

**Technical University of Applied Sciences Würzburg-Schweinfurt
(THWS)**

Faculty of Computer Science and Business Information Systems

Master Thesis

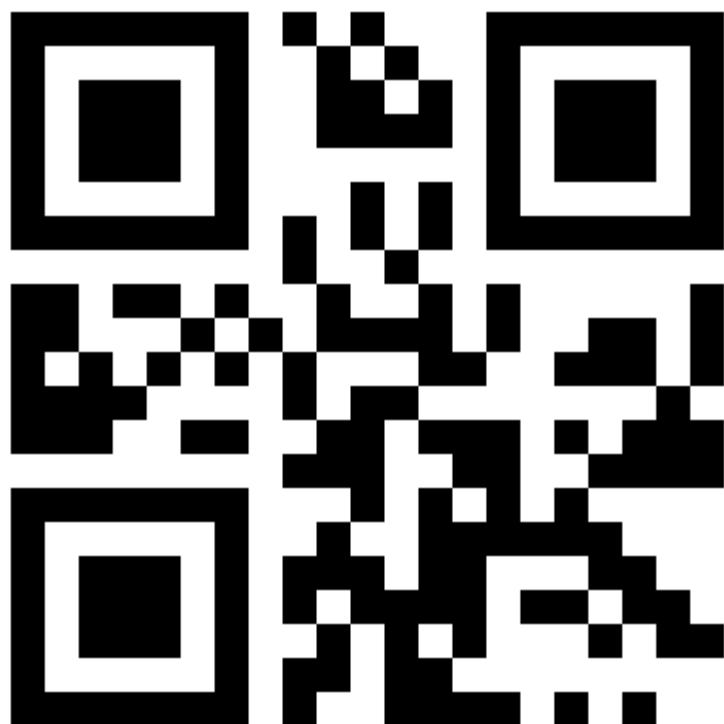
Electrical Engine Efficiency Prediction Bypassing Finite Element Analysis

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. Gracia Herranz Mercedes



Abstract

The thesis explores an approach to predict Key Performance Indicators (KPI)s of topology invariant Interior Permanent Magnet Synchronous Motors (PMSM) Electric Motors from its geometric, physical and simulation parameters. We intend to model the dynamics of Electric Motor functionality by creating surrogate models trained with Finite Element Method (FEM) simulations from its parametric description. The KPIs to be predicted are vectors of numerical values which can be displayed as the Torque curve(2D) and the Efficiency grid(3D) respectively. We note the relationship between the Torque curve and the Efficiency grid and incorporate our learnings of both the KPI's nature into our solutions's modelling.

We aim to first parameterize the Electric Motor design such that it is feasible to convert into a tabular representation.

Next, we would create a table with relevant attributes and design a Multi Linear Perceptron(MLP) to train it in a supervised manner.

Subsequently we will regularize the loss functions in a way that would smoothen out the plot curves for both the KPIs.

Then we will evaluate the predictions with the test target values by experimenting with various hyperparameter tuning settings and as a baseline with the average of the parameters.

Additionally we conduct a study to model the task as its graph representation and use Graph Neural Networks(GNN) to predict the KPIs.

Lastly we will enable the KPI's plot visualisation in a manner presentable to the client Valeo(Automaker Company specialiyes in manufacturing 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 this thesis. Her dedication and commitment to our work has been inspiring to me especially 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 Mother and my Siblings my accomplishments. They have been very instrumental in making it possible for me to pursue a Master's degree outside my home country and for the endless support throughout. 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

Abstract	2
Acknowledgement	3
Contents	4
Abbreviations	6
1 Introduction	7
1.1 Objective	10
1.2 Motivation	11
1.3 Problem Statement	11
1.4 Research Question	11
1.5 Thesis Structure	11
2 Literature Review	12
3 Dataset	15
3.1 Data Preprocessing for Multi Linear Perceptron (MLP)	16
3.1.1 Data Exploration of the Input Parameters	16
3.1.2 Data Exploration of the Torque Key Performance Indicator (KPI)(Torque Curve) . .	16
3.1.3 Data Exploration of the Efficiency KPI(Efficiency Grid)	17
3.2 Scaling	19
3.3 Dataset splitting	20
4 Modelling & Evaluation	21
4.1 MLP Model	21
4.2 Loss Functions	22
4.2.1 Loss for Torque KPI(Torque curve)	23
4.2.2 Loss for Efficiency KPI(Efficiency Grid)	23
4.3 Optimizer	26
4.4 Evaluation Metrics	26
4.4.1 Evaluation Metrics for Torque KPI	26
4.4.2 Evaluation Metrics for Efficiency KPI	26
4.5 Post Processing	27

5 Experiments and Results	28
5.1 Experiments with MLP	28
5.2 Results with MLP	31
5.2.1 Torque KPI Results with MLP	31
5.2.2 Efficiency KPI Results with MLP	33
5.3 Results with Baseline	38
5.3.1 Torque KPI Results with Baseline	38
5.3.2 Efficiency KPI Results with Baseline	39
5.4 Results with Smoothening Loss Regularization	40
5.4.1 Torque KPI Results with Smoothening Loss Regularization	40
5.5 Results with No Loss Regularization	41
5.5.1 Torque KPI Results with No Loss Regularization	41
5.5.2 Efficiency KPI Results with No Loss Regularization	42
5.6 Ablation Studies	42
6 Graph Modelling	44
6.1 Introduction	44
6.1.1 Applications	45
6.2 Heterogeneous Graph Neural Network (GNN)	46
6.3 GNN Literature Review	47
6.4 Electric Motor (EM) Heterogeneous GNN Model	48
6.4.1 EM Heterogeneous Graph Construction	49
7 Conclusion	53
7.1 Future Improvements	53
A Appendix	54
A.1 Dataset Supplementary Information	54
A.1.1 File structure	54
A.1.2 Data Summary Statistics	54
A.2 Experimental setup	57
A.3 GNN Terminologies	58
List of Figures	60
List of Tables	61
Bibliography	62
Declaration on oath	65
Consent to Plagiarism Check	66

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	Dimension
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 Motor

Chapter 1

Introduction

In the design of Electric Motor (EM)s, vast amounts of data are generated to determine which design of an EM fits best to Key Performance Indicator (KPI)s. Permanent Magnet Synchronous Motor (PMSM)s are largely classified as Surface Mounted PMSM and Interior PMSM.

KPIs of an EM are its representative characteristics of a motor drive and is essential to judge the performance of the motor before manufacturing. 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).

The Torque multiplied by the speed gives an insight of the total power of the motor. The Efficiency map is a behavior map that expresses the motor efficiency function in the torque-speed domain of an EM. One would calculate such kind of behavior maps when there are multiple operating points to consider the motor's drive cycle. The other KPIs such as Losses and Torque ripple are also behavior maps similar to the Efficiency KPI. The paper in [35] also cites that the efficiency vs torque and speed map is most representative of the characteristics of the EM.

FEA simulations are numerical methods that discretize the geometric parameters of an EM into finite elements and solve 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 complicated setup to run simulations typically in Matlab.

In computational electromagnetics, deep learning can be seen as the most eligible surrogate of FEA simulator as they can provide almost accurate results in relatively no time. This is because the FEA simulations may take hours to days as it needs to generate for all operating points in the Efficiency map. Despite the computational burden it comes with, the authors from [28] defends that FEA is most appropriate to generate EM efficiency maps in terms of accuracy based on experimenting overall error rates across operating ranges for different EM losses. FE-based ANN models have been utilized for predicting stress distribution in a 3D printing process [30], bend angles in laser-guided bending [31], and performance of a thermoelectric generator [32], for example.[07]

This process, though well established in the EM design is very resource intensive, time consuming and does not allow for high-throughput engine design optimization.

The actual engine data of Valeo is used here as the dataset comprising of multiple variant designs of the Double-V topology. Figure 1.1 depicts the structure of a 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 Rotor.

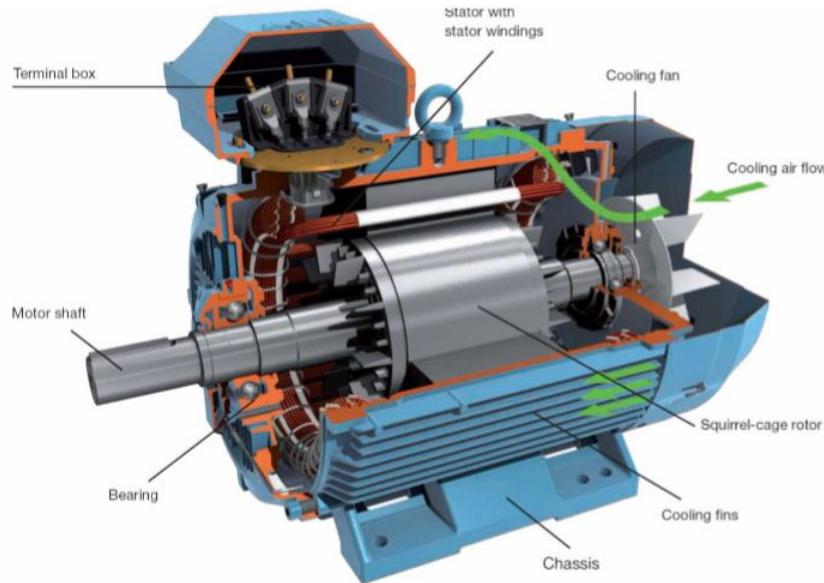


Figure 1.1: EM Motor (Source : Valeo)

The 3 motor topologies manufactured by Valeo are displayed in Figure 1.2:

1. Single V Magnet - Consists of a single V magnet shown in Figure 1.2a.
2. Double V Magnet - Consists of double V magnets shown in Figure 1.2b.
3. Nabla Magnet - Consists of a single V Magnet and a delta magnet shown in Figure 1.2c.

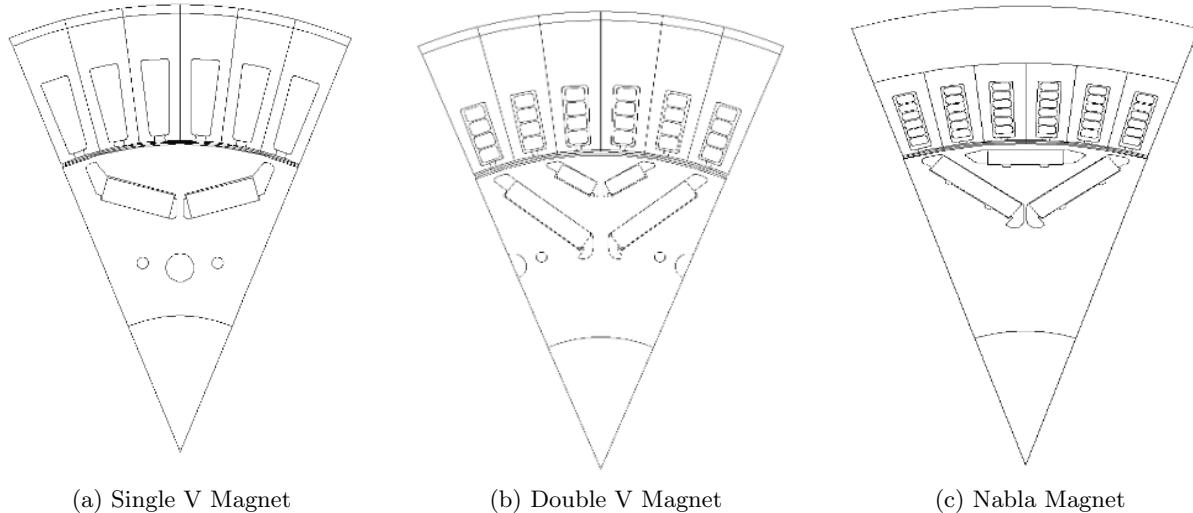


Figure 1.2: EM Magnet Topologies (Source : Valeo)

Figure 1.3 and Figure 1.4 give a glimpse of the EM KPIs to be predicted.

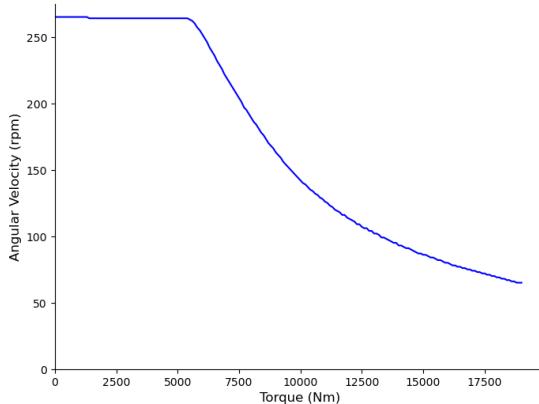


Figure 1.3: Torque Curve

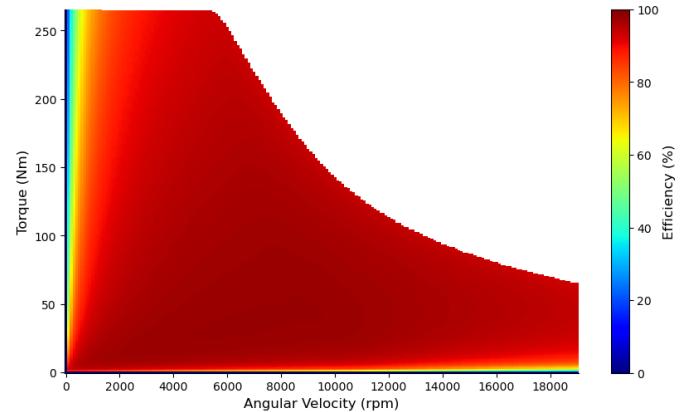


Figure 1.4: Efficiency Grid

We can observe the relation between the Torque curve and the Efficiency grid. The values within the Efficiency envelope can be understood with the different contour shades whose level of shading is shown to the right. Most importantly we can also make out that the Efficiency envelope is of the same shape as the Torque curve.

Figure 1.5 gives an outlook of both the current approach and our proposed approach to generate the KPIs. We discuss each component of the flowchart below :

1. EM designs parameterized

The design of each of the EM variants described parametrically for its geometric, physical and simulation features are considered as the input.

2. Current Approach

Approach followed currently by Valeo.

(a) Matlab Script 1

Creates a design mesh from the parametric description of the EM's geometric and physical features.

(b) FEA simulator

Multiple FEA simulations which are by nature PDE is done on this mesh. The FEA solver then generates the byproducts 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 mat format are fed to the next stage.

(d) Motor Builder

Another round of post processing is done on the magnetic flux products to generate the Power and Torque associated with the EM design. Motor builder settings such as length of the motor, maximum rotation speed as well as electrical settings for e.g. input voltage and current are given to generate the KPIs. The Motor Builder also is a GUI that provides visualization of the KPI plots to the designer.

3. Proposed Approach

Approach we undertake as surrogate modelling.

(a) Data Preprocessing

The input features typically as its numerical equivalent is preprocessed 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

We use MLP as our deep learning model which is made up of fully feedforward connected layers. For training, also the targets are 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 of the **Matlab Script 2** in terms of dimensions and is plotted.

4. KPI Plots

The targets which are vectors of numerical values are displayed graphically.

This master thesis explores a way to do light weight surrogate modelling of the current process as is highlighted in Figure 1.5 by exploiting data-driven deep neural networks to approximate the KPIs derived from FEA simulations.

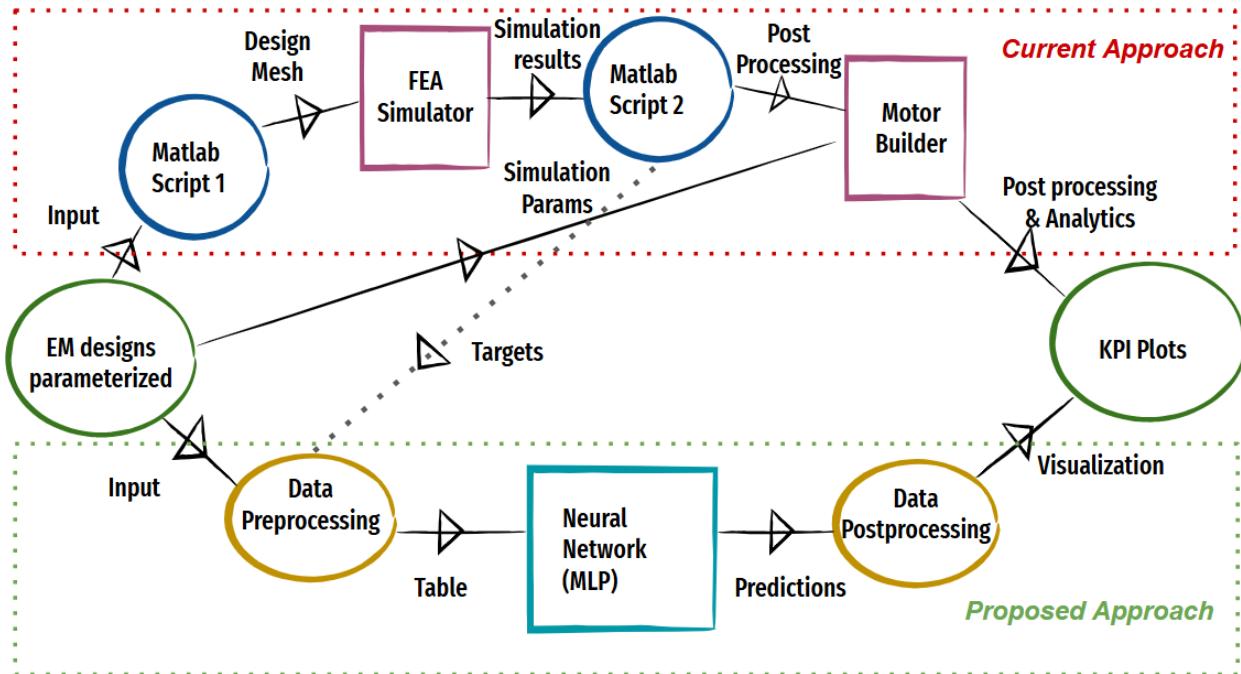


Figure 1.5: EM Design Flowchart

Complete replacement of FEA simulations with deep learning models is not feasible however we 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. The inverse step of generating the motor designs is most useful for companies as they can get a feel of what the best designs would look before manufacturing them.

