MagNet Challenge 2023 Report

Xinyu Liu, Chaoying Mei, Rui Zhao, Gaoyuan Wu, Hao Wu

***Abstract*—This report is proposed by the team from Fuzhou University, which introduces the team's work in this competition. First, validation results on the final given data are provided including the accuracy and model size. Then, the method we used for core loss modeling is described. The network structure, fine-tuning strategy, and training tricks are discussed. Moreover, we record some attempts and observations during the experiment, which may provide potential insights.**

I. Results

Results on the "2023 MagNet Challenge Testing Data" are reported in this section including accuracy and model size. The accuracy is obtained on the given training data, which is further split to secondary training set and validation set. The model size is indicated by the trainable parameters in the network, which is counted by public tools and is further verified by manual calculation.

Notice: The accuracy in this report does not represent our final test results in 2023 MagNet Challenge. It only reflects the model performance on the validation set partitioned from the given training data.

A. Experimental Setup

Experiments are run on a cloud sever with Intel(R) Xeon(R) Platinum 8255C CPU, RTX 3080 GPU, Ubuntu 20.04, and CUDA 11.8. The deep learning framework is Pytorch 2.0.0 with Pytorch lightning 2.1.2 using Python 3.8. In order to ensure the reproducibility, we make the following settings in our code:

```
lightning.pytorch.seed_everything(666)
torch.backends.cudnn.benchmark=False
torch.use_deterministic_algorithms(True)
```

We also apply another local workstation to verify that the experiment results are reproducible. The environment of this workstation is: Intel(R) Xeon(R) w5-2465X CPU, NVIDIA GeForece RTX 4090 GPU, Windows 10 22H2, and CUDA 12.1. The deep learning framework is Pytorch 2.1.2 with Pytorch lightning 2.1.2 using Python 3.8. The validation results are consistent in both environment.

Training data in "2023 MagNet Challenge Testing Data" are further split into training subset (90%) and validation subset (10%). For data pre-processing, the frequency f and core loss P are transformed using natural logarithm function. Moreover, all the inputs and outputs are scaled by Z-score normalization. The normalization parameters include mean and variance that are calculated from all training samples (i.e., 100% of the given training data).

Xinyu Liu, Chaoying Mei, Rui Zhao, Gaoyuan Wu, and Hao Wu are with the College of Electrical Engineering and Automation, Fuzhou University, Fuzhou 350108, China (e-mail: xinyu3307@163.com; 184278164@qq.com; 1131495139@qq.com; 1834237464@qq.com; 2593303519@qq.com).

B. Accuracy

Error distributions for 5 testing materials are depicted in Fig. 1. For the appointed evaluation metric in this challenge, i.e., 95th percentile error, our model achieves 3.76%, 2.04%, 2.76%, 7.58%, and 4.15% for materials A, B, C, D, and E, respectively. A common sense finding is that more data leads to higher accuracy (e.g., compare material B and D). Another observation is that if the gap between 95-Prct and Max is too large, it may mean that there are invalid samples included in the dataset (e.g., material E).

C. Model size

Five models are trained using the same network structure for five test materials. The trainable parameter number in the network is 8914. Details of the parameters counting are shown in TABLE IV, which are counted by using the Python package "torchinfo" [1]. We also calculate the parameter number by hand according to the network structure and the input data size. The input size of the $B(t)$ field, frequency f, and temperature T are (1024,1), (1,), and (1,), respectively. The output P is a scalar with size (1,), which refers to the predicted core loss.

TABLE I
Parameters counting of the network.

Layer	Input size	Output size	Parameters[a]
Projector B			
Linear (Tanh[b])	(1024,1)	(1024,24)	48
Linear	(1024,24)	(1024,24)	600
Transformer Encoder			
MHSA	(1024,24)	(1024,24)	2400
FFN	(1024,24)	(1024,24)	1984
LN	-	-	96
Projector Fusion			
Linear (Tanh)	(1024,26)	(1024,40)	1080
Linear (Tanh)	(1024,40)	(1024,40)	1640
Linear	(1024,40)	(1024,1)	41
Regression Head			
Linear	1024	1	1025

[a] Total parameters: 8914.
[b] The activation function of this Linear layer is Tanh.

