Fine-Tuning of Large Language Models for Analog Circuitry

Ahsan Mustafa
Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, Canada
ahsanm@mun.ca

Morteza Golzan
Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, Canada
smgolzan@mun.ca

Zhenxin Zhao
Innovation Center for Electronic Design Automation Technology
Hangzhou Dianzi University
Hangzhou 310018, Zhejiang, China
zxzhao@hdu.edu.cn

Hamed Nasiri
Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, Canada
hnasiri@mun.ca

Lihong Zhang
Electrical and Computer Engineering
Memorial University of Newfoundland
St. John’s, Canada
lzhang@mun.ca

***Abstract*— Recently, utilizing Large Language Models (LLMs) has transcended conventional boundaries, marking their significance in analog integrated circuit design. This paper primarily explores the innovative application of LLMs in designing and optimizing multi-stage operational amplifiers, a cornerstone in analog electronics. By leveraging the computational prowess of LLMs, we introduce a novel approach to determining optimal component values such as resistors, capacitors, biasing voltages, and transistor sizing (W/L ratios) for TSMC 65nm CMOS technology, guided by specific performance metrics, including gain, bandwidth, phase margin (PM), gain margin (GM), and power consumption. Our innovative approach focuses on fine-tuning two cutting-edge large language models (LLMs), Llama 3.1-8B-Instruct and Mistral 7B-Instruct, using a specifically curated dataset of analog circuit examples. By leveraging parameter-efficient techniques like LoRA, we empower these models to accurately predict component values, significantly enhancing the efficiency of the analog design process.**

Keywords—large language models, LLMs, Operational amplifiers, RAG, Llama 3.1-8B, Mistral-7B, RAG, RL-Reinforcement Learning, SPICE, LoRA

I. Introduction

One of the paramount obstacles in harnessing LLMs for analog integrated circuit design is the scarcity of extensive, relevant datasets crucial for effectively training these models. Traditional training datasets for LLMs comprise large corpora of text and code, which, while comprehensive, do not necessarily align with the nuanced requirements of analog circuitry optimization. One example is deducing the optimal values for different components in complex circuitry. Addressing this challenge, our contribution includes developing and presenting a meticulously curated dataset tailored for operational amplifier design.

The aim is to fine-tune LLMs on our dataset and analyze how well the LLMs adapt to the task at hand of deducing the component values for two-stage amplifier design when given specific performance metrics. We explore the computational efficiency of fine-tuning techniques such as LoRA – (Low-Rank Adaptation) and how well the LLMs' pattern recognition can work for this specific use case.

While this study focuses on fine-tuning, our methodology lays the groundwork for further advancements using complementary techniques such as Retrieval-Augmented Generation (RAG) and Reinforcement Learning (RL). These techniques provide promising opportunities for further enhancing the model's performance for optimizing current analog circuitry and generating highly accurate circuits to be used readily.

This paper conducts a comparative analysis of three LLM` to see the advantages of fine-tuning specific models for analog circuit design tasks. Investigates how well the models can perform solely by fine-tuning to automate the design task of two-stage amplifiers. The two LLMs for comparative analysis are fine-tuned Llama 3.1-8B-Instruct and fine-tuned Mistral 7B-Instruct. Through our investigation, the Llama 3.1-8B-Instruct model emerges as a notably efficient, accessible, well-supported choice, outperforming LLMs in accuracy and computational efficiency. The process of fine-tuning, alongside creating a specialized dataset, not only paves the way for enhanced model performance but also significantly reduces the computational resources required for training. This work not only advances the capabilities of LLMs in handling analog design tasks but also opens avenues for future applications where these models can serve as intelligent design assistants in the electronics industry.

Section II discusses the literature on LLMs and different approaches for fine-tuning the analog circuitry domain. Section III presents the methodology for fine-tuning and data preparation for different models and hyperparameters during training. Section IV analyzes the generated responses from the fine-tuned models and discusses the results for each model in depth. Finally, Section V presents the potential future work that can further enhance the model's performance and how fine-tuning can be adapted for different use cases.