1.1 Objective

The motivation of our work is to verify whether surrogate modelling to replace the FEA simulations could be done. Our task is a supervised learning task to predict 2 KPIs namely the Torque curve and the Efficiency grid which are vectors of parameters from the parametric description of topology invariant EM and FEA simulated KPIs as ground truth. The remaining KPI such as Losses can be calculated from these 2 KPIs as they are inversely proportional to the Efficiency values. The Torque curve is a 2D plot ie, a vector across speed ranges and the Efficiency grid is a 3D plot ie, a matrix of dimensions torque ranges times speed ranges. Both the outputs are continuous values which makes it a regression problem.

1.2 Motivation

1.3 Problem Statement

1. We have a regression problem but as we are predicting two values it makes it a multi regression problem. The model architectures in Section 4.1 shines light on how we go about it.
2. Another concern we came across is the Torque Curve typically harbours integers values albeit our task is a regression problem and we discuss how we tackle it in Section 4.1.
3. The Efficiency grid dimensions vary across EM variants, and we need the target from the model to be of a fixed size. This problem is mitigated in Section 3.1.3
4. The ranges among the 1st and 2nd target vary significantly and is yet another challenge we overcome in Section 4.2.2

1.4 Research Question

We were skeptical on building a model architecture wherein the 2nd KPI is dependent on the 1st KPI as typically the former's shape is dependent on the latter and not its values. However we can assume the values within the shape to be non zero values and otherwise 0s and model it in future.

Alternatively we could also build 2 models one for each KPIs 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 we might need to generate EM parameteric descriptions. Additionally we deemed it unnecessary as the dimensions of the Efficiency grid vary with the torque curve and not necessarily the Efficiency values.

1.5 Thesis Structure

Over the course of the thesis we 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, Graph Modelling, Modelling, Experiments and Results, Conclusion, and Bibliography. In Literature Review section will introduce the works that has already been carried out in this domain. In the Dataset section a detailed insight to how our data is structured is elaborated. In the Modelling section, we introduce the network architectures and loss regularization techniques used to tackle the problem. The outcomes of our work are presented in Experiments and Results chapter in addition to other findings we unearth. Graph Modelling section is an empirical study which outlines the background of GNNs primarily Heterogeneous GNNs and defines its concepts. we also attempt to present a workflow on how to use GNNs for our task. Conclusion chapter summarizes the thesis briefly and would also give a glimpse into possible areas of improvement. Lastly the Bibliography section lists out the articles cited for this thesis.

Chapter 2

Literature Review

There has been extensive research in modeling the Electric Motor with Convolution Neural Network (CNN) based on the images of the motor cross-section.

However our approach is progressive in the sense that once the KPIs are predicted we would like to be able to generate the inputs.

Reproducing images is not known to be the best approach given the infamous known fact that AI generated images are faulty. However by generating the parameters of the motor we can be rest assured of more precise results. Hence the need to focus on the inputs as they are with their parametric description. Vivek et al. in [37] also 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 sample the latent space to generate new EM designs.

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. We do surrogate modelling of the FEA simulations which serves as the baseline to eliminate the role of evolutionary algorithms as well for generating designs. The latter could potentially be a future work to be carried out 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 [37] undertake works on optimizing electric motor design generation by compressing its parametric description across multiple topologies into a compressed latent space thereafter from which new EM designs can be sampled. They also disclose 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 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.

The authors of [08] presents a method to predict 2Dimension (D) flux maps of an Interior PMSM 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 our usecase for modelling EM for cars, the writers of [33] experiments to compute the efficiency map of Toyota Prius have been computed. The methodology used is to observe Magnetic field flux density to predict how it evolves over time for different operating points of the map. The paper [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 speed, 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.

In [13], the authors compare and evaluate the performance of surrogate modelling Surface Permanent Magnet's parametric and image based 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 which is not. To learn the non linear functions, they also experimented with CNNs which could infer all targets just as well as it inherently takes into account the pixel spatiality from motor designs as RGB images. In addition they built a hybrid model to utilize the image and parametric description but have comparable reported 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 constrained the precision of the inputs.

Existing literature also covers works on modelling this work as tabular data using MLPs.

We review the concept of surrogate modelling and its applications in EM domain. The researchers of [07] talks about the use of domain knowledge to improve 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 [38] and [39] also claim that a physics assisted neural network significantly improved the accuracy performance when predicting the cogging torque of Permanent Magnet Synchronous Motors. Their experiments suggest that 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 suffice. Works by [16] explores methodologies to extend an already model to predict efficiency maps for a topology it was untrained for by exploiting the concept of transfer learning. The writers claim to get this accomplished by 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.

Furthermore, research by [15] also presents 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, this is evident from the Figures in 1.2. 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 the inputs 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, the FEA is used yet 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 also be beneficial for both parameter and topology optimization of EM as well as for prediction of its other KPIs.

Arbaaz et al. in [29] also tries 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, as the number of excitation points vary across designs, thus a model suitable for handling variable input sequence length is needed. Meaning the values only within the torque curve i.e., excitation points are predicted. This 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 propose 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 to generate the KPIs with the FEA simulations or the surrogate model.

Studies in [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 is not projected as much having less dominance with respect to coverage area. They experimented this finding with Induction Motors to generate its Torque Vs Speed 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 [32] also discusses approaches to judge the accuracy with a physics aware metric for estimating magnetic field of low frequency electromagnetic devices. They do so by quantifying the uncertainty of the predictions by adding a probabilistic component to the weights of the neurons in the network architecture. This technique they employ is called Monte Carlo dropout with which they generate uncertainty maps. The distribution of the predictions they thus 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 Networks 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 simulator as is shown in the Figure 1.5.

The writers of [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.

Although this is fairly good foreseeing the impact of generating the inverse process yet MLPs cannot necessarily learn all the intricacies within motor components. Hence the need to better represent the data typically in the form of graphs and model GNNs to achieve the desired results. Modelling with graphs preserves the spatiality of the design as well as also gives us provision to incorporate the physical and magnetic characteristics of the designs if any which would not have been possible with CNNs. There has been close to no work of GNNs in this domain. However we see progress of GNNs in molecular chemistry and social networks usecases from which we draw inspiration.

Chapter 3

Dataset

Valeo has shared 1481 Excel Workbook files for each motor variant of three-phase interior PMSM motors with 8 poles and 48 slots. 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 we made based on our understanding of the data.

Figure 3.1 shows the geometry of a whole Double V motor which can be sliced into 8 identical parts owing to the motor design's rotational symmetry.

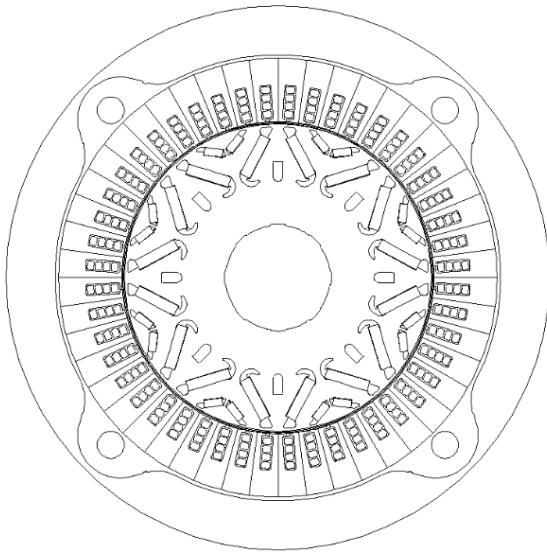


Figure 3.1: Complete EM Geometry(Source:Valeo)

Figure 3.2 is a hand drawn sketch showing our understanding of how the geometry of 1/8 cross-section of the same motor looks like. This comes in handy when creating the graph representation of the motor. From the Figure, we can make out that the motor very largely is comprised of the Rotor and the Stator separated by the air gap between which the magnetic flux flows. The Rotor hosts the permanent magnets which when rotated generates the magnetic field. The 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 along fixed points. 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 sit at. The Stator yoke separates each tooth shoe from the outer radius of the motor.

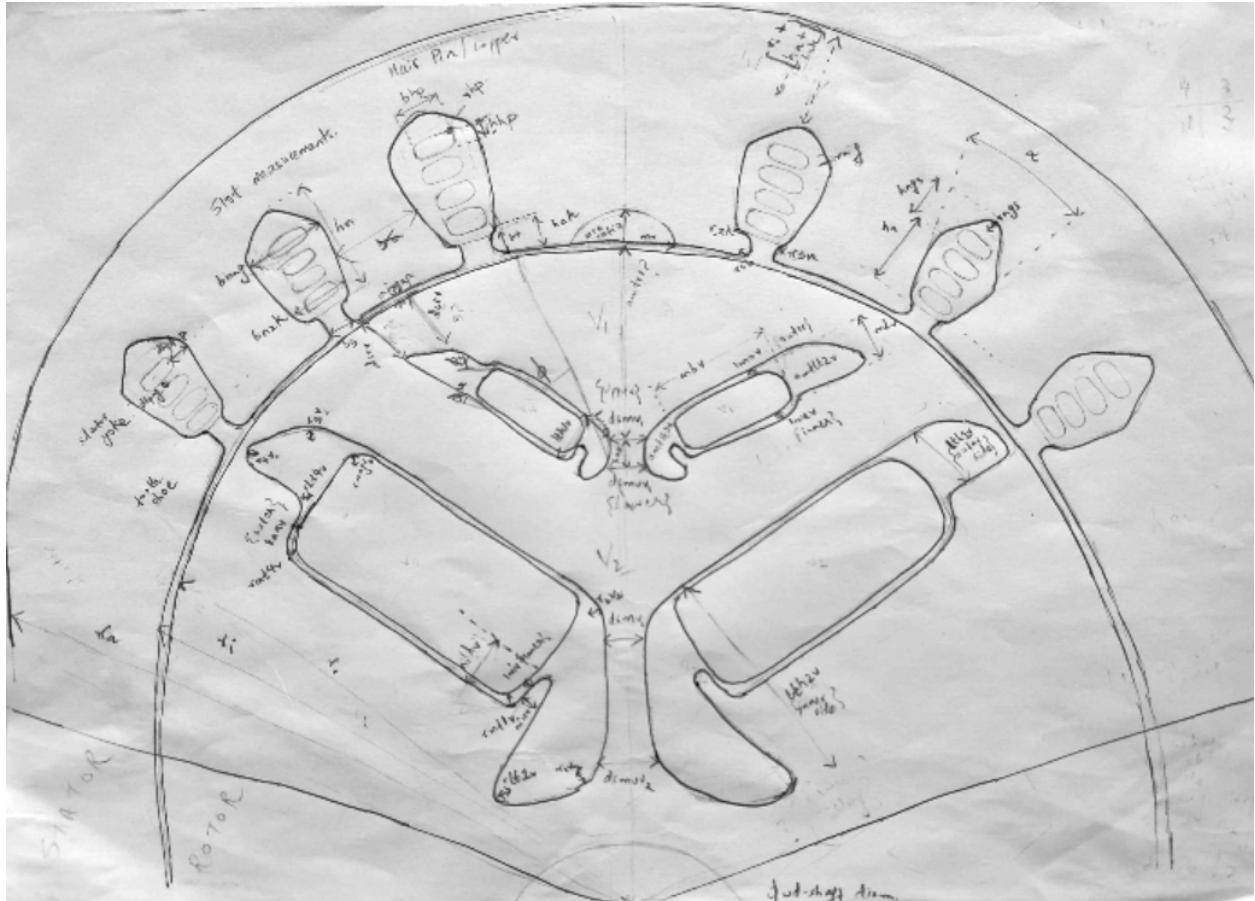


Figure 3.2: 1/8 Motor Crossection

3.1 Data Preprocessing for MLP

For modelling the MLP, we present the data in tabular form with the parameters corresponding to columns. In order to make the data compatible with our model, some level of data processing was carried out as elaborated below.

3.1.1 Data Exploration of the Input Parameters

All parameters including the additional ones in each topology are considered as a separate columns and therefore if a particular column is topology dependent then the column corresponding to the missing data of this topology is treated as 0 values. Defaulting geometrical values as 0 seems to be the most reasonable approach in comparison to its standardised norm. 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 the latter are directly related to the geometry and makes derivation of other parameters much simpler whereas degrees is a notion of denoting a sliced angle.

CITE...Explain better

3.1.2 Data Exploration of the Torque KPI(Torque Curve)

Figure ?? shows the standard deviation of few samples of the Torque KPI from the entire dataset. The x-axis represents the angular velocity of the motor ranging from 0 to 19000 rpm. Meanwhile, the y-axis represents the torque values corresponding to the angular velocity. For our dataset, the range of torque values are

between 55 and 280 Nm. We do not start the y-axis from 0Nm as the torque curve stops its decline on reaching torque saturation point of 55 Nm and we presumed it best to conserve spacing in the plot. The mean of all samples of the dataset is displayed with an overlay of how its standard deviation is around the mean. Furthermore, we display a twin y-axis in red showing the standard deviation of random handpicked samples with the mean.

We make the following observations:

1. The Root Mean Squared Error (RMSE) 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 visible with the plateauing of the curve.
2. The curve to an extent resembles a mirrored S shape. This finding is critical for how we modelled the loss regularization for the target and will be further elaborated in Section 4.2.1

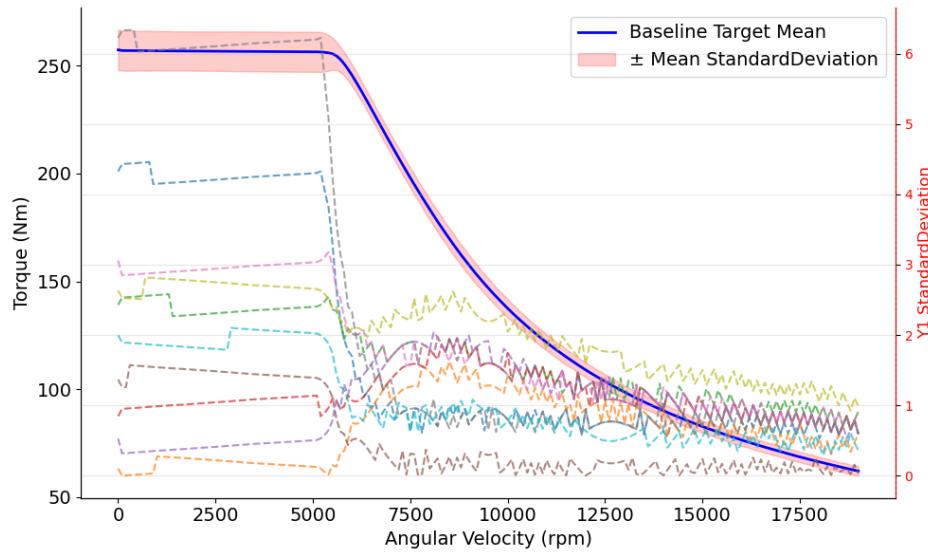


Figure 3.3: Standard Deviation of 2D KPI

3.1.3 Data Exploration of the Efficiency KPI(Efficiency Grid)

As the target values of the Efficiency KPI are not provided with the correct dimensions we have an additional step which takes the maximum torque value from the Torque KPI and slices off the Efficiency grid to only range from [-Maximum Torque, Maximum Torque]. Subsequently we choose only the rows of the MM grid detailed in Table A.1 which correspond to the indices of the sliced efficiency grid. This step ensures that we grant the model the correct dimensions of the Efficiency KPI based on its Torque KPI.

Figures 3.4 and 3.5 both illustrate the standard deviation of the efficiency values plotted against its angular velocity. The standard deviation levels is shown as a scale with colors darkening as the standard deviation increases. Efficiency values for negative torque values correspond to when motor is in generating mode and those of positive torque values to when motor is in monitoring mode.

In both modes, the efficiency is almost similar albeit from Figure 3.4, we note it is not the case for our dataset. This is evident from low angular velocity-high torque distribution area where we can see a clear distinction with Not a Number (NaN) values. Since these are FEA simulations, it is probably an effect of a post processing step taken by the Motor builder. 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, we do not do so, as it is not relevant to evaluate a duplicate again. From Figure 3.5, we can observe the deviation is at its peak at low torques, extreme angular velocities. We discuss the modelling of this information in Section 4.2.2. We also can observe that beyond the Efficiency envelope,

there are no efficiency values and can be translated as blank values in the grid or NaN values in the padded Efficiency matrix. we incorporate this information in Section 4.2.2. In addition, we notice skewness at the border of the curve within the grid, we discuss in Section 4.5 how we handle this scenario.

As we had noted already 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 we use in Post Processing as is further elaborated in Section 4.5

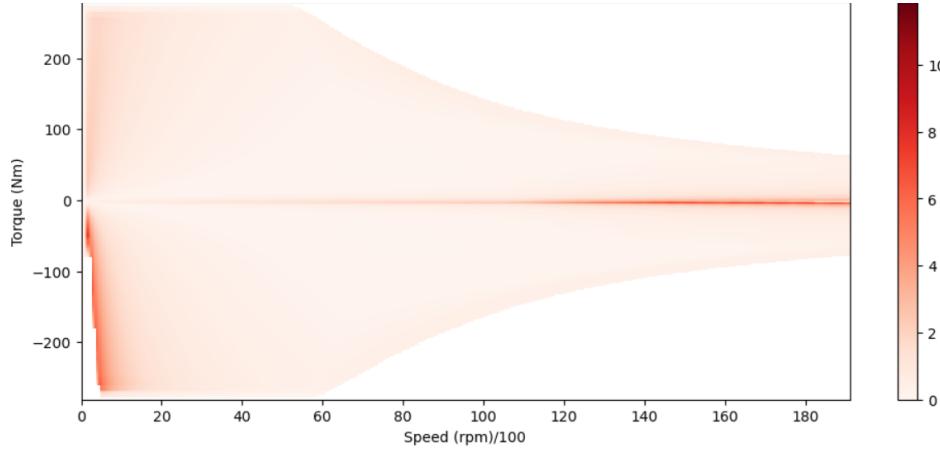


Figure 3.4: Standard Deviation of Efficiency KPI

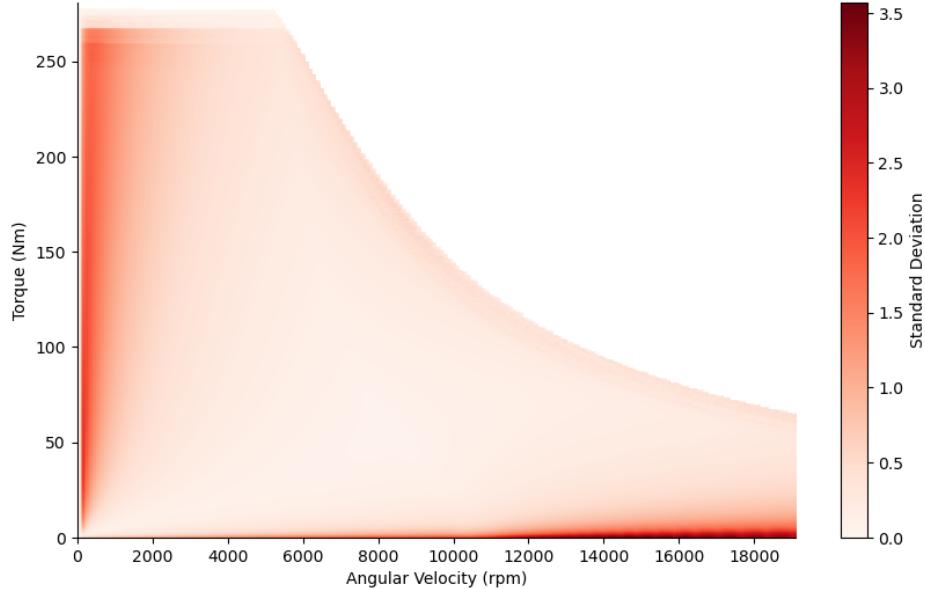


Figure 3.5: Standard Deviation of Efficiency KPI Positive Grid

Figure 3.6 gives us the holistic view of how the efficiency values are distributed across equally spaced intervals of angular velocity. The Figure plots the standard deviation of all the Efficiency values of all samples of our dataset as an error bar for specific angular velocities at equal intervals of 2000 rpm. The distributions shows maximum skewness towards extreme angular velocities and is relatively stable between 4000 and 6000 rpm. We discuss in Section 4.2.2 how we integrate this finding into teaching over model.

