Technical University of Applied Sciences Würzburg-Schweinfurt (THWS)

Faculty of Computer Science and Business Information Systems

Master Thesis

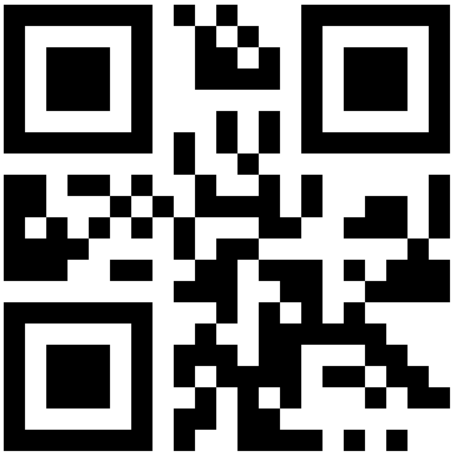
Electrical Engine Efficiency Prediction Bypassing Finite Element Analysis

Submitted to the Technical University of Applied Sciences Würzburg-Schweinfurt in the Faculty of Computer Science and Business Information Systems to complete a course of studies in Master of Artificial Intelligence

Lilly Abraham
K64889

Submitted on: 11.12.2024

Initial examiner: Prof. Dr. Magda Gregorová
Secondary examiner: Prof. Dr.-Ing. Mercedes Herranz Gracia



Abstract

The thesis explores an approach to predict Key Performance Indicators (KPI)s of topology invariant Permanent Magnet Synchronous Machine (PMSM) Electric Motors from its geometric, physical and simulation parameters. I posit to model the dynamics of Electric Motor functionality by creating surrogate models trained with Finite Element Analysis (FEA) simulations from its parametric description. The KPIs to be predicted are vectors of numerical values which can be visualized as the Torque curve (2D) and the Efficiency grid (3D) respectively. I note the relationship between the Torque curve and the Efficiency grid and incorporate my learnings of both the KPIs nature into the solutions's modelling. I first parameterize the Electric Motor design such that it is feasible to convert into a tabular representation. Next, I create a table with relevant attributes and design a Multi Linear Perceptron (MLP) to train it in a supervised manner. Subsequently I regularize the loss functions in a way that would smoothen out the plots for both the KPIs. Then I evaluate the predictions with the test target values by experimenting with various hyperparameter settings and as a baseline with the average of the labels of the training dataset. Additionally I conduct a study to model the task as its graph representation and use Graph Neural Network (GNN)s to predict the KPIs. Lastly I enable the KPIs plot visualization in a manner presentable to the client Valeo(Automaker Company specializes in electric motors for cars).

Acknowledgement

I would like to thank my supervisor Prof. Dr. Magda Gregorová for her guidance and support throughout the course of my thesis. Her dedication and commitment to our work has been inspiring to me. Notably on how we transformed statistical math into modelling that I might have developed a new love for academia. I would also like to express my sincere gratitude to Valeo for providing us with the dataset. Special thanks are in order to Daniel and Leo for sharing valuable insights of the data from an electromechanical standpoint and for giving me a detailed understanding of my task. I owe my accomplishments to my Family who have been very instrumental in making it possible for me to pursue a Master's degree outside my home country and their endless support throughout enabled me to get to my thesis moving in the right direction. Finally, I humble myself before God Almighty for all his blessings and for giving me the strength to persevere and bring my dreams to fruition.

Contents

Abbreviations

GNN	Graph Neural Network
MLP	Multi Linear Perceptron
KPI	Key Performance Indicator
EM	Electric Motor
FEA	Finite Element Analysis
CNN	Convolution Neural Network
D	Dimensional
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
NaN	Not a Number
ReLU	Rectified Linear Unit
GPU	Graphics Processing Unit
MP	Message Passing
PDE	Partial Differential Equation
PMSM	Permanent Magnet Synchronous Machine
GCN	Graph Convolutional Network

Chapter 1

Introduction

Electric Motor (EM) are the heart of automobiles and are responsible for the propulsion of the vehicle. They operate by converting the electrical energy into mechanical energy by magnetic field excitation. Permanent Magnet Synchronous Machine (PMSM)s consists of permanent magnets in the EM's rotor which is responsible for generating the magnetic field. In the design of PMSM EM, vast amounts of data are generated to determine which design fits best to Key Performance Indicator (KPI)s. Traditionally these KPIs are inferred from the description of an EM design using Finite Element Analysis (FEA) simulations by approximating the solutions of the Maxwell's equations which are essentially Partial Differential Equation (PDE).

KPIs of an EM are representative characteristics of a motor drive and is essential to judge the performance of the motor before manufacturing it. The Torque multiplied by the Angular Velocity gives us an insight of the total Power of the EM. The Efficiency map is a behavior map that expresses the motor efficiency function in the torque-angular velocity domain of an EM. One calculates such kind of behavior maps when there are multiple operating points to consider within the EM's drive cycle. The other KPIs such as Losses and Torque ripple are also behavior maps similar to the Efficiency KPI. In both the KPIs, maximum torque is dependent on the EM thermal limit whereas the maximum angular velocity is dependent on the structure of the rotor. The paper in Ref. [35] also cites that the Efficiency vs Torque and Angular Velocity map is most representative of the characteristics of the EM.

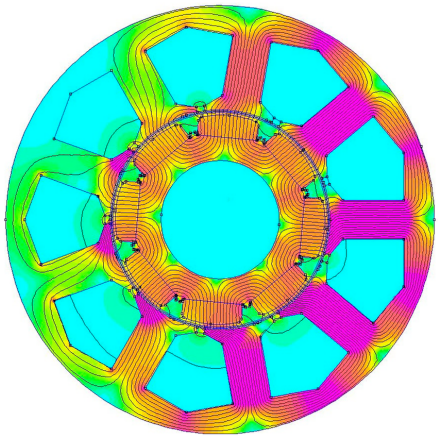


Figure 1.1: FEA Simulation of a PMSM (Source : [09])

FEA are numerical methods that discretize any physical phenomena to simulate and understand its performance under given loads and boundary conditions during design phase. This enables designers to understand the performance and make a well informed decision on manufacturing it. Although FEA only gives approximate solutions, it is regarded the best among the alternatives that exists. FEA as a method

are used in several domains ranging from civil, mechanical, aerospace, automotive and electrical engineering to name a few.

Fig. 1.1 represents an FEA discretized mesh of a PMSM EM. The color scale obtained in represents possibly the magnetic flux density flowing across different parts of the EM.
FEA simulations here discretize the geometric parameters of an EM into finite elements. The simulations are then carried out by creating a mesh with the discretized points that approximate the shape of the EM and then solving the PDEs for each of them. This is typically done to simulate the magnetic field distribution of the EM and thus assists to generate its KPIs. It requires domain knowledge in motor physics and additional setup to run simulations typically in `Matlab`[1]. Moreover, FEA simulations are famously expensive in terms of time taken to solve the equations which are also non differentiable. Despite the computational burden it comes with, the authors from Ref. [28] defends FEA to be most appropriate to generate EM efficiency maps in terms of accuracy based on experimenting overall error rates across operating ranges for different EM losses.

FEA though well established in the EM design is very resource intensive, time consuming and does not allow for high-throughput engine design optimization. Hence, the growing need to replace it with a worthy substitute. Deep learning can be seen as the most eligible surrogate of FEA simulator as they can provide almost accurate results in relatively no time. Particularly for the Efficiency KPI, FEA simulations may take hours to days as it needs to generate for all operating points in the Efficiency map. FEA-based artificial neural network models have been utilized in other applications such as for predicting stress distribution in 3D printing, bend angles in laser-guided bending, and performance of thermoelectric generator as cited in Ref. [07].

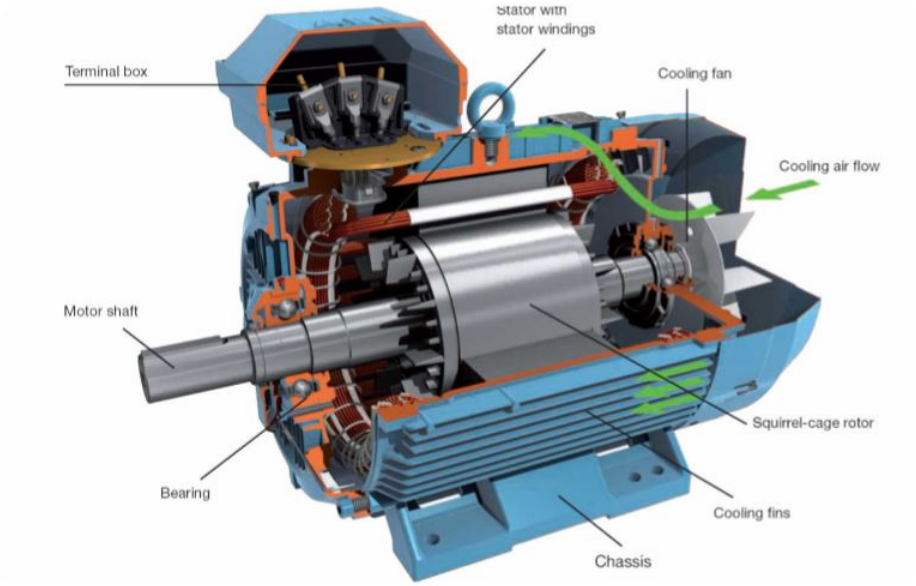


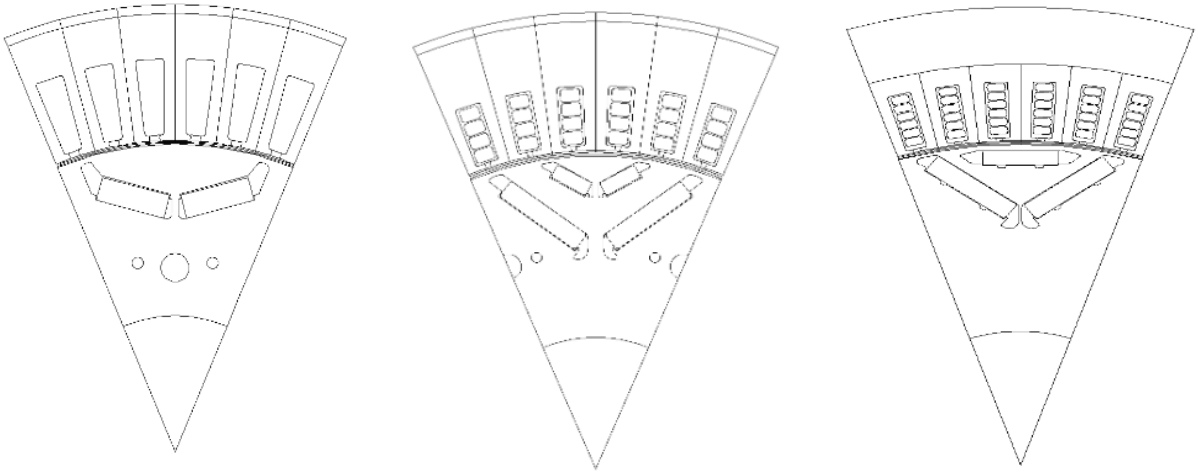
Figure 1.2: EM Motor (Source : https://www.valeoservice.in/en-in/newsroom/basic-understanding-automotive-electric-motors)

The actual engine data of `Valeo`[2] is used here as the dataset comprising of multiple variant designs of the Double-V magnet EM topology. Valeo is an Automaker company that specializes in manufacturing electric motors for automobiles.
Fig. 1.2 depicts the structure of a PMSM EM manufactured by Valeo. The rotor rotates around the Stator to produce the magnetic field whereas the Stator is the stationary part that generates the force to rotate the

[1] `https://de.mathworks.com/`
[2] `https://www.valeo.com/`

Rotor.

The 3 motor topologies manufactured by Valeo are displayed in Fig. 1.3:
1. Single V Magnet - Consists of a single V magnet shown in Fig. 1.3a.
2. Double V Magnet - Consists of double V magnets shown in Fig. 1.3b.
3. Nabla Magnet - Consists of a single V Magnet and a delta magnet shown in Fig. 1.3c.



(a) Single V Magnet (b) Double V Magnet (c) Nabla Magnet

Figure 1.3: EM Magnet Topologies (Source : Valeo)

Fig. 1.4 and Fig. 1.5 give a glimpse of the EM's KPIs to be predicted.

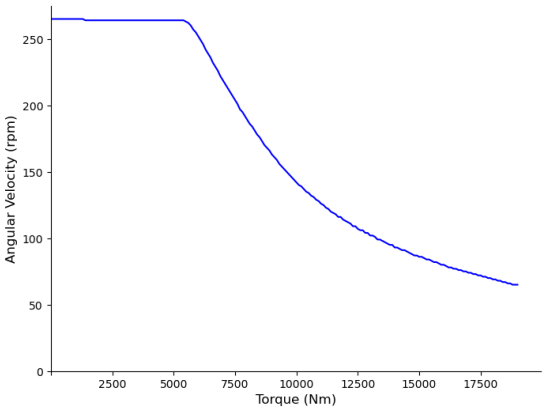


Figure 1.4: Torque Curve

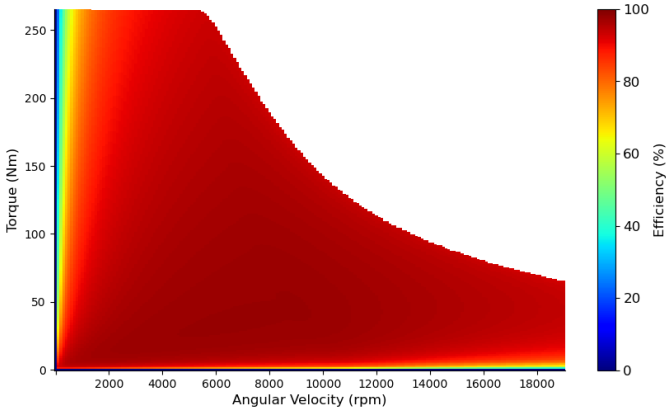


Figure 1.5: Efficiency Grid

Figures Fig. 1.4 and Fig. 1.5 give a visualization of the Torque curve and the Efficiency Grid of a EM design variant across angular velocities ranging from 0-19100 rpm and for positive torque values. The angular velocities are represented as rotations per minute(rpm) and the torque as Newton meter(Nm).
I let the relation between the Torque curve and the Efficiency grid speak for itself from the images. The relationship is that the Efficiency envelope is of the same shape as the Torque curve. The values within the Efficiency envelope can be identified with differing contour shades whose level of shading is shown to the right.

The thesis explores a way to do light-weight surrogate modelling of the current process as is highlighted in Fig. 1.6 by exploiting data-driven deep neural networks to approximate the KPIs derived from FEA

simulations. Complete replacement of FEA simulations with deep learning models is not feasible however I can exploit the use of deep neural networks trained on FEA simulated data to reduce the computation burden of running these simulations repeatedly in the future.

Fig. 1.6 gives an outlook of both the current approach and the proposed approach to generate the KPIs. I discuss each component of the flowchart below :

1. **EM designs**
   The design of each of the EM variants described parametrically for its geometric, physical and simulation features are regarded as the input. The inputs here are in its numerical equivalent format.

2. **Current Approach**
   Approach followed currently by Valeo.

   (a) **Matlab Script 1**
   First a Matlab script creates a design mesh of the EM from the parametric description of the EM's geometric and physical features.

   (b) **FEA Solver**
   Then, multiple FEA simulations are carried out on this EM mesh. The FEA Solver then generates the by-products associated with the magnetic flux of the EM.

   (c) **Matlab Script 2**
   The outputs from the FEA simulations are then post processed and the intermediary outputs which are matrices of values in matlab compatible format are fed to the next stage.

   (d) **KPI Creator**
   Another round of post processing is done on the magnetic flux products to generate the Power and Torque associated with the EM design. KPI Creator settings such as length of the motor, maximum rotation speed as well as electrical settings for instance input voltage and current are varied to generate the KPIs.

3. **Proposed Approach**
   Approach I undertake as surrogate modelling.

   (a) **Data Preprocessing**
   The input features are pre-processed and converted into its tabular representation such that it is suitable to be fed into the Neural Network. For training the network, additionally the targets are taken from the **Matlab Script 2** which also serves as the ground truth represented as a dotted line.

   (b) **Neural Network**
   I have used Multi Linear Perceptron (MLP) as the deep learning model which is made up of fully feedforward connected layers. For training, the targets are also considered to minimize the loss of the predictions to be generated. However for inference, only the inputs are used to generate its approximated targets.

   (c) **Data Postprocessing**
   The predictions generated by the neural network is post processed to be similar to the targets obtained after final postprocessing in the **KPI Creator** in terms of dimensions.

4. **KPI Plots**
   The targets which are vectors of numerical values are displayed graphically.

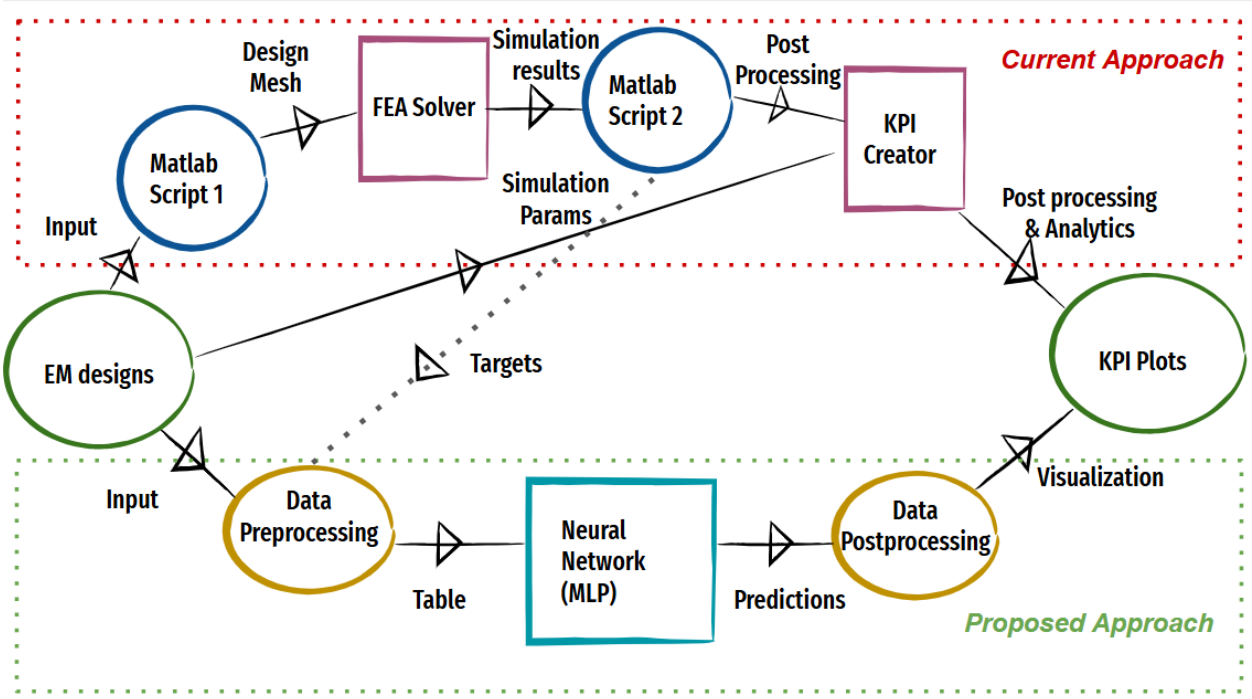


Figure 1.6: EM Design Flowchart

1.1 Objective

The objective of the work is to verify whether surrogate modelling to replace FEA simulations is feasible. Our task is a supervised learning task to predict 2 KPIs namely the Torque curve and the Efficiency grid which are vectors of numerical values from the parametric description of topology-invariant EM with FEA simulated KPIs as ground truth. The Torque curve is a vector of torque values across certain angular velocity ranges which can be visualized as a 2Dimensional (D) plot whereas the Efficiency grid is a matrix of efficiency values of all combinations of torque ranges $\times$ angular velocities ranges and can be visualized as 3D heatmap of contour plots. Both the KPIs are continuous values which makes the task a regression problem. The remaining KPIs such as Costs, Vibration losses, Torque ripple among others can be calculated from these 2 KPIs for instance the losses are inversely proportional to the Efficiency values.

1.2 Motivation

Our motivation to undertake this thesis is to lay the ground work for the problem's inverse formulation of generating EM designs in the future. To my knowledge, there has been no work yet on Generative AI in the domain of EM designs. The future research goal would be to condition the generative model on the predicted KPIs to be able to self-generate the most efficient EM designs. From an application standpoint, it would be beneficial for Automaker companies to refer the generated EM designs to be able to suggest the most appropriate EMs for automobiles as per customer requirements on say the horsepower of the car before manufacturing them. Companies could then evaluate the KPIs of the synthetically generated EM designs and judge on its usefulness.

1.3 Problem Statement

Given the Data $\mathcal{X} = [x_1, x_2, ..., x_n]^T \in \mathbb{R}^n$, where n is the number of parameters of the EM designs. The targets can be referred to Torque curve as $\mathcal{Y}_1 = [y_1, y_2, ..., y_h]^T \in \mathbb{R}^h$ where h is the number of participating

angular velocities and Efficiency map as $\mathcal{Y}_2 \in \mathbb{R}^{w \times h}$ representing a grid of efficiency values defining the operating range of the EM, where w denotes different torque levels and h the angular velocities. Each element $\mathcal{Y}_2(i, j)$ represents the efficiency at torque i and angular velocity j. The aim is to approximate the targets by training a MLP model $\mathcal{M}$ to learn the mapping between $\mathcal{X}$ and $\mathcal{Y}_1$, $\mathcal{Y}_2$.

1.4 Research Question

The Research Questions I aim to address are presented as follows :

1. **Is it feasible to predict the targets as a multi regression problem using a single model?**
   Since I have 2 targets of continuous values to predict, the task becomes a multi regression problem. The model architectures in Section 4.1 shines light on how I go about it.
2. **How to handle varying dimensions within a target for model training?**
   The dimensions of the Efficiency grid differ across EM variants. However the targets which has to be supplied to the model needs to be of a fixed size. I discuss in Section 3.1.3 how I mitigate this problem.
3. **How to accommodate predictions of targets of varying ranges with a single model?**
   The ranges among the 1st and 2nd target vary significantly and is yet another challenge I overcome in Section 4.2.2.
4. **How to predict 2 targets where the dimensionality of a target is dependent on the other using a single model?**
   Given the observation that the efficiency envelope is controlled explicitly by the torque curve. I discuss how to include this information into modelling in Section 4.5. There has been literature on modelling wherein the values of a target are dependent on the values of another target. However, my situation is tricky in the sense that instead of the values the dimensionality of the Efficiency grid is dependent on the values of the Torque curve.

1.5 Thesis Structure

Over the course of the thesis I shall refer the Torque curve as Torque KPI and the Efficiency grid as Efficiency KPI respectively. The remainder of the thesis is organized to follow sections namely Literature Review, Dataset, Modelling and Evaluation, Experiments and Results, Graph Modelling, Conclusion, Bibliography and Appendix. In Literature Review section I review the works that has already been carried out in this domain. In the Dataset section a detailed insight on how the data is structured is elaborated. In the Modelling and Evaluation section, I introduce the network architectures and loss regularization techniques used to tackle the problem. The outcomes of the work are presented in Experiments and Results chapter in addition to other findings I unearth. Graph Modelling section is an empirical study which I conduct outlines the background of Graph Neural Network (GNN)s primarily Heterogeneous GNNs and defines its concepts. I also attempt to present a workflow on how to use GNNs for my task. Conclusion chapter summarizes the thesis briefly and would also give a glimpse into possible areas of improvement as future works. Bibliography section lists out the articles cited for this thesis. Lastly I share all supplementary information in the Appendix section.

Chapter 2

Literature Review

In this chapter, I endeavour to review literature on works carried out in Surrogate Modelling the FEA simulations of EM designs. In addition, I have also attempted to include writers perspectives and approaches taken across multiple stages of modelling ranging from data preprocessing to evaluation.

2.1 Surrogate Modelling

2.1.1 Convolutional Neural Networks Vs Multi Layer Perceptron

There has been extensive research in modeling the EM with Convolution Neural Network (CNN) based on the images of the motor cross-section. Reproducing images is not the most wisest approach as can be inferred from Ref. [17] where the author presents numerous scenarios of generated images not being upto mark. These scenarios include difficulty to render the background accurately by only focussing on one subject, compression quality not being good enough, inability to focus on details and make images realistic among many other discrepancies. The research largely encompasses evaluation of image generation of faces which could be concluded that deepfakes generated were quite easy to distinguish from real ones.

Furthermore, researchers of Ref. [15] also present their approach on handling surrogate modelling of topology invariant Interior Permanent Magnets motors to predict its Torque characteristics. Generally the topology differs based on the count of the magnets as is evident from Fig. 1.3. The writers claim to have used the cross-sectional images of motor designs in addition to the magnetic flux distribution of the Stator as input to train the CNN. The auxiliary input is supplemented by FEA simulations which is enriched with the nature and placement of the magnets in the Rotor. Although, FEA is used, the product here is generated quicker and thus does not add on largely to its overhead on time complexity. They also claim to have improved the generalization performance of CNNs with this strategy. In addition, they suggest such domain knowledge modelling decisions can be beneficial for both parameter and topology optimization of EM as well as for prediction of other KPIs.

Vivek *et al.* in Ref. [36] highlights that predicting KPIs with tabular data is significantly more efficient than using the images because the latter could result in less accurate designs due to the need of high resolution images in addition to its overall computation cost required for training.
Their work involves the use of a Variational Autoencoder to predict the KPIs with an MLP as well as sampling the latent space to generate new EM designs. Their experiments conclude that MLPs trained on the parametric description can better infer targets such as Induced Voltage and harmonic distortion that are linearly dependent to the designs when compared to cogging torque. To learn the non linear functions, they had experimented with CNNs which could infer all targets just as well because it inherently takes into account the pixel spatiality from RGB images of motor designs. In addition, they had built a hybrid model to utilize the image and parametric description but have reported comparable performance with that of CNNs. They also suggest a slight decline in accuracy for a linearly dependent target for the CNN model as image resolution are constrained by the precision of the inputs.

Additionally, in Ref. [10], the authors compare and evaluate the performance of surrogate modelling Surface Permanent Magnet's parametric and image based designs.

2.1.2 Machine Learning Approaches

The authors of Ref. [08] presents a method to predict 2D flux maps of an Interior PMSM motor using classical machine learning ensemble regression models. Therefore, for each coefficients of the 2D flux maps, a separate regression model is trained which are then ensembled to make target predictions. Although there are classical Machine Learning models which handle multi regression, they do not perform very well with higher dimensions as opposed to deep neural networks. The writer also points out that these models fare better than deep neural networks when it comes to data required and training time.

Similar to my usecase for modelling EM of cars, the writers of Ref. [33] investigate how to compute the efficiency map of the automobile Toyota Prius. The methodology used is to observe Magnetic field flux density to predict how it evolves over time for different operating points of the efficiency map.
The paper Ref. [34] also examines a study of how a machine learning observer can augment the performance of torque estimation of induction motors trained with deep neural networks. The motivation behind using the observer is made by domain aware knowledge that the torque estimated depends on the stator's magnetic flux. The authors claim to have made this possible by incorporating the information of angular velocity, voltage and current into the network to realize the EM's torque. They also indicate that physics modelling of the network enabled them to develop light-weight models with better accuracy.

2.1.3 Hybrid Models

The researchers of Ref. [07] state that the use of domain knowledge improved the accuracy of surrogate modelling EM by creating a hybrid of both physics and data driven based models. In addition, it presents a workflow for surrogate modelling the air gap torque from FEA simulated data and compares and contrasts the computationally efficiency with regards to both approaches.

Similarly, Yusuke *et al.* in Ref. [38] and Ref. [39] diagnose that a physics assisted neural network significantly improved the accuracy performance when predicting the cogging torque of Permanent Magnet Synchronous Motors. Their experiments involve approximating the cogging torque with a linear subdomain model which serves as an additional input to the neural network when making the final prediction. With this methodology, they also disclose that a significant less data than estimated would suffice.

2.1.4 Transfer Learning

Works by Arbaaz *et al.* in Ref. [16] explores methodologies to extend an already trained model to predict efficiency maps for a topology it was untrained for by exploiting the concept of transfer learning. The writers claim to have this accomplished by a mixture of greedy pretraining and then overall finetuning the model. They do so by freezing the pretrained network which they identify as common knowledge so that gradient flow is disabled and substitute the other layers of the pretrained model with new layers which is essential for the new task.
In addition to a topology change, this paper also explore transfer learning for different label than the pretrained model given that the labels are similar in nature to each other. This eventually assist the model to generalize better and conserve the time it would have spend training from scratch by knowledge reusing. Finetuning is a also a boon when the dataset for the new topologies are limited in existence.

