Technical University of Applied Sciences Würzburg-Schweinfurt (THWS)

Faculty of Computer Science and Business Information Systems

Master Thesis

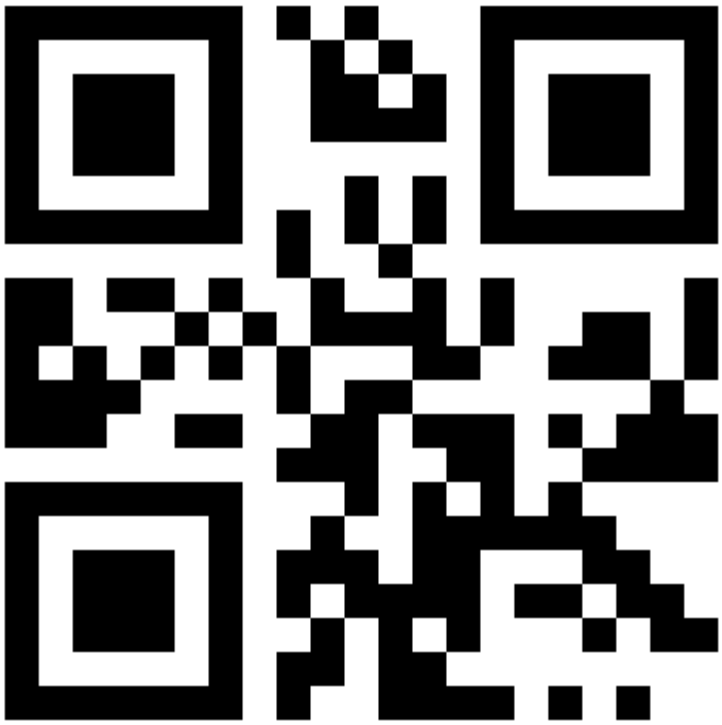
Electrical Engine Efficiency Prediction Bypassing Finite Element Analysis

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 Gregorova
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 Magnets (IPSM) 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 parameteric 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).

Acknowledgement

I would like to thank my supervisor Prof. Dr. Magda Gregorova 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 thoughout.
Finally, I humble myself before God Almighty for all his blessings and for giving me the strength to persevere and bring my dreams to fruition.

Contents

Abbreviations

GNN	Graph Neural Network
MLP	Multi Linear Perceptron
KPI	Key Performance Indicator
EM	Electric Motor
FEA	Finite Element Analysis
CNN	Convolution Neural Network
D	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
IPSM	Interior Permanent Synchronous Motor

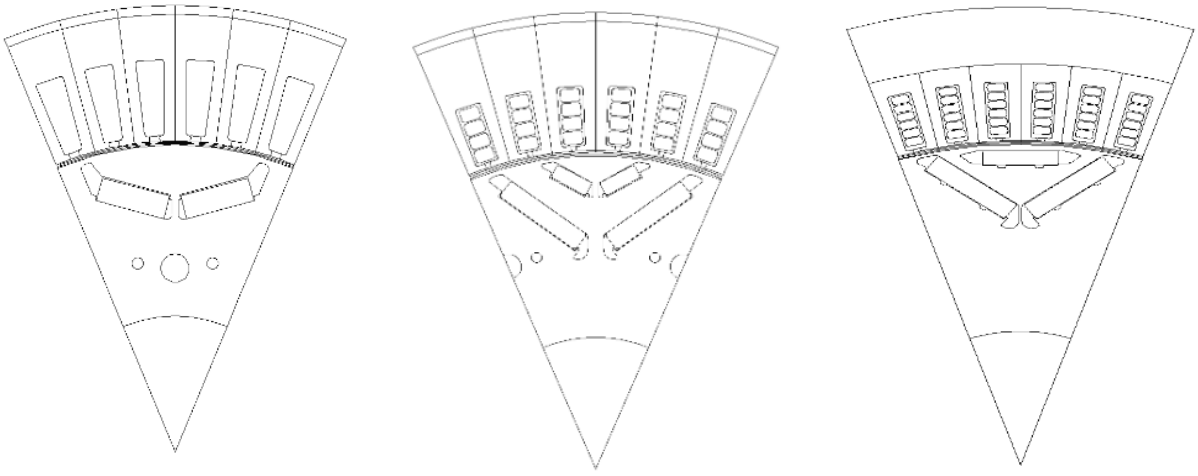
Chapter 1

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.
KPIs of an Electric Motor are its representative characteristics and is essential to judge the performance of the motor before it is manufactured.
Traditionally these KPIs are inferred from a description of an EM design via a Finite Element Analysis (FEA) approximating the solutions of the Maxwell's equations.
This process, though well established in the EM design, is very 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.

The 3 motor topologies manufactured by Valeo are displayed in Figure 1.1:

1. Single V Magnet - Consists of a single V magnet shown in Figure 1.1a.
2. Double V Magnet - Consists double V magnets shown in Figure 1.1b.
3. Nabla Magnet - Consists of a single V Magnet and a delta magnet shown in Figure 1.1c.



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

Figure 1.1: EM Magnet Topologies (Source : Valeo)

Figure 1.2 and Figure 1.3 give a glimpse of the EM KPIs to be predicted.

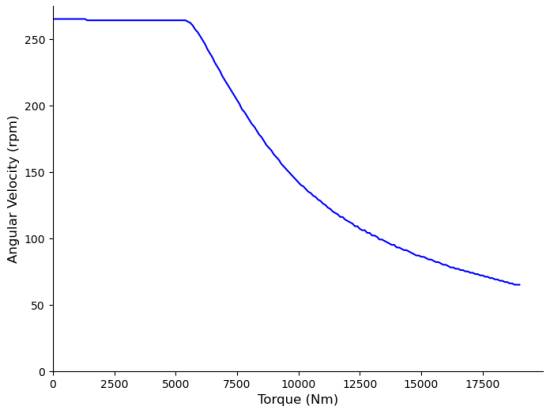


Figure 1.2: Torque Curve

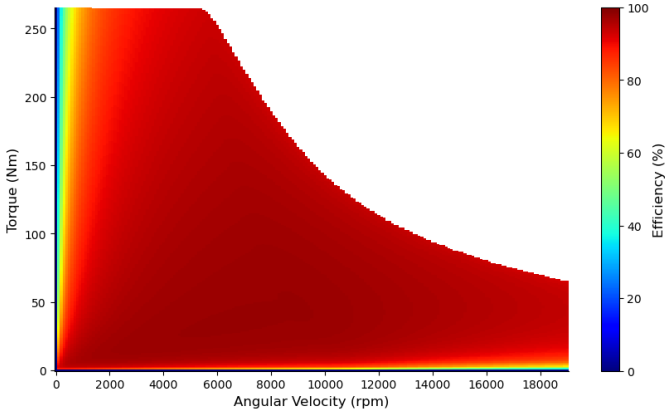


Figure 1.3: Efficiency Grid

FEA simulations are multi objective optimization methods wherein repeated numerical Partial Differential Equation (PDE) based simulations are required to effectively evaluate the characteristics of the input data by tuning simulation settings. It requires domain knowledge in motor physics and complicated setup to run simulations typically in Matlab. FEA simulations are time expensive because it is applied at each operating point.

In the field of computational electromagnetics, deep learning has been viewed as a promising substitute to physics based simulations as they can provide relatively equivalent accurate results in almost negligible 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.

PMSMs are largely classified as Surface Mounted PM and Interior PM machines 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 paper in [35] also cites that the efficiency vs torque and speed map is most representative of he characteristics of the EM.

The current pipeline to predict the KPIs of different EM design variants is to create a design mesh from the parametric description of the motor with Matlab. Multiple FEA simulations which are by nature Partial Differential Equations(PDE) is done on this mesh which is then post processed and the intermediary outputs are forwarded to the Motor Builder. Several Motor builder settings are then adjusted to get the plots of the desired KPIs.

This master thesis explores a way to do light weight surrogate modelling of the current process as is highlighted in Figure 1.4 by exploiting data-driven deep neural networks to approximate the KPIs derived from FEA simulations.

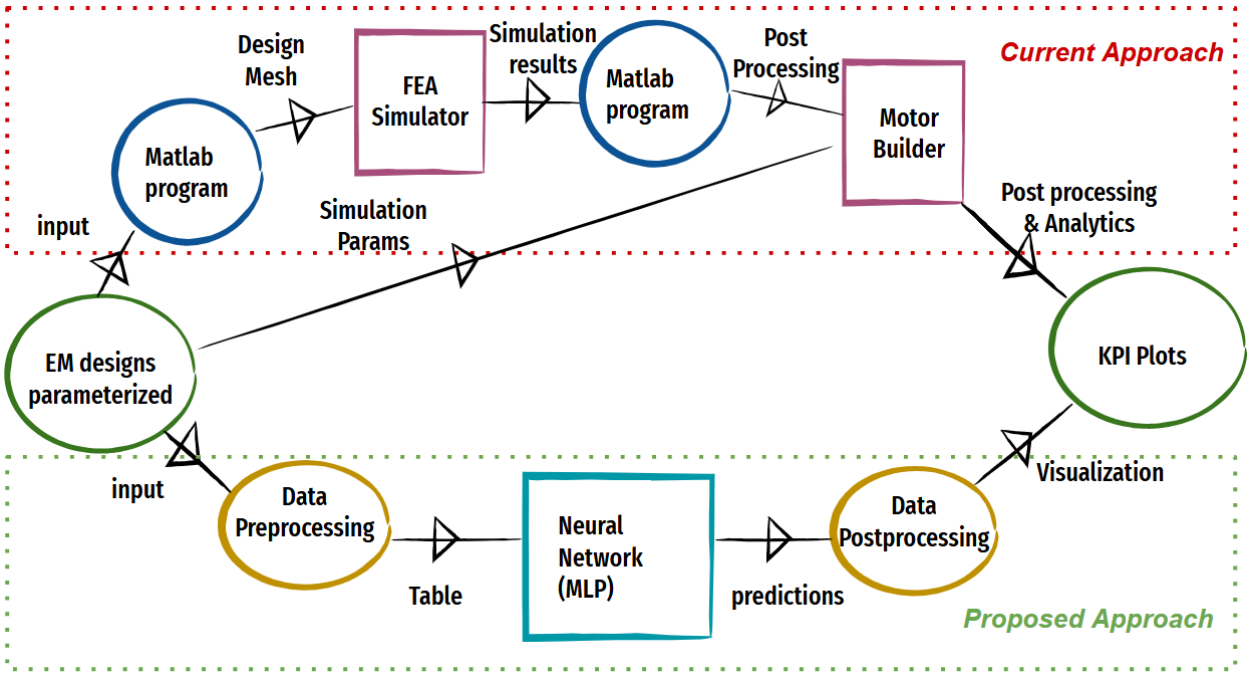


Figure 1.4: EM Design Flowchart

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 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

5. 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.

1.3 Research Question

GNNs are relatively more expressive models as they can acapture the variability and complexity of our data better.
However, GNN in general have not been to 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.

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 dependendent 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.

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

1.4 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 out task.
Conclusion chapter summarizes the thesis briefly and would also give a glimpse into 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.
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. This process is incredibly time consuming because fitness evaluation will need to be performed multiple times with FEA. 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.

The authors of [08] presents a method to predict 2Dimension (D) flux maps of an Interior Permanent Synchronous Motor (IPSM) 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, in [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 realise 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. [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. 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.

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 1.1. 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 auxillary 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., excitation points are predicted. This results ni 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 uncertaininty 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. The do so by quantifying the uncertainity of the predictions by adding a probablistic component to the wights of the neurons in the network architecture. This technique they employ is called Monte Carlo dropout with which they generate uncertainity 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 1.4.

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 forseeing 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.
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 an automotive company has supplied the dataset consisting of close to 1500 Double V Electric Motor parameters. 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.

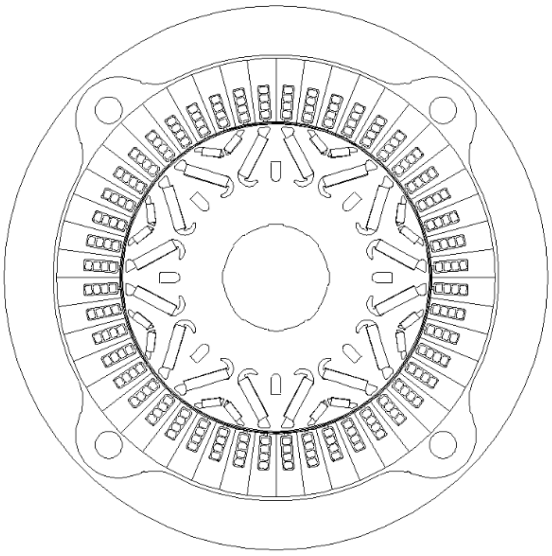


Figure 3.1: Complete EM Geometry(Source:Valeo)

Figure 3.2 is a handdrawn 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.