II. Literature Review

A. A Survey of Large Language Models [7]

The survey comprehensively examines recent advancements in Large Language Models (LLMs), addressing foundational concepts from data processing to the challenges and potential applications across various domains. This resource is an excellent introduction for those seeking to understand the operational mechanics of LLMs and the current trends influencing their development and usage. It effectively illustrates the progression of LLMs from basic statistical models to sophisticated neural architectures capable of executing complex language tasks. Despite the significant strides made—particularly with models such as OpenAI’s GPT-4—the mechanisms underlying the emergent capabilities of these models are still only partially understood.

The survey delineates LLM development into four key stages: Pre-training, Adaptation, Utilization, and Evaluation. The Pre-training stage involves the foundational training of LLMs on extensive datasets to build basic language comprehension. In contrast, the Adaptation stage is dedicated to refining these models for specific applications. Utilization pertains to the methods of deploying these models for practical downstream tasks, and Evaluation encompasses metrics used to assess LLM performance.

Focusing on two vital areas—Pre-training and Training Adaptation—the survey highlights the scaling laws that impact advanced language capabilities. It also reviews parameter-efficient fine-tuning methods, including Low-Rank Adaptation (LoRA), Adapter Tuning, Prefix Tuning, and Prompt Tuning. Techniques such as LoRA and the insights derived from pre-training methods are particularly pertinent to ongoing projects, as they provide effective strategies for fine-tuning LLMs with minimal resource requirements, aligning seamlessly with analog circuit design initiatives.

This survey indicates that these methods have been predominantly tested on smaller-scale pre-trained language models. Many fine-tuned models for specific tasks, along with custom datasets, can be found on the Hugging Face Platform.

B. LADAC - Large Language Model-driven Auto-Designer for Analog Circuits [1]

The LADAC (Large Language Model-driven Auto-Designer for Analog Circuits) paper presents an innovative framework that utilizes large language models (LLMs) to automate the design of analog circuits. This approach addresses the unique challenges associated with analog circuitry, which differ significantly from those related to digital circuit automation. While there has been notable progress in automating digital design—where LLMs can successfully generate Verilog code—there is limited research on applying these models to analog design. This gap arises from the complex interconnections of analog components, the iterative tuning of transistor sizing, and the LLMs' insufficient understanding of analog intricacies. Through testing with GPT-4, the authors demonstrate that although LLMs excel in digital design tasks, they struggle with independent transistor sizing and circuit simulation in analog contexts.

LADAC's approach consists of multiple components designed to overcome these limitations, with the LLM serving as the primary decision-making agent. The LADAC workflow comprises four key elements: User Input, which provides circuit descriptions and target performance metrics; the LLM Agent (GPT-4), responsible for optimization, parameter tuning, and knowledge retrieval; an Action Library that includes simulation and optimization tools; and a Local Library containing analog circuit knowledge, references, and functions to enhance decision-making.

The process involves breaking down tasks based on user input, solving problems systematically to manage complex designs in stages, iteratively tuning parameters based on simulation feedback, and integrating optimization algorithms such as the Artificial Bee Colony (ABC) for refining designs. Experimental results validate LADAC's capabilities in automating analog circuit design, showcasing its effectiveness in tasks like Bode plot generation and autonomous transistor sizing. This framework offers a promising foundation for future research on integrating reinforcement learning and retrieval-augmented generation to improve LLM-driven analog circuit design.

III. METHODOLOGY

We introduce a systematic approach for fine-tuning large language models (LLMs) specifically for analog circuit design, focusing on two-stage amplifiers. This methodology encompasses the development of a custom dataset alongside the fine-tuning of state-of-the-art models, specifically Llama 3.1-8B-Instruct and Mistral 7B-Instruct, to accurately predict component values based on defined performance metrics.

We will outline the essential steps of our methodology, which include dataset preparation, preprocessing, LoRA configuration, training configuration, and the evaluation of inference outcomes.

A. Dataset Curation

One of the key contributions of this paper is the curation of a specialized dataset composed of 8,740 examples derived from 65 nm TSMC CMOS technology, specifically focused on two-stage amplifiers. Each example has been meticulously verified to ensure precise component values and corresponding performance metrics. This dataset is designed to assist the model in predicting values for transistors, resistors, capacitors, and biasing voltages in two-stage amplifiers. Every entry in the dataset includes a range of performance metrics, such as gain, bandwidth, phase margin, and power consumption, which are all critical for the model to generate component values accurately.

