MOTORVEHICLE
UNIVERSITY OF
EMILIA-ROMAGNA

Demagnetization estimation with Machine Learning techniques

Jacopo Ferretti 981206

Machine Learning

A.Y. 2020/2021

Contents

0 Introduction

This report was made by Jacopo Ferretti for for the exam of *Machine Learning*.

I was asked to develop a project involving ML with any topic i liked. I decided to mix my knowledge in electrical machine, and to apply it with this new topic.

In this project, a ML is trained to predict how much overload current an electric motor is able to sustain before demagnetization.

ML software and dataset are available on Github[1] and Kaggle[2]

0.1 Used Software

- Matlab[3]: a programming language well known in engineering due to its simplicity and completeness. It has been used as the environment to create a script for producing the final dataset.
- FEMM[4]: an open source program that can simulate Electro-Mechanical behavior with Finite Element Model (FEM) analysis. It has been used to simulate and create the dataset.
- Google Colab[5]: ad hosted Jupyter notebook service. It has been used to create, train and evaluate the ML model. The programming language used is Python[6], that is particularly suited for ML.
- TensorFlow[7] and Keras[8] is the platform used to create the ML model.
- Pandas[9], Matplotlib[10] and Scikit-learn[11] are the Python packages used to visualize and manipulate data inside Python.

0.2 Used Hardware

Two different Hardwares have been used in this project.

My personal computer have been used to create the dataset. Its specifications in Tab. 0.1

CPU	AMD Ryzen 5 Mobile 3500U
GPU	AMD Radeon Vega 8
RAM	12 GB
OS	Windows

Table 0.1: Personal Computer specifications

Google Colab[5] remote server has been used to develop the ML model.

1 Electric Motor background

In current days, a lot of effort and research are focusing in electric motor design: it is one of the key component for the electrification process of the automotive industries.

There are many types of motors that can be used in automotive applications, and one of the most common is the Permanent Magnet Syncronous Motor (PMSM): as the name says, it is a synchronous motor using magnets to create the magnetic field that moves the rotor. An example of a disassembled PMSM in Fig. 1.1.

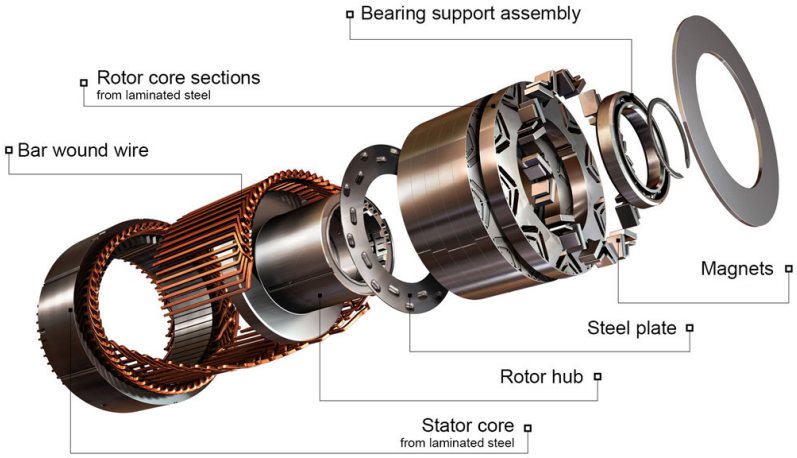


Figure 1.1: Working Components of PMSM[12]

1.1 Type of motor chosen

There are different ways to design a PMSM[12], that won't be covered in this paper. For the sake of simplicity, in this project it has been decided to focus on a specific design configuration of electric motor, otherwise the dataset creation would become too complex, and much more experience would be needed in PMSM design.

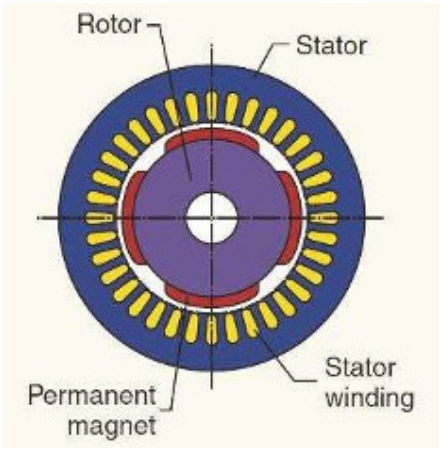
The two main part of an electric motor are the *Stator* and the *Rotor*, for both there are different design that could be chosen.

DISCLAIMER: Here i am assuming to use simplified notation, that could differ to precise keywords used in literature. This section is not a design review of electric motors, but just an help understanding some key design concept

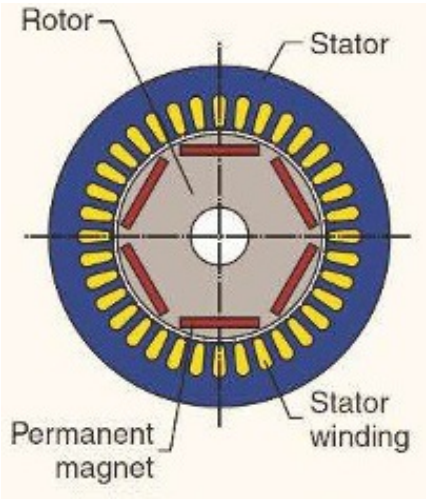
1.1.1 Rotor design chosen

As it is the most simple PMSM type to study and to deal with, Surface Permanent Magnet (SPM) rotor has been chosen to create the dataset. The second motivation to this choice is that this design has been already covered by a previous university course here at MUNER[13].

The characteristic of an SPM rotor is to have the magnets round shaped, and placed in the outer side of the rotor (as can be seen in Fig. 1.2)



(a) Surface Permanent Magnet



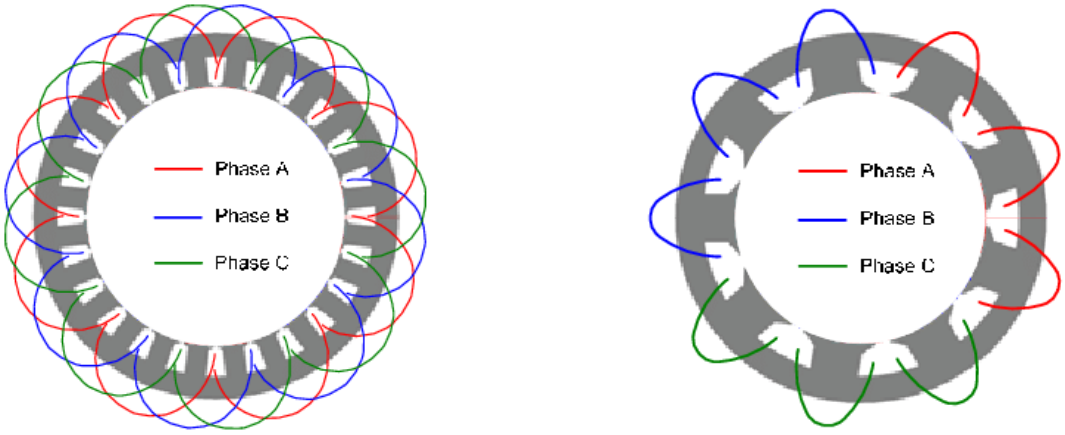
(b) Interior Permanent Magnet

Figure 1.2: Two of the most used rotor magnet placement

1.1.2 Winding layout chosen

The decision of the stator design was easier due to the fewer option available.