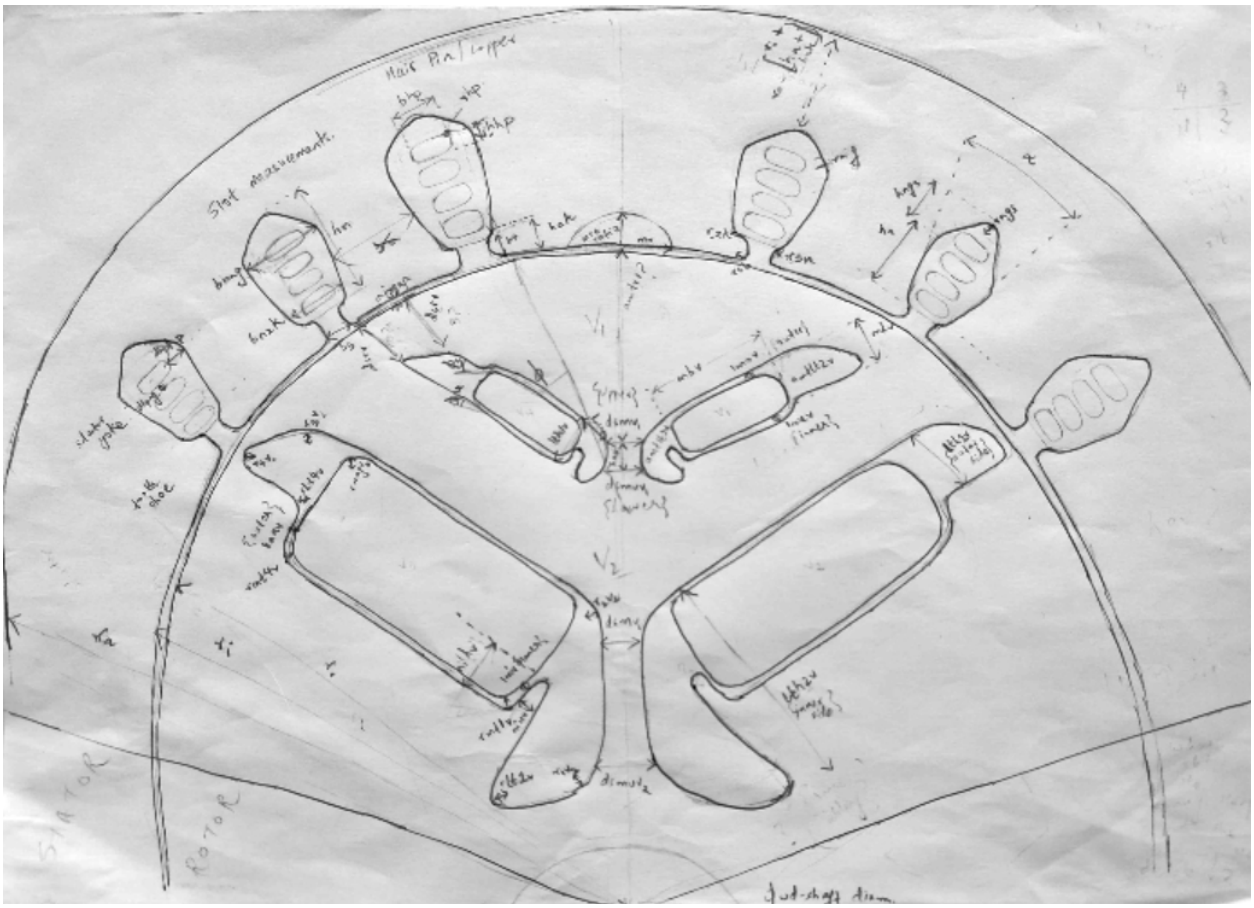


Figure 3.2: 1/8 Motor Crossection

Valeo has shared 1481 Excel Workbook files for each motor variant of three-phase IPSM motors with 8 poles and 48 slots. Each of the excel files contain multiple sheets. Table 3.1 summarizes the sheets of interest to us:

Sheet	**Sheet Name**	**Description**
Motor Parameters	input_data	Contains the input parameters for the motor model. Includes geometric, physical, and simulation properties. Unit dimensions, if applicable, are generally in mm or degrees.
Speed Grid	NN	Contains the speed grid. Used for plotting the Torque KPI and Efficiency KPI. Unit dimensions is rpm.
Torque Grid	MM	Contains the torque grid. Used for plotting the Efficiency KPI. Unit dimensions is Nm.
Efficiency Grid	ETA	Contains the Efficiency KPI. Has the same dimensions as the NN and MM sheets. Percentage values do not have unit dimensions.
Torque Curve	Mgrenz	Contains the values corresponding to the Torque KPI. Has the same columns as the speed grid. Unit dimensions is Nm.

Table 3.1: Excel File Structure of an EM variant

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 the data of the other topologies for that corresponding column is treated as 0 values.
The values are read and stored in their float equivalent to preserve 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.

3.1.2 Data Exploration of the Torque KPI(Torque Curve)

Figure 3.3 shows the standard deviation of few samples of the Torque KPI.

We make the below observations from it :

1. The Root Mean Squared Error (RMSE) is at its peak at low speeds.
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.

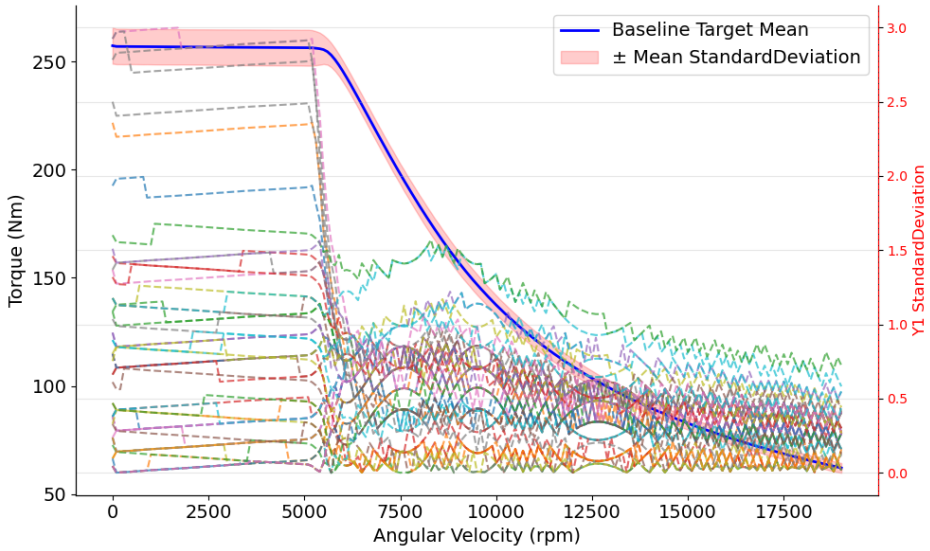


Figure 3.3: Standard Deviation of 2D KPI(Efficiency) Targets

3.1.3 Data Exploration of the Efficiency KPI(Efficiency Grid)

As the target values Torque KPI and 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 create a similar grid ranging from -maximum torque to maximum torque.
We then choose only the rows corresponding to this range from the actual MM grid supplied and the same

row indices is used to retrieve the Efficiency KPI.
This step ensures that we grant the model the correct dimensions of the Efficiency KPI based on Torque KPI and predict likewise.

Efficiency values for negative torque values correspond to when motor is in generating mode and those of positive torque values 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 data. This is evident from low speed-high torque distribution area where we can see 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 only predict the positive Efficiency KPI.
The latter can be mirrored in order to obtain the negative Efficiency KPI values if necessary. It can be mirrored to replicate the efficiency when it is in generating mode.

From Figure 3.5, we can observe the deviation is at its peak at low torques, low speeds and the border of the curve within the grid. We discuss the modelling of this information in Section 4.2.2.

Additionally 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 operating cycle. This is yet another finding we use in Post Processing as is further elaborated in Section 4.5

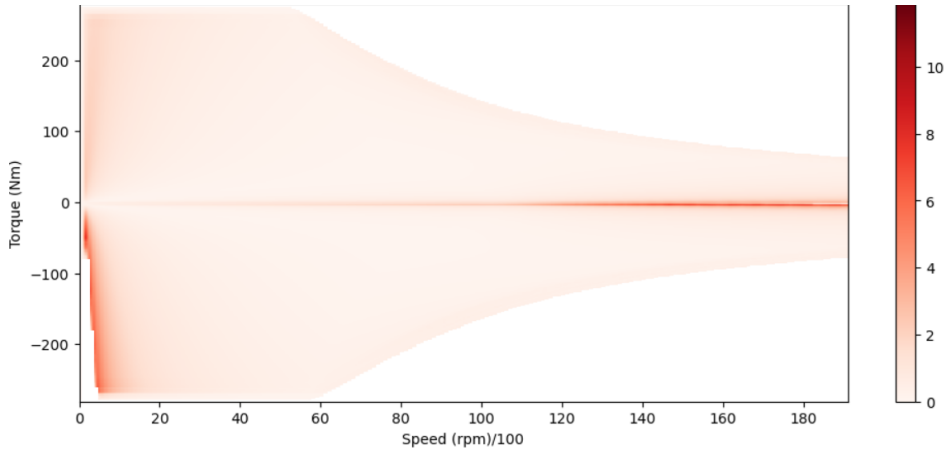


Figure 3.4: Standard Deviation of Efficiency KPI

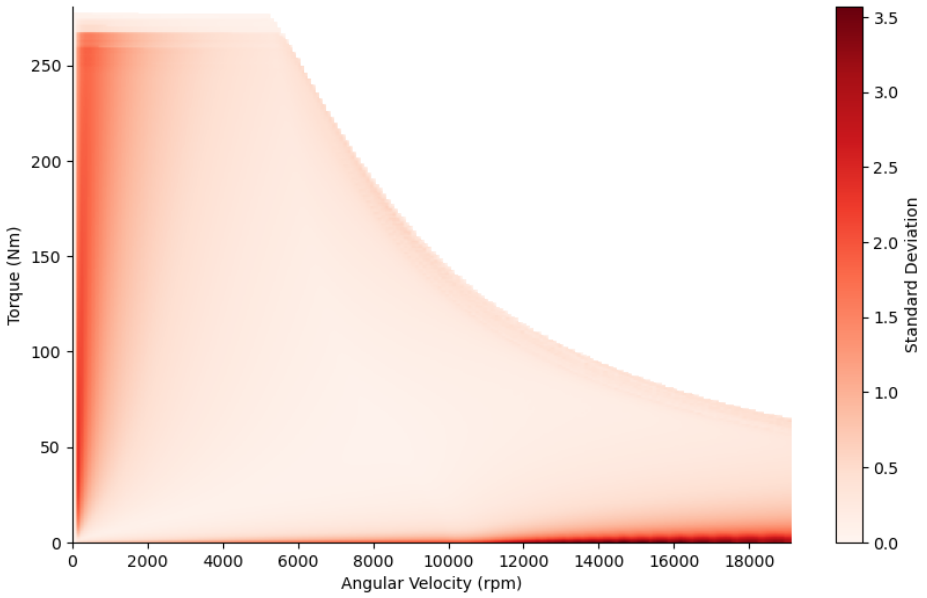


Figure 3.5: Standard Deviation of Efficiency KPI Positive Grid

Figure 3.6 gives us the big picture of how the efficiency values are distributed across equally spaced intervals of speed. The distributions shows skewness towards extreme speeds, extreme torques. We discuss in Section 4.2.2 how we integrate this finding into teaching over model. In addition, we notice skewness at the border of the curve within the grid, we discuss in Section 4.5 how this is learned.