Fig. 1. below depicts an example from our dataset illustrating the prompt with context and desired performance metrics to the LLM and the expected response.

B. Data Preprocessing

We implemented tailored preprocessing steps for each model to ensure the dataset aligns with the structural requirements of Llama 3.1-8B-Instruct and Mistral 7B-Instruct. This process involved organizing the input-output pairs according to the specific instruction-following format of each model, where every example starts with a system role, followed by user and assistant roles. We also incorporated special tokens and formatting elements, such as <|begin_of_text|> and <|eot_id|>, to aid the models in accurately interpreting the analog circuit tasks. This preprocessing phase enabled both models to consistently parse the dataset and maintain context regarding the analog circuit design problem.

According to [2] and [3] Llama 3.1-8B-Instruct model functions through four distinct roles: System, User, IPython, and Assistant.

The System role provides essential context to the AI model, including guidelines and foundational information that enhance the effectiveness of its responses. The User role involves submitting queries and requests to the model. Additionally, the ipython role facilitates the generation and execution of Python code in response to user inquiries. Finally, the Assistant role delivers the model's generated responses.

Each role is associated with specific tokens that ensure the model produces accurate and relevant results during inference. The data was pre-processed accordingly for Llama 3.1-8B-Instruct, and similarly, Mistral-7B-Instruct had its own format template. The dataset was split into three, 80% for training, 10% for testing, and 10% for validation.

C. LoRA Configuration

We have employed Low-Rank Adaptation (LoRA) to enhance efficiency, which allows for adapting these large models while minimizing the need for extensive computational resources.

LoRA allows us to fine-tune a pre-trained model by adding a small set of trainable parameters. We effectively adapt pre-trained LLMs by introducing low-rank matrices to existing model weights rather than updating all the parameters directly. This significantly reduces the memory and computational requirements, making fine-tuning possible with relatively limited resources for models like Llama 3.1-8B and Mistral-7B, comprising 8 billion and 7 billion parameters, respectively. In the case of Llama 3.1-8B, utilizing LoRA, we managed to fine-tune it by training only 0.0522% of the total number of trainable parameters. Meanwhile, for Mistral-7B, it was 1.5732%. Fig. 2, [4], below illustrates how LoRA allows us computational savings by only training a small number of parameters in an LLM. W_0 represents the pre-trained model parameters, which are frozen, and ΔW represents the added trainable parameters, where the number of columns of B or the number of rows of A represents the "rank" for LoRA. The rank essentially controls how much the LLM adapts to the dataset. For fine-tuning both models, we chose the rank to be 16. The task type to set to "CAUSAL_LM" refers to sequence generation. LoRA alpha, the learning rate of LoRA configuration, was set to 16, and LoRA dropout was chosen to be 0.02, which prevents overfitting to the training data by nullifying random weights during the training process.

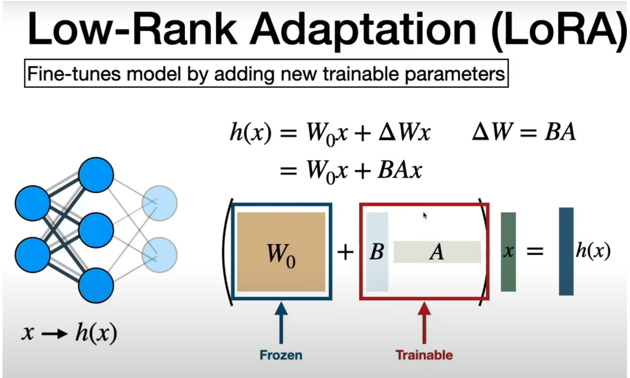

Fig. 2. [4] shows the trainable parameters, BxA, where columns of B or the number of rows of A represent the "rank" in LoRA configuration.

