LLP Research Report – Data Down Sampling and FFT Applied to Time Series Forecasting

Anthony Zhai, Thomas Guillod, Diego Serrano, Haoran Li, Shukai Wang, Minjie Chen
Princeton University

Abstract—**Modeling core loss is a challenging but important process in designing power electronics circuits. To date, an analytical model has yet to be developed. Thus, machine learning is applied to a rich dataset of magnetic cores to predict core loss using neural networks. A physical interpretation of the problem is developed using the Fast Fourier Transform (FFT) algorithm to encode the frequency domain features of the time series data, which are then passed into a simple feedforward neural network (NN). In contrast to more complex sequence to sequence models such as Long Short Term Memory networks (LSTMs), this FFT NN model drastically reduces the number of parameters in the model as well as the training time, while obtaining similar or better results on core loss predictions. This method is further explored with hyperparameter tuning, a one-hot encoded feature for material for an all-in-one model, and a custom loss function for computing core loss during backpropagation. Additionally, methods to reduce redundancy in the magnetic core dataset are explored to decrease the time and costs for data collection.**

I. Introduction

Magnetic core loss is a complex variable that can be subject to time-dependent changes based on several variables including DC bias, temperature, frequency, and flux density. Certain models for core loss such as the Generalized Steinmetz Equation (GSE) and the improved Generalized Steinmetz Equation (iGSE) have been developed [1], [2]. Although these equations can provide simple and quick models, they often lack the complexity needed to capture the varying effects of all the variables that can be important to core loss.

Thus, neural networks can be used to learn non-linear mappings between an input signal and an output signal, and subsequently, a core loss calculation. Considering the nature of the time series data necessary for calculating core loss, including the flux density (B) and magnetic field strength (H), one may think that recurrent neural networks such as LSTMs and natural language processing models such as Transformers may be necessary to create such mappings. However, a novel structure using the Fast Fourier Transform to encode the frequency domain features of flux density and magnetic field strength is explored in a simple feedforward neural network structure, reducing complexity and training time while applying a physical interpretation of the problem to the machine learning methods.

II. FFT Neural Network V1

The first neural network was developed using the PyTorch machine learning library. The low-level nature of the framework allowed for easy integration of custom loss functions and training loops.

A. Dataset

The MagNet magnetic core dataset [3] was used for training and testing the neural network outlined in this report. For the first iteration of the FFT NN (V1), an 18,000-point dataset for the N87 material was used with B field, H field, and frequency data. The processed data for the B and H fields consisted of a single-cycle signal of 100 evenly spaced points.

B. Architecture

During model training, the data is first normalized to have a range between -1 and 1 using the scikit-learn Python library [4]. Neither standardization nor batch normalization are used to preserve the uneven distribution of the data. The H Field target values are not normalized. After normalization, the FFT coefficients for the B field and the H field are calculated as the input and target values for the neural network. After FFT, the n complex-valued frequency coefficients for the B field are compressed into a single vector of the n real coefficients followed by the n imaginary coefficients. The FFT coefficients for B are then concatenated with the frequency and are passed into the neural network with two dense layers and one output layer. The loss is computed using the logits and the frequency domain signal for H. The hyperparameters used were 64 neurons for each hidden layer, a batch size of 64, a learning rate of 2e-4, and the Adam optimizer.

C. Preliminary Results

First, preliminary results for predicting the time domain H field signal are shown, with different hyperparameters tested and evaluated.

Three model architectures are compared in Fig. 1. Interestingly, the reduction in parameters from 160,000 to 33,000 (200 neurons per layer to 64 neurons per layer) shows little reduction in accuracy, where the 160,000 parameter network experiences considerable overfitting in its perfect matching of the noise of the signal. A more substantial reduction in accuracy can be seen from downsizing the neural network to 32 neurons per layer, where the model captures the shape of the signal but not the more specific fluctuations in the signal.

