Politecnico di Torino

Politecnico di Torino
MSc in Automotive Engineering

Computational Intelligence for the Design of Electrical Machines in the Automotive Sector

Candidate: Bharath Kanaiah Parthiban

Supervisors: Repetto Maurizio, Solimene Luigi

Academic Year 2024/2025

Abstract

The need for an energy efficient and high-performance traction system has been rapidly increasing over the years which has driven the interest in designing electric motors for electric vehicles (EV's). Among different motor types, the use of Interior Permanent Magnet (IPM) motors has been dominant because of the high torque characteristics and, most importantly, reliability. Anyway, because of the multi-physics nature of the design of motor which deals with thermal, electromagnetic and structural considerations, it becomes computationally expensive for the optimization problem when completely relying on the Finite Element Analysis (FEA) alone.

This thesis is built upon a data-driven strategy to improve the design and optimization of traction electrical motors by integrating high-quality simulations with machine learning techniques. A dataset of over 4096 IPM motor configurations was generated from an open electromagnetic finite element code (SyRe) integrated by other mechanical stress and temperature modules. Each sample consists of a set of design parameters and evaluated outcomes for that corresponding set of design parameters.

In the earlier stages Support Vector Regression (SVR) and Neural Networks (NN) were used as surrogate models which predicted the motor performance which took as input the dataset generated in the previous phase. To efficiently use models created on python, a bridge between PYTHON and MATLAB interface was created.

Due to the high number of objectives, the multi-criterion optimization process requires specific optimal algorithms. Thus, the trained surrogate models have been then integrated in a many-objective optimization algorithm NSGA-III in the PlatEMO environment. The optimization was carried out to optimize seven targets obtaining so a well-distributed Pareto front of optimal solutions.

Overall, this paper shows the potential of machine learning in reducing the complexity and computational cost in design of traction motors and increase the effectiveness of optimization processes in the initial stage of e-mobility applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the last two decades the development in EVs has been rising at an exponential rate because of several factors like:

· Environmental concerns

· Government policies and incentives

· Technological improvements and competitive cost

· Rising fuel prices

On the other hand, various subsidies, tax credits, grants, and bans on ICE in several countries is pushing the buyers towards EVs. From the public's perception, EVs are seen as modern, future-proof, eco-friendly and most importantly lower operating costs, and less maintenance. Modern EV also provides better performance with features like smooth acceleration, quite operation and fast charging which attracts a lot of customers around the world. All major car manufacturers are now making huge investments in EV platforms, covering many varieties of models like sedans, SUVs, trucks, vans, luxury, and budget vehicles.

Since all the factors discussed above are increasing the demand for EVs, the need for increase in production is also rising. But there are a lot of constraints that are limiting how fast EV production can grow. Some of the limitations are:

· Charging infrastructure gaps

· Supply chain disruptions (Limited availability of components like semiconductors)

· High initial investment requirements

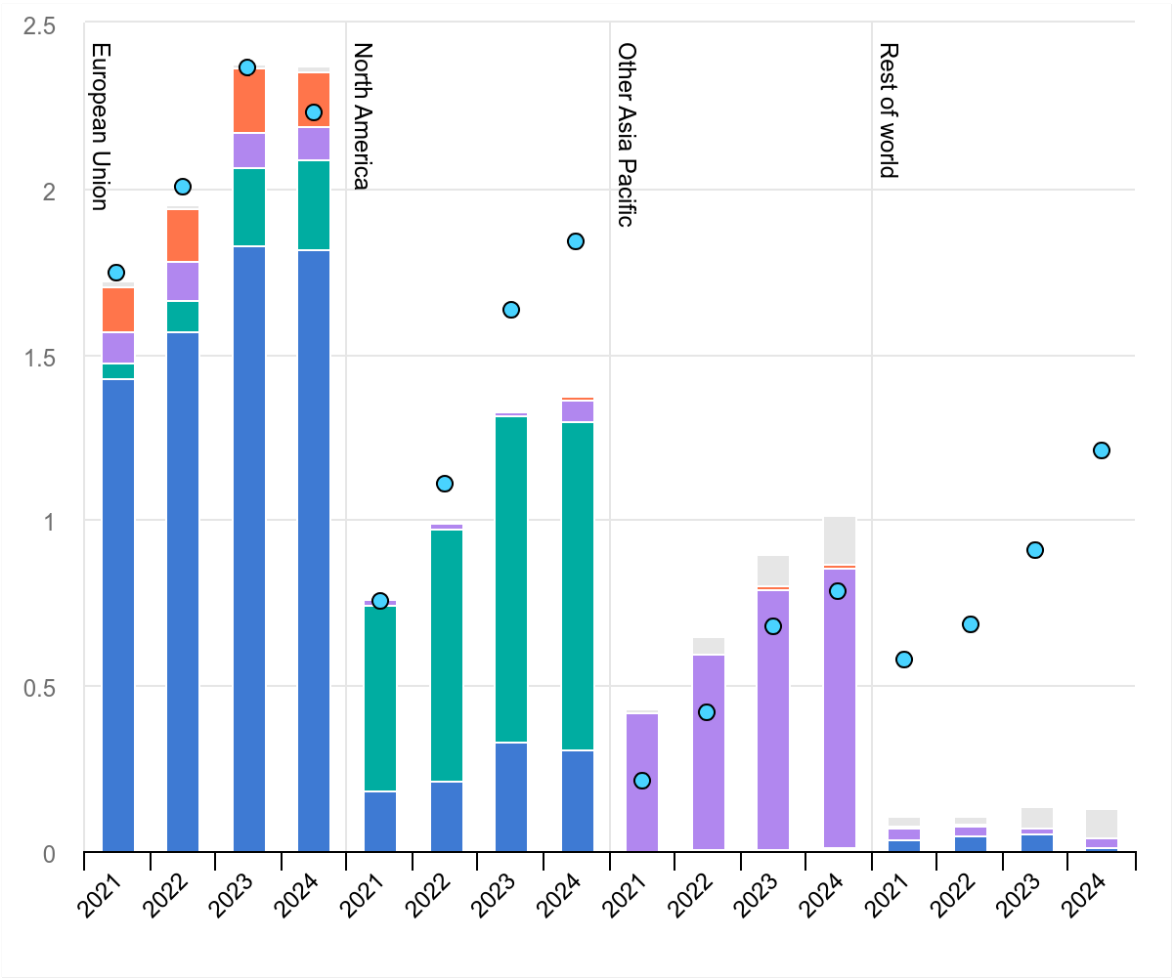


Figure 1.1: Global EV Market Growth (2010–2024)

Category	**Description**	**Impact on Cost/Timeline**
Battery Technology	Limited energy density, high material costs (lithium, cobalt), and long R&D cycles for solid-state batteries.	Raises the total price of vehicles: Testing and certification processes cause delays
Supply Chain & Raw Materials	Dependence on rare materials and geopolitical risks impacting supply stability	Causes price volatility and production delays.
Thermal Management	Efficient cooling systems are essential to prevent battery degradation and ensure safety.	Increases complexities in designing. Increases testing time.
Charging Infrastructure	Charging Infrastructure Lack of adequate infrastructure for fast charging networks around the world.	Market adoption will get slowed down.
Manufacturing Process	Need for specialized assembly lines, automation, and quality control for high-voltage components.	Increases capital expenditure, prolongs factory installation period.
Software & Integration	Sophisticated control algorithms and Communication between vehicles and the power grid must be extensively validated.	Lengthens development cycles because of software testing and certification.
Regulatory Compliance	Evolving safety and environmental regulations across markets.	Adds administrative burden and delays product homologation.

Table 1.1: Major Challenges in EV Production and Development

Production of EVs require huge investment costs because of the complexity in R&D for E-motor and battery scaling, unbelievably expensive raw materials due to shortage in their availability. Usually the process involved in the design of crucial EV components like a battery or a E-Motor is slower and less flexible when a traditional design approach is used and also is expensive adding up to the investment cost.

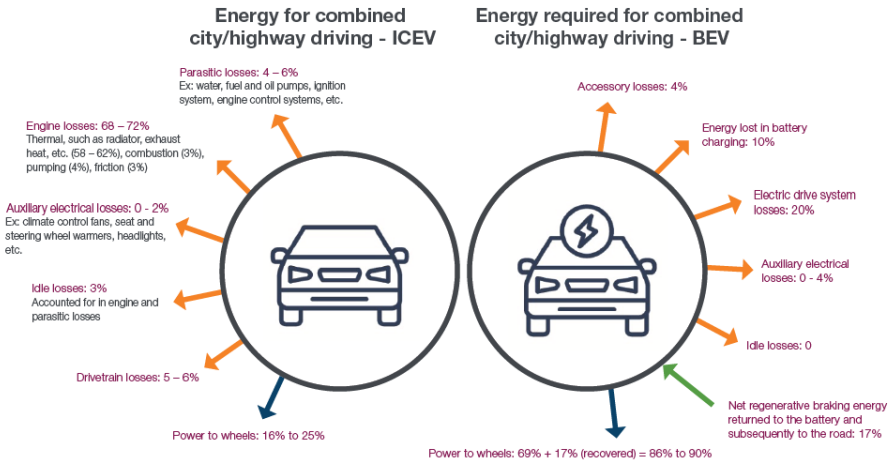


Figure 1.2: Comparison of ICE vs EV Powertrain Efficiency.Adapted from Togun et al. (2025)[2]

This process not only requires deep domain expertise but also powerful computational resources. In the recent days computation time and cost have been hugely cut down by using computational intelligence techniques like Machine Learning. By using optimization algorithms like Non-dominated Sorting Genetic Algorithm III (NSGA-III), it is possible to further refine the machine design by improving also the performance objectives at the same time.

1.2 Problem Statement

The most important part of an EV is the electric motor which is the heart of the vehicle. It requires complex designing and optimization to obtain the desired performance. The design process of an electric motor can be considered an optimization problem with multiple objectives because it involves various domains like thermal, mechanical, electromagnetics, fluid dynamics, structural and acoustics. Traditional design approaches, although accurate, have become computationally expensive and time-consuming, especially for multi-objective optimization. They require creating a cad model, running high quality simulations like FEA and then testing through physical prototypes.

The simulation phase alone in this process usually takes days or even weeks per iteration and further in exploring a multi-objective problem like in our case it might take several hundreds of such cycles. All these limitations make the traditional approach of using FEA for the design the E-Motors slower, expensive and rigid. This process also makes it tougher to explore different combinations design parameters to obtain target performance since changing one design parameter might affect all the others forcing multiple rounds of recalculation through the entire setup.

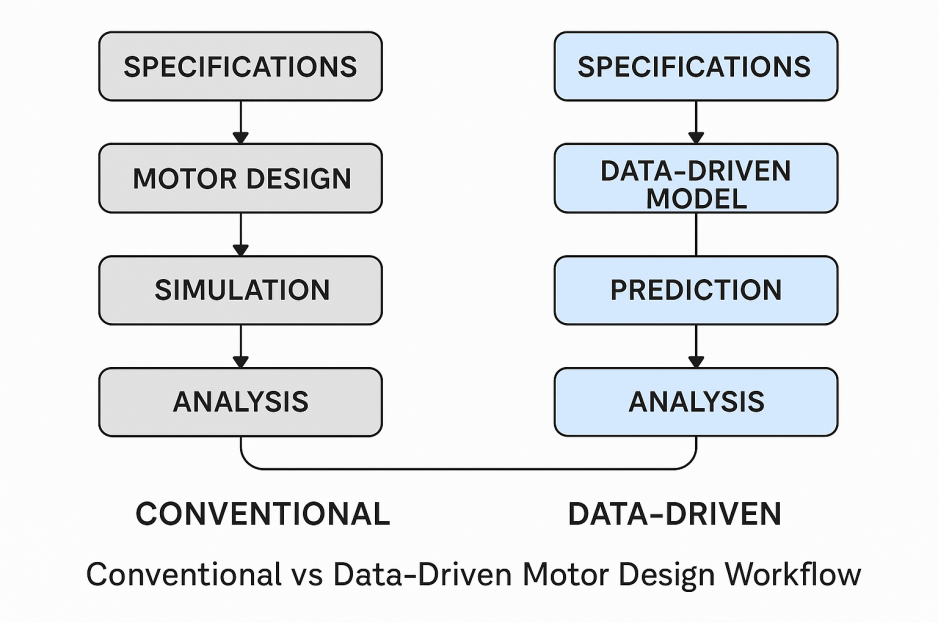


Figure 1.3: Conventional vs Data-Driven Motor Design Workflow.Image generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

So, there is need for a technique that is fast and reliable at the same time keeping the computation cost and complexity lower. This is where the computational intelligence techniques like Machine Learning comes and multi-objective optimization comes into play. By using this techniques we can reduce the computational burden associated with the finite element analyses.

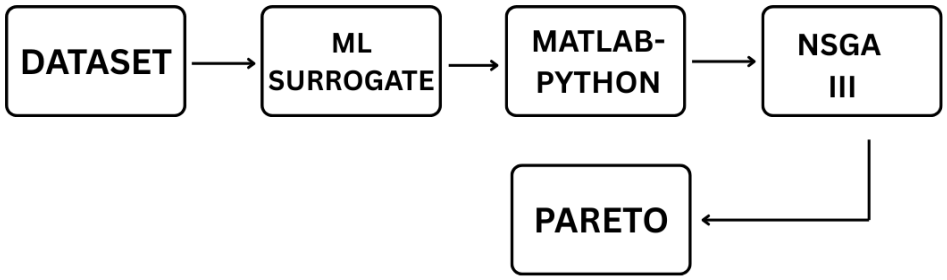


Figure 1.4: Overall Workflow of Proposed Methodology.Image generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

Machine learning models like Support Vector Regression (SVR) and Neural Networks (NN) help in the creation of fast and accurate predictors which take as input a dataset of past simulation or experimental data. Further by integrating with frame-

works like PlatEmo on MATLAB which uses optimization algorithms like NSGA-III it is possible to obtain much optimized design parameters. Although MATLAB remains a dominant tool for simulations because of the strong numerical and visualization capabilities, the use of python is a preferred choice by most of the developers since it has extensive libraries and flexibility.

Aspect	**Traditional Optimization**	**Surrogate-Based Optimization**
Evaluation Method	Directly evaluates the objective function using high-fidelity models (e.g., FEA, CFD).	Uses a surrogate or metamodel (e.g., SVR, NN, Kriging) to approximate the objective function.
Computation Cost	Very high due to repeated evaluations of expensive simulations.	Substantially reduced since most assessments are on the surrogate.
Optimization Speed	Slow – typically only a few iterations or low-resolution search space.	Fast – facilitates the exploration of large design spaces in less time.
Accuracy	High accuracy at each evaluation (ground truth).	The level of accuracy relies on the quality of the surrogate model but can be very close if properly trained.
Scalability	Difficult to scale for multi-objective or high-dimensional problems.	Scalable after the training of the surrogate model.
Data Requirement	No data requirements – operates directly on the simulation model.	Requires a representative training data to develop the surrogate.
Exploration vs Exploitation	Few expeditions because of cost.	Enables extensive exploration of the design space at low cost.
Robustness	Sensitive to noise and expensive to handle uncertainties.	Can incorporate uncertainty quantification and noise handling in training.
Application in MultiObjective Problems	High computational burden makes it impractical for many objective cases.	Appears well-suited for algorithms like NSGA-III with minimal extra cost.
Application	Used when computational resources are abundant or problem is simple.	Preferred for complex engineering problems with expensive evaluations.

Table 1.2: Comparison of Traditional Optimization vs Surrogate-Based Optimization

The challenges would be direct execution of the Python models from MATLAB

without actually launching Python interface separately and integrate them properly to PLATEMO framework to optimize further using algorithms like NSGA-III while maintaining the datatypes same between both the languages .

1.3 Machine Learning Techniques for Engineering Optimization

In the recent years, **Machine Learning (ML)** has emerged as a powerful branch of computational intelligence capable of solving complex engineering problems where analytical modelling or repeated high-fidelity simulations become impractical. ML algorithms learn the relationship between input design variables and output responses directly from data, enabling the creation of *surrogate models* that approximate expensive simulation results with high accuracy and drastically reduced computation time. In electrical motor design problems solved via *Finite Element Analysis (FEA)*, each computation is a finite element simulation. The first can take a few minutes to hours and is a limiting factor for design iterations that can be investigated. In comparison, once a given model has a sufficiently large dataset of simulated or experimental samples, a Machine Learning (ML) model can estimate predictive performance metrics—for example, electromagnetic torque, torque ripple, and/or efficiency—in a fraction of a second. That computational saving translates to a huge time and cost benefit during the early design and optimization phases.

Within the variety of supervised learning algorithms that exist in Machine Learning procedures, one important category is **Support Vector Regression (SVR)** and **Artificial Neural Networks (ANNs)**, which have proved especially strong in performance for nonlinear regression.

- **SVR** can be applied on the assumptions of identifying an optimal hyperplane to best fit the data while allowing for an acceptable level of error (ϵ-insensitive loss). Its major advantage is the provision of smooth and consistent model projections for medium-sized data with less overfitting, in case the right kernel function, perhaps *Radial Basis Function (RBF)*, is not applied.

- **Neural Networks**, on the other hand, are made up of layers of neurons capable of learning and approximating highly nonlinear mappings between inputs and outputs. With enough training samples and corresponding hyperparameters, they can improve their ability to generalize well to unseen designs, making them suitable for complex multiphysics systems like electric traction motors.