2.2 EM design generation

Evolutionary algorithms such as genetic algorithms are traditionally used as multi-objective optimization algorithms to generate the designs by iteratively updating the design parameters whereas the FEA simulations

evaluate the performance of each design. The optimization algorithms are itself incredibly time consuming because fitness evaluation will need to be performed multiple times with FEA. In addition a larger number of iterations of these algorithms is necessary to evaluate each design candidate in order to explore the entire design space. I intend to surrogate model the FEA simulations which serves as the baseline to eliminate the role of evolutionary algorithms as well for generating designs as was also discussed in Section 1.4. The latter could potentially be seen as a future work since the time complexity associated with the Genetic Algorithms may discourage users to wait until convergence and rather persuade them to be satisfied with a suboptimal design.

Marius *et al.* in Ref. [37] undertake works on optimizing EM design generation by compressing its parametric description across multiple topologies into a latent space using a Variational Autoencoder thereafter from which new EM designs can be sampled.
Bucher *et al.* in Ref. [14] proposed a deep conditional generative design workflow to generate complete parametric description of Engineering Structures by taking as input the partially defined design and its performance attributes generated by FEA software. The workflow comprises of a conditional variational autoencoder as proof of concept by learning a joint probability distribution between the parametric description and its performance attributes.
This would be a useful read when there is a plan to extend the current work with its inverse formulation. The authors also justify the usage of parametric description of the designs to be flexible for different model configurations and datasets. Furthermore, the writers claim that their approach as opposed to evolutionary algorithms have multiple benefits such as complete control over the sample space of generated designs, expressive designs, reduced computational costs and model transferability among others.

2.3 Data Preprocessing Techniques

The authors of Ref. [37] discloses the solutions taken to address the problem wherein unique parameters of 1 topology are absent in another topology. They mitigate this concern with 2 approaches: Firstly, they default the missing values with an arbitrary fixed constant such as 0s, this inturn pushes the model to learn a nonsensical value for irrelevant features. However they also note that this shortcut results in wasting model capacity making the learning unnecessarily more difficult. Therefore, they have also used a second approach termed as Masked Learning Process wherein only features relevant for the motor topology are considered for modelling loss reconstruction of the model.

2.4 Evaluation Techniques

Arbaaz *et al.* in Ref. [29] attempts to generate the efficiency map by first generating its Flux linkage maps and the Torque curve thus accounting for geometric and operating point variations. Two interesting points are made in this paper:
First, since the number of excitation points vary across designs a model suitable for handling variable input sequence length is needed which implies that the values within the torque curve which are the excitation points are predicted. This further results in conserving training time when predicting a fixed sized grid. They also use confusion matrix for efficiency threshold so designer can filter out efficiencies in the operating range one is most eager to find out.
Secondly, for efficiencies being predicted outside the excitation points, the authors devise to provide an uncertainty measure to quantify confidence level using Monte Carlo dropout. Reason being the error rate in the efficiency grid is maximum at the envelope of the curve. This factor gives the end user the flexibility to choose KPI generation with the FEA simulations or the surrogate model.

Studies in Ref. [30] presents an interesting take on evaluating the performance of EM with electrical engineering inspired benchmarks. They highlight the inadequacy of evaluations generated by classical Machine Learning approaches such as Mean Squared Error (MSE), R2 Score and Symmetric Mean Absolute Percentage Error. The argument is that these metrics favor the static area in the KPIs which are relatively easier to predict.
Meanwhile the dynamic areas in the KPIs are not projected as much having less dominance with respect

to coverage area. They experimented this finding with Induction Motors to generate its Torque Vs Angular Velocity predictions. The dynamic parts they considered were typically the regions in the curve before saturation is achieved. In such cases the Machine Learning metrics could give an over optimistic score when compared to Electrical Engineering designed metrics.

Researchers in Ref. [32] also discusses approaches to judge the accuracy with a physics-aware metric for estimating magnetic field of low frequency electromagnetic devices. The do so by quantifying the uncertainty of the predictions by adding a probabilistic component to the weights of the neurons in the network architecture. The technique they had employed is known as Monte Carlo dropout with which they generate uncertainty maps. Thus, the distribution of the predictions they generate more closely approximate that of the same generated by FEA simulations as the accuracy improves.
Interestingly they also discuss the possibility of modelling the usecase with Graph Convolutional Network (GCN) instead of CNNs. They suggest that rather than images, graphs are better suited as they can handle unstructured design mesh which is typically fed into the FEA Solver as is shown in Fig. 1.6. However, in my case study I aim to use GNN on the parameteric description of the EM.

The writers of Ref. [31] emphasize the importance of having a very low error rate for the Efficiency Maps by FEA simulations as the drive range of a vehicle's efficiency is determined with this KPI. The Efficiency KPIs generated by the FEA simulations are obtained from torque, copper loss iron loss and mechanical loss. They study methods to further improve the accuracy of FEA simulations of Interior Permanent Magnets by modelling the effect of losses such as minor hysteresis loss, stray loss, AC loss and manufacturing degradations at different operating points into these simulations.

Chapter 3

Dataset

Valeo has shared 1481 Excel Workbook files for each variant of three-phase interior PMSM EM with 8 poles and 48 slots which are the 6 stator slots per cross-section. Around 89 parameters which comprises of the geometric, physical and simulation properties of the motor are chosen among the 196 parameters depending on its overall variability and significance. This was a design decision I had made based on my understanding of the data.
Moreover, I detail out the structure of the dataset files in Appendix A.1.1. Fig. 3.1 shows the geometry of a whole Double V motor which can be sliced into 8 identical parts owing to the EM design's rotational symmetry.

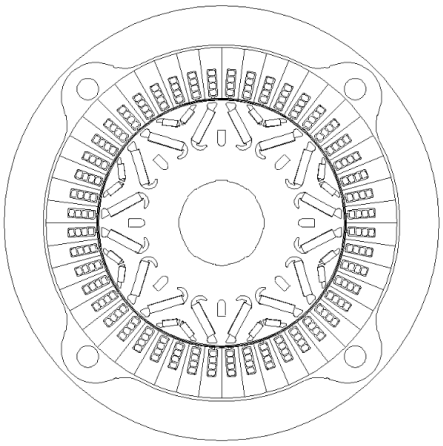


Figure 3.1: Complete EM Geometry (Source : Valeo)

I have also included a hand drawn sketch from my understanding of how the geometry of 1/8th cross-section of the same motor looks like in Fig. 3.2. This particularly comes in handy when creating the graph representation of the motor in Section 6.4.1. From the sketch, it can be made out that the EM very largely is comprised of the Rotor and the Stator separated by the air gap in between through the magnetic flux flows. The Rotor hosts the permanent magnets which when rotated generates the magnetic field. These magnets sits on top of free shaped air pockets in the design. The free shape of these air pockets are described geometrically by oblongs which draws the shape geometrically and the Double V magnets are aligned at a particular angle. The Stator comprises of the Stator poles which are 6 in number here. There are 4 slot windings for each stator slot in the tooth shoe, the slots are where the stator windings made up of copper reside at. The Stator yoke separates each tooth shoe from the outer radius of the EM.

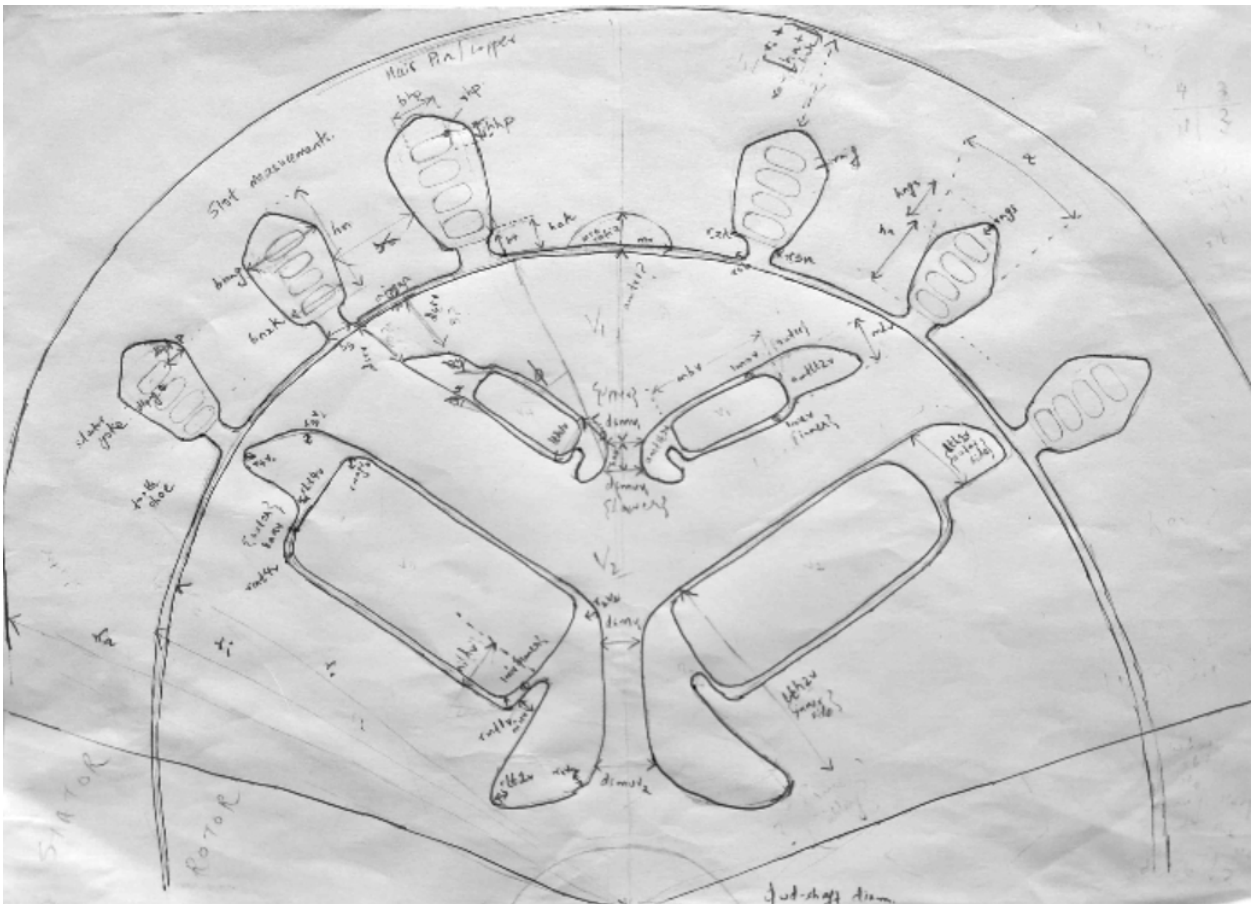


Figure 3.2: 1/8th cross-section of EM

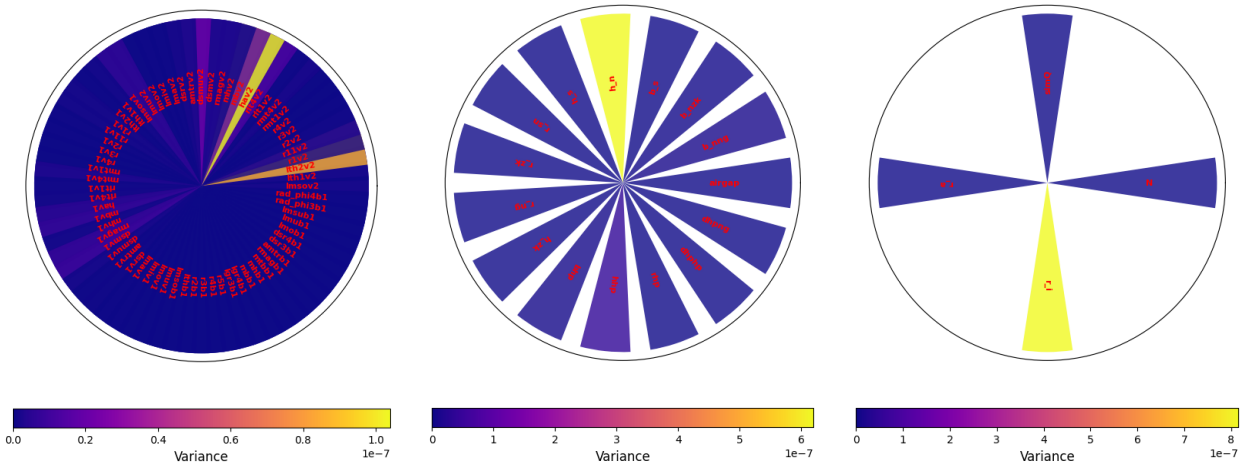
3.1 Data Preprocessing

For modelling the MLP, I represent the data in tabular form with the parameters corresponding to columns. Whilst exploring the parametric description of EMs, I have tried to include the 3 topologies in Section 3.1.1. However, I focus only on the Double V Magnet topology for the data exploration of the KPIs as well as the dataset I use throughout the thesis for training and evaluation due to lack of data for the remaining two topologies.
Nevertheless, I have made the architecture to be compatible for the 3 topologies however I cannot draw plausible inferences from the class imbalanced topologies.
In order to make the data compatible with the model, some level of data processing was carried out as elaborated below.

3.1.1 Data Exploration of Input Parameters

Fig. 3.3 illustrates the variance among the parameters on a color mapping scale for the different components of the EM. As is indicated by the scale, the variance is minimal to almost negligible with values ranging in the scale of 10^{-7}. This implicitly suggests that the tiny variations in values of parameters are not significant enough to be considered as different EM design. For the rotor parameters in Fig. 3.3a, the variance is reported the least with comparably 14 parameters showing a slight deviation when contrasted with the 70 parameters the rotor across the 3 topologies possess. Meanwhile for the stator parameters in Fig. 3.3b, the variance is at its most for 2 parameters in comparison to the 15 parameters it has. Lastly in Fig. 3.3c, the variance is at its peak for 1 parameter when compared to the other 3 parameters. The difference in

topologies are largely visible in its rotor components in terms of new parameters being introduced. However, no additional parameters crop up in the stator and general parameters with different topologies. In Appendix A.1.2, I provide the summary statistics of the input parameters of all the EM variants across all 3 topologies.



(a) Rotor Parameters (b) Stator Parameters (c) General Parameters

Figure 3.3: EM Parameter variance Analysis

All parameters including the additional ones in each topology are regarded as a separate columns and therefore if a particular column is topology dependent then the column corresponding to the missing data of this topology are treated as 0 values. Defaulting geometrical values as 0 seems to be the most reasonable approach in comparison to imputing them with the mean or median values. This is because the mean or median values are not representative of the actual data and could potentially mislead the model.

The values are then read and stored as their floating point equivalent to ensure data precision. Furthermore all degree columns are converted to their equivalent radian values as all trigonometric functions expects inputs to be in radian form and radian values are of a relatively narrow scale. I assumed this would be relevant given the variety of shapes in the EM design and the fact that they are 6 phase EM variants which implies phase shifted sinusoidal waves.

3.1.2 Data Exploration of Torque KPI

Fig. 3.4 shows the standard deviation of 10 random handpicked samples of the Torque KPI from the entire dataset. I strive to utilize the same set of 10 samples throughout the thesis to ensure effective comparison when drawing inferences. The x-axis represents the angular velocity of the motor ranging from 0 to 19100 rpm. Meanwhile, the y-axis represents the torque values corresponding to the angular velocity. For the dataset, the range of torque values are between 55 and 280 Nm. The mean of the samples of the dataset is displayed with an overlap of how its standard deviation is around the mean. Furthermore, I display a twin y-axis in red showing the standard deviation of the samples with the mean.

The following observations can be made from analysis of the Torque KPI:

1. **The Standard deviation is at its peak at low angular velocities**
   This is evident from the Standard deviation ranging up to 3 until 5000 rpm. Beyond which the standard deviation decreases drastically until saturation is visible with the plateauing of the curve.
2. **The curve to an extent resembles a Sigmoidal function mirrored along the x-axis**
   This finding is critical for how I modelled the loss regularization for the Torque KPI and will be further elaborated in Section 4.2.1.
3. **Samples are almost similar to one another**
   The samples shown here are almost resembling one another in shape and nature of curve. The finer

details are at the points of the curve where it makes its transitions. This observation is crucial and directly impacts my decision on choosing the Baseline model to be discussed in greater detail in Section 5.3.

4. **Some Samples Standard Deviation appears to follow the mean curve**
   These samples are among the few whose values lies on the outliers of the approximated distribution around the mean of all samples. This becomes more prominent on visualizing the results of the Torque curves and comparing the Root Mean Squared Error (RMSE) values in Sections 5.3, 5.2 among others.
5. **Samples have a smooth curve**
   From the shape of the Mean curve, it can be inferred that the samples have a smooth curve with no fluctuations. This finding is also taken into consideration in the modelling phase and will be discussed in Section 4.2 how we use it in teaching the model.

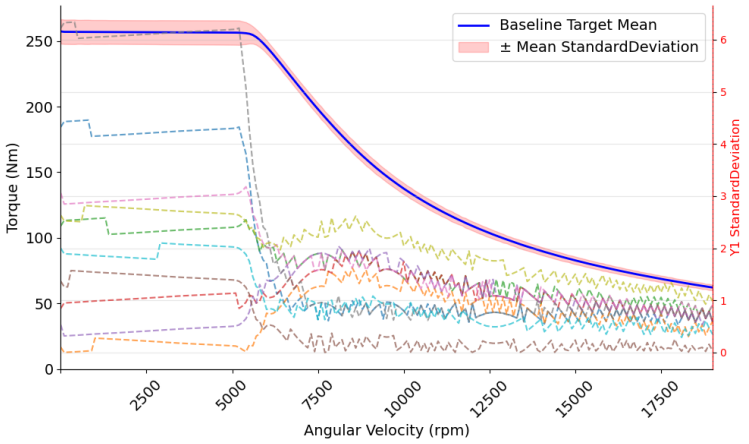


Figure 3.4: Standard Deviation of Torque KPI

3.1.3 Data Exploration of Efficiency KPI

As the target values of the Efficiency KPI are not provided with the same dimensions of the Torque range, I have an additional step which takes the maximum torque value from the Torque KPI and slices off the Efficiency grid to only range from [- Max($\mathcal{Y}_1$), Max($\mathcal{Y}_1$)]. Subsequently I choose only the rows of the MM grid introduced in Table A.1 which correspond to the indices of the sliced efficiency grid. This step ensures that I grant the model the correct dimensions of the Efficiency KPI based on its Torque KPI.

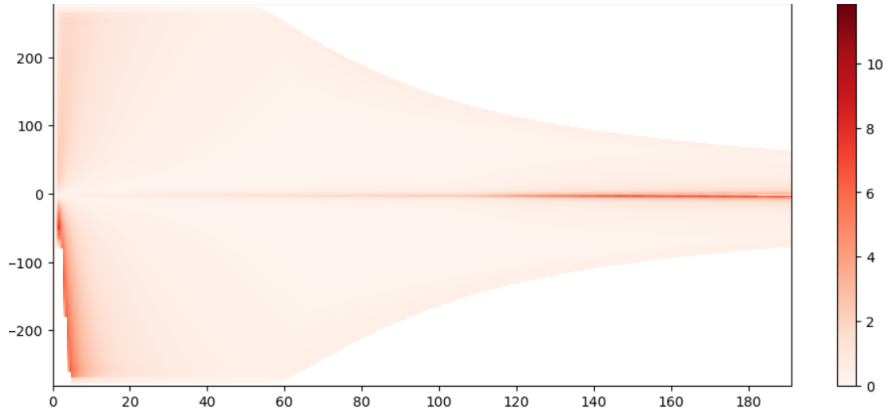


Figure 3.5: Standard Deviation of Mirrored Efficiency KPI

Fig. 3.5 and Fig. 3.6 both illustrate the standard deviation of the efficiency values displayed against its angular velocity considering all samples of the dataset. The standard deviation levels are shown as a scale

with colors darkening as the standard deviation increases.
Efficiency values for negative torque values correspond to when the EM is in generating mode and those of positive torque values to when the EM is in monitoring mode.

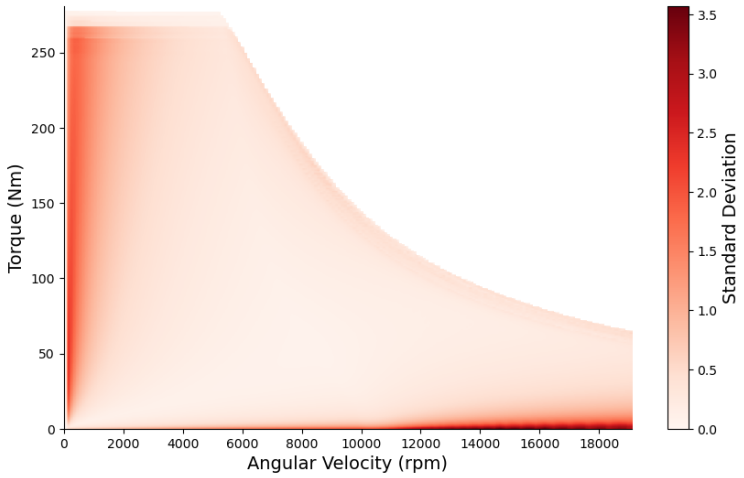


Figure 3.6: Standard Deviation of Efficiency KPI

Fig. 3.7 gives us a holistic view of how the efficiency values are distributed across equally spaced intervals of angular velocity. The image plots the standard deviation of all the Efficiency values for all samples of the dataset as an error bar at certain angular velocities over equally spaced intervals of 2000 rpm. I restrict the image to only show the Efficiency values from 2000 rpm onwards since at very low angular velocities falling in the range of 0 and 2000 rpm, the efficiency values vary drastically from 0% onwards.

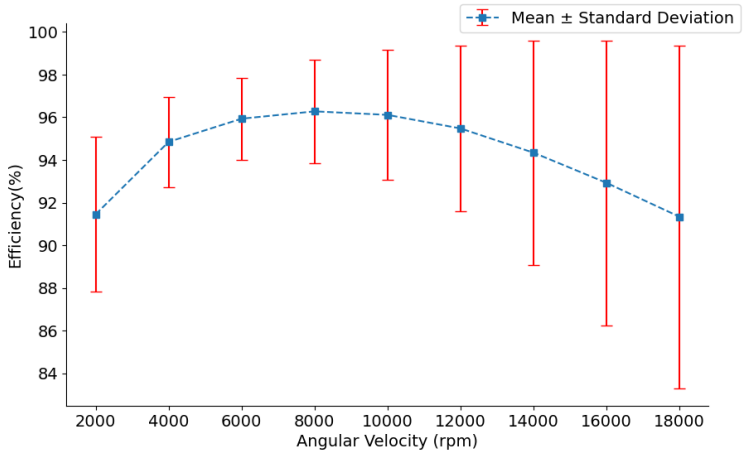


Figure 3.7: Standard Deviation of Efficiency KPI across Angular Velocity Intervals

The following observations can be made from analyzing the Efficiency KPI:

1. **Maximum deviation at low torques, extreme angular velocities**
   From Fig. 3.6, it can be observed that the deviation is at its peak at low torques, extreme angular velocities. In Fig. 3.7, the distribution shows maximum skewness towards extreme angular velocities and is relatively stable between 4000 and 6000 rpm. In Section 4.2.2, I discuss how I integrate this finding into teaching over model.
2. **Efficiency values beyond the Efficiency Envelope**
   I can also observe that beyond the Efficiency envelope, there are no efficiency values and can be

translated as blank values in the grid or Not a Number (NaN) values in the padded Efficiency matrix, I incorporate this information in Section 4.2.2.

3. **Considerable deviation along the Efficiency Envelope**
   In addition, skewness at the border of the curve within the grid can be seen in Fig. 3.6. In Section 4.5 how I handle this scenario is detailed out.
4. **Efficiency KPI envelope is dependent on its equivalent Torque KPI curve**
   As was noted already in Fig. 1.4 and Fig. 1.5, the Efficiency KPI envelope is completely dependent on its equivalent Torque KPI curve. The area beneath the boundary of which is looked into by the EM manufacturers to determine the car's efficiency in the drive range. This is yet another finding I use in Post Processing as is further elaborated in Section 4.5
5. **The Efficiency grid for all samples look almost alike**
   I make this assumption by noting that standard deviation at its maximal is at 3. This gives us leverage to decide on choosing the Baseline model which will be further discussed in Section 5.3.
6. **Efficiency values flawed in generating mode**
   In both monitoring and generating modes, the efficiency is almost similar albeit from Fig. 3.5, I note it is not the case for the dataset. This is evident from low angular velocity-high torque distribution area where a clear distinction can be noticed with NaN values. Since these are FEA simulations, it is probably an effect of a post processing step taken by the EM Design creator. This observation made us decide on dropping the negative Efficiency KPI and to focus on only predicting the positive Efficiency KPI.
   The latter can be mirrored to replicate the efficiency when it is in generating mode if necessary. However over the course of this thesis, I do not do so, as it is not relevant to evaluate a duplicate again.

Initially I had tried to set NaN values as an incredibly high value hoping the model would consider it as a default value for prediction instead of NaN. However, it resulted in poor predictions as the model must have been confused and tried to increase its spread of predictions to cover this large value and so all true values were also predicted to be close to this dummy value. Fortunately, I come up with a better way of handling this scenario which I elaborate in Section 4.2.2.
I also elaborate on technical details that was taken care of during development in Appendix **??**.

3.2 Scaling

Scaling is a common practice done before training a neural network. Standard scaling is the most prevalent scaling mechanism used for normalization as it results in a Gaussian Distribution centered around the mean. I have used the same for the input features to bring them to a common scale.

The Scaling is formulated mathematically as in Equation 3.1 :

$$\mathrm{z} = \frac{x - \mu}{\sigma} \tag{3.1}$$

where x is the Input, μ the Mean and σ the Standard Deviation.

For the Input features both Mean and Standard deviation are calculated across columns. This is attributed to the fact I have columns with different ranges for the input since I consider each feature a column mentioned in Section 3.1.1.

I decided against scaling the targets owing to below 2 reasons :

1. They do not enter the network architecture but are only used during loss calculation.
2. If I scale the target then I will have to scale each example from the train dataset and then average the calculated scaling parameters. It is not a good practice to do so as I will have lost a lot of originality in each example and is now introducing noise to new examples notably when they have a different data distribution.
3. For the Efficiency KPI I have loss regularization Equation 4.6 that has a constraint check on the maximum range of efficiency values that can be predicted. If I had scaled the target, the constraint check

would not be feasible to implement as the maximum value that the constraint takes would also have to be scaled with the same scaler for comparison.

During my experimentations where I initially scaled the targets, I observed then that the network required substantially less effort to learn and consequently lower learning rate and fewer epochs. Since this comes at a tradeoff of potentially losing precision, I continued with the original targets. Even if I were to scale the targets, although Standard Scaling seems to be the best approach as it approximates a gaussian centered around the mean. On the other hand, MinMax Scaler would have bound the data to be within the min and max of the data computed from the training dataset.

3.3 Dataset splitting

I have converted the data to be floating point tensors for better precision and collate them into a Tensor Dataset. I have also partitioned the dataset to have about 50 samples for test and the remaining is used for 5 fold cross validation with 80:20 split for training and validation. The reason I have a separate test dataset from the validation is to ensure that there is no data leakage as I do not want to overfit the test dataset with the hyperparameters I choose during training.

Across the 5 fold training runs, 4 sets would comprise of the training set and 1 of the test set which would be different for each fold run. Therefore, I expect to cover most grounds on training and have good monitoring of the model's performance for each fold.
Cross Validation also enables us to be able to monitor the network's overall stability and thus validate the model's generalization performance.
I have also used Data loaders to split the dataset into batches that fits into the Graphics Processing Unit (GPU) memory.

Chapter 4

Modelling and Evaluation

In this chapter, I discuss my modelling decisions of the neural network architecture, loss functions and evaluation metrics. I formulate the Torque KPI and Efficiency KPI mathematically as $\mathcal{Y}_1$ and $\mathcal{Y}_2$ respectively from the formal definition in Section 1.3.

