Technical University of Applied Sciences Würzburg-Schweinfurt (THWS)

Faculty of Computer Science and Business Information Systems

Master Thesis

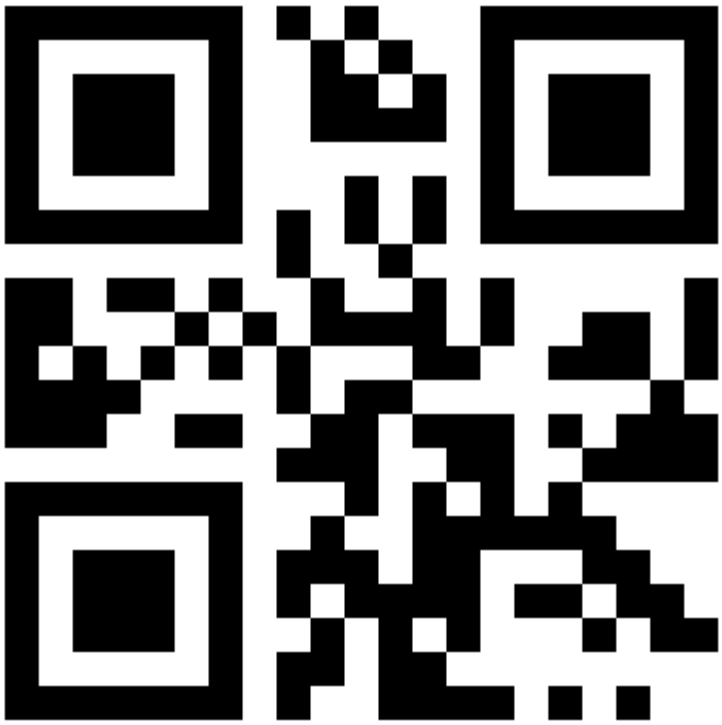
Electric Motor Modelling via Graph Neural Networks

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

To be 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 by transforming its geometric, physical and simulation parameters into a graph representation.
The KPIs to be predicted are plots on Efficiency grid(3D) and Torque curve(2D).
We aim to first parameterize the EM design such that it is feasible to convert into a graph representation.
Next, we would create a Graph with relevant attributes and design a Graph Neural Network(GNN)with the graph as input and the plots in the format of vectors as target values.
Additionally we may also need to customize the loss function in a way that would smoothen out the plot curves of the prediction values.
Then, we would evaluate the predictions with the test target values by experimenting with various hyper-parameter tuning settings and as a baseline with an Multi Layer Perceptron(MLP) model of the parameters in tabular form.
Finally we will enable the KPI's plot visualisation in a manner presentable to the client Valeo(Automaker Company).
Not necessary remove i suppose.... The aim of the Master Thesis is to train a neural network to learn the parameters of Electric Motors and thus be able to predict its KPIs. The KPIs are 2D and 3D plots on Torque(Mgrenz) curve(Mgrenz) and Efficiency grid(ETA). Other KPIs can be calculated from these two KPIs. For instance the Vibration Costs are inversely proportional to the Efficieny values predicted.

Acknowledgement

I would like to thank my supervisor Prof. Dr. Magda Gregorova for her guidance and support throughout the course of this thesis and Valeo for providing the data. Special thanks to Mr Daniel and Leo for sharing valuable insights of the data from an electromechanincal standpoint.

Contents

Abbreviations

GNN	Graph Neural Network
MLP	Multi Linear Perceptron
KPI	Key Performance Indicator
EM	Electric Motor
FEM	Finite Element Method
CNN	Convolution Neural Network
2D	2 Dimension
3D	3 Dimension
Wandb	Weights & Biases
MSE	Mean Squared Error

Introduction

In the design of electric motors, vast amounts of data are generated to determine which design of an Electric Motor (EM) fits best to Key Performance Indicator (KPI)s.
KPIs of an Electric Motor are 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 Method (FEM) 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 as below:

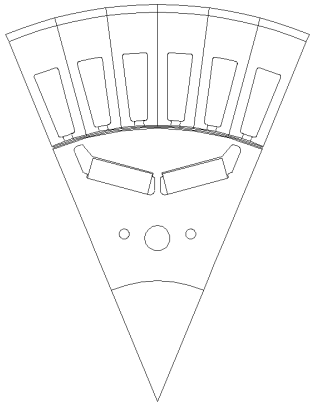


Figure 1: V1 Magnet (Source:Valeo)

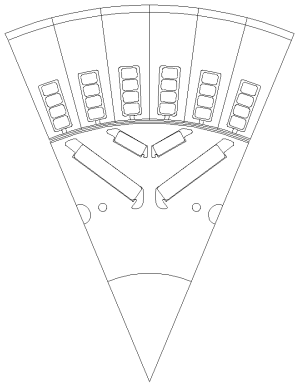


Figure 2: V2 Magnet (Source:Valeo)

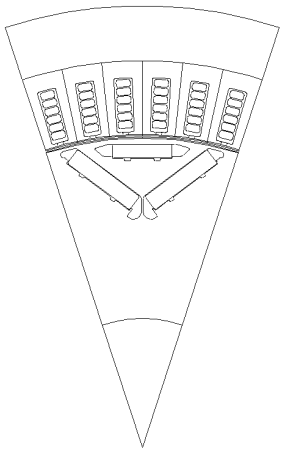


Figure 3: Nabla Magnet (Source:Valeo)

This master thesis explores a way to do surrogate modelling of the current process as is highlighted in Figure 4 by making use of GNN or MLP for the modelling of electrical engine designs described parameterically.

Problem Statement/ Research Qs,/ Objective

We formulate the problem as follows:
blah blah

Tasks

The step-wise objectives are:

- Literature review.