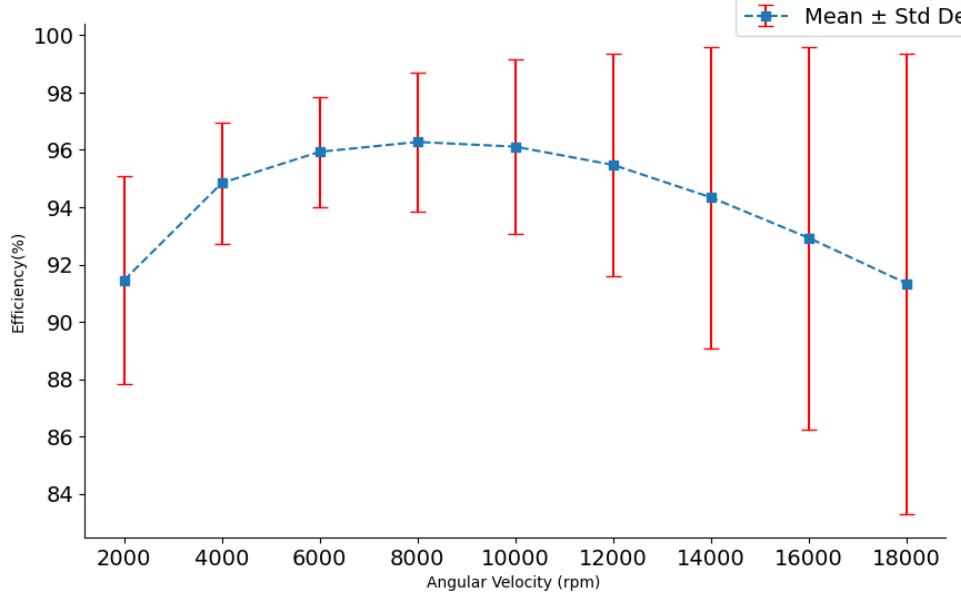


Figure 3.6: Standard Deviation of Efficiency KPI Positive Grid across Speed Intervals

Reading each sheet from the excel files particularly the grids take up a lot of time and compute, hence we 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 our case the value is 280 computed internally from the dataset but this is subject to change as we receive more data and there is a provision handy to override it on demand. The array is then saved locally for easy access and loading during training.

Initially we tried to set NaN values as an incredibly high value hoping the model would consider this as a default value 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, we have come up with a better way of handling this scenario and we elaborate in Section 4.2.2 how we tackle it.

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. We 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

$$z = \frac{x - \mu}{\sigma} \quad (3.1)$$

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 we have columns with different ranges for the input since we consider each feature a column mentioned in Section 3.1.1.

We 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 we scale the target we 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 we will have lost a lot of originality

in each example and is now introducing noise to new examples especially when they have a different data distribution.

3. For the Efficiency KPI we have loss regularization Equation 4.6 that has a constraint check on the maximum range of efficiency values that can be predicted. If we 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 scalar for comparison.

During our experimentations where we initially scaled the targets, we 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 losing precision, we continued with the original targets.

3.3 Dataset splitting

We convert the data to floating point tensors for better precision and collate them into a Tensor Dataset. We 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 we have a separate test dataset from the validation is to ensure that there is no data leakage as we do not want to overfit the test dataset with the hyperparameters we 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 we expect to cover most grounds on training and have good monitoring on 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 generalisation performance. We have also used Data loaders to split the dataset into batches that fits into our Graphics Processing Unit (GPU) memory.

Chapter 4

Modelling & Evaluation

For our multi regression problem, we train a MLP on the tabular representation of the data.

4.1 MLP Model

For the MLP model, we use a single 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 we have experimented with fewer feed forward layers in the former than in the latter.

We 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 thus 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. We have 2 hyperparameters to control the dropout rate at which we freeze the neurons when training also to be discussed in Table 5.1. We 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. Figure 4.1 gives an outline on how the MLP Model architecture is designed. We discuss each component of the architecture below :

1. Input

The input layer takes in all features of the tabular data which is 89 in our 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 we tune. We 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 we have Batch Normalization layers between each linear and ReLU activation function in addition to drop out layers. The dropout rate for the layers in this block is relatively higher as we want to encourage the model 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. We use a ReLU activation function at the end of the output as the target values are inherently always positive values and 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 we also increase the neurons of the hidden layers as we are not limited by the dimensionality of the output feature. This would enable the model to be more strong and grasp the complex patterns in the data better. As usual we have 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 we want to encourage the model to learn the specific nature of the grid towards the end. For the Last Linear layer we have the output features corresponding to the target dimensions and a ReLU activation again 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 19000 rpm at equal spaced intervals of 2000rpm. Although the targets for the Torque KPI are an array of integer values, we use the float tensor and not integer tensor 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 our case is the shape of the padded collated array we had created in Section 3.1.3

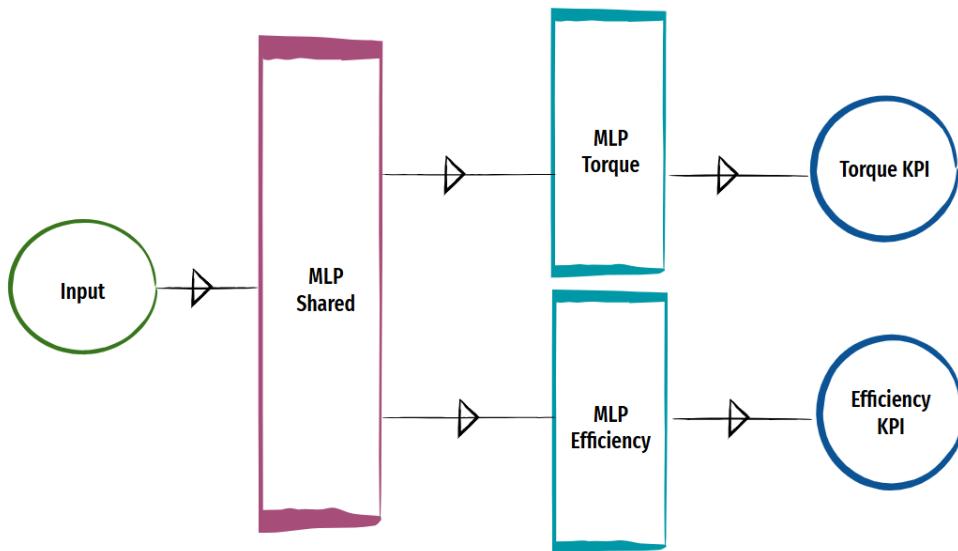


Figure 4.1: MLP Model Architecture

4.2 Loss Functions

The MSE loss is the error metric used for our problem with the intention that the squared losses penalize 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(Torque curve)

The MSE loss for the Torque KPI is formulated mathematically as in Equation 4.1

$$\text{Y1 Loss} = \frac{1}{n} \sum_{i=1}^n \frac{1}{h} \sum_{j=1}^h (y_{ij} - \hat{y}_{ij})^2 \quad (4.1)$$

where, n is 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, we have experimented with 2 Loss Regularization techniques. Both of which are L2 Regularization to be in sync with the dynamics of the MSE loss.

1. Smoothening Loss Regularization

To smoothen out the curve for the Torque KPI we apply the below loss regularisation factor to ensure that the torque values at neighboring operating points are as similar as possible. This is formulated mathematically as in Equation 4.2.

$$\text{Y1 Smoothening Loss Regularization} = \frac{1}{n} \sum_{i=1}^n \frac{1}{h-1} \sum_{j=1}^{h-1} (\text{ReLU}(|\hat{y}_{ij+1} - \hat{y}_{ij}| - 1))^2 \quad (4.2)$$

This nature is factored in by penalizing the loss by the magnitude if the neighboring values in the prediction are not close to each other. ReLU helps to clips the difference if it is negative which is the scenario when a violation is not warranted.

2. Decreasing Loss Regularization

The Torque curve closely resembles a decreasing sigmoidal curve and hence we 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. This is formulated mathematically as in Equation 4.3.

$$\text{Y1 Declining Loss Regularization} = \frac{1}{n} \sum_{i=1}^n \frac{1}{h-1} \sum_{j=1}^{h-1} (\text{ReLU}(\hat{y}_{ij+1} - \hat{y}_{ij}))^2 \quad (4.3)$$

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.

We have not combined both 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, we perform ablation studies with both the regularizations and report the results obtained in Table 5.3

4.2.2 Loss for Efficiency KPI(Efficiency Grid)

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

$$\text{Y2 Loss} = \frac{1}{n} \sum_{i=1}^n \frac{1}{w} \frac{1}{h} \sum_{j=1}^w \sum_{k=1}^h (M_{ijk} \cdot y_{ijk}) - (M_{ijk} \cdot \hat{y}_{ijk}) \|^2 \quad (4.4)$$

$$M_{ijk} = \begin{cases} 1 & \text{if } y_{ijk} \neq \text{NaN} \\ 0 & \text{if } y_{ijk} = \text{NaN} \end{cases} \quad (4.5)$$

where, M_{ijk} is Mask matrix, w is the rows of 2D vector and h the columns of 2D vector. Based on one of our observation in Section 3.1.3 on the efficiency values beyond the Efficiency envelope having blank values, which we had also padded them to be NaN values we construct a binary mask matrix to ignore them in the loss calculation. As ANN 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 is 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.

We 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, we modify the objective functions mathematically as in Equation 4.6.

$$\text{Y2 Maximum Efficiency Regularization} = \frac{1}{n} \sum_{i=1}^n \frac{1}{w} \sum_{j=1}^w \sum_{k=1}^h (\text{ReLU}(|\hat{y}_{ijk}| - 100))^2 \quad (4.6)$$

When the efficiency values of the prediction exceed 100, we penalize the overall loss by squared magnitude of the difference of the prediction from its ground truth. ReLU 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%. We refrain from instructing the model to not have values less than 0 since we mask NaN values as 0 and the model will for sure attempt to predict values close to 0. Moreover, these predictions are not relevant for us as after generating all predictions we 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, we do not see the need to needlessly punish the model for making mistakes for values we eventually donot use as 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 our observations gathered in Section 3.1.3 and from the regions in Figure 3.5 which showed the most standard deviation, we have tried to incorporate all of the below learnings via the loss function as Y2 Regularization techniques.

(a) Efficiency at Maximum Torque Loss Regularization

We cannot incorporate regularizing the loss to have the decreasing envelope of the Torque curve and so we only focus on what we 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 we make on the shape of the Torque curve in Section 3.1.2. To ensure that the shape of the Efficiency KPI is maintained, we also regularize the loss for the maximum torque value. To do so, we have attempted to retrieve the last rows of our Efficiency KPI and those of its target values and penalize the squared difference to have higher weight. We formulate it mathematically as in Equation 4.7

$$\text{Y2 Loss Regularization Max Torque} = \frac{1}{n} \sum_{i=1}^n \frac{1}{t_1} \sum_{j=-t_1}^w \frac{1}{h} \sum_{k=1}^h (y_{ijk} - \hat{y}_{ijk})^2 \quad (4.7)$$

where, t_1 is the threshold for initial Efficiency KPI Envelope boundary. The number of last rows is determined by a threshold t_1 . The intention of including the threshold value is to give the model the flexibility to search a certain portion of the Efficiency grid determined by the threshold and push the model to get this region right by having 0s for values in prediction but not in target.

(b) Efficiency at Low Angular Velocity 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. To force the model to pay more attention at lower angular velocities, we have regularized the loss for the first few columns

of each row of the predicted Efficiency grid and penalize the squared difference with that of its target. We formulate it mathematically as in Equation 4.8:

$$\text{Y2 Loss Regularization Low Angular Velocity} = \frac{1}{n} \sum_{i=1}^n \frac{1}{w} \sum_{j=1}^w \frac{1}{t_2} \sum_{k=1}^{t_2} (y_{ijk} - \hat{y}_{ijk})^2 \quad (4.8)$$

where, t_2 is the Threshold for Low Angular Velocities. The number of first columns is determined t_2 . The intention of including the threshold value here 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 the region of low angular velocities right by penalizing this region more.

(c) Efficiency at Low Torque Loss Regularization

At extreme speeds we find a greater deviation in the efficiency values particularly towards higher speeds. This is because we have fewer efficiency values as speed increases beyond a range since not all torque values participate. To force the model to be more careful at low torque, we have regularized the loss for the first few rows of each column of the Efficiency KPI and penalize the squared difference with that of the target. We formulate it mathematically as in Equation 4.9:

$$\text{Y2 Loss Regularization Low Torque} = \frac{1}{n} \sum_{i=1}^n \frac{1}{t_3} \sum_{j=1}^{t_3} \frac{1}{h} \sum_{k=1}^h (y_{ijk} - \hat{y}_{ijk})^2 \quad (4.9)$$

where, t_3 is the Threshold for Low Torque. The number of first rows is determined by t_3 . The intention of including the threshold value herein 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 the regions of low Torques right by penalizing this region more.

The above Y2 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 we can optimize if we observe the relevant parts of the Efficiency KPI's visualization not adhering to the expected look of the Efficiency grid. We aggregate all the above regularizations to form the Y2 Loss Regularization as in Equation 4.10

$$\begin{aligned} \text{Y2 Efficiency Grid Loss Regularization} &= \text{Y2 Loss Regularization Max Torque} \\ &+ \text{Y2 Loss Regularization Low Angular Velocity} + \text{Y2 Loss Regularization Low Torque} \end{aligned} \quad (4.10)$$

The Total Loss is calculated in Equation 4.11

$$\begin{aligned} \text{Total Loss} &= \text{wt} \times (\text{Y1 Loss} + (\lambda_{1y1} \times \text{Y1 Smoothening Loss Regularization}) + (\lambda_{2y1} \times \\ &\quad \text{Y1 Declining Loss Regularization})) + (1-\text{wt}) \times ((\text{Y2 Loss} + (\lambda_{y21} \times \\ &\quad \text{Y2 Maximum Efficiency Loss Regularization}) + (\lambda_{y22} \times \\ &\quad \text{Y2 Efficiency Grid Loss Regularization}))) \end{aligned} \quad (4.11)$$

where, λ_{1y1} is Y1 Smoothening Loss Regularization Parameter, λ_{2y1} is Y1 Declining Loss Regularization Parameter, λ_{y21} is Y2 Maximum Efficiency Loss Regularization Parameter, λ_{y22} is Y2 Efficiency Grid Loss Regularization Parameter, wt is Y1 Loss Weightage, $1 - \text{wt}$ is Y2 Loss Weightage.

We have added a Weightage parameter that controls the contribution of the Y1 Loss and Y2 Loss to the Total Loss. There are 2 reasons why this is useful for us :

- 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.

We 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 the factor of how much its value range varies.

2. When the prediction accuracy of one KPI is substantially more vital than the other. Our 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 as its value range is relatively higher we make decisions from monitoring the prediction performances.

We reflect on these decisions based on whether the envelope of the Efficiency KPI grid is more valuable than the efficiency values within it.

To counter the Torque KPI loss dominating the total loss, we have implemented quite a few regularization techniques to the Efficiency KPI loss.

Ultimately we decided on prioritizing the efficiency values.

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 [17].

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 we have considered for our regression problem is the average of the RMSE. Therefore, the model with the least prediction scores ie, closest to 0 is ideal for our application.

4.4.1 Evaluation Metrics for Torque KPI

The Y1 Score for the Torque KPI is formulated in Equation 4.12 :

$$Y1 \text{ score} = \frac{1}{n} \sum_{i=1}^n \sqrt{\underbrace{\frac{1}{h} \sum_{j=1}^h (y_{ij} - \hat{y}_{ij})^2}_{Y1 \text{ RMSE}}} \quad (4.12)$$

where, h is the columns of 1D vector and $Y1 \text{ RMSE}$ is RMSE for each test sample.

4.4.2 Evaluation Metrics for Efficiency KPI

The Y2 score for the Efficiency KPI is formulated in Equation 4.13 :

$$Y2 \text{ score} = \frac{1}{n} \sum_{i=1}^n \sqrt{\underbrace{\frac{1}{w} \frac{1}{h} \sum_{j=1}^w \sum_{k=1}^h (y_{ijk} - \hat{y}_{ijk})^2}_{Y2 \text{ RMSE}}} \quad (4.13)$$

where, w is the rows of 2D vector, h is columns of 2D vector and $Y2 \text{ RMSE}$ is the RMSE for each test sample.

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 we first convert it into the tabular representation our model consumes and then apply the scaling. This is why we preserve the same scalers used during training as we not only evaluate our dedicated test dataset but also for clients to use on demand.

