https://github.com/AlokPatel99/Inductor-Design-Using-Neural-Network

ECSE 549 – ESED

Project Report

Alok Patel – 260954024 and Abdulquadri Banuso – 260669341, McGill University

Abstract – **Due to evolution of the modern industry, systems are becoming more complex, and hence engineering design process has become a complex task, and hence the idea of a design assistant that can predict design variables based on knowledge base is required. In the given report, the documentation of the design project for the development of the C-core Inductor is presented which is based on the knowledge base. Our team goal was to design the Neural Network (NN) which can give the dimensions and other important parameters for the Inductor Design, according to the user requirements. The inputs and outputs of the NN are kept fixed, and the inputs to the NN will be from JESS system, and output of the NN which are the dimensions and other important parameters of the design is returned to the JESS for the constraint and the user satisfaction.**

I. Introduction

In this design project, the C-core inductor is designed, where first the background research was conducted on how the inductor can be designed. It was found that for the basic inductor design we can use the reluctance as the base for the inductance calculations and using the methodology and the set of equations provided in the MEC sheet by Prof. David Lowther, the calculations for the Inductance can be done. The inputs and outputs are defined for the network and then the data set were created. The total data set of around 5000 data was created for the training of the NN. For the creation of the dataset, various constraints were defined for the parameters of the Inductance. Then the Neural Network and its design is studied and explained. The Neural Network is then trained and tested based on the dataset created, and also it was tested for the output of JESS, which is the Input to our system. And the design parameters are estimated by the Neural Network. Let's first see the basis of the Inductor and its background search.

A. Inductor

Inductor [7] is the device where the energy is stored in the form of the magnetic field when the current is flown through it. It consists of an insulated wire wound into a coil, and when the current is flown through the coil changes, the time-varying magnetic field induces an electromotive force (emf) or Voltage in the conductor, described by Faraday's law of induction. The SI unit of the Inductor is Henry (H), and majorly it is measured in mH or μH. It has many applications such as filters in the filter circuits which is used for the tuning of the wave which consist of the harmonics which are not necessary for the system, as it only creates losses. It is also used in the RLC circuits for resonance which can be helpful for electronic applications, and it has also shown great possibilities for the efficient power electronic circuits and wireless charger. Also, the high valued inductors and high energy storage inductors which are known as reactors are used in Power system which are connected in shunts at the receiving side for the low loading conditions in order to mitigate the effect of Ferranti Effect [8]. Lastly, the major applications for the Inductors now a days are for the compensation systems in Flexible AC Transmission system. However, for this later applications inductor of high energy storage is required, and thus more complex inductance design is needed to be implement. For this project, the small inductance design is only taken into consideration, because of the various constraints. And for this simple small inductor which is useful for small electronic circuits the most basic equation that is useful is as shown below, which uses reluctance of the inductance to calculate the Inductance:

$$L = \frac{N\phi}{I}$$

Fig-1 shows the various shape of Inductances that are available in the market, it consists of the toroidal shape, cylindrical shape, ring shape, etc. For this project, the main concern is to design the C-core inductance.

https://github.com/AlokPatel99/Inductor-Design-Using-Neural-Network

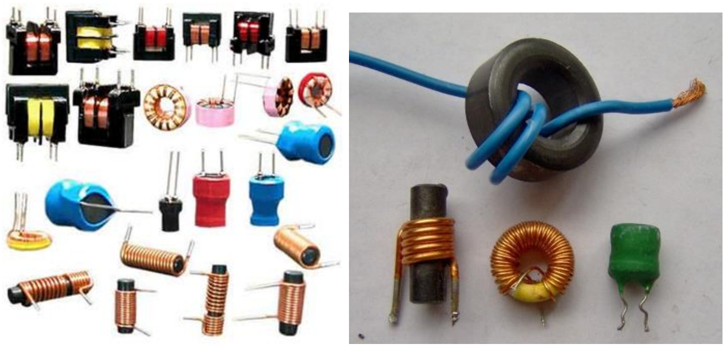


Fig-1: Types of Inductors

B. Background Search

The most basic method for the design of the Inductor is to go through the reluctance at the different areas of the core and calculating the total reluctance of the Inductor and then Flux and using the equation shown in the previous section, calculating the Inductance [1,2,4]. Reference-1 and 2 are the base for this design, where the former one provides the basic understanding of the design and the later one provides the basic tool for the parameters to get the equations for the data creation.