4.1 Multi Layer Perceptron Model

For my multi-regression problem, I made use of a MLP model with input features corresponding to all the features in the tabular topology-invariant representation of the data. The model architecture is build to predict both the Torque KPI and Efficiency KPIs by having 2 separate output layers for each of the KPIs. Since the Torque KPI's targets are relatively learnable than that of the Efficiency KPI's targets I have experimented with fewer feed forward layers in the former than in the latter.
I have a hyperparameter to control the number of neurons in each hidden layer this can be tuned and is further discussed in Section 5.1.
Rectified Linear Unit (ReLU) layers are also added in between to serve as the activation function and produce non-linearities and consequently noise in the network.
Dropout layers ensure that not all neurons in each layer are used up during training to prevent the model from memorizing the data and hence overfitting. I have 2 hyperparameters to control the dropout rate at which I freeze the neurons when training also to be discussed in Table 5.1. I use dropout for the shared layers of the MLP and for the layers corresponding to Efficiency KPI.
Batch normalization layers are used to normalize the input from the ReLU activations applied on it and so mitigate internal co-variate shift to the next layer and hence speed up the training process.
Thus both batch normalization and dropout layers stabilize the network training.

Fig. 4.1 gives an outline on how the MLP Model architecture is designed and each component of the architecture is listed as follows :

1. **Input**
   The input layer takes in all features of the tabular data which is 89 in my case as scaled tensors.

2. **MLP Shared**
   The MLP Shared block is a sequential block comprising of 2 Linear Layers with the input features and neurons of each hidden layer to be a hyperparameter that needs to be tuned. I do not increase the number of neurons in the hidden layers within this block as it needs to be in the range of input features and output features (in this case Torque KPI).
   Furthermore I have Batch Normalization layers between each linear and ReLU activation function in addition to dropout layers. The dropout rate for the layers in this block is relatively higher with the intention that the model is encouraged to focus largely on learning the generality of the data. The 2 Linear Layers enable the network to learn a rich representation of the data during the initial feature extraction phase.

3. **MLP Torque**
   This block comprises of a Linear Layer with the output feature to be the size of the Torque KPI and a ReLU activation function. I have placed a ReLU activation function at the end of the output layer as the target values are inherently always positive values so as to exploit ReLU's behavior of clipping negative values to 0.

4. **MLP Efficiency**
   This block comprises of 3 Linear Layers with the output feature to be the size of the Efficiency KPI. Here, the neurons of the hidden layers can be increased as there is not as much limitation constraining the dimensionality of the output feature which is a large grid. This would enable the model to be more strong and grasp the complex patterns in the data better. There are batch normalization, dropout and ReLU activation functions between the 1st two Linear layers. The dropout rate for the layers in this block is relatively lower as there is a need to encourage the model to focus more on the specific nature of the grid towards the end. For the Last Linear layer, the dimensions of the output features corresponds to the target dimensions and a ReLU activation is again placed after it as the target values are inherently always positive values and it again encourages the model to adhere to this fact.

5. **Torque KPI**
   The number of output features correspond to the target size 191 which is infact the range of angular velocities from 0 to 19100 rpm at equal spaced intervals of 2000 rpm. Although the targets for the Torque KPI are an array of integer values, the float tensor and not the integer tensor are used to represent the data otherwise it would become a classification problem and not a regression problem as it should be.

6. **Efficiency KPI**
   The number of output features correspond to the target size which in my case is the shape of the padded collated array created in Section 3.1.3.

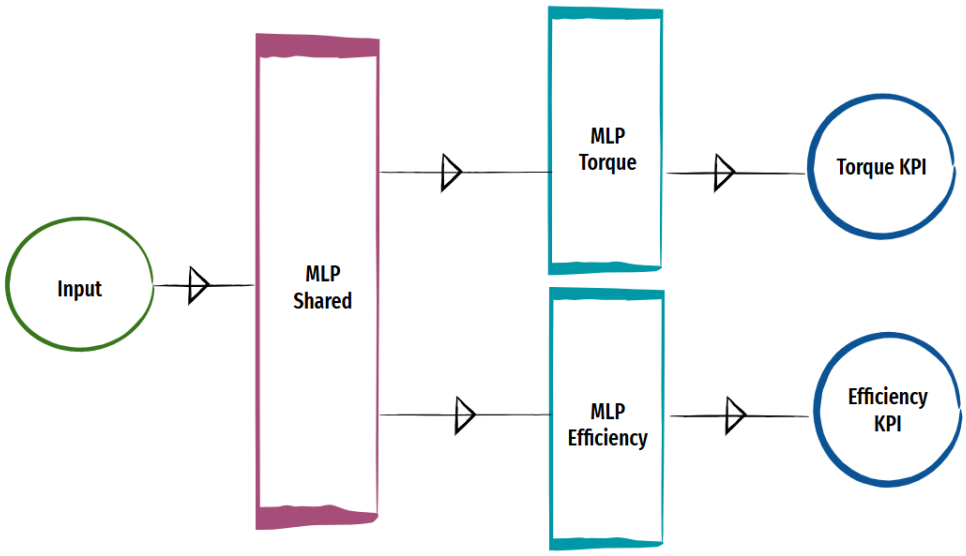


Figure 4.1: MLP Model Architecture

4.2 Loss Functions

The MSE loss is the error metric used for the problem with the intention that the squared losses penalizes the model and inturn encourage it to minimize the objective function even further. In addition to its contribution in exaggerating the loss by its square, MSE also ensures that deviations are positive and do not confuse the model by negating the losses of opposing signs.

4.2.1 Loss for Torque KPI

The MSE loss for the Torque KPI can be formulated mathematically as shown in Equation 4.1 :

$$\mathcal{Y}_1 \text{ Loss} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}\sum_{j=1}^{h}(y_{ij} - \hat{y}_{ij})^2, \tag{4.1}$$

where n is the number of EM samples, h is the columns of the 1D Torque vector, y_{ij} and $\hat{y}_{ij}$ are the ij-th ground truth and prediction respectively for the torque at each operating point.

To encourage the model to learn the nature of the curve, I have experimented with 2 Loss Regularization techniques based on the observations from Section 3.1.2. Both of which are L2 Regularization to be in sync with the dynamics of the MSE loss.

1. **Smoothening Curve Loss Regularization**
   To smoothen out the curve for the Torque KPI, a loss regularization factor is applied to ensure that the torque values at neighboring operating points are as close to each other as possible. This is formulated mathematically as shown in Equation 4.2 :

   $$\mathcal{Y}_1 \text{ Smoothening Curve Loss Regularization} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h-1}\sum_{j=1}^{h-1}\begin{cases}\left(\hat{y}_{ij} - \hat{y}_{i[j+1]}\right)^2 & \text{if } |\hat{y}_{ij} - \hat{y}_{i[j+1]}| > 1,\\ 0 & \text{otherwise.}\end{cases} \tag{4.2}$$

   It implies that if the difference in magnitude of consecutive values in the predicted array is greater than 1, then the loss is penalized by the square of the difference.

2. **Decreasing Curve Loss Regularization**
   based on one of our observation that the Torque curve closely resembles a mirrored sigmoidal curve, I use this knowledge to penalize the loss for increasing consecutive values within the prediction. This is intended to ensure that the torque values making up the Torque curve at each operating point is less than or equal to its prior operating point. It can be formulated mathematically as in Equation 4.3 :

   $$\mathcal{Y}_1 \text{ Declining Curve Loss Regularization} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h-1}\sum_{j=1}^{h-1}\left(\sigma(\hat{y}_{i[j+1]} - \hat{y}_{ij})\right)^2, \tag{4.3}$$

   where $\sigma(\cdot)$ is ReLU activation function.
   It takes into consideration the almost continuous decreasing nature of the curve as the regularization is such that a specific element in the array is less than or equal to its prior element.

I have not combined the above regularizations as they do not complement each other. This is because the loss regularized by Equation 4.2 will not necessarily be a decreasing curve. This holds true for the regularization in Equation 4.3 as it may not necessarily have gradual transitions in the curve. Nevertheless, I perform ablation studies with both the regularizations and report the results in Table 5.2.

4.2.2 Loss for Efficiency KPI

The MSE loss for the Efficiency KPI can be translated as in Equation 4.4 :

$$\mathcal{Y}_2 \text{ Loss} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{w}\frac{1}{h}\sum_{j=1}^{w}\sum_{k=1}^{h}\left(M_{ijk}\cdot y_{ijk}\right) - \left(M_{ijk}\cdot \hat{y}_{ijk}\right))^2, \tag{4.4}$$

$$M_{ijk} = \begin{cases}1 & \text{if } y_{ijk} \neq \text{NaN}\\ 0 & \text{if } y_{ijk} = \text{NaN}\end{cases}, \tag{4.5}$$

where M_{ijk} is Mask matrix, w are the rows of 2D vector and h the columns of 2D vector.
Based on one of the observation in Section 3.1.3 wherein the efficiency values beyond the Efficiency envelope

have blank values, which I further modified to be NaN values, I construct a binary mask matrix to ignore them in the loss calculation. Since, the network cannot be trained to predict NaN values, the binary mask is constructed such that values corresponding to NaN in the target have value 0 and all other values as 1. Mathematically, this process can be expressed as shown in Equation 4.5 and thus ensure that the NaN values are ignored in the loss calculation. The mask is then multiplied with both the target and its respective prediction.

I have also modelled 2 Loss Regularization techniques to encourage the model to learn the nature of the Efficiency KPI curve.

1. **Maximum Efficiency Loss Regularization**
   To ensure that the efficiency values do not exceed 100, I formulate the loss function mathematically as expressed in Equation 4.6 :

$$\mathcal{Y}_2 \text{ Maximum Efficiency Regularization} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{w}\frac{1}{h}\sum_{j=1}^{w}\sum_{k=1}^{h}\left(\sigma(|\hat{y}_{ijk}| - 100)\right)^2 \tag{4.6}$$

   When the efficiency values of the prediction exceed 100, the overall loss is penalized by squared magnitude of the difference of the prediction from its ground truth. ReLU once again assists to clips the difference if it is negative which is the scenario when the efficiency values are less than or equal to 100 when a violation is not warranted.
   Needless to say the efficiency values are percentage values and can only take up values in the range of 0-100%. I refrain from instructing the model to not have values less than 0 since I had masked NaN values as 0 and the model will surely attempt to predict values close to 0. Moreover, these predictions are not relevant for us as after generating all predictions I finally slice off the Efficiency grid to be of the shape of the Torque curve which implies that the values predicted in place of NaN are irrelevant. This is discussed more elaborately in Section 4.5. Therefore, I do not see the need to needlessly punish the model for making mistakes for values I eventually do not use since pessimistic decisions could discourage the model from realistic learning and thus affect its focus on predicting the other values in the Efficiency grid correctly.

2. **Efficiency Grid Loss Regularization**
   Additionally, to encourage the model to learn the nature of the Efficiency KPI from the observations gathered in Section 3.1.3, I have tried to incorporate all of the following learnings into the loss function as $\mathcal{Y}_2$ Regularization techniques.

   (a) **Efficiency at Maximum Torque Loss Regularization**
   Since I cannot regularize the loss to have the decreasing envelope of the Torque curve, I only focus on what I can effectively teach the model which herein is the stable portion of the Torque curve typically until Angular Velocity of 5000 rpm. This is derived from an observation I make on the shape of the Torque curve in Section 3.1.2. To ensure that the shape of the Efficiency KPI is maintained, the loss is regularized for the maximum torque value. To do so, I have attempted to retrieve the last rows of the Efficiency KPI and those of its target values and penalize the squared difference to have higher weight. It is formulated mathematically as described in Equation 4.7 :

$$\mathcal{Y}_2 \text{ Loss Regularization Max Torque} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t_1}\sum_{j=-t_1}^{w}\frac{1}{h}\sum_{k=1}^{h}(y_{ijk} - \hat{y}_{ijk})^2, \tag{4.7}$$

   where t_1 is the threshold for initial Efficiency KPI Envelope boundary. The number of last rows is determined by a threshold t_1.

   (b) **Efficiency at Low Angular Velocity Loss Regularization**
   At extreme angular velocities, there exists a greater deviation in the efficiency values particularly towards higher angular velocities. This is because there are fewer efficiency values as angular velocities increases beyond a range since not all torque values participate. To force the model to

pay more attention at lower angular velocities, the loss is regularized for the first few columns of each row of the predicted Efficiency grid by penalizing the squared difference with that of its target. It can be formulated mathematically as shown in Equation 4.8 :

$$\mathcal{Y}_2 \text{ Loss Regularization Low Angular Velocity} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{w}\sum_{j=1}^{w}\frac{1}{t_2}\sum_{k=1}^{t_2}(y_{ijk}-\hat{y}_{ijk})^2, \tag{4.8}$$

where t_2 is the Threshold for Low Angular Velocities. The number of first columns is determined t_2.

(c) **Efficiency at Low Torque Loss Regularization**
It is a known fact that at Torque 0 Nm, the corresponding efficiency values for the motor is 0%. Consequently the efficiency values close to this torque will be low as well. To force the model to be more careful at low torque, the loss is regularized for the first few rows of each column of the Efficiency KPI by penalizing the squared difference with that of the target. It can be formulated mathematically as shown in Equation 4.9 :

$$\mathcal{Y}_2 \text{ Loss Regularization Low Torque} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t_3}\sum_{j=1}^{t_3}\frac{1}{h}\sum_{k=1}^{h}(y_{ijk}-\hat{y}_{ijk})^2, \tag{4.9}$$

where t_3 is the Threshold for Low Torque. The number of first rows is determined by t_3.

The above $\mathcal{Y}_2$ Regularizations are indeed purely MSE but with higher weights for specific regions of the Efficiency KPI. Consequently being L2 Regularizations it goes hand in hand with MSE Loss calculated in Equation 4.4.
The thresholds t_1,t_2 and t_3 are tunable hyperparameters that can be optimized on observing the relevant plots of the Efficiency KPIs not adhering to the expected look of the Efficiency grid. The intention of including the threshold values is to give the model the flexibility to search the relevant portion of the Efficiency grid determined by the threshold and push the model to get those regions right by penalizing them more.

I aggregate all the above regularizations to form the $\mathcal{Y}_2$ Loss Regularization as described in Equation 4.10 :

$$\begin{aligned}\mathcal{Y}_2 \text{ Efficiency Grid Loss Regularization} = \mathcal{Y}_2 \text{ Loss Regularization Max Torque}\\ +\mathcal{Y}_2 \text{ Loss Regularization Low Angular Velocity} + \mathcal{Y}_2 \text{ Loss Regularization Low Torque}\end{aligned} \tag{4.10}$$

The Total Loss is calculated in Equation 4.11 :

$$\begin{aligned}\text{Total Loss} = \text{wt} \cdot (\mathcal{Y}_1 \text{ Loss} + (\lambda_{1y1} \cdot \mathcal{Y}_1 \text{ Smoothening Curve Loss Regularization}) + (\lambda_{2y1} \cdot\\ \mathcal{Y}_1 \text{ Declining Curve Loss Regularization })) + (\text{1-wt}) \cdot ((\mathcal{Y}_2 \text{ Loss} + (\lambda_{y21} \cdot\\ \mathcal{Y}_2 \text{ Maximum Efficiency Loss Regularization}) + (\lambda_{y22} \cdot\\ \mathcal{Y}_2 \text{ Efficiency Grid Loss Regularization}))),\end{aligned} \tag{4.11}$$

where λ_{1y1} is $\mathcal{Y}_1$ Smoothening Curve Loss Regularization Parameter, λ_{2y1} is $\mathcal{Y}_1$ Declining Curve Loss Regularization Parameter, λ_{y21} is $\mathcal{Y}_2$ Maximum Efficiency Loss Regularization Parameter, λ_{y21} is $\mathcal{Y}_2$ Efficiency Grid Loss Regularization Parameter, wt is $\mathcal{Y}_1$ Loss Weightage, $1-wt$ is $\mathcal{Y}_2$ Loss Weightage.

The Weightage parameter is utilized for the following 2 reasons:
1. When the targets are not of the same scale.
   Without scaling, the losses for both targets Torque and Efficiency KPIs being of different ranges are drastically different. I circumvent this by weighing up the loss of the target not performing better on validation dataset and weighing down the loss of the targets by an approximation of how much its value range varies.

2. When the prediction accuracy of one KPI is substantially more vital than the other.
   The task demands the same as the Efficiency KPI is post processed to be within the shape of the Torque KPI.

Therefore, in theory have higher weightage for the Torque KPI as its loss in performing well is costlier but because its value range is relatively higher I make the decisions from monitoring the predictions performance. I reflect on these decisions based on whether the envelope of the Efficiency KPI grid is more valuable than the efficiency values within it.
Ultimately I decided on prioritizing the efficiency values to counter the Torque KPI's loss dominating the total loss as I had implemented quite a few regularization techniques to the Efficiency KPI loss.
Furthermore, the weightage parameters for both target is designed to sum upto 1 keeping in mind improved training stability as a result of normalized weights.

Finally the aggregated loss is backpropagated.

4.3 Optimizer

Adam optimizer is used for optimization as it is known to be computationally efficient and requires less memory. It infers the gradients of the loss and how it impacts the weights and biases of each layer and thus controls learning by guiding the model to decrease the loss over the course of training. The optimizer acts once the loss is backpropagated across training each batch of the dataset. The optimizer also uses the learning rate to control the step size with which the model parameters are updated.

4.4 Evaluation Metrics

The evaluation metrics regarded for the regression problem is the average of the RMSE. Therefore, the model with the least prediction scores such that it is closest to 0 is the ideal score. Although, RMSE is used as the metric for both evaluation and loss functions, the loss regularizations play a vital role on the loss functions on top of RMSE.
Needless to mention, I have the Masking mechanism as per Equation 4.5 for the Efficiency KPI which does not exists for the evaluation function of the same KPI, the latter in a way thus shows a pessimistic view of the model's performance. This is particularly prominent during training as I do not have visibility of the Efficiency envelope until its Torque KPI values are generated. However, during inference I make use of not NaN differencing to mask out values beyond the Efficiency envelope.

4.4.1 Evaluation Metrics for Torque KPI

The $\mathcal{Y}_1$ Score for the Torque KPI is formulated in Equation 4.12 :

$$\mathcal{Y}_1 \text{ score} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\sqrt{\frac{1}{h}\sum_{j=1}^{h}(y_{ij} - \hat{y}_{ij})^2}}_{\mathcal{Y}_1 \ RMSE}, \tag{4.12}$$

where h is the columns of 1D vector and $\mathcal{Y}_1$ RMSE is RMSE for each test sample.

4.4.2 Evaluation Metrics for Efficiency KPI

The $\mathcal{Y}_2$ score for the Efficiency KPI is formulated in Equation 4.13 :

$$\mathcal{Y}_2 \text{ score} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\sqrt{\frac{1}{w}\frac{1}{h}\sum_{j=1}^{w}\sum_{k=1}^{h}(y_{ijk} - \hat{y}_{ijk})^2}}_{\mathcal{Y}_2 \ RMSE}, \tag{4.13}$$

where w is the rows of 2D vector, h is columns of 2D vector and $\mathcal{Y}_2$ RMSE is the RMSE for each test sample.

The Aggregated $\mathcal{Y}$ score for KPIs is formulated in Equation 4.14 :

$$\text{Aggregated } \mathcal{Y} \text{ score} = \text{wt} \cdot (\mathcal{Y}_1 \text{ score}) + (1 - \text{wt}) \cdot (\mathcal{Y}_2 \text{ score}), \tag{4.14}$$

where wt and $1 - wt$ are the weightage assigned to its $\mathcal{Y}_1$ score and $\mathcal{Y}_2$ score respectively. It is the same weightage assigned to aggregate the losses of both the KPIs in Equation 4.11.

As the value ranges for both the KPIs are so starkly different, I give a glimpse of the corresponding percentage differences and have narrowed down scoring to follow the criteria recorded in Table 4.1.

% Difference	**0-5%**	**5-10%**	**10-15%**	**15-20%**	**20-25%**	**25-30%**	**30-35%**	**35-40%**	**40-100%**
$\mathcal{Y}_1$ Score	0-11	11-22	22-33	33-44	44-55	55-66	66-77	77-88	>88
$\mathcal{Y}_2$ Score	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	>40

Table 4.1: Scoring Criteria

This is deduced from the Equations 4.15 and 4.16.

$$\mathcal{Y}_1 \text{ Percentage Difference} = (\mathcal{Y}_1 \text{ Score}/(Max(\mathcal{Y}_1) - Min(\mathcal{Y}_1))) \cdot 100 \tag{4.15}$$

$$\mathcal{Y}_2 \text{ Percentage Difference} = (\mathcal{Y}_2 \text{ Score}/(Max(\mathcal{Y}_2) - Min(\mathcal{Y}_2))) \cdot 100 \tag{4.16}$$

From my observations of the dataset, the target values for Torque KPI range between 50-280 and those of the Efficiency KPI range between 0-100. The scoring of the $\mathcal{Y}_2$ at inference is slightly different from that of training in the sense that for inference, I have the Torque KPI to accurately slice of the envelope from the Efficiency KPI. However, during training I do not have the Torque KPI generated yet to slice the Efficiency KPI.

4.5 Post Processing

The mean and standard deviation from the train-validation datasets are applied to transform the test dataset to maintain uniformity in the predictions generated. In the case of new files not part of the segregated test dataset, I first convert it into the tabular representation the model consumes and then apply the scaling. This is why I preserve the same scalers used during training as I not only evaluate the dedicated test dataset but also new files for clients to use on demand.

Furthermore, as I am predicting a padded matrix to ensure dimensionality sync across different Efficiency KPIs, the grid contains values even outside the boundary of the Efficiency KPI. First, I attempt to slice of the grid to ignore 0 values for all rows excluding the 1st row as per the Equation 4.9 in the hopes that the Mask constructed from Equation 4.5 would force the model to learn 0s. However it tried to approximate values close to 0 and was almost never 0.
Therefore, I decided to slice the shape of the Torque KPI curve from the Efficiency KPI by counting the number of columns a row can have based on consecutive values in the curve. This brings us back to the point that it is imperative that the prediction of the Torque KPI is close to perfect since the envelope of the Efficiency KPI is inherently dependent on it. However if in the worst case scenario where the $\mathcal{Y}_1$ and $\mathcal{Y}_2$ predictions are so starkly different that Efficiency KPI cannot be sliced off from the Torque KPI because the former is smaller than the latter then I decide not to slice the Efficiency KPI and instead use the predictions as is. I also assert the end user with a warning when this scenario is encountered.

Chapter 5

Experiments and Results

5.1 Experiments

After optimizing over an exhaustive random search of the hyperparameter space, I present the chosen hyperparameters in Table 5.1 in addition to the range of values I searched from by monitoring the model's performance across 5 fold cross validation. I use different splits for each round of training and tune the hyperparameters on the validation set. The final splits are saved locally and can be used later to ensure reproducibility.

Hyperparameters	**Value**	**Value Ranges**
Learning Rate (*lr*)	0.075	0.025 - 0.25
Dimensionality of Hidden Layers (*hidden size*)	128	64, 128
Exponential Learning Rate Scheduler Gamma (*lr gamma*)	0.9	0.5-0.9
Batch Size (*batch size*)	72	32-72
Number of Epochs (*epochs*)	10	6-10
Dropout Probability for Shared Layers (p_{y1})	0.35	0.4-0.2
Dropout Probability for Efficiency Layers (p_{y2})	0.2	0.2-0.1
$\mathcal{Y}_1$ Smoothening Curve Loss Regularizer (λ_{1y1})	0.5	0-0.75
$\mathcal{Y}_1$ Decreasing Curve Loss Regularizer (λ_{2y1})	0.5	0-0.75
$\mathcal{Y}_2$ Maximum Efficiency Value Regularizer (λ_{y21})	3	0-5
$\mathcal{Y}_2$ Efficiency Grid Loss Regularizer (λ_{y22})	5	0-5
$\mathcal{Y}_2$ Initial Envelope Boundary Threshold (t_1)	5	0-5
$\mathcal{Y}_2$ Low Speed Threshold (t_2)	20	0-30
$\mathcal{Y}_2$ Low Torque Threshold (t_3)	20	0-30
Weightage of $\mathcal{Y}_1$ Loss (*wt*)	0.05	0-1
Weightage of $\mathcal{Y}_2$ Loss (*1-wt*)	0.95	0-1

Table 5.1: Hyperparameter Tuning

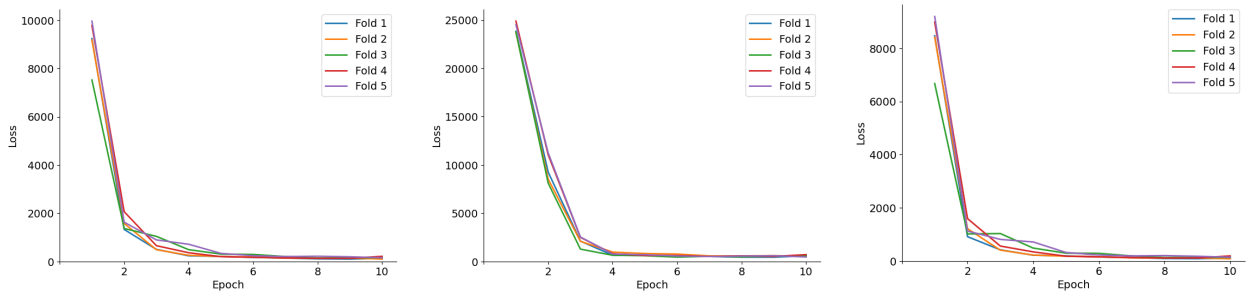
I am using an exponential learning rate scheduler which reduces the learning rate exponentially by the *lr gamma* to decay learning as training progresses across epochs. This is to ensure that the model does not overshoot after few epochs of training.
I also would like to point out that the 2 outputs typically would thrive on different learning rates and might be a better approach to have separate learning rates in addition to separate loss functions.
The *batch size* is limited to the capacity of the GPU memory. I use the maximum batch size to train faster after which I hit the memory roof of the GPU.
The *hidden size* refers to the number of neurons in the hidden layers of the MLP model. I have experimented with just 2 hidden sizes as I am constrained by the fact that the number of neurons must be between the range of input features and output features which was discussed in Section 4.1. In addition, it has to be of the multiples of 8 as GPUs are most optimized for the same.

Dropout rate during training is controlled by the parameter p_{y1} for shared layers and p_{y2} for layers specific to Efficiency KPI. This is discussed in Section 4.1. I am not so much concerned of the data being dropped in larger extent for the Efficiency KPI than anticipated for the Torque KPI because the latter is relatively simpler to learn and the value range is larger leading to its loss dominating the total loss.
λ_{1y1}, λ_{2y1}, λ_{y21} and λ_{y22} parameters control the regularization weight for the regularization terms for the Torque KPI and Efficiency KPI respectively detailed in Equation 4.11. The thresholds t_1, t_2 and t_3 are used to control the number of rows and or columns to be considered for the loss regularization terms to learn the nature of the Efficiency KPI as discussed in Section 4.2.2. The weightage parameter to control the direction of loss is denoted by wt and is also discussed in Equation 4.11.
I also observe that a simple architecture of the model is not a significant downside for limited data set as it seems to learn the patterns within the data especially since I supplement the loss regularization with more knowledge.

5.1.1 Monitoring MLP with Efficiency Grid Regularization

We monitor the MLP model which only uses the loss regularization parameters λ_{y21} and λ_{y22} and it does not consider the loss regularization for the Torque KPIs whose performance I discuss in Sections 5.6.
I have chosen a 3rd party application `Wandb`[1] to log metrics from the training run and to monitor model performance across the 5 folds. Training and validation occurs 5 times on different combinations of cross validation splits independent per fold.

Fig. 5.1 illustrates the loss at work as training progresses across epochs for the 5 folds. Fig. 5.1b and Fig. 5.1c illustrates monitoring of how the losses for both the KPIs vary. As was discussed in Equation 4.1 and Equation 4.4, both the MSE losses also include their respective loss regularization terms weighted by the tunable regularization coefficients. Fig. 5.1a depicts the weighted sum of the losses discussed in Equation 4.11. The Training loss for the Torque KPI starts at an all-time high of 25000 and over 2 epochs seems to have converged to value close to 500. Meanwhile, the Training loss for the Efficiency KPI starts at 9000 and converges to a value close to 100 by 5 epochs.
The Validations plots for losses also tells us the same tale although the Efficiency KPI stops to learn after a certain point and can be monitored in Fig. A.2.