Furthermore as we are 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 we tried to slice off the grid to ignore 0 values for all rows excluding the 1st row as per the Equation 4.8. We hoped the Mask constructed from equation 4.5 would force the model to learn 0s for however it tried to approximate close to 0 and was almost never 0. Hence, we decided to slice the shape of the Torque KPI curve from the Efficiency KPI by counting the number of columns a row to have based on consecutive values in the curve. This brings us back to the point that it is imperative the prediction of the Torque KPI is close to perfect as the envelope of the Efficiency KPI inherently is dependent on it. However if in the worst case scenario where the y_1 and 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 we decide not to slice the Efficiency KPI and instead use the predictions as is.

Chapter 5

Experiments and Results

5.1 Experiments with MLP

After optimizing over an exhaustive random search of the hyperparameter space, we present them in Table 5.1 by monitoring the model's performance across 5 fold cross validation. We 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 Parameter (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
Y1 Smoothening Curve Loss Regularizer (λ_{1y1})	0.5	0-0.75
Y1 Decreasing Curve Loss Regularizer (λ_{2y1})	0.5	0-0.75
Y2 Maximum Efficiency Value Regularizer (λ_{y21})	5	0-5
Y2 Efficiency Grid Loss Regularizer (λ_{y22})	3.75	0-5
Y2 Initial Envelope Boundary Threshold (t_1)	5	0-5
Y2 Low Speed Threshold (t_2)	20	0-30
Y2 Low Torque Threshold (t_3)	20	0-30
Weightage of Y1 Loss (wt)	0.05	0-1
Weightage of Y2 Loss (1-wt)	0.95	0-1

Table 5.1: Hyperparameter Tuning

We are using a 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 cycles of training. The *batch size* is limited to the capacity of our GPU memory. We use the maximum batch size to train faster and therefore use 72 after which we hit the memory roof of our GPU. The *hidden size* refers to the number of neurons in the hidden layers of the MLP model. We have experimented with just 2 hidden sizes as we are 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. We are not so much concerned of the data being dropped in larger extent than anticipated for the Torque KPI because it is relatively simpler to learn and the value range is comparatively larger leading its

loss to dominate 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 detailed 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.

We 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 we supplement the loss regularization with more knowledge. Our main MLP model only uses the loss regularization parameters λ_{2y1} and λ_{y2} ie, it does not consider the Smoothing curve regularization for the Torque KPI whose performance we will separately discuss in Section 5.6.

We chose a 3rd party application Wandb¹ to log metrics from the training run and to monitor model performance across the 5 folds. All 5 folds are monitored and shown in the Figures as training and validation occurs 5 times on different combinations of cross validation splits independent per fold.

Figure 5.1 illustrates the loss at work as training progresses across epochs. Figures 5.1 b and 5.1 c provides 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. Figure 5.1 a depicts the weighted sum of losses discussed in Equation 4.11.

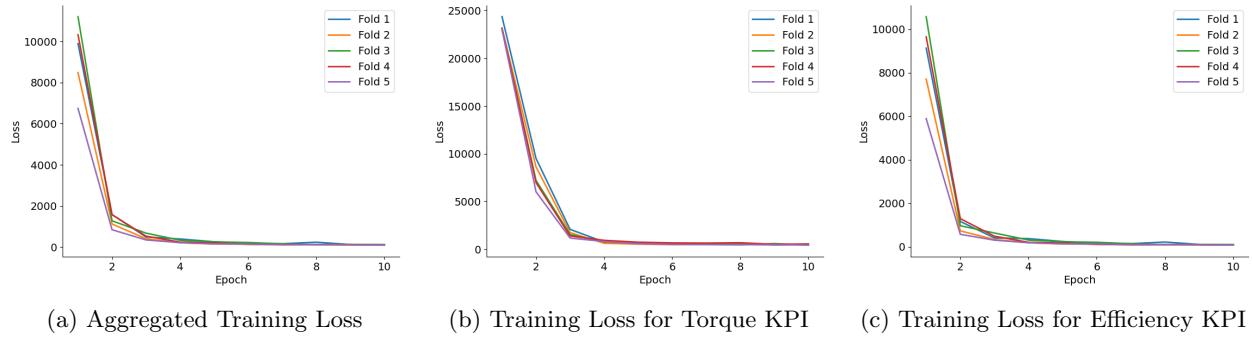


Figure 5.1: Training Loss Metrics

From the training plots we see that the model has converged after having run for 10 epochs with a total run time of 3 MINUTES VERIFY: Even though the weightage 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 we hypothesize to further increase the weightage of the Efficiency KPI. The Torque KPI shows promise in learning better but it would be at the cost of overfitting the Efficiency KPI.

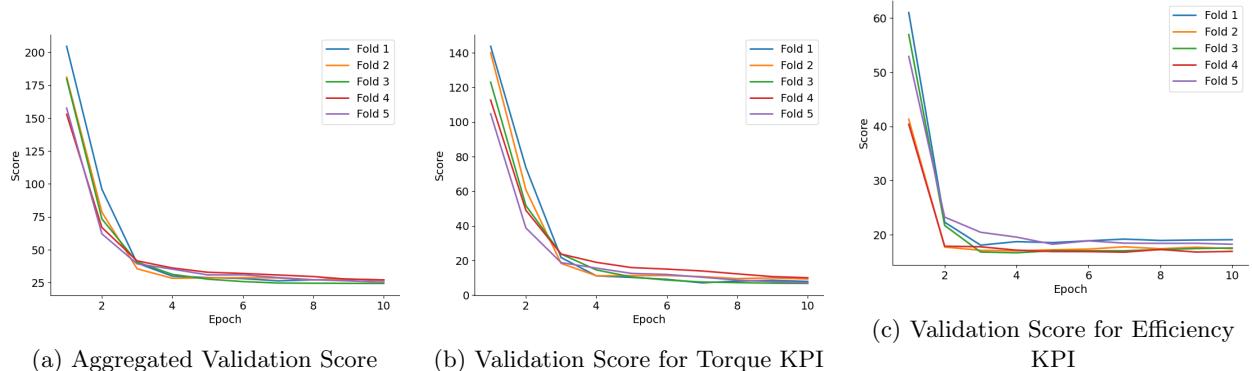


Figure 5.2: Validation Evaluation Metrics

¹Weights & Biases

The Validations plots for losses also tells us the same tale although the Efficiency KPI stops to learn after a certain point can be monitored in A.2. Figure 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. Figure a shows the aggregated validation score which is again the weighted sum of the scores as was formulated in Equation CITE for both the KPIs displayed in Figure b and Figure c respectively. Likewise the training plots evaluation metrics are displayed in Figure A.1.

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.2 for further reflection.

Our observations from the validation plots are : LIST THEM

We have also enabled saving the best performant fold locally so it can be loaded on demand by the client when in need to only run inference.

As the value ranges for both the KPIs are starkly different, we give a glimpse of the percentage difference. We have narrowed down scoring to follow the criteria as recorded in Table 5.2.

% Difference	0-5%	5-10%	10-15%	15-20%	20-25%	25-30%	30-35%	35-40%	40-100%
Y1 Score	0-11	11-22	22-33	33-44	44-55	55-66	66-77	77-88	>88
Y2 Score	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	>40

Table 5.2: Scoring Criteria

This is deduced from the Equations 5.1 and 5.2.

$$Y1 \text{ Percentage Difference} = (Y1 \text{ Score}/(Y1Max - Y1Min)) \times 100 \quad (5.1)$$

$$Y2 \text{ Percentage Difference} = (Y2 \text{ Score}/(Y2Max - Y2Min)) \times 100 \quad (5.2)$$

From our observations the target values for Torque KPI range between 50-280 and those of the Efficiency KPI range between 0-100. The scoring of the Y2 at inference is slightly different from that of training in the sense that for inference, we have the Torque KPI to accurately slice of the envelope from the Efficiency KPI. Also need to ensure how the difference works for NAN especially at inference coz for Training there are no NANs and in targets it would be 0. But for training, we cant slice the envelope as we dont have the Torque curve yet.-TO BE REMMOVED

5.2 Results with MLP

5.2.1 Torque KPI Results with MLP

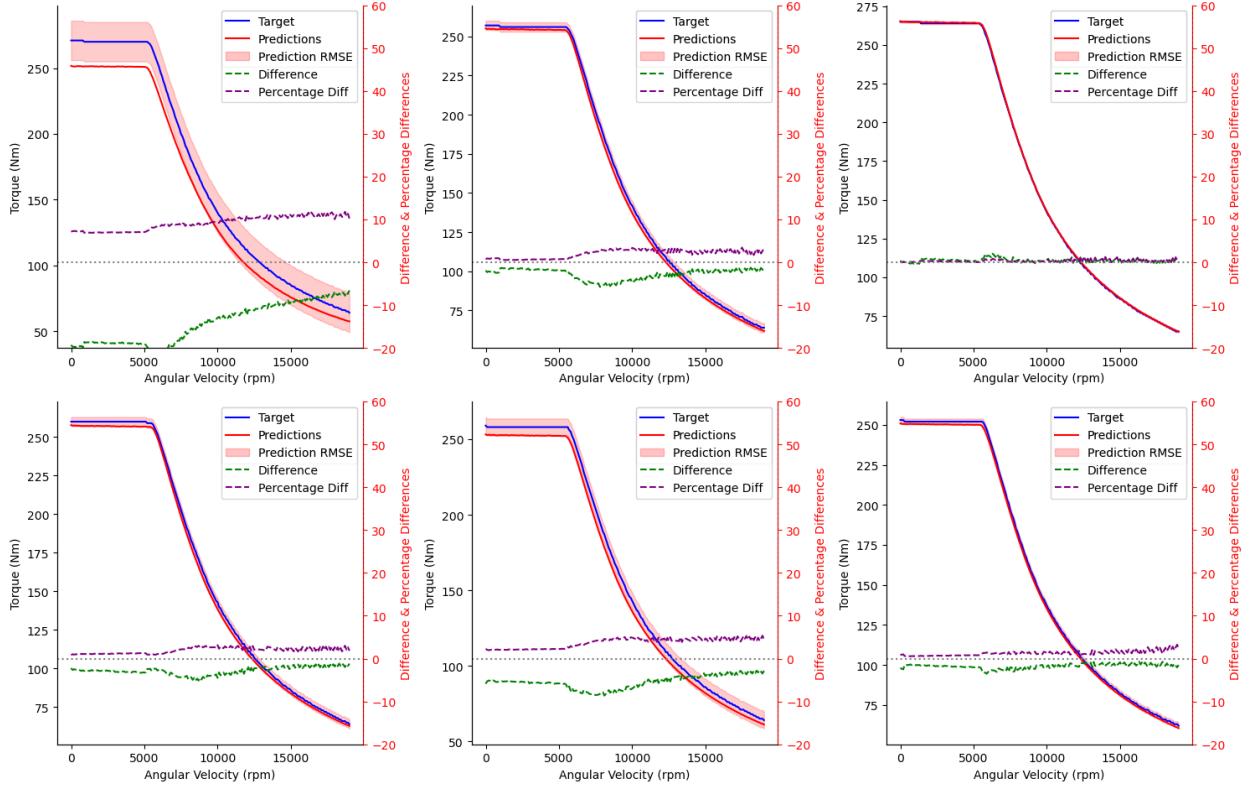


Figure 5.3: MLP Training Results for Torque KPI

Figure 5.3 illustrates 6 test dataset predicted Torque Values and their corresponding ground truths displayed against their angular velocities. The torque values are only displayed for the range of torque participating in the Torque curves and so the continuity to 0 Nm is not maintained here to ensure effective space utilization. We also depict the difference and percentage difference on a twin y axis to give a rough overview of the prediction deviations from the targets. As the percentage difference formulated in Equation 5.1 would always result in positive values, they appear above the 0 line dotted separator. Although we do see deviations, we notice for most of the cases, the predictions are such that the torque values predicted have smaller values than its respective ground truths. In addition, the RMSE equivalent to the Y1 score tells us that for 50% of the plots shown, the predictions are off by about 5% from the target values as per Table 5.2.

Figure 5.4 shows us the Average RMSE and element wise RMSE for the test dataset performance with the MLP. This figure closely resembles the skeleton of the Figure 3.3 in terms of what is displayed. The only difference we can highlight is that the RMSE with that of the ground truth is plotted instead of standard deviation on the twin y-axis in red. Overall our inferences are the predictions 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. In addition, granting a higher weight wt can also help in this direction.

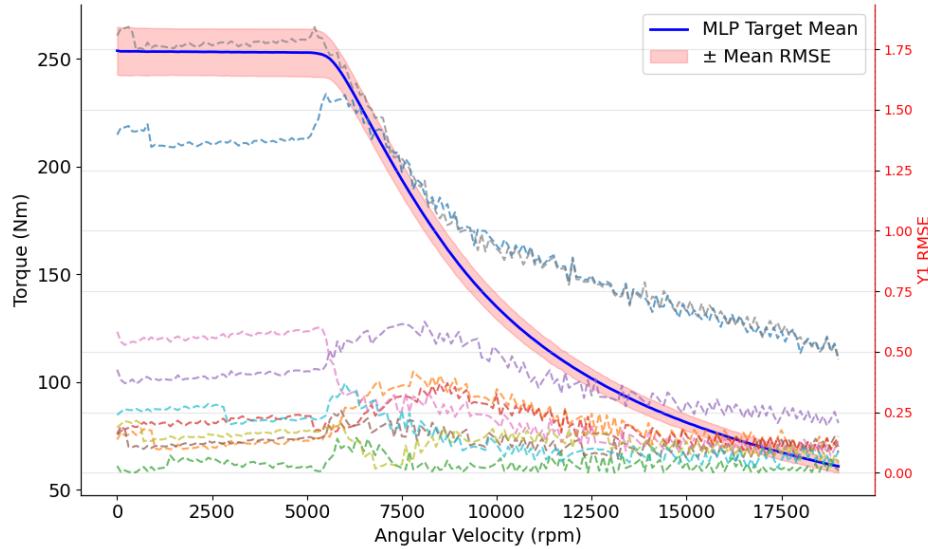


Figure 5.4: MLP RMSE Evaluation for Torque KPI

Figure 5.5 shows the score statistics of the model performance of Torque KPI over few samples from the test dataset. The Y1 RMSE from Equation 4.12 is calculated for each sample and shown as a histogram in Figure 5.5a. The Y1 Percentage Difference from Equation 5.1 is calculated for each sample and shown as a histogram in Figure 5.5b.

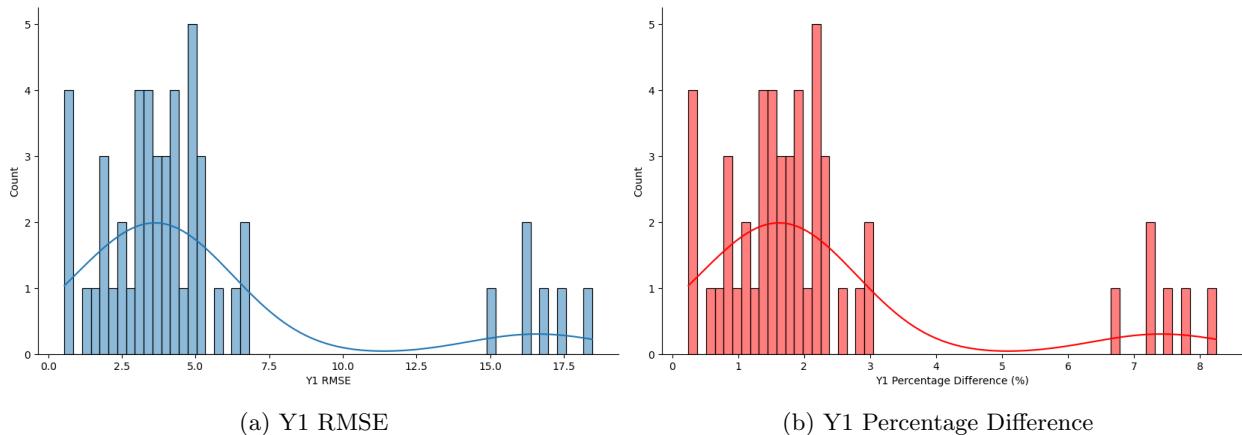


Figure 5.5: Y1 Evaluation Statistics of MLP

It also tells us that the RMSEs are almost evenly distributed between 0.5-12.5 thus encompassing a range of 0-6% difference with the target values.

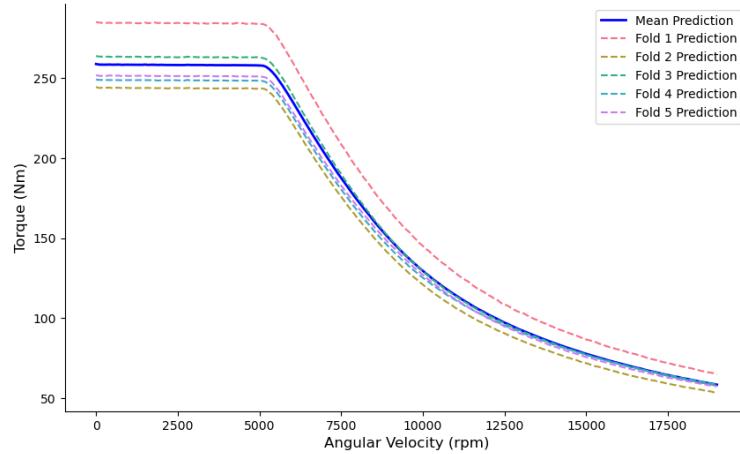


Figure 5.6: MLP Training Deviation across Folds for Torque KPI

Figure 5.6 illustrates how each fold's prediction deviates and from our observation it is as close as can be to the mean of all the Folds predictions. This figure is a good indicator of the model's generalization performance and we have used only 1 engine for inference from the test dataset.