The two main stator design (or better, winding layout) are Distribuited Winding (DW) and Fractional Winding (FW). There are advantages and disadvantages for both layouts, but FW has been preferred for this project because demagnetization is more difficult to predict analytically. An example in Fig. 1.3



(a) Distribuited Winding (b) Fractional Winding

Figure 1.3: Two of the most used rotor Winding layout

The wire inside the slots have been simplified as a normal round copper wire, without considering skin effects. Different copper inside slots can be seen in Fig. 1.4

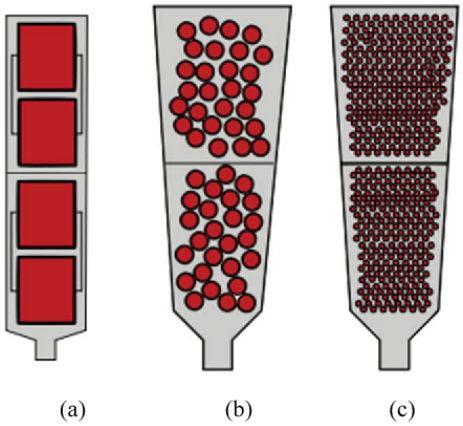


Figure 1.4: Slot area view: (a)hairpin, (b)round wire, (c)litz wire

1.2 Description of the problem

Demagnetization of PMSM motors is a tedious topic in high performance machines: It does happen due to high temperatures and high electric load during the utilization of the motor.

Demagnetization of a magnet happens if the magnetic field inside the magnet H_{magn} is greater, in module, of its intrinsic coercivity H_{CI}(that depends on the material and the temperature of the magnet).

The analytical equation that describe when demagnetization happens is:

$$|H_{magn}| = \left| \frac{\frac{N}{2} I - \frac{\delta_0}{\mu_0} B_r}{l_{magn} + \delta_0 \frac{\mu_d}{\mu_0}} \right| < |H_{CI}| \tag{1.1}$$

where

- N: Series conductors per phase.
- I: Peak current crossing the windings.
- δ_0: Airgap thickness.
- μ_0: magnetic permeability of free space.
- $B_r = B_{r0}(1 + \alpha_{B_r}(T - T_0))$: Magnet remanence.
- l_{magn}: magnet thickness.
- μ_d: Permeability of the magnet.
- $H_{CI} = H_{CI0}(1 + \alpha_{H_{CI}}(T - T_0))$: intrinsic coercivity
- H_{CI0}, B_{rI0}: Reference Value.
- $\alpha_{H_{CI}}, \alpha_{B_r}$: temperature coefficient.

The downside of this equation, is that it assumes that the magnetic field produced by the magnet is sinusoidal, without taking in consideration side harmonics.

FW motors are known to produce an higher number of side harmonics, so this formula is not particularly suited for that stator type.

Another way to check demagnetization is to measure the H_{magn} value inside magnets using FEM Software.

Temperature was set to the magnet temperature limit: $T = 140\,°\text{C}$ as datasheet. Magnet used is *Cibas ren35h NeFeb*.

1.3 Why this work could be useful?

As already said, it is possible to check for demagnetization using using FEM, but that approach has different disadvantages:

- The duration of FEM simulation can last up to tens of minutes → Design optimization time will be considerable.
- FEM simulation can only be used at the end ot the motor design process, when geometry has already been shaped → Precise prediction for demagnetization is not possible during early design stages.

This two disadvantages are particularly bad for high performance PMSM motors used in automotive: those type of machines are usually pushed to the limit by exploiting their Overload (OL) capability.

1.3.1 Overload of a motor

The OL is an important features of high performance electric motors: it is possible to inject more current, for a short interval of time time, and produce more torque than what the motor was built for.

How is it possible to run in OL without damaging the motor? the electric/mechanic dynamics is relatively fast compared to the slow thermal dynamics: so to avoid overheating it is necessary to use OL for a short period of time.

If the motor temperatures are kept in a safe range, the only concern about OL is the demagnetization.

1.3.2 Overload and Demagnetization

As the approximated equation in Eq. (1.1) suggest, demagnetization can happen if the current I is too high. This mean that for each motor it is possible to have a maximum overload current that it is not possible to exceed. it can be useful to predict how much OL it is possible to apply for two main reason:

- Predict OL at early development stage to modify it if different behavior are desired.
- Apply optimization algorithm at high speed.

1.4 Proposed target of this work

The target of this work is to create a machine learning tool capable to predict how much OL is possible without going in demagnetization, given few initial parameters.

This work is made purely for didactic purpose, this is why a single type of motor is investigated, and limited parameters/materials can be chosen.

2 Dataset Creation

As already mentioned, the Dataset has been created with Matlab And FEMM. It was possible to create a script that simulates in FEM a randomly generated electric motor, and measure the magnetic field inside the magnets to check for demagnetization.

Thanks to Giacomo Sala, professor of Electric Motor Design at MUNER[13], that provided to the students the code to easily simulate an electric motor using FEMM and Matlab.