It can be seen that the Transformer Encoder is the major part of the network that occupies nearly half of the parameters. This is reasonable since the excitation flux density $B(t)$ is the main input of the core loss model. It has a long sequence of sample points containing the physical information of flux density, dc-bias, waveform shape, etc. This information can only be extracted for subsequent analysis by a sufficiently powerful feature extractor. The parameter number N_{Encoder} of Transformer Encoder can be calculated by:

$$N_{\mathrm{Encoder}} = N_{\mathrm{MHSA}} + N_{\mathrm{FFN}} + N_{\mathrm{LN}} \tag{1}$$

[1] [Online]. Available: https://pypi.org/project/torchinfo/

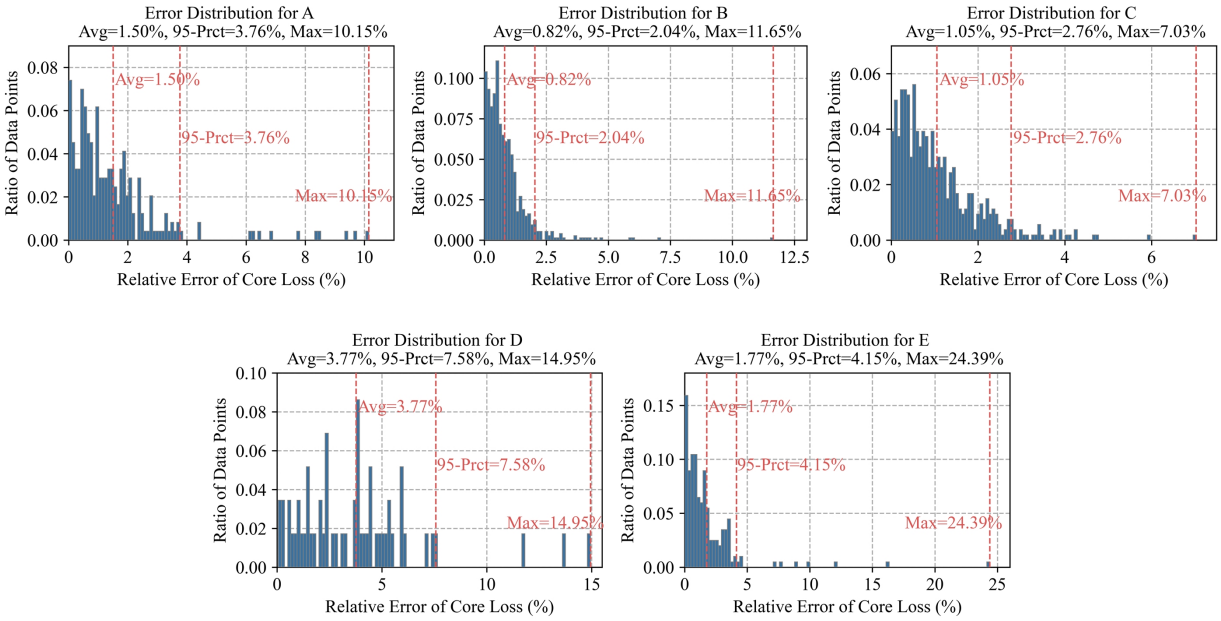


Fig. 1. Error distributions for 5 materials in "2023 MagNet Challenge Testing Data". Results are obtained from the self-separation sets using the training data, in which 90% for training and 10% for validation.