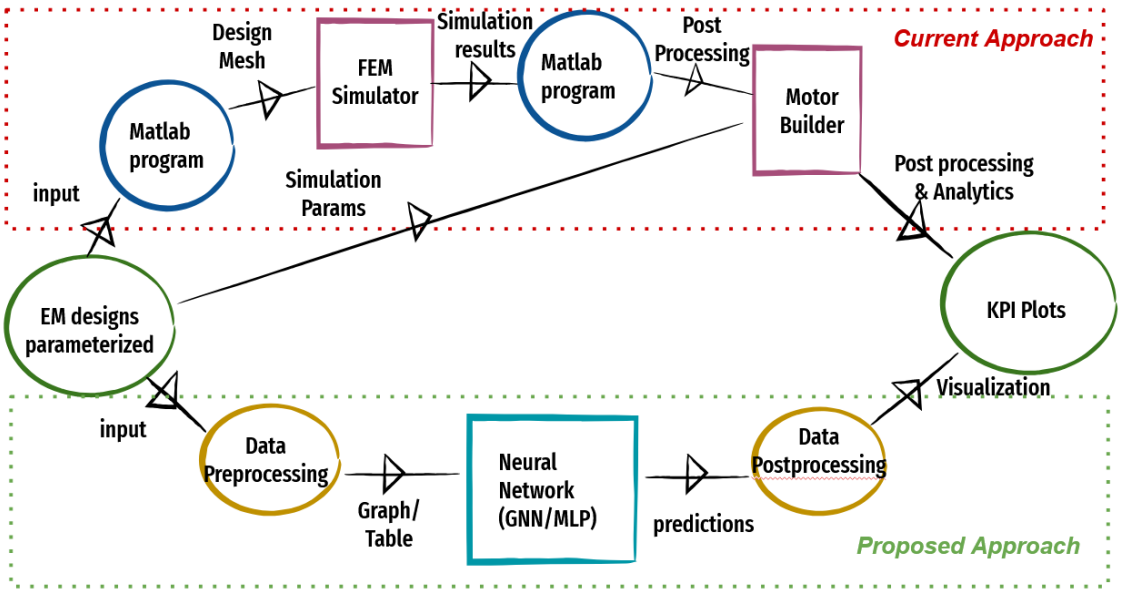


Figure 4: EM Design Flowchart

- Data gathering.
- Converting the parameters into a graph representation with node-edge-global context and its attributes.
- Data preprocessing such that all data is aligned into a numerical format.
- Preparing the train-test-validation datasets.
- Proposing and implementing different approaches on how to train the graph neural network for 1 KPI.
- Extend the architecture to cater to multiple KPIs.
- Evaluating the predictions with the already simulated KPIs.
- Visualize the plots.

Thesis Structure

The thesis is structured to follow sections namely Literature Review, Dataset, Modelling, Experiments and Results, Conclusion, Appendix 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 methodologies used to tackle the problem.
The Experiments and Results chapter gives an outlook on the outcomes of our work in addition to other findings we unearth.
Conclusion chapter summarizes the thesis briefly and would also give a small glimpse into areas of improvement.
Finally the Bibliography section lists out the articles cited for this thesis.

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 and reproducing images is not known to apt given the infamous known fact that AI generated images are faulty.
Instead 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.
Existing literature also covers works on modelling this work as tabular data using MLPs. 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 Graph Neural Networks 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.

Dataset

Valeo an automotive company has supplied the dataset consisting of close to 1500 Double V Electric Motor parameters. There are close to 196 parameters which comprises of the geometric, physical and simulation properties of the motor.

The geometry of a whole Double V motor is as below

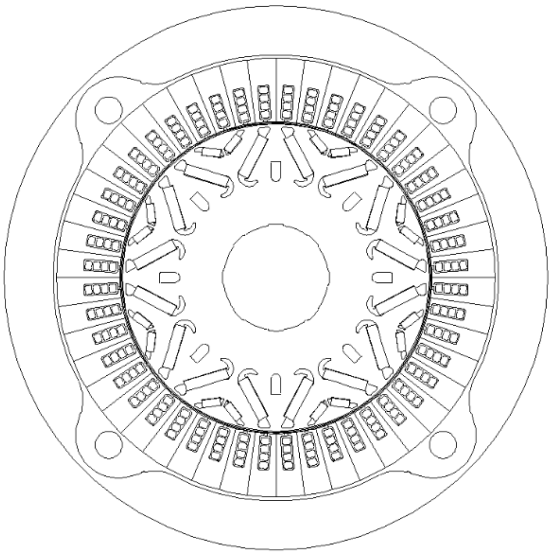


Figure 5: Complete EM Geometry(Source:Valeo)

Below is the geometry of 1/8 cross-section of the same motor.