5.2.2 Efficiency KPI Results with MLP

The results of the MLP model from inference with its corresponding overlap is shown below:

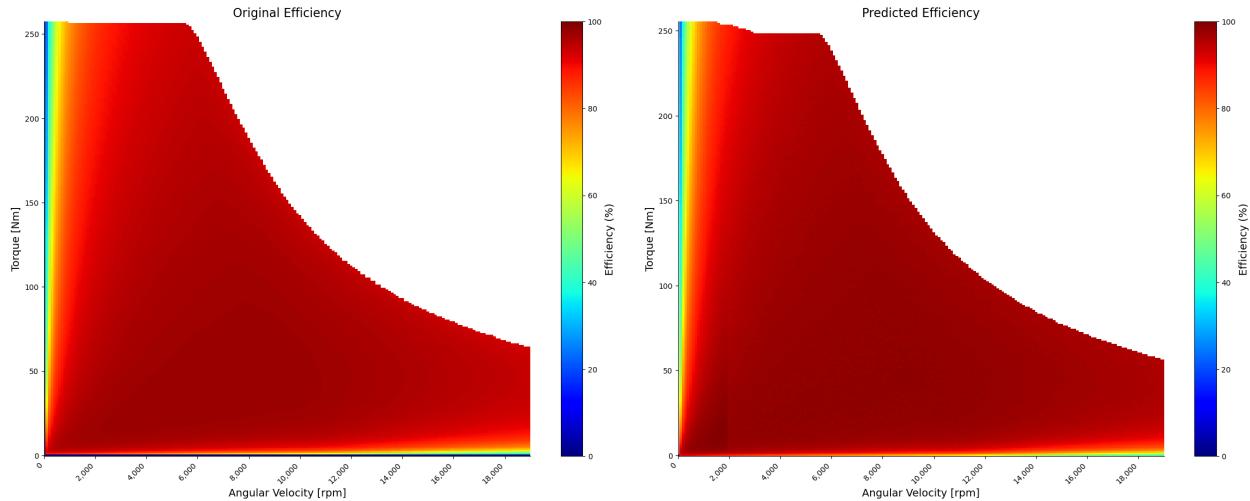


Figure 5.7: 1st MLP Training Results for Efficiency KPI

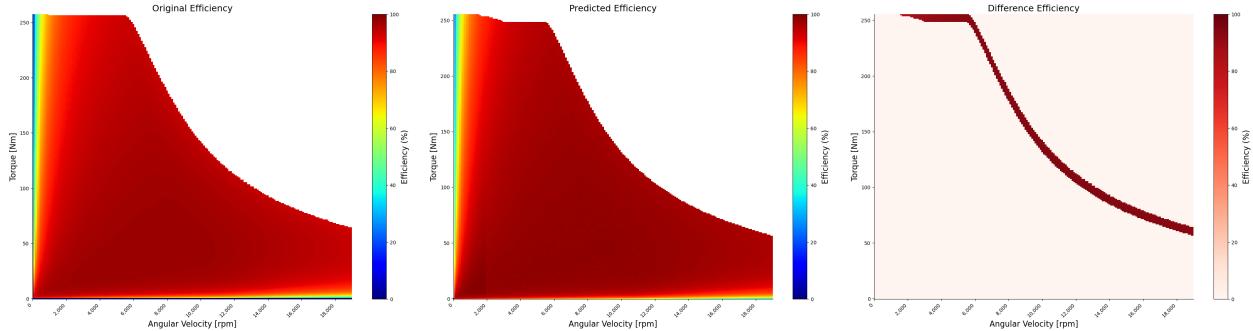


Figure 5.8: 1st Eval MLP Training Results for Efficiency KPI

From a visual perspective, the predictions to relative extent resembles the target and the regularization of the Efficiency KPI seems to be already teaching the model that the envelope should follow the Torque KPI curve shape. The curve being sliced off irregularly is the effect of us trying to retain the Torque KPI shape as was discussed in Section 4.5.

This 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 based on which of the 2 targets to prioritize.

Moreover we calculate the RMSE and differences of prediction from the target by truncating the matrices to be of common shape. We also replace NaN values with 0 in either if it occurs as the Efficiency KPI's were padded with NaN's to be of the same shape but originally had no values as was discussed in Section 3.1.3.

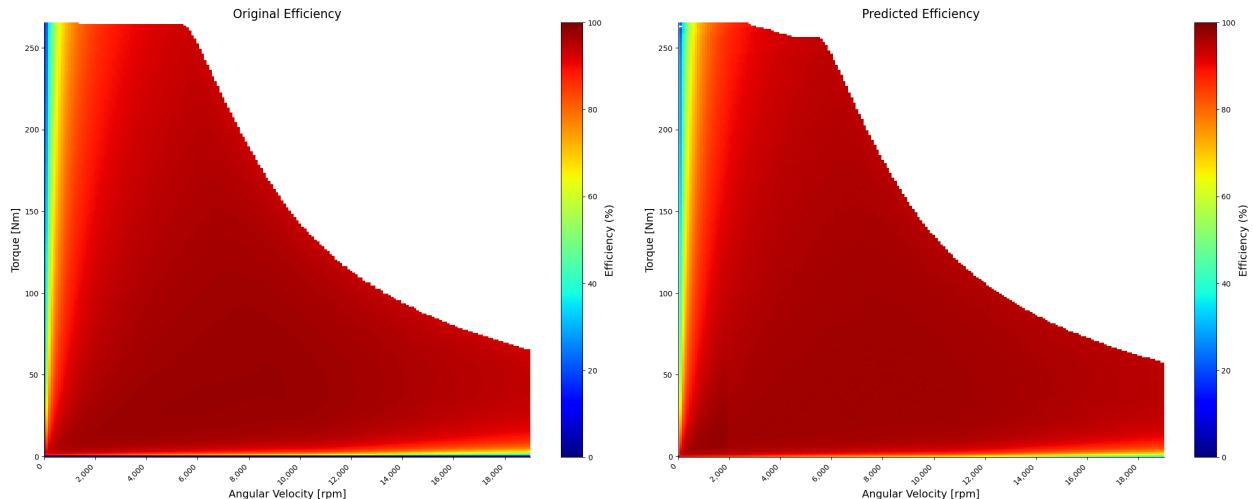


Figure 5.9: 2nd MLP Training Results for Efficiency KPI

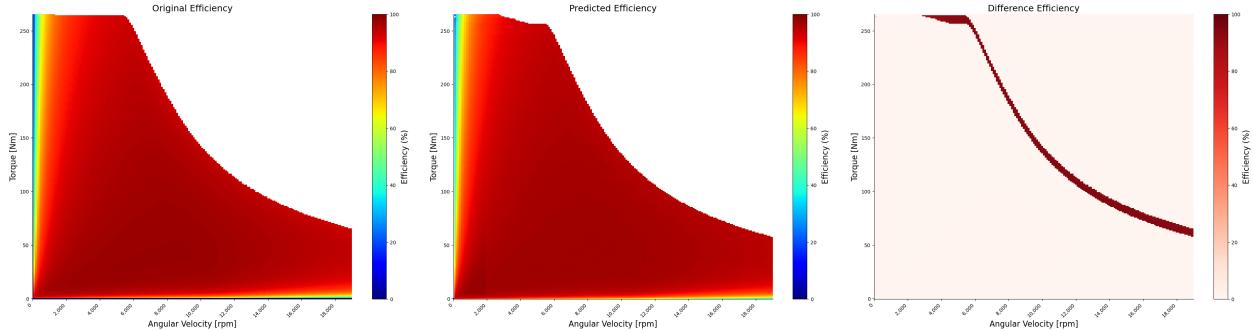


Figure 5.10: 2nd Eval MLP Training Results for Efficiency KPI

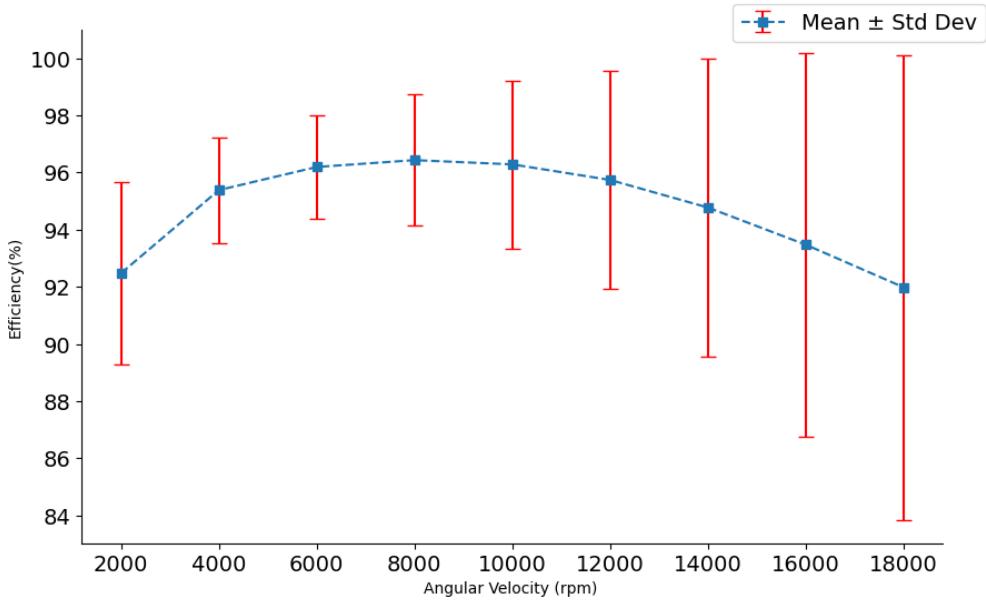


Figure 5.11: MLP Standard Deviation of Efficiency KPI Positive Grid across Speed Intervals

Figure 5.11 shows the standard deviation across different speed ranges for the MLP predictions. In comparisions to Figure 3.6 we infer that as the speed increases the target deviations of efficiency values are not so accurately captured by the model.

Efficiency KPI is harder to evaluate scoring as it is a 3D plot. Therefore, to visualize the Efficiency prediction deviation with its respective targets we do so for specific speeds across the entire torque range with Figure 5.12. The speeds are chosen at equal intervals of 2000 rpm.

We can observe that the most deviation occurs towards the higher torque values for each corresponding speed. In layman terms, this would be towards the Efficiency envelope tapering off.

This anomaly can be explained by the Torque KPI predictions not being up to mark as it is responsible for trimming the edges of the envelope.

However it is not only the border of the envelope a question of concern here but also for neigbouring values of the envelope specifically from 1/4 the speed onwards. We donot have any way to teach the model this part of the grid as loss regularization because we 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 deterioirates from 1/4 the speed onwards is because the Efficiency KPI's envelope starts to converge from around 6000 rpm as we can see in Figure 1.4.

Nevertheless, we see on average among the speeds, the RMSE is close to 5 which is 5% deviation from the target values as per Table 5.2.

To summarize 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 our development for instance in the Efficiency grid we replaced 0 with NaN values which we later understood were both represented different in the grid.

As Efficiency values can take up values only between 0-100, we consider the same as constant across plots and use it as a baseline for determining the levels in the contour plot.

We have also left the output predictions for the Torque curve to remain as float values even when the target values are integers to preserve data precision. We give the client the flexibility to turn this on/off demand.

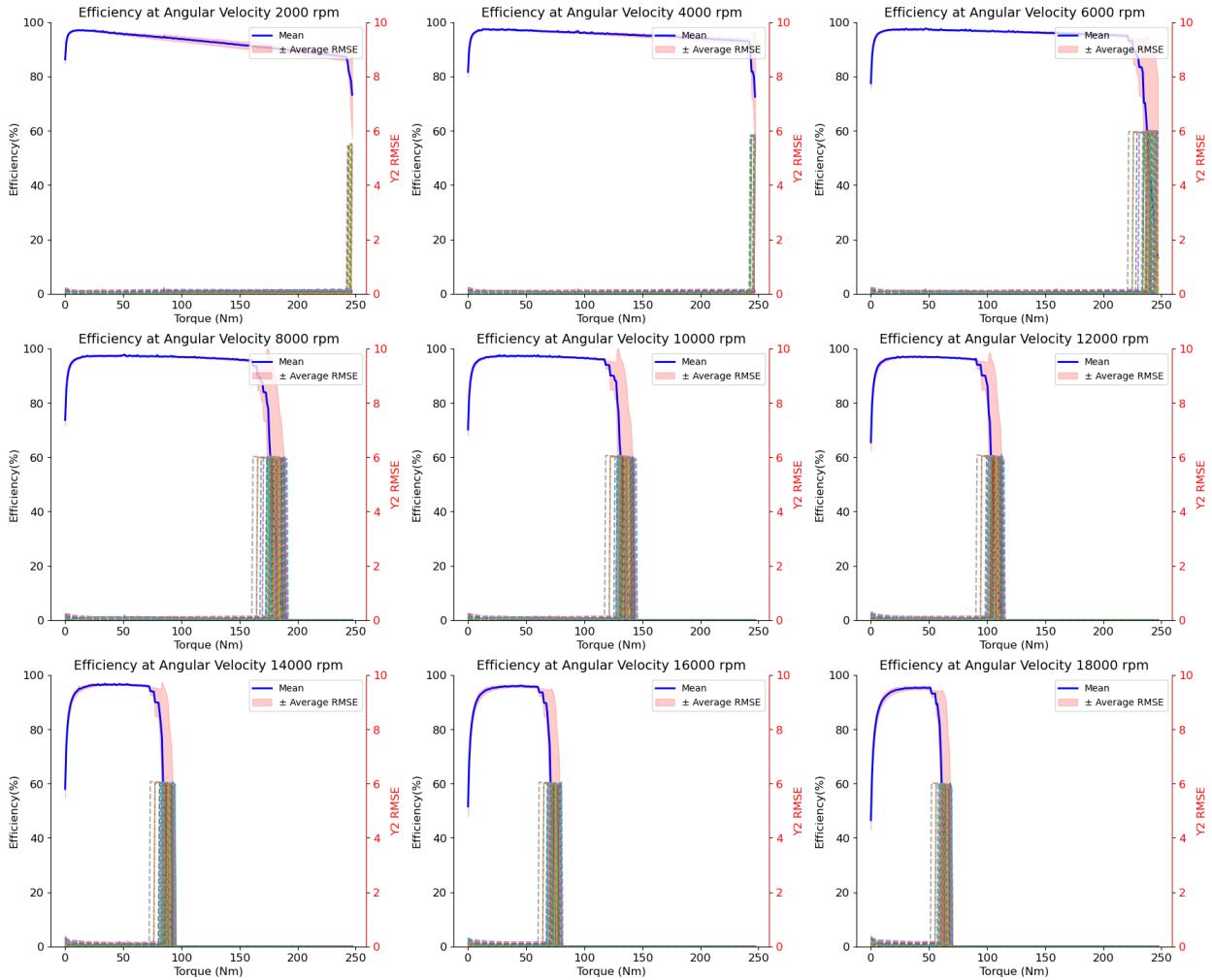


Figure 5.12: Eval MLP Efficiency RMSE KPI

Figure 5.13 shows the score statistics of the model performance of Efficiency KPI over the entire test dataset. The Y2 RMSE from Equation 4.13 is calculated for each sample and shown as a histogram in Figure 5.13a. The Y2 Percentage Difference from Equation 5.2 is calculated for each sample and shown as a histogram in Figure 5.13b.

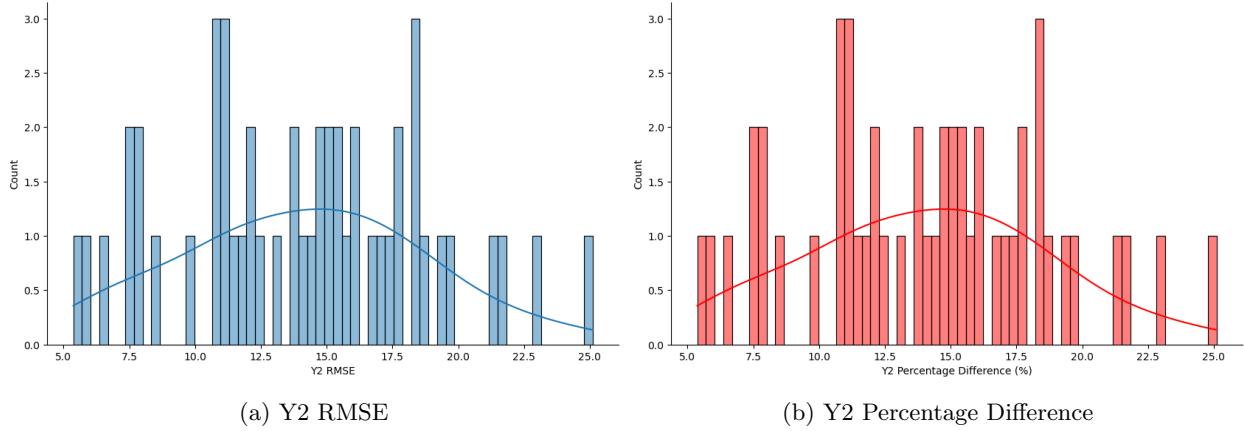


Figure 5.13: Y2 Evaluation Statistics of MLP

We have improved the scores of y_2 because we have had cases where the curve is not contained within the grid, ie, when efficiency values beyond the range 0-100% is encountered. Although, we are aware of this constraint, we havent been able to incorporate this check into the loss regularization for Efficiency KPI. The Y2 RMSE from Equation 4.13 is calculated for each sample and shown as a histogram. On an average the RMSE is close to 13, which is 13% deviation from the target values as per Table 5.2.