D. Training Configuration

For our training process, we thoroughly explored specific hyperparameters to effectively fine-tune the Llama 3.1-8B and Mistral 7B models for our unique dataset. Key training hyperparameters included a carefully selected learning rate of 1^{-4}, which is critical in controlling the speed at which our model learns. We also chose a batch size of 4, determining the number of training samples processed before the model's internal parameters are updated. To ensure a comprehensive training experience, we conducted a total of 4 epochs, allowing the model to adjust and enhance its understanding of the data iteratively.

Prompt

Determine the resistor and capacitor values, the sizing of MOSFET transistors, and the biasing voltages (i.e., R0, C0, L1, W1, L2, W2, L3, W3, L4, W4, L5, W5, VIN (for both VINP and VINN), VB1) for the provided tv
First Stage: Input Transistors: M1, M2 (PMOS): Differential input transistors with the same sizing (L1, W1). Differential input signal applied.
Current Mirror & Load: M3, M4 (NMOS): Current mirror and load transistors with the same sizing (L3, W3).
Biasing Transistor: M5 (PMOS): Current source transistor with sizing (L2, W2) and gate voltage (VB1).
Second Stage: Biasing Transistor: M7 (PMOS): Biasing transistor with sizing (L5, W5) and gate voltage (VB1).
Amplifying Transistor: M6 (NMOS): Connected with a resistor (R0) in series with a capacitor (C0) as feedback. Sizing (W4, L4) controls the gain of the second stage.
Netlist: MOSFET Transistors:

```
M1 pch net3 VINP net2 VDD L1 W1
M2 pch net1 VINN net2 VDD L1 W1
M5 pch net2 VB1 VDD VDD L2 W2
M3 nch net3 net1 GND GND L3 W3
M4 nch net1 net1 GND GND L3 W3
M6 nch VOUT net3 GND GND L4 W4
M7 pch VOUT VB1 VDD VDD L5 W5
```

Capacitor:

```
C0 net3 net4 C0_value
```

Resistor:

```
R0 net4 VOUT R0_value
```

Netlist Column Explanation:
Component Name (M for MOSFET, C for Capacitor, R for Resistor)
For transistors:Type (pch for PMOS, nch for NMOS), Drain Net Name, Gate Net Name, Source Net Name, Bulk Net Name, Length (L) of the Transistor, Width (W) of the Transistor
For resistor and capacitor: One terminal net name, Other terminal net name, Capacitor or Resistor Value

Biasing Voltages: VINP, VINN, VB1.
Performance Metrics: Power=0.000568653W, Gain=10.73705dB, 3dB Bandwidth (BW_3dB)=3383769.0Hz, Unity Gain Bandwidth (UGB)=16295760.0Hz, Phase Margin (PM)=0.0degree, Gain Margin (GM)=0.0dB

Completion

To achieve the given performance metrics with the provided amplifier netlist, you can use the following values in the netlist:

```
M1 pch net3 VINP net2 VDD L1=638.0nm W1=6.0um
M2 pch net1 VINN net2 VDD L1=638.0nm W1=6.0um
M5 pch net2 VB1 VDD VDD L2=139.0nm W2=40.0um
M3 nch net3 net1 GND GND L3=426.0nm W3=98.0um
M4 nch net1 net1 GND GND L3=426.0nm W3=98.0um
M6 nch VOUT net3 GND GND L4=130.0nm W4=12.0um
M7 pch VOUT VB1 VDD VDD L5=391.0nm W5=17.0um
C0 net3 net4 C0_value=2.0pF
R0 net4 VOUT R0_value=6.0KOhm
```

And with biasing voltages of VINP=VINN=VIN= 237.0mV, VB1= 312.0mV.

Fig. 1. shows an example from the dataset, providing the desired performance metrics and the structure of the netlist. While Completion illustrates how the model should respond.