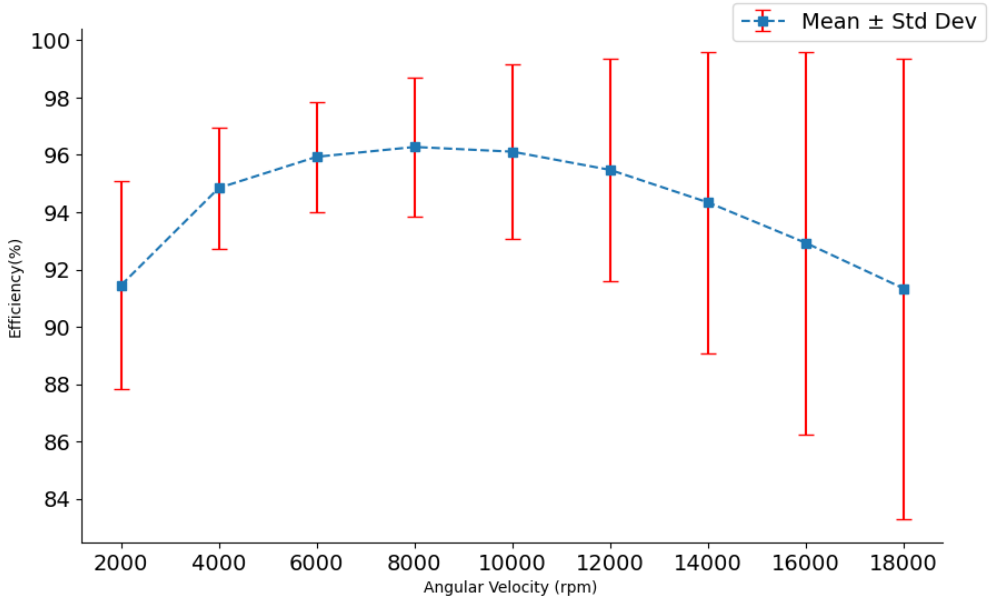


Figure 3.6: Standard Deviation of Efficiency KPI Positive Grid across Speed Intervals

Due to the infamous fact that reading sheets in excel files take up a lot of time and compute, we read the files as a onetime job when creating the table and store them into pythonic objects for faster access for training.
Both the input and target values for the Torque KPI is 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 but this is subject to change as we receive more data and can be overridden by the user. 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 discuss 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

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

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

For the Input features both Mean and Standard deviation are calculated across columns. This is attributed to the fact we have columns with different ranges for the input.

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 scaler for comparision.

During our experimentations where we initially scaled the targets, we observed that then 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 float 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

MLP is a feed forward network with no recurrent connections back to previous layers. The first layer of the network is the input layer, which distributes the input values to each neuron in the first hidden layer. A hidden neuron can be thought of as a simple processing unit, as shown in Fig. 2. The input values coming from the neurons of the previous layer has its own weights, and each hidden neuron has its own bias parameter. The weighted inputs and the bias are added up and the sum is fed for the input to activation function, which can be linear or nonlinear. The output of the activation function is then distributed to the next layers, or if the neuron is in the output layer, it is the model output

the weights and biases are adjusted so that the model fits the training data

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 were also added in between to serve as the activation function and produce non-linearities and so 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. The 2 dropouts are for shared layers of the MLP and for the layers corresponding to Efficiency KPI.
Batch normalisation layers are used to normalize the input from the ReLU activations applied on it and so mitigate internal covariate shift to the next layer and hence speed up the training process.
Thus both batch normalisation 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 learnable size we tune. We donot 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 Normalisation 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 at the start to learn a rich represntation of the data at the intial feature extraction phase.

3. **MLP Torque**
   This block comprises of Sequential 1 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 layer as the targets are inherently always positive values and exploit ReLU's behavior of clipping negative values to 0.

4. **MLP Efficiency**
   This block comprises of Sequential 3 Linear Layer 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 normalisation, 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 targets 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. 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 else 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 collated array we created as was discussed in Section 3.1.3

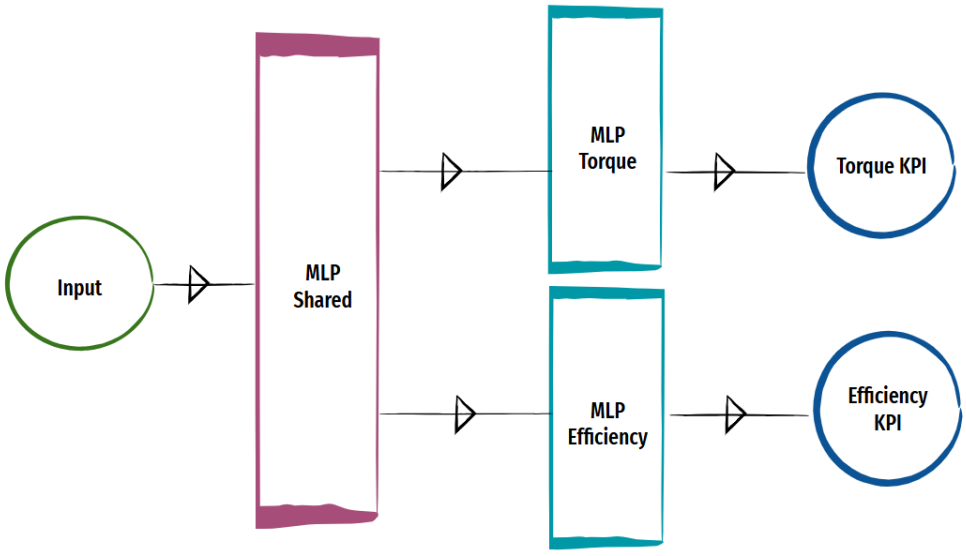


Figure 4.1: MLP Model Architecture

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 it further. In addition to its contribution in exaggerating the loss, MSE also ensures that deviations are positive and do not confuse the model by negating the losses of different 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

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

where, n is EM samples, h is the columns of 1D vector, the prediction and y_{ij} and $\hat{y_{ij}}$ are the ij-th ground truth and the prediction.

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 take into account.
   This is formulated mathematically as in Equation 4.2.

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

   This nature is factored in by penalising the loss by the magnitude if the neighbouring 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 non-decreasing values within each prediction.
   This is formulated mathematically as in Equation 4.3.

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

   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 donot 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 is formulated as in Equation 4.4.

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

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

where, M_{ijk} is Mask matrix, w is the Rows of 2D vector and h the Columns of 2D vector.

The Efficiency KPI is a 3D plot of real numbers representing percentage values and is always in the range of 0-100%.
We noticed in some portions of the Efficiency KPI, the plot not visible as it had NaN values.
As ANN cannot be trained to predict NaN values we have a binary mask 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.

$$\text{Y2 Maximum Efficiency Regularization} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h-1}\sum_{j=1}^{h-1}(\text{ReLU}(|\hat{y}_{ij+1} - \hat{y}_{ij}| - 1))^2 \tag{4.6}$$

   This nature is factored in by penalising the squared loss by the magnitude if the values in the prediction exceed 100. ReLU again assists to clips the difference if it is negative which is the scenario 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. However, these predictions are not relevant for us as after generating all predictions we finally slice off the Efficiency grid to be of the dshape of the Torque curve which implies that the values predicted in place of NaN are irrelevant. 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 and affect its focus on predicting the other values in the 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, we have tried to incorporate all of the below learnings via the loss function as Y2 Regularization.

   (a) **Efficiency at Maximum Torque Loss Regularization**

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 our Efficiency KPI and those of its target values and penalise the squared difference to have higher weight.
We formulate it mathematically as in Equation 4.7

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

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

(b) **Efficiency at Low Speeds Loss Regularization**

To force the model to pay more attention at lower speeds, we have regularized the loss for the first few columns of each row of the Efficiency grid and penalise the squared difference with that of the target.
We formulate it mathematically as in Equation 4.8:

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

where, t_2 is the Threshold for Low Speed
The number of first columns is determined by a threshold t_2.
It is a known fact that at 0 Torque, the corresponding efficiency values for the motor is 0. With this regularization, this learning as well is incorporated into the loss function.

(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 as 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 penalise the squared difference with that of the target.
We formulate it mathematically as in Equation 4.9:

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

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

We aggregate all the above regularizations to form the Y2 Loss Regularization as in Equation 4.10

$$\begin{aligned}\text{Y2 Efficiency Grid Loss Regularization} = \text{Y2 Loss Regularization MM Max Torque} \\ +\text{Y2 Loss Regularization Low Speed} + \text{Y2 Loss Regularization Low Torque}\end{aligned} \tag{4.10}$$

The Total Loss is calculated in Equation 4.11

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

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_{y21} is Y2 Efficiency Grid Loss Regularization Parameter, wt is Y1 Loss Weightage, $1 - wt$ 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 :

1. When the targets are not of the same scale.
   Without scaling, the losses for both targets 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 perdiction performances.
   We reflect on these decisions based on whether the envelope of the Efficiency KPI grid is more valuable that 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 little 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 :

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

where, h is the Columns of 1D vector and $Y1$ 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 :

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

where, w is the Rows of 2D vector, h is Columns of 2D vector and $Y2$ 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 we first convert it into the tabular representation our model consumes and then apply the scaling.
Hence the reason 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 KPI's, the grid contains values even outside the boundary of the Efficiency KPI. Hence, we attempted 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.

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 training.
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 an exponential learning rate scheduler which reduces the learning rate exponentially by the *lr gamma parameter* 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 2 hidden sizes only as we are constrained with 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 to it being of the multiples of 8 as GPUs are most optimized for the same. Dropout rate during training is controled 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 donot concern much about 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 comparitively 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 regularization terms for 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 the simple architecture of the model is not a significant downside for that limited data set.

We chose a 3rd party application `Wandb`[1] to log metrics from the training run and to monitor model performance across the 5 folds.
Our MLP model only uses the loss regularization parameters λ_{2y1} and λ_{y2} ie, it does not consider the Smoothening curve Regularization for the Torque KPIs.

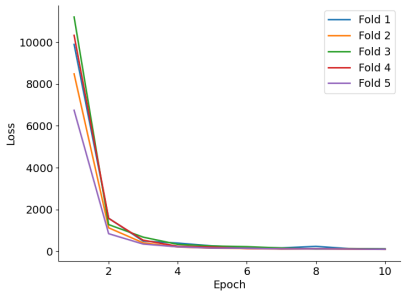


Figure 5.1: Aggregated Training Loss

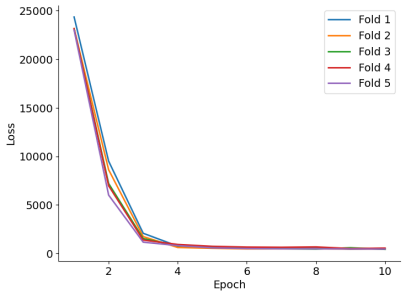


Figure 5.2: Training Loss for Torque KPI

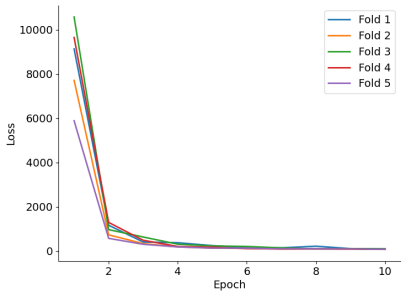


Figure 5.3: Training Loss for Efficiency KPI

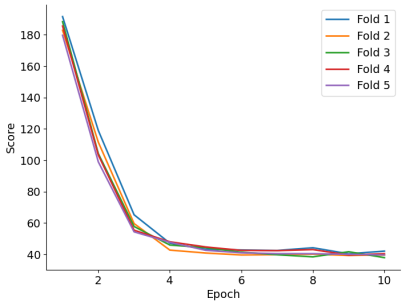


Figure 5.4: Aggregated Training Score

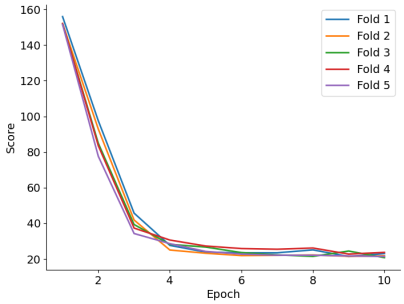


Figure 5.5: Training Score for Torque KPI

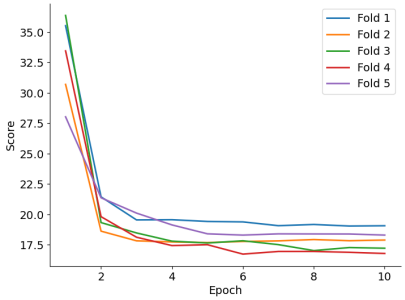


Figure 5.6: Training Score for Efficiency KPI

From the training plots we see that the model has converged after having run for 10 epochs with a total run time of 3 MNIUTES:::TABLE CREATION MINUTES:::
Even though the weightage assigned to the Efficiency KPI is significantly more in addition to loss regularization parameter, it seems to struggle to learn beyond a threshold and overfit by the 5th epoch.
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.
The Validations plots tells us the same tale although the Efficiency KPI stops to learn after a certain point.