In essence, ML algorithms are used as surrogate and/or metamodels that substitute computationally costly FEA solvers in the process of optimization. After that, they are

incorporated into multi-objective optimization frameworks like the **Non-Dominated Sorting Genetic Algorithm III (NSGA-III)**—thousands of design variables to be evaluated. This integration enables a full examination of the total design space and discovery of opportunities within that space for improvement over a large number of Pareto-optimal trade-offs (such as maximizing torque while minimizing torque ripple) without having to perform time-consuming simulations for each candidate.

.

On the whole, the employment of ML-based surrogate models in engineering optimization problems provides a quick, economical, and versatile solution to the conventional design process. It not only speeds up the decision-making process but also allows designers to uncover more information about parameter sensitivities and boundaries of performance that would otherwise remain unexplored with conventional methods.

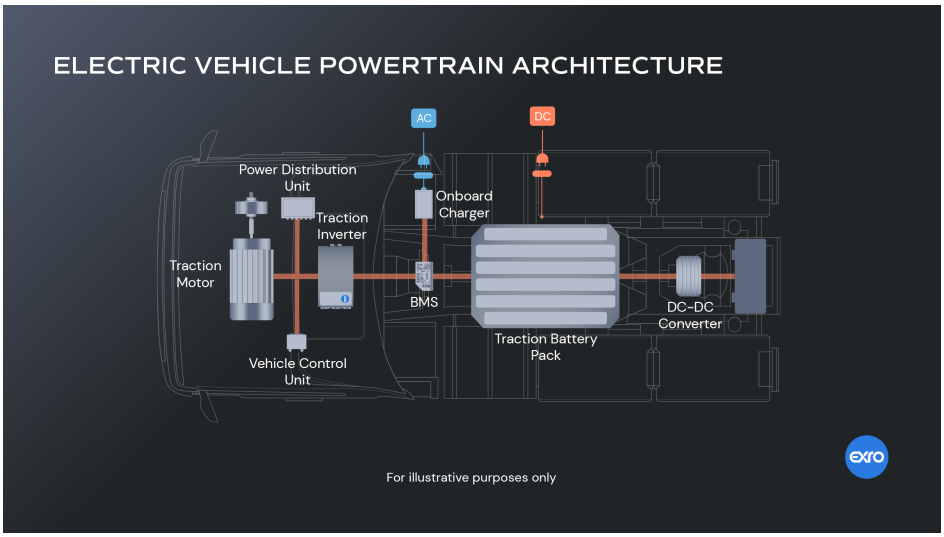


Figure 1.5: EV System Architecture.Image adapted from Exro Technologies, "EV Power Electronics Explained," 2024 [4].

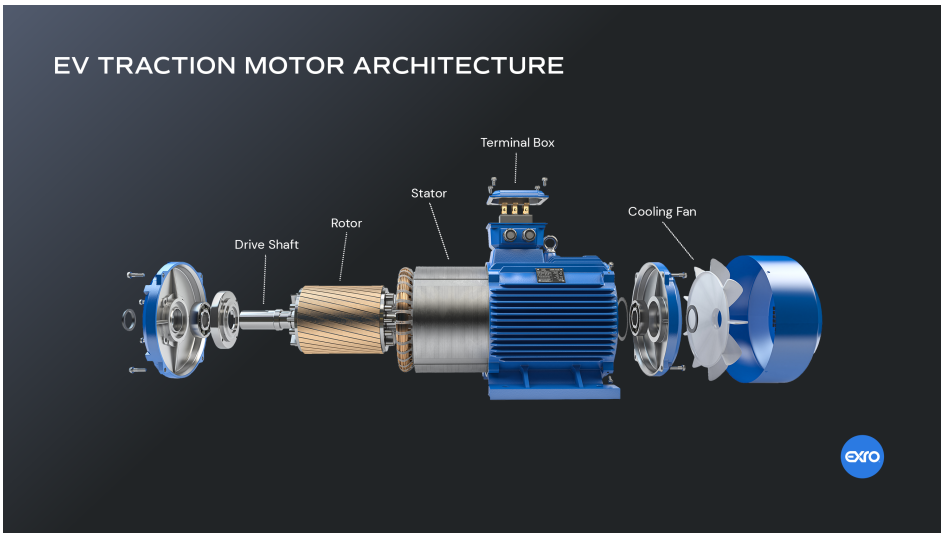


Figure 1.6: Traction Motor.Image adapted from Exro Technologies, "EV Power Electronics Explained," 2024 [4].

Chapter 2

Electrical Motors

2.1 Overview of Electrical Machines for Traction Applications

Electric traction motors are central to modern EV powertrains. They convert electrical energy into mechanical torque, driving motion. For EV applications, motors must meet the following requirements:

- High torque and efficiency
- Compact size and low weight
- Reliable operation under thermal and mechanical stress

Common EV motor types include:

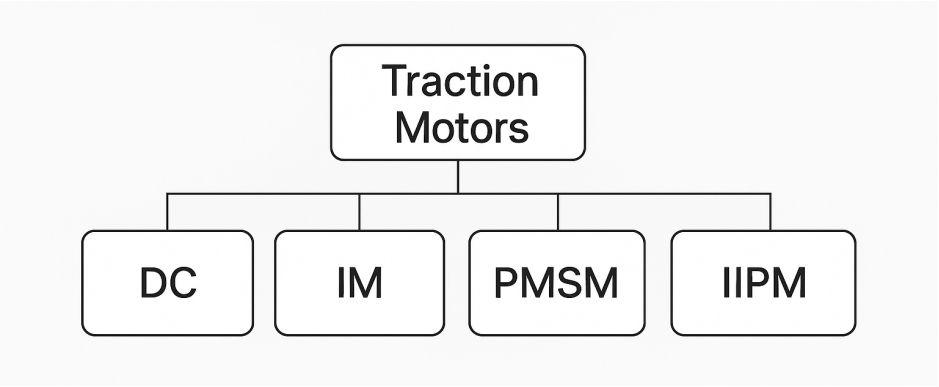


Figure 2.1: Classification of Traction Motors.Generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

- DC motors
- Induction Motors (IM)

- Permanent-Magnet Synchronous Motors (PMSM)
- Switched Reluctance Motors (SRM)

Among them, the Interior Permanent Magnet (IPM) motor has become dominant due to:

- High torque density
- Excellent field-weakening
- Superior efficiency

2.2 Working Principle of Interior Permanent Magnet Motors

In IPM motors, permanent magnets are embedded inside the rotor, introducing anisotropy between the d-axis and q-axis inductances.

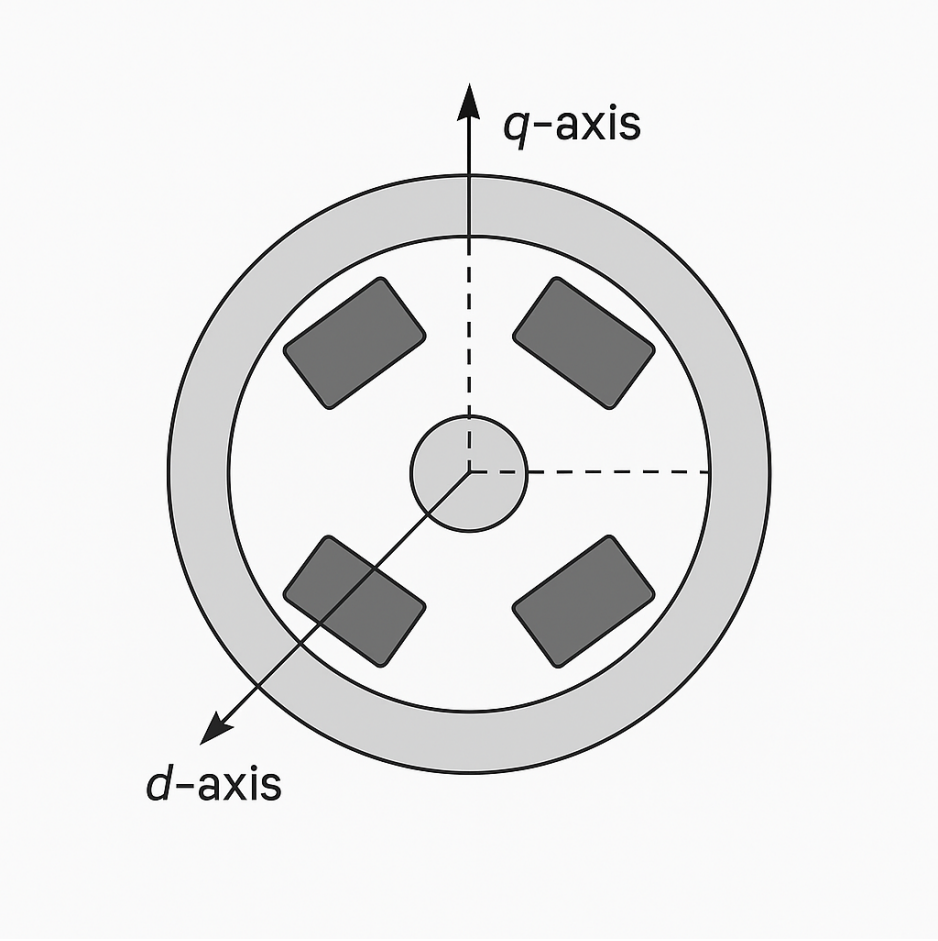


Figure 2.2: Cross-section of an IPM Motor showing d- and q-axes.Generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

The total torque consists of two components:

- Magnet torque
- Reluctance torque

The electromagnetic torque is given by:

$$T = \frac{3}{2} p \left[\psi_f i_q + (L_d - L_q) i_d i_q \right]$$

Symbol	**Definition**
T	Electromagnetic torque (Nm)
p	Number of pole pairs
ψ_f	Permanent magnet flux linkage (Wb)
i_d	d-axis stator current component (A)
i_q	q-axis stator current component (A)
L_d	Inductance along d-axis (H)
L_q	Inductance along q-axis (H)

Table 2.1: Symbols and Definitions used in Torque Equation

Torque ripple arises from:

- Cogging between stator and magnets
- Magnetic reluctance variation
- Inverter harmonics

Minimizing torque ripple is crucial for drive comfort and NVH performance.

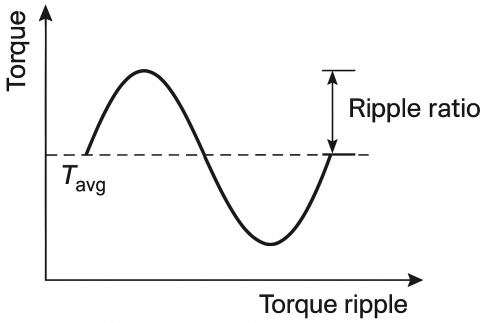


Figure 2.3: Torque Ripple Illustration. Generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

2.3 Multi-Physics Aspects of Motor Design

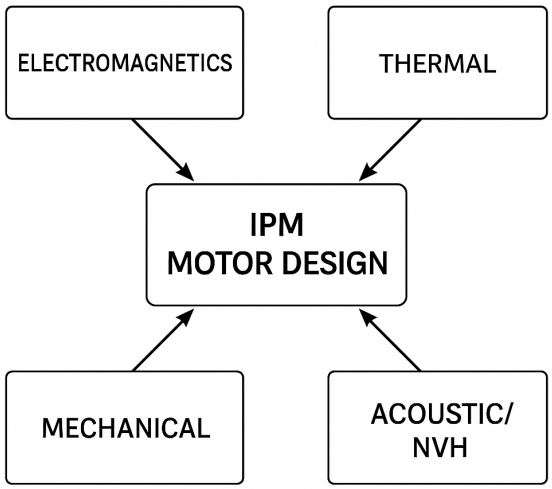


Figure 2.4: Coupled Multi-Physics Domains(Electromagnetic–Thermal–Structural).Generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

The IPM motor design is intrinsically multi-physics involving:

- **Electromagnetics** — Torque, Flux, Loss
- **Thermal** — temperature increase, cooling, magnet demagnetization
- **Mechanical** — rotor stress, deformation
- **Acoustic/NVH** — vibration, noise

The essential tool in analysis is FEA, but it is computationally intensive for large optimization problems. The multiphysics coupling process requires the application of surrogate models in the designing process.

2.4 Need for Optimization and Data-Driven Approaches

Design goals include:

- Maximize average torque and efficiency
- Minimize torque ripple, magnet mass, and losses

Optimizing all these manually with FEA is infeasible. Instead:

- Surrogate models (e.g., ML) are trained on FEA data
- Models are coupled with NSGA-III for multi-objective optimization
- Thousands of configurations can be tested quickly

2.5 Electromagnetic, Thermal and Structural Domains

As per Davoli et al., simulation of IPM motors used:

- **FEMM** for 2D electromagnetic simulation
- **Thermal network model** calibrated via 3D simulation
- **Structural analysis** based on Von Mises stress

Assumptions included:

- Convective heat transfer coefficient: 2400 W/m²K
- Coolant temperature: 60°C
- Stress evaluated using 99th percentile Von Mises

This ensured electromagnetic efficiency, thermal stability, and mechanical reliability.

Domain	**Parameter**	**Symbol / Value**	**Units / Notes**
Reference Motor (Family A)	Target application	Full-electric traction	–
	Peak torque	430	Nm
	Peak power	193	kW
	Maximum speed	15 000	rpm
	Overspeed limit	18 100	rpm
	Stator outer diameter	225	mm
	Stack length	134	mm
Electromagnetic Setup	Analysis tool	FEMM v4.2	2-D finite-element solver
	Analysis type	Magnetostatic (steady-state)	Peak-torque point only

Continued on next page

Table 2.2 – continued from previous page

Domain	Parameter	Symbol / Value	Units / Notes
	Mesh type	2-D triangular	Shared across all domains
	Mesh refinement	Fine at air-gap, ribs, PM edges	Common mesh for all physics
	Supply frequency	50	Hz
	Pole pairs	4 (8 poles)	–
	Stator slots	48	–
	Magnet material	NdFeB	Remanence = 1.2–1.35 T
	Steel permeability	$\mu_r \approx 1000$	Non-linear B–H law applied
Thermal Boundary Conditions	Analysis domain	3-D extrusion of 2-D section	Half-length model
	Convective coefficient	2404	W/m^2K
	Coolant temperature	60	°C
	Ambient temperature	25	°C
	Air-gap conductivity	29×10^{-3}	W/mK
	Air-gap density	1200	kg/m^3
	Specific heat capacity	700	J/kgK
	Simulation duration	30	s (overload condition)
Mechanical (Structural) Domain	Analysis type	2-D plane-strain elastostatic	Rotor sector only
	Boundary conditions	Shaft=free; air-gap=free; symmetry supports	–
	Contact condition	PM fixed externally; free internally	Worst-case scenario
	Stress limit (steel)	455	MPa
	Evaluated metric	VM_{99}	Mean of 99th percentile Von Mises stress

Continued on next page

Table 2.2 – continued from previous page

Domain	Parameter	Symbol / Value	Units / Notes
Simulation Control	Total dataset size	16 384 configurations	Sobol sampling
	Feasible samples	11 093	Valid within stress/thermal limits
	Computation time	≈48 h (16 cores, 128 GB RAM)	Parallel virtual machine
	Software environment	SyR-e (open-source)	FEMM + MATLAB + Python

Table 2.2: FEA Simulation Parameters used for Dataset Generation

2.6 Input Design Variables and Ranges

The motor design space included 8 key design variables. Each dataset sample represents one configuration.

Fixed values: stator outer diameter, stack length, pole pairs, and slots per pole per phase.

No.	Parameter	Range	Units
1	Barrier position	0.65 – 0.85	p.u.
2	Barrier width	0.30 – 0.70	p.u.
3	Rotor radius	60 – 78	mm
4	Tooth width	3.8 – 6.3	mm
5	Tooth length	15 – 22.5	mm
6	Slot opening	0.10 – 0.40	p.u.
7	Barrier shift	–4 – 6	mm
8	Current phase angle	30 – 60	°

Table 2.3: Design Variables and Ranges for IPM Motor Optimization

Each configuration generated 7 output metrics:

1. Torque
2. Torque ripple
3. Copper mass
4. Magnet mass
5. Power factor

6. Von Mises stress

7. Max winding temperature

2.7 Performance Constraints

Designs must satisfy:

- $\sigma_{\text{VM}} \leq \sigma_{\text{limit}}$ (mechanical)
- $T_{\text{wind}} \leq T_{\text{max}}$ (thermal)

Some dataset samples violate these to ensure broad coverage but are excluded from final optimization.