II. Design

For the overall design of the system, the system design starts with the decision on the equations to use for the data creation and generalize the NN with the equations using the reverse engineering process. Then the constraints are decided to satisfy the user requirements (according to the application), and the design parameters values actuality. For the detailed design and the rules setting for the JESS system (which the team implemented on the Python using CLIPS) can be understand by looking at the report of the Team-1 whose goal was to develop the JESS system which suggest user the input requirement for the NN if the user is not aware about the input parameters for the NN. Then using MATLAB, the data for the training of the Neural network is created and the NN is trained and tested based on that dataset. Lastly, the NN is tested on the input, which is provided by the JESS system, and then the output of the NN is send to the JESS (or python) system, as feedback and then the JESS system checks whether the parameters are in constraint range or not and checks the user satisfaction as well. Fig-2 shows the flow chart of the whole system.

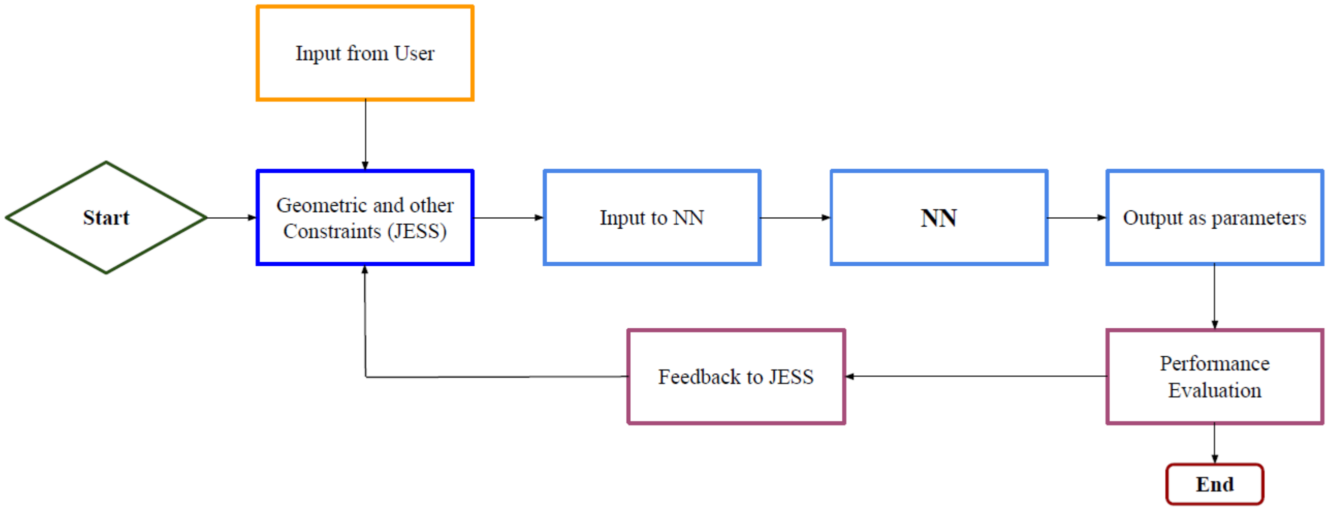


Fig-2: Flow Chart of the system

https://github.com/AlokPatel99/Inductor-Design-Using-Neural-Network

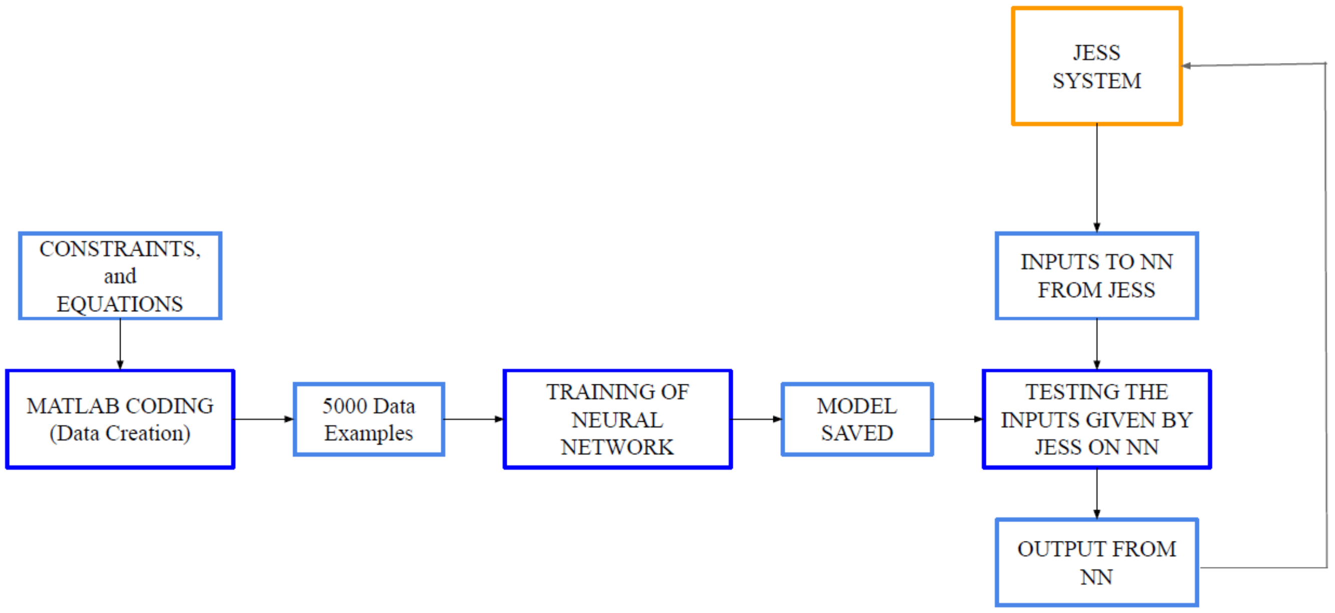

JESS SYSTEM
CONSTRAINTS, and EQUATIONS
INPUTS TO NN FROM JESS
MATLAB CODING (Data Creation)
5000 Data Examples
TRAINING OF NEURAL NETWORK
MODEL SAVED
TESTING THE INPUTS GIVEN BY JESS ON NN
OUTPUT FROM NN

Fig-3: Our system Design flow and its workflow

Let us look at the detailed system design of the NN, for the design of the C-core Inductor, which is shown in the Fig-3. Each block is explained in details and their function during designing and while testing.

1) Constraints and Equations

To create the dataset, we first need to decide the equations that are useful for the data creation. The data set are formed using the equations, and as the range of the parameter increases, in order to generalize the data and equations properly, we require more data as the range of the parameter (constraint range) increases. And to generalize data properly for large constraint range, we need more data and more complex NN. Thus, here to design for the small inductors we need small dataset. We will look at all the constraints and equations in details in the section of Data Creation.

2) Data Creation

For the training and validation of the NN, we need to create the dataset. Initially in the code some of the parameters are randomly generated, and then using the equations, we derive other parameters and then the whole dataset is configured into csv file. In total we generated different number of the examples, and we trained and tested the system on that.

3) Training of NN

For the training of the network, we use the Deep Learning toolbox provided by the MATLAB [9]. The dataset is divided into the training, validation, and the testing dataset, and then using it the performance of the NN is tested, and then the trained model is saved on the local storage, which will be used later for generating the design parameters according to the user requirements.

4) Testing NN on JESS Output

Once the NN model is saved on the local file, it can be loaded back into the program and it is tested on JESS data using the interface through CSV file. Then again, the outputs of the NN are stored into the CSV file, and then it is accessed back by the JESS system for the user and system constraints satisfaction.