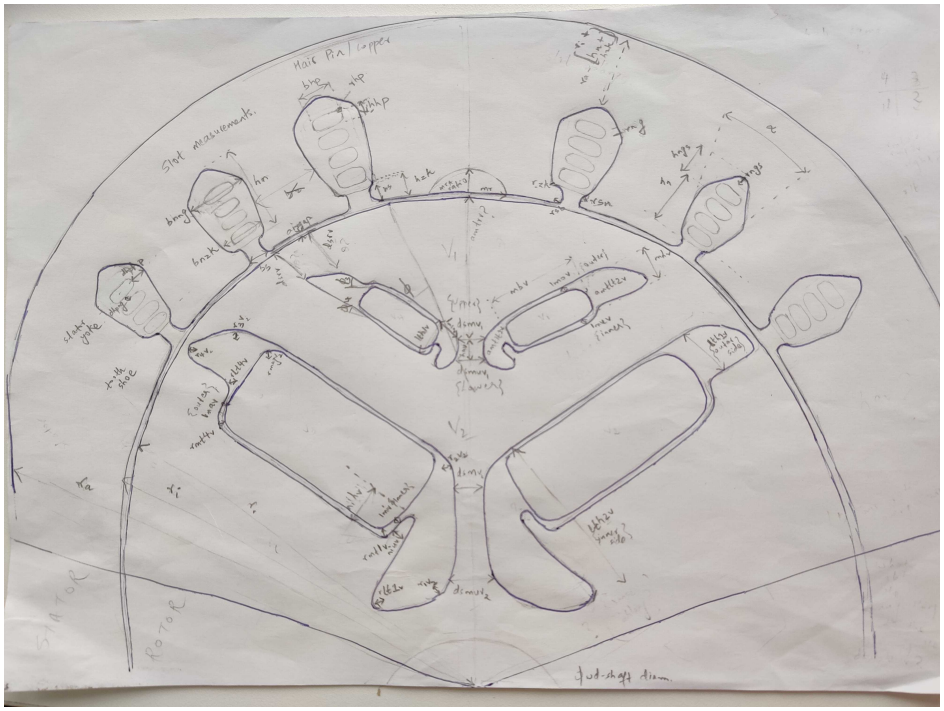


Figure 6: 1/8 Motor Crossection

Valeo has shared approximately 1500 .xslx files for each motor variant. Each of the excel files contain multiple sheets. The sheets of interest to us are as below :

- **Motor Parameters: input_data**
  - Contains the input parameters for the motor model.
  - Includes geometric, physical, and simulation properties.
  - Unit dimensions if applicable are generally mm or degrees
- **Speed Grid: NN**
  - Contains the speed grid ranging from 0 to 19000 rpm in steps of 100.
  - Used for plotting the ETA grid and Mgrenz curve.
- **Torque Grid: MM**
  - Contains the torque grid.
  - Used for plotting the ETA grid.
- **ETA Grid: ETA**
  - Contains the ETA grid.
  - Have the same dimensions as that of NN and MM sheets
- **Torque Curve: Mgrenz**
  - Contains the values corresponding to torque curve.
  - Have the same columns corresponding to the speed grid.

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. All parameters including the additional ones in each topology is considered as a separate column and therefore if a particular column is topology dependent the data of the other topologies for that corresponding column is treated as NAN 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 to be fed to the model..EXPLAIN THE REASON WHY WITH CITATION PROBABLY::::
As the target values Mgrenz and ETA grids are not provided with the correct dimensions we have an additional step which takes the maximum torque value from the Mgrenz grid 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 ETA grid.
This step ensures that we grant the model the correct dimensions of the ETA grid based on Torque curve and predict likewise.

As reading the files take up a lot of time and compute, we read the files as a onetime job at the start and store them into pythonic objects for faster access in the future.
Both the input and target values for 1st KPI is stored locally as pandas dataframe .csv files whereas the 2nd KPI is stored as separate csv files per variant considering it is in the form of a 2 Dimension (2D) array.
The csv files are then concatenated and stored into a numpy array conserving dimensionality by padding nan values to match dimensionality of the largest 2D array among them.
The array is then saved locally as a .npy file for easy access and loading during training.

The input data consists of about 1500 examples mostly from the Double V Magnet Topology (and about 3 examples each for the other 2 topologies).

Data Preprocessing for GNN

For modelling the GNN, we represent the data in the form of a heterogeneous graph with different node and edge types.

Node types

1. **General**

   - General parameters:
   $$r = \{r_i\} \quad \forall i \in \{a, r, o\}$$

   where:
   - r_a*: Outer Radius of the Stator*
   - r_r*: Outer Radius of the Rotor*
   - r_o*: Center of the EM*

2. **Stator**

   - Slot windings:
   $$sw = \{s_i w_j\} \quad \forall i \in \{1, \ldots, QSim\}, \quad \forall j \in \{1, \ldots, N\}$$
   - Slots:
   $$s = \{s_i\} \quad \forall i \in \{1, \ldots, QSim\}$$

   where

- *Qsim : Count of slots in the Stator*
- *N : Count of copper windings per slot*

3. **Rotor**

- Magnet Flux Barriers:

$$v = \{v_{ij}\} \quad \forall i \in \{1, \ldots, T\}, \quad \forall j \in \{1, \ldots, V\}$$

- Magnets:

$$vm = \{v_i m_j\} \quad \forall i \in \{1, \ldots, T\}, \quad \forall j \in \{1, \ldots, V\}$$

where

- *T : Topology type of the EM*
- *V : Type of Magnet*

As Valeo only manufactures Double V magnets we consider it to be 2

Edge types

1. **Angle**
Relevant Paths

$$vm - -vm = \{v_{i_1} m_{j_1} - v_{i_2} m_{j_2}\} \forall i_1, i_2 \in \{1, \ldots, T\}, \quad \forall j_1, j_2 \in \{1, \ldots, V\} \mid i_1 = i_2, \quad j_1 \neq j_2$$

angle=vm-vm

2. **Distance**
Relevant Paths

$$vi - -vi = \{v_{ij_1} - v_{ij_2}\}, \forall i \in \{1, \ldots, T\}, \forall j_1, j_2 \in \{1, \ldots, V\} \mid j_1 \neq j_2$$

$$vi - -vj = \{v_{i_1 j} - v_{i_2 j}\}, \forall i_1, i_2 \in \{1, \ldots, T\}, \forall j \in \{1, \ldots, V\} \mid i_1 \neq i_2$$