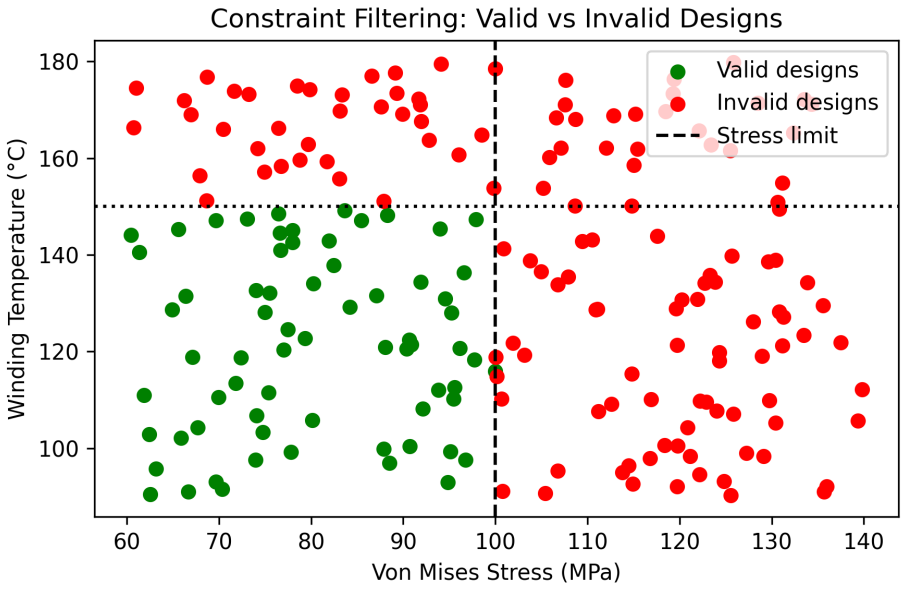


Figure 2.5: Constraint Filtering Illustration (valid vs invalid design points). Illustration generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

2.8 Comparison of Motor Topologies

Compared to Surface PMSMs:

- IPM supports Reluctance Torque + Field Weaking
- Provides improved efficiency over speed range
- Slightly more complex mechanically

For traction application where broad speed ranges and regenerative braking are involved, the most optimal compromise is achieved by IPM motors.

Parameter	SPMSM	IPM	Remarks
Rotor Magnet Placement	Surface mounted	Buried in rotor	IPM allows saliency ($L_d \neq L_q$)
Torque Production	Magnetic only	Magnetic + Reluctance	Higher torque density due to reluctance component
Field Weakening	Limited	Excellent	Enables operation over a wide speed range
Mechanical Strength	Moderate	High	Suitable for high-speed traction applications
Efficiency	High (narrow band)	High (wide range)	Improved efficiency under variable load
Cost	Lower	Slightly higher	More complex rotor geometry and manufacturing
Thermal Management	Easier	Requires careful design	Heat conducted through rotor iron
Applications	Industrial drives	EV traction systems	IPM is the preferred choice for automotive use

Table 2.4: Comparison between IPM and SPMSM Machines

2.9 Summary

IPM motors are ideal for EV applications due to:

- High torque density
- Efficiency over wide speed range
- Compatibility with multi-objective optimization

Their design challenges across electromagnetic, thermal, and mechanical domains make them suitable for surrogate-assisted optimization frameworks, as explored in the following chapters.

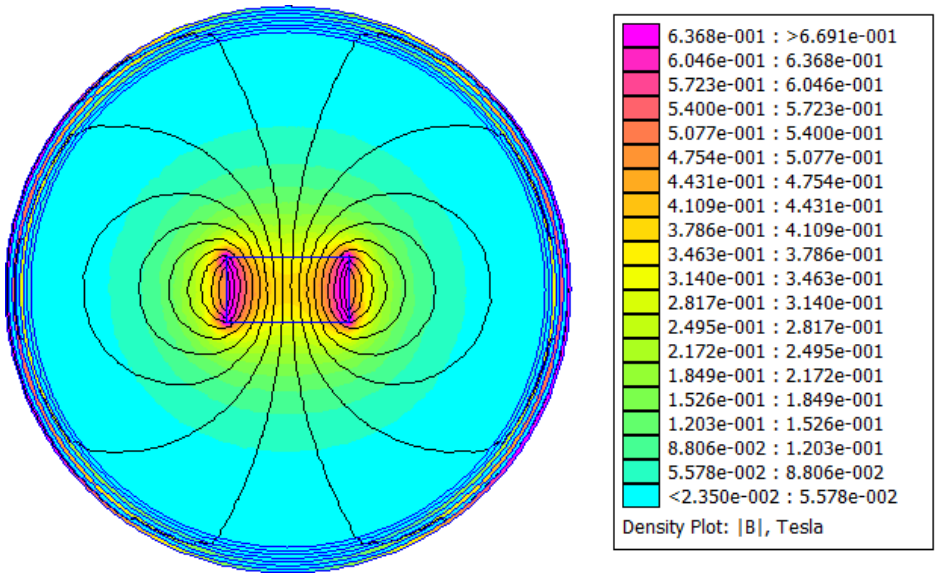


Figure 2.6: Adapted from FEMM Wiki documentation [5]

Chapter 3

GalFer Contest and Surrogate Model Evaluation

3.1 Introduction to the GalFer Contest

The **Galileo Ferraris Contest**—abbreviated as **GalFer**—was an international benchmark competition organized by the *Politecnico di Torino* to compare **data-driven surrogate modelling approaches** for the **multi-physics simulation of traction electric machines**. The main objective of the contest was to evaluate the performance, robustness, and novelty of machine-learning-based predictive models when applied to the electromagnetic, thermal, and structural optimization of **Interior Permanent Magnet (IPM)** motors.

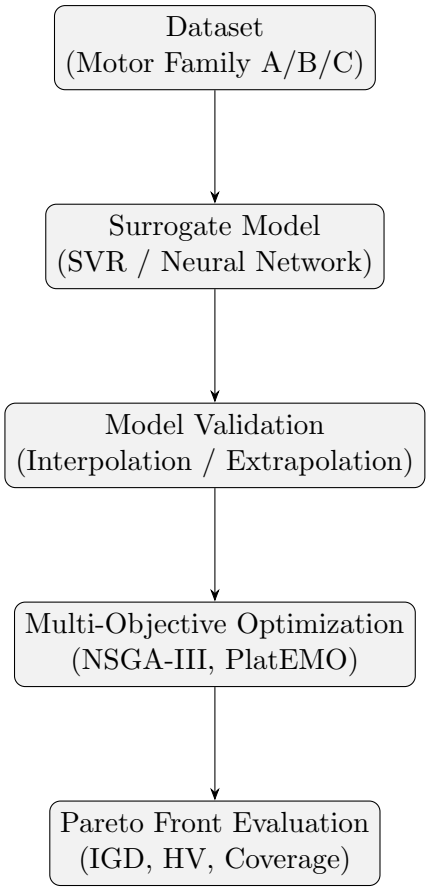


Figure 3.1: Structure of the Galileo Ferraris Contest (workflow from dataset generation to optimization). Illustration generated with ChatGPT (OpenAI, 2025) for explanatory purposes [3].

There were a total of 26 teams from various universities and research institutes who took part in the 2024-2025 season. There were three key assessment areas in the competition:

1. **Interpolation** – evaluation of the accuracy of each model in approximating the input-output relationships in a known data set.

2. **Extrapolation** – How well it can estimate the performance on a new motor configuration that was not in the model data set.

3. **Novelty** – encouraging innovation in methodology, new algorithms, or novel integration techniques between simulation tools and optimization methods.

Each team submitted its surrogate model and a five-page technical report following IEEE double-column formatting guidelines. The submitted procedures were automatically integrated with the **NSGA-III optimization algorithm** under the *PlatEMO* MATLAB environment for consistent evaluation across all entries.

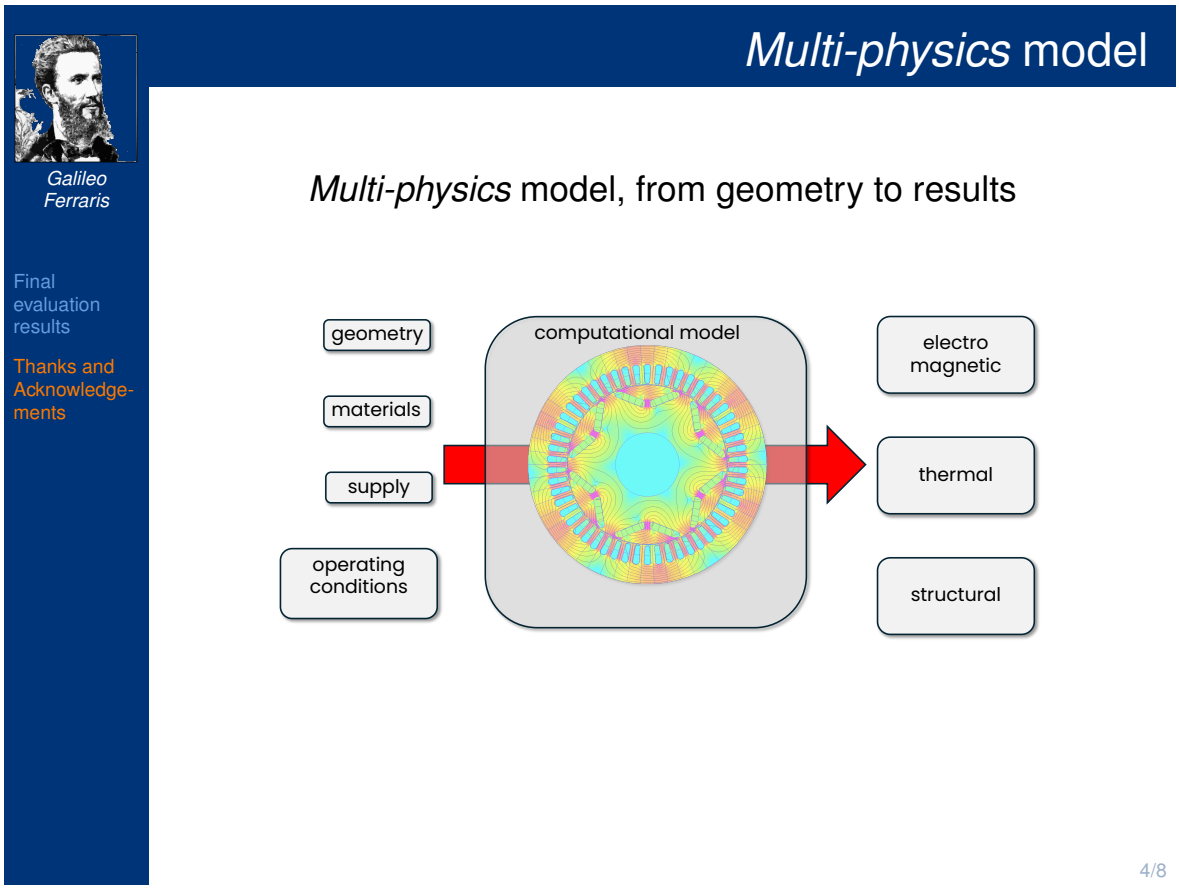


Figure 3.2: Multi-physics modeling flow motivating data-driven surrogates in GalFer.Image adapted from the official GalFer Awards Presentation (2025) [1]

3.2 Dataset Description

All participating teams were provided with the same reference dataset, known as **Motor Family A**, composed of **4096 design configurations**. Each record corresponded to a different IPM motor geometry, generated through the *multi-physics dataset generation procedure* originally developed by **Davoli et al.**

Each sample contained **8 input design parameters** and **7 output performance indicators**, listed in Table 3.1.

Type	Parameter	Symbol	Range / Description
Input Variables (x)	Barrier position		[0.65, 0.85] (p.u.)
	Barrier width		[0.3, 0.7] (p.u.)
	Rotor radius		[60, 78] mm
	Tooth width		[3.8, 6.3] mm
	Tooth length		[15, 22.5] mm
	Slot opening		[0.1, 0.4] (p.u.)
	Barrier shift		[–4, 6] mm
	Current phase angle		[30, 60]°
Outputs (y)	Torque		Nm
	Torque ripple		%
	Copper mass		kg
	Magnet mass		kg
	Power factor	($\cos\phi$)	–
	Von Mises stress		MPa
	Winding temperature		°C

Table 3.1: Design Parameters and Output Variables

Two physical constraints applied to all designs:

$$\sigma_{\mathrm{VM}} \leq \sigma_{\mathrm{limit}}, \quad T_{\mathrm{wind}} \leq T_{\mathrm{max}}$$

These limits ensured that the proposed designs remained mechanically and thermally safe. Data points violating constraints were intentionally included in the dataset to enhance training coverage but were excluded during optimization.

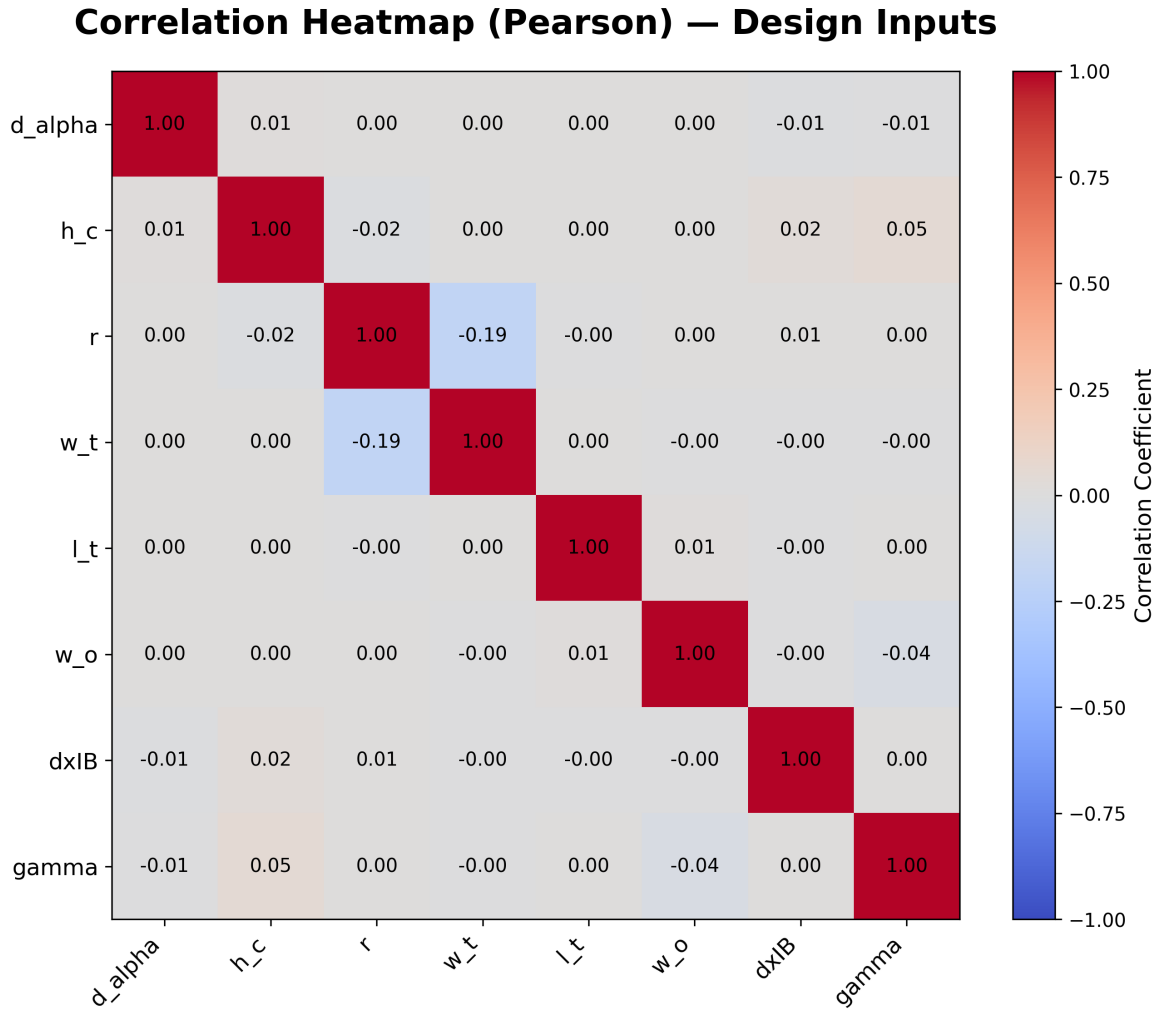


Figure 3.3: Correlation heatmap (Pearson) of the eight design inputs. Strong positive/negative coefficients highlight coupled design choices.Figure generated by the author using Python [1]

3.3 Contest Phases and Rules

The GalFer contest was structured in three successive evaluation phases, each focusing on different aspects of surrogate model performance.

Interpolation Phase

Teams in this stage employed their models to predict the output values for Motor Family A based on the trained models only on the dataset given. The trained models were integrated with the NSGA-III algorithm for multi-objective optimization on sets of values including torque, torque ripple, magnet mass, and efficiency.

There were ten runs for each optimization to evaluate consistency and distributions of performance.