D. Filtering

Often, many of the measured signals contain high frequency oscillations that affect the smoothness of the predicted H field. These oscillations may be an artifact of imperfect measurement systems rather than properties of the magnetic core itself. Thus, filtering methods are analyzed.

The main filtering method analyzed consists of zeroing out the low amplitude frequency coefficients. Out of a 100-point sample, 50 unique FFT coefficients are obtained, and the smallest coefficients are zeroed. Differing number of coefficients to be zeroed were tested and compared using the relative error metric calculated as below. After experimenting with varying numbers of coefficients from 0 to 35 in increments of 5, it was found that zeroing out 10 of the 50 frequency coefficients yielded the lowest relative error of 8.24%, as shown in Fig. 2. Relative error is defined as:

$$Rel\ Err = \frac{RMS(H_{pred} - H_{meas})}{RMS(H_{meas})}$$

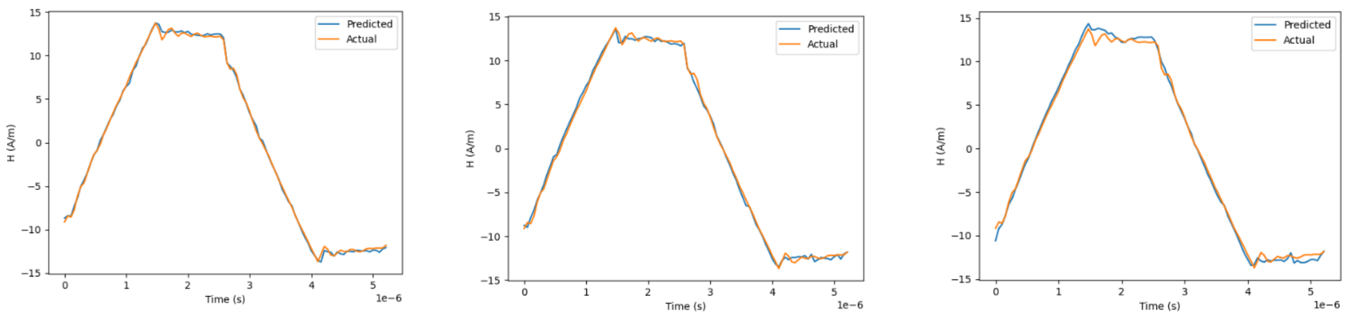

Fig. 1. Comparison of neural networks trained with three different architectures with the same H field signal. Left: 200 neurons in each of the three hidden layers. Middle: 64 neurons in each of the hidden layers. Right: 32 neurons in each of the hidden layers.

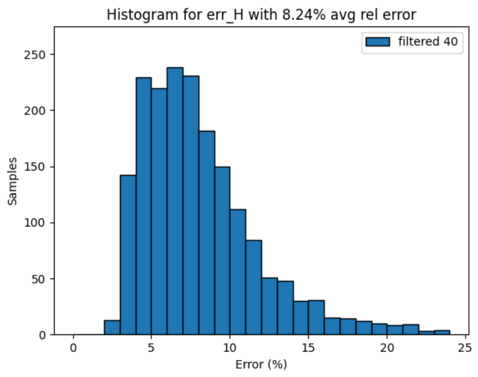

Fig. 2. Histogram for relative error of filtering with 40 frequencies.

E. *Custom Loss Function*

The goal of training a neural network to map a B field signal to an H field signal is to calculate core loss. Thus, a custom loss function that incorporates this core loss calculation is explored to better inform the loss value calculation for the backpropagation of the neural network.

First, the model takes the input B field frequency domain data and predicts the H field frequency coefficients. However, rather than directly using the H field frequency coefficients as logits for calculating the loss of the neural network, an additional matrix multiplication step is added that computes the energy loss based on the B and H fields. Energy loss is calculated rather than core loss density as a more stable value for backpropagation.