$$v - -vm = \{v_{ij} - v_i m_j\} \forall i \in \{1, \ldots, T\}, \quad \forall j \in \{1, \ldots, V\}$$

$$v - -rr = \{v_{ij} - r_r\}, \forall i, j \in \{1, \ldots, T\}$$

$$o - -r = \{(o - r_r), (o - r_a)\}$$

$$rr - -s = \{r_r - s_i\}, \forall i \in \{1, \ldots, QSim\}$$

$$s - -sw = \{s_i - s_i w_j\}, \forall i \in \{1, \ldots, QSim\}, \forall j \in \{1, \ldots, N\}$$

$$s - -ra = \{s_i - r_a\}, \forall i \in \{1, \ldots, QSim\}$$

$$sw - -sw = \{s_i w_{j_1} - s_i w_{j_2}\}, \forall i \in \{1, \ldots, QSim\}, \forall j \in \{1, \ldots, N\} \mid (j_1 == j_2 - 1)$$

distance = vi–vi + vi–vj + v–vm + v–rr + o–r + rr–s + s–sw + s–ra + sw–sw

Node Features

1. **v** = {lmsov, lth1v, lth2v, r1v, r11v, r2v, r3v, r4v, rmt1v, rmt4v, rlt1v, rlt4v, hav}
2. **vm** = {mbv, mhv, rmagv}
3. **r** = {r}
4. **s** = {b_nng, b_nzk, b_s, h_n, h_s, r_sn, r_zk, r_ng, h_zk}
5. **sw** = {bhp, hhp, rhp}

Path Features

1. **vm–vm** = {deg_phi}
2. **vi–vi** = {dsm, dsmu}
3. **vi–vj** = {amtrvj-amtrvi}
4. **v–vm** = {lmav, lmiv, lmov, lmuv}
5. **v–r** = {amtrv, dsrv}
6. **o–r** = {r}
7. **rr–s** = {airgap}
8. **s–sw** = {dhphp}
9. **sw–sw** = {dhpng}
10. **s–ra** = {r_a-(r_i + h_n + h_zk)}

The heterogeneous graph that was constructed earlier is as below:

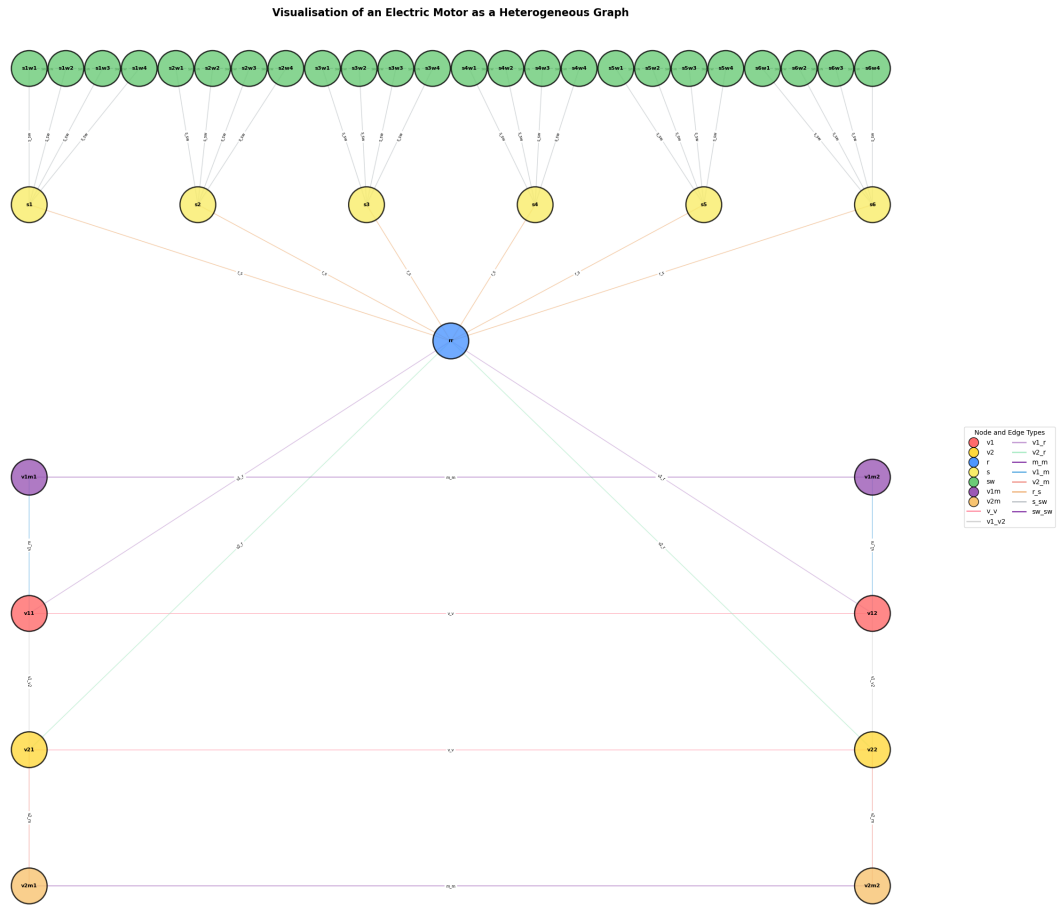


Figure 7: HetGraph

Scaling

To scale the input features, we have used the standard scaler from the sklearn library to scale the values to be shifted such that it is of 0 mean and unit standard deviation. The target values for both KPIs however are scaled with the MinMax Scaler from sklearn to be between 0 and 1.
The scalers are applied to the train validation datasets and the same scaler is used to transform the test dataset to maintain uniformity on the predictions generated.