[1] Weights & Biases

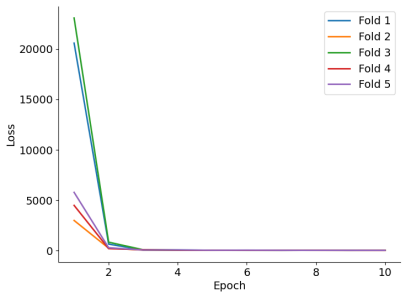


Figure 5.7: Aggregated Validation Loss

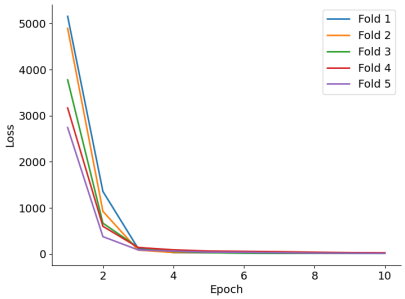


Figure 5.8: Validation Loss for Torque KPI

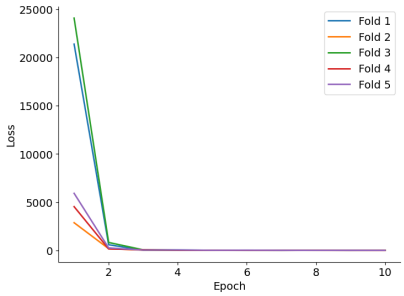


Figure 5.9: Validation Loss for Efficiency KPI

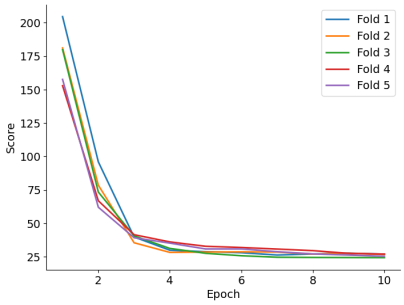


Figure 5.10: Aggregated Validation Score

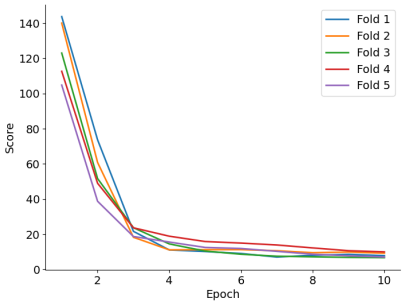


Figure 5.11: Validation Score for Torque KPI

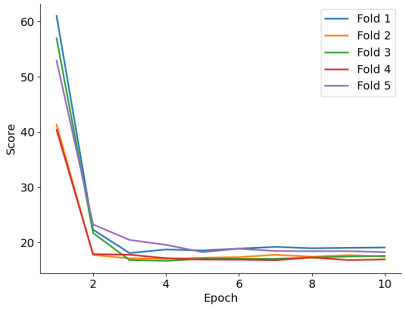


Figure 5.12: Validation Score for Efficiency KPI

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.
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.

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

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

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 difference 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.

5.2 Results with MLP

5.2.1 Torque KPI Results with MLP

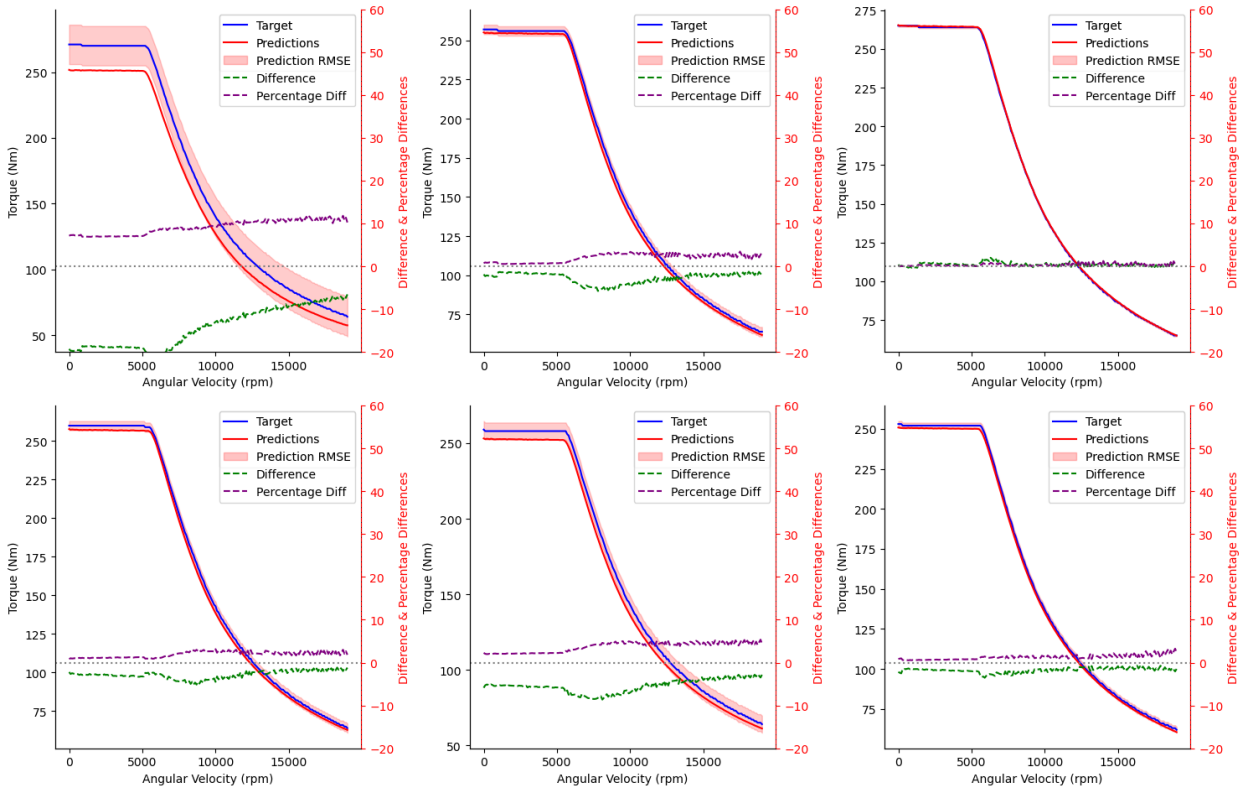


Figure 5.13: MLP Training Results for Torque KPI

Figure 5.13 depicts the difference and percentage difference on a twin scale to give a rough overview of the prediction deviations from the targets. Whereas 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 5.14 shows us the Average RMSE and element wise RMSE for the test dataset performance with the MLP.
Our inferences are overall the predictions closely resemble the trajectory of the target values allthough 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.

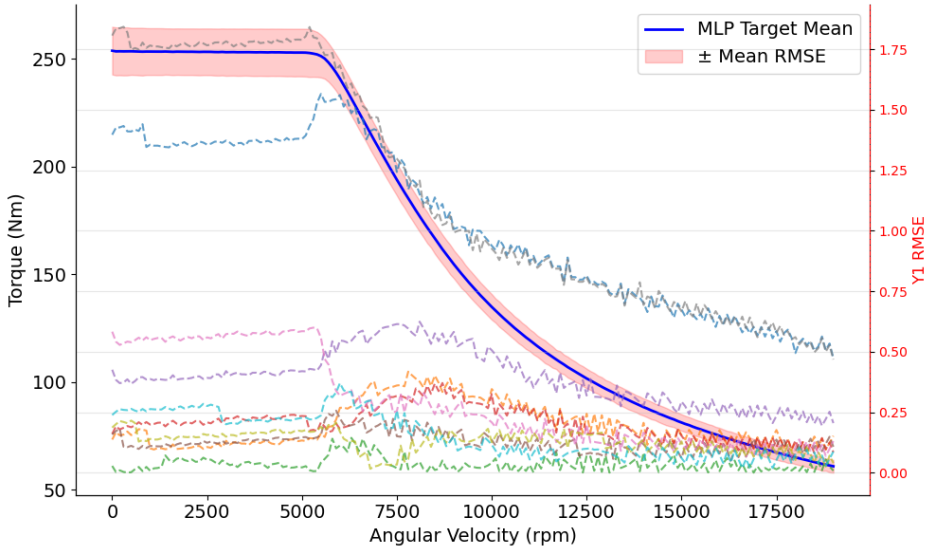
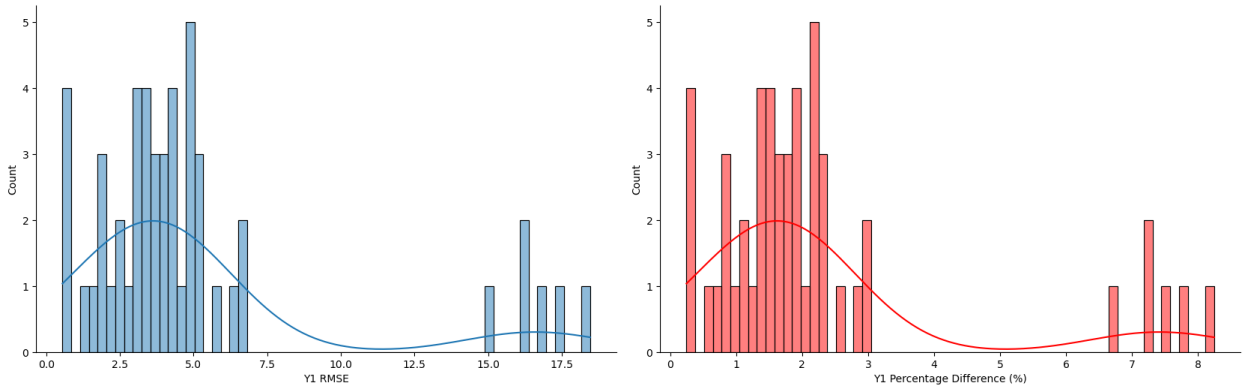


Figure 5.14: MLP RMSE Evaluation for Torque KPI

Figure 5.15 shows the score statistics of the model performance of Torque KPI over the entire test dataset. The Y1 RMSE from Equation 4.12 is calculated for each sample and shown as a histogram in Figure 5.15a. The Y1 Percentage Difference from Equation 5.1 is calculated for each sample and shown as a histogram in Figure 5.15b.



(a) Y1 RMSE

(b) Y1 Percentage Difference

Figure 5.15: 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.

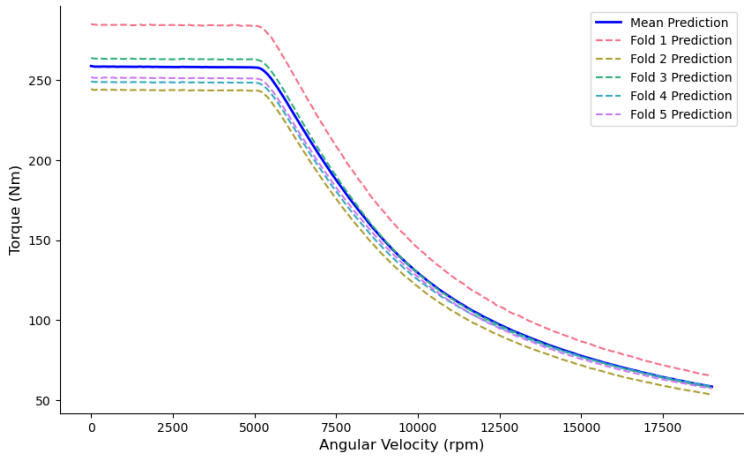


Figure 5.16: MLP Training Deviation across Folds for Torque KPI

Figure 5.16 illustrates how each fold's perdiction 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:

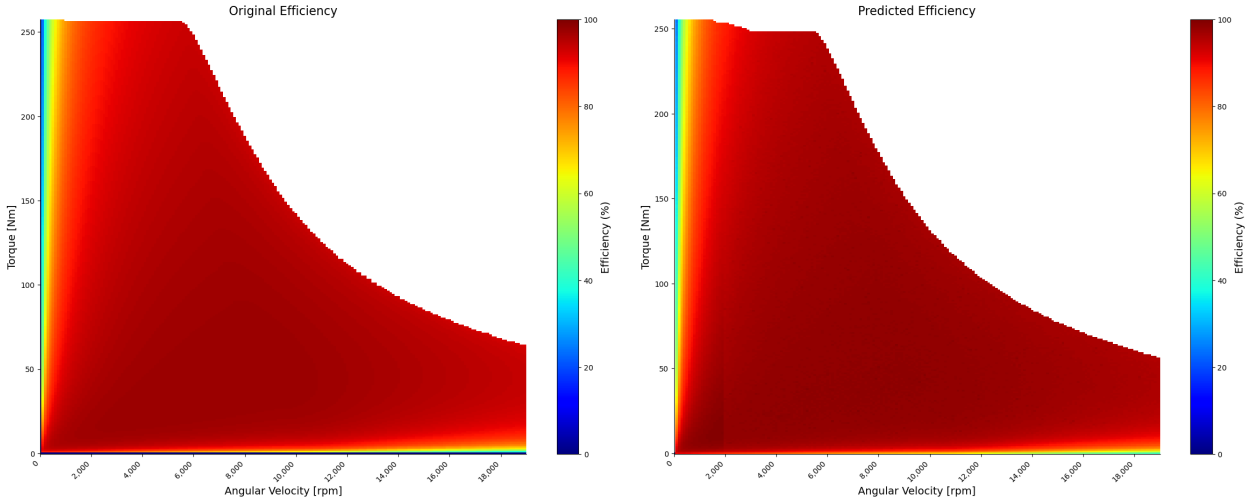


Figure 5.17: 1st MLP Training Results for Efficiency KPI

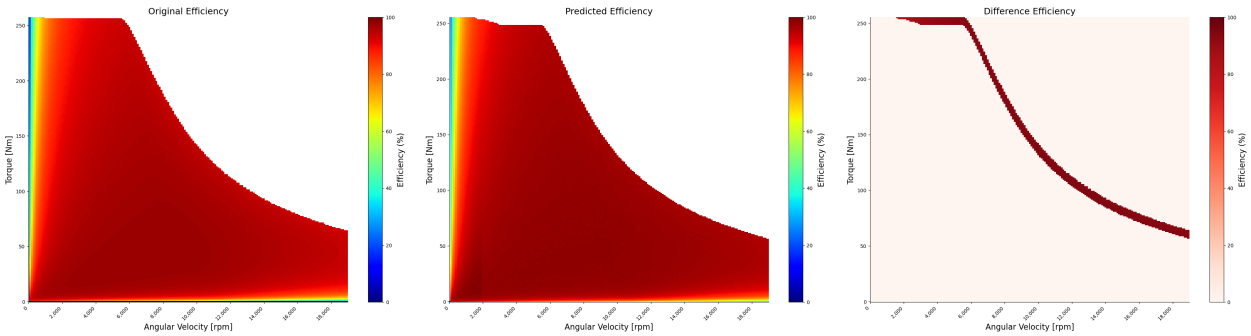


Figure 5.18: 1st Eval MLP Training Results for Efficiency KPI

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.