The energy loss is calculated using the equation below. B and H represent the B and H fields in the frequency domain, *w* represents the energy loss per cycle, *p* represents the power loss, and *Re* acts as an operator for taking only real values of the operand.

$$w = \frac{p}{f} = \frac{1}{2}\mathcal{R}e((2\pi jnf)BH^*)$$

F. *Hyperparameter Tuning*

Hyperparameter tuning studies were conducted to fully optimize the neural network performance. The Optuna framework was used because of its easy integration with the PyTorch framework. The following variables were considered for tuning: number of layers, number of neurons in each layer, optimizer, learning rate, and batch size.

For tuning the neural network, an objective function was defined which returned the metric in question to be minimized during the running of the tuning study. To best capture the model's ability to generalize to new data, the validation loss for the last epoch was used as the metric to be minimized. In addition to creating an objective function, pruning was also implemented using a pruning algorithm which stops a trial if it is performing worse than the median of all previous trials at the same epoch step. Finally, parallelization was utilized to run multiple trials in parallel, speeding up the tuning process by up to 5 times.

Using pruning and sequential model-based optimization, Optuna's algorithm to select the optimal hyperparameters, the hyperparameter search is configured to run quickly and efficiently. The informed parameter search method is illustrated in Fig. 3, where the tuner concentrates more on regions in the parameter space that yield good results.

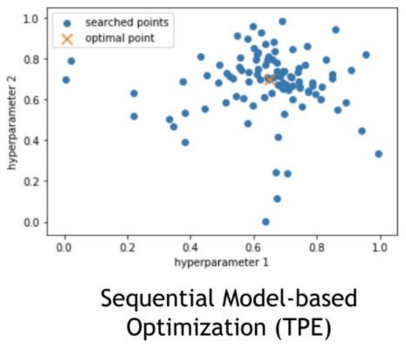

Fig. 3. Illustration of the method used to search the hyperparameter space in Optuna, using sequential model-based optimization.

To obtain an even distribution for sampling the number of neurons in each layer, a special algorithm was developed. First, the total number of neurons in the network is sampled from 80 to 160. Then, the number of layers is sampled from 2 to 3. For each layer, a fraction is sampled between 1/3 and 1, and then scaled for all fractions to add to 1. Next, the number of neurons in each layer is determined by multiplying the fraction sampled for each layer with the total number of neurons.

In total, 600 trials were run. The best result obtained was a validation loss of 3.5e-3. However, it is noteworthy that most of the trials obtained values within a range of 3.5e-3 to 6e-3, demonstrating the stability of the network and its robustness without regards to optimized hyperparameters.

III. FFT Neural Network V2

To take full advantage of GPUs for training the neural network, a more optimized pipeline was developed using the TensorFlow Keras framework [5].

A. Dataset

A new dataset was used for the 2nd iteration of the FFT NN (V2). The new dataset consisted of several magnetic core materials (including N87, 3C90, and 3C94) with B field, H field, H bias, frequency, and temperature data. For the purposes of testing, the N87 dataset was used which contained 134,000 data points of 128 time steps each for the B and H field time-domain signals.

B. Architecture

The model consists of a preprocessing component, a custom Keras layer to perform FFT, two dense layers, and one output layer.

First, the four input variables of B field, H bias, frequency, and temperature are concatenated into a single vector to be used in training. The preprocessing of the data uses the same normalization method from Section II, in which the data is normalized between -1 and 1 along the 1st axis (for each feature individually). The preprocessed data is then passed to the FFT layer, where the B field is extracted from the concatenated vector and is transferred to the frequency domain using the FFT algorithm. The frequency domain signal is concatenated back into a feature vector which is then passed through the two dense layers and the final output layer. The same FFT and concatenation algorithm is used for computing the frequency coefficients of the H field as target values, to ensure consistency in computing the loss of the neural network.

This architecture allows for users to simply input raw data that then undergoes processing entirely within the model training pipeline, rather than requiring preprocessing in separate pipelines before being passed for training into the model. Thus, the more optimized TensorFlow model pipeline saves time while reducing the complexity of the training process.