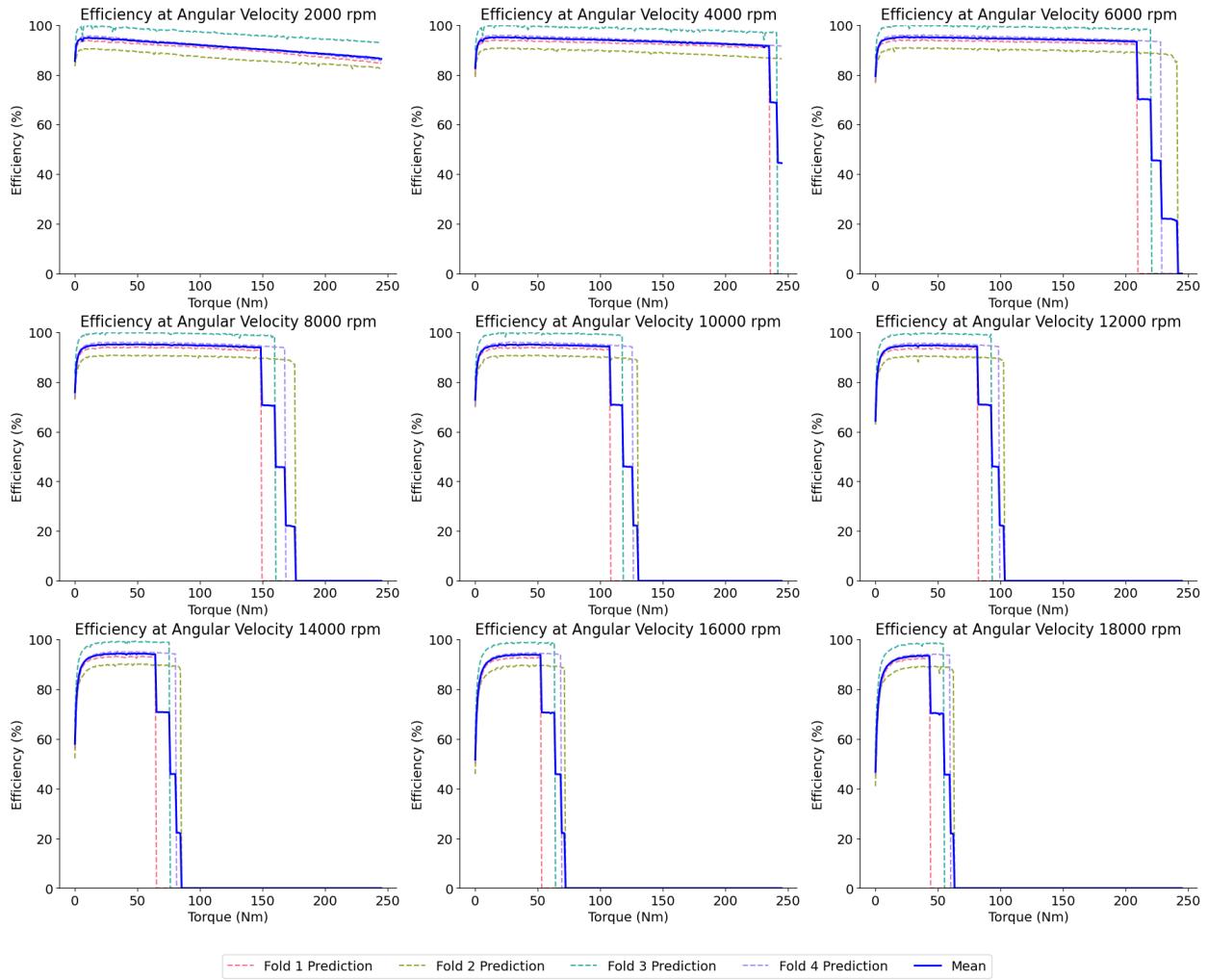


Figure 5.14: MLP Training Deviation across Folds for Efficiency KPI

Figure 5.14 illustrates how each fold's prediction deviates from our observation; it is as close as can be to the mean of all the folds' predictions. This figure is a good indicator of the model's generalization performance and we have used only one engine for inference from the test dataset.

5.3 Results with Baseline

From our observations on how the predictions closely resembled those of the target values in Section 3.1, we have developed a Baseline model which is intrinsically the average of the train dataset.

5.3.1 Torque KPI Results with Baseline

The Y1 Baseline score for the Efficiency KPI is formulated in Equation 5.3 :

$$\text{Y1 Baseline Score} = \frac{1}{n} \sum_{i=1}^n \sqrt{\underbrace{\frac{1}{h} \sum_{j=1}^h (\bar{y} - y_{ij})^2}_{\text{Y1 Baseline RMSE}}} \quad (5.3)$$

where, \bar{y} is the Baseline Average mean, h is the Columns of 1D vector, $Y1$ Baseline RMSE is the RMSE for each test sample.

Figure 5.15 shows the Average RMSE and element wise RMSE for the test dataset performance with the Baseline Model.

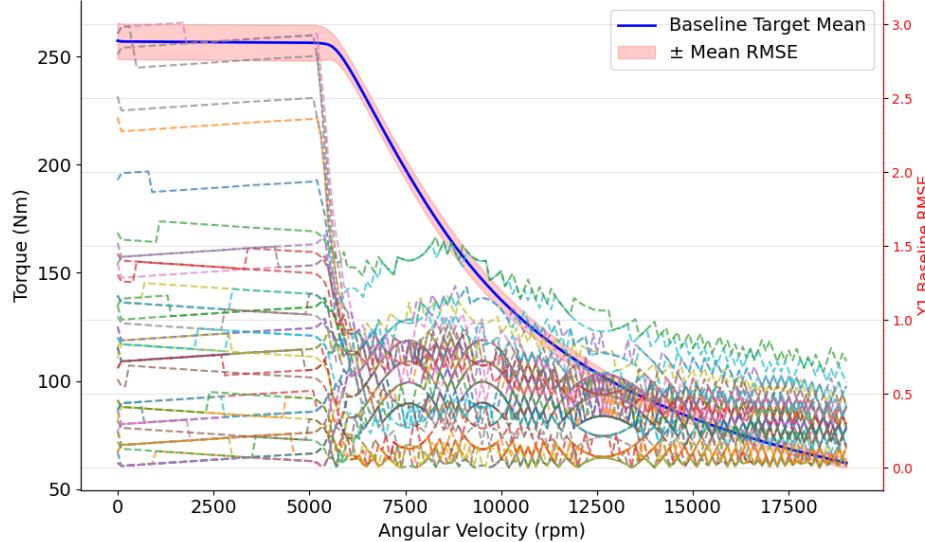


Figure 5.15: Baseline RMSE Evaluation for 2D KPI(Torque)

We can see that the deviations are at its most towards the beginning of the curve. We also observe that there are no fluctuations in the curve as compared to the Predictions.

5.3.2 Efficiency KPI Results with Baseline

The $Y2$ Baseline score for the Efficiency KPI is formulated in Equation 5.4 :

$$Y2 \text{ Baseline Score} = \frac{1}{n} \sum_{i=1}^n \sqrt{\underbrace{\frac{1}{w} \frac{1}{h} \sum_{j=1}^w \sum_{k=1}^h (\bar{y} - y_{ijk})^2}_{Y2 \text{ Baseline RMSE}}} \quad (5.4)$$

where, w is the Rows of 2D vector, h is the Columns of 2D vector and $Y2$ Baseline RMSE is the RMSE for each test sample

Figure 5.16 gives a neat visualization of the Baseline Efficiency RMSE with its respective targets for specific speeds across the entire torque range. The speeds are chosen at equal intervals of 2000 rpm. We infer from the plots that the targets have essentially the same values across the entire grid. We arrive at this conclusion from the fact that the only region where we see significant deviation is at the Efficiency KPI envelope and we attribute this pattern to stem from the corresponding Torque KPI's curve. Thus, the efficiency values are almost the same for more than 3/4th of the grid across all samples.

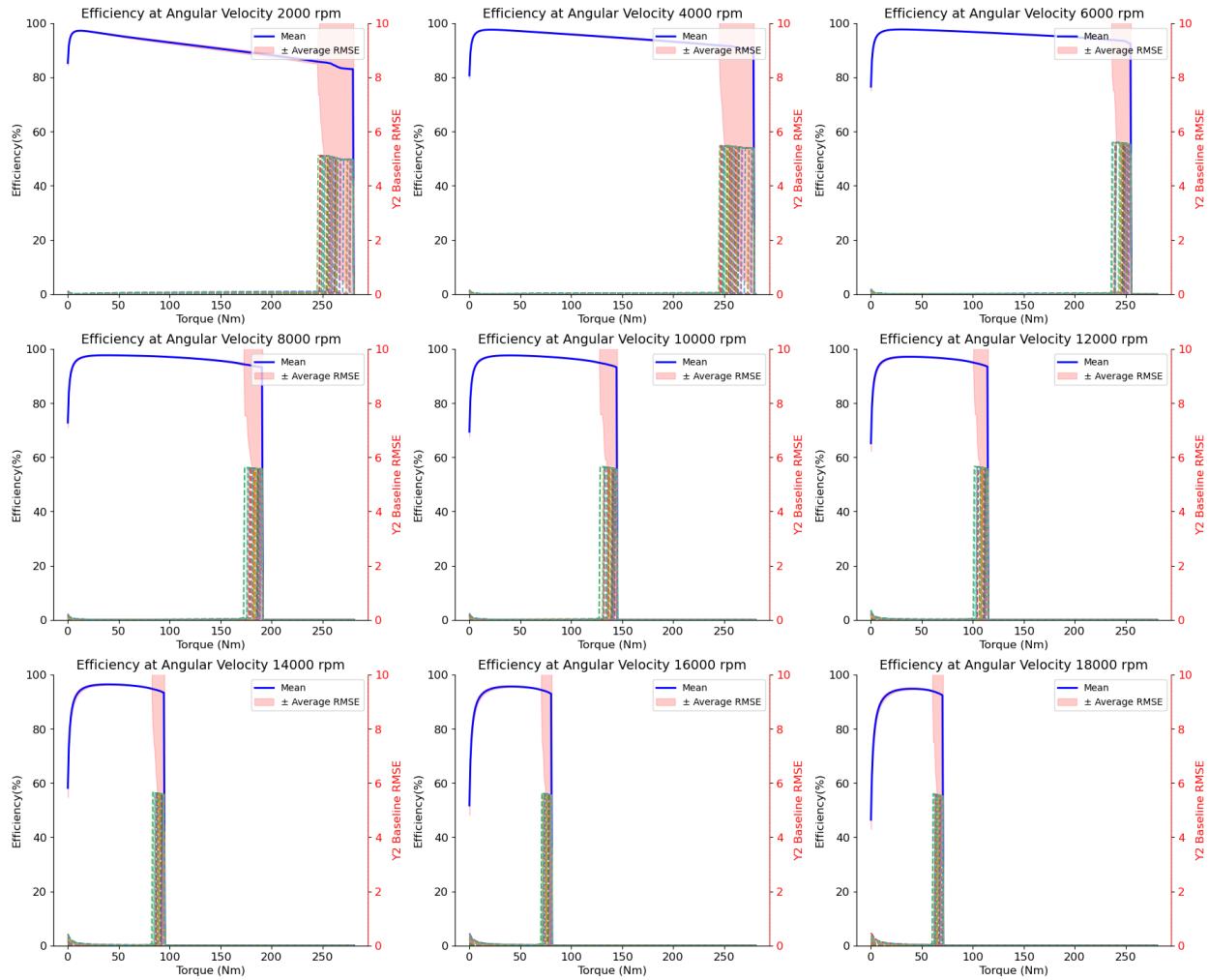


Figure 5.16: Eval Baseline Efficiency RMSE KPI

5.4 Results with Smoothening Loss Regularization

5.4.1 Torque KPI Results with Smoothening Loss Regularization

Figure 5.17 shows the Average RMSE and element wise RMSE for the test dataset performance with the MLP Model with Smoothening Curve regularization discussed in Equation 4.2.

We can see that the deviations are at its peak towards the beginning of the curve. We also observe that there are fluctuations in the curve but relatively less when compared to Figure 5.4

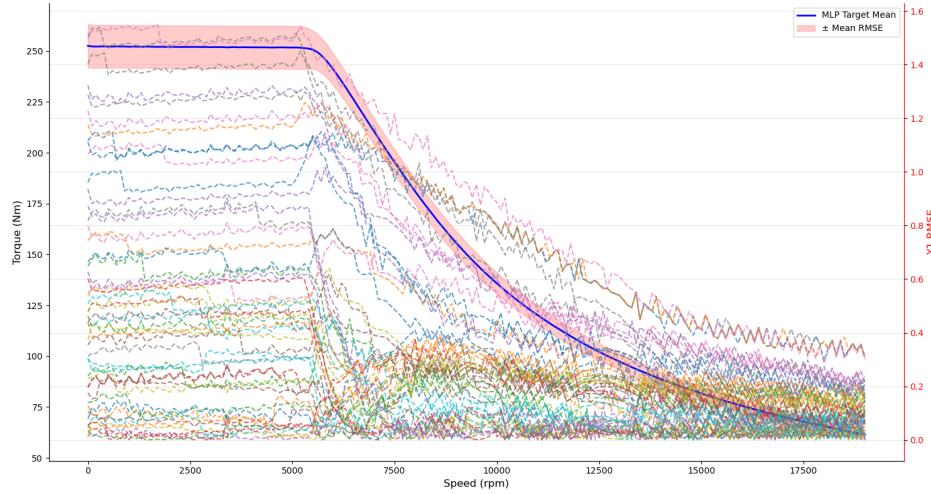


Figure 5.17: Smoothening Torque RMSE Evaluation for 2D KPI(Torque)

5.5 Results with No Loss Regularization

5.5.1 Torque KPI Results with No Loss Regularization

Figure 5.18 shows the Average RMSE and element wise RMSE for the test dataset performance with the MLP Model without any regularizations.

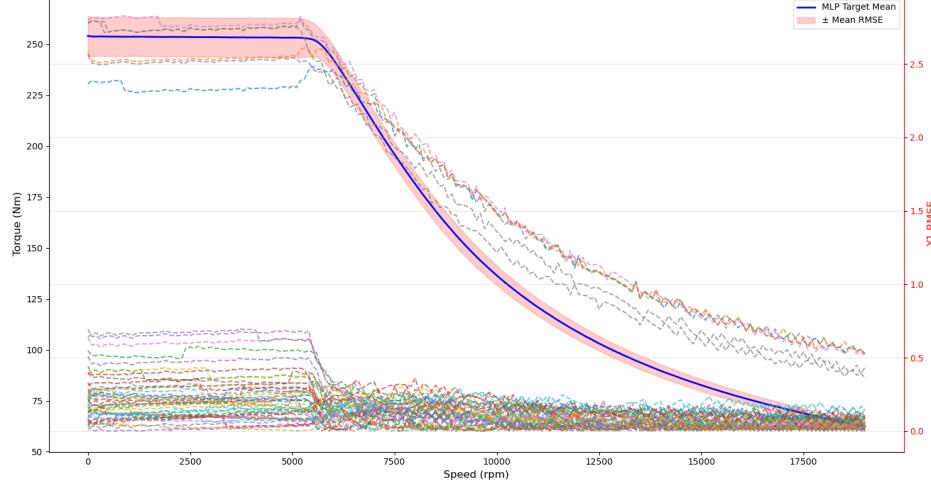


Figure 5.18: No Loss Regularization RMSE Evaluation for 2D KPI(Torque)

We can see that the deviations are at its peak towards the beginning of the curve however the RMSE for most samples is close to negligible. We also note there are fewer fluctuations in the curve as compared to the Predictions with Loss Regularizations. This conclusion prompts us to omit the loss regularization for the Torque KPI's.

5.5.2 Efficiency KPI Results with No Loss Regularization

Figure 5.19 gives a neat visualization of the Baseline Efficiency RMSE with its respective targets for specific speeds across the entire torque range. The speeds are chosen at equal intervals of 2000 rpm.

We observe similar patterns to the ones generated with loss regularization in place. This fuels us to also omit the loss regularizations for the Efficiency KPI's.

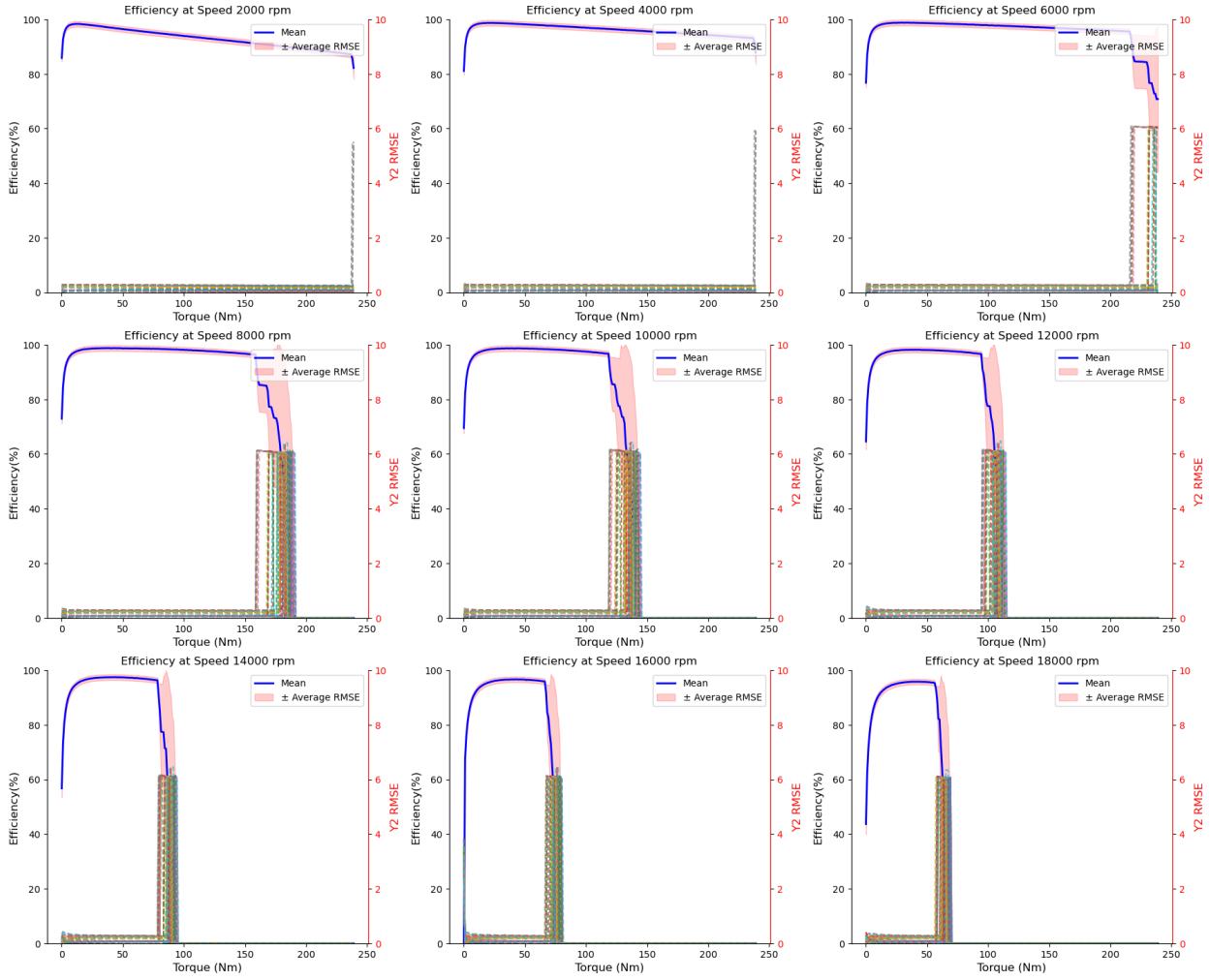


Figure 5.19: Eval No Loss Regularization Efficiency RMSE KPI

5.6 Ablation Studies

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

Model	Y1 Score	Y2 Score
Baseline	4.8201	15.3708
MLP ^a	5.1213	14.099
MLP ^b	5.5052	15.0257
MLP ^c	4.9887	12.6612

Table 5.3: Ablation Studies

^aMLP With Decreasing Y1 Loss Regularization^bMLP With Smoothening Y1 Loss Regularization^cMLP without both Y1 and Y2 Loss Regularization

Results in Table 5.3 indicate that our MLP model without loss 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 we undertook demonstrate the robustness of our method across varying hyperparameter settings. The inferences shown are strictly speaking from the Double V Magnet Topology which assumed the bulk of the data we had received.