Dataset splitting

We have also split 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.
Within 5 folds, we expect to cover most grounds on training and have good monitoring on the model's performance for each fold. The best performing model is then chosen from the folds based on the model which has a combination of least aggregated loss for both outputs and scores for each output which are closest to 0.

Modelling

Since we aim to predict continuous vector values, we model this task into a regression problem As a baseline, we first train a MLP on the tabular representation of the data and work on it further to do the same with a heterogeneous GNN.

MLP Model

For the MLP model, we use a single model with input features corresponding to all the features in the tabular topology invariant representation of the data.
The model architecture is build to predict both the 2D and 3 Dimension (3D) KPIs by having 2 separate output layers for each of the KPIs.
Since the 2D KPI's targets are relatively learnable than that of the 3D KPI's targets we have experimented with fewer feed forward layers in the former than in the latter. RELU layers were also added in between to serve as the activation function and produce non-linearities in the model.
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. Batch normalisation layers are used to normalize the input to the next layer and hence speed up the training process.

Th model architecture is designed to have 2 fully connected layers with RELU as an activation function and a 1D batch normalisation layer with a dropout layer for the 1st output.
We have also used a dropout percentage of 0.2 to ensure the model does not overfit the data by dropping out 20 percent of the neurons in the layer. Although the targets for the first output 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.
For the second output we have 5 fully connected layers with RELU as an activation function and dropout and 2D batch normalisation for each layer.

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 addition the count of certain parameters with the motor such as stator poles with its corresponding slot 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.
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.

Loss Function and Evaluation Metrics

Loss Functions

The Mean Squared Error (MSE)loss is the loss function used for our problem with the intention that the losses are minimized. We have adopted 2 methodologies to regularize the loss one each for the 2D and 3D KPI.

Loss Regularization for 2D KPI(Torque curve)

The MSE loss for the 2D KPI is formulated as shown below :

$$\text{MSE\_KPI2D} = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{n}\sum_{j=1}^{n}(y_{ij} - \hat{y}_{ij})^2$$

where

- *p : EM samples*
- *n : dimensionality of 1D vector*

To smoothen out the curve for the 2D KPI(Torque curve) we apply a loss regularisation factor to take into account before backpropagating it the model during training.
We observed the curve inherently follows a decreasing pattern CITE AGAIN::COULD BE A KNOWN FACT:::and hence using this knowledge penalize the loss for non-decreasing values within each prediction.
CITE!!!!If the electromagnetic coil is enabled by the commutator for the time span t3, the (almost) maximal current is running through it's loops and the (almost) maximal magnetic field strength is generated. The (almost) maximal torque is acting on the rotor. If the time span is shortened to t2 by increasing rotational speed, a slightly lower torque is acting, because the current through the coil is decreasing slightly. When reducing the time span to t1, the coil gets disconnected from the input voltage even though just half the maximum current is reached. Accordingly the torque decreases significantly:

$$\text{violations}_{ij} = \text{ReLU}((\Delta\hat{y}_{ij})^2) = \text{ReLU}((\hat{y}_{ij+1} - \hat{y}_{ij})^2)$$

$$\text{regularization\_term} = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{N-1}\sum_{j=1}^{N-1}\text{violations}_{ij}$$

$$\text{Y1 Loss} = \text{MSE\_KPI2D} + \lambda_{\text{y1}} \times \text{regularization\_term}$$

where

$$\lambda_{\text{y1}} : \text{regularization weight}$$

Theoretically, we would expect the model to generate better predictions but on closer observation we notice the curve is still not smooth.
A reason to attribute this could be the model's incapability to infer that loss decrease depends on not just the prediction and target values but also within prediction values. Our deduction is that a single value

calculated for the entire curve may not be sufficient to regularize this loss as we imagined.

A reason to attribute this could be the model's struggle to attempt to generate the next point in the curve to be smaller than the previous point yet also for the further point in addition to making sure the MSE is calculated accurately. Our hypothesis is this loss regularization creates a tug of war between the MSE loss and the regularization term when predicting the whole curve and hence the ever fluctuating curve being generated.

Loss Customization for 3D KPI(Efficiency Grid)

The MSE loss is calculated for the 3D KPI is formulated as shown below :

$$\text{MSE\_KPI3D} = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{m}\frac{1}{n}\sum_{j=1}^{m}\sum_{k=1}^{n}(y_{ijk} - \hat{y}_{ijk})^2$$

where

- *p : EM samples*
- *m : 1st dimension of 3D vector*
- *n : 2nd dimension of 3D vector*

The ETA Grid is a 3D plot of real numbers ranging between 0 and 100.NEED TO CITE...EVEN IF IT IS A KNOWN FACT We noticed in some portions of the grid, the plot not visible as it had nan values.
As ANN cannot be trained to predict NAN values we had these replaced to -1 and have a binary mask constructed such that values corresponding to -1 in the target have value 0 and all other values as 1.
The mask is then multiplied with both the target and the prediction. After this step, the MSE loss is calculated and backpropagated.
Mathematically, this process can be expressed as follows:

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

$$\hat{y}_{ijk}^{\text{masked}} = \hat{y}_{ijk} \cdot M_{ijk}$$

$$y_{ijk}^{\text{masked}} = y_{ijk} \cdot M_{ijk}$$

$$\text{Y2 Loss} = MSE\_KPI3D(\hat{y}^{\text{masked}}, y^{\text{masked}})$$

where M_{ijk} : mask matrix

This formulation ensures that the -1 values (which replaced NaN values in the grid) are ignored in the loss calculation, as they are multiplied by 0 in the mask.

$$\text{Total Loss} = \text{Y1 Loss} + \text{Y2 Loss}$$

Optimizer

Adam optimizer is used for optimization as it is known to be computationally efficient and requires little memoryBLAH BLAH
The optimizer acts once the loss is backpropagated across training each batch of the dataset.
We have also experimented with a learning rate scheduler which reduced the learning rate exponentially by a gamma parameter to decay learning as training progresses across epochs.