C. DC Bias

Although the new dataset contains DC bias for the H field, whether the neural network performs better with or without the bias added to H requires further experimentation.

A model trained on the full N87 dataset with zero DC bias added to the H field signals is compared to a model trained on the same dataset with the DC bias added to the H field signals. Surprisingly, the performance of the neural network shows a significant increase with the addition of DC bias. Since the only difference between these two datasets is the addition of a constant in one of the features, it is unclear what the underlying reason behind this difference is – since the DC bias is a constant, the normalization of the data should cancel out any effect of adding a constant to the signal. Unfortunately, further experimentation was not possible due to time constraints and is left as a research question for future studies.

D. All-in-One Model

An all-in-one model is tested to assess the viability of a general FFT NN that can predict core loss across any magnetic core material.

A simple addition of a one-hot encoded "material" feature is added to the original FFT NN. This vector is generated as a list of 0s and 1s, where the index of the material with respect to the rest of the materials is encoded as a 1 while the rest of the indices (indicating the rest of the materials) are encoded as 0s. For example, for a dataset with N87 and 3C90, the material features would be [1, 0] and [0, 1] respectively.

The all-in-one model was trained on a dataset containing the data for the N87, 3C90, and 3C94 materials. The same hyperparameters were used in training on the full dataset.

Two graphs are shown below, illustrating the predicted vs. actual H field signal and real and imaginary FFT coefficients. In this example, the model performs exceptionally well on triangular waveforms and creates a nearly perfect match. It is important to note that such high accuracy across materials could stem from the similarity between the three materials used for training, where N87, 3C90, and 3C94 can often be used interchangeably as magnetic cores and do not exhibit significantly different behavior under similar conditions.

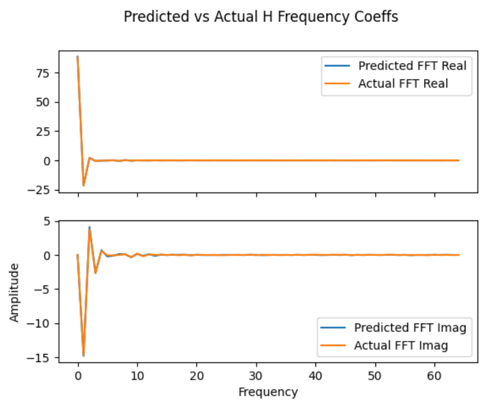

Fig. 4. Predicted vs. actual frequency coefficients for the H field. Top: real coefficients; bottom: imaginary coefficients.

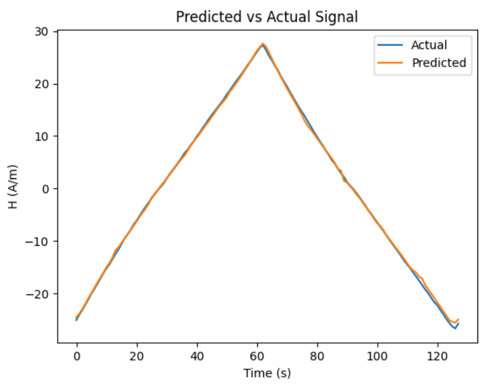

Fig. 5. Predicted vs. actual H field signal in the time domain.

IV. Data Down Sampling

In its current configuration, the automated data collection system in the MagNet research lab takes approximately one week to collect all data for a single material. More time is taken to fully postprocess the data into interpretable and meaningful results. Although obtaining a large amount of data can be advantageous to analyze the effects of certain variables, it can be costly, time-consuming, and impractical when considering the numerous materials being used in industry.

Thus, two variables (frequency and flux density) from the full datasets for three magnetic core materials (N87, 3C90, and 3C94) are analyzed for assessing redundancies, the relative significance of each variable on predicting core loss, and the potential for reducing the parameter search space for data collection.