We also show a glimpse of the time conserved with surrogate modelling and relate both approaches how we have demonstrated in the flowchart in Figure 1.5

Approach	Time taken(minutes)	Hardware
Surrogate Modelling (Proposed Approach)	2 mins	1500 CPU Cores
FEA simulations (Current Approach) ^a	5 minutes	32 CPU Cores & 1 GPU

Table 5.4: Compute Time Comparision

^aSource : Valeo

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 KPI's predictions and our learnings from it.

We also bring to attention that the time of taking 5 minutes to generate KPIs per design using the Current Approach highlighted in Figure 1.5 is only possible by running the 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. Additionally, it is the FEA simulator in the block that is the culprit of consuming time. Another pointer to note is 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 as reading the excel files take up a significant chunk of the time. The computing time of the ANN surrogate models varied between 50.3 to 68 ms/case, which makes the surrogates 2,911 to 2,154 times faster than the FE reference simulation, respectively. 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 we can potentially run both training and inference across multiple GPUs in parallel.

We have used Python 3.10.14 for our development and the Pytorch library compatible with Cuda. The model was trained on a NVIDIA Tesla V100 GPU with 32 GB Memory and the machine we used had 16 CPUs of the Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz with allocated disk space of 50 GB for our experiments. Source code is available at [Github Repository](https://github.com/Lilly-25/Masters-Thesis)².

²<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. Therefore we presume modelling our usecase as a graph will be more reasonable so that we can exploit graph dynamics and aggregate features that are semantically similar. A graph can be 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 neighbourhoods.

GNNs in general function by making use of Message Passing (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 node features. Over each hop of successive neighbourhoods in the graph, the information of neighbouring nodes of a node are shared across as messages and aggregated and the initial node is recursively updated with this information. This continues until there are no more nodes 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.

On the basis of the directionality of edges within graphs, they can have the following taxonomy: [18]:

1. Directed Graph - A graph in which the edges have a direction from source node to its target node.
2. Undirected Graph - A graph whose Adjacency matrix is symmetrical such that for each edge there will be its respective reverse edge in the opposite direction.

In addition to their ability to incorporate both entity features and network features into a single, simultaneously trained model, most GNNs scale linearly with the number of edges in the network, making them applicable to large networks. 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 [05] A hallmark of GNNs is that they can efficiently learn the architecture parameters from the training data as demonstrated by Owerko et al. in [06].

GNNs cannot be too deep with layers due to the oversmoothing problem. This is because the MP algorithm would have already traversed over all hops and made all features indistinguishable from one another. Besides a deep GNN would have a large number of parameters and would be computationally expensive to train. [22] sheds light on the oversmoothing concern faced by Graph Convolutional Networks. Studies also tells us that it can be mitigated by including residual connections in the network [41].

Researchers have broadly classified GNNs following 2 distinct paradigms:

1. Homogeneous GNN

Homogeneous GNN are designed for graphs with a single type of nodes and edges. MP is done for neighbouring nodes and edges over hops until it learns a representation equivalent from its neighbours. Homogeneous GNN are typically built to capture the structural information within a graph. These models are effective when the node and edge heterogeneity is negligible cited by the authors of [01].

2. Heterogeneous GNN

Heterogeneous GNN are designed for graphs with differing types of nodes and edges implying difference in features as well as dimensionality. The authors of [24] argue that each type of node and edge reveal unique semantic information. 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.

6.1.1 Applications

GNN predictions can be categorized broadly into : [18]

1. Node Level Prediction - represents node classification and regression
2. Edge Level Prediction - focuses on edge classification and link prediction
3. Graph Level Prediction - relate to graph prediction tasks.
Incidentally our task is a Graph Level prediction.

Some of the well known GNN tasks can be seen in Figure 6.1 :

1. Graph Classification -
2. Node Classification -
3. Link Prediction -
4. Community Detection -
5. Graph Embedding -
6. Graph Generation -

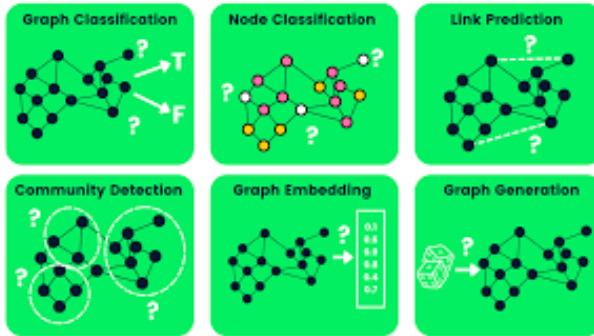


Figure 6.1: GNN Predictions Source :WWW

GNNs have several applications where data are generated from non-Euclidean domains and are represented as graphs with complex relationships and interdependency between objects [18]. Figure 6.2 illustrates various GNNs applications ranging from Recommendation Systems, Chemistry, Traffic, Academic Networks, Information Networks to Social Networks [26].

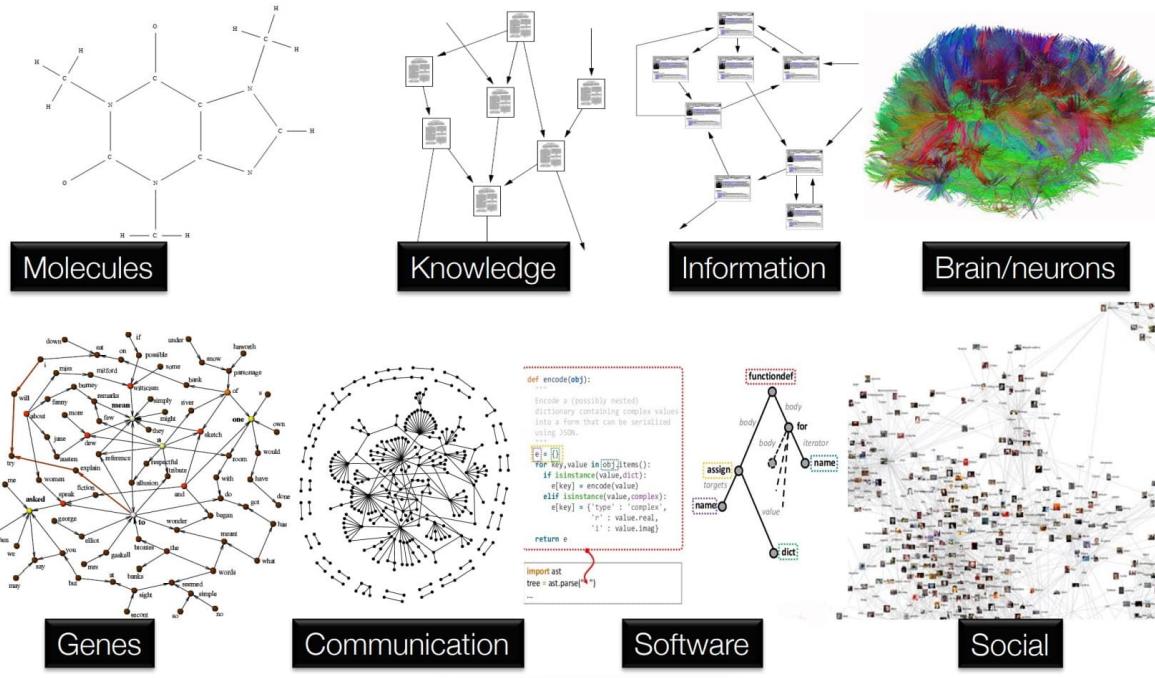


Figure 6.2: GNN Applications Source :WWW

These networks are generally represented by a giant graph in which case graph batching, splitting are unique. However our task requires us to build 1 graph each for each EM variant.

6.2 Heterogeneous GNN

Graphs are ubiquitously heterogeneous because of its capability to have different node and edge types on top of its inherent graph feature to abstract and model relations between objects.

Heterogeneous GNNs also takes into account homophily ie., nodes close on a network have similar embeddings [26] this is in essence the structural property and is true for all GNNs. We largely adopt the commonly used notations from [05] and reformulate it slightly for our task.

Heterogeneous Networks

Given a directed graph $\mathcal{HG} = (V, E, X, R, T_v, T_e, \phi, \psi)$.

The sets of possible node types can be represented as: $T_v = \{\phi(v) : \forall v \in V\}$ and the sets of possible edge types can be represented as: $T_e = \{\psi(e) : \forall e \in E\}$.

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

Further, let us denote by e_{uv}^ε an edge of type $\varepsilon \in T_e$ pointing from node u to v . 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 $R = \{R^\varepsilon \mid \varepsilon \in T_e\}$. Just like for nodes, the edge features may have different dimensions for different edge types.

When $|T_v| = |T_e| = 1$, the graph degenerates into a homogeneous graph [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 $\mathbf{h}_v^{(0)} \in \mathbb{R}^{d_0}$ [04].

Metarelation

For an edge $e = (s, t)$ linked from source node s to target node t , its meta relation is denoted as $\langle \tau(s), \varphi(e), \tau(t) \rangle$. This is in essence the relation triple ie., $(\text{nodetypeof} s, \text{edgetype}, \text{nodetypeof} t)$ used com-

monly in Knowledge Graphs.

Metapath

Given a path $\mathcal{P} = T_{v1} \xrightarrow{T_{e1}} T_{v2} \xrightarrow{T_{e2}} \dots \xrightarrow{T_{ek}} T_{el}$, if the path describes the composition of edges by its types $T_{e1} \circ T_{e2} \circ \dots \circ T_{el}$ 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 i.e., l . The metapath is also the sequence of many metarelations. 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.

Metapath-based Neighbors

Given a node v_1 and a meta-path ϵ in a heterogeneous graph, the neighbors $N_{v_1}^\epsilon$ of node v_1 based on the meta-path ϵ are defined as the set of nodes connected to v_1 via the meta-path ϵ . This set of neighbors based on the meta-path also includes v_1 itself.

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.

The graph structure of G can be represented by a series of adjacency matrices $\{A_r : r \in T_e\}$. For each relation $r_{c_t c_s} \in R$, $A_{c_t c_s} \in \mathbb{R}^{|V_{c_t}| \times |V_{c_s}|}$ is the corresponding adjacency matrix where the nonzero values indicate positions of edges $E_{c_t c_s}$ of the current relation.

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

The Metapaths in Heterogeneous GNNs are divided into 2 streams:

1. Metapath based method
It aggregates the neighbourhood features of the same semantic first and then fuses different semantic information.
2. Metapath free method
It aggregates message from neighbourhood of the same hop for all node types and so captures both structural and semantic information simultaneously.

Multiple MP operations are done across different metapaths which are finally aggregated into a new representation for the node.[05].

Each metapath captures the proximity of nodes in a graph from a specific semantic view [26].

6.3 GNN Literature Review

Existing works for instance Yu et al. in [21] model heterogeneous graphs by splitting the graph into multiple homogeneous subgraphs based on their node types each retaining an aspect of the heterogeneity. This is ineffective in exploiting hidden rich semantic associations between different types of edges for large scale multi-relational graphs. However Zhu et al. [20]'s work does not undertake this methodology and is also independent of metapaths making it effective in dealing with large number of complex relations.

Bias Amplification is a known problem most prevalent in recommender systems where the model is biased to certain outputs due to the training data statistics. It can be handled much better when a graph is modelled as a heterogeneous graph as is shown in [03] this is because differing edge types capture varying types of relationships in the graph.

Modelling Heterogeneous Graphs with metapaths generally require domain specific knowledge as metapaths are customized. In order to combat this, Hu et al. in [23] has used meta relations instead and a unique adjacency matrix for each edge type.

Works by Melton et al. in [25] also use metarelations to extract soft metapaths and does message passing over multi-hop neighborhood of nodes.

Metarelations are most often used in knowledge graphs as it contains richer schema. Such meta relations are shallow embeddings and not as deep as the meta paths as clarified by Yang et al. in [26].

One such work by Zou et al. in [24] proposes a heterogeneous network node classification model which converts the heterogeneous network into multiple semantic graphs through metapaths. The polynomial graph convolutional kernel is also described in this paper.

Li et al. in [22] demystifies the Graph Convolutional Network and addresses the Laplacian oversmoothing concern by self-training or co-training the model. Although it makes classification problem easier over multiple hops the features among different clusters will be the same. GCN achieves this by breaking down the heterogeneous graph into multiple homogeneous ones, one for each edge type. In each layer, GCN is applied to each homogeneous graph, and the resulting node embeddings are element-wise summed to form the final output. In RGCN, during message passing, neighbors under the same edge type will be aggregated and normalized first. A drawback of RGCN is that it does not take node heterogeneity into account.

Most of the existing works demonstrated by Yang et al. in [26] also do not consider the edge features in their model. However Johannessen et al. in [05] uses edge features as transactions in a financial network to detect cases of money laundering. It also breaks down the large heterogeneous graph into subgraphs based on different node type-edge type combinations. The paper also brings to light that GraphSAGE does not incorporate the edge weights.

However, the MP neural network framework applies a learned message-passing function that utilizes edge features.

Additionally Chana et al. in [01] has used heterogeneous GNNs to model the heterogeneity and sparsity in Electronic Health Records of patients. This work takes into account the relational features of the dataset which is arguably the most important factor in this domain. Their work also claim to learn its multi-task nature. However we note that this paper uses meta relations but donot factor in edge features.

Gilmer et al. in [27] introduces modelling that makes it capable of learning features of molecular graphs directly 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. It also implements a message passing network which factors in the edge types.

Dong et al. in [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 so similarly connected nodetypes of other metarelations. Customizing metapaths could bias it to be task specific and is limited to discrete space. 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..TO BE CITED OR Not Yu et al. in [42] addresses the concern of featureless nodes by padding 0s, imputing the mean of neighboring nodes and using pretrained graph embeddings as initial features.

6.4 EM Heterogeneous GNN Model

We find the Heterogeneous graph to be most 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 we will then have similar node types-edge types 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 D , our goal is to construct a heterogeneous graph G from D . Let $T1, T2$ on G be the 2 KPIs we need to learn from D . We aim to train a multi-task GNN model M such that M can deliver high performance on $T1$ and $T2$.

6.4.1 EM Heterogeneous Graph Construction

Inspired by the promising advantages of Heterogeneous GNN, we took the effort of costructing the graph for only the Double V Magnet Topology.

We construct the heterogeneous graph adaptable for each of the EM variants as a `NetworkX`¹ graph.

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 6.1: 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:

¹Python library for graphs and networks

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 $r_i - \text{airgap}$
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 6.2: GNN Edge Types

We define the meta relations as well for each of the edge type

Figure 6.3 shows the heterogeneous graph we constructed

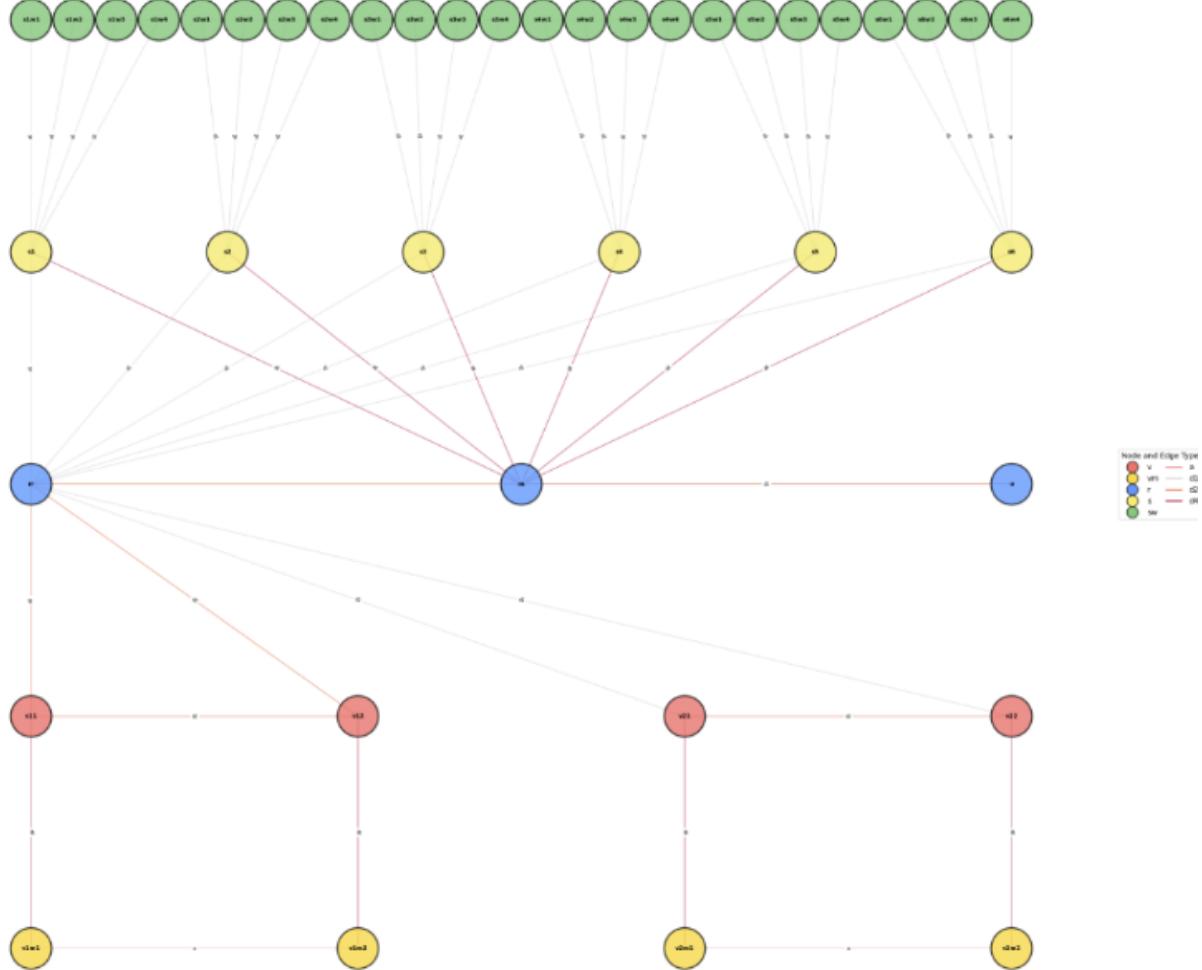


Figure 6.3: HetGraph

The count of certain parameters within the motor such as stator poles with its corresponding stator windings and rotor magnets is made more comprehensible 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. By leveraging the relational features introduced by the meta relations, we can obtain a good graph representation of the data.