III. DATA CREATION

For the training of the NN, we need the data set whose size is dependent on the number of input and outputs and as well as the constraint range. So, for the data we kept it to 5000 data, which could be high number for the small constraint range and small number of inputs and outputs, however, it was able to give us the good performance of the NN, which will be seen in the Neural Network section. So, for the data creation the inputs and the outputs are to be decided, which is nothing but the Client requirements. And thus, the Inputs and Outputs of the Neural Network with its constraints is as shown below:

1) ***Inputs***

Inductance required: [0.3 μH, 1.5 mH]
Cross sectional area of the core: [0.25, 0.6] x $10^{-4}\ m^2$
Space area to keep the inductance: [4, 30.25] x $10^{-4}\ m^2$
Core Material: {'Silicon Steel', 'Ferrite'} or {0, 1}
Max current capability: [0, 3.7] Amp

These are the inputs which the user enters, and if the user is unaware of some of the inputs, then the JESS system will suggest the values for this. Here, the inductance is for small values, however, if someone wants to develop the high valued inductance the design parameters constraints can be changed, and the type of the Inductor shape can be changed. And this space area is nothing but the area available for the user to keep the inductor in their circuit. We can additionally add the component in the code where if the user says that they have infinite space area to put (which may be possible), then just keep the if condition where if the user enter the area greater than 30.25 cm^2 then the code will randomly take the area in between this range, but as the JESS requirement was not there, we did not keep this feature in our code. Cross sectional area of the core will decide the dimension of the core, and finally the current rating will say the area of the winding. In the code where the NN is tested, feature can be added which detects the area of the winding and decide the gauge type, but that is the important part that can be added into the JESS system.

2) ***Outputs***

Cost of manufacturing the inductor
Number of turns: [5, 100]
Cross-sectional area of the cable: [1.2, 5.3] x $10^{-6}\ m^2$ (16-to-21-gauge cables)
Height of the core: [0.02, 0.05] m
Width of the core: [0.02, 0.05] m
Depth of the core: [0.005, 0.0077] m
Air gap Length: [0.001, 0.005] m

Cost of manufacturing is doing nothing but calculating the weight of the core and winding material and thus here the cost of CAD/kg is kept constant, but however, as it is varying on the daily basis, the cost as the output can be removed from the NN, and it can be added into the JESS (or Python), where simply it uses the simple equation to calculate the volume to weight to the cost.

The important assumptions that are taken into consideration here is as shown below:

- Square Core Cross Section
- Circular Cable is considered
- Fixed cost for the core material and copper material is taken.

Now, in terms of the equations for the calculations using the equations, what been done is that various parameters are uniformly randomly generated in the constraint range and then they are stored in the variables, and then reluctances are found, and other important variables are found, and at last using the inductance equation the inductance is found and it is also stored, and finally whole csv file of the input and the output data for 5000 examples is created. The code of the same can be found in the file named as *Data_Gen.m*. In the Dataset, as we are having five different type of cable gauge so in total, we will have 1000 dataset for each type, and for two different types of core material, we have 2500 dataset for each type of core material. And in the Fig-4 the C-core inductor with the given dimension is also shown.

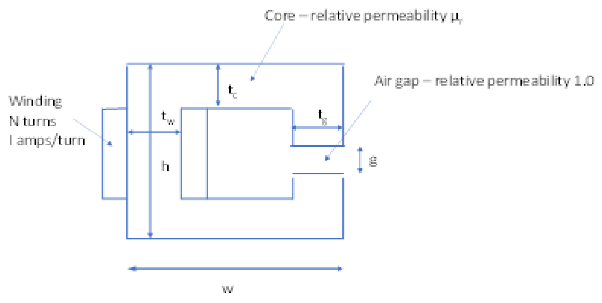

Core – relative permeability μ_r
Air gap – relative permeability 1.0
Winding
N turns
I amps/turn
t_c
t_w
t_g
h
g
w

Fig-4: C-core Inductor

Below are all the important equations taken for consideration while developing the data set. And using this whole data set is created.

→ $A_{core} = t_g \times d$ (Here, $t_d = t_c = t_g = t_w = d$)

→ Back of core reluctance (BR) $(H^{-1}) = \dfrac{h - 2^* t_c}{2^* \mu_r {}^* A_{back} {}^* \mu_o}$

→ Top left corner reluctance (TLCR) $= (t_c/(2^*\mu_r{}^* A_{back}{}^*\mu_o)) + (t_w/(2^* A_{top}{}^*\mu_r{}^*\mu_o))$

→ Top of core Reluctance (TCR) $= (w - t_w - t_g)/(A_{top}{}^*\mu_r{}^*\mu_o)$

→ Top right corner Reluctance (TRCR) $= (t_g/(2^* A_{top}{}^*\mu_r{}^*\mu_o)) + (t_c/(2^* A_{pole}{}^*\mu_r{}^*\mu_o))$

→ Pole reluctance (PR) $= (h - l_g - 2^* t_c)/(2^* A_{pole}{}^*\mu_r{}^*\mu_o)$

→ Air gap Reluctance (AGR) $= l_g/(2^* A_{airgap}{}^*\mu_o)$

⇒ Thus,
Total Reluctance (TR) $= BCR + TLCR + TCR + TRCR + PR + AGR$

$$\text{Flux} = \frac{mmf}{TR} \Rightarrow \text{Inductance (L)} = \frac{N^* \text{Flux}}{I}.$$

$mmf = NI$.