A. *Frequency*

In the three datasets, data is collected for 20 different frequencies. These frequencies are spaced evenly between 50 kHz and 450 kHz on a log scale, with 10 frequencies per decade. However, in the final measurements, the frequency values for each of the 20 different original frequencies are not perfectly equal, so preprocessing is required to match each frequency to its corresponding group.

An algorithm is developed that accomplishes this task by using a margin of 2 kHz to assess whether a given frequency is part of the same group. After sorting the frequency data points in the dataset, each frequency is iterated over to be assigned to a specific frequency group. If the frequency being iterated over is less than 2 kHz greater than the last frequency that was iterated over, then the algorithm classifies the two points as part of the same frequency group. However, if the difference is greater than 2 kHz, a new group is assigned to the higher frequency.

Next, 10 different data splits are used to train the model on each of the three datasets. These data splits are achieved by taking between two to twenty of all the frequencies (in increments of two). As a simple model for selecting the frequencies in each data split, the extremes (first and last frequencies) are selected with the rest of the frequencies chosen in even intervals between the extremes. This algorithm allows the model to interpolate the missing frequencies rather than extrapolate them, intuitively leading to more robust learning. A plot of certain data splits with down sampled frequencies are shown in Fig. 6. The complete pipeline for analyzing down sampled data is shown in the pseudocode in Algorithm 1.

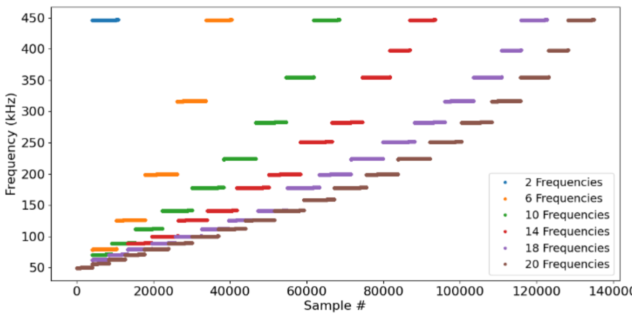

Fig. 6. Plot of different frequency groups selected for different down sampled splits of the data.

Trials were trained on the Princeton Adroit Computing Cluster, with the 10 data splits trained for each of the three materials. Metrics used for comparison were average cosine similarity and average relative error. The results are shown in Fig. 7 and Fig. 8. Cosine similarity is used as a metric to compare two vectors using the projection of the input vector on the target vector. The equation for cosine similarity is shown below.

$$k(x,y) = \frac{\boldsymbol{x}\boldsymbol{y}^T}{\| \boldsymbol{x} \| \| \boldsymbol{y} \|} \in [-1,1]$$

Algorithm 1: Data down sampling algorithm for a variable $\boldsymbol{w}$ to be down sampled.

```
Require: Variable $w$, data $X$, labels $y$, number of groups to select $n$
1. Splitting Phase;
Load data $X$ and sort by variable $w$.
Divide data points for $w$ into 20 log-scale intervals.
2. Selection Phase;
Select first group or interval.
Divide rest of groups into $n$ - 1 evenly spaced slices.
Select the last group from each of the $n$ - 1 evenly spaced slices.
3. Grouping Phase;
Set current group iterator variable to zero.
Initialize new dataset $df$.
for example $x$ in data do
  if variable $w$ for example $x$ is within interval of current group:
      if current group is in selected groups:
          Add example $x$ to new dataset $df$.
      end if
  else:
      Increment group iterator variable.
  end if
end for
Result: The resulting dataset $df$ will contain only $n$ groups of the variable $w$.
```