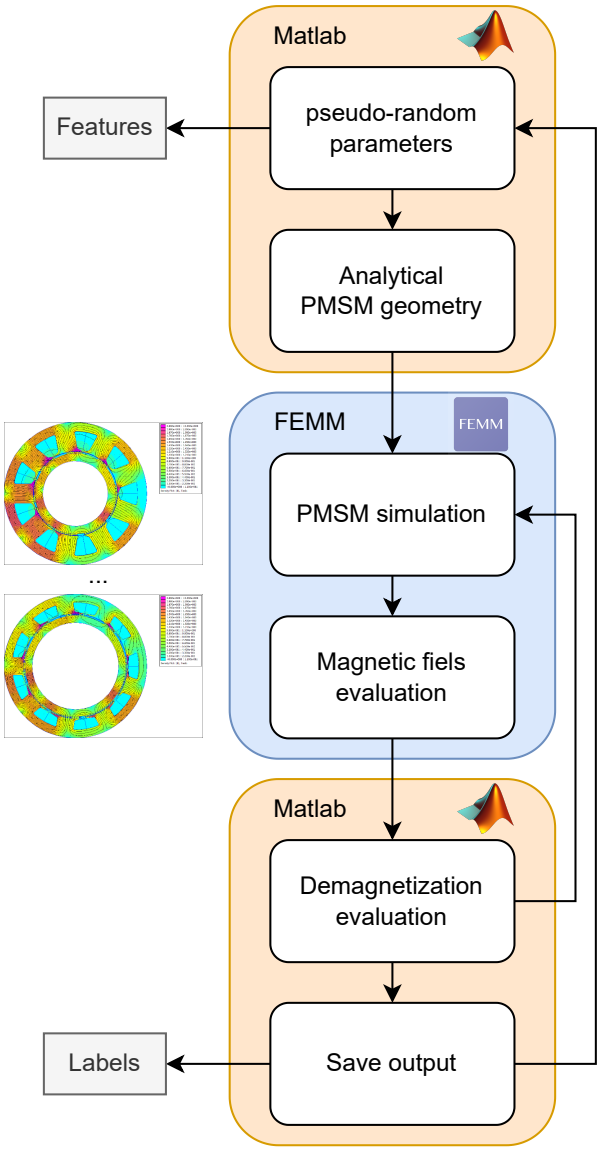


Figure 2.1: Dataset creation workflow

The main steps of the dataset generation process are summarized in Fig. 2.1. In greater details the steps are:

1. Pseudo randomly generate the motor parameters and values, to be used as *features*.
2. Generate the motor geometry analytically.
3. Run FEM simulation of the motor.
4. Measure the magnetic field inside the magnets, and extrapolate the point where field is maximum.
5. Compare the measured field to its maximum possible value H_{CI}.
6. Repeat measurements until demagnetization happens.
7. Save the last "safe" current overload value that will not create overload as a *label*.

2.1 Features

Features have been pseudo randomly generated: a known working motor configuration has been chosen, where some parameters (the features) have been changed randomly in a specific range.

It was not possible to chose the features completely randomly, or some motor configuration could have been physically impractical.

The features, and their range, are:

	Feature	Mean value	Range
T_{spec}	Nominal Torque	150 N m	± 75 N m
W_{spec}	Rated Speed	3.5 krpm	± 1.75 krpm
Bmr	Magnetic Loading	0.75 T	± 0.375 T
δ	Electric Loading	4 kA/m	± 2 kA/m
m	Aspect Ratio	1.5	± 0.75
S_o	Slot Opening	5 mm	± 1.75 mm
d_0	Airgap Thikness	1.2 mm	± 0.42 mm
l_{magn}	Magnet height	l_{magn0} from calculation	$l_{magn0} * 1.5 \pm l_{magn0} * 0.1$
h_4	Slot Opening height	2 mm	± 0.7 mm
h_3	Wedge Height	5 mm	± 1.75 mm

Table 2.1: Features summary

2.1.1 Note on l_{magn}

The only feature that has not chosen like the others is the magnet height l_{magn}. This because l_{magn} needs to be greater than a certain quantity (dependent on other chosen features), so it was not possible to randomize it from a given value. To make the output result of the simulations more interesting, it has been decided to increase the calculated value l_{magn0} by 1.5, and to randomize this last value: doing so, it was possible to see more variegate values of maximum overload in output.

If l_{magn} was chosen only considering the initial calculation, we would have seen a lot of low overload value, and the labels would have been much less interesting.

2.2 Labels

Each simulation outputs a single label OL_{max}: a number which represents the maximum overload that the motor is capable before demagnetizing.

2.2.1 Magnetic field H_{magn} measurement

The magnetic field is measured in FEM simulation along the perimeter of a circumference that touches the mean height of the magnet: an example can be seen as the red line in Fig. 2.2

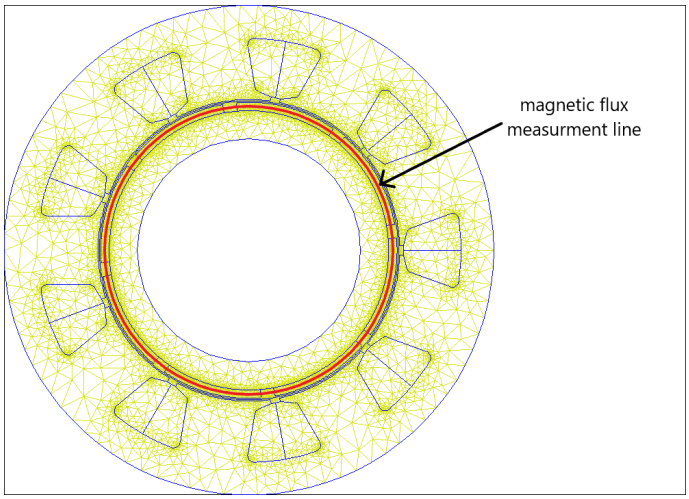


Figure 2.2: Magnetic field measurement line example

2.2.2 Maximum overload detection

The overload detection and the labels output follows the following scheme:

1. Matlab script starts by simulating the motor at nominal current.
2. The maximum value of magnetic field along that line is find.
3. Check if $|H_{magn}| < |H_{CI}|$.
4. If the check is **true**: no demagnetization is happening, reiterate the simulation with increased current.
5. If the check is **false**: demagnetization is happening, stop the simulation, save the last overload value as the label (that OL will be the last one before demagnetization).

This cycle is repeated for all the simulated motors.

2.2.3 Note on label value

Labels value are in a range between 0 and 11, where 0 means no overload, and 11 represents an overload of 6.5.

To get the overload value OL from the output of the script, the following equation needs to be applied:

$$OL = \frac{Label}{2} + 1 \tag{2.1}$$

3 ML Model

After the creation of the dataset, the main machine learning machine was trained and tested.

This remaining part has been developed on a Jupyter notebook, publicly available on Github and Kaggle alongside the dataset used.

- Github repo: `https://github.com`
- Kaggle repo: `https://www.kaggle.com`

3.1 Dataset analysis

The dataset used has `7000` entries in total, each one representing a simulated motor.

Thanks to the various libraries used, it was possible to analyze data using intuitive graphical plots.

3.1.1 Frequency plot

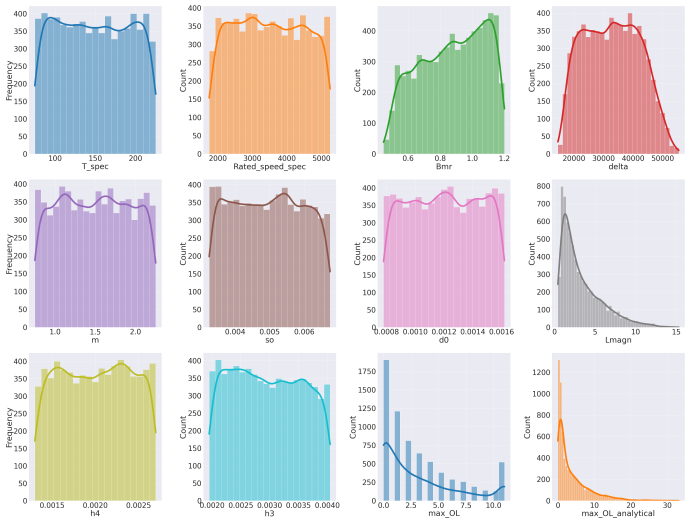


Figure 3.1: Frequency plot

The frequency plot in Fig. 3.1 shows that all the labels are randomly distributed, except fot the magnet height L_{magn}.

