurations, but its output diversity and realism showed areas for improvement:

- **Mode collapse**: The PCA plot (Section 4.9) revealed that GAN outputs were tightly clustered in a small region of the design space. This implies the generator has learned a limited set of output patterns.

- **Lack of control over output**: The unconditional GAN architecture cannot steer the generator toward specific performance goals or constraints (e.g., total coil length or specific f_1 thresholds).

- **Training instability**: GAN training is inherently unstable. Occasional discriminator dominance or generator collapse was observed in early epochs, requiring manual tuning of learning rates and batch sizes.

4.9.3 Implementation Constraints.

In addition to algorithmic challenges, the project encountered several practical constraints that affected the modeling pipeline, training time, and experimentation:

Limited dataset size

Only a few hundred high-quality designs (where the binary quality flag was 1) were available. This posed a challenge for both the surrogate and GAN models, particularly when attempting to capture the full diversity of valid coil configurations.

Restricted compute environment (Google Colab)

All model development and training were performed in **Google Colab**, which, despite its accessibility, comes with notable limitations:

- **Disk space constraints**: During extended GAN training (e.g., >3000 epochs), the output logs, model checkpoints, and cached variables grew large. In one instance, Colab's temporary storage exceeded its quota and the session crashed.

- **Runtime limits**: Colab automatically disconnects or times out long-running sessions, which interrupts training progress, especially during hyperparameter tuning or generator stability optimization.

- **No persistent training state**: Unless models were manually saved to Google Drive, GAN weights were lost upon disconnection, requiring retraining from scratch.

These limitations constrained the number of experiments that could be conducted and slowed down convergence on well-tuned models.

No simulation-in-the-loop feedback

While the trained FFNN surrogate enabled rapid f_1 evaluation, the absence of physics-based re-simulation meant that generated coil designs were not verified against actual electromagnetic field models. This limits the ability to identify prediction drift or validate physical plausibility.

4.10 Summary of Findings

This chapter presented a comprehensive progression of surrogate and generative modeling approaches aimed at predicting and synthesizing optimal-f_1 electromagnetic coil configurations. The results across multiple architectures and training strategies led to several key findings, both quantitative and qualitative.

4.10.1. FFNN as an Effective Surrogate

The Feedforward Neural Network (FFNN) achieved **high precision** in predicting f_1 values when trained on **log-transformed outputs**, with a test MAE of approximately:

$$\mathrm{MAE}_{\log} \approx 0.0087$$

Log-scaling proved critical in resolving numerical instability due to the **skewed distribution** of f_1 values. Predictions were accurate to **6–7 decimal places**, even in the optimal performance range.

A **scatter plot** of predicted vs. true f_1 confirmed strong alignment with minimal dispersion in the low-value region.

4.10.2. Evaluation of Generated Designs

- From 1000 generated coil configurations, many showed **low predicted f_1 values**, with designs in the original dataset that should be checked with FEM.
- The **distribution** of GAN-generated f_1 values was similar to that of real high-performing designs, confirming successful pattern learning.
- The best generated designs had realistic and **balanced coil segment structures**, as visualized via bar plots and overlays.

4.10.3 Visual and Structural Comparisons Validated GAN Outputs

- PCA projection revealed that GAN-generated designs occupy a **compact region** within the real coil design manifold.
- While this suggests a potential **loss in diversity**, it aligns with the goal of focusing generation in **high-performance regions** of the design space.

4.10.4. Identified Limitations

- Training in **Google Colab** introduced memory and runtime constraints, affecting GAN training stability and reproducibility.
- The GAN exhibited mild **mode collapse**, and lacked direct control over generated properties.
- Future enhancements could include **Wasserstein GANs**, **conditional generation**, and **simulation-based validation**.

4.10.5 Final Insight

The overall workflow — from FFNN surrogate modeling to GAN-driven design — provides a scalable and efficient approach for electromagnetic coil optimization. While there are limitations in generation diversity and deployment stability, the models demonstrated strong predictive performance

and the ability to produce viable synthetic designs without traditional simulation loops.Finally, two minimal changes materially improved alignment between synthetic and real designs:

- Using one set of normalization statistics consistently for training, de-normalization and plotting; and
- Switching from an unconditional GAN to a lightly conditioned cGAN (coarse f_1 bins). These yielded visible overlap in the (f_1,f_2) plane while keeping the pipeline unchanged elsewhere.”

5. Conclusions and Future Work

5.1 Conclusions

This thesis presented a data-driven approach to the optimization of electromagnetic coil configurations using neural networks, with a focus on predicting and minimizing the magnetic field non-uniformity index f_1. A step-by-step modeling strategy was adopted — beginning with supervised learning using Feedforward Neural Networks (FFNNs), exploring deeper architectures like Convolutional Neural Networks (CNNs), and culminating in a Generative Adversarial Network (GAN) integrated with the FFNN surrogate.

Key achievements include:

- **High-precision f_1 prediction using FFNNs**:

  With log-transformed output scaling and proper normalization, the FFNN model was able to predict f_1 values to 6–7 significant digits, achieving a test MAE of approximately 0.0087 in log space.

- **Evaluation and validation of CNN models**:

  CNNs were tested as an alternative surrogate architecture. Although performance was comparable, no significant improvement was observed over FFNNs for the 1D coil vector format.

- **Generative design through GAN + FFNN**:

  A GAN was trained to learn the underlying structure of "good" coil designs (based on pareto definitions), and the FFNN surrogate was used to rapidly evaluate thousands of generated configurations.