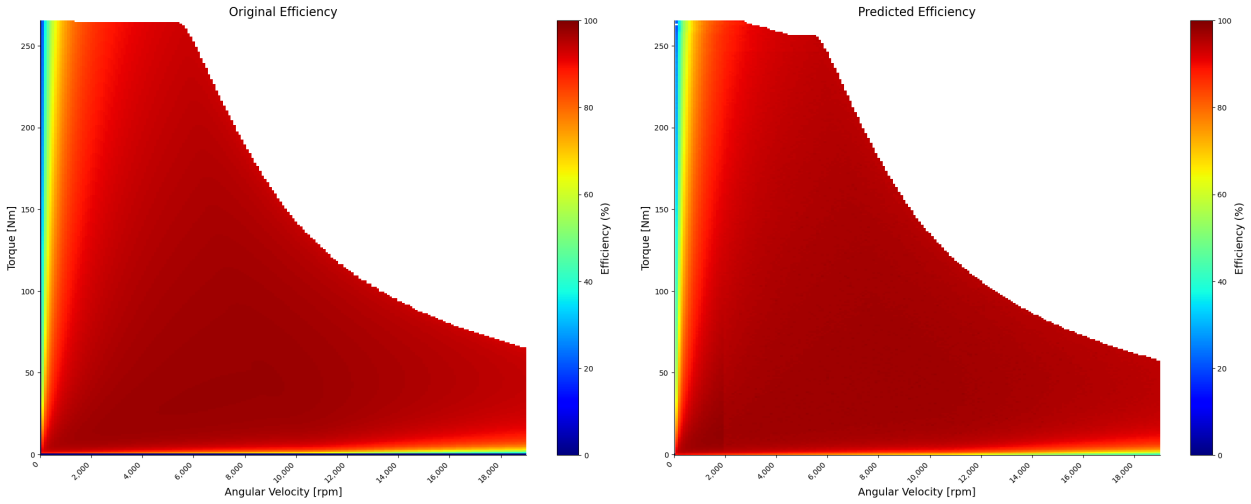


Figure 5.19: 2nd MLP Training Results for Efficiency KPI

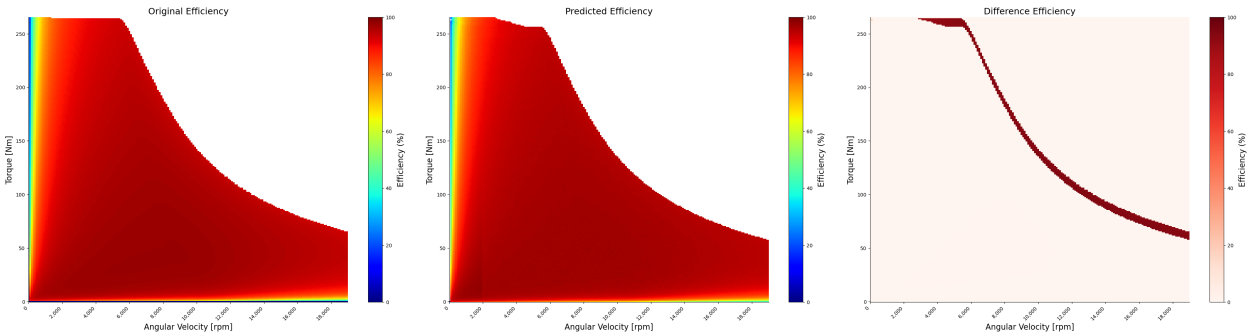


Figure 5.20: 2nd Eval MLP Training Results for Efficiency KPI

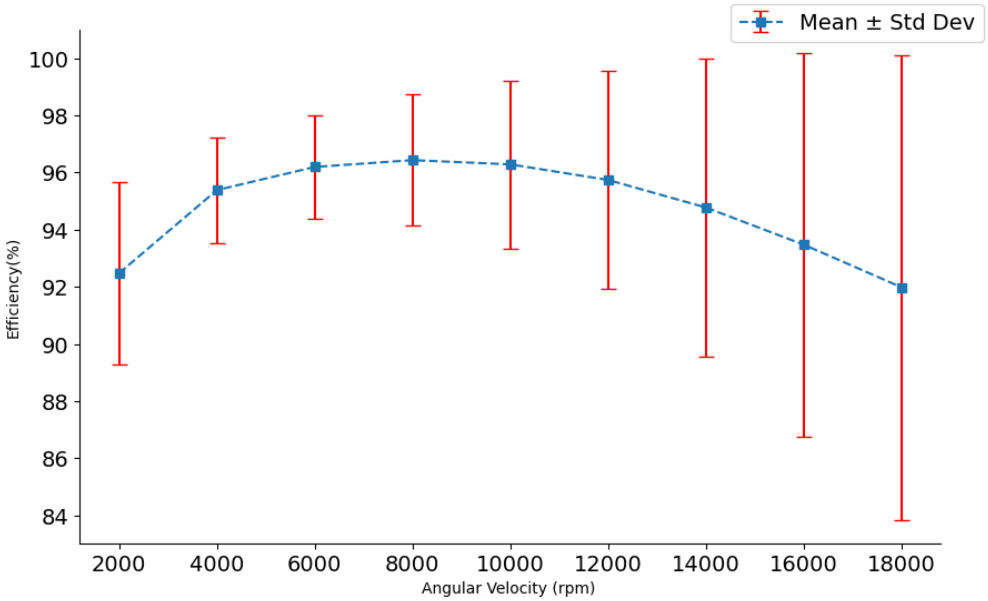


Figure 5.21: MLP Standard Deviation of Efficiency KPI Positive Grid across Speed Intervals

Figure 5.21 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 5.22. 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 ennvelope dimensionality having already padded the targets to be of the maximum shape referred in Section 3.1.3. The reason why it deteroirates 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 1.3.

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 at 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.

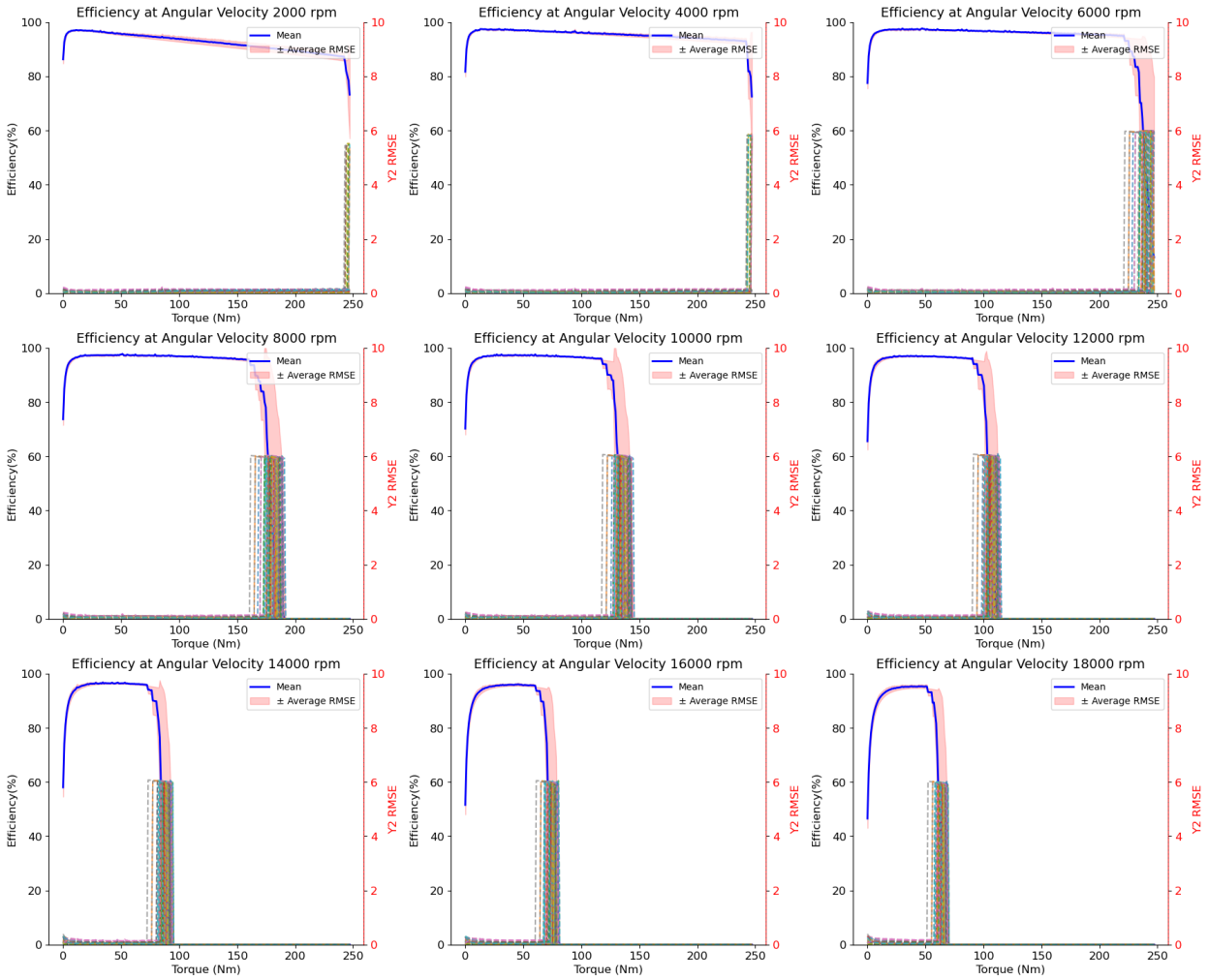
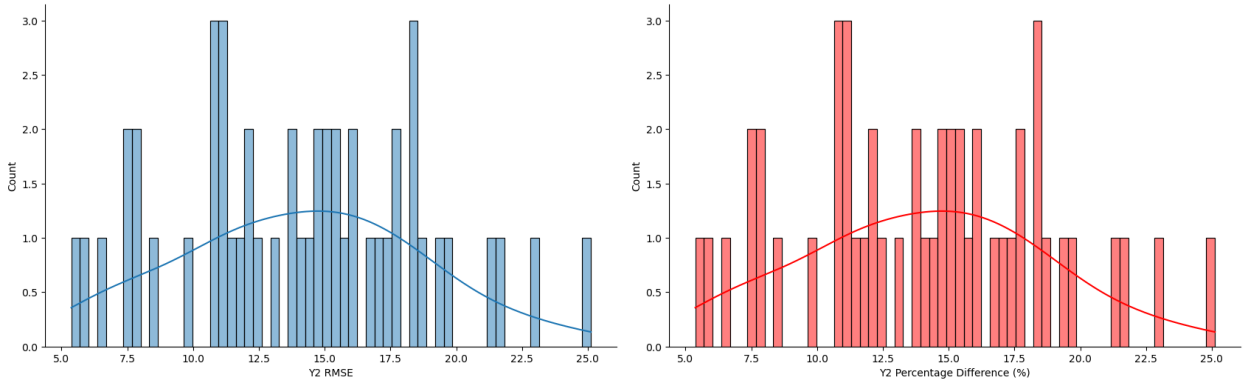


Figure 5.22: Eval MLP Efficiency RMSE KPI

Figure 5.23 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 5.23a. The Y2 Percentage Difference from Equation 5.2 is calculated for each sample and shown as a histogram in Figure 5.23b.