The last two plots, shows the distribution of FEM and analytical prediction.

3.1.2 Correlation Matrix

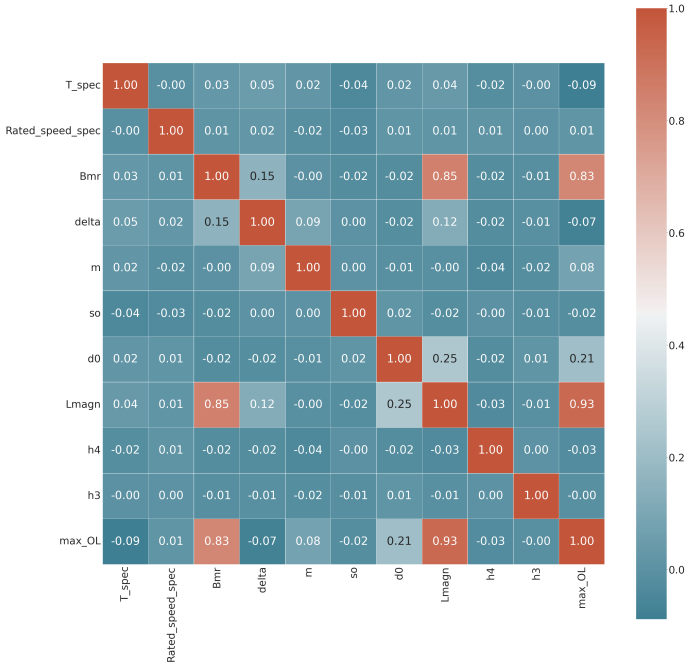


Figure 3.2: Correlation Matrix

As it is possible to see from the correlation matrix in Fig. 3.2, the magnet height L_{magn} and the magnetic loading B_{mr} are highly correlated with the maximum overload.

3.1.3 Dataset Split

The dataset was split in three different set:

- Train set: used to train the Machine.
- Test set: Used for the final evaluation of the Machine.
- Validation set: used to validate the Machine during the training.

3.1.4 Stratification

Stratification is used to be sure that the proportion of labels in the sample produced will be the same as the proportion of labels in the dataset. This will increase the reproducibility.

3.2 Metrics

There are different metrics useful to identify a good model.

Those metrics have been introduced for binary classification, but it is possible to extend those to multi class. Notation:

- TP: True Positive.
- TN: True Negative.
- FP: False Positive.
- FN: False Negative.

3.2.1 Precision

Number of items correctly identified as positive out of the total items identified as positive:

$$\frac{TP}{TP + FP} \tag{3.1}$$

3.2.2 Recall

Number of items correctly identified as positive out of the total actual positives:

$$\frac{TP}{TP + FN} \tag{3.2}$$

3.2.3 Accuracy

Number of items correctly identified as either truly positive or truly negative out of the total number of items:

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

Usually it is the most used metrics for a first evaluation of the model.

3.2.4 F1-Score

The harmonic average of the precision and recall, it measures the effectiveness of identification when just as much importance is given to recall as to precision:

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

In multiclass, it can be calculated as an average of all the F1 score, or as a weighted average. Weights can be the number of samples for each category.

3.2.5 Loss

Loss is a number indicating how bad the model's prediction was on a single example. It is very important because it is the number that a model should seek to minimize during training.

Usually Mean Square Error (MSE) is used as the loss function for regression:

$$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - Prediction(x))^2 \tag{3.5}$$

In multiclass classification, categorical Crossentropy is the most used:

$$CrossEntropyLoss = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \tag{3.6}$$

3.3 Regression model

The first machine trained is a linear regression model, that uses one or multiple parameters. The decision to use such simple machine is to start by using something simple and understandable, and it is interesting to see how performance increases by introducing complexity on the machine.

3.3.1 Linear regression - one variable

In order to train such machine with one variable, we need to chose a suitable feature that is going to weight a lot on the output label. In Fig. 3.2 it is possible to see that L_{magn} is the most correlated to the label, so that feature has chosen. The trained model can be summarized as follows:

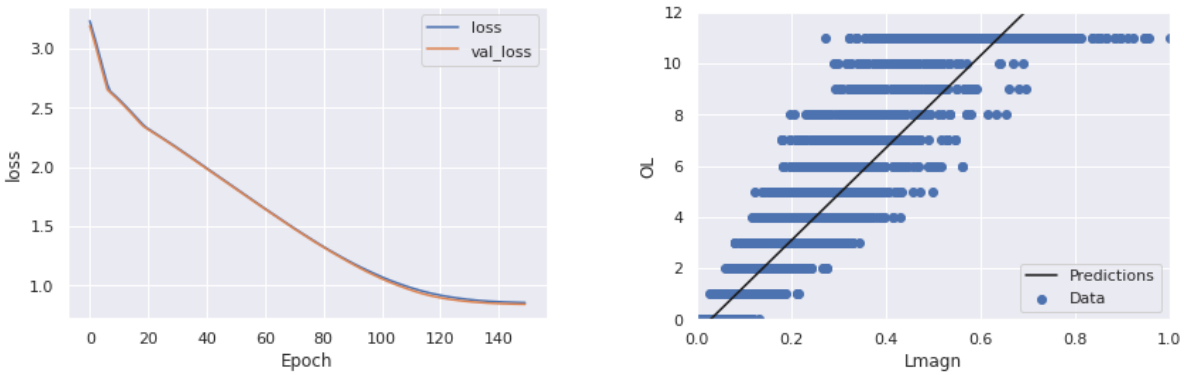
```
Model: "sequential"
_________________________________________________________________
    Layer (type)                 Output Shape              Param #
=================================================================
    flatten (Flatten)            (None, 1)                 0

    dense (Dense)                (None, 1)                 2

=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

Information about the training:

Optimizer	epoch	loss	metrics
Adam	150	mean_absolute_error	accuracy

The result of this machine can be visualized in Fig. 3.3, and its result metrics are:

Validation loss	Validation accuracy
0.8425	0.3482



(a) loss (b) plot

Figure 3.3: Linear regression - one variable

3.3.2 Linear regression - multi variable

Now the linear regression machine have all the features in input, so we except to have better metrics. The trained model can be summarized as follows:

```
Model: "sequential"
_________________________________________________________________
    Layer (type)                 Output Shape              Param #
=================================================================
    flatten (Flatten)            (None, 10)                0

    dense (Dense)                (None, 1)                 11

=================================================================
Total params: 11
Trainable params: 11
Non-trainable params: 0
_________________________________________________________________
```

Information about the training:

Optimizer	epoch	loss	metrics
Adam	150	mean_absolute_error	accuracy

The result of this machine can be visualized in Fig. 3.4, and its result metrics are:

Validation loss	Validation accuracy
0.6756	0.4089

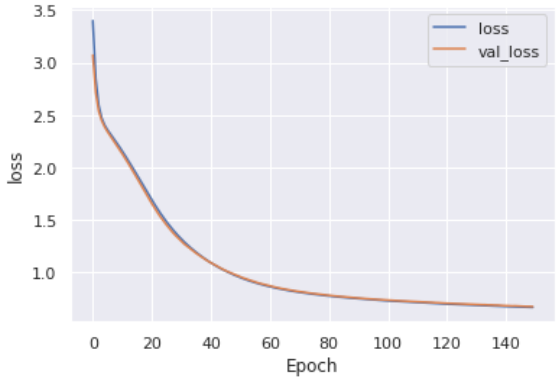


Figure 3.4: Linear regression - one variable - loss

3.3.3 NON Linear regression - one variable L_{magn}

L_{magn} is again the feature used to train this one variable regression, but now a non linear regression will be used. A Deep Neural Network (DNN) has been used to create this non linear regression machine. The trained model can be summarized as follows:

```
Model: "sequential"
_________________________________________________________________
    Layer (type)                 Output Shape              Param #
=================================================================
    flatten (Flatten)            (None, 1)                 0

    dense (Dense)                (None, 32)                64

    dense_1 (Dense)              (None, 64)                2112
```

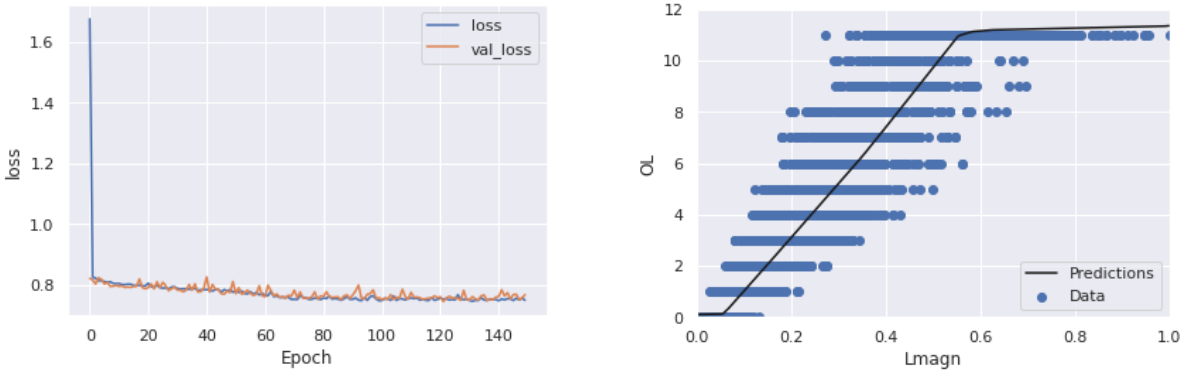
```
    dense_2 (Dense)              (None, 128)               8320

    dense_3 (Dense)              (None, 1)                 129

=================================================================
Total params: 10,625
Trainable params: 10,625
Non-trainable params: 0
_________________________________________________________________
```

Information about the training:

Optimizer	epoch	loss	metrics
Adam	150	mean_absolute_error	accuracy

The result of this machine can be visualized in Fig. 3.5, and its result metrics are:

Validation loss	Validation accuracy
0.7681	0.3518



(a) loss (b) plot

Figure 3.5: NON Linear regression - one variable - L_{magn}

3.3.4 NON Linear regression - one variable B_{mr}

The DNN non linear regression machine is the sames before, but now the feature used is B_{mr}: the second most relevant according to the correlation matrix (Fig. 3.2). The trained model can be summarized as follows:

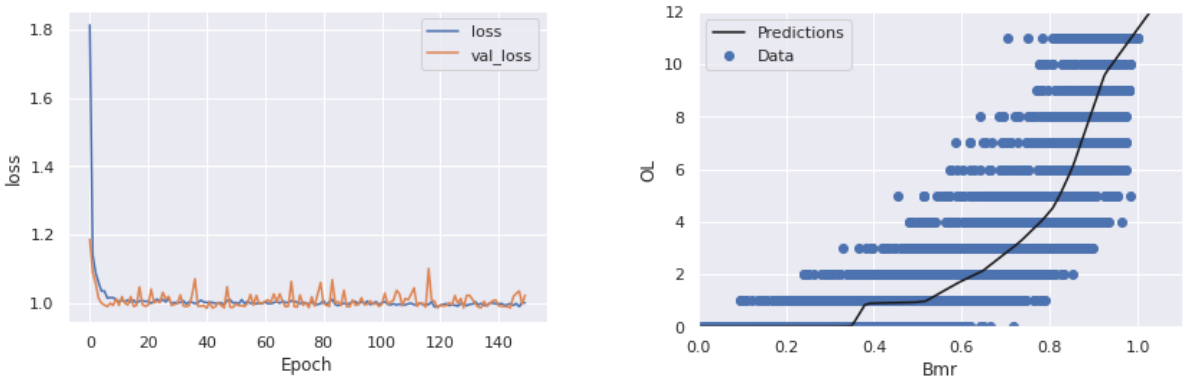
```
Model: "sequential"
_________________________________________________________________
    Layer (type)                 Output Shape              Param #
=================================================================
    flatten (Flatten)            (None, 1)                 0

    dense (Dense)                (None, 32)                64

    dense_1 (Dense)              (None, 64)                2112

    dense_2 (Dense)              (None, 128)               8320

    dense_3 (Dense)              (None, 1)                 129

=================================================================
Total params: 10,625
Trainable params: 10,625
Non-trainable params: 0
_________________________________________________________________
```

Information about the training:

Optimizer	epoch	loss	metrics
Adam	150	mean_absolute_error	accuracy

The result of this machine can be visualized in Fig. 3.6, and its result metrics are:

Validation loss	Validation accuracy
1.0235	0.3357



(a) loss

(b) plot

Figure 3.6: NON Linear regression - one variable - B_{mr}

3.3.5 NON Linear regression - multi variable

Now the non linear regression machine utilize in input all the features, so we except to have better metrics. The depth of the network is increased to try reaching an useful value of metrics. The trained model can be summarized as follows:

```
Model: "sequential"
_________________________________________________________________
    Layer (type)                 Output Shape              Param #
=================================================================
    flatten (Flatten)            (None, 10)                0

    dense (Dense)                (None, 32)                352

    dense_1 (Dense)              (None, 64)                2112

    dense_2 (Dense)              (None, 128)               8320

    dense_3 (Dense)              (None, 256)               33024

    dense_4 (Dense)              (None, 1)                 257

=================================================================
Total params: 44,065
Trainable params: 44,065
Non-trainable params: 0
_________________________________________________________________
```

Information about the training:

Optimizer	epoch	loss	metrics
Adam	150	mean_absolute_error	accuracy

The result of this machine can be visualized in Fig. 3.7, and its result metrics are:

Validation loss	Validation accuracy
0.1590	0.4339

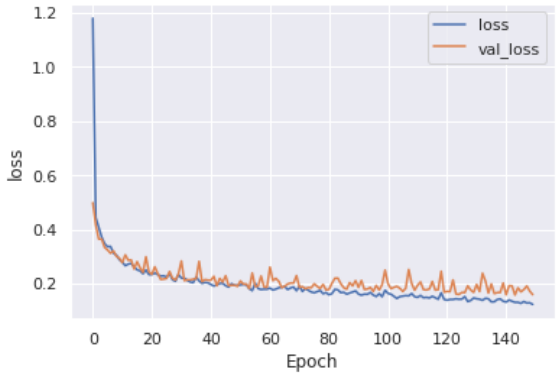


Figure 3.7: non Linear regression - Multi variable - loss

Accuracy and loss are the highest until now, but are still unusable in real word applications.

3.4 Classification Deep Neural Network

The machine using classification gives the best result for our problem, as it will be seen trough the metrics. The trained DNN model has been chosen by a trial and error approach, and it is quite deep. The trained model can be summarized as follows:

```
Model: "sequential"
_________________________________________________________________
    Layer (type)                 Output Shape              Param #
=================================================================
    flatten (Flatten)            (None, 10)                0
```