Evaluation Metrics

The evaluation metrics we have considered for our regression problem is the standard deviation.
The model with the least combined loss and prediction scores closest to 0 is ideal for our application.

Evaluation Metrics for 2D KPI

The y1 loss for the 2D KPI is formulated as shown below :

$$\text{y1\_KPI2D} = \frac{1}{p}\sum_{i=1}^{p}\sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_{ij} - \hat{y}_{ij})^2}$$

where

- p *: EM samples*
- n *: dimensionality of 1D vector*

Evaluation Metrics for 3D KPI

The y2 loss for the 3D KPI is formulated as shown below :

$$\text{y2\_KPI3D} = \frac{1}{p}\sum_{i=1}^{p}\sqrt{\frac{1}{m}\frac{1}{n}\sum_{j=1}^{m}\sum_{k=1}^{n}(y_{ijk} - \hat{y}_{ijk})^2}$$

where

- p *: EM samples*
- m *: 1st dimension of 3D vector*
- n *: 2nd dimension of 3D vector*

Experiments and Results

Experiments with MLP

In the MLP architecture, the hyperparameters are the learning rate, the number of hidden layers, the number of neurons per layer, learning rate scheduler gamma parameter, batch size, epochs, lambda regularization parameter, dropout probability. The hyperparameters were chosen via a random grid search and we finalized those that resulted in the ideal scores for both y1 ad y2 respectively. The hyperparmaters for the model was tuned over observations of the model's performance across 5 fold cross validation training.
The splits are saved locally and can be used to later to ensure reproducibility. Over the 5 folds, the model performance is observed to ensure its stability and the best performing model is saved to be loaded later for performance. The criteria for choosing the best performing model is roughly estimated based on the least combined loss and the combined score closest to 0.
We chose Weights & Biases (Wandb) to log metrics from the training run and to monitor model performance across folds.
Below is the visualisation of the training and validation metrics for both KPIs.

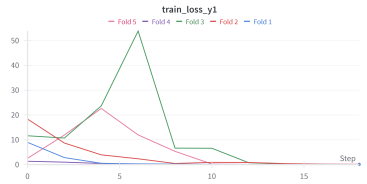


Figure 8: Training Loss for Torque Curve

Figure 9: Training Loss for ETA grid

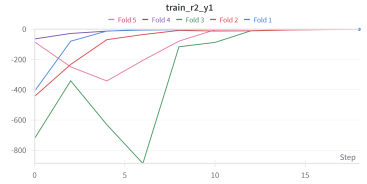


Figure 10: Training Score for Torque Curve

Figure 11: Training Score for ETA grid

From the training plots we see that the model has converged after having run for 10 epochs with a learning rate of 0.0075.

With the same scaler used for training we apply it to the test dataset and in the case of new files we first convert it into the tabular representation our model consumes and then do the scaling.
This is the reason why we preserve the same scalers used during training as we not only evaluate our test dataset we had set aside but also for clients to use on demand.

Figure 12: Validation Loss for Torque Curve

Figure 13: Validation Loss for ETA grid

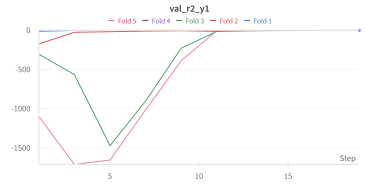


Figure 14: Validation Score for Torque Curve

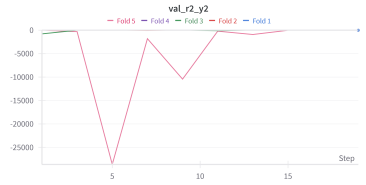


Figure 15: Validation Score for ETA grid

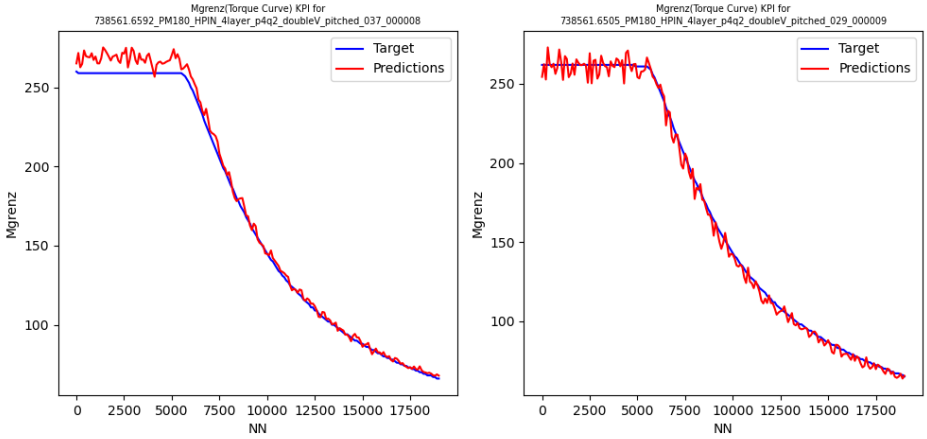
We see a good fit of the model to the data with corresponding y1 and y2 scores approaching close to 1. We have also enabled saving the trained model locally so it can be loaded on demand by the client to run inference.