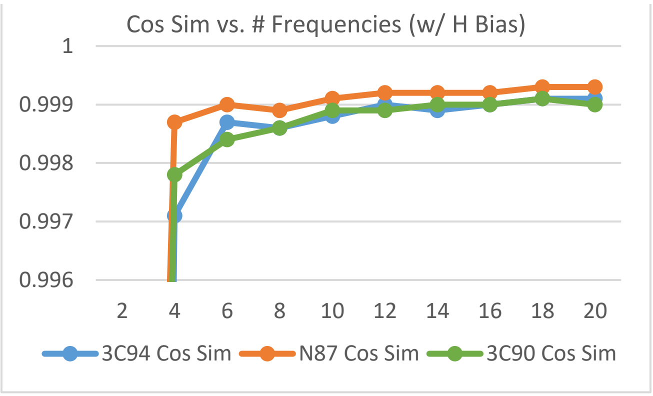

Fig. 7. Average cosine similarity of models trained vs. the number of frequencies the model trained with. All models are trained with data that has the H bias constant added to the H field signal.

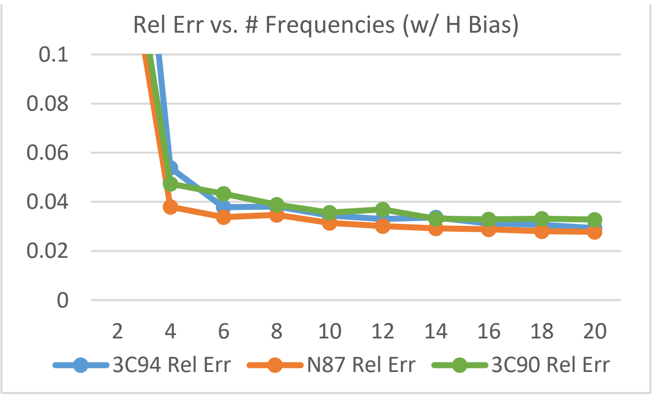

Fig. 8. Average relative error of models trained vs. the number of frequencies the model trained with. All models are trained with data that has the H bias constant added to the H field signal.

In both graphs, the first important observation is that every model with four or more frequencies sampled from the dataset performs at nearly the same level as training on the full dataset. Although minor improvements in performance are seen with more sampled frequencies, the difference is generally less than 1% for relative error and less than 0.001 for cosine similarity. This means that the parameter search space for frequency can be reduced significantly to only contain 20% to 30% of the original number of frequencies.

The second important observation is that the same trend is experienced across all three materials of N87, 3C90, and 3C94. The likely reason is that these three materials behave similarly within the same conditions. However, the consistency of models trained with 20% to 30% of the original number of frequencies shows that this observation is still significant for commonly used magnetic core materials.

B. *Peak Magnetic Flux Density*

Next, the peak magnetic flux density (peak value of the B field signal) is down sampled. The peak flux density points are much less evenly distributed than the frequency points, so a less precise algorithm is used to split the data.

First, the peak flux density is split into 20 evenly spaced log-scale intervals. Then for each of the 10 data splits, a certain number of these intervals are selected for training. Using the same methods as down sampling with frequency, a certain number of groups of peak flux density are selected in increments of two where the extremes are always selected. Although this does not provide directly interpretable results in terms of which peak flux densities can be discarded, this analysis gives an insight into the relative significance of peak flux density in modeling core loss and the extent to which peak flux density data points can be reduced. Data splits for peak flux density are shown in Fig. 9.

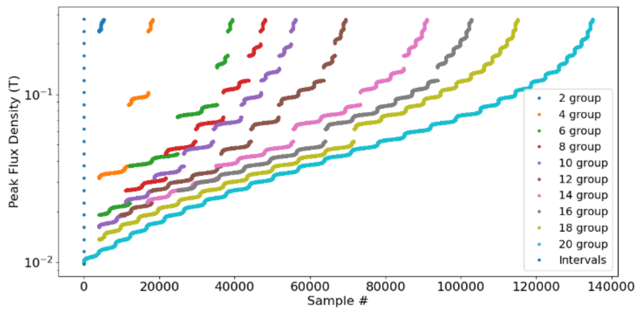

Fig. 9. Plot of different frequency groups selected for different down sampled splits of the data. Shown on a log scale.