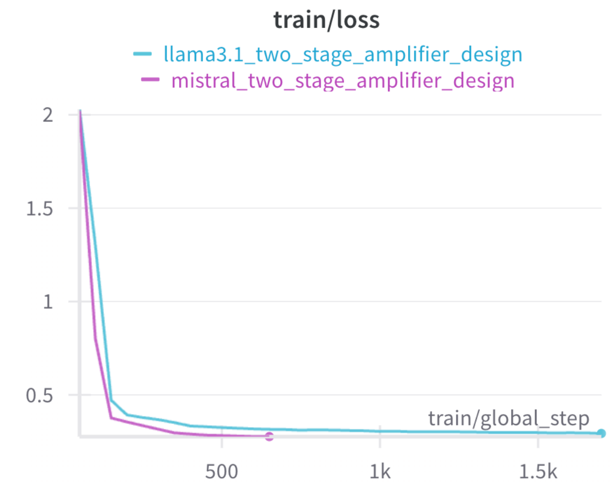

Fig. 3. Shows the training loss for both models

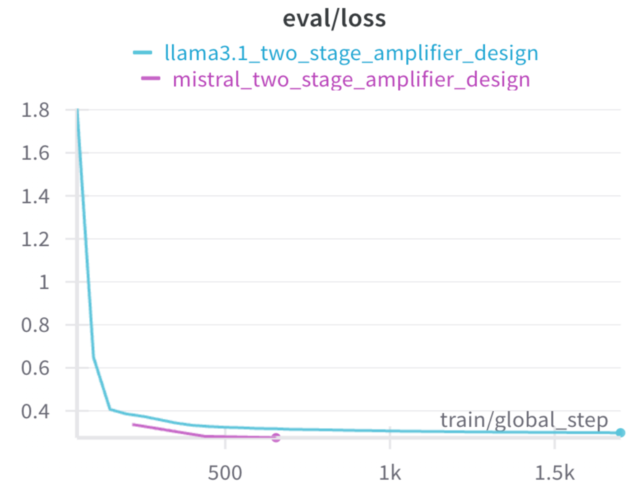

Fig. 4. Shows the validation loss for both models.

Throughout this training phase, we focused on iterative refinement, employing techniques such as early stopping and model checkpointing. This approach ensured that the models effectively learned the intricate correlations between performance metrics and component values while maximizing resource efficiency. By assessing the models' performance at various stages, we made informed decisions that contributed to the overall effectiveness of our training strategy.

E. *Evaluation for Inference*

Following the training phase, we assessed the models' outputs by conducting inference on a test subset of the dataset. We utilized metrics such as Mean Absolute Error (MAE) to gauge the accuracy of the predicted component values. Furthermore, we visualized the model's predictions compared to the actual values, analyzing the distribution and deviations to uncover any patterns or inconsistencies in performance.

This inference stage offered valuable insights into how effectively each model generalizes to unseen examples and fulfills the requirements for analog circuit design tasks.

IV. Inference Results and Analysis

The results are categorized into training and validation results of the model and the model responses compared to the actual values.

A. *Training and Validation Loss*

Fig. 3. shows the training loss for Llama 3.1-8B-Instruct and Mistral 7B-Instruct. Fig. 4. Shows the validation loss for both models respectively.