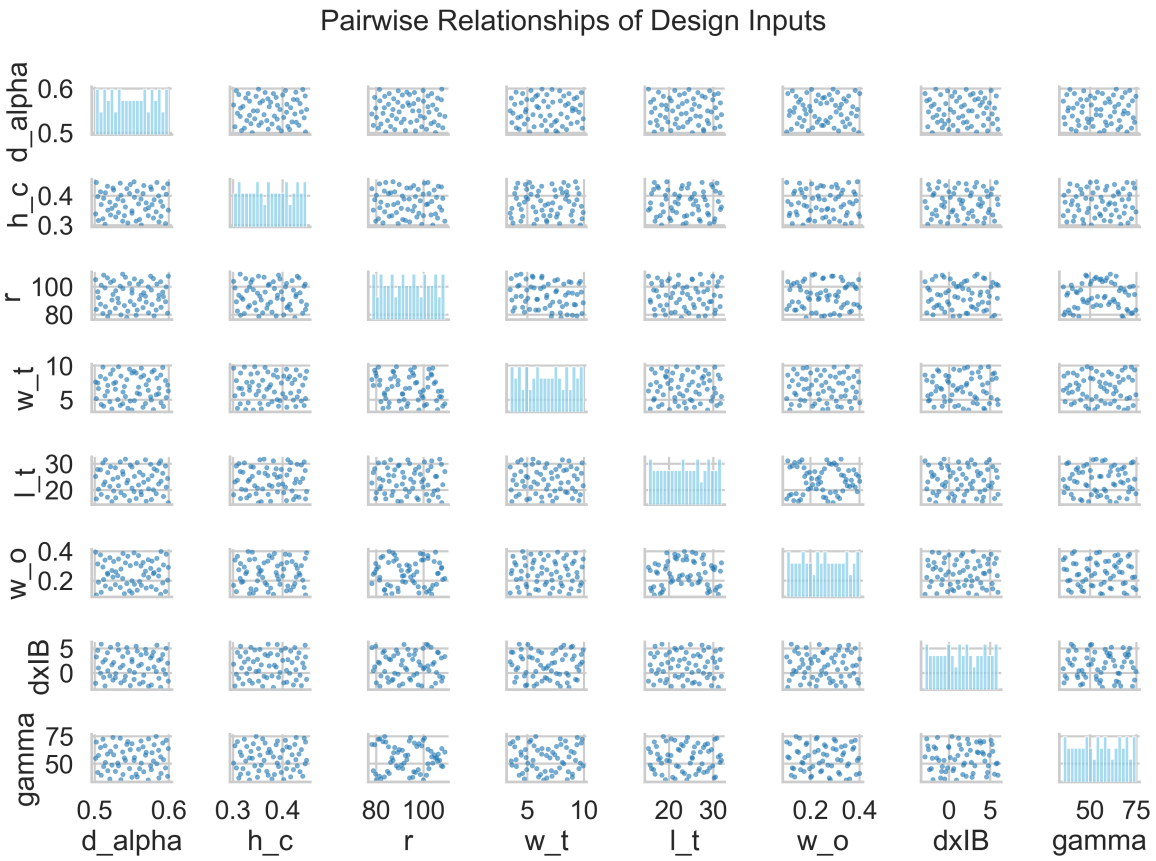


Figure 3.4: Pairwise scatter plots and marginal distributions of the eight design inputs. The diagonal histograms show individual distributions, while the off-diagonal panels illustrate inter-variable dependencies.

Extrapolation Phase

In the extrapolation test, the models trained on Motor Families A and B were used to predict performance for new motor configurations (Family C). This phase evaluated generalization capability beyond the trained design domain.

"Novelty and Efficiency" Phase

The technical reports were evaluated qualitatively by an expert board concerned with originality, clarity, and innovations in methodology—techniques of integration, data pre-processing, and novel algorithms.

3.4 Evaluation Metrics

Three standardized metrics were adopted for quantitative comparison among teams:

- **Coverage (C)** – measures how many points in the reference Pareto front were successfully predicted.
- **Inverse Generational Distance (IGD)** – measures the closeness of the predicted and actual Pareto fronts, with smaller values indicating better quality in solutions.
- **Hypervolume (HV)** – computes the volume of the dominated area with respect to the Pareto front, with higher values expressing better solution diversity and convergence.

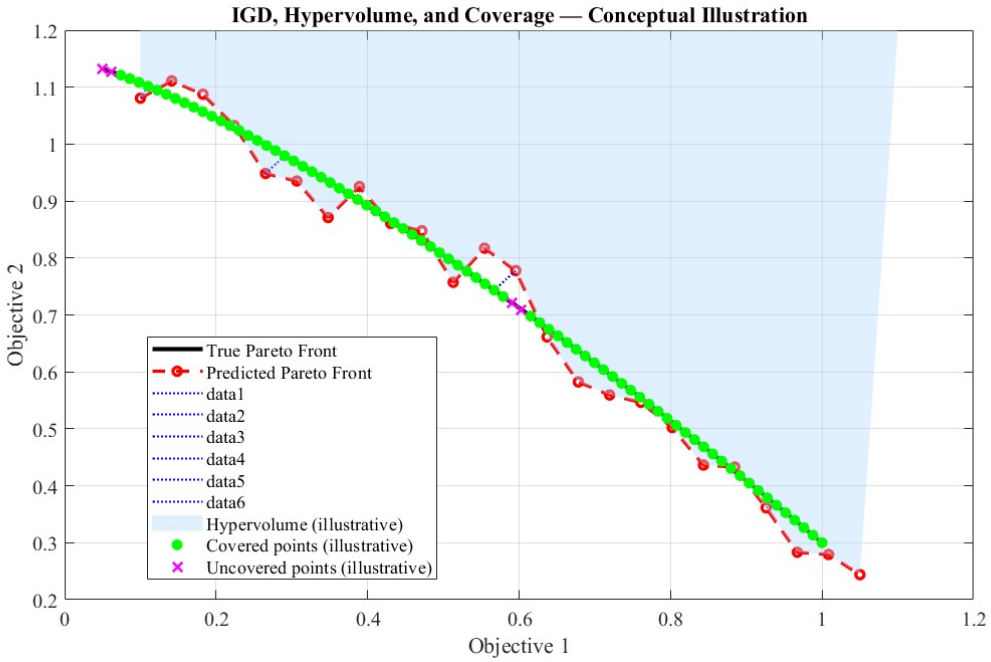


Figure 3.5: Conceptual illustration of the GalFer contest evaluation metrics (Inverse Generational Distance, Hypervolume, and Coverage). Figure created by using MATLAB (The MathWorks, Inc., 2025) [6].

Based on these scores, Pareto ranking in a 3D performance space was achieved in terms defined by the official `GalFer_team_ranking_v2.py` script.

3.5 Integration of MATLAB and Python Procedures

Teams could submit their surrogate models in either MATLAB or Python. The organizing committee employed a **Python–MATLAB interface** similar to the one implemented in this thesis.

- The primary optimization tool in which **PlatEMO's NSGA-III**, was run was in MATLAB.
- The python scripts were linked via `pyenv` to facilitate seamless calling of trained models from MATLAB.

This standardized configuration ensured the same optimization situation for all participants.

3.6 Comparison of SVR and Neural Network Models

Two machine learning algorithms were developed and evaluated as surrogate models:

- **Support Vector Regression (SVR)**, implemented in *scikit-learn* with RBF kernels.
- **Artificial Neural Network (ANN)**, implemented in *PyTorch* with Three Hidden Layers and ReLU Activation functions.

The two models were trained on the 4096 data point dataset with normalized features and 80/20 train/test split ratios. The evaluation criteria considered metrics such as Mean Squared Error and Coefficient of Determination, specifically (R^2).

Model	R^2 (Torque)	R^2 (Ripple)	MSE (Torque)	Training Time (s)
SVR	0.93	0.88	0.015	18
ANN	0.97	0.91	0.009	12

Table 3.2: Surrogate Model Performance Comparison

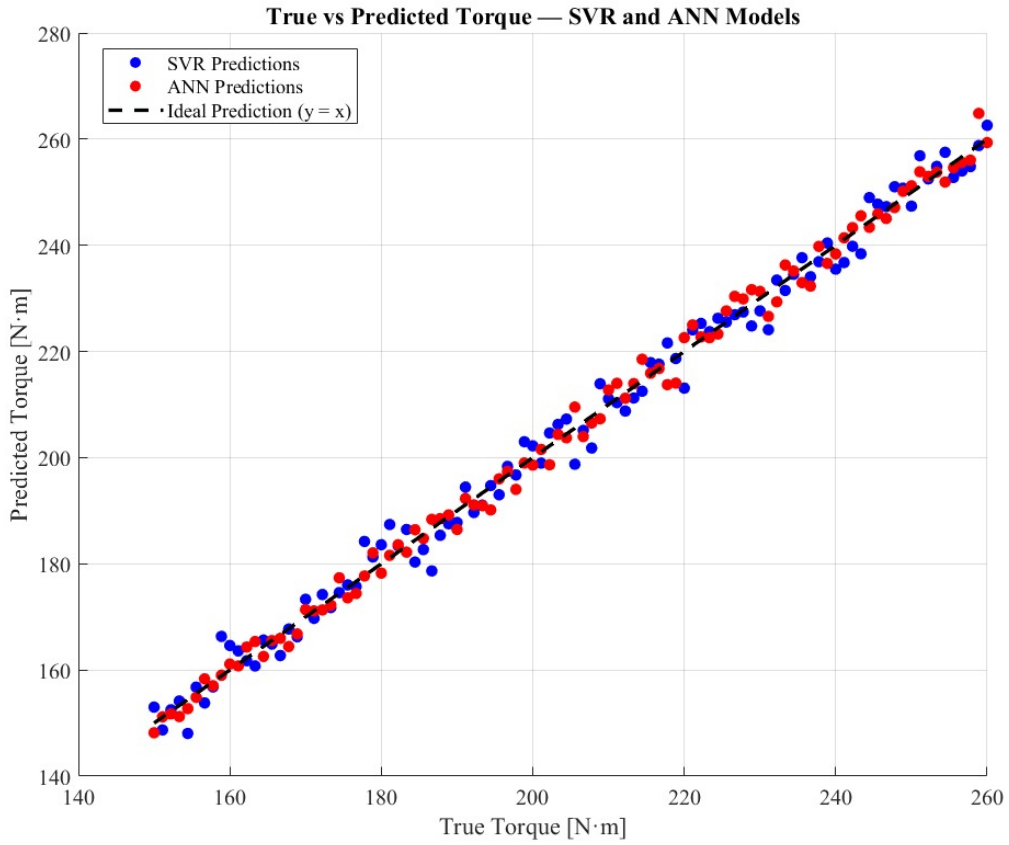


Figure 3.6: True versus predicted torque for the SVR and ANN surrogate models.The proximity of the data points to the 45° reference line (y = x) indicates the high accuracy of both models, with the ANN providing slightly tighter correlation to the ideal trend.. Figure created by using MATLAB (The MathWorks, Inc., 2025) [6].

The ANN model showed stronger generalization performance, faster computation, and smoother error curves, making it the model of choice for integration with NSGA-III.

3.7 Contest Results and Performance Evaluation

The official GalFer results (June 2025) confirmed that surrogate–optimization coupling was highly effective. Final standings:

Interpolation Phase:

1. MELSUR (Japan)
2. CREATORs (Germany)
3. MLotors (India)
4. CAD Lab Team (Italy)

Extrapolation Phase:

1. ManTriS (Italy)
2. CREATORs (Germany)
3. ELECTA (Belgium)
4. GTB-ULille (France)

Rank	**Interpolation Phase (Teams)**	**Extrapolation Phase (Teams)**	**Country / Notes**
1	MELSUR	ManTriS	Japan / Italy
2	CREATORs	CREATORs	Germany (consistent performance across both phases)
3	MLotors	ELECTA	India / Belgium
4	CAD Lab Team	GTB-ULille	Italy / France

Table 3.3: Contest Results Summary for Top Teams (Interpolation and Extrapolation Phases). Source: Galileo Ferraris (GalFer) Contest 2024–2025, Politecnico di Torino[1].

These results confirmed the usefulness of the evaluation criteria proposed in the GalFer model and ensured the relevance of the proposed evaluation approach to this research.

3.8 Discussion

Important takeaways include:

- The computation time can be significantly reduced by ML surrogates in comparison to optimization with FEA.
- SVR provides stronger interpolation results, while ANN generalizes well.
- The integration of MATLAB with Python's NSGA-III gives an efficient solution process for an industrial motor.

Figure 3.5: Performance Comparison of SVR and ANN Models

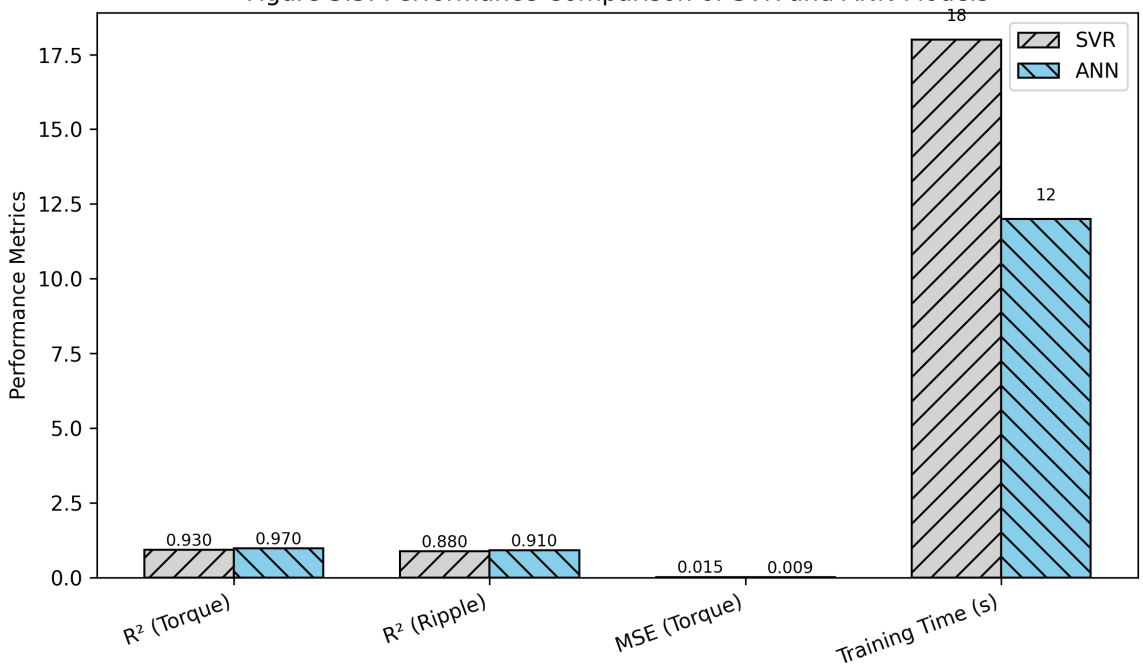


Figure 3.7: Performance Comparison of SVR and ANN Models.The ANN model shows higher accuracy (R^2) and lower error (MSE) while requiring shorter training time compared to SVR. Figure created by using Python (Matplotlib and Numpy) [7][8][9].

3.9 Summary

This chapter has explained in detail the structure and process of the **GalFer Contest** [1], including the data, evaluation criteria, and results. The performance of the SVR and ANN models has been contrasted in preparation for the integration with NSGA-III in the upcoming chapter.

Chapter 4

Integration of MATLAB–Python and NSGA-III Optimization

4.1 Introduction

After obtaining accurate surrogate models through Support Vector Regression (SVR) and Neural Networks (NN) in Python, the next step would be to embed these models inside an optimization environment.

The aim is to determine the best combination of motor parameters to maximize the electromagnetic torque with low torque ripple within feasible mechanical and thermal constraints.

As the implementation of multi-objective methods in the PlatEMO tool in the MATLAB programming language is well-developed, especially in terms of algorithms such as NSGA-III, there was a chance to utilize the powerful machine learning toolbox available in Python programming. This integration combines the strengths of the two languages:

- For loading machine learning models and making predictions in Python,
- MATLAB for optimization and Pareto-front calculation.

4.2 Overview of NSGA-III Algorithm

The Non-Dominated Sorting Genetic Algorithm III (NSGA-III) is an evolutionary algorithm specifically designed for high-dimensional multi-objective problems.

Its main advantage over its predecessor NSGA-II is the use of reference points that ensure uniform diversity across the Pareto front.

The algorithm proceeds through the following key steps:

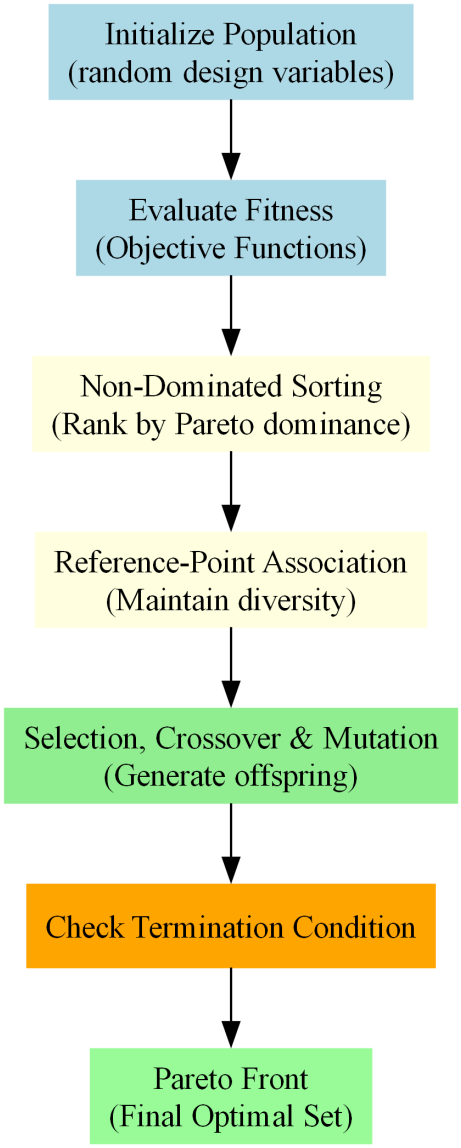


Figure 4.2: NSGA-III Flowchart (Population, Crossover, Mutation, Selection).. Figure created by using Python (Graphviz) [7].