where N_{MHSA}, N_{FFN}, and N_{LN} denote the parameter numbers of Multi-head self-attention (MHSA), Feed forward network (FFN), and Layer normalization (LN), respectively. Parameters in MHSA are constructed by the weight matrices of $W^Q \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{model}}}$, $W^K \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{model}}}$, $W^V \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{model}}}$, $W^O \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{model}}}$. Moreover, each weight matrix has its bias with d_{model} parameters. The d_{model} refers to the number of expected features in the input for the Transformer Encoder, which is 24 in this report. Then, the parameter number N_{MHSA} can be computed by:

$$\begin{aligned} N_{\mathrm{MHSA}} &= N_{W^Q} + N_{W^K} + N_{W^V} + N_{W^O} \\ &= 4 * (d_{\mathrm{model}}^2 + d_{\mathrm{model}}) \\ &= 4 * (24^2 + 24) \\ &= 2400 \end{aligned} \tag{2}$$

The Transformer Encoder has two FFN in which a FNN is composed of the Linear layers:

$$\mathrm{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2 \tag{3}$$

where σ refers to the activation function, $W_1 \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{FFN}}}$ and $W_2 \in \mathbb{R}^{d_{\mathrm{FFN}} \times d_{\mathrm{model}}}$ are weights of the Linear layers. The $b_1 \in \mathbb{R}^{d_{\mathrm{FFN}}}$ and $b_b \in \mathbb{R}^{d_{\mathrm{model}}}$ are biases. In this report, the d_{FFN} is 40. Therefore, parameters are counted by:

$$\begin{aligned} N_{\mathrm{FFN}} &= N_{W_1} + N_{W_2} + N_{b_1} + N_{b_2} \\ &= 2 * d_{\mathrm{model}} * d_{\mathrm{FFN}} + d_{\mathrm{FFN}} + d_{\mathrm{model}} \\ &= 2 * 24 * 40 + 24 + 40 \\ &= 1984 \end{aligned} \tag{4}$$

The LN has two learnable affine transform parameters γ and β. Each of them has the same dimension as the last dimension of the input data (i.e., d_{model}). Therefore, the number of trainable parameters in LN is equal to $2 * d_{\mathrm{model}}$. Considering that a Transformer Encoder has two LN, the parameter number N_{LN} can be computed by:

$$\begin{aligned} N_{\mathrm{LN}} &= 2 * 2 * d_{\mathrm{model}} \\ &= 2 * 2 * 24 = 96 \end{aligned} \tag{5}$$

Projector Fusion is another major part in the network, which aggregates information of excitation flux density $B(t)$, frequency f, and temperature T. Its parameter number N_{ProjF} is the summary of three Linear layers:

$$N_{\mathrm{ProjF}} = \sum_i N_{\mathrm{L}}^i, i \in \{1, 2, 3\} \tag{6}$$

where N_{L}^i refers to the parameter number of the i-th Linear layer, it can simply be computed in terms of dimensional transformations:

$$N_{\mathrm{L}} = d_{\mathrm{in}} * d_{\mathrm{out}} + d_{\mathrm{out}} \tag{7}$$

where d_{in} and d_{out} are feature dimensions of Linear layer input and output, respectively. Then, the parameter number of each Linear layer can be calculated by:

$$\begin{aligned} N_{\mathrm{L}}^1 &= 26 * 40 + 40 = 1080 \\ N_{\mathrm{L}}^2 &= 40 * 40 + 40 = 1640 \\ N_{\mathrm{L}}^3 &= 40 * 1 + 1 = 41 \end{aligned} \tag{8}$$

Projector B and Regression Head are both constructed by the Linear layer. Their parameter numbers are calculated in the same way as the Projector Fusion.