```
    dense (Dense)                (None, 32)                352

    dense_1 (Dense)              (None, 64)                2112

    dense_2 (Dense)              (None, 128)               8320

    dense_3 (Dense)              (None, 256)               33024

    dense_4 (Dense)              (None, 12)                3084

=================================================================
Total params: 46,892
Trainable params: 46,892
Non-trainable params: 0
_________________________________________________________________
```

Information about the training:

Optimizer	epoch	loss	metrics
Adam	150	SparseCategoricalCrossentropy	accuracy

The result of this machine can be visualized in Fig. 3.8, and its result metrics are:

Validation loss	Validation accuracy
0.2841	0.8893

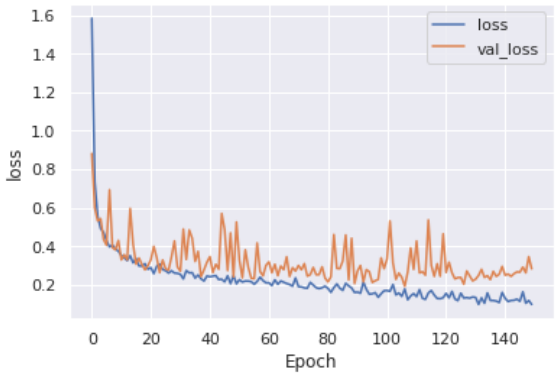


Figure 3.8: DNN Classification - loss

3.4.1 Evaluation with Test data

It is assumed that it is the most performing network for our model, so it is suitable to evaluate it using the test data. The result metrics of the DNN classification model are:

Validation loss	Validation accuracy
0.2963	0.9029

It can be seen that Loss increases compared with the non linear regression - multi variable. This is excepted because regression take in consideration the distance between output and labels more than a classification, and our problem is more prone to regression by design.

NOTE: comparing two losses, having different loss functions (like in this case) is not completely correct. Accuracy should be used instead.

Other metrics can be obtained, like the Precision, Recall and f1-score:

	precision	recall	f1-score	support
0	0.979	0.987	0.983	381
1	0.950	0.938	0.944	242
2	0.929	0.883	0.905	162
3	0.836	0.953	0.891	128
4	0.812	0.867	0.839	105
5	0.789	0.737	0.762	76
6	0.811	0.741	0.775	58
7	0.766	0.706	0.735	51
8	0.737	0.737	0.737	38
9	0.793	0.821	0.807	28
10	0.724	0.808	0.764	26
11	1.000	0.933	0.966	105
accuracy			0.903	1400
macro avg	0.844	0.843	0.842	1400
weighted avg	0.904	0.903	0.903	1400

Another interesting plot to see is the confusion matrix in Fig. 3.9. It can give an intuitive idea on the precision and loss of the network by using colors.

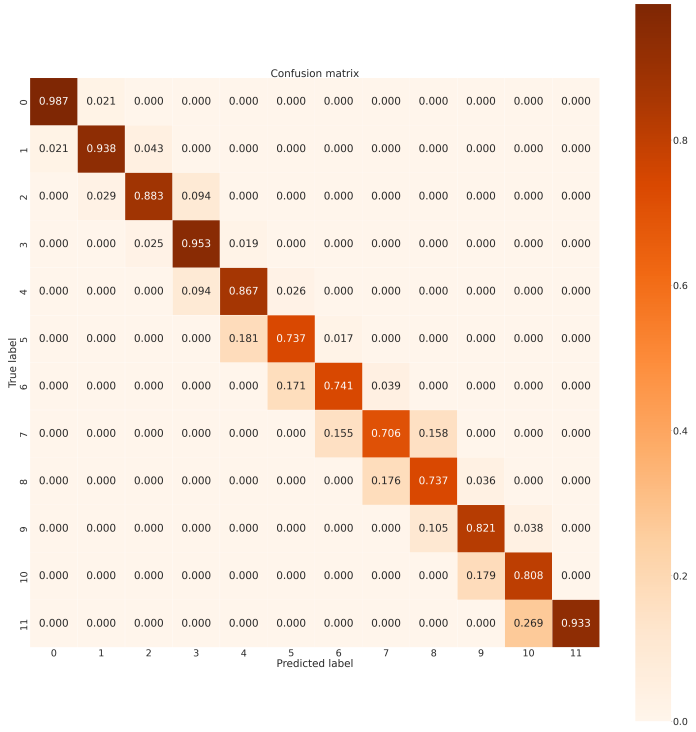


Figure 3.9: DNN Classification - Confusion Matrix

3.5 Comparison with analytical formula

It is possible to use the analytical formula in Eq. (1.1) to predict the maximum overload. As already discussed, it is a simplified formula, and worse accuracy is excepted.

The same dataset of 7000 samples has been used, and then the result of the function was rounded with different rounding method. Rounding is needed to make the analytical and ML results comparable by the same metrics (accuracy).

The used rounding method are

- *Nearest*: rounds the number to the closest integer.
- *Ceil*: rounds a number to the closest integer value larger than the current number.
- *Floor*: rounds the integer to the closest integer smaller than the current one.

The resulting accuracy of the formula with different rounding methods is shown on the following table:

Rounding method	Accuracy
Nearest	0.588
Ceil	0.240
Floor	0.685

The floor rounding got the best accuracy, the analysis will be focusing on it. Confusion matrix of the analytical model using floor rounding can be seen in Fig. 3.10. Precision, Recall and f1-score are the following:

	precision	recall	f1-score	support
0	0.722	0.901	0.801	1903
1	0.520	0.454	0.485	1211
2	0.815	0.608	0.697	812
3	0.945	0.783	0.857	641
4	0.911	0.759	0.828	526
5	0.721	0.692	0.706	380
6	0.609	0.581	0.595	289
7	0.413	0.377	0.394	257
8	0.239	0.277	0.257	191
9	0.177	0.209	0.191	139
10	0.261	0.281	0.271	128
11	0.826	0.937	0.878	523
accuracy			0.685	7000
macro avg	0.597	0.572	0.580	7000
weighted avg	0.692	0.685	0.682	7000

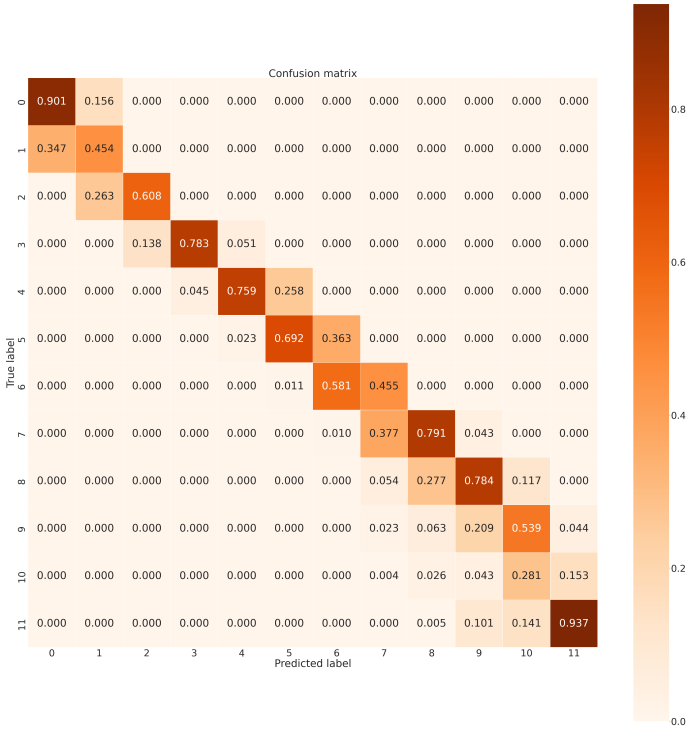


Figure 3.10: Analytical Model - Confusion Matrix

4 Results, future works and improvements

In the following table, all the interesting result metrics can be appreciated:

	Validation loss	Validation accuracy
Linear regression - one variable	0.8425	0.3482
Linear regression - multi variable	0.6756	0.4089
NON Linear regression - one variable L_{magn}	0.7681	0.3518
NON Linear regression - one variable B_{mr}	1.0235	0.3357
NON Linear regression - multi variable	0.1590	0.4339
DNN classification model	0.2841	0.8893
Analytical Equation	−−	0.685

It can be seen that the DNN classification model outperforms any other model used in this report. this work can be considered as a good start for ML application in PMSM demagnetization.

It is proper to remember that this model is strongly limited: only one type of motor is used, with limited amount motor specifications, and with fixed materials. A future work could be to extend the network, and make it work for more type of motors. Despite this limitation, it is possible to see how ML can be successfully be used to speed up and improve electric motor design, that was the initial target of this work.

In Fig. 4.1, the final network of the DNN classifier.

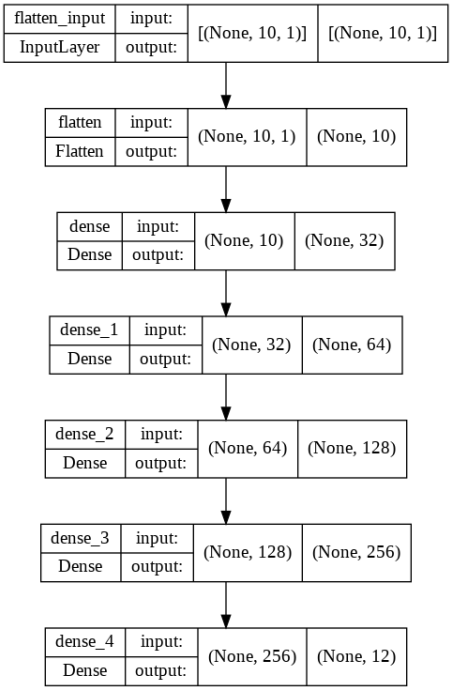


Figure 4.1: DNN Classification network

Acronyms

DNN Deep Neural Network 2, 18, 20, 22–25, 27, 28

DW Distribuited Winding 5, 6

FEM Finite Element Model 3, 7, 9–11, 13