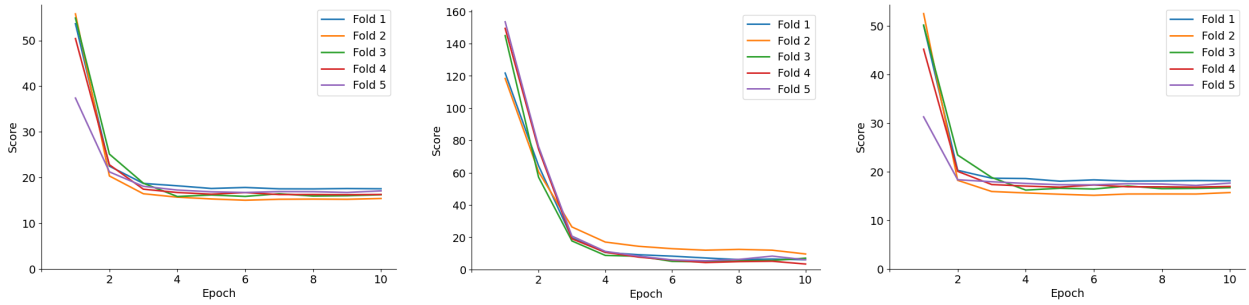
(a) Aggregated (b) Torque KPI (c) Efficiency KPI

Figure 5.1: Training Loss Metrics

Fig. 5.2 illustrates the evaluation metrics changing across each epoch of the validation dataset. This is important to us to know the model's performance on unseen data for the 5 folds. Fig. 5.2a shows the aggregated validation score which is again the weighted sum of the scores as was formulated in Equation 4.14 for both the KPIs. Meanwhile, Fig. 5.2b and Fig. 5.2c logs monitoring of the evaluation metrics as per the Equation 4.12 and Equation 4.13. The Torque KPI evaluation metrics start at a score of 160 and converge to around 7 by the 7th epoch. Meanwhile, th Efficiency KPI evaluation metrics starts off at 50 and converge by a score close to 15 by the 4th epoch. The aggregated score starts of at almost similar fashion

[1]Weights & Biases - `https://wandb.ai/`

as the Efficiency KPI.
Likewise the training plots evaluation metrics are displayed in Fig. A.1.



(a) Aggregated (b) Torque KPI (c) Efficiency KPI

Figure 5.2: Validation Evaluation Metrics

The following are the observations I infer from monitoring the training and validation plots:

1. The difference in the both losses and scores at which training and validation begins for both KPIs can be attributed to larger value range for Torque KPI (n values).
2. Although the loss regularization increases the loss range for Efficiency KPI, it is not so much pronounced due to its relatively smaller value range (n^2 values being as matrix).
3. As the loss regularization does not come into play for the evaluation metrics, the value range between the losses and the evaluation are not equivalent. The other factor which also contributes significantly as cause would be the contrasting train-validation split percentages.
4. Evident from both the aggregated loss and scores, the Efficiency KPI's range is more prominent which can be attributed to the weightage parameter $1 - wt$.
5. The model has converged after having run for 10 epochs with a total run time of 2 minutes wherein each fold took around 20 seconds.
6. Even though the weightage parameter $1 - wt$ assigned to the Efficiency KPI is significantly more in addition to loss regularization parameter, it seems to have stopped learning beyond a threshold and so I hypothesize to further increase $1 - wt$ of the Efficiency KPI.

The training loss and validation evaluation plots are most important to us to be able to judge the hyperparameters to optimize. Therefore, the other plots are moved to Appendix A.3 for further reflection. I have also enabled saving the best performant fold locally so that it can be loaded on demand by the client when in need to only run inference.

5.2 Results with MLP Efficiency KPI Regularization

The model we have used here is the MLP model with the Efficiency KPI loss regularization terms.

5.2.1 Torque KPI Results

Fig. 5.3 illustrates 6 samples from the test dataset for the predicted Torque values and their corresponding ground truths displayed against their angular velocities. The difference and percentage difference between the prediction and ground truth are also displayed on a twin y axis to give a rough overview of the prediction deviations from the targets. As the percentage difference formulated in Equation 4.15 would always result in positive values, they appear above the 0 line dotted separator.

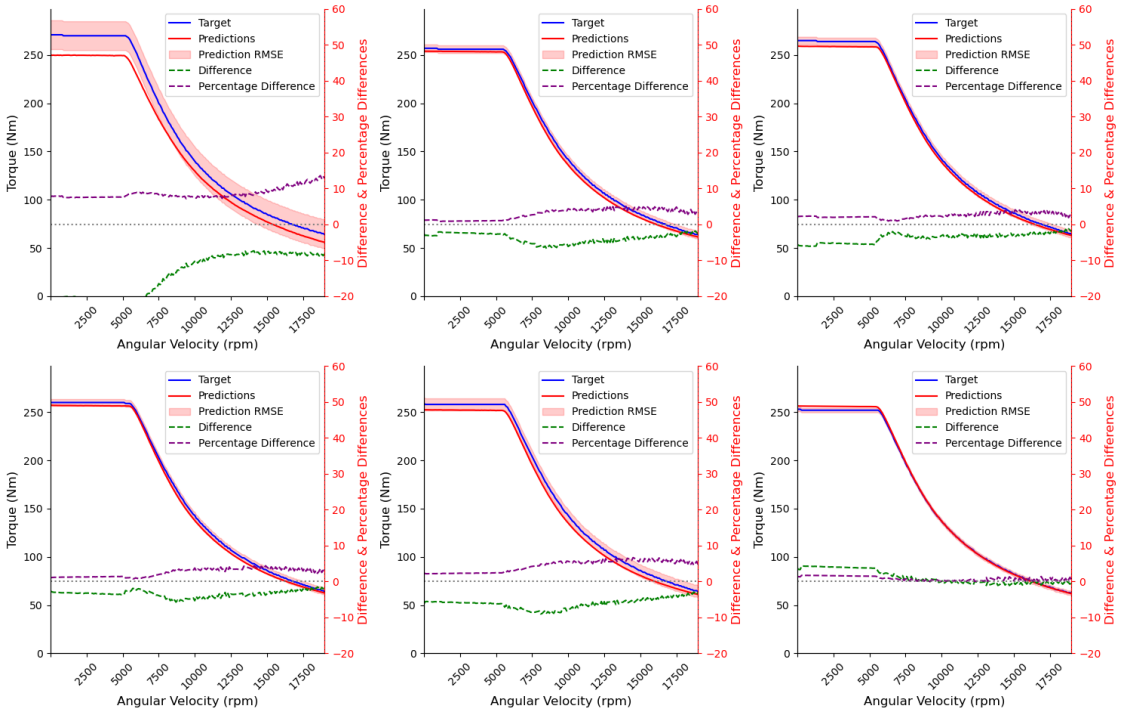


Figure 5.3: MLP Predictions for Torque KPI

Fig. 5.4 displays the mean, average RMSE and element wise RMSE for the 10 handpicked samples from the test dataset with its predictions from the model. The angular velocity of the motor is displayed against its torque values. The mean of the samples of the dataset is displayed with an overlap of how its RMSE is around the mean. Furthermore, I display a twin y-axis in red showing the RMSE with that of the ground truth.

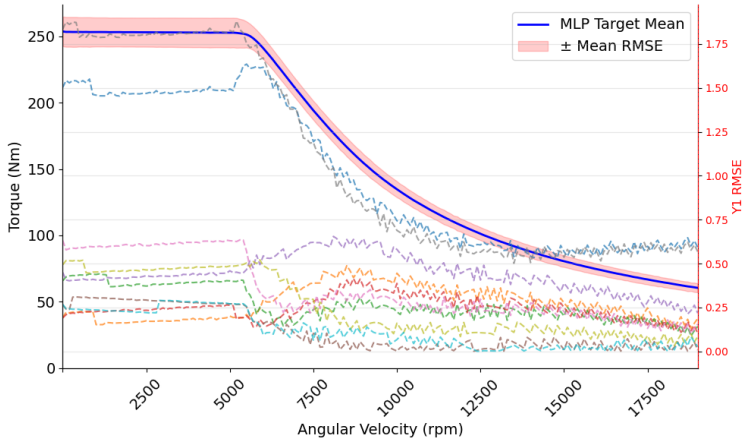
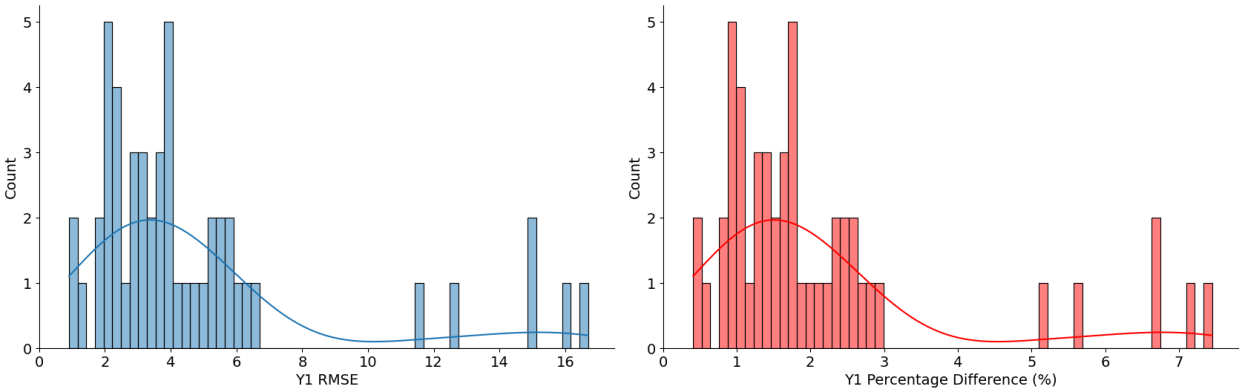


Figure 5.4: MLP RMSE Evaluation for Torque KPI

Fig. 5.5 shows the score statistics of the model performance of Torque KPI over all 50 samples from the test dataset with the number of test dataset samples on the y-axis. The $\mathcal{Y}_1$ RMSE from Equation 4.12 is calculated for each sample and shown as a histogram in Fig. 5.5a. In addition, the $\mathcal{Y}_1$ Percentage Difference from Equation 4.15 is also shown as a histogram in Fig. 5.5. The latter is essential to be able to relate to its effect on total range of deviation due to its differing value range. The histogram looks the same in shape for both the figures however the finer details lies within the measure on the x-axis.



(a) $\mathcal{Y}_1$ RMSE

(b) $\mathcal{Y}_1$ Percentage Difference

Figure 5.5: $\mathcal{Y}_1$ Evaluation Statistics of MLP

Fig. 5.6 gives a good view of how the torque curve for each folds prediction represented in dotted lines deviate from the mean of all the folds predicted torque curve. I also take care to note that I have used only 1 sample from the test dataset for inference across all folds.

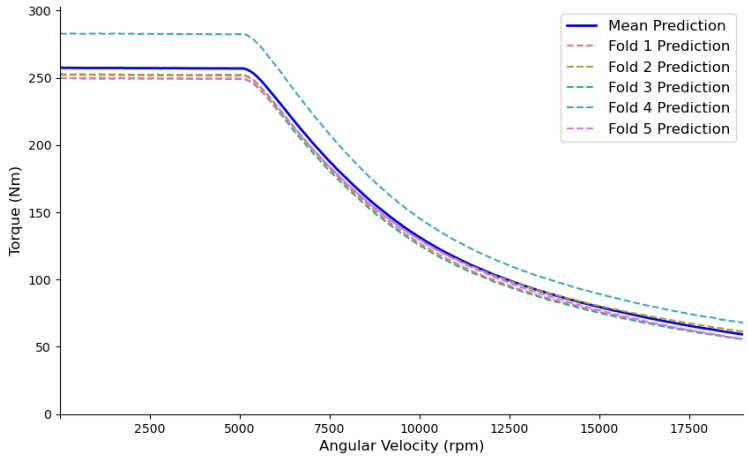


Figure 5.6: MLP Training Deviation across Folds for Torque KPI

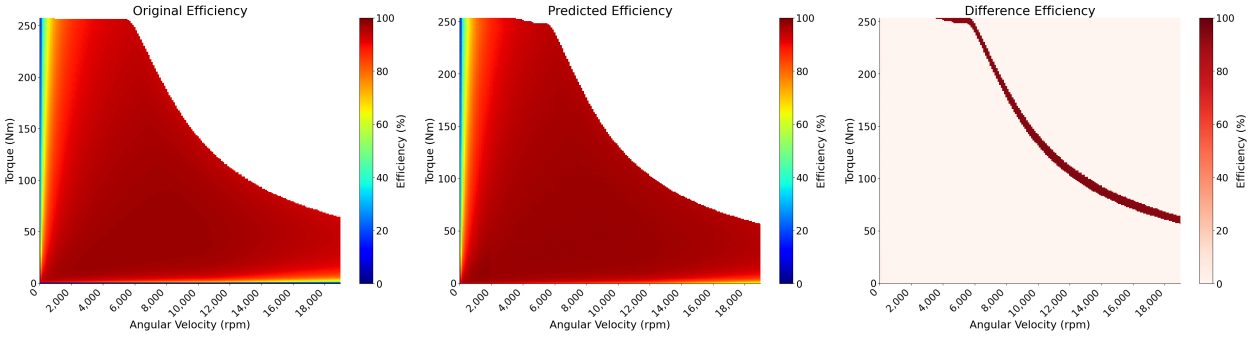
The following are the observations I infer of the Torque KPI from the model:

1. In Fig. 5.3, even though deviations can be seen it can be noticed for few of the cases the predictions are such that the torque values predicted have smaller values than its respective ground truths. However, most of the examples show very close correlation between prediction and ground truth and differences in addition to percentage differences on average do not exceed 5%.
2. In Fig. 5.4, the RMSE equivalent to the $\mathcal{Y}_1$ score tells us that for 80% of the plots shown, the predictions are off by less than 5% from the target values by cross-referencing Table 4.1. Only for 2 samples a greater deviation can be noticed however these are the samples discussed in Section 3.1.2 as outliers from the mean distribution.
3. The predicted Torque curves closely resemble the trajectory of the target values although they fluctuate. Experimenting with the hyperparameters λ_{1y1} and λ_{2y1} has potential to improve this anomaly. Subsequently, granting a higher weight wt for $\mathcal{Y}_1$ can also help in this direction.
4. It can be deduced from Fig. 5.11 that the $\mathcal{Y}_1$ scores are approximately gaussian distributed between 0.5-4 with outliers for 5 samples upto 7.5. Thus encompassing a range of 0-4% percentage difference with the target values as derived from the Table 4.1.

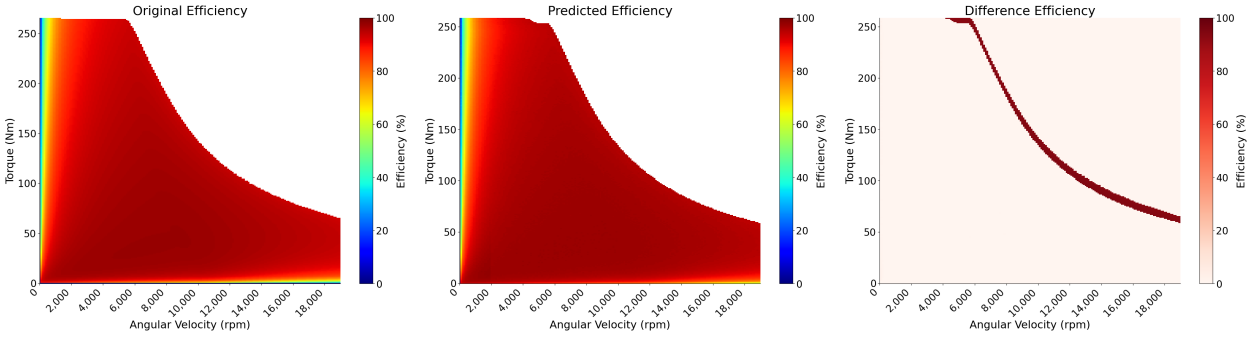
5. It can be remarked from Fig. 5.12 that the predicted torque curves for all the folds except Fold 4 is as close as can be to each other. Although the torque values predicted by Fold 4 are relatively larger values, it can be observed that the shape of the curve is approximated correctly in all folds. This image is a good indicator of the model's generalization capability and also sheds light to the model's stability.
6. The Torque KPI shows promise in learning better but it would be at the cost of overfitting the Efficiency KPI.

A noteworthy point on the implementation is that I have left the output predictions for the Torque curve to remain as floating point values even when the target values are integers to preserve data precision during my evaluations. The client has the flexibility to turn this on or off demand.

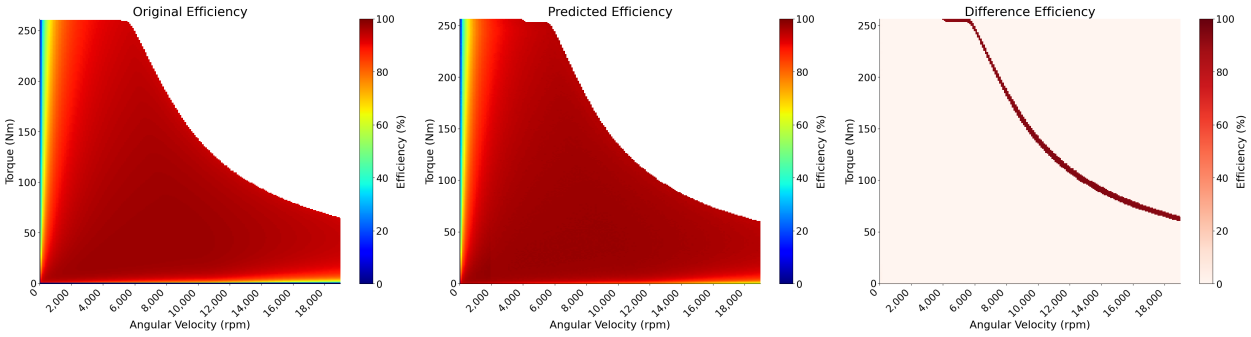
5.2.2 Efficiency KPI Results



(a) 1st Difference Sample



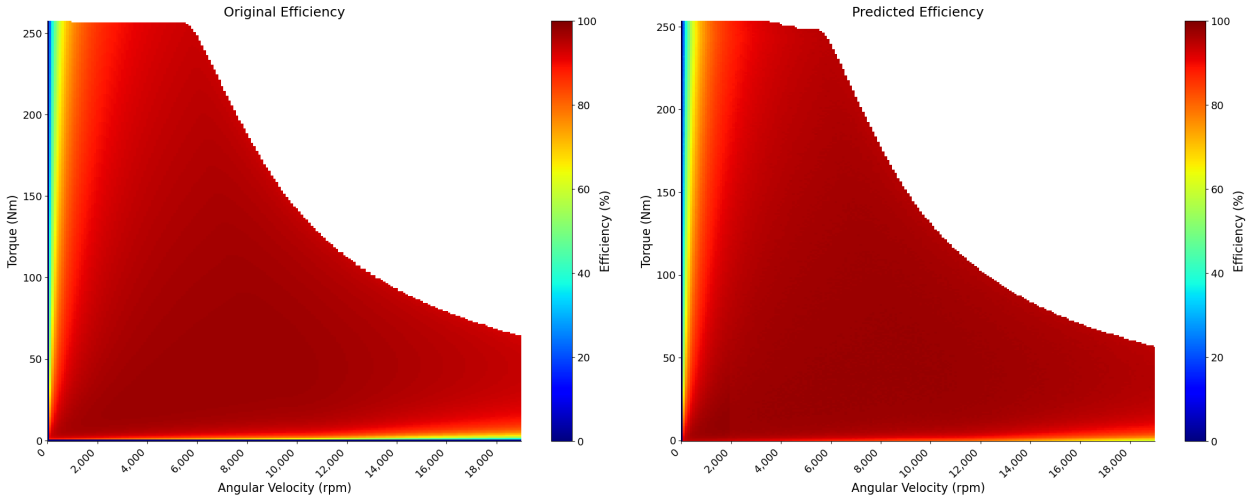
(b) 2nd Difference Sample



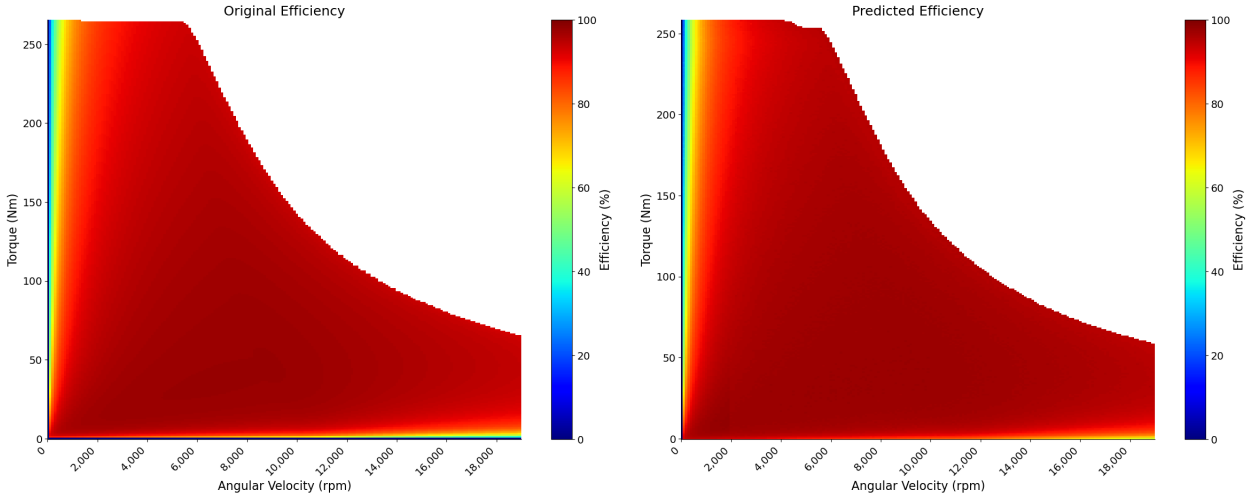
(c) 3rd Difference Sample

Figure 5.7: Efficiency KPI Predictions Vs Targets Vs Difference with MLP

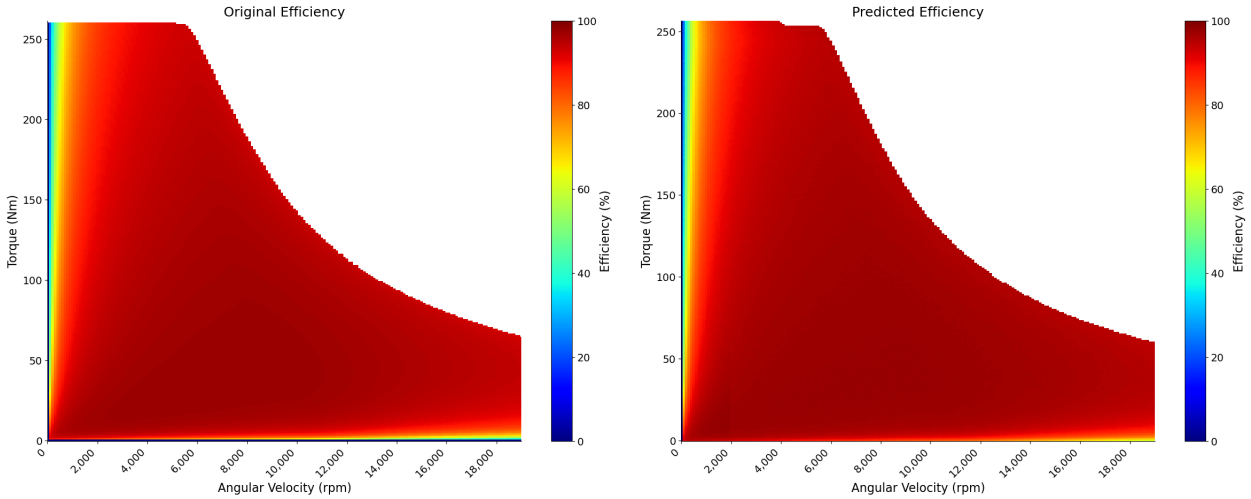
,



(a) 1st Sample



(b) 2nd Sample



(c) 3rd Sample

Figure 5.8: Efficiency KPI Predictions Vs Targets with MLP

Fig. 5.7 shows 3 sample predictions of the Efficiency KPI from the test dataset alongside its ground truth. Torque is displayed against its angular velocities and all efficiency values within the envelope can be viewed as differing colors in a contour plot. The figures are augmented with a scale showing the efficiency values in percentage across varying levels.
I have also taken the liberty of displaying the error margin as the overlap difference of the predicted and target efficiency values for the samples. The margins of the errors can be seen as levels referring to the difference in efficiencies to the right of each subplot. The 3 subplots are the prediction, target and its difference respectively such that all of them are displayed side by side for easy comparison.

I visualize the images for the same samples in Fig. 5.8 however in a larger scale to be able to decipher the intricacies within the Efficiency grid. Thus, these figures serve to give a larger view for comparing and contrasting the predictions and its targets.

Fig. 3.7 depicts the standard deviation across different angular velocity ranges for the model's predictions as error bar plots around the mean at equally spaced intervals of 2000 rpm. I restrict the image to only show the Efficiency values from 2000 rpm onwards since at very low angular velocities falling in the range of 0 and 2000 rpm, the efficiency values vary drastically from 0% onwards.

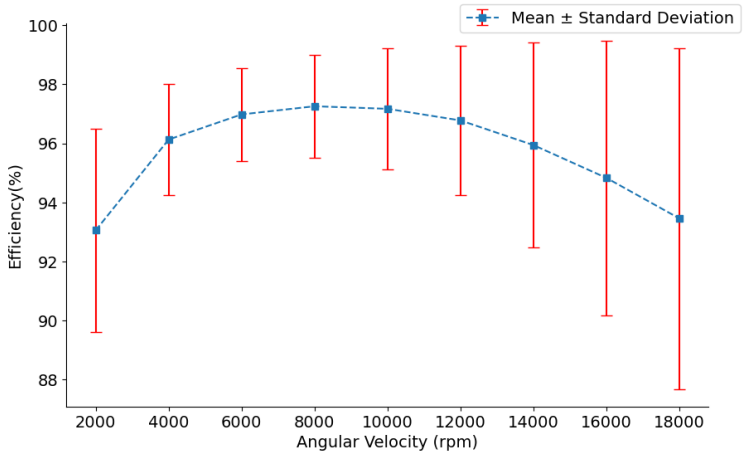


Figure 5.9: MLP Standard Deviation of Efficiency KPI across Angular Velocity Intervals with MLP

Efficiency KPI is relatively more challenging to evaluate scoring being 3D with dimensions angular velocity, torque and efficiency. There exists 191 angular velocity ranges containing efficiency values. Since it is infeasible to show the deviations of the efficiency with respect to all angular velocity intervals, I visualize the deviation with its respective targets only for certain angular velocities across the entire torque range in Fig. 5.10. Therefore, the whole figure encompasses 6 subplots for specific angular velocities.

The figure shows the mean Efficiency values and the RMSE surrounding it for the 10 handpicked samples from the test dataset. The angular velocities are chosen at equal intervals of 2000 rpm. The Efficiency values as usual range between 0 and 100% percentage whereas the torque values are shown from 0 to approximately 250 Nm which could be the maximum torque value from the predictions of the handpicked samples. The RMSE of each prediction from its target can be viewed on the twin y-axis.