II. Methodology

Fig. 6 provides the overview of our core loss modeling method. The main idea of this method is transfer learning that reuses knowledge from previous task by using pre-training and fine-tuning approaches. It has been adopted in core loss

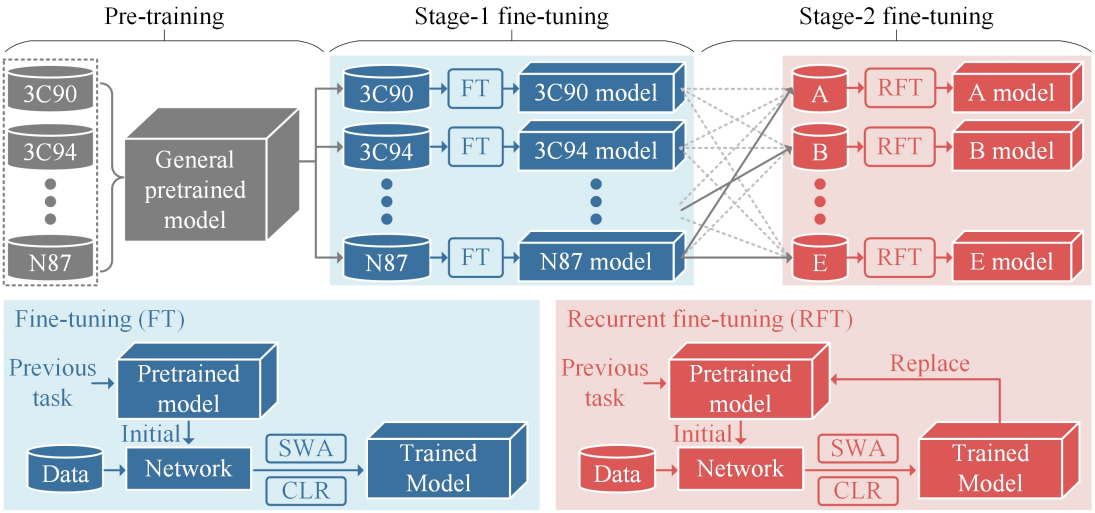


Fig. 2. Overview of our core loss modeling method based on a multi-stage fine-tuning strategy. In the pre-training, data from all 10 materials are used to train a general pre-trained model. This model is then employed in the stage-1 fine-tuning to train a specific model for each material, using data exclusively from that particular material. In stage-2 fine-tuning, stage-1 models are used as pre-trained model to train models for 5 unseen materials, respectively. Finally, we obtain 10 models for each material, but only the best model for each material is retained. SWA: Stochastic weight averaging. CLR: Cyclical learning rate.

modeling and has been proven to be effective. Based the previous works, we introduce a number of modifications and tools to further enhance the effectiveness of the transfer-learning-based approach, which are described in this section.

A. Sequence-to-Scalar Transformer

In the tutorial provided by MagNet Challenge, the model is constructed by the sequence-to-sequence Transformer. However, in our experiments, a sequence-to-scalar model achieves a much better performance, e.g., 17.37% to 6.72% on 3F4 (For reference only). An intuitive interpretation is that predicting one data point is much easier than predicting multiple data points. As the core loss is calculated through the B-H loop in the sequence-to-sequence model, the accumulation of prediction errors from multiple data points can result in a degradation of performance.

The network structure in our method is shown in Fig. 6. Detials of the network can be seen in TABLE IV. The input $B(t)$ is embedded by the Projector B and is then processed by a Transformer Encoder to extract deep abstracted features. Subsequently, the output features are aggregated with two other inputs, i.e., frequency f and temperature T. Then, the aggregated features are further processed by the Projection Fusion to mine intrinsic correlation of different kinds of inputs. Finally, the core loss P is predicted by the Regression Head.