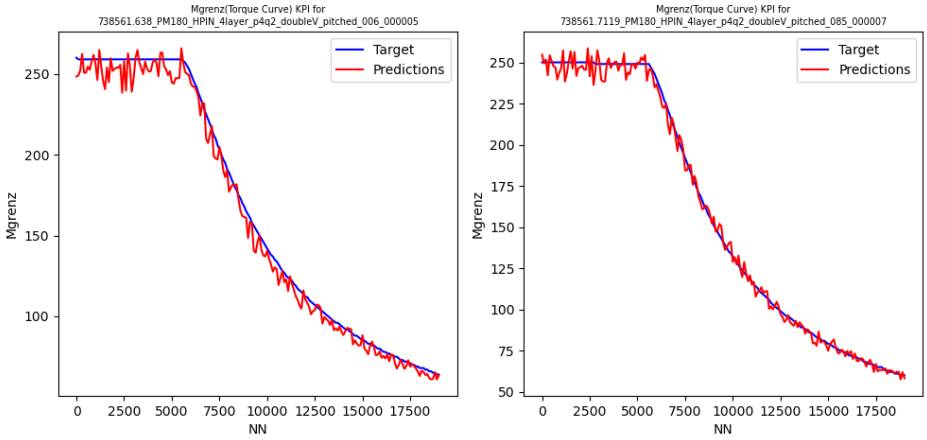
Results with MLP

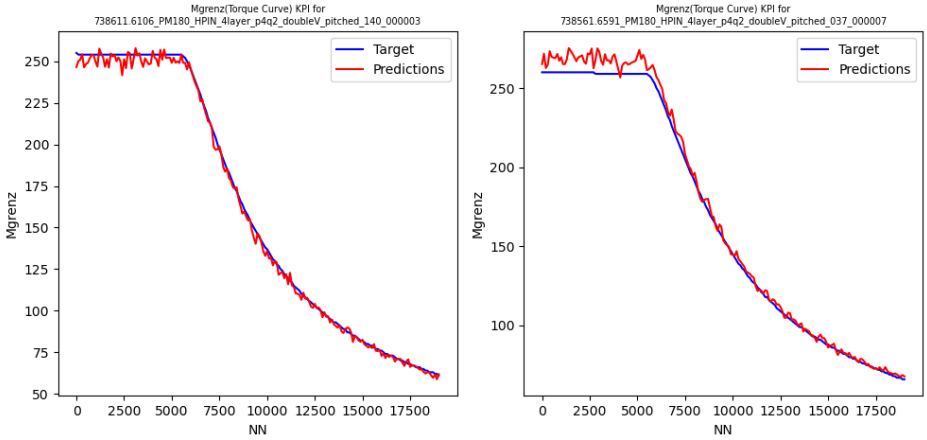
The results of the MLP model from inference is as below:

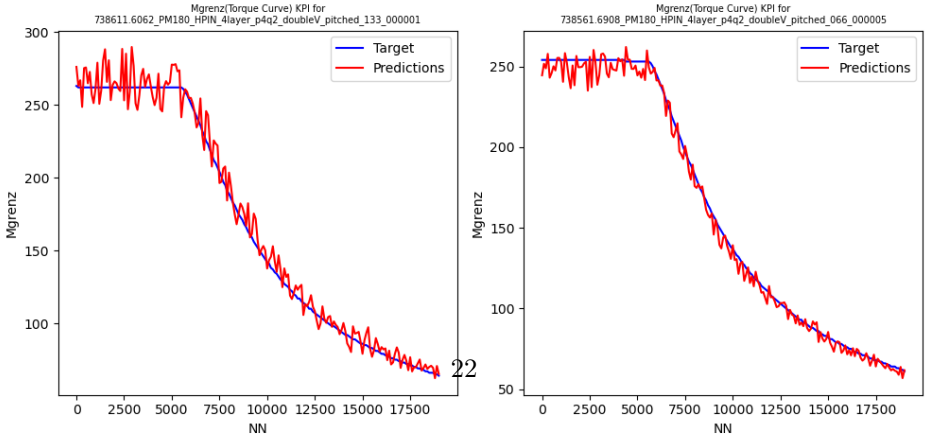
Observations from the predictions helped to correct few discrepancies in our development for instance in the ETA grid we replaced 0 with NAN values which we later understood were both represented different in the grid.
As efficiency values can take up values only between 0 and 100, we consider the same 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.
We have used python 3.12.2 for our development and the pytorch library compatible with cuda. The model was trained on a NVIDIA V100 GPU with blah blah.

Plots of 25 to 35 Examples from the Test Dataset









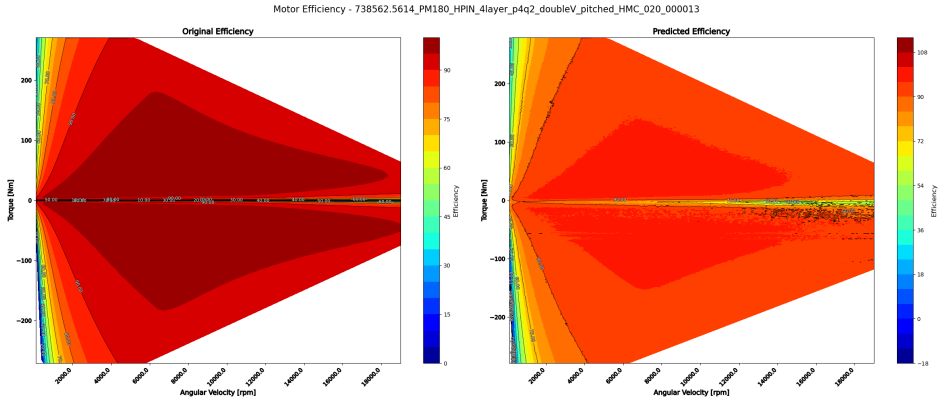


Figure 17: 1st MLP Training Results for 3D KPI(ETA)