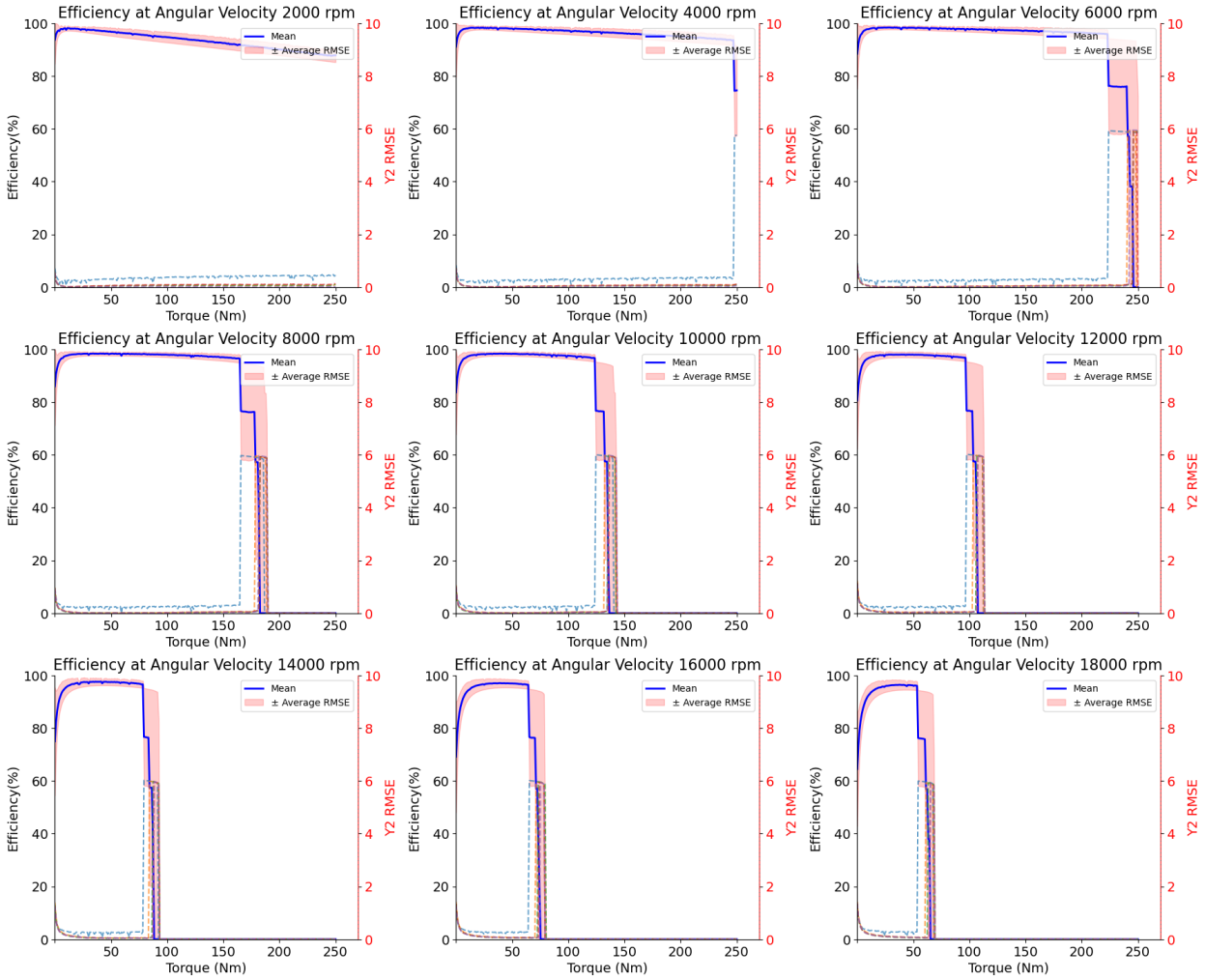
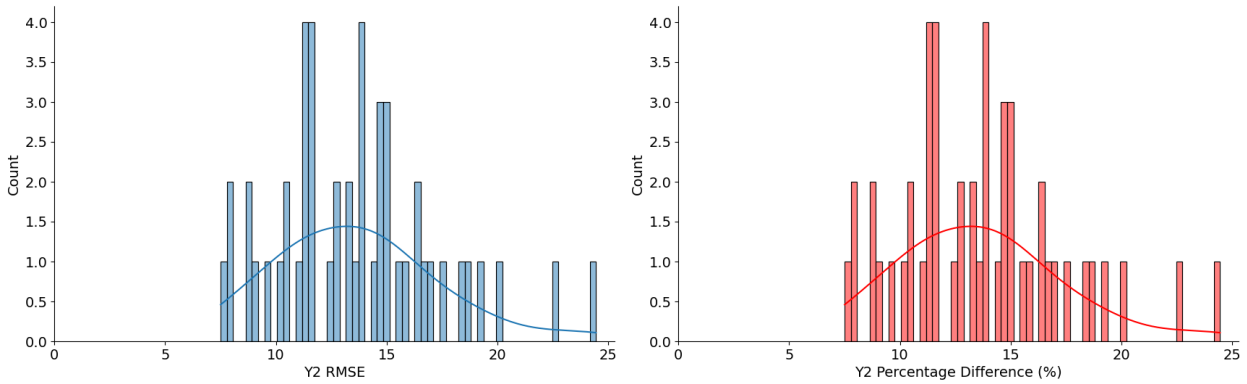


Figure 5.10: MLP RMSE Evaluation for Efficiency KPI



(a) $\mathcal{Y}_2$ RMSE

(b) $\mathcal{Y}_2$ Percentage Difference

Figure 5.11: $\mathcal{Y}_2$ Evaluation Statistics of MLP

Fig. 5.11 shows both the Fig. 5.11a and Fig. 5.11b from Equations 4.13 and 4.16 as histogram plots over the entire test dataset of 50 samples with the count of samples on the y-axis. The histograms for both the

figures are the same, nevertheless the intention is to be able to relate to the scores and percentage difference easily due to differing value ranges of the targets.

Fig. 5.12 illustrates how each fold's prediction of efficiency deviates with the mean of the efficiency predictions of all folds against different angular velocities. Once again, to overcome the limitation of not being able to evaluate for all angular velocities, the Figure hosts only 6 subplots for specific angular velocities. The evaluations is for the same target used in Fig. 5.6 for all of the 5 folds.

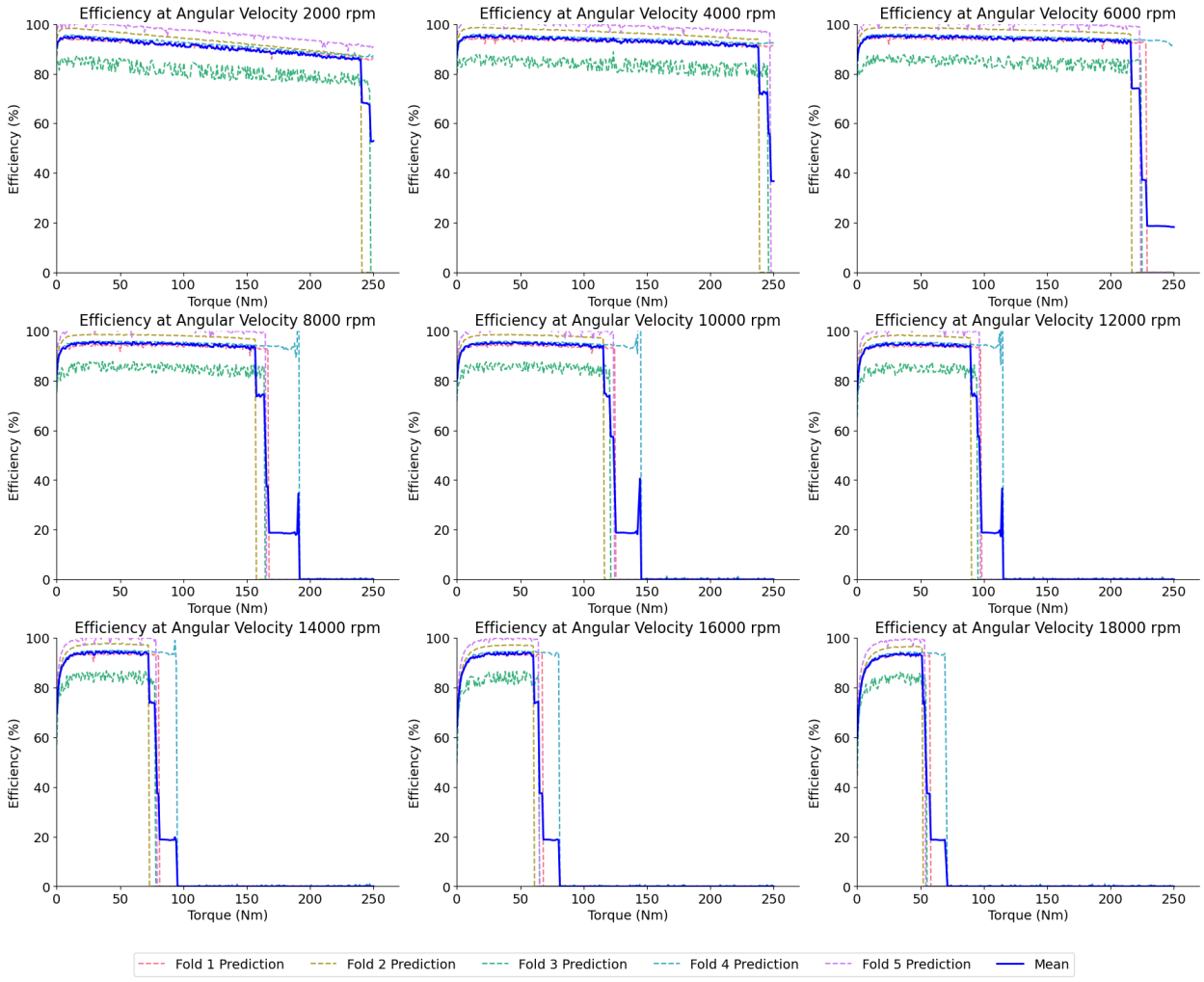


Figure 5.12: MLP Training Deviation across Folds for Efficiency KPI

The following are the observations I infer of the Efficiency KPI from the model:

1. Visually in Fig. 5.8, the predictions to a relative extent resembles the target and the regularization of the Efficiency KPI seems to be already teaching the model the nature of the Efficiency KPI's grid at extreme angular velocity and torque.
2. It can also be noted the efficiency values are being predicted well from the contours of the Efficiency grid.
3. From Fig. 5.7, it can be observed that for all 3 samples, the difference highlighted is the Efficiency Envelope which is the Torque Curve being sliced off irregularly. This is the effect of us trying to retain the Torque KPI shape as was discussed in Section 4.5. Additionally, it could be having negative impacts for the Efficiency KPI when the predictions for the Torque KPI are not perfect. This is yet again a decision to be made on which of the 2 targets to prioritize with the weightage parameter wt.

4. Additionally, Fig. 5.8, Fig. 5.7 would also highlight the anomaly of efficiency values exceeding beyond 0-100% if it occurs since the levels of the contours are fixed likewise. Any values beyond this range would be visualized haphazardly.
5. In comparison to Fig. 3.7 it can be inferred that in Fig. 5.9, the standard deviation of efficiency values predicted by the model are slightly broader indicating the spread of values in the prediction.
6. Fig. 5.10 illustrates that for 1 sample the efficiency values compared to the other 9 samples RMSE which appear well approximated. This inturn indicates that for 90% of the handpicked samples, the predictions are upto expectations.
7. Another observation that can be made is that the average RMSE overlap along the Mean Efficiency is at its peak towards the extreme torques across the Efficiency envelope. The distinction is particularly prominent along the Efficiency envelope.
8. However it is not only the border of the envelope a question of concern here but also for neighboring values of the envelope namely from 1/4th the angular velocity onwards which is close to 6000 rpm. I do not have any way to teach the model this part of the grid with loss regularization because I cannot estimate the envelope dimensionality having already padded the targets to be of the maximum shape referred in Section 3.1.3. The reason why it deteriorates from 1/4th the angular velocity onwards is because the Efficiency KPIs envelope starts to converge from around 6000 rpm as can be concluded in Fig. 1.5.
9. Additionally, I would like to point out from both Fig. 3.7 and Fig. 5.10 that the efficiency values do not exceed 100% which makes the effect of the regularization in Equation 4.6 more evident.
10. It can be deduced from Fig. 5.11 that the RMSEs are gaussian distributed between 7-25 with the mean close to 14 thus encompassing a range of 7-25% percentage difference with the target values as derived from the Table 4.1.
11. It can be remarked from Fig. 5.12 that the predicted torque curves for all the folds except Fold 3 are as close as possible to each other in terms of efficiency values. However, Fold 4 approximated the Efficiency envelope very different from the other folds. As 3 folds approximation is close to each other, it indicates the model's generalization capability and its stability.
12. To put in a nutshell, the relatively higher errors are located in the operating area along the shape of the torque curve.

Observations from the predictions helped to correct few discrepancies in my development for instance in the Efficiency grid I had replaced 0s with NaN values which I later understood were both represented differently in the contour grid. Since Efficiency values can take up values only between 0-100%, I regard the same as constant across plots and use it as a baseline for determining the levels in the contour plot.
Additionally, I calculate the RMSE and differences of prediction from the target by truncating the predicted and ground truth matrices to be of common shape. I also replace NaN values with 0s in either the target or prediction if it occurs as the Efficiency KPIs were padded with NaNs to be of the same shape but originally had no values as was discussed in Section 3.1.3.

5.3 Results with Baseline

From my observations on how the predictions closely resembled that of the target values in Section 3.1.2 and Section 3.1.3, I have developed a Baseline model which is intrinsically the average of all samples of the training dataset.

5.3.1 Torque KPI Results

The $\mathcal{Y}_1$ Baseline score for the Efficiency KPI can be formulated as shown in Equation 5.1.

$$\mathcal{Y}_1 \text{ Baseline Score} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\sqrt{\frac{1}{h}\sum_{j=1}^{h}(\bar{y} - y_{ij})^2}}_{\mathcal{Y}_1 \ Baseline \ RMSE}, \tag{5.1}$$

where n is the number of EM samples, $\bar{y}$ is the Baseline Average mean, h is the columns of 1D vector, $\mathcal{Y}_1$ *Baseline* RMSE is the RMSE for each test sample.

Fig. 5.13 shows the Average RMSE and element wise RMSE for the test dataset performance with the Baseline Model. The Figure is similar in structure to Fig. 5.4, however the distinction is in the values being displayed. Herein, the mean of the predictions from Baseline Model and RMSE surrounding it is displayed for the 10 samples from the test dataset. Additionally, in the twin y-axis, the RMSE of the Baseline predictions from its target are shown in dotted lines.

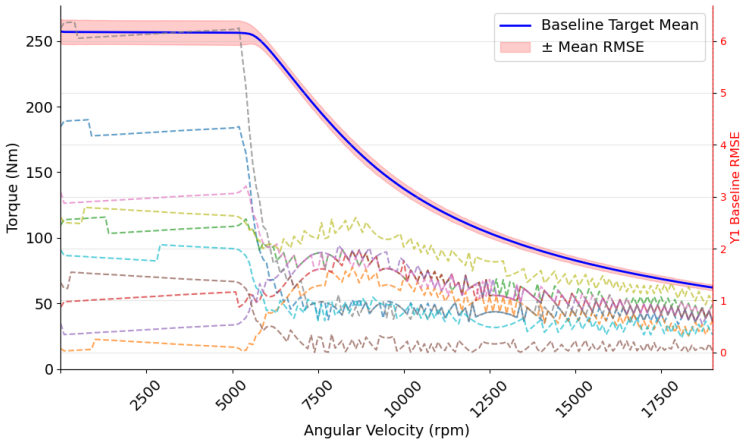


Figure 5.13: Baseline RMSE Evaluation for Torque KPI

The following are the observations that can be inferred for the Torque KPI:

1. In Fig. 5.13, the deviations are at its most towards the beginning of the curve which steadily decreases along the curve.
2. There are almost minimal fluctuations in the RMSEs as compared to the predictions from the MLP model in Fig. 5.4 for 1/4th of the initial angular velocities.
3. Most importantly, it can be noted that RMSE is at its highest soaring upto 6 compared to any of the MLP models.
   This particular observation is the basis for the rationale that I believe the Baseline model will not serve as a good approximation of the Torque KPI.

5.3.2 Efficiency KPI Results

The $\mathcal{Y}_2$ Baseline score for the Efficiency KPI can be expressed as shown in Equation 5.2.

$$\mathcal{Y}_2 \text{ Baseline Score} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\sqrt{\frac{1}{w}\frac{1}{h}\sum_{j=1}^{w}\sum_{k=1}^{h}(\bar{y} - y_{ijk})^2}}_{\mathcal{Y}_2 \; Baseline \; RMSE}, \tag{5.2}$$

where w is the rows of 2D vector, h is the columns of 2D vector and $\mathcal{Y}_2$ *Baseline* RMSE is the RMSE for each test sample.

Fig. 5.14 gives a neat visualization of the Baseline Efficiency RMSE with its respective targets for certain angular velocities across the entire torque range. This figure is yet again similar in structure to Fig. 5.10 except that the efficiency values shown are the predictions of the Baseline model.

The following are the observations that can be inferred for the Efficiency KPI:

1. The efficiency values are almost the same for more than 3/4th of the grid across the chosen samples. This is notable from observing the overlap of the average RMSE with the Mean Efficiency.
2. However for the area along the Efficiency envelope, it can be noted that almost all samples have a deviation from the mean Efficiency values.
3. The deviation at the Efficiency envelope is most prominent at lower angular velocities and gets better as the angular velocity increases.
4. There are almost minimal fluctuations in the RMSEs as compared to the predictions from the MLP model in Fig. 5.4 for 1/4th of the initial angular velocities.

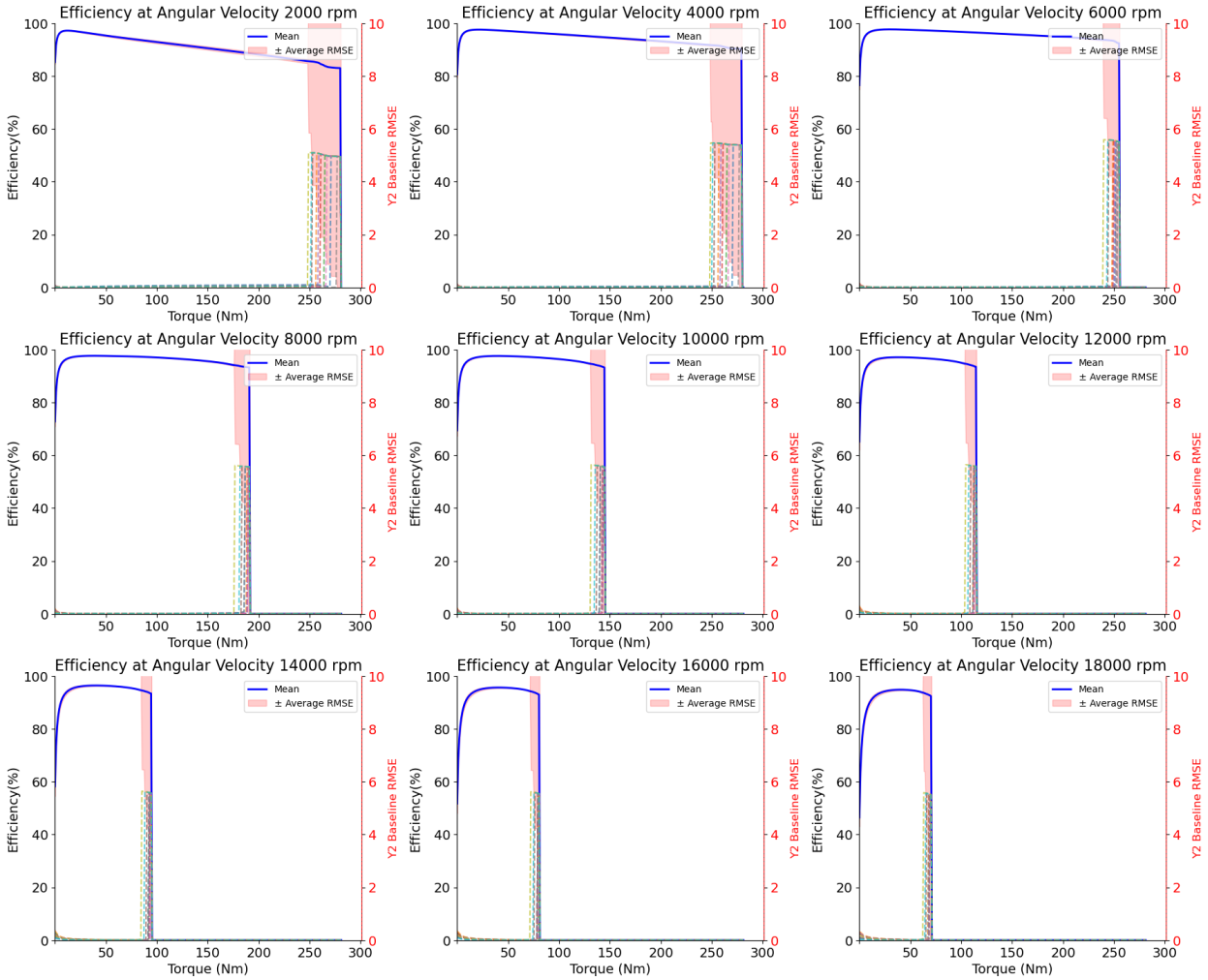


Figure 5.14: Baseline RMSE Evaluation for Efficiency KPI

5.4 Results with MLP Torque KPI Loss Regularizations

The model we have used here is the MLP model with the Torque KPI loss regularization terms. As we are only regularizing the Torque KPI, I have refrained from evaluating the Efficiency KPI here.

5.4.1 Torque KPI Results with Smoothening Curve Loss Regularization

The model I have used here is the MLP model with the Torque KPI with Smoothening Curve Loss Regularization discussed in Equation 4.2.
Fig. 5.15 shows the Average RMSE and element wise RMSE for the 10 samples from the test dataset with

the model. The structure of the figure is the same as Fig. 5.4.

The following are the observations that can be inferred for the Torque KPI:

1. In Fig. 5.15, it can be noted that RMSE is considerably high compared to other models soaring upto 2.5.
2. Apart from the 2 samples which we expect the RMSE to be high owing to them being outliers as was discussed in Section 3.1.2, 5 additional samples appear to also have a high RMSE.
3. The deviations are at its peak towards the beginning of the curve.

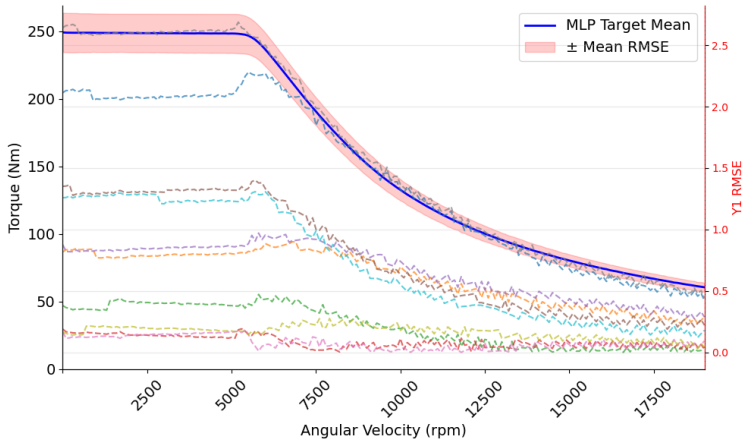


Figure 5.15: Smoothening Curve RMSE Evaluation for Torque KPI

5.4.2 Torque KPI Results with Decreasing Curve Loss Regularization

The model I have used here is the MLP Model with Decreasing Curve regularization discussed in Equation 4.3.
Fig. 5.16 shows the Average RMSE and element wise RMSE for the 10 samples from the test dataset with the model. The structure of the figure is the same as Fig. 5.4.

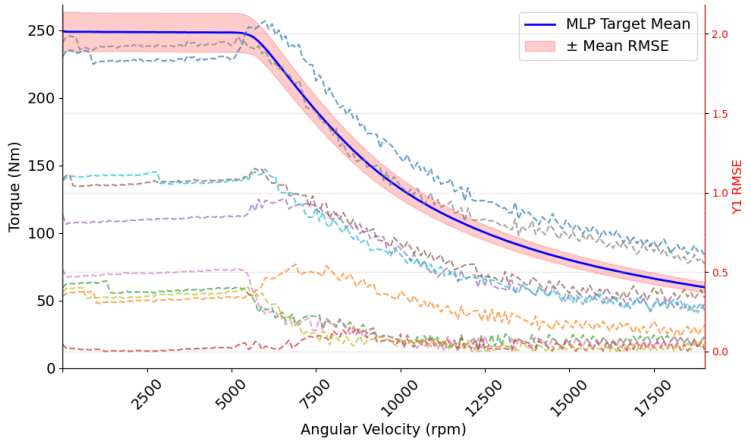


Figure 5.16: Decreasing Curve RMSE Evaluation for Torque KPI

The following are the observations that can be inferred for the Torque KPI:

1. In Fig. 5.16, it can be noted that RMSE is considerably at its second high ranging upto 2.
2. Apart from the 2 samples which we expect the RMSE to be high owing to them being outliers, 3 additional samples appear to also have a high RMSE.

3. The deviations are at its peak towards the beginning of the curve.

5.5 Results with MLP No Loss Regularization

The model we have used here is the MLP model without any loss regularizations.

5.5.1 Torque KPI Results

Fig. 5.17 shows the Average RMSE and element wise RMSE for the 10 samples from the test dataset of the model without any regularizations. The structure of the figure is the same as Fig. 5.4.

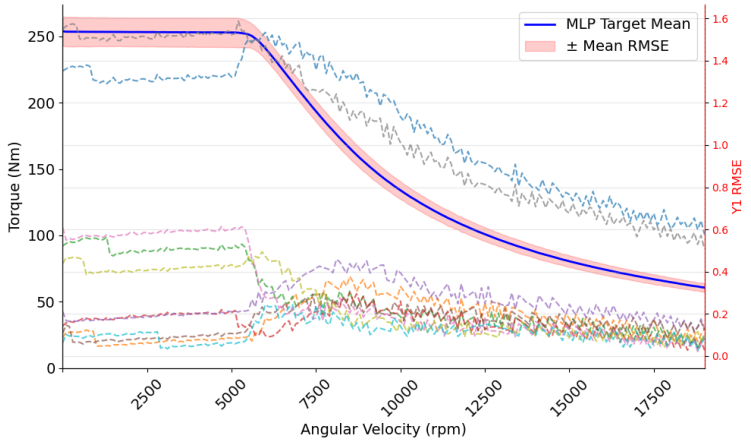


Figure 5.17: No Loss Regularization RMSE Evaluation for Torque KPI

The following are the observations that can be inferred for the Torque KPI:

1. In Fig. 5.17, it can be noted that RMSE is considerably the least among the other models.
2. Apart from the 2 samples which we expect the RMSE to be high owing to them being outliers, the other samples appear to be performing well in terms of low RMSE. This is a stark comparison to the Predictions with Loss Regularizations in Section 5.4.
3. The deviations are at its peak towards the beginning of the curve and gradually decrease when transitioning along the curve.

5.5.2 Efficiency KPI Results

Fig. 5.18 gives a visualization of the Efficiency RMSE with its respective targets for certain angular velocities across the entire torque range. The angular velocities are chosen at equal intervals of 2000 rpm. This figure is similar in structure to Fig. 5.10.

The following are the observations that can be inferred for the Efficiency KPI:

1. Similar patterns to the images generated with loss regularization in place can be remarked in Fig 5.18 with the RMSE varying the most along the Efficiency envelope.
2. However, on looking closely into Fig. A.3e and contrasting with the target Fig. A.3f, it can be noted the efficiency at low angular velocities do not have uniform coloring in blue at the border. This is representative of just 1 sample and on visual inspection of other samples, the fact that the nature of the grid has not been captured becomes more visible in addition to efficiency values not conforming to be below 100%. This observation makes it of utmost importance to factor in the Efficiency KPI loss regularization to the MLP model.

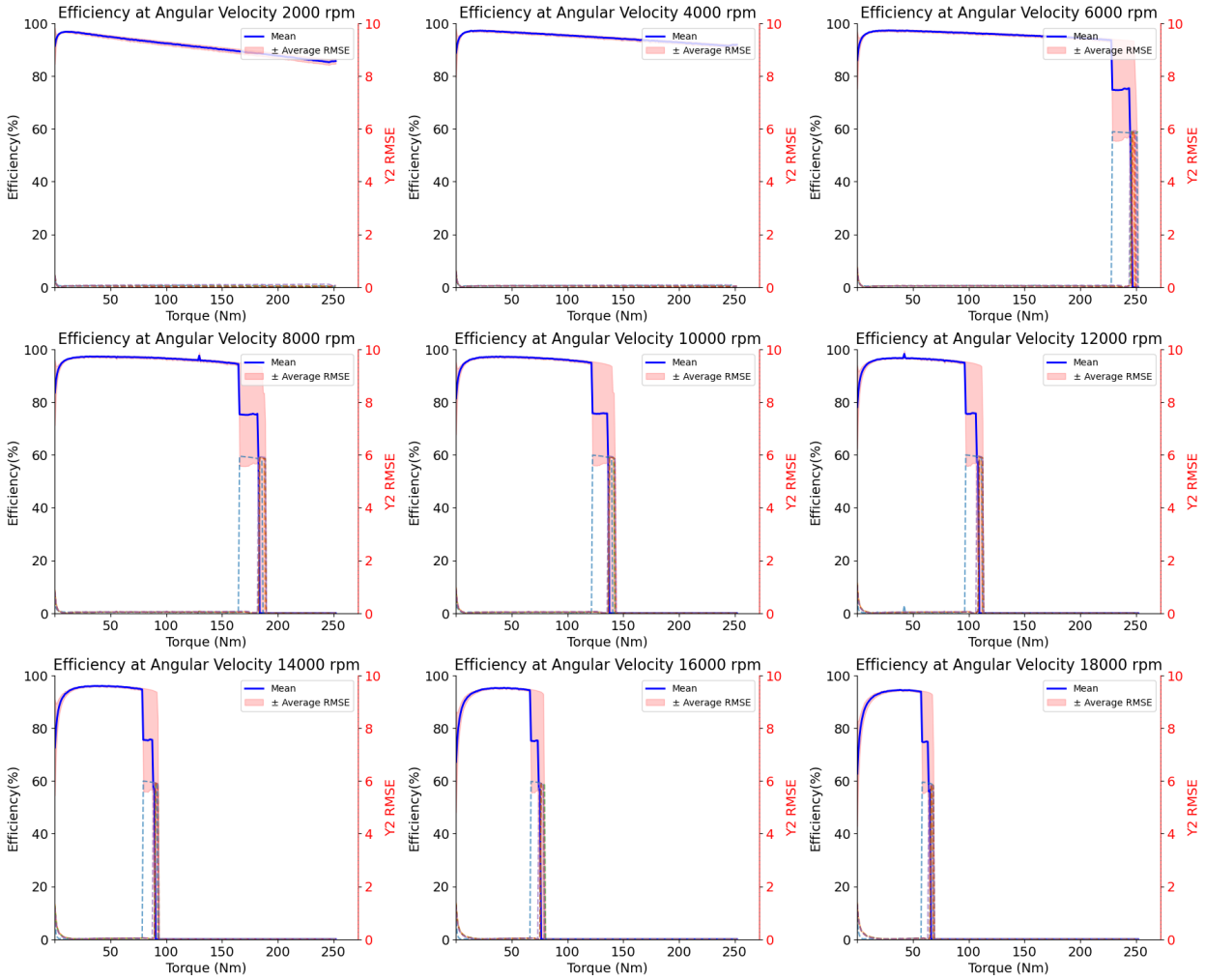


Figure 5.18: No Loss Regularization RMSE Evaluation for Efficiency KPI

5.6 Ablation Studies

I conduct comprehensive ablation studies on the MLP model with different loss regularizations and the Baseline model for both our targets.