The average cosine similarity and relative error for models trained on each data split are shown in Fig. 10 and Fig. 11. Similar to down sampling frequencies, down sampling peak flux density shows that only utilizing 20% to 30% of the original number of groups is adequate in training a robust neural network. For example, the model trained with 6 peak flux density groups from the N87 dataset obtained a cosine similarity value within 2e-4 of the model trained with 18 peak flux density groups. Although the relative error shows a greater amount of variation in model performance with respect to number of peak flux densities, most models fall within 1% relative error from 20% of the data to 100% of the data. Additionally, the behavior of these error metrics across different size datasets is very similar when compared across the three different materials.

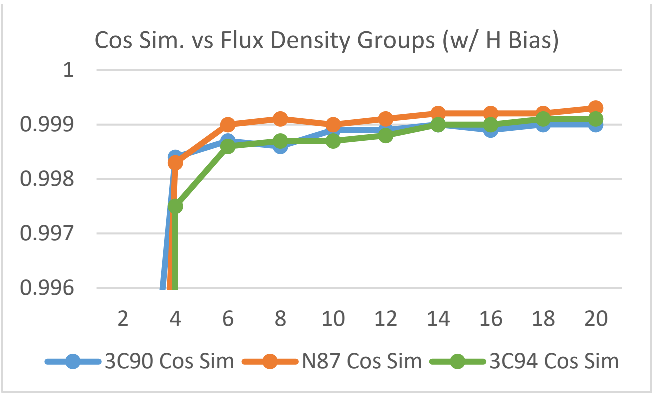

Fig. 10. Average cosine similarity of models trained vs. the number of flux density gruops the model trained with. All models are trained with data that has the H bias constant added to the H field signal.

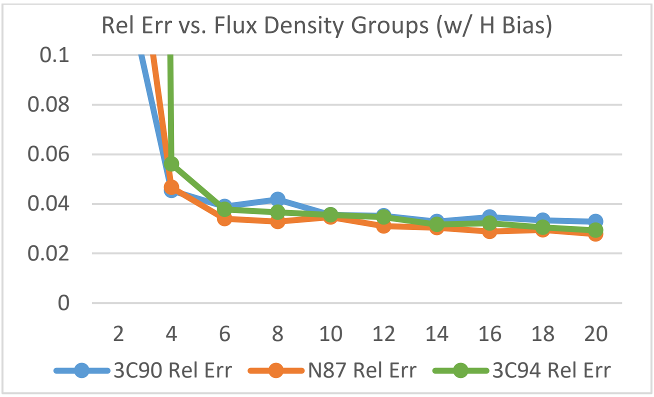

Fig. 11. Average relative error of models trained vs. the number of flux density groups the model trained with. All models are trained with data that has the H bias constant added to the H field signal.

C. *Comparison with DC Bias*

In congruence with the analysis of the effects of adding DC bias to the H field signal in Section III, model performance is compared across adding DC bias and not adding DC bias to the H field signal. The graphs of relative error and cosine similarity can be seen below in Fig. 12 and Fig. 13.

As described in Section III, adding H bias considerably improves the performance of the model. For instance, the model trained with the N87 dataset with H bias using just four frequencies outperforms the model trained with the N87 dataset without H bias using all 20 frequencies.

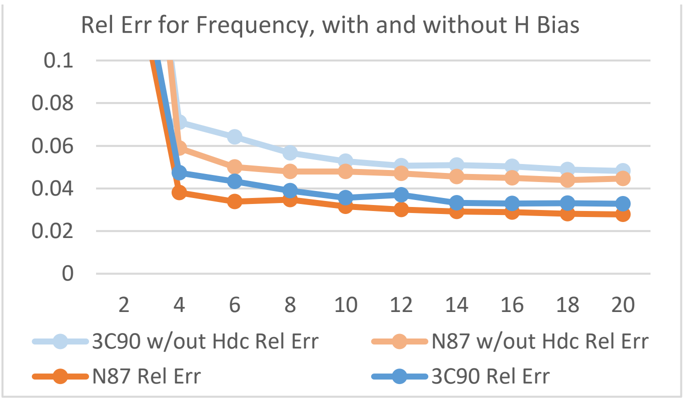

Fig. 12. Relative error for down sampled frequency data splits. Comparing N87 and 3C90 datasets with and without H bias added to the H field signal.