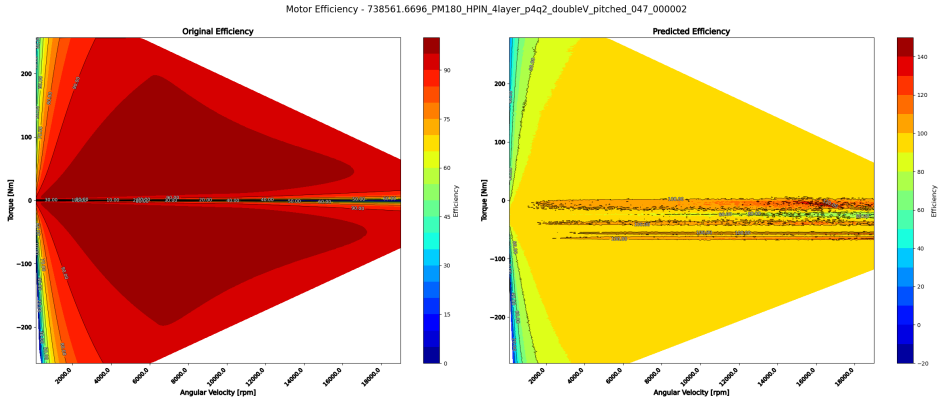


Figure 18: 2nd MLP Training Results for 3D KPI(ETA)

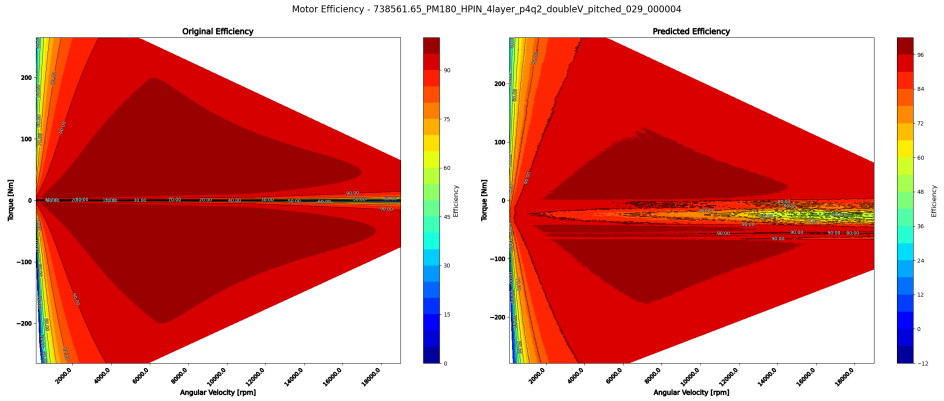


Figure 19: 3rd MLP Training Results for 3D KPI(ETA)

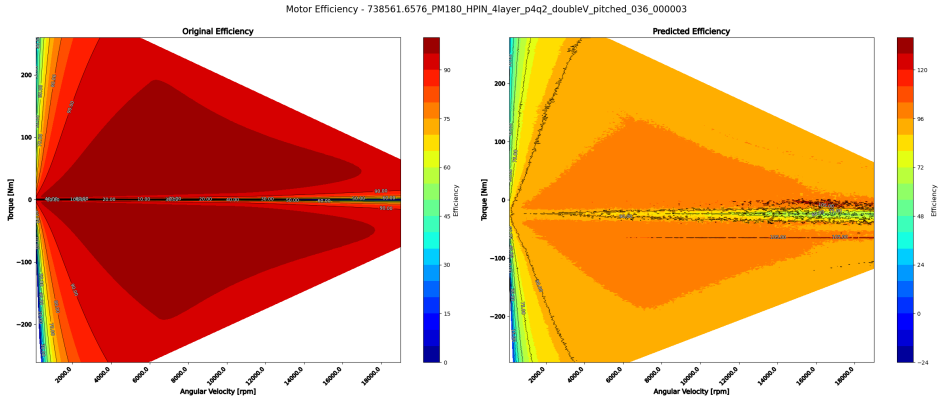


Figure 20: 4th MLP Training Results for 3D KPI(ETA)

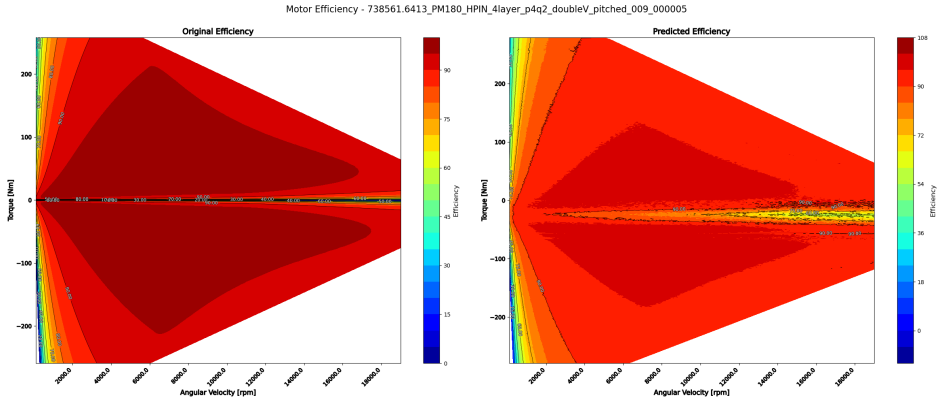


Figure 21: 5th MLP Training Results for 3D KPI(ETA)

Conclusion

List of Figures

List of Tables

Appendix

Bibliography

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