Model	$\mathcal{Y}_1$ **Score**	$\mathcal{Y}_2$ **Score**
Baseline	4.8201	15.3708
MLP[a]	4.8552	13.6537
MLP[b]	6.7359	15.5493
MLP[c]	8.7409	14.3229
MLP[d]	4.9058	15.5399

Table 5.2: Ablation Studies

[a]MLP with $\mathcal{Y}_2$ Loss Regularization
[b]MLP With Decreasing curve $\mathcal{Y}_1$ Loss Regularization
[c]MLP With Smoothening curve $\mathcal{Y}_1$ Loss Regularization
[d]MLP without both $\mathcal{Y}_1$ and $\mathcal{Y}_2$ Loss Regularization

Results in Table 5.2 indicate that my MLP model with only the Efficiency KPI regularization's performance is in par with the Baseline model on the Torque KPI and has outperformed it on the Efficiency KPI. This could be because the regularizations are restrictive and discourage the model to learn better. The enriched ablation studies I undertook demonstrate the robustness of the method across varying hyperparameter settings.

Additionally in Appendix Fig. A.3, I generate KPIs of 1 sample for all the models we developed to compare and contrast its predictions with the ground truth.

The following are the observations that can be inferred from the ablation studies:

1. The MLP model with no loss regularization best approximates the Torque KPI on visual inspection of the start of the curve. The MLP models with loss regularizations appear to perform the worst in this regard.
2. Visually, efficiency values in Efficiency KPI appear to be best for Baseline model and the MLP model with Efficiency KPI loss regularization. This is notable from the clear segregation of contours in the Efficiency maps. Although, the baseline model looks visually more appeasing yet it would be wiser to trust the scoring as each color in the contour holds a range of values and finer details could be overlooked.
3. For the MLP models with no loss regularizations, the nature of the Efficiency grid as was mentioned already is not accurately captured. This particularity made me realize that the Efficiency grid regularization is good to have for the main model.
4. For both the MLP models with Torque KPI regularizations, the torque values in the torque are not well approximated as it is predicted to be much lower than the actual values. This prompts us to exclude the Torque KPI regularization from our main model.
5. In addition, for the MLP models with Decreasing Curve loss regularization, the efficiency grid appears to be cut off quite a lot, indicating that the loss regularization forces the model to decrease the torque values more than required and could potentially be controlled with λ_{2y1} parameter.

The inferences shown are strictly speaking from the Double V Magnet Topology which assumed the bulk of the data I had received. Considering the limitations in dataset, conducting thorough examinations of a substantial volume is typically unfeasible. Therefore, the primary purpose of the thesis is to study the possibility of modelling the KPIs predictions and my learnings from it.

5.7 Compute and Hardware Comparison

Table 5.3 gives a glimpse of the time conserved with surrogate modelling and relates both the approaches highlighted in Fig. 1.6 in terms of time comparison and hardware requirements.

Approach	**Time taken(minutes)**	**Hardware**
Surrogate Modelling (Proposed Approach)	18.5 seconds	32 CPU Cores & 1 GPU
FEA simulations (Current Approach)[a]	5 minutes	1500 CPU Cores

Table 5.3: Compute and Hardware Comparison

[a]Source : Valeo

I would also like to bring to attention that the time of taking 5 minutes to generate KPIs per design using the Current Approach highlighted in Fig. 1.6 is only possible by running the FEA simulator across High Performance Computing machines of nearly 1500 CPU cores in parallel. As the number of cores decrease the simulation would take minutes to hours to converge. Needless to mention, it is the FEA Solver in the block that is the culprit for consuming most of the time.

Another pointer to note is that in reality the time needed to generate KPIs with Surrogate Modelling is in the range of milliseconds. However most time is utilized during the dataset creation in Data Preprocessing phase since reading the excel files to retrieve the tabular data takes up a significant chunk of time close to 18 seconds whereas only 5 seconds is required to generate the prediction and display both KPIs. The computing time of the Proposed Approach is approximately 16 times faster than the Current Approach even without considering the disparity in hardware used. As opposed to FEA simulations, the time required to generate predictions for several motor designs is memory bound and computation is not so much of a bottleneck since I can potentially run both training and inference across multiple GPUs in parallel.

I have used Python 3.10.14 for my development and the `Pytorch`[2] library compatible with cuda. The model was trained on a NVIDIA Tesla V100 GPU with 32 GB Memory and the machine I used had 16 CPUs of the Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz with allocated disk space of 50 GB for my experiments. Source code is available at `Github Repository`[3].

[2] `https://pytorch.org/`
[3] `https://github.com/Lilly-25/Masters-Thesis`

Chapter 6

Graph Modelling

6.1 Introduction

Tabular representation of data in itself cannot account for the spatiality of an EM design especially on how its different components interact with one another. Meanwhile, graphs thrive at better representing data in the non-Euclidean domain. Therefore, I presume modelling my usecase as a graph will be more reasonable so as to exploit its graph dynamics and aggregate features that are semantically similar. Additionally, graphs would be even more realistic with achieving topology invariance than the tabular representation and conserve memory and compute in the long run.

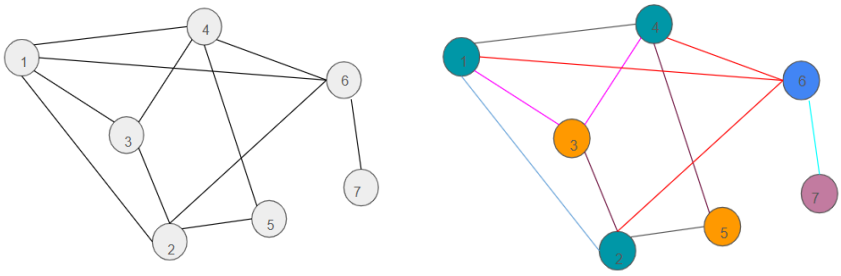
A graph is composed of nodes and edges with their corresponding features in addition to the graph attributes. The graph attributes contain the information relevant to the graph as a whole whereas the node features captures the information of the node's role in the network and likewise the edge features captures the information of the edge's role in the network. Graphs also take into account their respective neighborhoods.

Researchers have broadly classified GNNs into the following 2 distinct paradigms:

1. **Homogeneous GNN**
   They are designed for graphs with a single type of node and edge. Homogeneous GNN are typically build to capture the structural information within a graph. Message Passing (MP) is done for neighbouring nodes and edges over hops until it learns a representation equivalent from its neighbours.
2. **Heterogeneous GNN**
   They are designed for graphs with differing types of nodes and edges implying difference in features as well as in dimensionality. As a single function cannot cater to each type, hence different MP and node updating functions needs to be implemented for each edge and node type respectively. Therefore MP is conditioned on the node and edge type thus allowing the flow of information to be more controlled. In addition to the structural information, Heterogeneous GNNs also excel to capture semantic information within the graph.

Meanwhile, there exists Knowledge graphs which are essentially heterogeneous where the nodes are entities and the relationships the edges of the graph. They are used for large corpus of text and incorporate semantic meaning well into the graph as text embeddings. Knowledge graphs have been gaining popularity as a Retrieval Augmented Generation alternative because of its ability to hold semantic information and generate structured data by traversing multiple hops of each node. All knowledge grtaphs are heterogeneous graphs but all heterogeneous graphs are not knowledge graphs.

Fig. 6.1 distinguishes the 2 graphs with equivalent number of nodes and edges by the coloring used for different nodes types and edge types they possess. Both the figures are of graphs comprising of 7 nodes and 11 edges. Fig. 6.1b consists of 6 edge types and 4 node types whereas Fig. 6.1b contains just 1 node type and edge type.

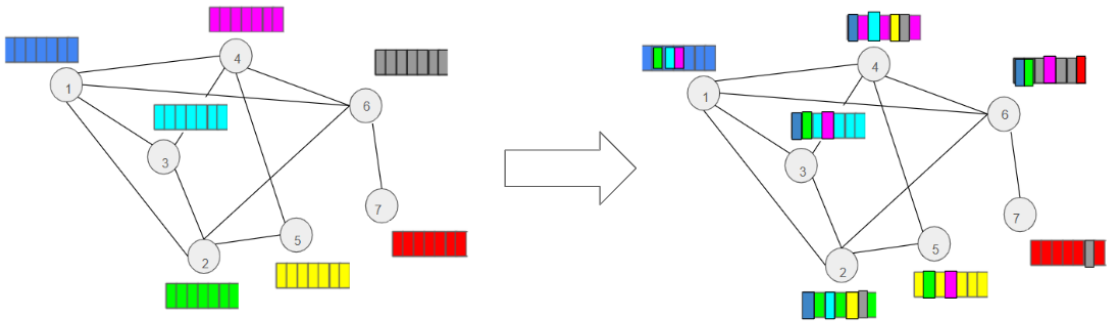


(a) Homogeneous Graph (b) Heterogeneous Graph

Figure 6.1: Graph Classification

6.1.1 Message Passing

GNNs in general function by making use of MP which is a recursive algorithm that aims to learn a representation vector for each node. MP is based on the graph structure and initial features per node. Over each successive hop in the graph, the information of neighboring nodes are shared across as messages and aggregated after which the initial node is recursively updated with this information. The cycle continues until there are no more nodes left to traverse within the graph. Thus, at the end of the MP algorithm, each node in the graph would have a good understanding of the other nodes within the same graph. Therefore, the final representation will be rich enough to have captured all the information within the graph and can be used for downstream tasks.



(a) Message Passing for whole graph

(b) Heterogeneous Graph

Figure 6.2: Message Passing

Fig. 6.2 illustrates the working of MP in a homogeneous graph of 7 nodes and 11 edges. Fig. 6.2a shows the effect of MP for all nodes in the graph whereas Fig. 6.2b zooms in on the effect of MP for a single

node in the graph. The neigboring node vectors of node 1 are aggregated and updated to node 1's vector. Aggregation is made visually understandable by color coding each node's respective vector representation distinctly and node updation is implied by updating its respective color to the vector index corresponding to the node number.

GNNs cannot be too deep with layers due to the oversmoothing problem. Oversmoothing is said to have occurred in a GNN when the MP algorithm would have already traversed over all hops and made all node features indistinguishable from one another. Besides a deep GNN would have a large number of parameters and would be computationally expensive to train.

6.1.2 Applications

Some of the well known GNN tasks can be seen in Fig. 6.3 and listed as follows :

1. Graph Classification - Used for classifying graphs into different categories. Although our usecase is a graph level prediction, it is a graph multi-regression task as we need to predict continuous values as was clarifed in Section 1.1
2. Node Classification - A node's class is predicted based on its proximity with other nodes in its neighborhood, node features among other factors. An application could be in spam tagging of emails.
3. Link Prediction - Predict whether there exists relationships between nodes in the graph. A common application is predicting drug treatment based on gene-protein interactions.
4. Community Detection - Classifies groups of nodes into clusters. One scenario of application is to detect mutual communities in a social network.
5. Graph Embedding - Embeds high dimensional graph into a lower dimensional space to be able to visually inspect it better. Visualizing huge corpus of text as a Knowledge graph in lower dimension, enables one to detect communities of topics and subtopics in each cluster.
6. Graph Generation - Generates synthetic graphs based on initial graph representation. An application I can relate to is the inverse formulation of our problem modelled as a graph which can be envisioned in the future as was discussed in Section 1.2.

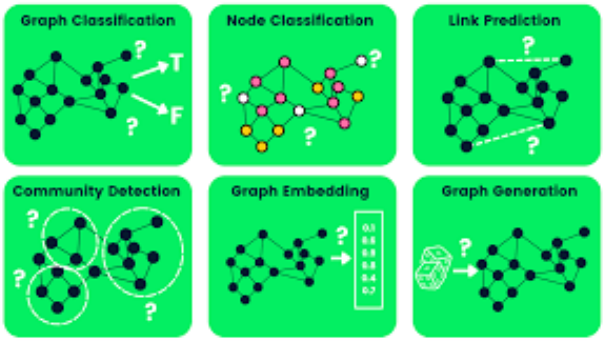


Figure 6.3: GNN Tasks (Source : https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial)

6.1.3 Case Study Structure

I have organized the outline of the case study to be as follows: First, in Section 6.1, I provide an introduction to GNNs. I then delve into finer details of the Heterogeneous GNNs in Section 6.2 and formulate definitions mathematically. Next, I review Literature done on Heterogeneous GNNs in Section 6.3. After which I present the methodology in Section 6.4 and share my view on how to create the heterogeneous graph for our usecase. Lastly, I wrap up with a brief discussion of the pros and cons of using heterogeneous GNNs in Section 6.5.

6.2 Heterogeneous GNN

Graphs are generally heterogeneous because of its capability to have different node and edge types in addition to its graph property of modelling relationships between objects. Heterogeneous GNNs also takes into account homophily wherein nodes close on a network have similar embeddings as stated by the writers of Ref. [26]. This is in essence the structural property and is true for all GNNs.

I largely adopt the commonly used notations from Ref. [05] and reformulate the definitions slightly for my task as follows.

Heterogeneous Networks

Given a directed graph $\mathcal{H} = (V, E, X, R, T_v, T_e, \phi, \psi)$, where V is the set of nodes, E is the set of edges, X is the set of node features, R is the set of edge features, T_v is the set of node types and T_e is the set of edge types.

Each node $v \in V$ has a node type $\phi(v) = \nu \in T_v$, where $\phi(\cdot)$ is a node type mapping function.
Furthermore, for $\phi(v) = \nu$, v has features $x_v^\nu \in X^\nu$, where $X^\nu = \{x_v^\nu \mid v \in V, \phi(v) = \nu\}$ and $X = \{X^\nu \mid \nu \in T_v\}$. Thus, the set of node types, $T_v = \{\phi(v) : \forall v \in V\}$.
The dimension and attributes of the node feature x_v^ν may be different for different node types ν.

Likewise, each edge e_{uv}^ε has an edge type $\varepsilon \in T_e$ connecting node u to v, where $\psi(\cdot)$ is an edge type mapping function.
Each edge e_{uv}^ε has features $r_{uv}^\varepsilon \in R^\varepsilon$, where $R^\varepsilon = \{r_{uv}^\varepsilon \mid u, v \in V\}$ and $R = \{R^\varepsilon \mid \varepsilon \in T_e\}$. Thus, the set of edge types $T_e = \{\psi(e) : \forall e \in E\}$.
Similar to nodes, the edge features r_{uv}^ε can also have different dimensions and attributes for different edge types ε.

If $|T_v| = |T_e| = 1$, it implies there is only 1 node and edge type then the graph degenerates into a homogeneous graph as stated by the authors of Ref. [02].

'Heterogeneous GNNs aim to learn a representation vector $\mathbf{h}_v^{(L)} \in \mathbb{R}^{d_L}$ for each node v after L -layer transformations, based on the graph structure and the initial node feature $\mathbf{h}_v^{(0)} \in \mathbb{R}^{d_0}$' as stated by Lv *et al.* in Ref. [04].

Fig. 6.4 illustrates each of the below terminologies to relate better.
Fig. 6.4a shows a metapath connecting 4 nodes whereas Fig. 6.4b dissects the metapath to only connect 2 nodes. Fig. 6.4c illustrates the neighbors of the nodes in the metapath and Fig. 6.4c connects the nighbors of a particular node to form its subgraph.

Metarelation

For an edge $e = (s, t)$ linking source node s to target node t, its meta relation is denoted as $\langle \phi(s), \psi(e), \phi(t) \rangle$.
Metarelations are the relation triples used in knowledge graphs as it contains richer schema.
It is also interesting to note that the metapath is a sequence of many metarelations.

Metapath

Metapaths are composite relationships between nodes that help to capture the structural information of heterogeneous graphs.

Given a path $\mathcal{P} = T_{v1} \xrightarrow{T_{e1}} T_{v2} \xrightarrow{T_{e2}} \cdots \xrightarrow{T_{el}} T_{vl}$, if the path describes the composition of edges by its types $T_{e1} \circ T_{e2} \circ \cdots \circ T_{el}$ between nodes of type T_{v1} and T_{vl} and $\circ$ denotes the composition operator then the path is called a metapath.
The length of the metapath can be calculated from its magnitude which is *l-1*.
If all the combinations of nodes and edges in the graph are used, then the total number of metapaths would increase exponentially with the number of node and edge types. Hence, the reason to subselect only few metapaths that are significantly most relevant for the task at hand.

Metapath selection requires domain knowledge to be able to choose those which are most semantically meaningful and let the Heterogeneous GNN use only these paths for learning the graph. Thus, metapaths control the direction of message passing in a neural network.

The Metapaths in Heterogeneous GNNs are divided into 2 streams:

1. Metapath based method
   It strictly follows the metapaths defined for the network. Here information from each metapath is aggregated before fusing with information of other metapaths that are not so much semantically related.
2. Metapath free method
   There are no metapaths defined for the network and so this method attempts to learn from the graph structure. It aggregates message from neighbourhood of the same hop for all node types and so captures both structural and semantic information simultaneously.

When designing a heterogeneous GNN based on metapaths, then MP operations are done across ech metapaths which are finally aggregated into a new representation for the node as stated in Ref. [05]. Each metapath captures the proximity of nodes in a graph from a unique semantic perspective as cited by authors of Ref. [26].

Metapath-based Neighbors
Given a node v_1 and a metapath $\mathcal{P}$ in a heterogeneous graph, the neighbors $\mathcal{N}_{v_1}^{\mathcal{P}}$ of node v_1 along the meta-path $\mathcal{P}$ are defined as the set of nodes connected to v_1 along the meta-path $\mathcal{P}$ which includes v_1.

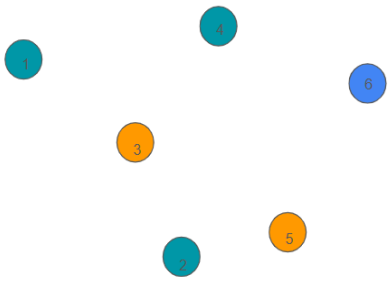
Metapath-based Subgraph
Given a heterogeneous graph, $\mathcal{H}$ with a metapath, $\mathcal{P}$, then the metapath-based subgraph, $\mathcal{H}_{\mathcal{P}}$, is defined as a graph consisting of all its metapath based neighbors. Moreover, $\mathcal{H}_{\mathcal{P}}$ degenerates to a homogeneous subgraph if $\mathcal{P}$ is symmetric.

The graph structure of $\mathcal{H}$ can be represented by a series of adjacency matrices $\{A_\varepsilon : \varepsilon \in T_e\}$. For each relation $\varepsilon_{\nu_t\nu_s} \in R$, $A_{\nu_t\nu_s} \in \mathbb{R}^{|V_{\nu_t}|\times|V_{\nu_s}|}$ is the corresponding adjacency matrix where the nonzero values indicate positions of edges $E_{\nu_t\nu_s}$ of the current relation.

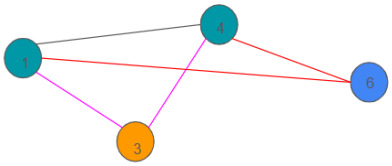
Heterogeneous GNN generally work by having separate non-linear functions convolve over each edge type during message computation and over each node type when aggregating the learned information. During MP in heterogeneous graphs, first the message between nodes of the same type in the hop are aggregated then the non-linear function convolves over each of the aggregated messages. Finally, all messages generated after final convolution which are of the same dimensions are again aggregated and the initial node is updated.

(a) Metapath

(b) Metarelation



(c) Metapath based Neighbors



(d) Metapath based Subgraph

Figure 6.4: Heterogeneous GNN Terminologies

6.3 Heterogeneous GNN Literature Review

I attempt to review the recent works on Heterogeneous GNN by categorizing them across the following subsections:

6.3.1 Metapaths

Xu *et al.* in Ref. [43] demonstrates a technique of splitting metapaths from inside out to be able to capture the semantic meaning as much as possible when developing a semi-supervised MP hetergogeneous GNN. This is particularly useful when the metapaths are too long and the intermediary paths within the metapaths are prioritized to be the newly created metapaths.
Yang *et al.* in Ref. [02] conducted a study on building an efficient Heterogeneous GNN by using single layer longer metapaths rather than multilayer smaller metapaths to capture semantic information of the graph. To capture the structural information instead of attention mechanism of neighboring nodes repeatedly, the authors propose to compute the mean aggregate of the neighborhood as part of a preprocessing step thus simplifying the heterogeneous GNN.
The authors of Ref. [12] propose a relevance measure to capture the sematics of nodes of different types in a heterogeneous graph based on context path bypassing the need of having semantically meaningful metapaths. Context path is principally a path linking two nodes of the same type however the intermediary nodes in the path are auxillary nodes of different types. The nodes making up the context path are weighted with varying importance scores on the basis of their corresponding relevance measure.

6.3.2 Subgraphs

Existing works for instance Yu *et al.* in Ref. [21] model heterogeneous graphs by splitting the graph into multiple homogeneous subgraphs based on their node types each by retaining an aspect of its heterogeneity for each subgraph. However, the methodology is ineffective in exploiting hidden rich semantic associations between different types of edges for large scale multi-relational graphs.
Zou *et al.* in Ref. [24] proposes a heterogeneous GNN for node classification which converts the heterogeneous network into multiple semantic graphs from its defined metapaths. Attention mechanism is leveraged to learn the weight of each of the subgraphs.
Guan *et al.* in Ref. [13] presents a methodology to learn a heterogeneous graph by decomposing it into homogeneous and heterogeneous subgraphs from its metapaths as opposed to symmetrical metapaths. This approach was undertaken because of the fear that symmetric metapaths ignore intermediary nodes resulting in node biased learning. The information from each subgraph is then attention aggregated which would serve as the final representation of the model.

6.3.3 Metarelations

As was already discussed, modelling Heterogeneous Graphs with metapaths require domain specific knowledge since metapaths are customized. In order to combat this, Hu *et al.* in Ref. [23] has used meta relations instead and a unique adjacency matrix for each edge type.
Melton *et al.* in Ref. [25] present a multilayer heterogeneous GNN which uses metarelations to extract soft metapaths and does message passing over multi-hop neighborhood of nodes. Such meta relations are shallow embeddings and not as deep as the meta paths as was clarified by Yang *et al.* in Ref. [26].
Additionally Chana *et al.* in Ref. [01] has used heterogeneous GNNs to model the heterogeneity and sparsity in Electronic Health Records of patients. This work claims to learn the multi-task nature and the relational features of the dataset which is arguably the most important factor in this domain for example past medical history. However it is noteworthy to point out that although this paper uses meta relations it does not factor in edge features.

6.3.4 Oversmoothing

GNNs often face a dilemma to balance between resisting over-smoothing and capturing long range dependencies within the network. Li *et al.* in Ref. [22] demystifies the Graph Convolutional Network (GCN) and addresses the Laplacian oversmoothing concern by self-training or co-training the model. HNonethelss, smoothing makes classification problem easier because over multiple hops the features among different clusters will be the same. GCN achieves this by breaking down the heterogeneous graph into multiple homogeneous graphs, one for each edge type. In each layer, GCN is applied to the homogeneous graph, and the resulting node embeddings are aggregated element-wise to form the final output. Ahn *et al.* in Ref. [41] propose to mitigate oversmoothing by introducing skip connections between layers in the network to favor node specific embeddings. Similarly in Ref. [04], Lv *et al.* study existing heterogeneous GNNs and also cites preactivated residual connections can help resist over-smoothing from occuring.

6.3.5 Applications

Most of the existing works for example Yang *et al.* demonstrates in Ref. [26] that they do not consider edge features in their model. However, Johannessen *et al.* in Ref. [05] uses edge features as transactions in a financial network to detect cases of money laundering. They breaks down the large heterogeneous graph into subgraphs based on different node type-edge type combinations.
Gilmer *et al.* in Ref. [27] introduces modelling for chemical prediction problems that makes it capable of learning features of molecular graphs and is invariant to graph isomorphism. Graph isomorphism is the property of graph being similar even when the ordering of nodes and edges are different. They also suggest that edge features in the network can be learned by introducing hidden states for all edges in the graph and updating them. Additionally, the writers have implemented a MP network which factors in the edge types.
Bias Amplification is a known problem most prevalent in recommender systems where the model is biased to certain outputs due to the bias existing in its training data. It can be handled much better when a graph is modelled as a heterogeneous graph as is shown in Ref. [03]. This is because differing edge types capture varying types of relationships in the graph. The authors of this paper propose a heterogeneous GNN to predict the purchase probability of electric vehicles by modelling different characteristics such as charging infrastructure, branding, environmental factors, local policies among others in the form of a heterogeneous graph.
Dong *et al.* in Ref. [40] review pitfalls in Heterogeneous representation Learning. They also clarify that for metarelations connecting 2 different node types, its corresponding edge type will be unique to similarly connected nodetypes of other metarelations. Customizing metapaths could bias it to be task-specific and is limited to discrete space.
Yu *et al.* in Ref. [42] addresses the concern where featureless nodes exists by padding 0s, imputing the mean of neighboring nodes and using pretrained graph embeddings as initial features. Alternatively, Heterogeneous Graph Transformers can be used to learn the implicit metapaths first from the graph by using feature propagation across multiple layers in its neural network architecture to augment the original graph.
Gao *et al.* in Ref. [11] proposes an architercture search for heterogeneous GNNs to effectively choose the best network architecture by using reinforcement learning. This methodology can also be extended to search the hyperparameter space to find the best hyperparameters for the model.

6.4 Heterogeneous GNN Modelling

Although I was certain that GNNs could best represent our usecse, I realize that our problem cannot be solved using Homogeneous GNN which is relatively simpler and is built on a single node and edge type. Heterogeneous graph is more apt for our use case with its different node and edge types as it preserves both the structural and semantics of our data. This property is crucial in modelling our use case as there will then be similar number of nodes and edge per topology. In contrast, Homogeneous graphs would lead to suboptimal results as it cannot factor in the heterogeneity and thus the semantic nature of the usecase fully.

Given the EM data $\mathcal{D}$, the goal is to construct a heterogeneous graph $\mathcal{H}$ from $\mathcal{D}$. Let $\mathcal{Y}_1$, $\mathcal{Y}_2$ on $\mathcal{H}$ be the 2 KPIs that needs to be learned from $\mathcal{D}$ then the aim would be to train a multi-task GNN model $\mathcal{M}$ such that $\mathcal{M}$ can deliver high performance on $\mathcal{Y}_1$ and $\mathcal{Y}_2$.

6.4.1 Heterogeneous Graph Construction

Inspired by the promising advantages of Heterogeneous GNN, I have taken the effort of cnostructing the graph for only the Double V Magnet Topology. I create the heterogeneous graph adaptable for each of the EM variants as a `NetworkX`[1] graph.
The count of certain parameters within the motor such as stator poles with its corresponding stator windings and rotor magnets is made more comprehendable to the model having new nodes and edges whereas for the MLP architecture this information is represented as yet another number in a separate column or as new columns unique to rotor structure. By leveraging the relational features introduced by the meta relations, I can obtain a good graph representation of the data.
The hand drawn sketch of the EM cross-section comes in handy when creating the heterogeneous graph. In Table A.3 and Table A.4, I illustrate the nodes and edges along with its respective types which I had considered when creating the graph. I also share the metapaths that I used during development and would like to reiterate that this is a study and its results are not to be taken as final. I then learn the constructed heterogeneous graph through a heterogeneous GNN model.