→ volume of core $= h \times d \times A_{core} + 2 \times (w - t_w) \times t_c \times A_{core} + (h - 2 \times t_c - l_g) \times t_g \times A_{core}$

↳ using density and cost (A0/kg → Total cost is calculated.

→ Volume of winding $= (\pi \times d_{new} \times N) \times A_{winding}$

where, $d_{new} = 2d + t_w$
↳ of wire.

→ Additionally, 10% of manufacturing cost is considered.

III. Neural network

To design the neural network the Matlab Deep Learning Toolbox [9] was used, as it is easy to use and understand, allows experimentation with different neural structures, excellent performance analysis of neural networks, visually pleasing models and integration with its SIMULINK interface to allow neural network implementation with dynamical systems.

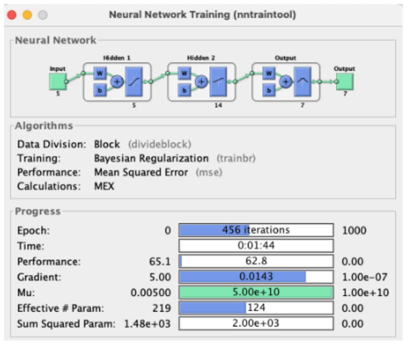
A. Modified Neural Network Architecture

The feed-forward network is created using the MATLAB Deep Learning Toolbox with the command **feedforwardnet** that creates a neural network object with an input layer of 10 neurons, input and output size is 0 waiting to be configured/trained to the right size depending on the data, in our case this will be 5 inputs and 7 outputs (Air Gap, Height ,Width , Depth , Number of Turns, Cost , Wire Area). For optimal design, we used the mean squared error as design variable in delivering a neural network to evaluate performance and deliver good sizing parameters of the C-Core Inductor.

The data is split in 40/30/30 format in the train, validate and test ratios for the neural network. and trained. The neural network has a 94% accuracy (6% mean squared error accuracy), which is expected as the network designed with a single layer of 10 neurons and hence training computation is not as extensive as in a multilayer network with non-linear transfer functions. The network is optimized with the default option **trainlm** which is a Levenberg-Marquardt training algorithm, with a **mean squared error** performance algorithm.

The feedforward network is modified and optimized by adjusting the number of hidden layers, neuron in each layer, layer transfer functions, and training algorithm of the network in an effort to get an optimum design. The choice of this parameters is made arbitrarily, and performance is considerably better than the single layer model of the network developed although with more time taken to train the model.

Modifications was made to the existing neural network architecture, by training different networks with different sizes, layer transfer function mixes, training algorithm to observe performance with different arbitrarily selected parameters of the networks. The table below shows the results of training of the different network architectures, from this result we can deduced that effective of a network is based on its training algorithm and its layer transfer functions.

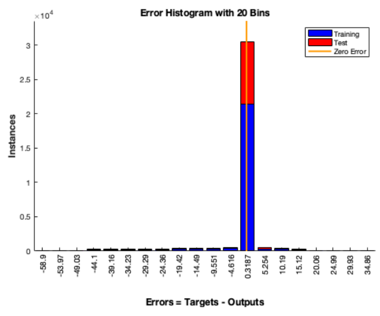


Algorithms

Data Division:	Block (divideblock)
Training:	Bayesian Regularization (trainbr)
Performance:	Mean Squared Error (mse)
Calculations:	MEX

Progress

Epoch:	0	456 iterations	1000
Time:		0:01:44	
Performance:	65.1	62.8	0.00
Gradient:	5.00	0.0143	1.00e-07
Mu:	0.00500	5.00e+10	1.00e+10
Effective # Param:	219	124	0.00
Sum Squared Param:	1.48e+03	2.00e+03	0.00