(a) Y2 RMSE

(b) Y2 Percentage Difference

Figure 5.23: Y2 Evaluation Statistics of MLP

We have improved the scores of y2 because we have had cases where the curve is not contained witin the 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.

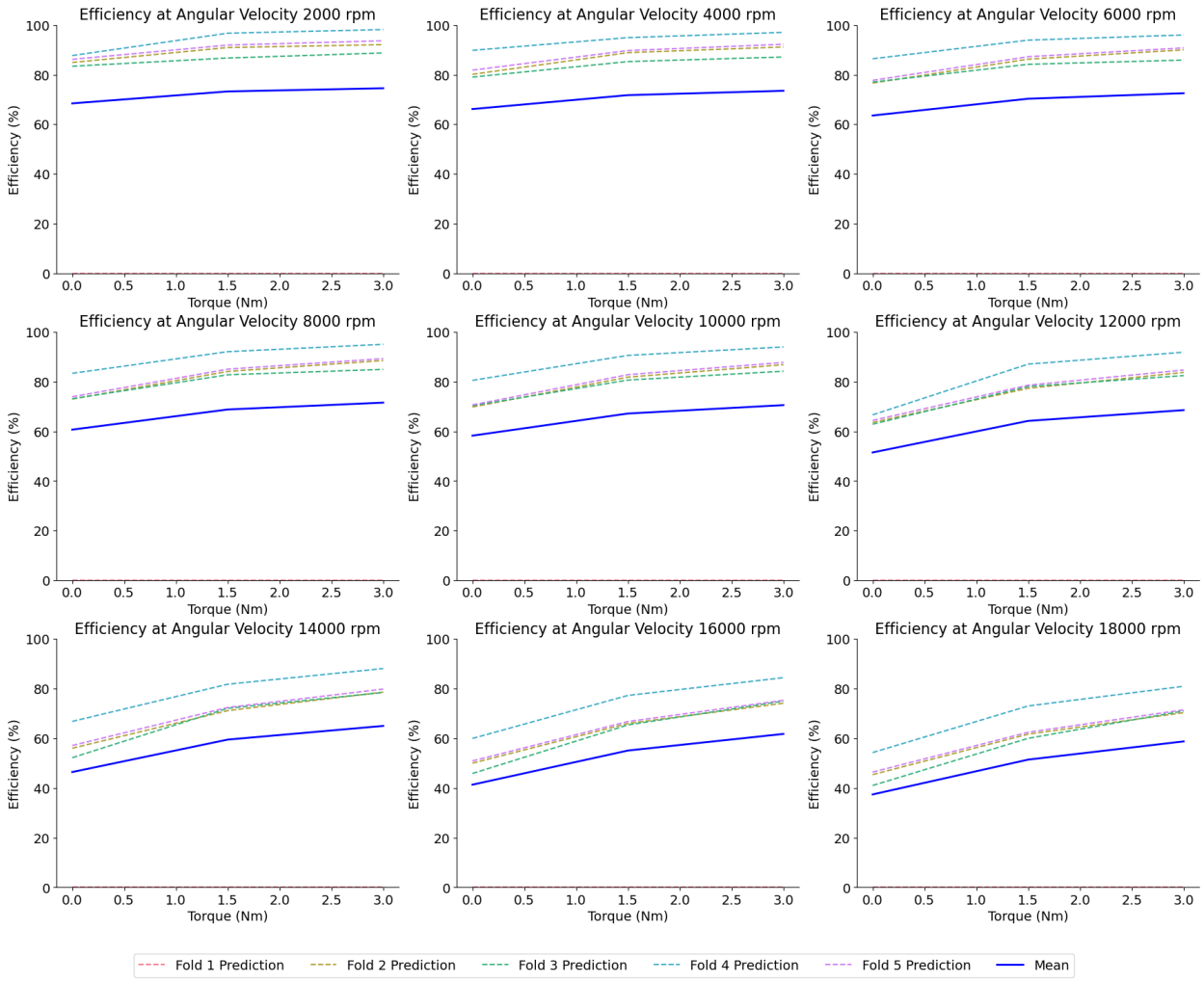


Figure 5.24: MLP Training Deviation across Folds for Efficiency KPI

Figure 5.24 illustrates how each fold's perdiction 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.

5.3 Results with Baseline

From our observations on how the predictions closely resembled that 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 :

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

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

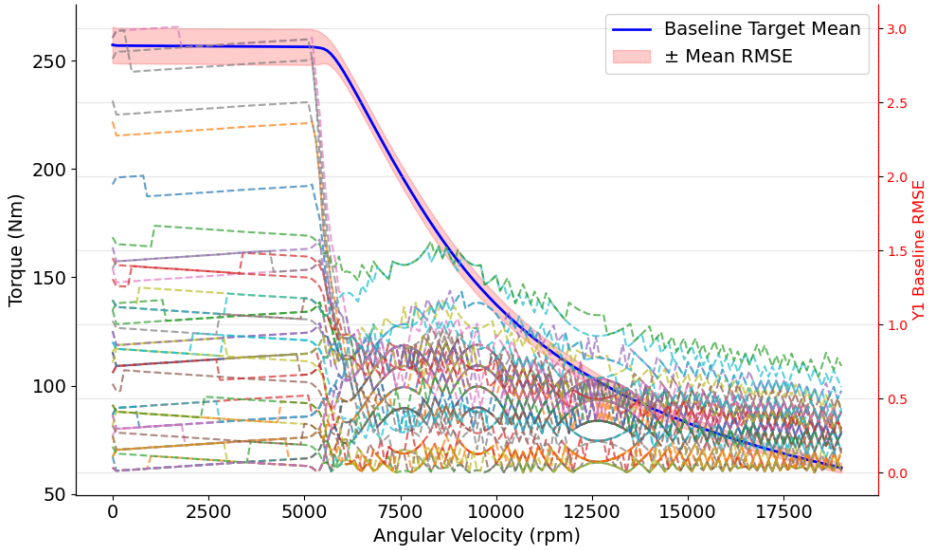


Figure 5.25: Baseline RMSE Evaluation for 2D KPI(Torque)

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 :

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

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 5.26 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.

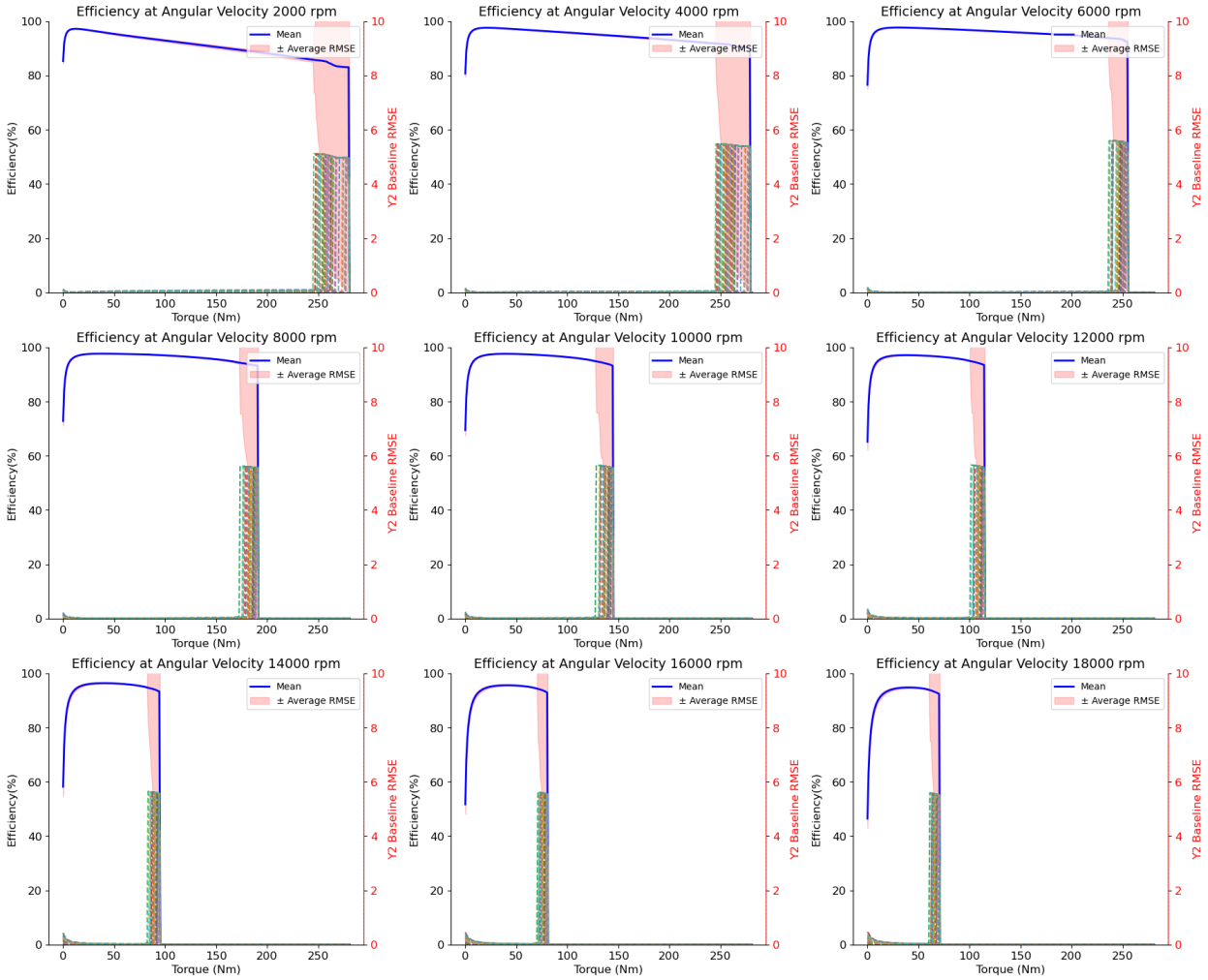


Figure 5.26: Eval Baseline Efficiency RMSE KPI

5.4 Results with Smoothening Loss Regularization

5.4.1 Torque KPI Results with Smoothening Loss Regularization

Figure 5.27 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 5.14

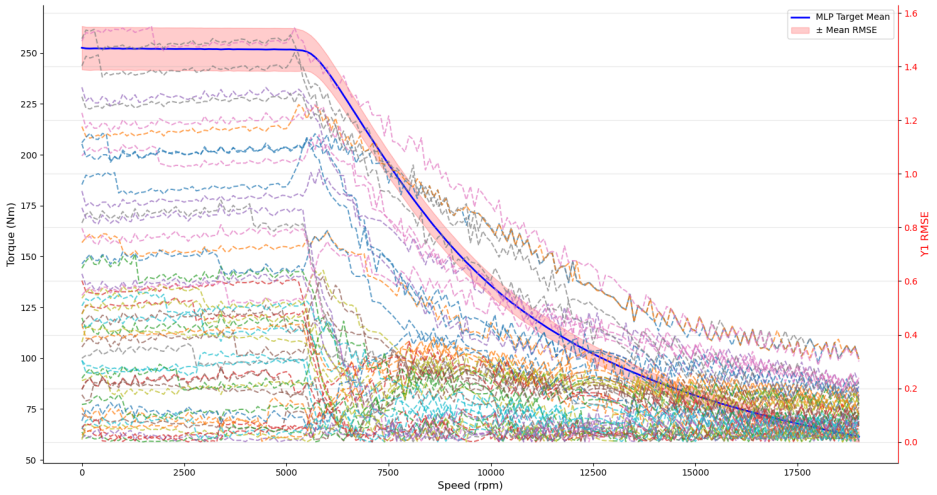


Figure 5.27: Smoothening Torque RMSE Evaluation for 2D KPI(Torque)

5.5 Results with No Loss Regularization

5.5.1 Torque KPI Results with No Loss Regularization

Figure 5.28 shows the Average RMSE and element wise RMSE for the test dataset performance with the MLP Model without any regularizations.

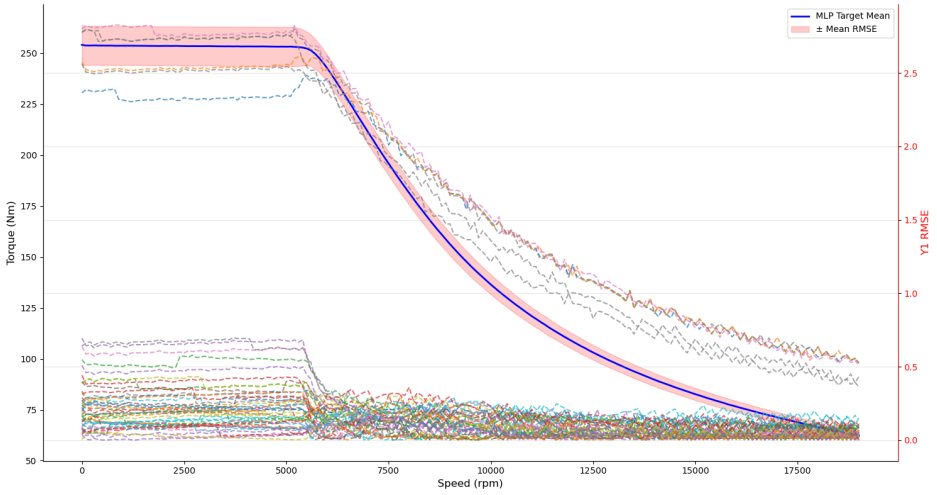


Figure 5.28: No Loss Regularization RMSE Evaluation for 2D KPI(Torque)

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 5.29 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.