B. Multi-stage Fine-tuning Strategy

Compared to typical fine-tuning strategy that only fine-tunes once on the target task, we apply a multi-stage fine-tuning strategy. In the pre-training (stage-0), all 10 materials are aggregated to train a General pre-trained model (GPM). And the network is randomly initialized. In the stage-1 fine-tuning, each material from 10 known materials is trained using its data. And the network is initialized from the GPM. In the stage-2 fine-tuning, each material from 5 unknown materials

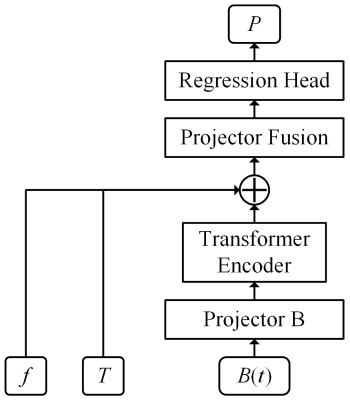


Fig. 3. Network structure of the sequence-to-scalar Transformer.

is trained using its data. And the network is initialized from the model obtained in stage-1. Finally, 10 models are trained for each material while the best one is kept.

1) Why not use GPM in stage-2?: We find that using GPM even worse than random initialization. A possible reason is that for each test material, there are both similar and dissimilar materials in the 10 known materials. Knowledge of dissimilar materials would affect the process of transfer learning thus leading to performance degradation.

2) Why use GPM in stage-1?: The impact of dissimilar materials for stage-1 is relatively small. This is because the knowledge of a known material is already included in the GPM. Thus, applying GPM is better than training from scratch. However, in our recent experiments for 10 known materials, it is sometimes better to use models from similar materials as pre-trained models. This result further validates that knowledge of dissimilar materials is likely to lead to worse transfer learning.

3) Star material N87.: In stage-2 fine-tuning, pre-trained models from 10 known materials show varying effectiveness,

as can be seen in TABLE II. An interesting observation is that N87 model achieves promising results among all test materials. We speculate that there may be some materials that represent the generic core loss modeling.

TABLE II
RESULTS OF STAGE-2 FINE-TUNING (TRAIN:VAL=8:2).

Material	A	B	C	D	E
Scratch	5.36	2.61	3.71	14.21	7.32
3C90	4.73	2.30	**3.09**	13.03	5.67
3C94	4.86	2.41	3.21	13.95	5.47
3E6	4.71	2.55	3.78	17.93	6.03
3F4	6.14	2.82	3.55	15.71	6.32
77	5.20	2.61	3.29	14.05	6.03
78	5.01	2.65	3.34	14.88	5.42
N27	5.51	2.69	3.47	13.16	5.94
N30	5.37	2.34	3.56	21.18	6.25
N49	5.43	2.31	3.55	18.68	6.61
N87	**4.51**	**2.27**	3.10	**12.07**	**5.13**

4) A possible way to foundational core loss model.: Solving all problems with one foundational model is a current research goal in many domains such as CV and NLP. The results in transfer learning of core loss modeling show that a better pre-trained model can achieve better performance. This suggests that there is some general knowledge in deep learning models that can be used for other new materials. This also provides a possible way to build a foundational model for core loss modeling, which can be adapted to different materials without any adjustments, just like large language model GPT. However, before we can achieve this ambitious goal, we first need to build a foundational pre-trained model: more diverse data and better pre-training method.

C. Tricks for Model Training

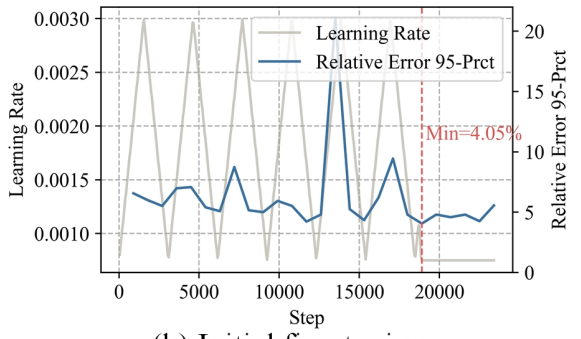
1) Efficient Coding Framework for Deep Learning: Coding is important in our research pipeline. We refactor the deep learning code with Pytorch Lightning, which greatly accelerated our research process. We do the research, Pytorch Lightning does the engineering.