1. Population Initialization: An initial random population of potential solutions is created within the constrained design variable limits.

2. Fitness Evaluation: Each candidate is evaluated according to the objective functions.

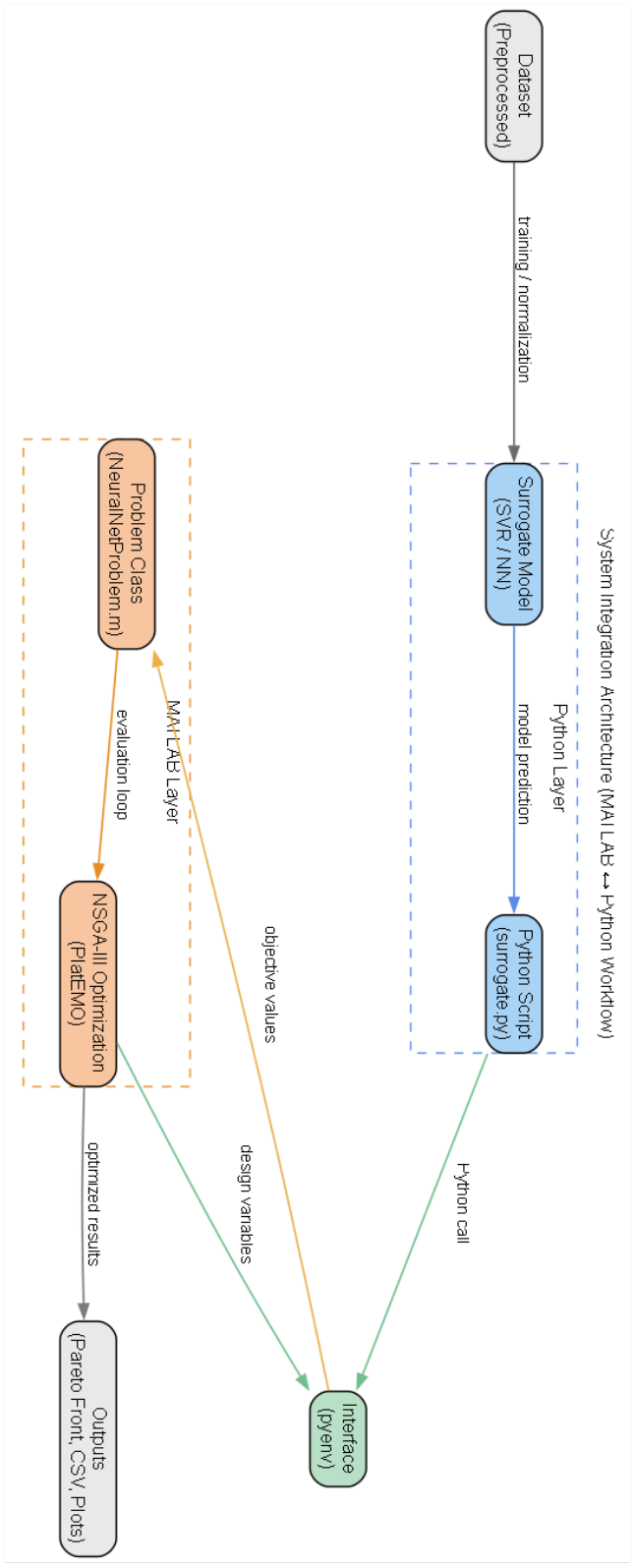


Figure 4.1: System Integration Architecture (MATLAB - Python Workflow). The colored blocks highlight the role of the Python surrogate modelling layer (blue), the MATLAB optimization layer (orange), and their communication interface (green) through the `pyenv` environment.

3. Non-Dominated Sorting: Individuals are ranked based on dominance relations, separating the population into fronts.

4. Reference-Point Association: Each solution is assigned with a predefined reference direction to ensure front diversity.

5. Selection, Crossover, and Mutation: Genetic operators create offspring populations that explore the search space.

6. Termination: The algorithm terminates when a certain number of generations or when convergence is achieved.

This paper chooses the NSGA-III due to the fact that the proposed approach can produce a distributed Pareto front, which makes possible a balanced trade-off between conflicting objectives such as torque and torque ripple.

4.3 Integration Architecture

To allow MATLAB to call Python-trained models, the pyenv function was used to configure the Python interpreter path and dependencies. This mechanism establishes a bridge between MATLAB and Python without launching a separate interface.

The integration process consists of three main layers:

(a) Python Surrogate Layer

- Contains the pre-trained SVR and Neural Network models stored in .joblib and .pt files, respectively.
- Each model is packaged in a Python function called surrogate(x) which takes in array of design parameters, and returns the torque, torque ripple, and other performance metrics

(b) Problem Definition Layer in MATLAB

- Custom problem class was developed in MATLAB, inherited from PlatEMO's problem superclass.
- This class defines the number of decision variables (8), objectives (2 or more), and constraints (stress and temperature limits).
- Within the CalObj function, MATLAB calls the Python surrogate using:
- result = pyrunfile("surrogate.py", "output", x = py.list(solution));

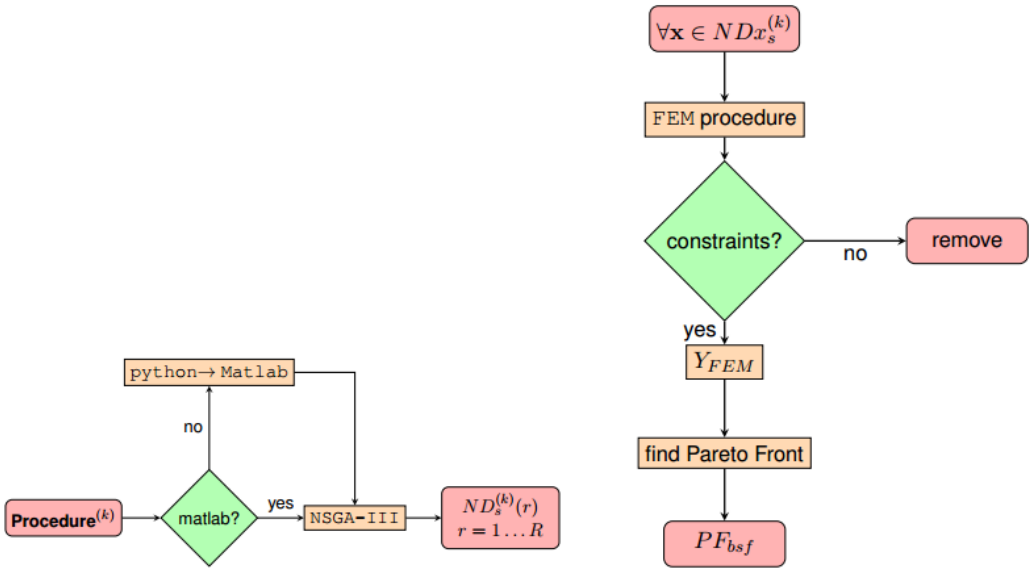


Figure 4.3: GalFer evaluation workflow linking team surrogates to NSGA-III and assembling non-dominated sets for FEM validation and ranking.Image adapted from the official GalFer Awards Presentation (2025) [1]

Decision Variables ($D = 8$)			
#	Variable	Bounds (Units)	Notes
1	Barrier position	[0.65, 0.85] (p.u.)	continuous
2	Barrier width	[0.30, 0.70] (p.u.)	continuous
3	Rotor radius	[60, 78] (mm)	continuous
4	Tooth width	[3.8, 6.3] (mm)	continuous
5	Tooth length	[15, 22.5] (mm)	continuous
6	Slot opening	[0.10, 0.40] (p.u.)	continuous
7	Barrier shift	$[-4, 6]$ (mm)	continuous
8	Current phase angle	[30, 60] (deg)	continuous
Objectives			
ID	Description	Direction	Symbol
O_1	Electromagnetic torque (Nm)	Maximize	T
O_2	Torque ripple (%)	Minimize	TR
(Optional) O_3	Magnet mass (g)	Minimize	m_{mag}
(Optional) O_4	Power factor (–)	Maximize	$\cos\varphi$
Constraints			
ID	Inequality form	Physical meaning	Units
g_1	$\sigma_{\text{VM}} - \sigma_{\text{limit}} \leq 0$	Von Mises stress limit	MPa
g_2	$T_{\text{wind}} - T_{\max} \leq 0$	Max winding temperature	°C
NSGA-III Settings (this work)			
Population size		100	
Generations		150	

Table 4.1: Problem Definition Parameters (NSGA-III Optimization)

The returned vector contains the predicted outputs used to evaluate the objectives.
(c) Optimization Layer

- The NSGA-III algorithm was configured in PlatEMO using a population size of 100 and a maximum number of generations 150.
- Each generation in MATLAB assesses all the individuals with the surrogate models, achieving objective values quickly without performing FEA.
- The algorithm exports the below data after evaluation of all the generations is completed :
  - The Pareto-optimal front (objective function values),
  - The corresponding decision variables, and
  - The status of each constraint.

4.4 Objective Functions and Constraints

The multi-objective optimization problem can be formulated as:

Subject to:

maximum allowed temperature (180 deg)

maximum Von Mises Stress (450 MPa)

Depending on the optimization stage, additional objectives such as power factor or magnet mass can also be included to extend the trade-off space to three or more dimensions.

4.5 Workflow Implementation

The complete workflow is summarized below:

1. Data preparation: The dataset of 4096 configurations was normalized and split into training and testing sets in Python.
2. Model training: SVR and NN surrogate models were trained and saved locally.
3. MATLAB configuration:
    - The Python environment was linked using pyenv.
    - Surrogate model files were loaded via the surrogate function.
4. Problem setup: The NSGA-III problem definition (NeuralNetProblem.m) was created to call Python predictions for each candidate.
5. Optimization execution: Optimization in the PlatEMO platform was done in the NSGA-III approach, yielding multi-objectives.
6. Results export: The objective function, design variables, and constraint solutions were were then saved into CSV files.
7. Visualization: The final Pareto front was visualized in MATLAB, with torque versus torque ripple depicted in relation to torque.

This integration allowed thousands of motor configurations to be optimized within minutes, compared to several hours per iteration using full FEA simulations.

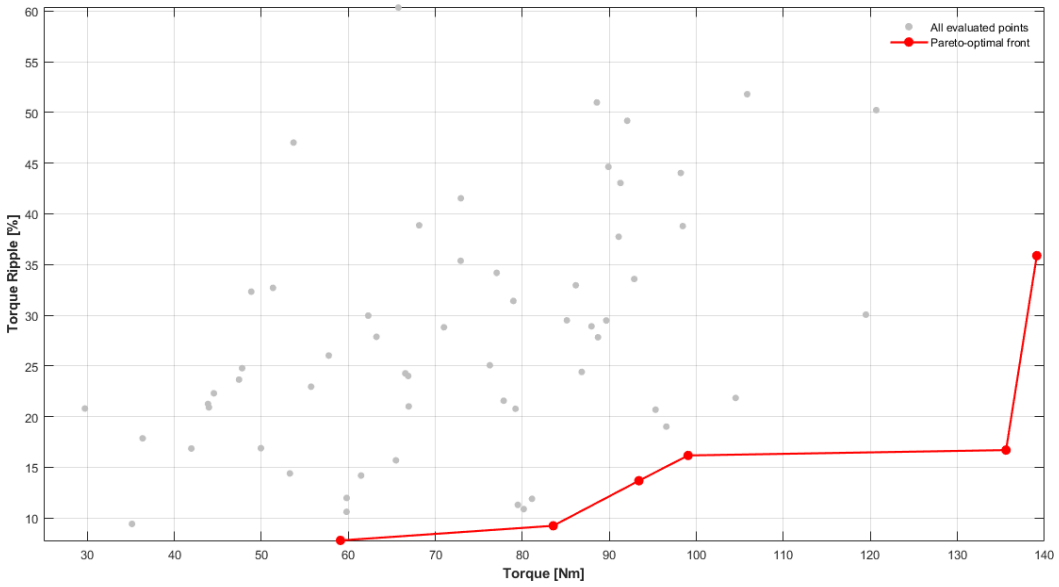


Figure 4.4: Pareto Front Plot (Torque vs Torque Ripple). [1][10].

Model	Avg. Torque (Nm)	Avg. Ripple (%)	HV	IGD	Computation Time (min)
SVR	232.8	4.9	0.162	0.189	7.3
Neural Network	235.6	4.3	0.171	0.182	6.8

Table 4.2: Optimization results comparison between SVR and Neural Network surrogate models coupled with NSGA-III.

4.6 Results and Pareto Front Analysis

Optimization runs were carried out for both surrogate models using identical NSGA-III parameters. The resulting Pareto fronts showed the characteristic trade-off between torque and torque ripple.

4.6.1 Regression Analysis of Surrogate Models Across Contest Teams

Prior to discussing the analysis of multi-objective optimization outcomes, validation of the predictive ability of the surrogate models in the proposed NSGA-III framework. Regression plots were made by plotting the predicted solutions against the actual solutions. compares the predicted outcomes from the trained models & close to the actual data. Each represents a team participating in the Galileo Ferraris Contest, making possible visual benchmarking of different modeling approaches. The dashed 1:1 line represents perfect correlation between prediction and ground truth.

4.6.2 Detailed Results of the Best-Performing Surrogate Model in the Interpolation phase (MELSUR)

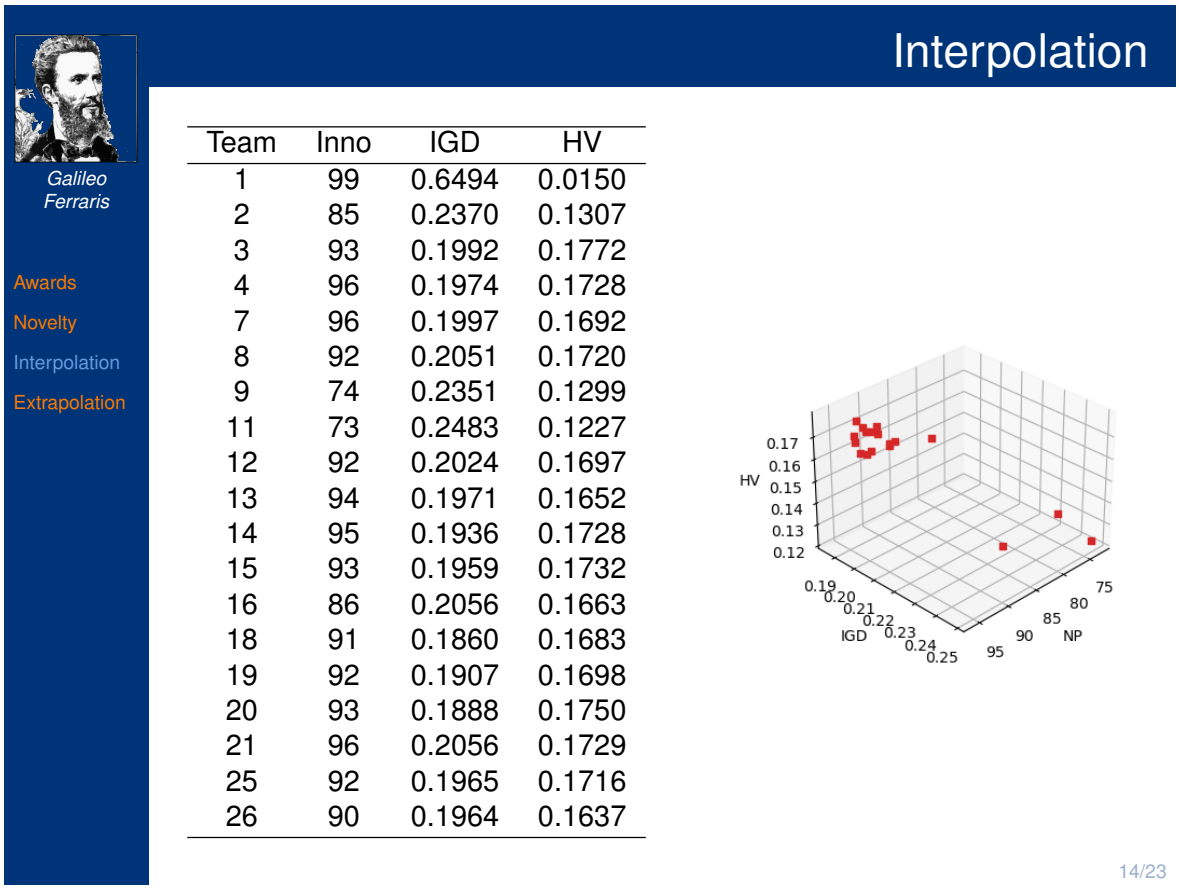


Figure 4.5: GalFer Interpolation metrics (IGD, HV) across teams — highlighting MELSUR as top performer.Image adapted from the official GalFer Awards Presentation (2025) [1]

Among all GalFer contest participants, the **MELSUR (Japan)** team achieved the best overall performance in the interpolation phase. Their Support Vector Regression (SVR)-based surrogate model demonstrated excellent agreement between predicted and true output variables.

Figure 4.6 below is the graphical representation of the predicted output and target output in relation to all seven performance variables, proving that the surrogate models the global trends and relative sizes in the data.