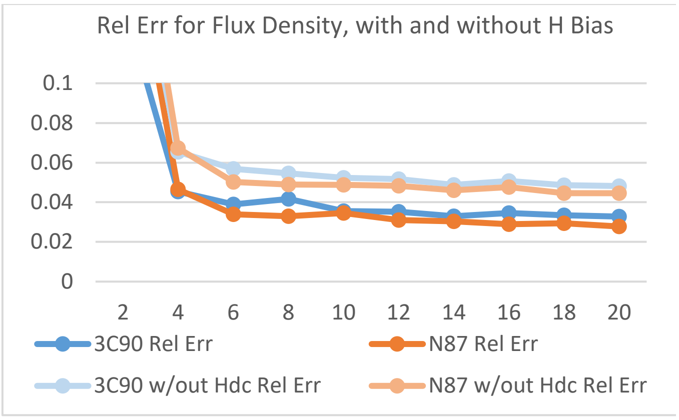

Fig. 13. Relative error for down sampled flux density data splits. Comparing N87 and 3C90 datasets with and without H bias added to the H field signal.

V. Conclusion

In this study, a novel neural network utilizing the Fast Fourier Transform algorithm is developed to model power magnetic core loss and data down sampling methods are explored to systematically assess the relative significance of certain variables in predicting core loss. A first iteration of the neural network is implemented in PyTorch and tuned with the Optuna library. Results are further improved using filtering methods that zero out the smallest frequency coefficients and a custom loss function that computes core loss as an extra logit to be used in backpropagation.

Next, a second iteration of the neural network is created using the Tensorflow Keras framework which implements data normalization and the Fast Fourier Transform into a streamlined training process. In addition to improved speed and simplicity, the second iteration also shows advantages with the possibility of adding the magnetic core material as an additional feature for an all-in-one neural network.

Results are assessed using the cosine similarity and relative error metrics as well as graphs comparing predicted vs. actual results for frequency coefficients and reconstructed H field signals.

Lastly, frequency and peak magnetic flux density are systematically down sampled and analyzed as variables for predicting core loss. Neural networks are trained using the different size down sampled datasets, and performance is compared using error metrics. Remarkably, model performance showed convergence using only 20% to 30% of the original dataset with similar behavior displayed across the three magnetic core material datasets that were compared. Although these results show that frequency and peak flux density are stable variables for predicting core loss, further experimentation is required to analyze different variables such as temperature and DC bias.

References

[1] J. Li, T. Abdallah and C. R. Sullivan, “Improved calculation of core loss with nonsinusoidal waveforms,” IEEE Industry Applications Conference Annual Meeting, Chicago, IL, USA, 2001, pp. 2203-2210, vol.4.

[2] J. Muhlethaler, J. Biela, J. W. Kolar and A. Ecklebe, “Improved core-Loss calculation for magnetic components employed in power electronic Systems,” IEEE Trans. on Power Electron., vol. 27, no. 2, pp. 964-973, Feb. 2012.

[3] H. Li, D. Serrano, T. Guillod, E. Dogariu, A. Nadler, S. Wang, M. Luo, V. Bansal, Y. Chen, C. R. Sullivan, and M. Chen, "MagNet: an Open-Source Database for Data-Driven Magnetic Core Loss Modeling," IEEE Applied Power Electronics Conference (APEC), Houston, 2022.

[4] Pedregosa et. al, “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, pp. 2825-2830, 2011.

[5] Chollet, F. & others, 2015. Keras. Available at: https://github.com/fchollet/keras.