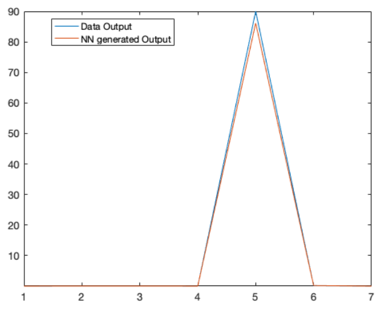


Fig-5: System architecture and performance measure

Table-1: Results of the NN performance

Training Sessions Results				
Layers	**Neurons**	**Layer Transfer Function**	**Network Training Algorithm**	**Performance**
2 [5;7]	23	logsig & pure-lin	trainlm	94%
3 [5;7;7]	39	tansig, radbas & purelin	trainbr	98%
3[5;14;7]	33	tansig, tansig & purelin	trainbr	98%

https://github.com/AlokPatel99/Inductor-Design-Using-Neural-Network

In Fig-5 the architecture is presented in the first image along with the snapshot of the in between training, and other images are the error measurement of the test data. In Fig-6 the results of the different NN structure results are presented. Where the last two provides the best performance.

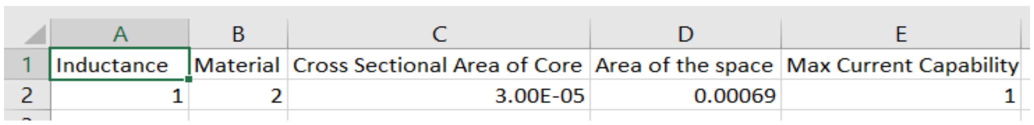
IV. INTERFACE AND TESTING

In order to test the NN for the design input by the user which is actually coming from the JESS system, so we need to interface the JESS and MATLAB system, where we don't have any particular strategy to interface both the system, so we are using the csv files as the medium of transmission. So, when the JESS give output in the csv file, MATLAB code reads the file which will be named as *NN_Input.csv* and using the loaded NN named as *c_core_net_Final.mat*, then the output of the NN is again stored in another csv file named as *NN_Output.csv*. And this file is again accessed by the JESS code to check whether the output of the NN is lying in the constraint range, and it is checked for the user satisfaction.

Here, if the engineer wants to develop the whole system into one, then he/she can use the scikit-learn library or PyTorch Library for the NN development and then use clipspy for the rule development in the python. Using this, we can develop the whole environment into the single software console and will just interface between each other using simple variables in code.

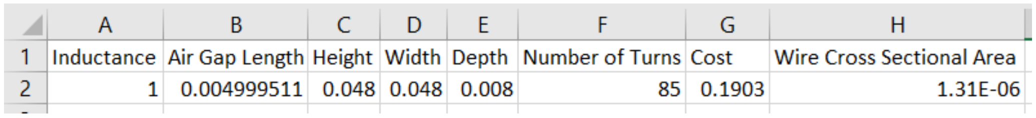
The result of the Neural Network for the given Input by the JESS system is as shown below:

Input:



	A	B	C	D	E
1	Inductance	Material	Cross Sectional Area of Core	Area of the space	Max Current Capability
2	1	2	3.00E-05	0.00069	1

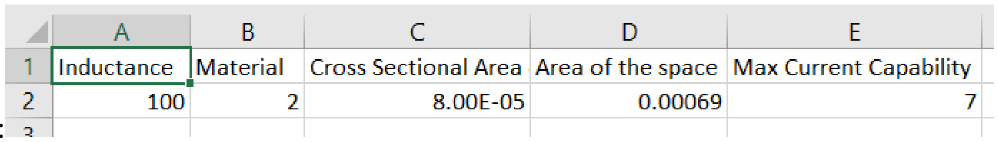
Output:



	A	B	C	D	E	F	G	H
1	Inductance	Air Gap Length	Height	Width	Depth	Number of Turns	Cost	Wire Cross Sectional Area
2	1	0.004999511	0.048	0.048	0.008	85	0.1903	1.31E-06