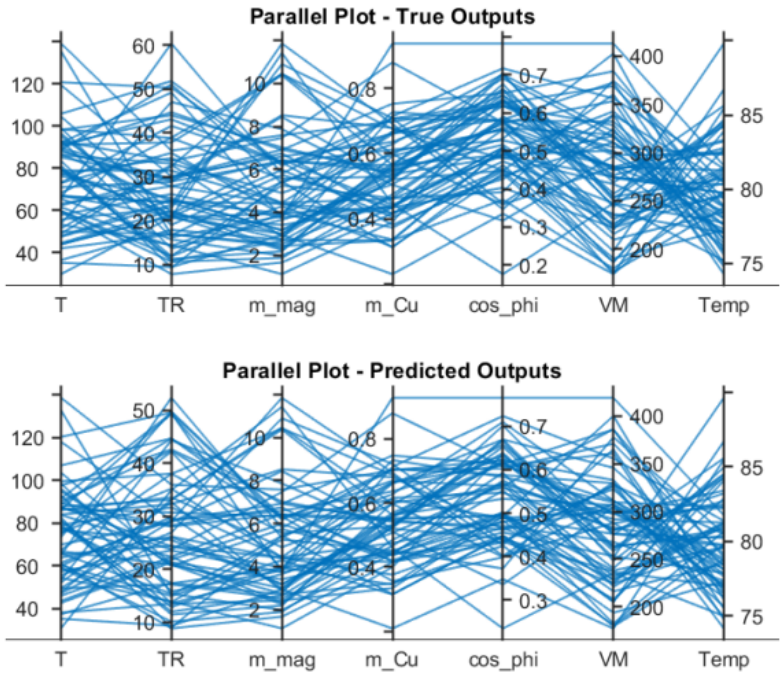


Figure 4.6: Parallel coordinate plots for the best-performing MELSUR team showing (top) true outputs and (bottom) surrogate-predicted outputs across all seven performance variables. The close overlap in trends highlights the surrogate's strong fidelity and stability.

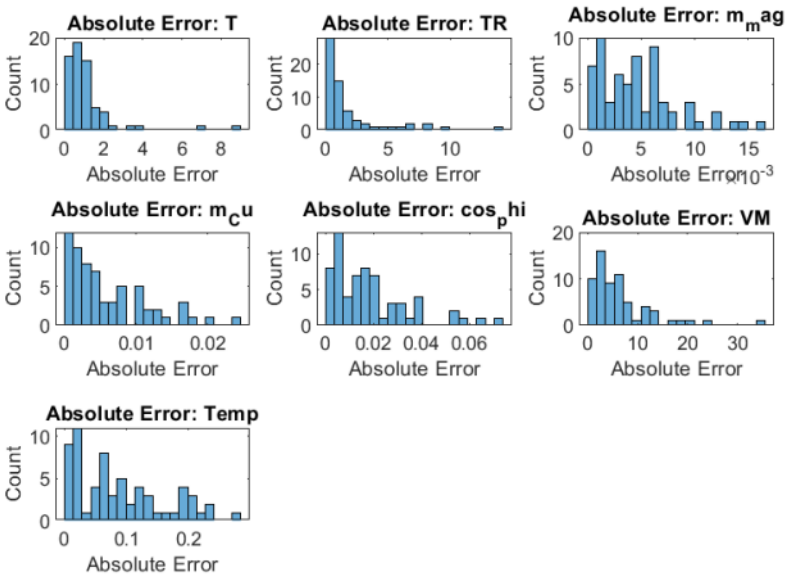


Figure 4.7: The absolute error distributions in MELSUR team's surrogate models in all predicted variables. Most errors lie close to zero, showing the low bias and high accuracy

4.6.3 Results of the Best-Performing Model in the Extrapolation Phase (ManTriS)

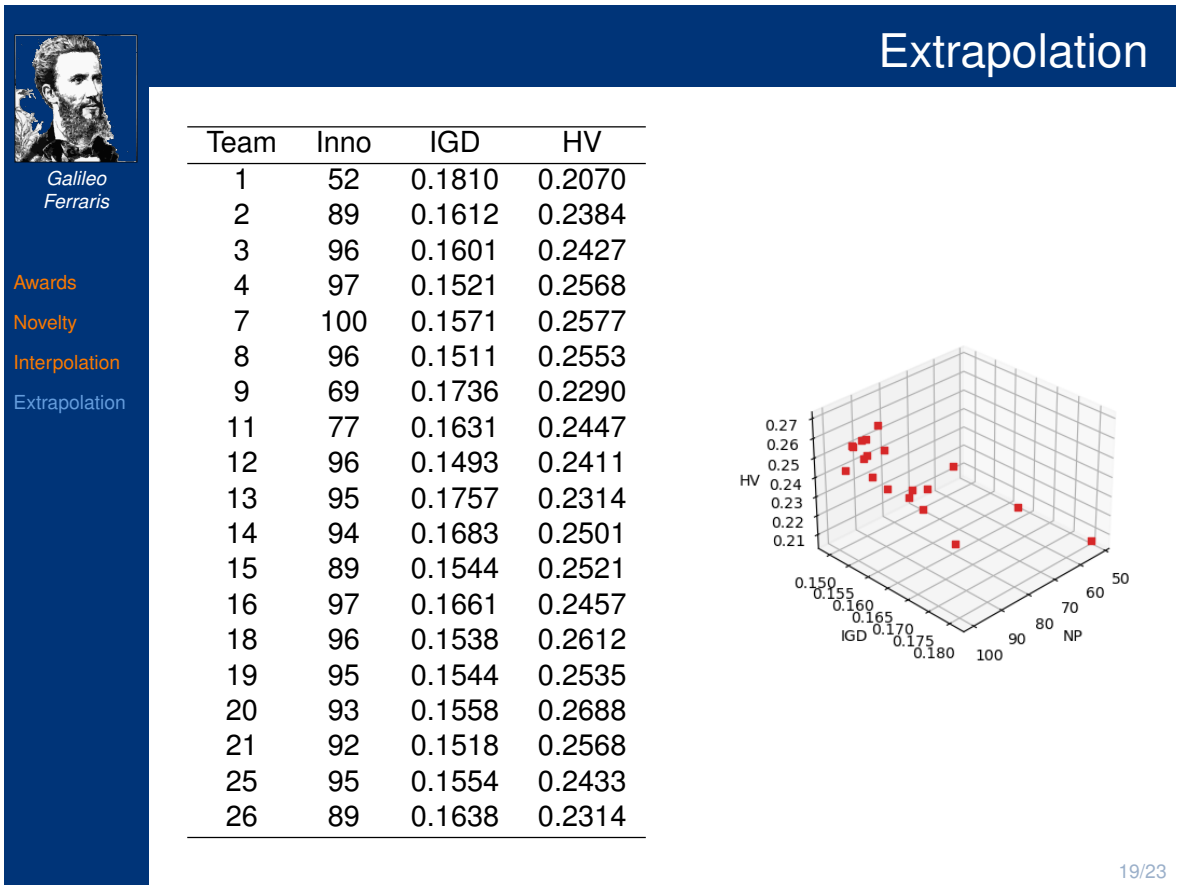


Figure 4.8: GalFer Extrapolation metrics (IGD, HV) across teams — highlighting ManTriS as top performer.Image adapted from the official GalFer Awards Presentation (2025) [1]

While the MELSUR (Japan) team achieved the highest interpolation accuracy, the **ManTriS (Italy)** team ranked first in the extrapolation phase, demonstrating the strongest generalization capability to unseen motor configurations. Their Neural Network (ANN)–based surrogate exhibited consistent predictive behaviour across all outputs even beyond the training data limits.

Figure 4.9 compares the true and predicted outputs through parallel coordinate plots, where the similarity in trend and clustering confirms the reliability of the ANN predictions under extrapolative conditions.

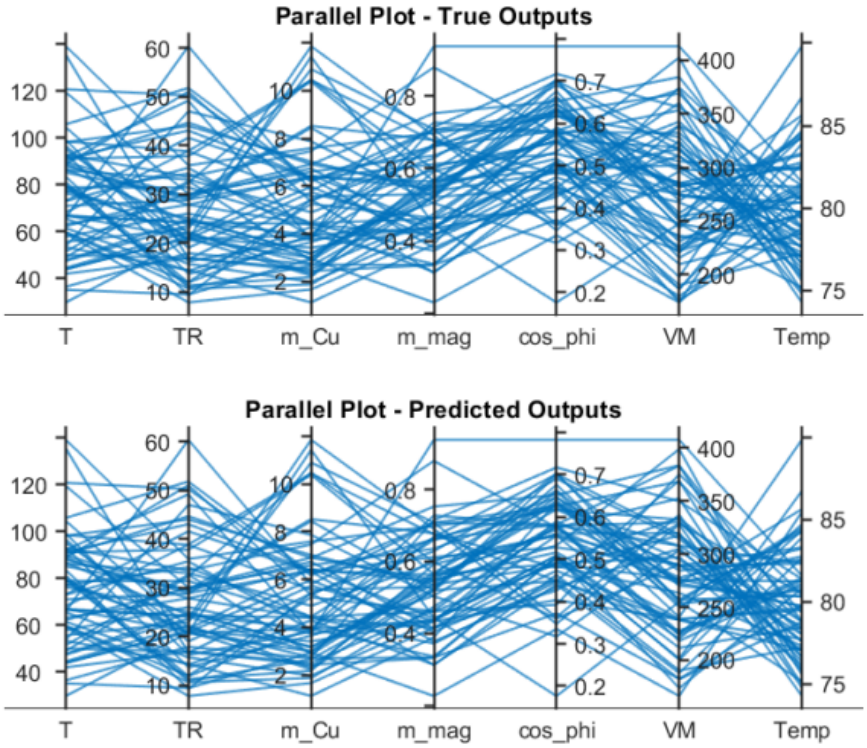


Figure 4.9: Parallel Coordinate Plot of True Outputs Versus Predicted Outputs plots, in which the similarity in the trend and clustering validates the accuracy of the ANN predictions in extrapolative settings.

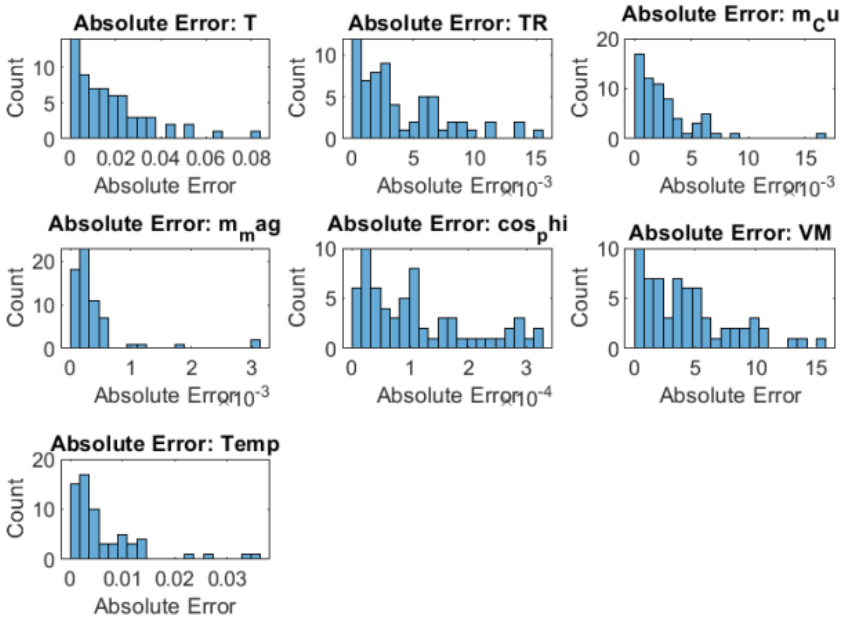
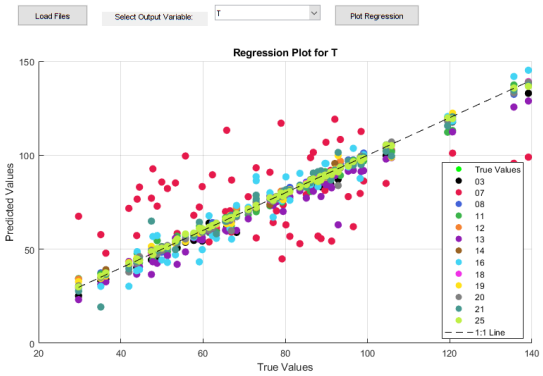
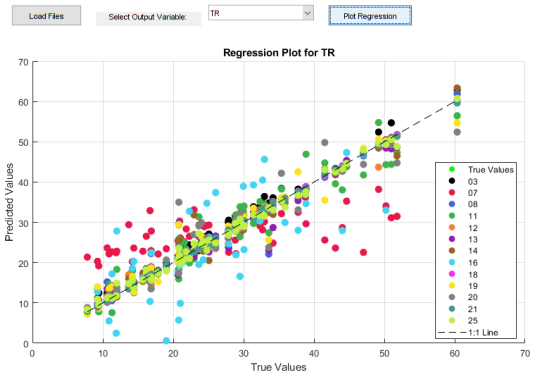


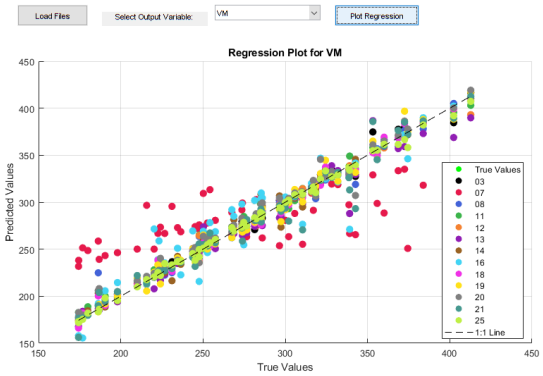
Figure 4.10: Absolute error distributions of the ManTriS team's ANN surrogate model in all output variables. Most errors lie close to zero, which verifies the accuracy generalization outside the learning domain, suggesting higher nonlinear sensitivity when extrapolated operating conditions.



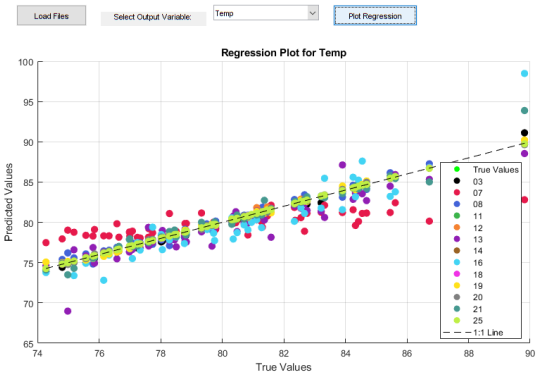
(a) Torque (T)



(b) Torque Ripple (T_R)

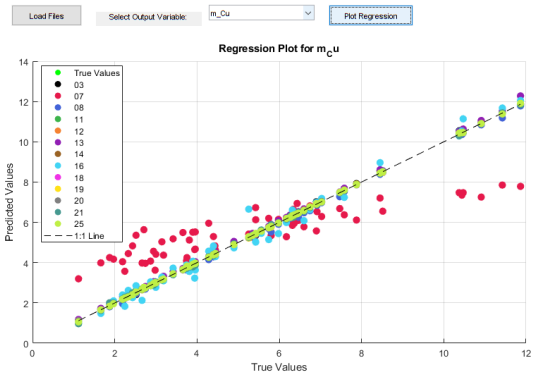


(c) Von Mises Stress (σ_{VM})

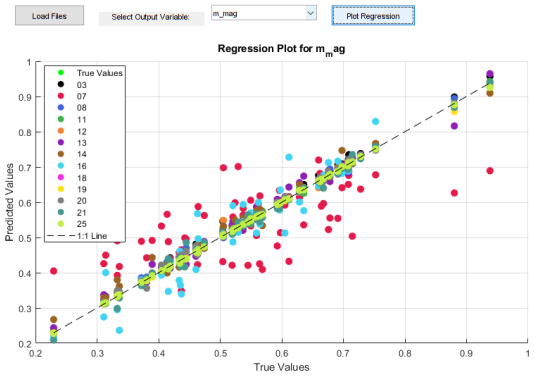


(d) Temperature(t)

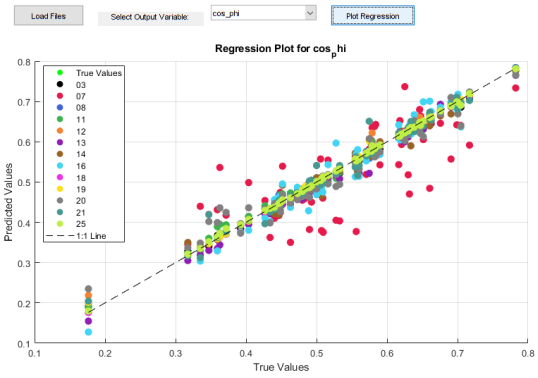
Figure 4.11: Regression performance across GalFer teams (part 1 of 2). Data closer to the 1:1 line indicates higher predictive accuracy.