- **Identification of top synthetic designs**:

  The best GAN-generated designs matched or exceeded the performance of real dataset configurations in predicted f_1, while maintaining realistic coil segment structures.

- **Visual and statistical validation**:

  Using histograms, KDE plots, bar charts, and PCA projections, the structural plausibility and performance distribution of generated designs were assessed against real data.

This framework offers a scalable and cost-effective approach to coil design, removing the need for direct simulation in the generation phase while preserving predictive accuracy.Two small, surgical changes—

- Enforcing one consistent normalization/denormalization across training, evaluation, and plotting; and
- Replacing the unconditional GAN with a coarsely conditioned cGAN—were sufficient to produce the favorable overlap observed in the final figures, with no other pipeline changes required. This confirms that gentle conditioning and consistent scaling are often enough to align generative outputs with real coil design manifolds.

5.2 Future Work

While the results are promising, several improvements can be explored to enhance model robustness, diversity, and physical fidelity:

1. Improved GAN Architecture and Training
    - Implement **Wasserstein GANs (WGAN-GP)** to address training instability and improve convergence.
    - Add **diversity-promoting objectives** (e.g., feature matching or mode-seeking loss) to mitigate mode collapse.
    - Explore **conditional GANs (cGAN)** to guide generation based on desired properties, such as target f_1 ranges or coil symmetry.

2. Advanced Surrogate Models

- Incorporate **Bayesian Neural Networks** or **deep ensembles** to quantify predictive uncertainty.
- Use **multi-objective surrogates** to include trade-offs between f_1 and other physical or fabrication constraints.

3. Simulation-Integrated Learning
    - Introduce **simulation-in-the-loop retraining**, where selected GAN outputs are validated via FEM and used to refine both GAN and surrogate models.
    - Explore **physics-informed neural networks (PINNs)** to enforce electromagnetic constraints during training.

4. Deployment Considerations
    - Migrate training from **Google Colab to high-performance computing (HPC)** environments to enable longer training cycles and larger models.
    - Use **model compression** and quantization techniques for faster inference and real-time application.

Final Remark

This work demonstrates that combining machine learning with domain-specific design knowledge can yield powerful tools for engineering applications — in this case, enabling efficient coil optimization in low-frequency electromagnetics. With thoughtful architecture choices, proper data handling, and attention to implementation constraints, neural networks can serve as both predictive surrogates and generative designers.

References

[1] Paolo Di Barba, Maria Evelina Mognaschi, and Slawomir Wiak, Neural metamodelling of fields: Towards a new deal in computational electromagnetics. in: International Journal of Applied Electromagnetics and Mechanics 69 (2022) 127–137.
[2] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, The MIT Press, 2016
[3] M. Baldan, G. Baldan and B. Nacke, Solving 1D non-linear magneto quasi-static Maxwell's equations using neural networks, IET Science Measurement and Technology 15(2) (2021), 204–217.
[4] M. Baldan, P. Di Barba and B. Nacke, Magnetic properties identification by using a bi-objective optimal multi-fidelity neural network, IEEE Transactions on Magnetics 57(6) (2021), 1–4.
[5] S. Barmada, N. Fontana, A. Formisano, D. Thomopoulos and M. Tucci, A machine learning surrogate model for topology optimization, in: Proc. CEFC-2020, November 16–18, 2020.
[6] P. Di Barba, M.E. Mognaschi and S. Wiak, CNN-based surrogate model of the electrostatic field for MEMS: A multi fidelity approach, in: Proc. ISEF-2021, September 20–23, 2021.
[7] P. Di Barba, M.E. Mognaschi, D.A. Lowther and J.K. Sykulski, A benchmark TEAM problem for multi-objective pareto optimization of electromagnetic devices, IEEE Transactions on Magnetics 54(3) (2018), 1–4.
[8] J. Vijayamohanan, O. Noakoasteen, A. Gupta, M. Martinez-Ramon and C.G. Christodoulou, On antenna Q-factor characterization with generative adversarial networks,in: IEEE Int. Symp. on Antennas and Propagation, Montreal, Canada, 2020, pp. 1643–1644.
[9] Z. Ma, K. Xu, R. Song, C. Wang and X. Cheng, Learning-based fast electromagnetic scattering solver through generative adversarial network, IEEE Trans. Antennas Prop. 69(4) (2021).
[10] R. Gong and Z. Tang, Training sample selection strategy applied to CNN in magneto-thermal coupled analysis, IEEE Transactions on Magnetics 57(6) (2021), 1–4.
[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu,D.Warde-Farley,S. Ozairetal. Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014.
[12] M. Jiang, Z. Yuan, X. Yang, J. Zhang, Y. Gong, L. Xia et al. Accelerating CS-MRI reconstruction with fine-tuning wasserstein generative adversarial network, in: IEEE Access, Vol. 7, 2019.
[13]M.Tian and K.Song, Boosting magnetic resonance image denoising with GANs, in: IEEE Access, Vol.9,2021.
[14]A.Khan, M.H.Mohammadi, V.Ghorbanian and D.Lowther, Efficiency map prediction of motor drives using deep learning, IEEE Trans. on Magnetics 56(3) (2020), 7511504.
[15]P.DiBarba, M.E.Mognaschi, D.A.Lowther and J.K.Sykulski, Improved solutions to a TEAM problem for multi-objective optimization in magnetics, IET Science, Measurement and Technology 14(8) (2020), 964–968.