We then learn the constructed heterogeneous graph through a heterogeneous GNN model.

Furthermore, we presume graph representation of the data will be more logical than tabular representation due to its ability to grasp the connections within the EM structural properties better. This would be even more realistic with achieving topology invariance than the tabular representation and conserve memory and compute in the long run. However, we realize that our problem cannot be solved using Homogeneous GNN which is relatively simpler and is built on a single node and edge type. In order to model our problem as a graph, we need to represent it as a Heterogeneous graph.

GNNs are relatively more expressive models as they can capture the variability and complexity of our data better.

However, GNN in general have not been less explored even so more the heterogeneous GNN. Particularly in the scenario of EM Modelling, there has been no publications with GNNs. Hence the need to check its feasibility and its performance with our benchmarks on tabular data. Additionally, existing Heterogeneous

GNNs works e.g on recommendation networks, academic networks, information networks, social networks etc involves one large graph with multiple node and edge types. However, our problem involves creation of multiple heterogeneous graphs ie, 1 per EM variant.

Therefore, the applicability of Heterogeneous GNNs for our problem is to be seen.

Our thesis is loosely inspired from [05] which has strived to extend MP into Heterogeneous GNNs.

Chapter 7

Conclusion

This thesis offers a fresh outlook to the possibility of modelling the performance of an EM using GNNs. We have developed a topology invariant MLP model to predict 2 KPI's of an EM from its parametric requirements. It would be very beneficial to EM manufacturers to understand the operating range of the vehicle using the motor and make calculated assumptions of whether to manufacture it.

This work enables EM designers to generate KPI's at almost both negligible costs and minimal time. Thus offering an escape from the cons of using FEA simulations. It also exhibits almost close accuracy to FEA simulations with significantly better run time efficiency PROVE

It also lays the foundation for future work on being able to generate electric motor design parameters conditioned on the 2 KPIs we predicted. We envision that our study will provide new perspectives to researchers engaging in the field of electric motor design and hope our contribution will aid in the same.

7.1 Future Improvements

I would suggest the following improvements to our study :

1. Build a model that uses the Torque KPI prediction to predict its corresponding Efficiency KPI.
2. Although we have designed the model to be topology invariant for 3 topologies, we only had sufficient data from Double V Topology to draw evaluations from. Further evaluations on the other topologies would be beneficial to critically assess our 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 we 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 our problem.

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 Figures 1.3 and 1.4 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

Table A.2 shows the summary statistics of the input parameters of the motor.

Parameter	Unit	Mean	Standard Deviation	Value Range	Single V Mag- net Topol- ogy	Double V Mag- net Topol- ogy	Nabla Mag- net Topol- ogy
General Parameters							
N	-	4	0	4 – 4	✓	✓	✓
simQ	-	6	0	6 – 6	✓	✓	✓
r_a	mm	9×10^{-2}	2.5×10^{-15}	$9 \times 10^{-2} - 9 \times 10^{-2}$	✓	✓	✓

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}	$6.4 \times 10^{-2} - 6.7 \times 10^{-2}$	✓	✓	✓
Rotor Parameters							
rad.phiv2	rad	-0.5	6.36×10^{-2}	-0.6 – 0	✗	✓	✗
lmsov2	mm	-2.8×10^{-4}	3.5×10^{-5}	$-3 \times 10^{-4} - 0$	✗	✓	✗
lth1v2	mm	5.39×10^{-3}	3.5×10^{-4}	$0 - 5.45 \times 10^{-3}$	✗	✓	✗
lth2v2	mm	2.78×10^{-3}	1.7×10^{-4}	$0 - 2.8 \times 10^{-3}$	✗	✓	✗
r1v2	mm	2.09×10^{-3}	3.22×10^{-4}	$0 - 2.2 \times 10^{-3}$	✗	✓	✗
r11v2	mm	3.26×10^{-4}	4×10^{-5}	$0 - 6 \times 10^{-4}$	✗	✓	✗
r2v2	mm	1.8×10^{-3}	1.33×10^{-4}	$0 - 1.9 \times 10^{-3}$	✗	✓	✗
r3v2	mm	6.97×10^{-4}	4.4×10^{-5}	$0 - 7 \times 10^{-4}$	✗	✓	✗
r4v2	mm	7.47×10^{-4}	4.8×10^{-5}	$0 - 7.5 \times 10^{-4}$	✗	✓	✗
rmt1v2	mm	2.49×10^{-4}	1.6×10^{-5}	$0 - 2.5 \times 10^{-4}$	✗	✓	✗
rmt4v2	mm	2.49×10^{-4}	1.6×10^{-5}	$0 - 2.5 \times 10^{-4}$	✗	✓	✗
rlt1v2	mm	1.85×10^{-4}	4.5×10^{-5}	$0 - 2 \times 10^{-4}$	✗	✓	✗
rlt4v2	mm	2.49×10^{-4}	1.6×10^{-5}	$0 - 2.5 \times 10^{-4}$	✗	✓	✗
hav2	mm	4.9×10^{-3}	3.36×10^{-4}	$0 - 5 \times 10^{-3}$	✗	✓	✗
mbv2	mm	1.7×10^{-2}	1.17×10^{-3}	$0 - 1.8 \times 10^{-2}$	✗	✓	✗
mhv2	mm	3.6×10^{-3}	2.65×10^{-4}	$0 - 3.8 \times 10^{-3}$	✗	✓	✗
rmagv2	mm	4.98×10^{-4}	3.2×10^{-5}	$0 - 5 \times 10^{-4}$	✗	✓	✗
dsmv2	mm	2.9×10^{-3}	1.9×10^{-4}	$0 - 3.1 \times 10^{-3}$	✗	✓	✗
dsmuv2	mm	2.9×10^{-3}	1.9×10^{-4}	$0 - 3.1 \times 10^{-3}$	✗	✓	✗
amtrv2	mm	1.58×10^{-2}	1.024×10^{-3}	$0 - 1.6 \times 10^{-2}$	✗	✓	✗
dsrv2	mm	9.96×10^{-4}	6.4×10^{-5}	$0 - 1 \times 10^{-3}$	✗	✓	✗
lmav2	mm	1×10^{-4}	3×10^{-5}	$0 - 1.1 \times 10^{-4}$	✗	✓	✗
lmiv2	mm	1.09×10^{-4}	8×10^{-6}	$0 - 1.10 \times 10^{-4}$	✗	✓	✗
lmov2	mm	5.5×10^{-5}	1.5×10^{-5}	$0 - 1 \times 10^{-4}$	✗	✓	✗
lmuv2	mm	1.45×10^{-4}	10	$0 - 1.5 \times 10^{-4}$	✗	✓	✗
rad.phiv1	rad	-6.9×10^{-1}	5.4×10^{-2}	$-7.8 \times 10^{-1} - 4.5 \times 10^{-1}$	✓	✓	✓
lmsov1	mm	-5.01×10^{-4}	9.4×10^{-5}	$-5.3 \times 10^{-4} - 5 \times 10^{-4}$	✓	✓	✓
lth1v1	mm	2.8×10^{-3}	1.7×10^{-4}	$2.8 \times 10^{-3} - 5.45 \times 10^{-3}$	✓	✓	✓
lth2v1	mm	2.104×10^{-3}	5.9×10^{-5}	$2.1 \times 10^{-3} - 3.2 \times 10^{-3}$	✓	✓	✓
r1v1	mm	4.07×10^{-4}	1.08×10^{-4}	$4 \times 10^{-4} - 2.2 \times 10^{-3}$	✓	✓	✓
r11v1	mm	2.19×10^{-4}	4.5×10^{-5}	$1 \times 10^{-4} - 6 \times 10^{-4}$	✓	✓	✓
r2v1	mm	2.16×10^{-4}	1×10^{-4}	$2 \times 10^{-4} - 1.9 \times 10^{-3}$	✓	✓	✓
r3v1	mm	8.99×10^{-4}	1.3×10^{-5}	$7 \times 10^{-4} - 9 \times 10^{-4}$	✓	✓	✓
r4v1	mm	5.01×10^{-4}	1.6×10^{-5}	$5 \times 10^{-4} - 7.5 \times 10^{-4}$	✓	✓	✓
rmt1v1	mm	2.5×10^{-4}	8.8×10^{-18}	$2.5 \times 10^{-4} - 2.5 \times 10^{-4}$	✓	✓	✓
rmt4v1	mm	2.5×10^{-4}	8.8×10^{-18}	$2.5 \times 10^{-4} - 2.5 \times 10^{-4}$	✓	✓	✓
rlt1v1	mm	1.17×10^{-4}	5.6×10^{-5}	$5 \times 10^{-5} - 2.5 \times 10^{-4}$	✓	✓	✓

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

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}	$5 \times 10^{-4} - 8 \times 10^{-4}$	✓	✓	✓
dhpfp	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 Experimental setup

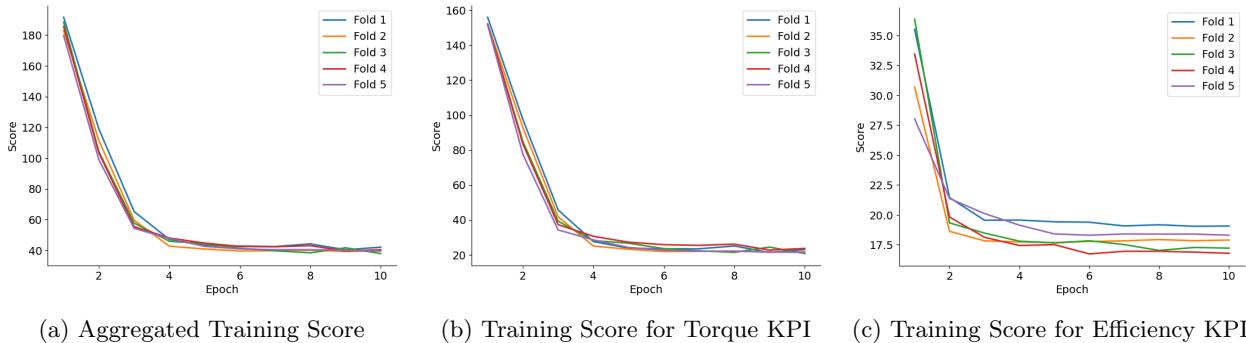


Figure A.1: Training Evaluation Metrics

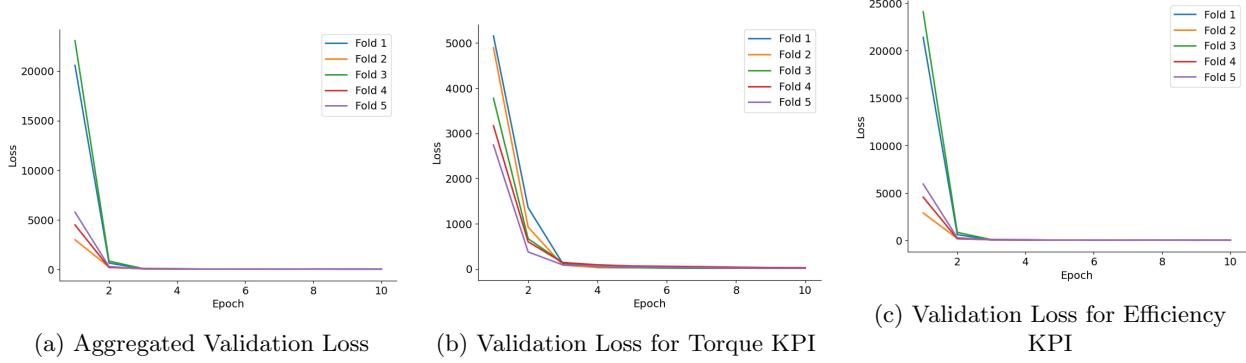


Figure A.2: Validation Loss Metrics

A.3 GNN Terminologies

We largely adopt the commonly used notations from [18] and reformulate it slightly to meet our needs. We 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 $D = \text{diag}(d_1, d_2, \dots, d_n)$, 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, $V = [v_1, v_2, \dots, v_n], v_i \in \mathbb{R}^F$, 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 we have an increasing shaped kernel for GNNs, this is clear in the Figure A.3. 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

$$V' = \sigma\left(\sum_{k=0}^K R^k V \theta^k\right)$$

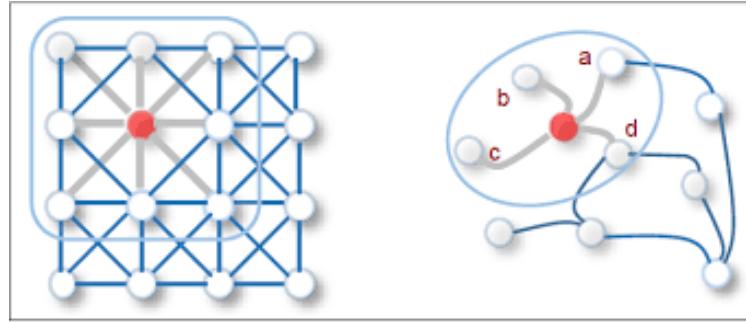


Figure A.3: GNN Kernel Filter Source :WWW

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

$$\begin{aligned} e'_k &= \phi_e(e_k, v_{r_k}, v_{s_k}, u) \\ v'_i &= \phi_v(\bar{e}'_i, v_i, u) \\ u' &= \phi_u(\bar{e}', \bar{v}', u) \\ \bar{e}'_i &= \rho_{e \rightarrow v}(E'_i) \\ \bar{e}' &= \rho_{e \rightarrow u}(E') \\ \bar{v}' &= \rho_{v \rightarrow u}(V') \end{aligned}$$

List of Figures

1.1	EM Motor (Source : Valeo)	8
1.2	EM Magnet Topologies (Source : Valeo)	8
1.3	Torque Curve	9
1.4	Efficiency Grid	9
1.5	EM Design Flowchart	10
3.1	Complete EM Geometry(Source:Valeo)	15
3.2	1/8 Motor Crossection	16
3.3	Standard Deviation of 2D KPI	17
3.4	Standard Deviation of Efficiency KPI	18
3.5	Standard Deviation of Efficiency KPI Positive Grid	18
3.6	Standard Deviation of Efficiency KPI Positive Grid across Speed Intervals	19
4.1	MLP Model Architecture	22
5.1	Training Loss Metrics	29
5.2	Validation Evaluation Metrics	29
5.3	MLP Training Results for Torque KPI	31
5.4	MLP RMSE Evaluation for Torque KPI	32
5.5	Y1 Evaluation Statistics of MLP	32
5.6	MLP Training Deviation across Folds for Torque KPI	33
5.7	1st MLP Training Results for Efficiency KPI	33
5.8	1st Eval MLP Training Results for Efficiency KPI	34
5.9	2nd MLP Training Results for Efficiency KPI	34
5.10	2nd Eval MLP Training Results for Efficiency KPI	35
5.11	MLP Standard Deviation of Efficiency KPI Positive Grid across Speed Intervals	35
5.12	Eval MLP Efficiency RMSE KPI	36
5.13	Y2 Evaluation Statistics of MLP	37
5.14	MLP Training Deviation across Folds for Efficiency KPI	38
5.15	Baseline RMSE Evaluation for 2D KPI(Torque)	39
5.16	Eval Baseline Efficiency RMSE KPI	40
5.17	Smoothening Torque RMSE Evaluation for 2D KPI(Torque)	41
5.18	No Loss Regularization RMSE Evaluation for 2D KPI(Torque)	41
5.19	Eval No Loss Regularization Efficiency RMSE KPI	42
6.1	GNN Predictions Source :WWW	45
6.2	GNN Applications Source :WWW	46
6.3	HetGraph	51
A.1	Training Evaluation Metrics	57
A.2	Validation Loss Metrics	58
A.3	GNN Kernel Filter Source :WWW	59

List of Tables

5.1	Hyperparameter Tuning	28
5.2	Scoring Criteria	30
5.3	Ablation Studies	43
5.4	Compute Time Comparision	43
6.1	GNN Node Types	49
6.2	GNN Edge Types	50
A.1	Excel File Structure of an EM variant	54
A.2	EM Input Parameters	57

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 Owerkowicz, 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] Kazuhisa Iwata, Hidenori Sasaki, Hajime Igarashi, Daisuke Nakagawa, and Tomoya Ueda *Generalization Performance in Predicting Torque Characteristics Using Convolutional Neural Network and Stator Magnetic Flux.* ICDSCA, 3 Mar 2024.
- [10] Simone Ferrari, Paolo Ragazzo, Gaetano Dilevrano, Gianmario Pellegrino *Flux-Map Based FEA Evaluation of Synchronous Machine Efficiency Maps.* WEMDCD, 2021.
- [11] Vivek Parekh, Dominik Flore, Sebastian Schöps *Deep Learning-Based Prediction of Key Performance Indicators for Electrical Machines.* Jan 22, 2021.
- [12] 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.* 2020.
- [13] Yihao Xu, Bingnan Wang, Yusuke Sakamoto, Tatsuya Yamamoto, and Yuki Nishimura *Comparison of Learning-based Surrogate Models for Electric Motors.* COMPUMAG, 2023.
- [14] Marius Stender, Oliver Wallscheid, Joachim Böcker *Accurate Torque Estimation for Induction Motors by Utilizing a Hybrid Machine Learning Approach.* PEMC, 2021.
- [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] Diederik P. Kingma, Jimmy Ba *Adam: A Method for Stochastic Optimization*. 30 Jan 2017.
- [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.

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