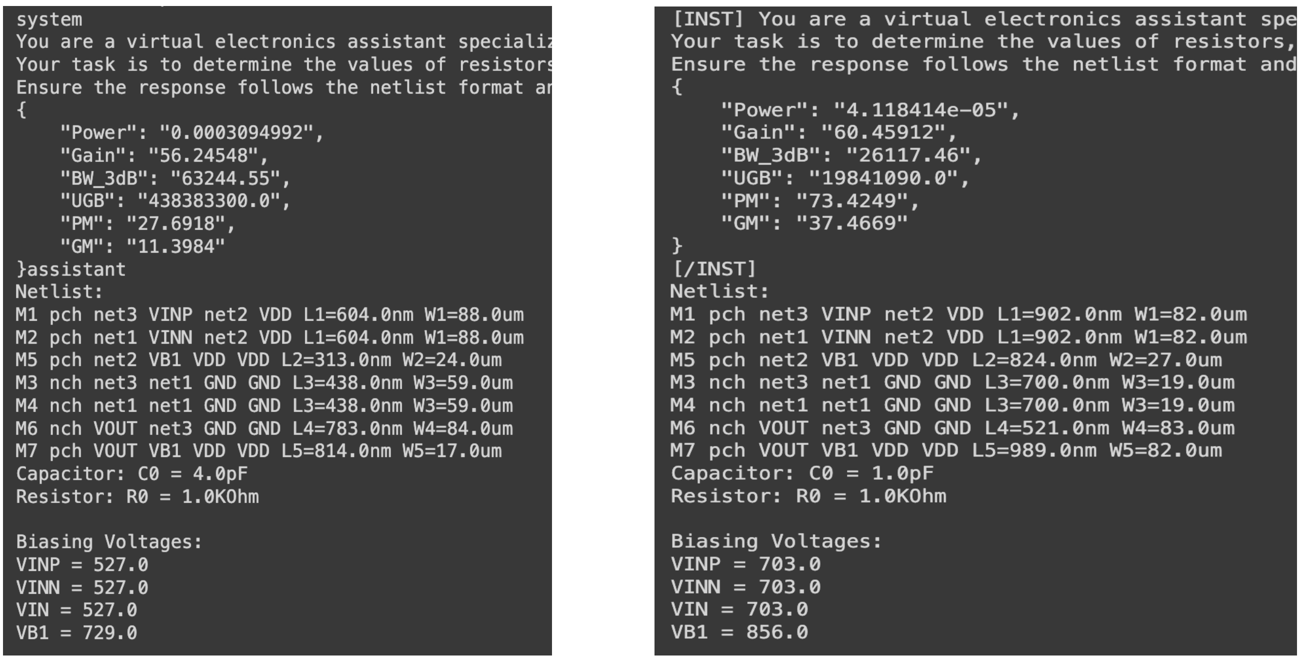

Fig. 5. Shows the inference with Llama 3.1-8B-Instruct (left) and Mistral-7B-Instruct (right) when provided instruction in the format of our dataset. The model's are fine-tuned quite well as they adapt to the output format and provide in the dataset.

The training loss and validation decreased steadily over each epoch, indicating that both models successfully captured the relationship between performance metrics and component values and did not overfit the training data. As Llama 3.1 8B-Instruct had significantly more trainable parameters, the training continued for more epochs than Mistral 7B-Instruct, and the best model was loaded for inference in the end.

B. *Inference Results and Analysis*

We ran inference on a subset of test data, 100 examples, to evaluate the model's real-world applicability and compared generated and actual responses. Fig. 5 shows the inference results for Llama 3.1-8B-Instruct and Mistral 7B-Instruct.

Fig. 6 illustrates the distribution of the actual and predicted values. The model generally adapts very well to the actual values for most of the parameters; however, it falters significantly in some cases, especially when observing the lengths of the MOSFETs. This is also evident in Fig. 7, which shows a box plot of the error distribution computer using MAE. This is expected as the model's generated responses may fulfill the requirement of the desired performance metrics.

Even though the model outputs some accurate results, according to our analysis, fine-tuning alone might not be able to provide us with a readily used product.

FUTURE WORK

To enhance the accuracy and adaptability of our model, future research may involve the integration of Retrieval-Augmented Generation (RAG) and Reinforcement Learning (RL) techniques. RAG stands to significantly improve the model's comprehension of analog circuit design by facilitating access to relevant external resources, datasets, and design heuristics that support real-time decision-making. By leveraging RAG, the model can tap into an extensive repository of analog design literature and historical design examples, thereby enriching its responses and enabling adaptation to novel inquiries.

Furthermore, incorporating RL can establish a feedback-driven learning loop, wherein the model iteratively refines its predictions based on performance evaluations. For example, an RL framework could operate in conjunction with SPICE simulation software to provide structured feedback concerning circuit performance metrics such as gain, bandwidth, and power consumption. The SPICE simulator would assess each generated design and furnish feedback on the degree to which the output aligns with specified targets. Over time, this methodology would allow the model to fine-tune its predictions for enhanced accuracy, ultimately streamlining the analog design process by amalgamating LLM-based generation with iterative simulation validation.