GNNs are relatively more expressive models as they can capture the variability and complexity of our data better. However, GNN in general have been less to not explored even so more the heterogeneous GNN. Particularly in the scenario of EM Modelling, to my knowledge there has been no publications with GNNs. Hence, the need to check its feasibility and performance with my benchmarks on tabular data. Additionally, existing Heterogeneous GNNs works for example on recommendation networks, academic networks, information networks, social networks and other applications involves one large graph with multiple node and edge types. These networks are generally represented by a giant graph in which case graph batching, splitting are unique. However, the problem involves creation of multiple heterogeneous graphs i.e., 1 per EM variant. Therefore, the applicability of Heterogeneous GNNs for the problem is yet to be seen.

In addition, I refer to Section A.4 where the terminologies in GNN are more elaborartely defined. Our thesis is loosely inspired from Ref. [05] which has strived to extend MP into Heterogeneous GNNs.

6.5 Summary

I am faced with a dilemma most common when designing Heterogeneous GNNs with predefined metapaths. The challenge is to choose whether I want to represent each component of the EM as dedicated nodes and edges to preserve the semantics of the data and be more flexible across topologies or to define reasonably sized metapaths. As the nodes and edges increases, the metapath length exponentially increases. This makes computation tricky since for each metapath, the MP is done across the nodes and edges its composed of. As I do not have enough domain knowledge in EM modelling, our metapaths comprised of the stator, rotor and the its combination components.

[1] `https://networkx.org/`

Chapter 7

Conclusion

7.1 General Discussion

This thesis offers a fresh outlook to the possibility of modelling the performance of an EM by surrogate modelling the FEA solver's role. I have developed a topology invariant MLP model to predict 2 KPIs of an EM from its parameteric desacription. It would be very beneficial to EM manufacturers to understand the operating range of the vehicle from the EM description and make calculated assumptions of whether to manufacture it.
This work enables EM designers to generate KPIs at almost both negligible costs and minimal time. Thus offering an escape from the cons of using FEA simulations. It also exhibits almost close to better accuracy than FEA simulations with significantly better run time efficiency. The latter can be inferred from Section 5.7.

It also lays the foundation for future work on being able to generate EM design parameters conditioned on the predicted 2 KPIs. I envision that the study will provide new perspectives to researchers engaging in the field of EM design and hope my contribution will aid in the same.
Alternatively I could also build 2 models one for each KPI and thus feed in the dependent predictions when training the latter. However, it would be computationally expensive and does not help in the scenario when there might be a need to generate EM parametric descriptions. Additionally I deemed it unnecessary as the dimensions of the Efficiency grid vary with the Torque curve and not necessarily the Efficiency values.

A contribution I make is to predict the efficiency map based on the Torque curve. To my knowledge, works on predicting the efficiency map has not been dependent on any other performance characteristic of the EM let alone its parametric description. Furthermore, although there has been works undertaken similar to our usecase, to my knowledge there has not been any reliable implementation that can be reused or benchmarked against. Therefore, what makes the thesis stand out is that it has been done from scratch and can be easily modified to reproduce the work or similar works. I would also like to note that incorporating the nature of the KPIs particularly the Efficiency KPI into our loss funtions significantly helped the model generate more accurate predictions. This is yet again a contribution as the literature, I have reviewed do not touch base on loss function modelling for EM surrogate modelling.

A limitation I can note in my work, is the weighing parameter is used for 2 tasks namely balancing the significance of the 2 KPIs and the value ranges between them. This makes implementation tricky when the mentioned 2 factors are contradictory to each other. A possible solution would be to scale the targets to the same scale so that weighing parameter only focuses on the job of balancing the significance of the 2 KPIs.
I would like to highlight, that my experiments are conducted on a relatively small dataset and the approach has potential to work better with a larger variety of data.

7.2 Future Improvements

I would suggest the following improvements to my study :

1. Build a model that uses the Torque KPI prediction to predict its corresponding Efficiency KPI.
2. Although I have designed the model to be topology invariant for 3 topologies, I only had sufficient data from Double V Topology to draw evaluations from. Further evaluations on the other topologies would be beneficial to critically assess my model's performance.
3. Another interesting study would be to benchmark model performance when one backpropagates the losses for each KPI individually rather than weighing them.
4. Additionally, the motivation of building a topology invariant model was the reason I have considered building a heterogeneous graph to model the data. The machinery is elaborated in Section 6.4. Such a model could serve as yet another ablation study to my problem.
5. Exploring different learning rates would also potentially aid in balancing the importance of the 2 KPIs.
6. Scaling the targets to be of the same range to combat significantly different value ranges.

Appendix A

Appendix

Supplementary material to the thesis.

A.1 Dataset Supplementary Information

A.1.1 File structure

Among the excel files shared by Valeo per EM variant, Table A.1 summarizes the sheets of interest to us. This is helpful for reproducing the experiments, visualizing the images Fig. 1.4 and Fig. 1.5 and for carrying out further analysis with the same dataset in the future.

Sheet	**Sheet Name**	**Plotting Axis**	**Unit**	**Description**
Motor Parameters	input_data	-	mm, °	Includes geometric, physical and simulation properties.
Speed Grid	NN	x-axis	rpm	Used for plotting the Torque KPI and Efficiency KPI.
Torque Grid	MM	y-axis	Nm	Used for plotting the Efficiency KPI.
Efficiency Grid	ETA	z-axis	-	Has the same dimensions as in NN and MM.
Torque Curve	Mgrenz	y-axis	Nm	Has the same columns as in NN.

Table A.1: Excel File Structure of an EM variant

A.1.2 Data Summary Statistics

Parameter	**Unit**	**Mean**	**Standard Deviation**	**Value Range**	**Single V Magnet Topology**	**Double V Magnet Topology**	**Nabla Magnet Topology**
General Parameters							
N	-	4	0	4 –4	✓	✓	✓
simQ	-	6	0	6 – 6	✓	✓	✓
r_a	mm	9×10^{-2}	2.5×10^{-15}	9×10^{-2} – 9×10^{-2}	✓	✓	✓

Continued on next page

Table A.2 – continued from previous page

Parameter	Unit	Mean	Standard Deviation	Value Range	Single V Magnet Topology	Double V Magnet Topology	Nabla Magnet Topology
r_i	mm	6.4433×10^{-2}	9.02×10^{-4}	6.4×10^{-2} – 6.7×10^{-2}	✓	✓	✓
Rotor Parameters							
rad_phiv2	rad	-0.5	6.36×10^{-2}	-0.6 – 0	×	✓	×
lmsov2	mm	-2.8×10^{-4}	3.5×10^{-5}	-3×10^{-4} – 0	×	✓	×
lth1v2	mm	5.39×10^{-3}	3.5×10^{-4}	0 – 5.45×10^{-3}	×	✓	×
lth2v2	mm	2.78×10^{-3}	1.7×10^{-4}	0 – 2.8×10^{-3}	×	✓	×
r1v2	mm	2.09×10^{-3}	3.22×10^{-4}	0 – 2.2×10^{-3}	×	✓	×
r11v2	mm	3.26×10^{-4}	4×10^{-5}	0 – 6×10^{-4}	×	✓	×
r2v2	mm	1.8×10^{-3}	1.33×10^{-4}	0 – 1.9×10^{-3}	×	✓	×
r3v2	mm	6.97×10^{-4}	4.4×10^{-5}	0 – 7×10^{-4}	×	✓	×
r4v2	mm	7.47×10^{-4}	4.8×10^{-5}	0 – 7.5×10^{-4}	×	✓	×
rmt1v2	mm	2.49×10^{-4}	1.6×10^{-5}	0 – 2.5×10^{-4}	×	✓	×
rmt4v2	mm	2.49×10^{-4}	1.6×10^{-5}	0 – 2.5×10^{-4}	×	✓	×
rlt1v2	mm	1.85×10^{-4}	4.5×10^{-5}	0 – 2×10^{-4}	×	✓	×
rlt4v2	mm	2.49×10^{-4}	1.6×10^{-5}	0 – 2.5×10^{-4}	×	✓	×
hav2	mm	4.9×10^{-3}	3.36×10^{-4}	0 – 5×10^{-3}	×	✓	×
mbv2	mm	1.7×10^{-2}	1.17×10^{-3}	0 – 1.8×10^{-2}	×	✓	×
mhv2	mm	3.6×10^{-3}	2.65×10^{-4}	0 – 3.8×10^{-3}	×	✓	×
rmagv2	mm	4.98×10^{-4}	3.2×10^{-5}	0 – 5×10^{-4}	×	✓	×
dsmv2	mm	2.9×10^{-3}	1.9×10^{-4}	0 – 3.1×10^{-3}	×	✓	×
dsmuv2	mm	2.9×10^{-3}	1.9×10^{-4}	0 – 3.1×10^{-3}	×	✓	×
amtrv2	mm	1.58×10^{-2}	1.024×10^{-3}	0 – 1.6×10^{-2}	×	✓	×
dsrv2	mm	9.96×10^{-4}	6.4×10^{-5}	0 – 1×10^{-3}	×	✓	×
lmav2	mm	1×10^{-4}	3×10^{-5}	0 – 1.1×10^{-4}	×	✓	×
lmiv2	mm	1.09×10^{-4}	8×10^{-6}	0 – 1.10×10^{-4}	×	✓	×
lmov2	mm	5.5×10^{-5}	1.5×10^{-5}	0 – 1×10^{-4}	×	✓	×
lmuv2	mm	1.45×10^{-4}	10	0 – 1.5×10^{-4}	×	✓	×
rad_phiv1	rad	-6.9×10^{-1}	5.4×10^{-2}	-7.8×10^{-1} – -4.5×10^{-1}	✓	✓	✓
lmsov1	mm	-5.01×10^{-4}	9.4×10^{-5}	-5.3×10^{-4} – 5×10^{-4}	✓	✓	✓
lth1v1	mm	2.8×10^{-3}	1.7×10^{-4}	2.8×10^{-3} – 5.45×10^{-3}	✓	✓	✓
lth2v1	mm	2.104×10^{-3}	5.9×10^{-5}	2.1×10^{-3} – 3.2×10^{-3}	✓	✓	✓
r1v1	mm	4.07×10^{-4}	1.08×10^{-4}	4×10^{-4} – 2.2×10^{-3}	✓	✓	✓
r11v1	mm	2.19×10^{-4}	4.5×10^{-5}	1×10^{-4} – 6×10^{-4}	✓	✓	✓
r2v1	mm	2.16×10^{-4}	1×10^{-4}	2×10^{-4} – 1.9×10^{-3}	✓	✓	✓
r3v1	mm	8.99×10^{-4}	1.3×10^{-5}	7×10^{-4} – 9×10^{-4}	✓	✓	✓
r4v1	mm	5.01×10^{-4}	1.6×10^{-5}	5×10^{-4} – 7.5×10^{-4}	✓	✓	✓
rmt1v1	mm	2.5×10^{-4}	8.8×10^{-18}	2.5×10^{-4} – 2.5×10^{-4}	✓	✓	✓
rmt4v1	mm	2.5×10^{-4}	8.8×10^{-18}	2.5×10^{-4} – 2.5×10^{-4}	✓	✓	✓
rlt1v1	mm	1.17×10^{-4}	5.6×10^{-5}	5×10^{-5} – 2.5×10^{-4}	✓	✓	✓

Continued on next page

Table A.2 – continued from previous page

Parameter	Unit	Mean	Standard Deviation	Value Range	Single V Magnet Topology	Double V Magnet Topology	Nabla Magnet Topology
rlt4v1	mm	2.5×10^{-4}	8.8×10^{-18}	2.5×10^{-4} –2.5×10^{-4}	✓	✓	✓
hav1	mm	2.9×10^{-3}	1.36×10^{-4}	2.9×10^{-3}– 5×10^{-3}	✓	✓	✓
mbv1	mm	7.6×10^{-3}	5.8×10^{-4}	7.5×10^{-3} – 1.8×10^{-2}	✓	✓	✓
mhv1	mm	2.8×10^{-3}	1.4×10^{-4}	2.7×10^{-3} – 5×10^{-3}	✓	✓	✓
rmagv1	mm	5×10^{-4}	1.7×10^{-17}	5×10^{-4} – 5×10^{-4}	✓	✓	✓
dsmv1	mm	1.07×10^{-3}	1.41×10^{-4}	8×10^{-4} – 2.8×10^{-3}	✓	✓	✓
dsmuv1	mm	1.07×10^{-3}	1.47×10^{-4}	8×10^{-4} – 2.92×10^{-3}	✓	✓	✓
amtrv1	mm	5.5×10^{-3}	6.34×10^{-4}	5.5×10^{-3} – 1.9×10^{-2}	✓	✓	✓
dsrv1	mm	7.5×10^{-4}	3.2×10^{-5}	7.5×10^{-4} – 1.25×10^{-3}	✓	✓	✓
lmav1	mm	9.2×10^{-5}	2.6×10^{-5}	1×10^{-5} – 1.1×10^{-4}	✓	✓	✓
lmiv1	mm	1×10^{-4}	6.35×10^{-7}	1×10^{-4} – 1.1×10^{-4}	✓	✓	✓
lmov1	mm	5.5×10^{-5}	1.5×10^{-5}	5×10^{-5} – 1×10^{-4}	✓	✓	✓
lmuv1	mm	1.45×10^{-4}	1.5×10^{-5}	1×10^{-4} – 1.5×10^{-4}	✓	✓	✓
rad_phi3b1	rad	- 2.5×10^{-3}	5.6001×10^{-2}	-1.25 – 0	×	×	✓
rad_phi4b1	rad	-5.3×10^{-4}	1.1775×10^{-2}	-0.26 – 0	×	×	✓
lmsob1	mm	2×10^{-6}	3.4×10^{-5}	0 – 7.5×10^{-4}	×	×	✓
lthb1	mm	6×10^{-6}	1.28×10^{-4}	0 – 2.9×10^{-3}	×	×	✓
r2b1	mm	2×10^{-6}	4.5×10^{-5}	0 – 1×10^{-3}	×	×	✓
r3b1	mm	2×10^{-6}	3.4×10^{-5}	0 – 1×10^{-3}	×	×	✓
r4b1	mm	5.06×10^{-7}	1.1×10^{-5}	0 – 2.5×10^{-4}	×	×	✓
r5b1	mm	5.06×10^{-7}	1.1×10^{-5}	0 – 2.5×10^{-4}	×	×	✓
lgr3b1	mm	1×10^{-6}	2.2×10^{-5}	0 – 5×10^{-4}	×	×	✓
lgr4b1	mm	6.07×10^{-7}	1.3×10^{-5}	0 – 3×10^{-4}	×	×	✓
mbb1	mm	3×10^{-5}	6.75×10^{-4}	0 – 1.5×10^{-2}	×	×	✓
mhb1	mm	6×10^{-6}	1.4×10^{-4}	0 – 3.2×10^{-3}	×	×	✓
mtbb1	mm	3×10^{-5}	6.75×10^{-4}	0 – 1.5×10^{-2}	×	×	✓
rmagb1	mm	1×10^{-6}	2.2×10^{-5}	0 – 5×10^{-4}	×	×	✓
amtrb1	mm	4×10^{-6}	9.8×10^{-5}	0 – 2.5×10^{-3}	×	×	✓
dsr3b1	mm	3×10^{-6}	6.6×10^{-5}	0 – 1.85×10^{-3}	×	×	✓
dsr4b1	mm	4×10^{-6}	8.3×10^{-5}	0 – 1.85×10^{-3}	×	×	✓
lmob1	mm	2.02×10^{-7}	4.49×10^{-6}	0 – 1×10^{-4}	×	×	✓
lmub1	mm	3.03×10^{-7}	6.7×10^{-6}	0 – 1.5×10^{-4}	×	×	✓
lmsub1	mm	4×10^{-6}	8.1×10^{-5}	0 – 1.8×10^{-3}	×	×	✓
Stator Parameters							
airgap	mm	1×10^{-3}	3.5×10^{-17}	1×10^{-3} – 1×10^{-3}	✓	✓	✓
b_nng	mm	5.04×10^{-3}	9.1×10^{-5}	4.6×10^{-3} – 5.1×10^{-3}	✓	✓	✓
b_nzk	mm	4.5×10^{-3}	5.7×10^{-5}	4.45×10^{-3} – 4.6×10^{-3}	✓	✓	✓

Continued on next page

Table A.2 – continued from previous page

Parameter	Unit	Mean	Standard Deviation	Value Range	Single V Magnet Topology	Double V Magnet Topology	Nabla Magnet Topology
b_s	mm	1×10^{-3}	2.5×10^{-5}	$1 \times 10^{-3} - 1.4 \times 10^{-3}$	✓	✓	✓
h_n	mm	1.1×10^{-2}	8.06×10^{-4}	$9.2 \times 10^{-3} - 1.39 \times 10^{-2}$	✓	✓	✓
h_s	mm	1×10^{-3}	3.5×10^{-17}	$1 \times 10^{-3} - 1 \times 10^{-3}$	✓	✓	✓
r_sn	mm	2.5×10^{-4}	8.8×10^{-18}	$2.5 \times 10^{-4} - 2.5 \times 10^{-4}$	✓	✓	✓
r_zk	mm	5.01×10^{-4}	1.9×10^{-5}	$5 \times 10^{-4} - 8 \times 10^{-4}$	✓	✓	✓
r_ng	mm	5.01×10^{-4}	1.9×10^{-5}	$5 \times 10^{-4} - 8 \times 10^{-4}$	✓	✓	✓
h_zk	mm	1×10^{-3}	3.5×10^{-17}	$1 \times 10^{-3} - 1 \times 10^{-3}$	✓	✓	✓
bhp	mm	3.7×10^{-3}	8.1×10^{-5}	$3.5 \times 10^{-3} - 3.8 \times 10^{-3}$	✓	✓	✓
hhp	mm	2.38×10^{-3}	1.9×10^{-4}	$1.9 \times 10^{-3} - 2.8 \times 10^{-3}$	✓	✓	✓
rhp	mm	6.36×10^{-4}	4.5×10^{-5}	$5 \times 10^{-4} - 8 \times 10^{-4}$	✓	✓	✓
dhphp	mm	2.6×10^{-4}	1.4×10^{-5}	$2.6 \times 10^{-4} - 4.8 \times 10^{-4}$	✓	✓	✓
dhpng	mm	4.06×10^{-4}	3×10^{-6}	$4.06 \times 10^{-4} - 4.5 \times 10^{-4}$	✓	✓	✓

Table A.2: EM Input Parameters

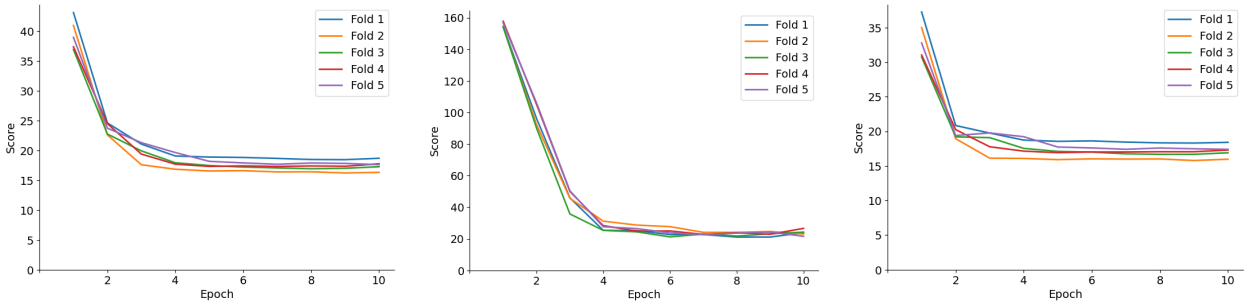
It gives us a clear idea of how the parameters of the Rotor can vary across Topologies. In addition the N and $simQ$ parameters are count of Stator poles and its windings respectively and may vary across EM variants.
1481 examples are of the Double V Magnet Topology apart from which 3 examples each for the other 2 topologies. Hence, the imbalanced topologies parameters would have relatively not so reliable statistics.

A.2 Technical Details

Reading each sheet from the excel files particularly the grids take up a lot of time and compute, hence I read the files as a onetime job when creating the tabular data for training and store them into pythonic objects for faster access for training. Both the input and target values for the Torque KPI are stored locally as csv files whereas those of the Efficiency KPI is stored as separate csv files per variant considering it is in the form of a 2D array. The csv files are then concatenated and stored into an array conserving dimensionality by padding NaN values to match dimensionality of the grid corresponding to the Torque KPI with the largest torque value. In my case the value is 280 computed internally from the dataset but this is subject to change as I receive more data and there is a provision to override it on demand. The array is then saved locally for easy access and loading during training.

A.3 Experimental setup

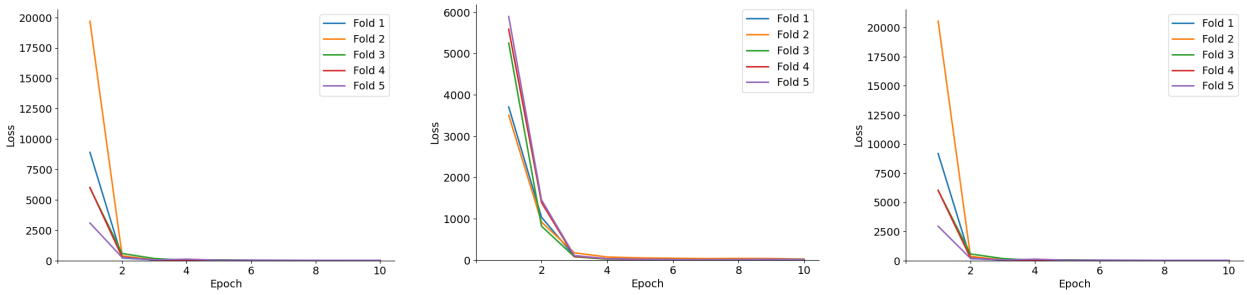
A.3.1 Monitoring Plots



(a) Aggregated (b) Torque KPI (c) Efficiency KPI

Figure A.1: Training Evaluation Metrics

Fig. A.1 illustrates the evaluation metrics for both the KPIs and the aggregated loss during training. Fig. A.1 shows that evaluation during training also does not get better after a threshold for both the KPIs particularly the Efficiency KPI which is the ultimate focus of this work.



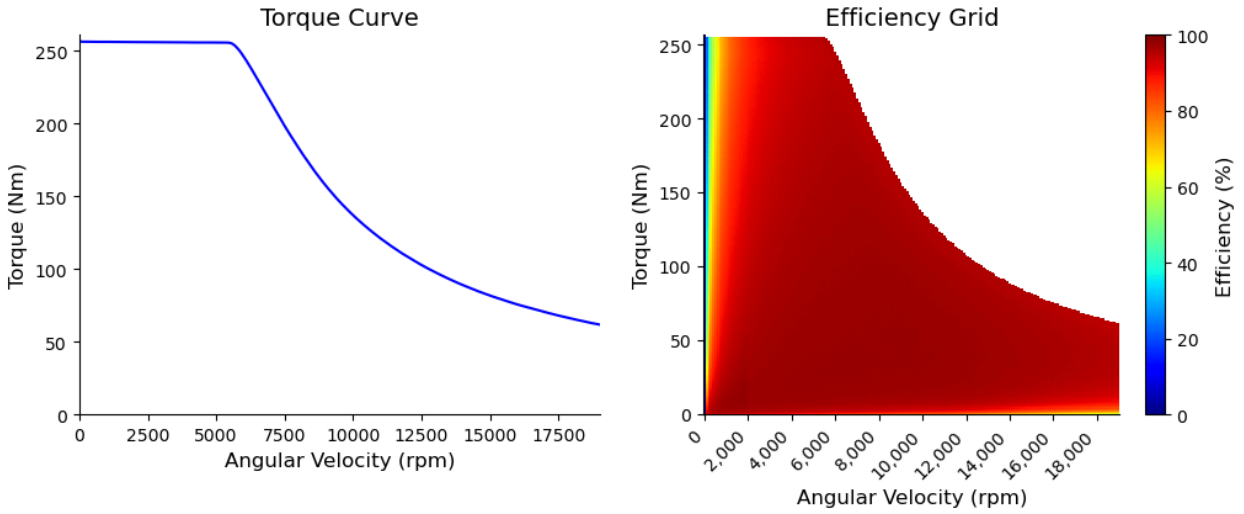
(a) Aggregated (b) Torque KPI (c) Efficiency KPI

Figure A.2: Validation Loss Metrics

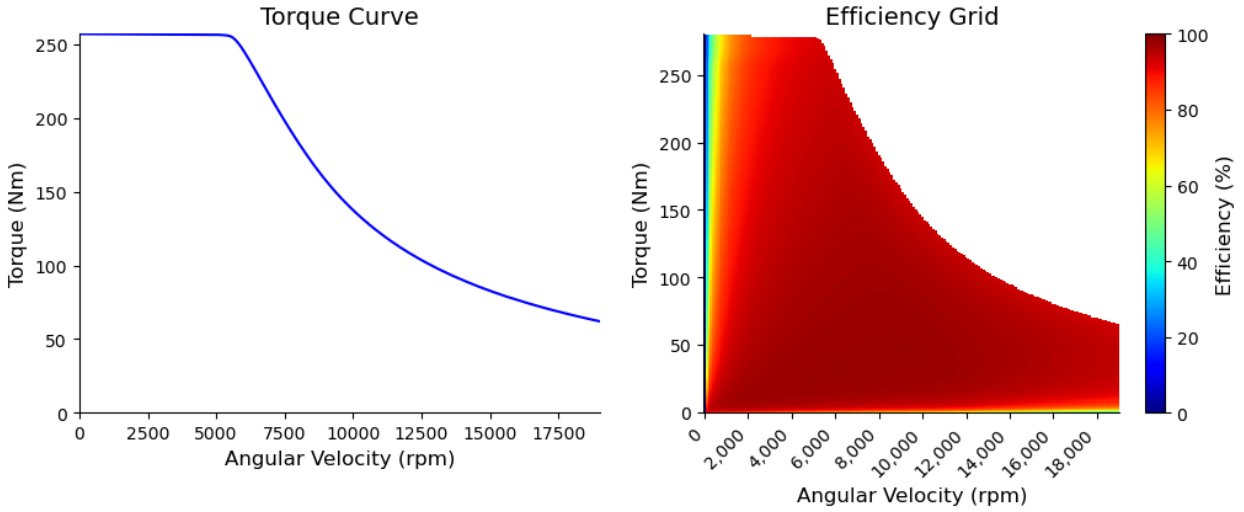
Fig. A.2 illustrates the loss functions at work for both the KPIs and the aggregated loss. It indicates that the losses reduce for both the KPIs of the validation dataset which is 20% of the whole dataset. The loss also starts off high and tapers off quickly although due to the value ranges in y-axis, it appears to still improve very slightly before plateauing.

A.3.2 Ablation Studies Results

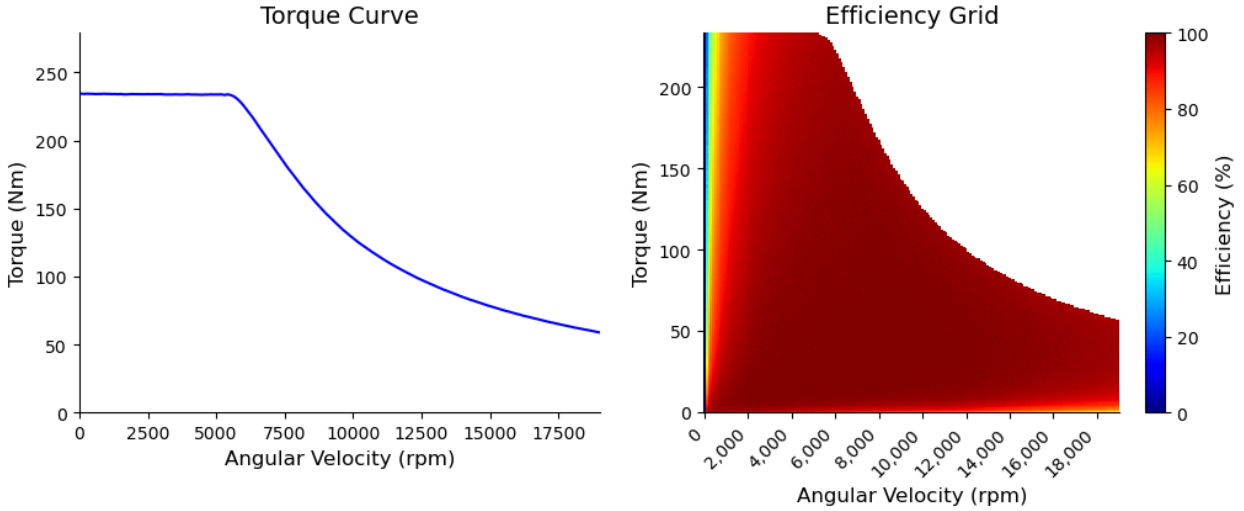
I have generated predictions for each of the models for a unique sample and visualize them below for comparision with its ground truth.