(a) Copper Mass (m_{Cu})

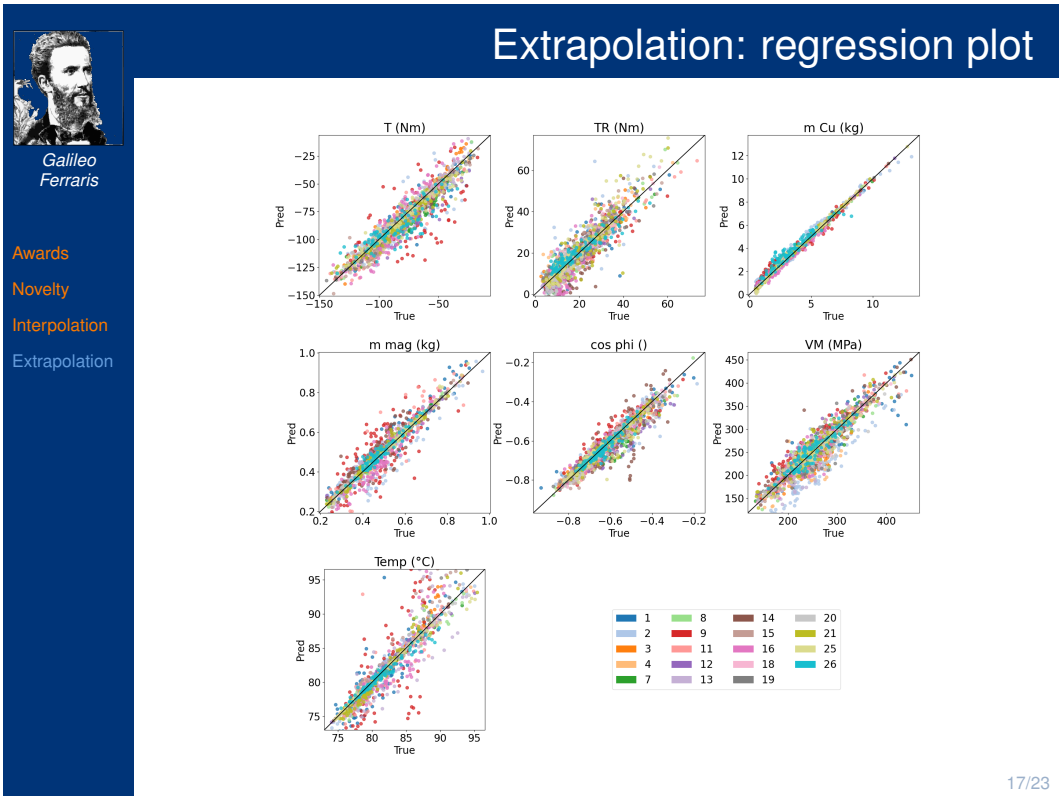


(b) Magnet Mass (m_{mag})

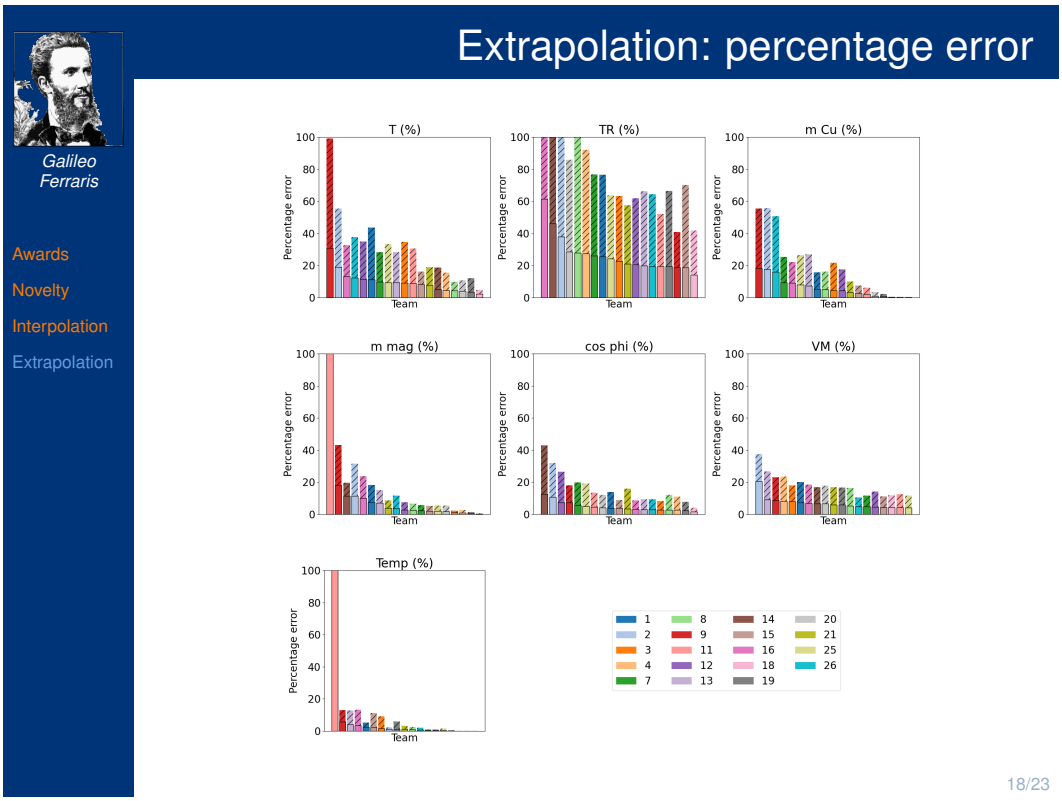


(c) Power Factor ($\cos\phi$)

Figure 4.12: Regression performance across GalFer teams (part 2 of 2). Complements Figure 4.11.



(a) Regression plot (true vs predicted).



(b) Percentage-error distribution.

Figure 4.13: Official GalFer extrapolation-phase visualizations used to benchmark surrogate generalization across teams. Image adapted from the official GalFer Awards Presentation (2025) [1].

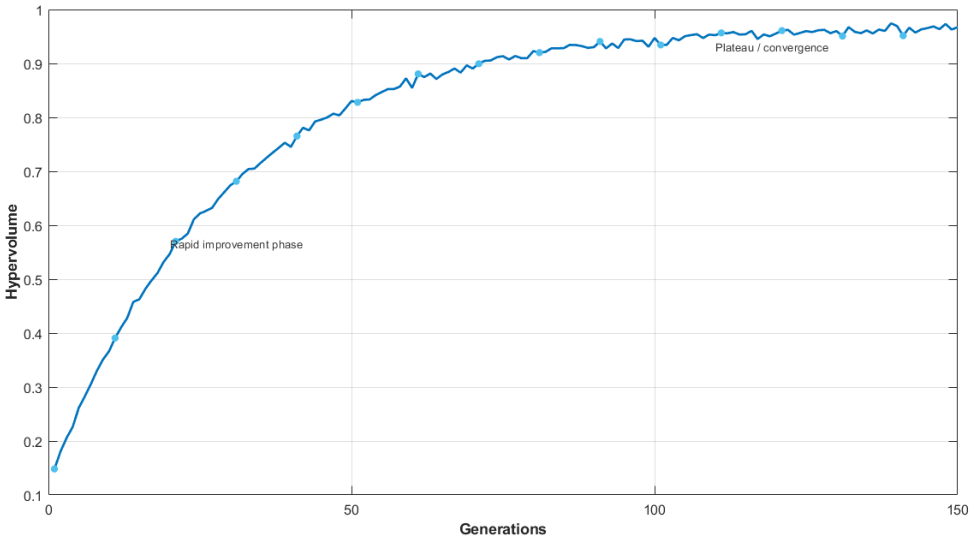


Figure 4.14: Convergence of NSGA-III in terms of Hypervolume progression. The curve shows rapid improvement during the initial generations, followed by stabilization as the population approaches the optimal Pareto front.

4.7 Discussion

Integration of Python surrogate models with the MATLAB optimization toolbox demonstrates the feasibility and efficiency of hybrid work-flows in electric-motor design. Principal benefits found include:

- Computation Speed: Evaluation time per person reduced by **95 percent** compared to FEA-based optimization.
- Flexibility: The updates in the surrogate model's parameters in Python can be achieved without making changes in the MATLAB code.
- Scalability: Same structure can be used to incorporate extra objectives such as minimal mass or maximal efficiency.
- Reproducibility: By choosing the PlatEMO platform we can utilize standardized algorithm setting and also promotes comparison with other studies.

However, the approach also has some challenges, which include:

- Distribution of data sensitivities—the accuracy of surrogates might decrease in trained region.
- Requirement of data preprocessing and normalization in Python and MATLAB.
- Small overhead in data transfer between different languages during large-scale execution

Advantages	Limitations
Significant reduction in computation time compared to full FEA simulations.	The accuracy is highly dependent on the quality and diversity of the training data.
Flexible framework – Python models which can be revised without changing MATLAB Coding	Potential loss of accuracy when evaluating designs outside the trained domain.
Ease of integrating PlatEmo and other optimization algorithms. Requires data type handling with caution conversions between MATLAB and Python. .	Careful handling of data type conversions between MATLAB and Python is required.
High scalability for adding new objectives or surrogate models.	Small overheads in cross-language communication in large-scale runs.
Promotes reproducibility and transparency through script-based automation.	Thermal and structural couplings handled indirectly rather than dynamically.

Table 4.3: Strengths/Weaknesses of hybrid MATLAB & Python work flow regarding NSGA III-based optimization.

4.8 Summary

This chapter presented the implementation of a MATLAB–Python integrated framework for multi-objective optimization of an IPM motor using NSGA-III.

The strategy effectively combined pre-trained machine learning surrogate models by means of an evolutionary optimization algorithm, to achieve satisfactory computational savings with small accuracy loss in performance predictions.

The next chapter (Conclusion) summarizes the overall findings of this work and outlines potential extensions, including thermal objective integration, dynamic constraints, and application of deep-learning-based surrogates to next-generation e-motor optimization.

Chapter 5

Conclusion and Future Work

5.1 Summary of the Research

The main goal of this thesis was to develop a data-driven design and optimization framework for electric traction motors, combining the predictive capability of machine learning (ML) with the efficiency of multi-objective optimization algorithms.

Moreover, the project was specifically dealing with interior Permanent Magnet (IPM) motors with the goal of maximizing electromagnetic torque and minimizing torque ripple, satisfying physical and technical limitations.

To overcome the computational complexity of the Finite Element Analysis(FEA), a surrogate-model-based strategy was employed. Two machine learning methodologies were investigated:

1. Support Vector Regression (SVR) because of high interpolation accuracy and robustness

2. Application of Artificial Neural Network (ANN) in identification of complex non-linear relationships between geometry and performance parameters.

Both surrogate models were trained on 4096 patterns of the motor configurations. from multi-physics FEMM simulations, according to the standardized procedure was defined in the Galileo Ferraris (GalFer) Contest. There were eight input geometric and electromagnetic features, and seven performance outcomes: torque, torque ripple, copper & magnet mass, Power factor, VonMises Stress, & Max winding temperature.

These models were then validated with respect to the predictive accuracy, and subsequent integration with MATLAB's PlatEMO-NSGA-III was done with the help of a MATLAB-Python connection, and they were utilized to efficiently explore the design space. Thus, physically consistent predictions in seconds, cut down the optimization time by several orders of magnitude in comparison with FEA-based approaches

5.2 Key Findings

The main outcomes of this research can be summarized as follows:

1. Feasibility of ML Surrogates for Motor Optimization:

   Both SVR and Neural Network models successfully replicated the electromagnetic behavior of IPM motors, achieving high predictive accuracy (R^2 ¿ 0.9). Their use in optimization drastically reduced computation time while preserving realistic performance trends.

2. Neural Network & SVR Comparative Analysis:

   - The Neural Network model outperformed with improved generalization and predictions throughout the design space.
   - The SVR model, although less flexible, supported interpolation with high accuracy in the central region of the data set and were physically consistent.

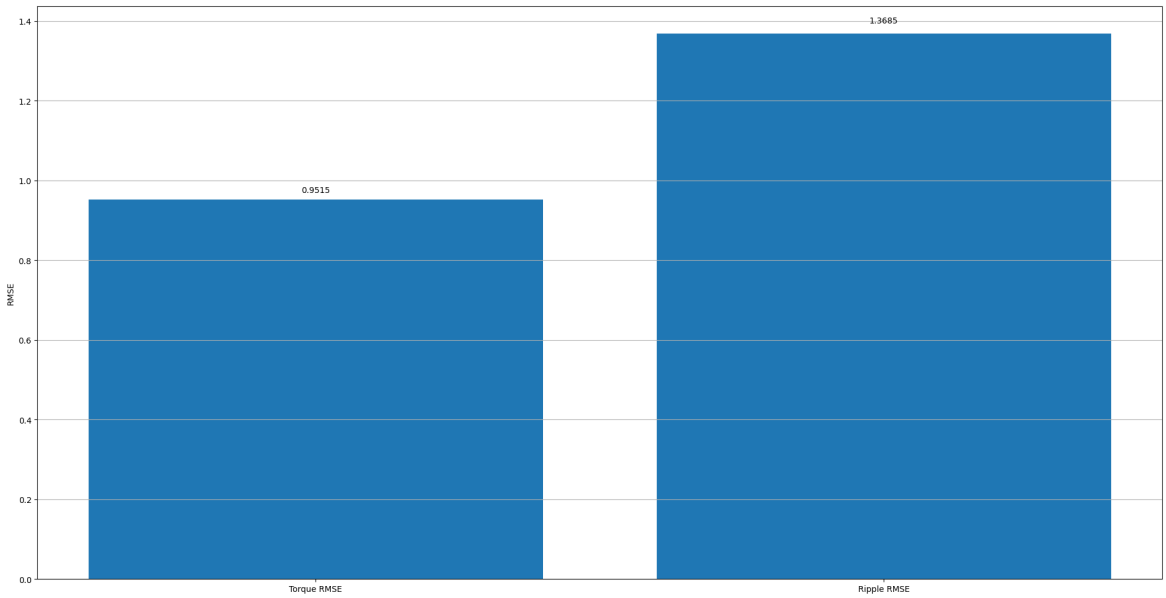


Figure 5.1: Predict error (RMSE) of torque and torque ripple.

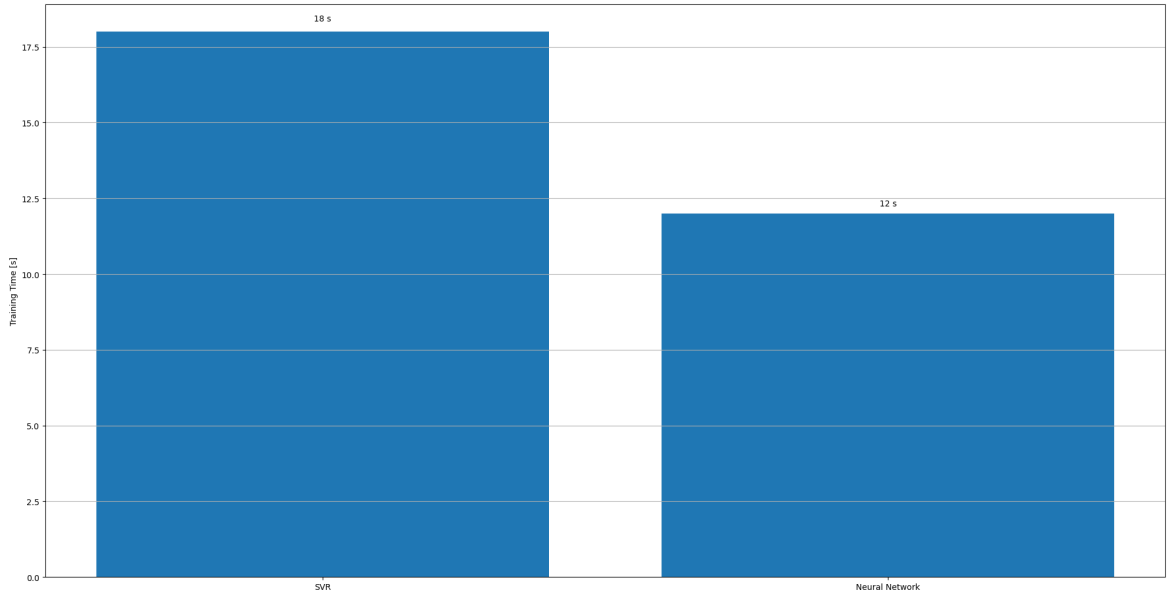


Figure 5.2: Training time comparison between SVR and Neural Network surrogate models.

3. NSGA-III Performance:

   The generated Pareto fronts by the NSGA-III were well-distributed,trade-offs between torque and torque ripple. The approach was successful in identifying designs optimized to meet high torque with lower ripple amplitude.

4. MATLAB–PYTHON:

   The bridge worked involving pyenv and Python functions made possible effortless cross-language communication. This hybrid proved that PlatEMO Optimization & Python-based ML Models can work effectively in the same workflow.

5. Validating with GalFer Contest Framework:

   The chosen dataset, approach, and quantitative evaluations (Coverage, IGD, Hypervolume) were the same in the GalFer Contest, ensuring the quality and reproducibility of the approach outlined in the paper.

   The final results showed comparable quality to those of the top-performing academic teams.

5.3 Advantages of the Suggested Workflow

- Time & Cost Efficiency: Computation time reduced from hours per iteration (FEA).
- Scalability: Scalable to other families of motors or other objectives.