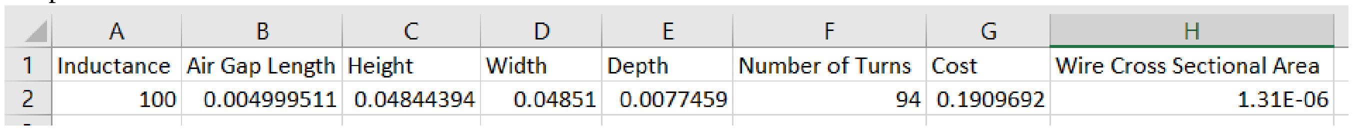
When the system is provided with the inputs that are out of the constraint range, and which is outside the dataset range of the NN, then the results can be uneven and not true, as for the NN dataset which is trained on, so actually not been able to go out of its bound on the results and giving the outputs on which, it is trained on.

Input:



	A	B	C	D	E
1	Inductance	Material	Cross Sectional Area	Area of the space	Max Current Capability
2	100	2	8.00E-05	0.00069	7

Output:



	A	B	C	D	E	F	G	H
1	Inductance	Air Gap Length	Height	Width	Depth	Number of Turns	Cost	Wire Cross Sectional Area
2	100	0.004999511	0.04844394	0.04851	0.0077459	94	0.1909692	1.31E-06

V. WORK REMAINING AND FUTURE ASPECTS

As we are giving feedback to the JESS system back for the user and constraint satisfaction, currently, JESS is not returning back new inputs for the NN, for new outputs by tuning some parameters. So, in the remaining work we can make the loop between the JESS and NN recursive. This can be easily achieved if the Rules and the NN is made itself in the python console. Additionally, applying the Multi-objective optimization for this design process to examine tradeoff between design variables for the optimum design. Other future consideration to take into account is the copper loss into the consideration, frequency into account, Cable cross section type, etc. Also, we can look at the performance measure of the Inductor automatically optimizing the design outputs of the NN. Lastly, we can also do normalization of the dataset in order to increase the accuracy of the NN. Future implementation of the neural network assistant will need to take into account non-linearities like copper losses, real time permeability changes which in turn will need implementation of a more complex neural network to accurately predict sizing parameters for the C-Core Inductor.

To make it into real system a month of full-time work is required, by considering other parameters and increasing the range of the parameters. And the team of 2 to 3 people is enough to complete this work. By making some of the changes to input-outputs and the dataset, NN can generalize most of the dataset that we have if we develop the complex NN structure.

IV. DISCUSSION AND CONCLUSION

In this project, the design system is developed, using the NN, so first, we decided the equations and the inputs and outputs of the NN. And then the constraints are decided for each input and outputs. Overall, the overall efficiency achieved is 98% in the testing dataset. As we are working on the small inductor design, we have small number of input and output, and less complex NN structure. However, if we want to design the more complex inductors, we need more input and outputs, and more complex network.

In conclusion, we developed the NN which is trained on the dataset generated using the equations we have, and we then tested the NN using the JESS output. And it can generate the good results for the given constraint range, but for the design of the more complex inductors, we need more complex NN and more inputs & outputs and more data set.

VII. REFERENCES

[1] https://ecee.colorado.edu/~ecen5797/course_material/Ch14slides.pdf

[2] MEC for C-Core Inductance.xlsx by Prof. David Lowther

[3]https://www.allaboutcircuits.com/technical-articles/basic-inductor-design-constraints/

[4] http://ecee.colorado.edu/~ecen4517/materials/Inductor.pdf

[5] J. W. McLean, "Inductor design using amorphous metal C-cores," in IEEE Circuits and Devices Magazine, vol. 12, no. 5, pp. 26-30, Sept. 1996, doi: 10.1109/101.537353.

[6] T. Guillod, P. Papamanolis and J. W. Kolar, "Artificial Neural Network (ANN) Based Fast and Accurate Inductor Modeling and Design," in IEEE Open Journal of Power Electronics, vol. 1, pp. 284-299, 2020, doi: 10.1109/OJPEL.2020.3012777.

[7]https://en.wikipedia.org/wiki/Inductor#:~:text=An%20inductor%2C%20also%20called%20a,wire%20wound%20into%20a%20coil.

[8] https://en.wikipedia.org/wiki/Ferranti_effect

[9] https://www.mathworks.com/help/deeplearning/ug/deep-learning-in-matlab.html