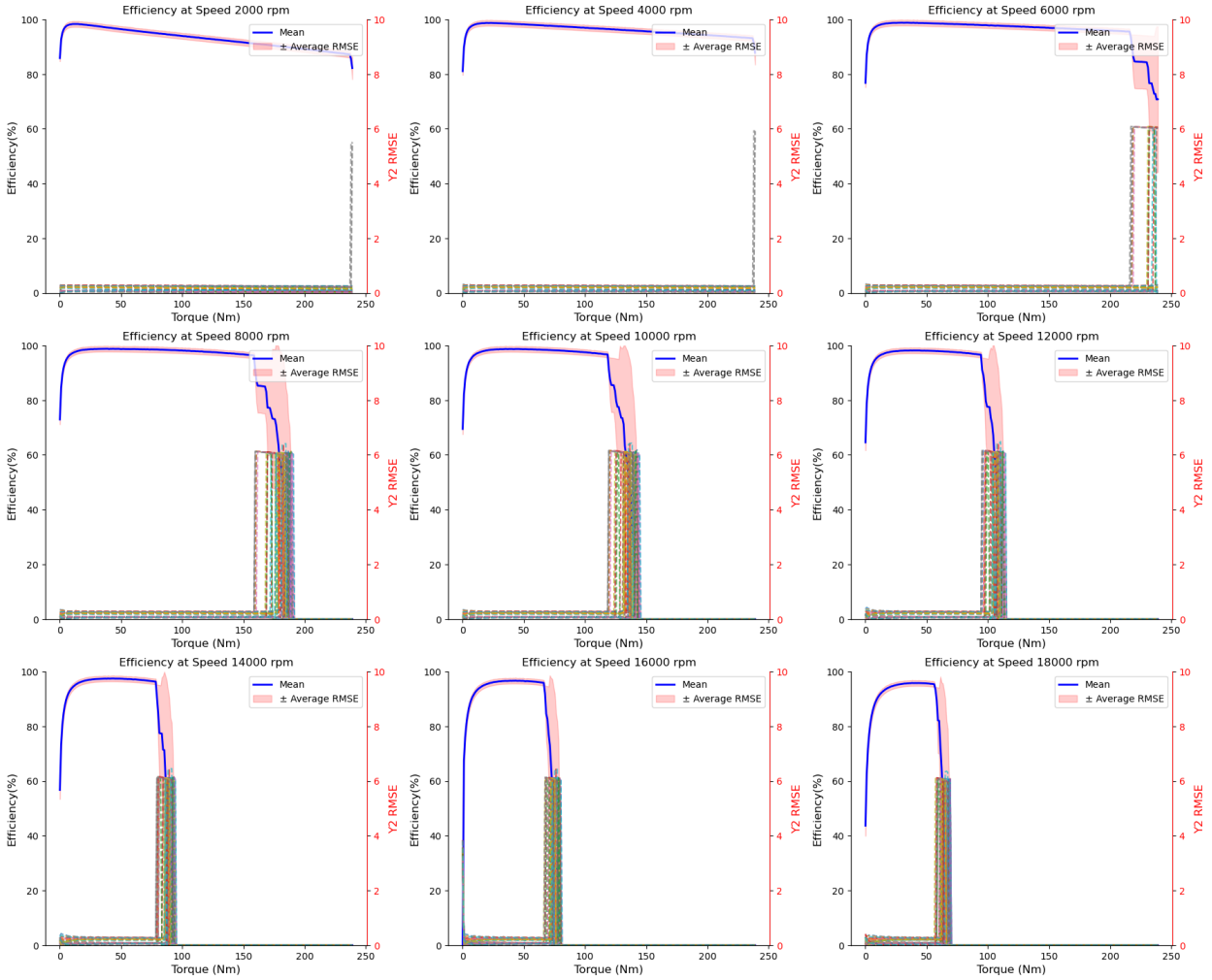


Figure 5.29: Eval No Loss Regularization Efficiency RMSE KPI

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

[a]MLP With Decreasing Y1 Loss Regularization
[b]MLP With Smoothening Y1 Loss Regularization
[c]MLP 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 setings. 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 surogate modelling and relate both approaches how we have demonstrated in the flowchart in Figure 1.4

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

[a]Source : 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 1.4 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`[2].

[2]https://github.com/Lilly-25/Masters-Thesis

Chapter 6

Graph Modelling

We intended to model our problem as a Graph and solve using a GNN. We presume this will be a more clever way of representing our usecase in comparision to tabular data. The idea was to make use of the Graph dynamics in our usecase and aggregate features that are semantically similar.

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. They also take into account their respective neighbourhoods.

GNNs aim to learn a representation vector for each node using the MP algorithm. MP is based on the graph structure and initial node features. For each node in the graph over each successive hop of its neighbourhood until it has covered the entire graph, the model updates the learned node representation recursively aggregating its neighbour node features.
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 be used for downstream tasks.

GNN predictions can be categorized broadly into : [18]

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

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

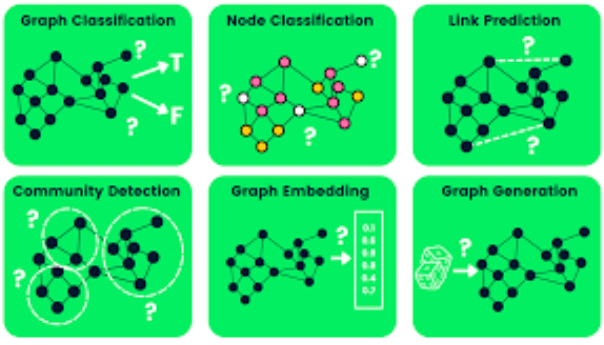


Figure 6.1: GNN Predictions Source :WWW

Graphs in general can have the following taxonomy: [18]

1. Directed Graph - A graph in which the edges have a direction.
2. Undirected Graph - A graph in which for each edge there will be its respective reverse edge in the opposite direction and the Adjacency matrix is symmetric.

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. [05]

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].

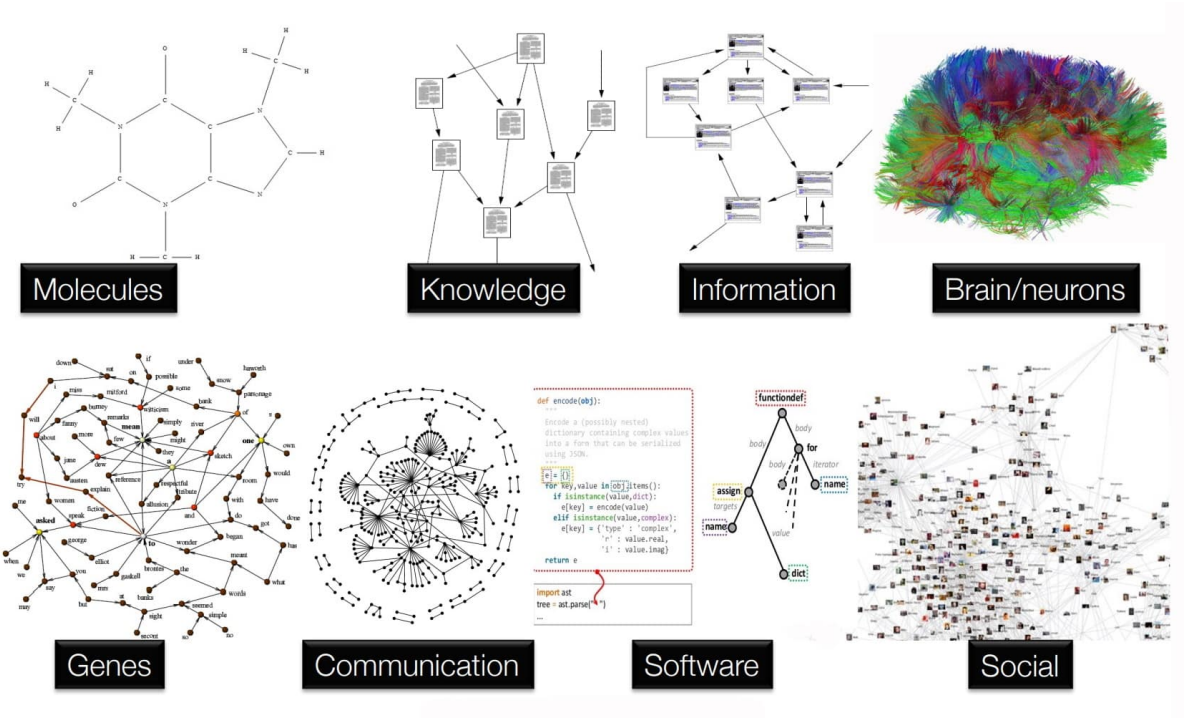


Figure 6.2: GNN Applications Source :WWW

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.

6.1 Introduction

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 $G = (V, E)$ 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.
The neighborhood of a node v is defined as $N(v) = \{u \in V \mid (v, u) \in E\}$.

Adjacency Matrix.
The adjacency matrix A is a $n \times n$ matrix with $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$.

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

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

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

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

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

Message Passing.
For Graph signals, $V = [v_1, v_2, \ldots, v_n], v_i \in \mathbb{R}^F$, the updated Graph Signals is $V' = RV\theta$ where RV is the weighted sum of graph signals neighbourhood. RV also ensures only nodes with edges is considered.
In contrast to CNNs fixed shape kernel which considers pixels locality we have an increasing shaped kernel for GNNs. This is clear in the Figure 6.3.
It is also called a Polynomial Graph Filter or Radial Filter and can be expressed as

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

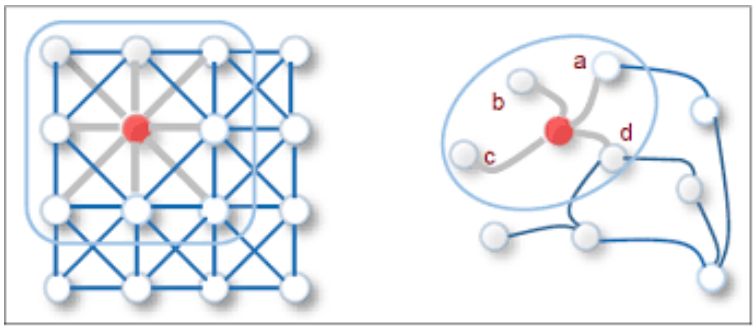


Figure 6.3: GNN Kernel Filter Source :WWW

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

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

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

A hallmark of GNNs is that they can efficiently learn the architecture parameters from the training data [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 pre activation residual connections in the network. CITE

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 build to capture the structural information within a graph.
   These models are effective when the node and edge heterogeneity is negligible [01].

2. **Heterogeneous GNN**
   Heterogeneous GNN are designed for graphs with different type of nodes and edges. Different types of nodes and edges also imply difference in features and dimensionality. Therefore, each type of node and edge reveal unique semantic information [24].
   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.

6.2 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 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 $G = (V, E, X, R, T_v, T_e, \phi, \psi)$

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 specifications 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., $(nodetypeofs, edgetype, nodetypeoft)$ used commonly in Knowledge Graphs.

Metapath
Given a path $P_1 \xrightarrow{R_1} P_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} P_{l+1}$, if the path describes the composite relationship $R_1 \circ R_2 \circ \cdots \circ R_l$ between nodes of type T_{v1} and T_{vl+1}, then the path is called a metapath.
The metapath is also the sequence of many metarelations.

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 [21] on modelling heterogeneous graphs usually split 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.
[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 [03] as 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, [23] has used meta relations instead and a distinct edge based

matrix for each edge type.
Works by [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 deep as the meta paths [26].

[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.

[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.
[26] Most of the existing works also do not consider the edge features in their model. However [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 of different node type-edge type combinations. The paper also brings to light that GraphSAGE does not incorporate the edge weights.
However, the Message Passing Neural Network (MPNN) framework applies a learned message-passing function that utilizes edge features.

Additionally [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.

[27] introduces modelling that makes it capable of learning features of molecular graphs directly and is invariant to graph isomorphism. 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.

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-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].