2) Mixed Precision: By conducting operations in half-precision format while keeping minimum information in single-precision to maintain as much information as possible in crucial areas of the network, mixed precision training delivers significant computational speedup without accuracy loss.

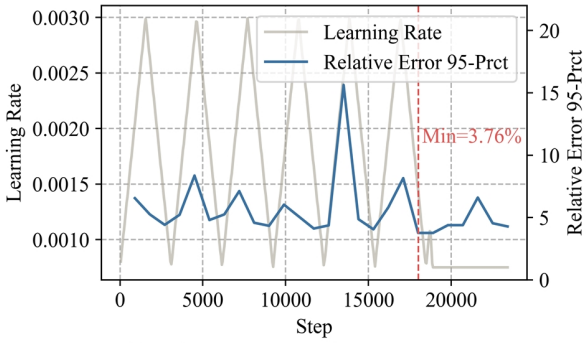
3) Cyclical Learning Rate: We use Cyclical learning rate (CLR) policy instead of warm up. The CLR policy cycles the learning rate between two boundaries with a constant frequency, which eliminates the need to perform numerous experiments to find the best values and schedule with essentially no additional computation.

4) Stochastic Weight Averaging: Stochastic Weight Averaging (SWA) is adopted after CLR training, which achieves significantly better generalization at virtually no additional cost. SWA performs an equal average of the weights traversed by a stochastic optimizer (Adam in this report) with a modified learning rate policy. In our understanding, SWA's role is similar to the model ensemble that can aggregate multiple learners to build a strong learner.

5) Recurrent Fine-tuning: Based on CLR and SWA, we propose a Recurrent fine-tuning (RFT) approach which consists of initial fine-tuning and continuous fine-tuning. The training log of RFT is depicted in Fig. 4. In the initial fine-tuning, the network is initialized by using the model from previous task, e.g., N87 model obtained from stage-1 fine-tuning. In the continuous fine-tuning, the well-trained model from initial fine-tuning is used as pre-trained model. In our understanding, RFT equates to increasing the training epoch, but has two differences. First, CLR ensures that the model does not always fall into a local optimum. Second, SWA is applied twice in the entire training period, which improves pre-trained model quality for latter half training process.



(b) Initial fine-tuning



(b) Continuous fine-tuning

Fig. 4. Training log of material A using RFT.

Results of RFT are shown in TABLE II. In materials A, B, and C, RFT achieves a better performance than that of fine-tuning directly using a longer training time. However, RFT may encounter overfitting in materials with small sample.

TABLE III
RESULTS OF RECURRENT FINE-TUNING. (TRAIN:VAL=9:1)

	A	B	C	D	E
Scratch	4.90	2.28	3.61	14.45	7.61
Fine-tuning	4.05	2.17	2.93	**7.58**	4.74
Fine-tuning (long)	4.21	2.13	2.93	**7.58**	**4.15**
Recurrent fine-tuning	**3.76**	**2.04**	**2.76**	7.79	4.29

III. DISCUSSION OF SOME ATTEMPTS AND OBSERVATIONS

A. Extra Data from Self-measurement

The performance of the data-driven method is highly dependent on the data fed into the model. Deep learning, a data-

hungry machine learning paradigm, demands more from the amount of training data. It's a natural idea to use extra data for better performance. Therefore, we construct a measurement system to obtain more samples. However, after obtaining very unsatisfactory results (6.69% to 14.07% on 3F4) and considering the expensive labor cost, we abandon this approach.

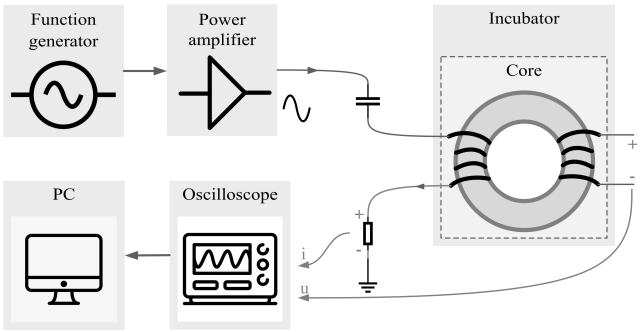


Fig. 5. Overview of our measurement system.