FW Fractional Winding 5–7

ML Machine Learning 1, 3, 13, 25, 27, 28

MSE Mean Square Error 16

OL Overload 1, 8, 12

PMSM Permanent Magnet Syncronous Motor 4, 6, 8, 27

SPM Surface Permanent Magnet 4, 5

References

[1] **Github homepage**. https://github.com.

[2] **Kaggle homepage**. https://www.kaggle.com.

[3] **Matlab homepage**. `https://it.mathworks.com/products/matlab.html`. Last accessed January 22, 2023.

[4] **FEMM homepage**. `https://www.femm.info/wiki/HomePage`. Last accessed January 22, 2023.

[5] **Google Colab homepage**. `colab.research.google.com`. Last accessed January 22, 2023.

[6] **Python homepage**. https://www.python.org.

[7] **Python homepage**. https://www.tensorflow.org.

[8] **Keras homepage**. https://keras.io.

[9] **Python homepage**. https://pandas.pydata.org.

[10] **Matplotlib homepage**. https://matplotlib.org.

[11] **scikit-learn homepage**. https://scikit-learn.org.

[12] **Dmitry Levkin Permanent magnet synchronous motor**. https://en.engineering-solutions.ru/motorcontrol/pmsm/.

[13] **MUNER homepage**. https://en.engineering-solutions.ru/motorcontrol/pmsm/.

Appendix A m400-50a

Typical data for SURA® M400-50A

T	W/kg at 50 Hz	VA/kg at 50 Hz	A/m at 50 Hz	W/kg at 100 Hz	W/kg at 200 Hz	W/kg at 400 Hz	W/kg at 1000 Hz	W/kg at 2500 Hz
0,1	0,02	0,07	32,6	0.07	0.16	0,48	2.12	8.64
0,2	0,09	0,18	43,5	0.26	0.64	1,80	7.49	30.1
0,3	0,19	0,33	50,8	0.54	1.35	3,77	15.3	62.7
0,4	0,31	0,50	57,2	0.88	2.25	6,29	25.7	109
0,5	0,46	0,69	63,4	1.27	3.33	9,37	39.0	172
0,6	0,62	0,91	69,9	1.73	4.58	13,1	56.1	256
0,7	0,81	1,16	77,3	2.24	6.03	17,5	77.1	367
0,8	1,01	1,46	86,0	2.80	7.68	22,7	103.1	509
0,9	1,24	1,81	97,2	3.44	9.58	28,8	135.0	685
1,0	1,49	2,23	113,2	4.15	11.7	35,9	173.3	899
1,1	1,76	2,79	137,8	4.95	14.2	44,2	218.8	1155
1,2	2,09	3,60	180,2	5.85	17.0	53,8	272.4	1453
1,3	2,46	5,07	269,5	6.88	20.2	64,9	334.6	1793
1,4	2,96	8,80	516,8	8.18	23.8	77,4	405.6	2130
1,5	3,57	21,6	1307	9.82	28.3	91,7	488.4	
1,6	4,38	57,2	3180					
1,7	5,02	128	6361					
1,8	5,47	243	10890					

Loss at 1.5 T , 50 Hz, W/kg	3,57
Loss at 1.0 T , 50 Hz, W/kg	1,49
Anisotropy of loss, %	8
Magnetic polarization at 50 Hz	
H = 2500 A/m, T	1,59
H = 5000 A/m, T	1,68
H = 10000 A/m, T	1,79
Coercivity (DC), A/m	50
Relative permeability at 1.5 T	1050
Resistivity, μΩcm	42
Yield strength, N/mm²	325
Tensile strength, N/mm²	465
Young's modulus, RD, N/mm²	200 000
Young's modulus, TD, N/mm²	210 000
Hardness HV5 (VPN)	165

RD represents the rolling direction
TD represents the transverse direction
Values for yield strength (0.2 % proof strength)
and tensile strength are given for the rolling direction
Values for the transverse direction are approximately 5% higher

cogent
Surahammars Bruks AB

Oct 2009

PRODUCT INFORMATION

DAMID 180

Round enamelled winding wire of copper, heat resistant, class 180

Product name:
Damid 180 - Gr 1
Damid 180 - Gr 2

Specifications:
IEC 60317-8

UL approval:
Approved: Damid 180
UL-file no: E101843

Class: 180
Temperature index ≥ 180 °C
Heat shock: ≥ 200 °C

Conductor material:
EN 1977 - ETP1 CW003A
EN 1977 - ETP CW004A
ASTM B49 - ETP C11000/C11040

Insulation:
THEIC-modified esterimide

Properties:
- Suitable for winding in high speed machines
- Very good resistance to transformer oils
- Very good resistance to typical solvent
- Freon resistant
- Excellent resistance to mechanical stress

Field of application:
- Electrical devices
- Oil-immersed transformers
- Cast-resin transformers

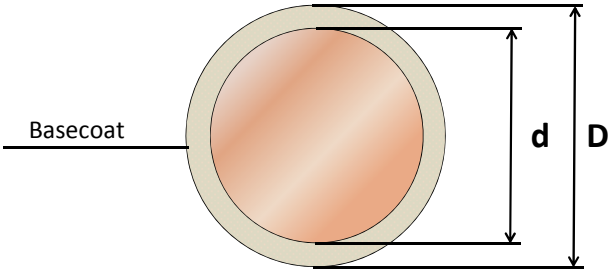
Dimension range:

Damid 180 - Gr 1	0,090 ≤ Ø ≤ 6,00
Damid 180 - Gr 2	0,090 ≤ Ø ≤ 6,00

Standard packaging:

0,150 ≤ Ø ≤ 3,35 mm	A250/400, A315/500
3,35 < Ø ≤ 6,00 mm	K500, K630, K710

Shelf life:
6 years, under normal ambient conditions



D - d = Increase

LWW group
dahréntråd | isodraht | śląska | 利里达尔线材 liljedahl wire

Certified according to ISO 9001:2000, ISO-TS 16949, ISO 14001

Appendix C Cibas ren35h NeFeb datasheet

REN | GEN

Sintered NdFeB

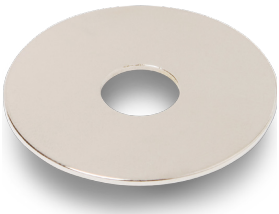
GRADES	Br		HcB		HcJ		BH max		Max. Working Temp.**
	G	T	kOe	kA/m	kOe	kA/m	MGOe	kJ/m³	°C
REN35SH	11.800 - 12.400	1,18 - 1,24	≥ 11,1	≥ 883	≥ 20	≥ 1.592	33 - 38	263 - 302	140

MATERIAL TYPE
Metallic Alloy

SURFACE PROTECTION
NiCuNi / Zinc / Epoxy / Passivation / Rilsan / Aluminum / Parylene

ORIENTATION
Axial / Diametral / Radial

MAGNETIZATION
Single or multiple poles on the functional surface



TEMPERATURE BEHAVIOR

Br TEMPERATURE COEFFICIENT*	% / °C	-0,10 / -0,12
HcJ TEMPERATURE COEFFICIENT*	% / °C	-0,40 / -0,78

*The temperature coefficients are nominal reference values only. They can vary for different temperatures and don't need to be linear.
**The maximum operating temperature is depending on the magnet shape, size and on the specific application.

PHYSICAL AND MECHANICAL TYPICAL PROPERTIES

CURIE TEMPERATURE	°C	>310
RECOIL PERMEABILITY	(μr)	1,05
SATURATION FIELD	kOe	30-60
ELECTRICAL RESISTIVITY	Ωm	$150x10^{-8}$
COMPRESSIVE STRENGTH	N/mm^2	~ 1.050
DENSITY	g/cm^3	~ 7,5
FLEXURAL STRENGTH	N/mm^2	250
TENSILE STRENGTH	N/mm^2	75
VICKERS HARDNESS	HV	~ 600
YOUNG'S MODULUS	N/mm^2	$160x10^3$
SPECIFIC HEAT	kcal/kg/°C	0,12
THERMAL CONDUCTIVITY	kcal/m/hr/°C	~ 7,7
THERMAL EXPANSION COEF ⊥ c	10^{-6}/ °C	-1,5
THERMAL EXPANSION COEF // c	10^{-6}/ °C	5

CIBAS GROUP