[1]Python library for graphs and networks

1. Node Types
   $T_v = \{v, vm, r, s, sw\}$
   The node type mapping functions of each node type in the graph are as follows :

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

Table 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:

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

Table 6.2: GNN Edge Types

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

Figure 6.4 shows the heterogeneous graph we constructed

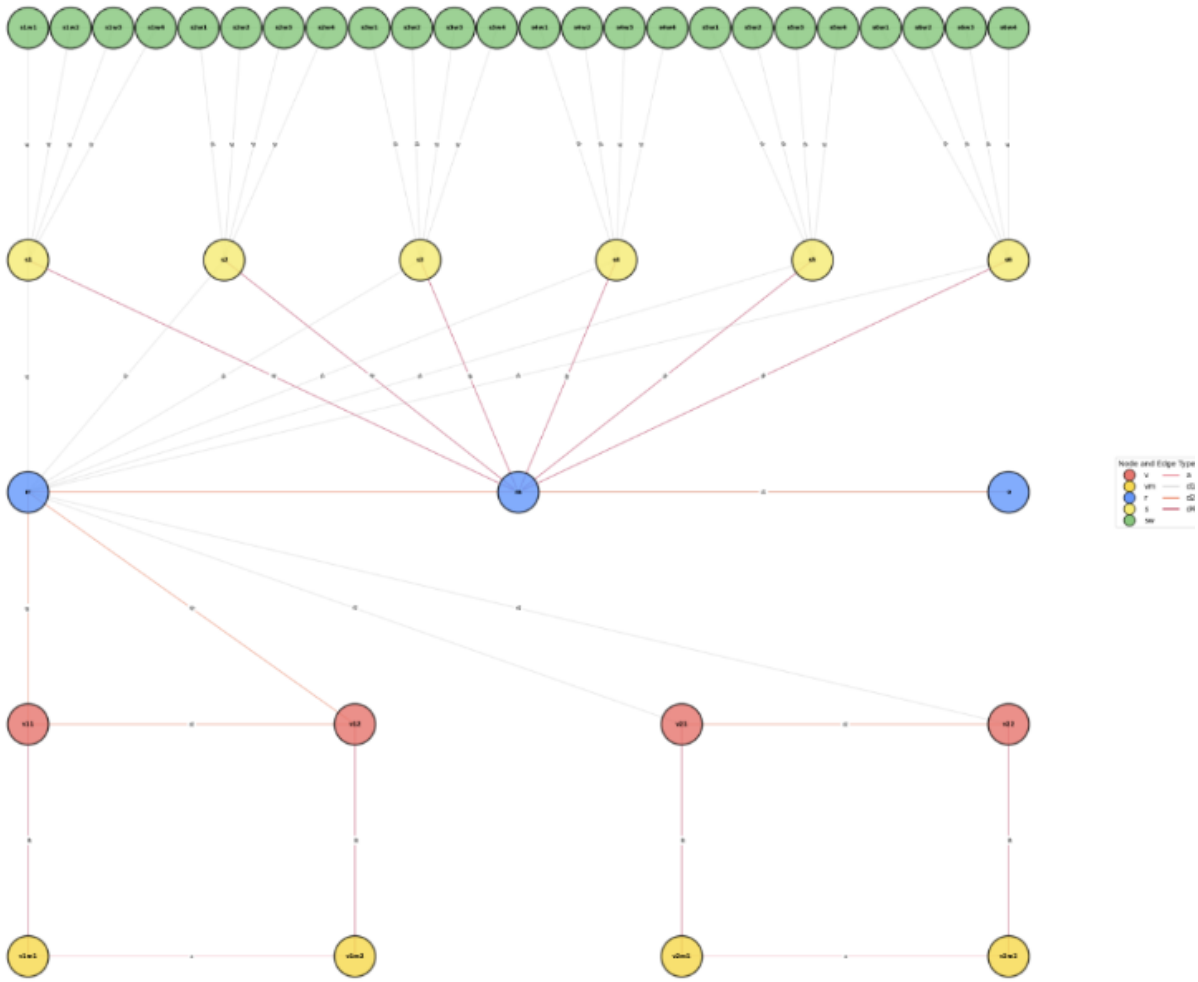


Figure 6.4: HetGraph

The count of certain parameters within the motor such as stator poles with its corresponding stator windings and rotor magnets is made more comprehendable to the model having new nodes and edges whereas for the MLP architecture this information is represented only as a number in yet another 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.

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 parameteric 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

Table 7.1 shows the summary statistics of the input parameters of the motor.

Parameter	Unit of Dimension	Mean	Standard Deviation	Value Range	Single V Magnet Topology	Double V Magnet Topology	Nabla Magnet Topology
General Parameters							
N	#	4	0	4 –4	✓	✓	✓
simQ	#	6	0	6 – 6	✓	✓	✓
r_a	mm	9×10^{-2}	2.5×10^{-15}	$9 \times 10^{-2} - 9 \times 10^{-2}$	✓	✓	✓
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}	-3×10^{-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	$0 - 1.5 \times 10^{-4}$	×	✓	×

Continued on next page

Table 7.1 – continued from previous page

Parameter	Unit of Dimension	Mean	Standard Deviation	Value Range	Single V Magnet Topology	Double V Magnet Topology	Nabla Magnet Topology
rad_phiv1	rad	-6.9 × 10^{-1}	5.4 × 10^{-2}	-7.8 × 10^{-1} – -4.5 × 10^{-1}	✓	✓	✓
lmsov1	mm	-5.01×10^{-4}	9.4 × 10^{-5}	-5.3 × 10^{-4} – 5 × 10^{-4}	✓	✓	✓
lth1v1	mm	2.8 × 10^{-3}	1.7 × 10^{-4}	2.8 × 10^{-3} – 5.45 × 10^{-3}	✓	✓	✓
lth2v1	mm	2.104 × 10^{-3}	5.9 × 10^{-5}	2.1 × 10^{-3} – 3.2 × 10^{-3}	✓	✓	✓
r1v1	mm	4.07 × 10^{-4}	1.08 × 10^{-4}	4×10^{-4} – 2.2×10^{-3}	✓	✓	✓
r11v1	mm	2.19 × 10^{-4}	4.5 × 10^{-5}	1 × 10^{-4} – 6 × 10^{-4}	✓	✓	✓
r2v1	mm	2.16 × 10^{-4}	1 × 10^{-4}	2×10^{-4} – 1.9×10^{-3}	✓	✓	✓
r3v1	mm	8.99 × 10^{-4}	1.3 × 10^{-5}	7 × 10^{-4} – 9 × 10^{-4}	✓	✓	✓
r4v1	mm	5.01 × 10^{-4}	1.6 × 10^{-5}	5×10^{-4} – 7.5×10^{-4}	✓	✓	✓
rmt1v1	mm	2.5 × 10^{-4}	8.8 × 10^{-18}	2.5 × 10^{-4} – 2.5 × 10^{-4}	✓	✓	✓
rmt4v1	mm	2.5 × 10^{-4}	8.8 × 10^{-18}	2.5 × 10^{-4} – 2.5 × 10^{-4}	✓	✓	✓
rlt1v1	mm	1.17 × 10^{-4}	5.6 × 10^{-5}	5×10^{-5} – 2.5×10^{-4}	✓	✓	✓
rlt4v1	mm	2.5 × 10^{-4}	8.8 × 10^{-18}	2.5 × 10^{-4} –2.5 × 10^{-4}	✓	✓	✓
hav1	mm	2.9 × 10^{-3}	1.36 × 10^{-4}	2.9×10^{-3}– 5×10^{-3}	✓	✓	✓
mbv1	mm	7.6 × 10^{-3}	5.8 × 10^{-4}	7.5 × 10^{-3} – 1.8 × 10^{-2}	✓	✓	✓
mhv1	mm	2.8 × 10^{-3}	1.4 × 10^{-4}	2.7×10^{-3} – 5×10^{-3}	✓	✓	✓
rmagv1	mm	5 × 10^{-4}	1.7 × 10^{-17}	5 × 10^{-4} – 5 × 10^{-4}	✓	✓	✓
dsmv1	mm	1.07 × 10^{-3}	1.41 × 10^{-4}	8×10^{-4} – 2.8×10^{-3}	✓	✓	✓
dsmuv1	mm	1.07 × 10^{-3}	1.47 × 10^{-4}	8 × 10^{-4} – 2.92 × 10^{-3}	✓	✓	✓
amtrv1	mm	5.5 × 10^{-3}	6.34 × 10^{-4}	5.5 × 10^{-3} – 1.9 × 10^{-2}	✓	✓	✓
dsrv1	mm	7.5 × 10^{-4}	3.2 × 10^{-5}	7.5 × 10^{-4} – 1.25 × 10^{-3}	✓	✓	✓
lmav1	mm	9.2 × 10^{-5}	2.6 × 10^{-5}	1×10^{-5} – 1.1×10^{-4}	✓	✓	✓
lmiv1	mm	1 × 10^{-4}	6.35 × 10^{-7}	1×10^{-4} – 1.1×10^{-4}	✓	✓	✓
lmov1	mm	5.5 × 10^{-5}	1.5 × 10^{-5}	5 × 10^{-5} – 1 × 10^{-4}	✓	✓	✓
lmuv1	mm	1.45 × 10^{-4}	1.5 × 10^{-5}	1×10^{-4} – 1.5×10^{-4}	✓	✓	✓
rad_phi3b1	rad	- 2.5 × 10^{-3}	5.6001 × 10^{-2}	-1.25 – 0	×	×	✓
rad_phi4b1	rad	-5.3 × 10^{-4}	1.1775 × 10^{-2}	-0.26 – 0	×	×	✓
lmsob1	mm	2 × 10^{-6}	3.4 × 10^{-5}	0 – 7.5 × 10^{-4}	×	×	✓
lthb1	mm	6 × 10^{-6}	1.28 × 10^{-4}	0 – 2.9 × 10^{-3}	×	×	✓
r2b1	mm	2 × 10^{-6}	4.5 × 10^{-5}	0 – 1 × 10^{-3}	×	×	✓
r3b1	mm	2 × 10^{-6}	3.4 × 10^{-5}	0 – 1 × 10^{-3}	×	×	✓
r4b1	mm	5.06 × 10^{-7}	1.1 × 10^{-5}	0 – 2.5 × 10^{-4}	×	×	✓
r5b1	mm	5.06 × 10^{-7}	1.1 × 10^{-5}	0 – 2.5 × 10^{-4}	×	×	✓

Continued on next page

Table 7.1 – continued from previous page

Parameter	Unit of Dimension	Mean	Standard Deviation	Value Range	Single V Magnet Topology	Double V Magnet Topology	Nabla Magnet Topology
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}	$4.45 \times 10^{-3} - 4.6 \times 10^{-3}$	✓	✓	✓
b_s	mm	1×10^{-3}	2.5×10^{-5}	$1 \times 10^{-3} - 1.4 \times 10^{-3}$	✓	✓	✓
h_n	mm	1.1×10^{-2}	8.06×10^{-4}	$9.2 \times 10^{-3} - 1.39 \times 10^{-2}$	✓	✓	✓
h_s	mm	1×10^{-3}	3.5×10^{-17}	$1 \times 10^{-3} - 1 \times 10^{-3}$	✓	✓	✓
r_sn	mm	2.5×10^{-4}	8.8×10^{-18}	$2.5 \times 10^{-4} - 2.5 \times 10^{-4}$	✓	✓	✓
r_zk	mm	5.01×10^{-4}	1.9×10^{-5}	$5 \times 10^{-4} - 8 \times 10^{-4}$	✓	✓	✓
r_ng	mm	5.01×10^{-4}	1.9×10^{-5}	$5 \times 10^{-4} - 8 \times 10^{-4}$	✓	✓	✓
h_zk	mm	1×10^{-3}	3.5×10^{-17}	$1 \times 10^{-3} - 1 \times 10^{-3}$	✓	✓	✓
bhp	mm	3.7×10^{-3}	8.1×10^{-5}	$3.5 \times 10^{-3} - 3.8 \times 10^{-3}$	✓	✓	✓
hhp	mm	2.38×10^{-3}	1.9×10^{-4}	$1.9 \times 10^{-3} - 2.8 \times 10^{-3}$	✓	✓	✓
rhp	mm	6.36×10^{-4}	4.5×10^{-5}	$5 \times 10^{-4} - 8 \times 10^{-4}$	✓	✓	✓
dhphp	mm	2.6×10^{-4}	1.4×10^{-5}	$2.6 \times 10^{-4} - 4.8 \times 10^{-4}$	✓	✓	✓
dhpng	mm	4.06×10^{-4}	3×10^{-6}	$4.06 \times 10^{-4} - 4.5 \times 10^{-4}$	✓	✓	✓

Table 7.1: 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.

List of Figures

List of Tables

Bibliography

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

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

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

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

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

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

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

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

[09] 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.

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