1) Measuring Environment: The overview of our measurement system is shown in Fig. 5. The core loss is measured by using the two-winding method. Sinusoidal excitation with different frequencies and amplitudes is provided by a function signal generator (Agilent 33522A) that is connected to the power amplifier (NF HSA4014). The voltage and output power of the excitation thus can be amplified and then be applied to the measured magnetic component. The operating temperature of the magnetic component is controlled by an incubator. Subsequently, the voltage waveform and current waveform are acquired using an oscilloscope (HD4096). Finally, by processing the acquired data, the magnetic flux density curve, magnetic field intensity curve, and magnetic core loss can be obtained. The measuring devices can be seen in Fig. 6.

Fig. 6. Devices for data acquisition.

For the measured material, we choose 3F4 that has the minimum training data among 10 materials provided in pretest phase. Details of the measuring environment are depicted in TABLE IV. Notice that the frequency interval is 25 KHz and all curves of B and H are resampled to 1024 sampling points. Finally, 662 samples are obtained and converted to MagNet format. These extra data will be packaged into our commit files for possible research purposes.

2) Analysis of Performance Degradation: Possible reasons for performance degradation are: 1) Differences in measurement environments. Although we try to emulate the measurement environment of MagNet as much as possible, due to the limited resources, it is difficult to achieve complete consistency. In particular, the frequency and excitation cannot be aligned with that of the MagNet data. 2) Differences in pre-process and post-process of the measured data. A standard publicly available guideline or toolkit may narrow the differences mentioned above. 3) The amount of extra data is still too small that can not cover more working conditions. One solution to alleviate this problem is to improve the intelligence of our measurement procedure. 4) Excitation is sinusoidal wave only, which leads to a significant bias in the model.

TABLE IV
DETAILS OF THE MEASURING ENVIRONMENT.

Property	Information
Temperature	25°C, 50°C, 75°C, 90°C
Frequency	100 KHz $\sim$ 500 KHz
Excitation	Sinusoidal wave
Sampling interval	4×10^{-9} s
Sampling point	500 $\sim$ 2500

B. Other Attempts and Observations

1) Inconsistency of Training, Validation, and Testing: In deep learning coding, there are many factors that would affect the model performance. Here we share some of the factors we encountered: 1) Mixed Precision. If mixed precision is used during training, there may be subtle differences in the results when testing with full float precision. 2) Batch size. Using different batch size for training and testing produces different results even if no batch-related modules are used. We still haven't figured out why. 3) Method in "np.percentile". Different computing methods lead to different 95-Prct results.

2) Frozen Layers: Freeze some layers (usually shallow layers) in pre-trained model and fine-tune the rest is a common trick for a deep learning application. However, this trick did not work well in our experiments (2.85% for fine-tune all layers and 3.34% for freeze Transformer Encoder on material B), probably because the pre-trained model is not powerful enough.

3) Projection of T and f: We project (or embed) T and f before Projector Fusion in an attempt to allow the model to better mine knowledge of the measuring environment. However, performance get worse (14.21% to 15.85% on material D) while model size is larger (8.9K to 11K).

4) Data Normalization: The Z-score normalization parameters are calculated using a group of samples (The more, the better), and they are material-specific (one-to-one). If the test material type is unknown, the normalization can not be applied. A general normalization parameter or a material-independent normalization approach may solve this problem.

5) Measurement Error: The equation to obtain B curve assumes a uniform distribution of magnetic induction intensity within the core. However, the distribution of magnetic induction intensity varies at different positions. Under large excitation, due to local saturation, the actual loss may exceed the computed value.