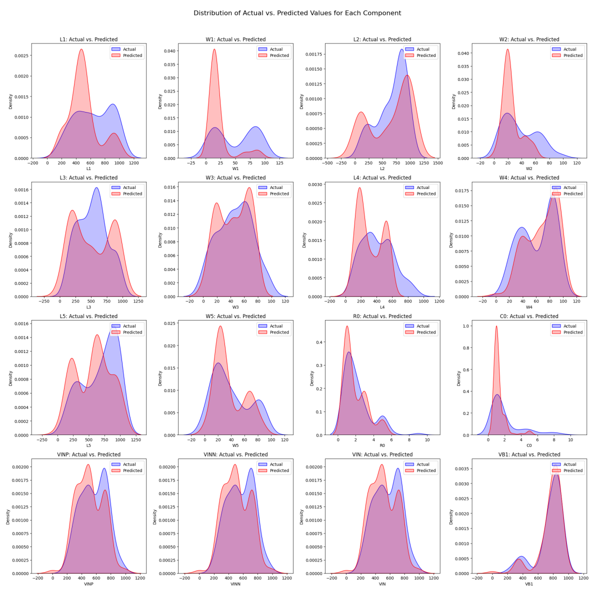

Fig. 6. Shows the distribution of actual and predicted values for each parameter.

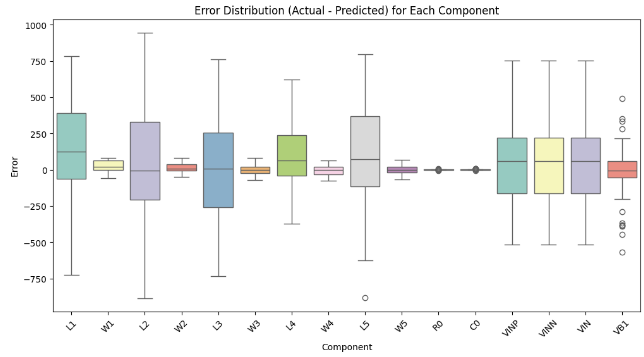

Fig. 7. Shows the error distribution, using MAE, for each component.

REFERENCES

[1] C. Liu, Y. Liu, Y. Du, and L. Du, "LADAC: Large Language Model-driven Auto-Designer for Analog Circuits," *TechRxiv*, Jan. 2024, doi: 10.36227/techrxiv.170473941.10097233/v1.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] "Llama 3.1 | Model Cards and Prompt formats." https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_1/

[3] "meta-llama/Llama-3.1-8B-Instruct · Formats for prompting the model using Hugging face," Aug. 16, 2001. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct/discussions/103

[4] S. Talebi, "QLORA — How to Fine-Tune an LLM on a single GPU | by Shaw Talebi | towards Data Science," *Medium*, Aug. 06, 2024. [Online]. Available: https://towardsdatascience.com/qlora-how-to-fine-tune-an-llm-on-a-single-gpu-4e44d6b5be32

[5] T. Liao and L. Zhang, "Efficient parasitic-aware hybrid sizing methodology for analog and RF integrated circuits," *Integration*, vol. 62, pp. 301–313, Mar. 2018, doi: 10.1016/j.vlsi.2018.03.017.

[6] T. Liao and L. Zhang, "High-Dimensional Many-Objective Bayesian Optimization for LDE-Aware Analog IC Sizing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 1, pp. 15-28, Jan. 2022, doi: 10.1109/TVLSI.2021.3102088.

[7] W. X. Zhao *et al.*, "A survey of large language models," *arXiv.org*, Mar. 31, 2023. https://arxiv.org/abs/2303.18223

[8] Z. Zhao and L. Zhang, "Efficient Performance Modeling for Automated CMOS Analog Circuit Synthesis," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 11, pp. 1824-1837, Nov. 2021, doi: 10.1109/TVLSI.2021.3107404.

[9] Z. Zhao and L. Zhang, "Deep Reinforcement Learning for Analog Circuit Structure Synthesis," 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 2022, pp. 1157-1160, doi: 10.23919/DATE54114.2022.9774699.

[10] Z. Zhao, J. Luo, J. Liu and L. Zhang, "Signal-Division-Aware Analog Circuit Topology Synthesis Aided by Transfer Learning," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 42, no. 11, pp. 3481-3490, Nov. 2023, doi: 10.1109/TCAD.2023.3245979.

[11] Z. Zhao and L. Zhang, "Analog Integrated Circuit Topology Synthesis With Deep Reinforcement Learning," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 12, pp. 5138-5151, Dec. 2022, doi: 10.1109/TCAD.2022.3153437