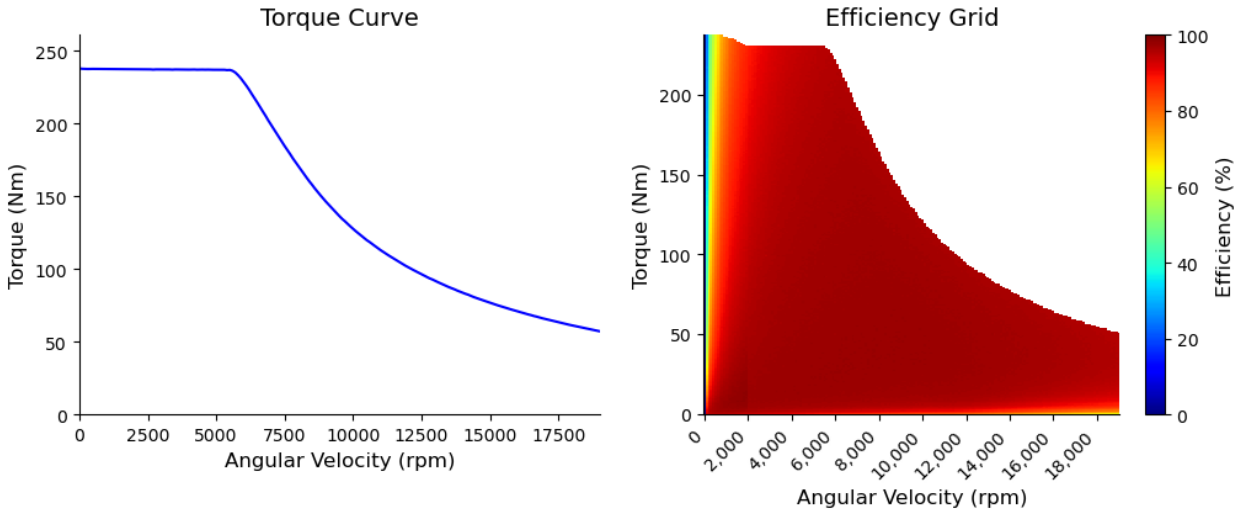
(a) KPI Predictions with MLP Efficiency KPI Regularization



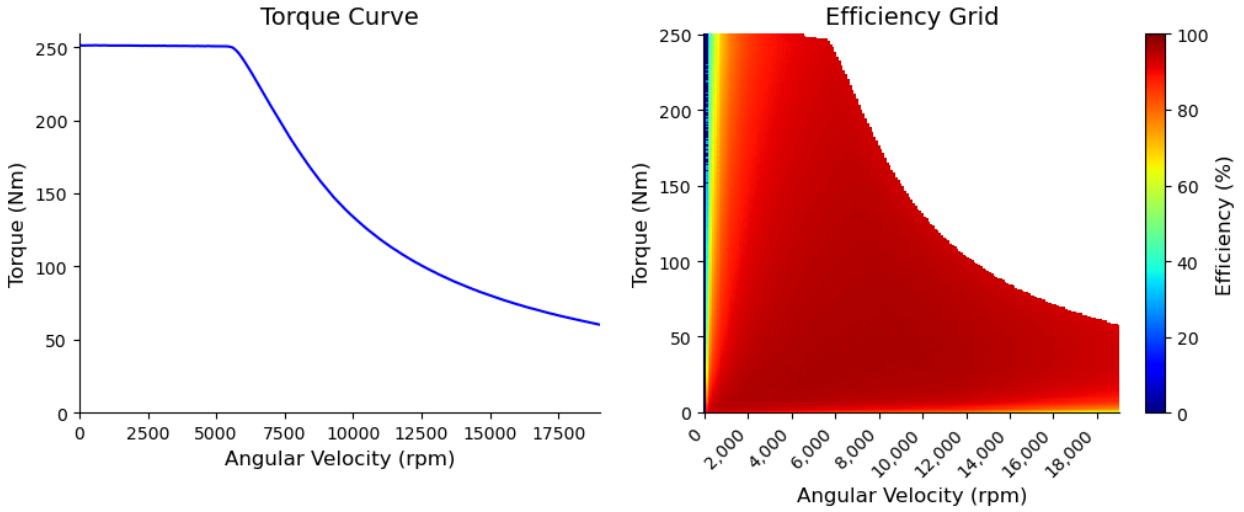
(b) KPI Predictions with Baseline



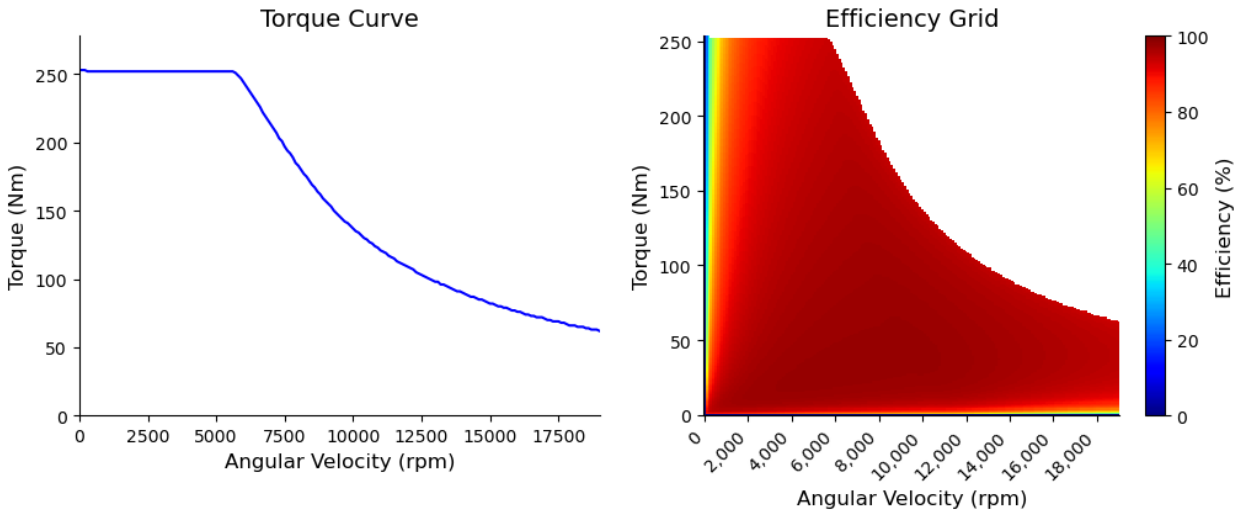
(c) KPI Predictions with Smoothening curve Loss Regularization



(d) KPI Predictions with Decreasing curve Loss Regularization



(e) KPI Predictions with no Loss Regularization



(f) Target KPIs

Figure A.3: KPI Predictions for all Models of the same sample

A.4 GNN Terminologies

I have largely adopt the commonly used notations from Ref. [18] and reformulate it slightly to meet my needs. I use bold uppercase characters to denote matrices and bold lowercase characters to denote vectors.

Graphs
A graph is represented as $\mathcal{G} = (V, E)$ where V is the set of vertices or nodes and E is the set of edges. Let $v_i \in V$ denote a node and $e_{ij} = (v_i, v_j) \in E$ denote an edge pointing from v_j to v_i. A graph may have node attributes X, where $X \in \mathbb{R}^{n \times d}$ is a node feature matrix with $x_v \in \mathbb{R}^d$ representing the feature vector of a node v. Meanwhile, a graph may have edge attributes X_e, where $X_e \in \mathbb{R}^{m \times c}$ is an edge feature matrix with $x_{v,u} \in \mathbb{R}^c$ representing the feature vector of an edge (v, u).

Neighbourhood
Neighbourhood essentially holds the information of the set of nodes that are in the vicinity of a node in question. The neighborhood of a node v is defined as $N(v) = \{u \in V \mid (v, u) \in E\}$.

Adjacency Matrix
Adjacency matrix gives an overview of the neighbourhood of all nodes in the graph. It is typically referenced when performing MP algorithm. The adjacency matrix A is a $n \times n$ matrix with $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$.

Diagonal Matrix
Diagonal Matrix of A denoted by $D = \text{diag}(d_1, d_2, \ldots, d_n)$, where $d_i = \sum_j a_{ij}$ is the degree of vertex i.

Laplacian Matrix
The graph Laplacian is defined as $L := D - A$
and the normalized Adjacency Matrix can be defined as

$$A_{\text{norm}} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}.$$

Reference Operator
The Reference Operator R is a matrix of the same sparsity as the Adjacency Matrix A. It is also called as Structure or Graph Shift operator.

Transformation Operator
The Transformation Operator is a feature map or a learnable matrix of weights, $\theta \in \mathbb{R}^{F \times F'}$:

Message Passing
For Graph signals, $V = [v_1, v_2, \ldots, v_n], v_i \in \mathbb{R}^F$, the updated Graph Signals is $V' = RV\theta$ where RV is the weighted sum of graph signals neighbourhood. RV also ensures only nodes with edges is considered.

In contrast to CNNs fixed shape kernel which considers pixels locality, GNNs have an increasing shaped kernel. This is clear from the Fig. A.4. The fixed shaped kernel does not work for graphs as they are irregular in size and not constrained to pixel dimensions as images are. It is also called a Polynomial Graph Filter or Radial Filter and can be expressed as $V' = \sigma(\sum_{k=0}^{K} R^k V \theta^k)$

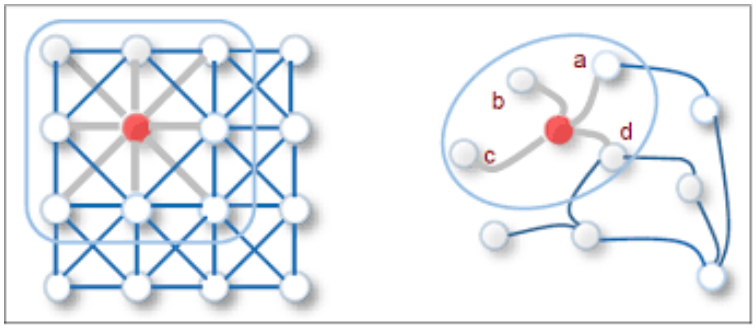


Figure A.4: GNN Kernel Filter (Source : [18])

A GNN block contains three update functions, ϕ, and three aggregation functions, ρ for the nodes, edges and graph attributes, Ref. [19]

$e'_k = \phi_e(e_k, v_{r_k}, v_{s_k}, u)$, $v'_i = \phi_v(\bar{e}'_i, v_i, u)$, $u' = \phi_u(\bar{e}', \bar{v}', u)$

$\bar{e}'_i = \rho_{e\to v}(E'_i)$, $\bar{e}' = \rho_{e\to u}(E')$, $\bar{v}' = \rho_{v\to u}(V')$

A huge benefit of GNNs in practical use cases is that they are inductive rather than transductive. While transductive model can only be used on the specific data that was present during training, inductive models can be applied to entirely new data without having to be retrained. This is defended by Johannessen *et al.* in Ref. [05].

A.5 Heterogeneous Graph Neural Networks

1. Node Types

   $T_v = \{v, vm, r, s, sw\}$

   The node type mapping functions of each node type in the graph are as follows :

Node Type (ν)	**Nodes (v^ν)**	**Node Features(X^ν)**
Rotor Airgap (ν_v)	v11, v12, v21, v22	lmsov, lth1v, lth2v, r1v, r11v, r2v, r3v, r4v, rmt1v, rmt4v, rlt1v, rlt4v, hav, lmsov1, lth1v1, lth2v1, r1v1, r11v1, r2v1, r3v1, r4v1, rmt1v1, rmt4v1, rlt1v1, rlt4v1, hav1
Rotor Magnet (ν_{vm})	v1m1, v1m2, v2m1, v2m2	mbv, mhv, rmagv, mbv1, mhv1, rmagv1
Radius (ν_r)	rr, ra, o	0
Stator Poles (ν_s)	s1, s2, s3, s4, s5, s6	b_nng, b_nzk, b_s, h_n, h_s, r_sn, r_zk, r_ng
Stator Windings (ν_{sw})	s1w1, s1w2, s1w3, s1w4, s2w1, s2w2, s2w3, s2w4, s3w1, s3w2, s3w3, s3w4, s4w1, s4w2, s4w3, s4w4, s5w1, s5w2, s5w3, s5w4, s6w1, s6w2, s6w3, s6w4	bhp, hhp, rhp

Table A.3: GNN Node Types

2. Edge Types

   $T_e = \{a, d1, d2, d4\}$

   The edge type mapping functions of each edge type in the graph are as follows:

Edge Type (ϵ)	**Edges** (e^{ϵ})	**Edge Features** (R^{ϵ})
Angular/Radian Edges (ϵ_a)	$e_{v1m1,v1m2}$ $e_{v2m1,v2m2}$	deg_phiv1 deg_phiv2
Distance Edge with 1 feature (ϵ_{d1})	$e_{v21,rr}$, $e_{v22,rr}$ $e_{rr,s1}$, $e_{rr,s2}$, $e_{rr,s3}$, $e_{rr,s4}$, $e_{rr,s5}$, $e_{rr,s6}$ $e_{s1,s1w1}$, $e_{s1,s1w2}$, $e_{s1,s1w3}$, $e_{s1,s1w4}$, $e_{s2,s2w1}$, $e_{s2,s2w2}$, $e_{s2,s2w3}$, $e_{s2,s2w4}$, $e_{s3,s3w1}$, $e_{s3,s3w2}$, $e_{s3,s3w3}$, $e_{s3,s3w4}$, $e_{s4,s4w1}$, $e_{s4,s4w2}$, $e_{s4,s4w3}$, $e_{s4,s4w4}$, $e_{s5,s5w1}$, $e_{s5,s5w2}$, $e_{s5,s5w3}$, $e_{s5,s5w4}$, $e_{s6,s6w1}$, $e_{s6,s6w2}$, $e_{s6,s6w3}$, $e_{s6,s6w4}$ $e_{s1w1,s1w2}$, $e_{s1w2,s1w3}$, $e_{s1w3,s1w4}$, $e_{s2w1,s2w2}$, $e_{s2w2,s2w3}$, $e_{s2w3,s2w4}$, $e_{s3w1,s3w2}$, $e_{s3w2,s3w3}$, $e_{s3w3,s3w4}$, $e_{s4w1,s4w2}$, $e_{s4w2,s4w3}$, $e_{s4w3,s4w4}$, $e_{s5w1,s5w2}$, $e_{s5w2,s5w3}$, $e_{s5w3,s5w4}$, $e_{s6w1,s6w2}$, $e_{s6w2,s6w3}$, $e_{s6w3,s6w4}$ $e_{o,ra}$	dsrv2 airgap dhphp dhpng r_a
Distance Edge with 2 features (ϵ_{d2})	$e_{v11,v12}$, $e_{v21,v22}$ $e_{v11,rr}$, $e_{v12,rr}$ $e_{o,rr}$	dsmv1, dsmuv1 amtrv1, dsrv1 $r_i - airgap$
Distance Edge with 4 features (ϵ_{d4})	$e_{v11,v1m1}$, $e_{v12,v1m2}$ $e_{v21,v2m1}$, $e_{v22,v2m2}$ $e_{s1,ra}$, $e_{s2,ra}$, $e_{s3,ra}$, $e_{s4,ra}$, $e_{s5,ra}$, $e_{s6,ra}$	lmav1, lmiv1, lmov1, lmuv1 lmav2, lmiv2, lmov2, lmuv2 $r_a - (r_i + h_n, h_z k)$

Table A.4: GNN Edge Types

I have define the meta relations as well for each of the edge type.

Fig. A.5 shows the heterogeneous graph I had constructed

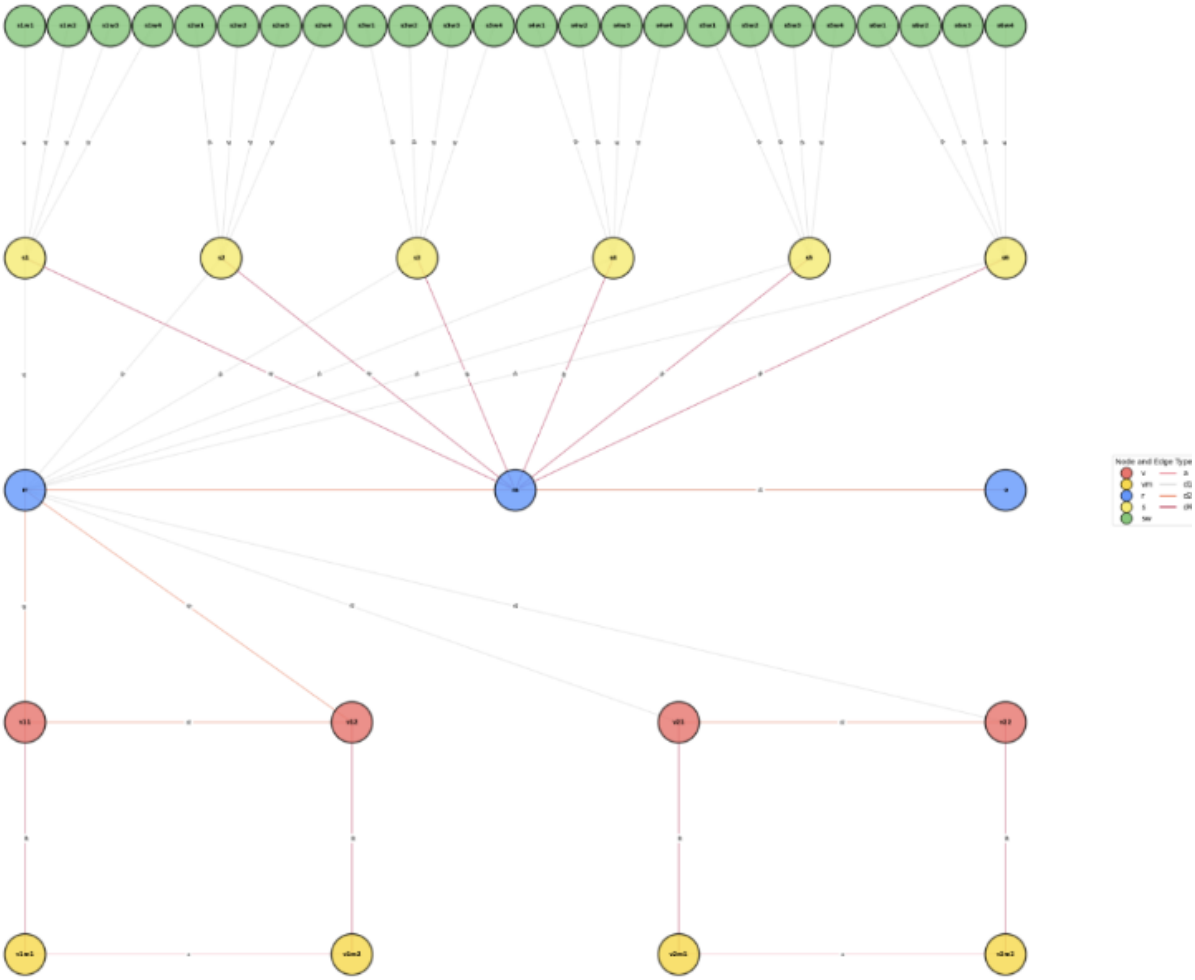


Figure A.5: HetGraph

List of Figures

List of Tables

Bibliography

[01] Tsai Hor Chana, Guosheng Yina, Kyongtae Baeb, Lequan Yu *Multi-task Heterogeneous Graph Learning on Electronic Health Records.* cs.LG, 14 Aug 2024.

[02] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, Dongrui Fan *Simple and Efficient Heterogeneous Graph Neural Network.* cs.LG, 1 Sep 2023.

[03] Xinru Huang *A Heterogeneous Graph Neural Network Model for Electric Vehicle Purchase Propensity Prediction.* ICDSCA, Oct 2023.

[04] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, Jie Tang *Are we really making much progress? Revisiting, benchmarking, and refining heterogeneous graph neural networks.* cs.LG, 30 Dec 2021.

[05] Fredrik Johannessen, Martin Jullum *Finding Money Launderers Using Heterogeneous Graph Neural Networks.* ICDSCA, 25 July 2023.

[06] Damian Owerko, Fernando Gama, Alejandro Ribeiro *Predicting Power Outages Using Graph Neural Networks.* cs.LG, Nov 2018.

[07] Mikko Tahkola, Janne Keränen, Denis Sedov, Mehrnaz Farzam Far, Juha Kortelainen *Surrogate Modeling of Electrical Machine Torque Using Artificial Neural Networks.* ICDSCA, Dec 17 2020.

[08] AKM Khaled Ahsan Talukder, Bingnan Wang and Yusuke Sakamoto *Electric Machine Two-dimensional Flux Map Prediction with Ensemble Learning.* ICEMS, 2022.

[09] Dario Pasqualotto, Mauro Zigliotto *A comprehensive approach to convolutional neural networks-based condition monitoring of permanent magnet synchronous motor drives.* May 20, 2020.

[10] Yihao Xu, Bingnan Wang, Yusuke Sakamoto, Tatsuya Yamamoto, and Yuki Nishimura *Comparison of Learning-based Surrogate Models for Electric Motors.* COMPUMAG, 2023.

[11] Yang Gao, Peng Zhang, Zhao Li, Chuan Zhou, Yongchao Liu, Yue Hu *Heterogeneous Graph Neural Architecture Search.* ICDM, Dec, 2021.

[12] Linhao Luo, Yixiang Fang, Moli Lu, Xin Cao, Xiaofeng Zhang, Wenjie Zhang *GSim: A Graph Neural Network Based Relevance Measure for Heterogeneous Graphs.* Dec, 2023.

[13] Mengya Guan, Xinjun Cai, Jiaxing Shang, Fei Hao, Dajiang Liu, Xianlong Jiao, Wancheng Ni *HMSG: Heterogeneous graph neural network based on Metapath SubGraph learning.* 2023.

[14] Martin Juan José Bucher, Michael Anton Kraus, Romana Rust, Siyu Tang *Performance-Based Generative Design for Parametric Modeling of Engineering Structures Using Deep Conditional Generative Models.* Dec, 2023.

[15] Kazuhisa Iwata, Hidenori Sasaki, Hajime Igarashi, Daisuke Nakagawa, Tomoya Ueda *Generalization Performance in Predicting Torque Characteristics Using Convolutional Neural Network and Stator Magnetic Flux* 3 March, 2024.

[16] Arbaaz Khan, Mohammad Hossain Mohammadi, Vahid Ghorbanian, David Lowther *Transfer Learning for Efficiency Map Prediction.* CEFC, 2020.

[17] Ali Borji *Qualitative failures of image generation models and their application in detecting deepfakes.* cs.CV, 2023.

[18] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, Philip S. Yu *A Comprehensive Survey on Graph Neural Networks.* 04 Dec 2019.

[19] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, Razvan Pascanu *Relational inductive biases, deep learning, and graph networks.* 17 Oct 2018.

[20] Shichao Zhu, Chuan Zhou, Shirui Pan, Xingquan Zhu, Bin Wang *Relation Structure-Aware Heterogeneous Graph Neural Network.* 2019.

[21] Xingtong Yu, Yuan Fang, Zemin Liu, Xinming Zhang *HGPROMPT: Bridging Homogeneous and Heterogeneous Graphs for Few-shot Prompt Learning.* 5 Jan 2024.

[22] Qimai Li, Zhichao Han, Xiao-Ming Wu *Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning.* 22 Jan 2018.

[23] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Yizhou Sun *Heterogeneous Graph Transformer.* 3 Mar 2022.

[24] Yiling Zou, Jian Shu *Heterogeneous Network Node Classification Based on Graph Neural Networks.* IEEE, 2023.

[25] Joshua Melton, Siddharth Krishnan *muxGNN: Multiplex Graph Neural Network for Heterogeneous Graphs.* IEEE, 9 Sept 2023.

[26] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, Jiawei Han *Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark.* cs SI, 17 Dec 2020.

[27] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl *Neural Message Passing for Quantum Chemistry.* cs SI, 12 June 2017.

[28] Katsuyuki Narita, Hiroyuki Sano, Nicolas Schneider, Kazuki Semba, Koji Tani, Takashi Yamada, Ryosuke Akaki *An Accuracy Study of Finite Element Analysis based Efficiency Map for Traction Interior Permanent Magnet Machines.* IEMS, 2009.

[29] Arbaaz Khan, Mohammad Hossain Mohammadi, Vahid Ghorbanian, David Lowther *Efficiency Map Prediction of Motor Drives using Deep Learning.* IEMS, 12 July 2019.

[30] Sagar Verma, Nicolas Henwood, Marc Castella, Al Kassem Jebai, Jean-Christophe Pesquet *Neural Networks based Speed-Torque Estimators for Induction Motors and Performance Metrics.* 2020.

[31] H. Sano, K. Narita, N. Schneider, K. Semba, K. Tani, T. Yamada, R. Akaki *Loss Analysis of a Permanent Magnet Traction Motor in a Finite Element Analysis based Efficiency Map.* ICEM, 2020.

[32] Arbaaz Khan, Vahid Ghorbanian, David Lowther *Deep Learning for Magnetic Field Estimation.* 6 March 2019.

[33] Carlos Candelo-Zuluaga, Antonio Garcia Espinosa, Jordi-Roger Riba, Pere Tubert Blanch *Computationally Efficient Analysis of Spatial and Temporal Harmonics Content of the Magnetic Flux Distribution in a PMSM for Efficiency Maps Computation.* IEEE, 2020.

[34] Marius Stender, Oliver Wallscheid, Joachim Böcker *Accurate Torque Estimation for Induction Motors by Utilizing a Hybrid Machine Learning Approach.* IEEE, 2021.

[35] Simone Ferrari, Paolo Ragazzo, Gaetano Dilevrano, Gianmario Pellegrino *Flux-Map Based FEA Evaluation of Synchronous Machine Efficiency Maps.* WEMDCD, 2021.

[36] Vivek Parekh, Dominik Flore, and Sebastian Schöps *Variational Autoencoder based Metamodeling for Multi-Objective Topology Optimization of Electrical Machines.* cs.LG, 7 April 2022.

[37] Marius Benkert, Michael Heroth, Rainer Herrler, Magda Gregorová, Helmut C. Schmid *Variational autoencoder-based techniques for a streamlined cross-topology modeling and optimization workflow in electrical drives.* 24 May 2024.

[38] Yusuke Sakamoto, Yihao Xu, Bingnan Wang, Tatsuya Yamamoto, Yuki Nishimura *Electric Motor Surrogate Model Combining Subdomain Method and Neural Network.* COMPUMAG, 2023.

[39] Yusuke Sakamoto, Yihao Xu, Bingnan Wang, Tatsuya Yamamoto, Yuki Nishimura *Multi-Objective Motor Design Optimization with Physics-Assisted Neural Network Model.* IEMDC, 2023.

[40] Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun, Jie Tang *Heterogeneous Network Representation Learning.* IJCAI, 2020.

[41] Hongjoon Ahn, Yongyi Yang, Quan Gan, Taesup Moon, David Wipf *Descent Steps of a Relation-Aware Energy Produce Heterogeneous Graph Neural Networks.* NeurIPS, 2022.

[42] Lingfan Yu, Jiajun Shen, Jinyang Li, Adam Lerer *Scalable Graph Neural Networks for Heterogeneous Graphs.* cs.LG, 19 Nov 2020.

[43] Lei Xu, Zhen-Yu He, Kai Wang, Chang-Dong Wang, Shu-Qiang Huang *Explicit Message-Passing Heterogeneous Graph Neural Network.* 7 July 2023.

Declaration on oath

I hereby certify that I have written my master thesis independently and have not yet submitted it for examination purposes elsewhere. All sources and aids used are listed, literal and meaningful quotations have been marked as such.

Lilly Abraham K64889, 11.12.2024

Consent to Plagiarism Check

I hereby agree that my submitted work may be sent to PlagScan (www.plagscan.com) in digital form for the purpose of checking for plagiarism and that it may be temporarily (max. 5 years) stored in the database maintained by PlagScan as well as personal data which are part of this work may be stored there.

Consent is voluntary. Without this consent, the plagiarism check cannot be prevented by removing all personal data and protecting the copyright requirements. Consent to the storage and use of personal data may be revoked at any time by notifying the faculty.

Lilly Abraham K64889, 11.12.2024