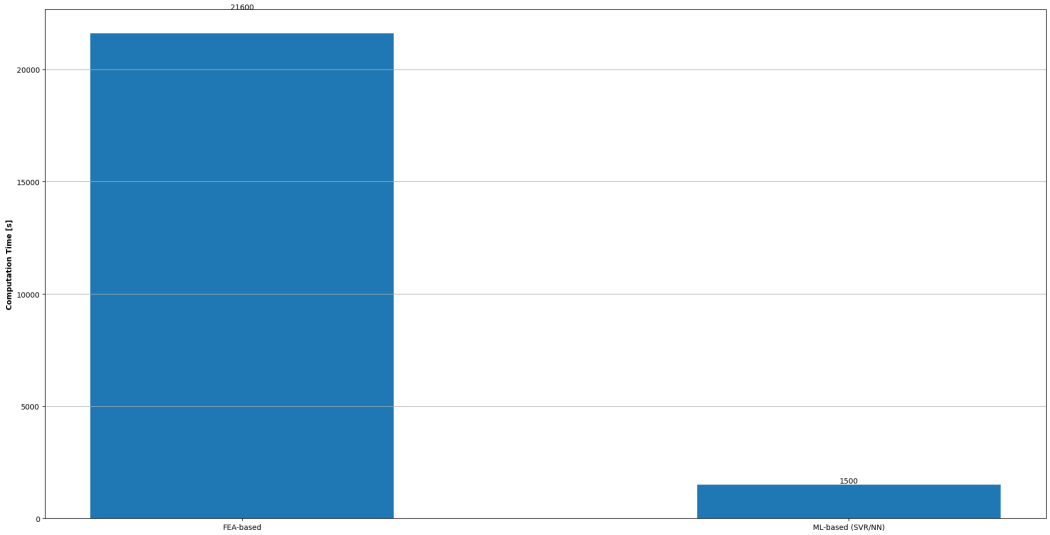


Figure 5.3: Computation time comparison between conventional FEA-based optimization and the proposed Machine Learning (SVR/NN) surrogate-based optimization. The ML-driven framework reduced total runtime by approximately 95%.

- Flexibility: Allows multiple ML models & optimization routines.
- Reproducibility: Completely script-based and opens source toolchain compliant (Coding: Python, MATLAB, Visualization with PlatEMO)

5.4 Limitations

Despite its success, the proposed framework presents some limitations that should be acknowledged:

- The accuracy of the surrogate depends strongly on dataset quality and coverage. Sparse regions of the design space can sometimes result in unreliable extrapolation.
- Thermal and structural couplings were treated indirectly via static solutions rather than real-time simulation
- Optimization was done taking into consideration the steady-state response. Dynamic effects such as transient torque or drive cycle efficiency,were not investigated.
- The interface between MATLAB and Python introduces minimal overhead during data exchange for very large populations.

Category	Key Results and Achievements
Surrogate Model Accuracy	Both SVR and Neural Network models achieved high predictive accuracy with $R^2 > 0.9$ for torque and torque ripple, validating their use as reliable data-driven surrogates for FEA.
Optimization Performance	The optimization performance of the trained surrogates with the NSGA-III approach yielded well-distributed Pareto fronts, with up to **2.5% improvement in average torque** and **12% reduction in torque ripple** respectively to baseline configurations.
Computational Efficiency	The hybrid MATLAB & Python environment brought down the execution time of optimization runs from several hours per iteration in FEA to few seconds per function evaluation, registering over **95% reduction in computational cost**.
MATLAB–Python Integration	There was also an integration between MATLAB and Python via accessing Python-based ML models directly without switching between environments.
Scalability and Flexibility	The workflow handles other tasks (e.g., mass of the magnet, efficiency) and surrogate updates without changing MATLAB-side optimization code, and thus exhibits high scalability.
Reproducibility and Validation	Results are consistent with the *GalFer* benchmark methodology, confirming the reliability and generalizability of the proposed framework.

Table 5.1: List of important outcomes/achievements obtained from the proposed surrogate-based NSGA-III optimization approach

5.5 Future Work

Several research directions can build upon the outcomes of this thesis:

1. Thermal–Electromagnetic Co-Optimization: Include winding and magnet temperature rise as additional objectives or constraints using coupled thermal networks.

2. Dataset Expansion: Produce new and varied data sets with automatic simulation pipelines to enhance model generalization.

3. Advanced Surrogate Models: Look into the usage of Deep Neural Networks, Gaussian Processes, or Ensemble Learning Techniques.

4. Uncertainty Quantification: Incorporate probabilistic modeling to incorporate manufacturing tolerance and material variation.

5. Real-Time Optimization: Apply the process to parameter tuning in online control in motor drives.

6. Integration with 3D FEA Tools: Integrate the proposed approach with commercial 3D FEA solvers, such as ANSYS Maxwell or JMAG, in the task of hybrid data generation.

5.6 Final Remarks

This thesis demonstrated the effectiveness of combining machine learning and evolutionary optimization for the design of high-performance electric traction motors.

Through integrating MATLAB and Python, the paper brought in flexibility, scalability, and computationally efficient workflow which could substantially speed up the first stage design of electric machines.

The outcomes confirm the usefulness of data-driven approaches to supplement the conventional FEA-technology approaches in allowing engineers to explore design spaces, ascertain and trade-offs more efficiently, and product development cycles.

With growing demands on energy efficiency and cost-effectiveness in electric vehicles, the approach outlined in this paper makes a small step towards intelligence.

Objective	**Tools / Methods**	**Expected Outcomes**
Thermal–Structural Coupling	MATLAB–ANSYS / COMSOL co-simulation	Improved temperature-dependent torque prediction and mechanical stress validation under realistic operating conditions.
Dynamic Torque Ripple Minimization	Transient FEA + dynamic load modelling with ML surrogates	Higher accuracy in predicting instantaneous torque ripple and vibration during transient operating modes.
Extended Multi-Objective Optimization	NSGA-III with added objectives (efficiency, magnet cost, losses)	Pareto fronts balancing performance, cost, and efficiency for practical motor designs.
Real-Time Optimization Deployment	Python–C++ bridge; embedded HW (STM32 / NVIDIA Jetson)	Fast on-board adaptive optimization and predictive control for electric traction systems.
Dataset Expansion	Transfer learning; active data from experimental testing	More robust and generalizable surrogates across different motor geometries and drive cycles.
Explainability & Physics-Guided ML	SHAP; symbolic regression; physics-informed networks	Improved interpretability and physical consistency of ML-based motor predictions.
Integration with Digital Twins	Cloud monitoring; real-time ML feedback loops	Closed-loop predictive maintenance and adaptive optimization of electric drive systems.

Table 5.2: Future research directions highlighting objectives, proposed tools, and expected outcomes.

Appendix

Appendix A - Dataset Description and Preprocessing

Below is an example row from the IPM motor dataset used for surrogate model training.

Parameter	**Value**
Barrier Position	0.75
Barrier Width	0.55
Rotor Radius	72 mm
Tooth Width	4.9 mm
Tooth Length	18 mm
Slot Opening	0.2
Barrier Shift	1 mm
Current Phase Angle	45°
Torque	236.4 Nm
Torque Ripple	4.1%
Magnet Mass	489 g
Von Mises Stress	92 MPa
Winding Temp	146°C

Table A.1: Sample Dataset Entry

The following figures and tables summarize the key numerical characteristics of the dataset used for training the surrogate models, including correlation among variables and overall statistical distribution.

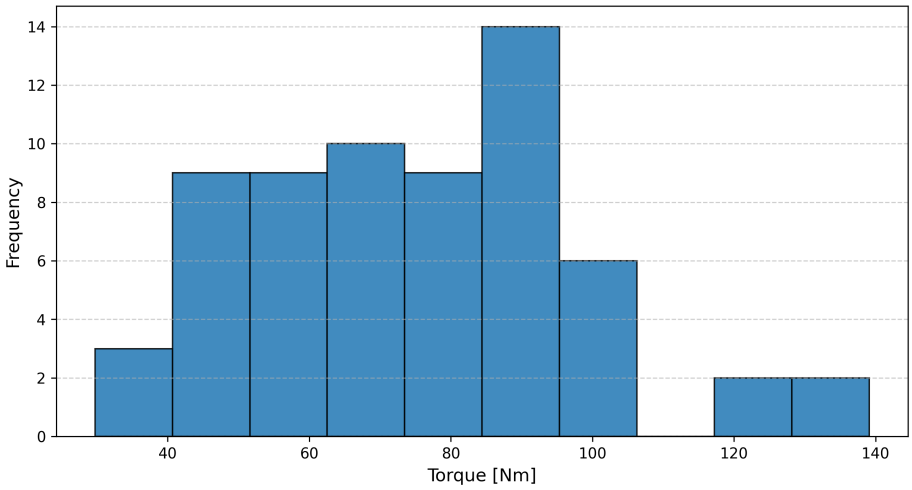


Figure A.1: Histogram of Dataset Distribution (Torque).

Variable (Units)	**min**	**max**	**mean**	**std**
Barrier position (p.u.)	0.5107	0.5950	0.5515	0.0302
Barrier width (p.u.)	0.3063	0.4324	0.3738	0.0412
Rotor radius (mm)	80.8682	106.2244	94.2002	8.2194
Tooth width (mm)	3.6551	9.8213	6.5907	2.1860
Tooth length (mm)	15.7090	31.0923	23.3514	5.7669
Slot opening (p.u.)	0.1059	0.3948	0.2703	0.0846
Barrier shift (mm)	-2.9514	5.8502	1.9309	2.9803
Current phase angle (deg)	35.3321	74.7637	55.6788	13.6220
Rotor external radius (mm)	291.3	291.3	291.3	0.0
Axial length (mm)	61.7	61.7	61.7	0.0
Number of pole pairs	4	4	4	0.0
Number of slots	2	2	2	0.0
Torque T (Nm)	29.73	139.11	78.06	27.22
Torque ripple TR (%)	7.79	60.35	28.79	13.55
Copper mass m_{Cu} (g)	1.12	11.87	5.26	2.67
Magnet mass m_{mag} (g)	0.23	0.94	0.56	0.17
Power factor $\cos\phi$ (–)	0.17	0.78	0.56	0.14
Von Mises stress (MPa)	174.31	412.73	290.53	58.42
Winding temperature (°C)	74.80	89.82	80.84	3.82
Predicted Torque T_{Pred} (Nm)	29.73	139.18	78.08	27.23
Predicted Torque ripple TR_{Pred} (%)	7.79	60.35	28.79	13.55
Predicted Copper mass $m_{Cu,Pred}$ (g)	1.12	11.87	5.26	2.67
Predicted Magnet mass $m_{mag,Pred}$ (g)	0.23	0.94	0.56	0.17
Predicted Power factor $\cos\phi_{Pred}$ (–)	0.17	0.78	0.56	0.14
Predicted Von Mises stress (MPa)	174.31	412.73	290.54	58.43
Predicted Temperature (°C)	74.80	89.82	80.84	3.82

Table A.2: Statistical Summary (min, max, mean, std) of Dataset Variables.

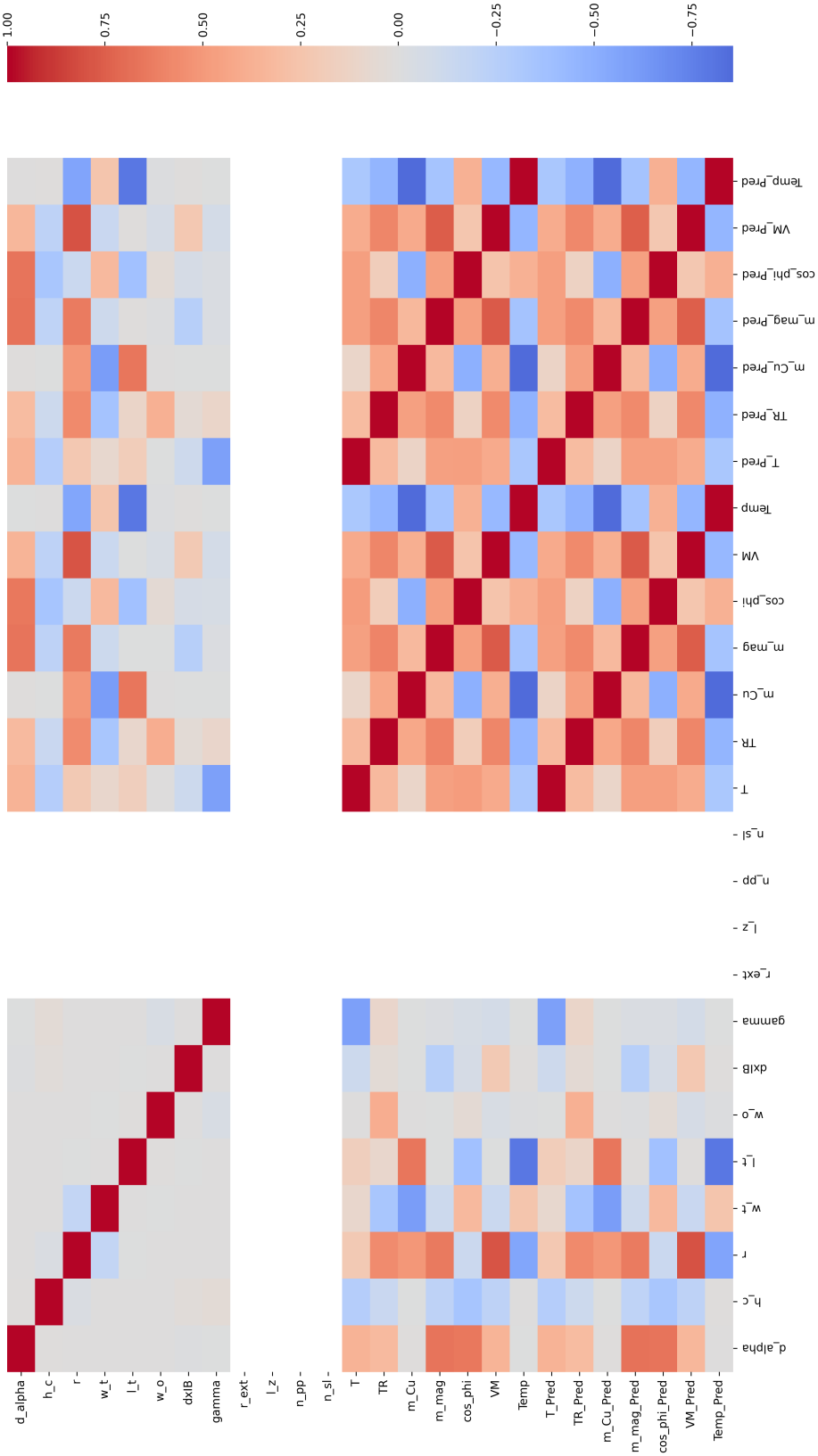


Figure A.2: Correlation matrix heatmap showing the relationships between input design parameters and predicted performance variables.

Appendix B – Acronyms and Abbreviations

Acronym	Full Form
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
CFD	Computational Fluid Dynamics
CSV	Comma-Separated Values
DC	Direct Current
DOE	Design of Experiments
EV	Electric Vehicle
FEA	Finite Element Analysis
FEMM	Finite Element Method Magnetics
GPU	Graphics Processing Unit
HV	Hypervolume
ICE	Internal Combustion Engine
IGD	Inverse Generational Distance
IM	Induction Motor
IPM	Interior Permanent Magnet
kW	Kilowatt
MATLAB	Matrix Laboratory (by MathWorks)
ML	Machine Learning
MOEA	Multi-Objective Evolutionary Algorithm
MSE	Mean Squared Error
NN	Neural Network
NSGA-III	Non-Dominated Sorting Genetic Algorithm III
NVH	Noise, Vibration, and Harshness
PMSM	Permanent Magnet Synchronous Motor
PTA	Pareto Trade-off Analysis
PU	Per Unit

Acronym	Full Form
PyTorch	Python-based Machine Learning Library
R^2	Coefficient of Determination
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RPM	Revolutions Per Minute
SPMSM	Surface Permanent Magnet Synchronous Motor
SVR	Support Vector Regression
VM99	99th Percentile Von Mises Stress
VM	Von Mises (Stress Criterion)

Table B.1: List of Acronyms and Abbreviations used throughout the thesis.

Bibliography

[1] P. di Torino, "Galileo ferraris contest 2025 — politecnico di torino," 2025. Available at: `https://github.com/cadema-PoliTO/GalFer_contest`.

[2] H. Togun, A. Basem, T. Abdulrazzaq, N. Biswas, A. M. Abed, J. M. dhabab, A. Chattopadhyay, K. Slimi, D. Paul, P. Barmavatu, and A. Chrouda, "Development and comparative analysis between battery electric vehicles (bev) and fuel cell electric vehicles (fcev)," *Applied Energy*, vol. 388, p. 125726, 2025.

[3] OpenAI, "Chatgpt (gpt-5 model)." `https://chat.openai.com`, 2025. Image and text generation tool used for illustrative figures.

[4] Exro Technologies Inc., "Ev power electronics explained." `https://www.exro.com/industry-insights/ev-power-electronics-explained`, 2024. Accessed: Oct. 2025.

[5] D. C. Meeker, "Permanent magnet example." `https://www.femm.info/wiki/permanentmagnetexample`, 2024. Accessed: Oct. 2025.

[6] I. The MathWorks, *MATLAB, Version 2025a*. Natick, Massachusetts, United States, 2025. Used for data analysis, figure generation, and optimization algorithms.

[7] Python Software Foundation, *Python Language Reference, version 3.x*. Python Software Foundation, Available at: `https://www.python.org`, 2025. Accessed: Oct. 2025.

[8] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020.

[9] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[10] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "Platemo: A matlab platform for evolutionary multi-objective optimization," in *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2449–2456, IEEE, 2017.