Narrow Transformer: Mono-lingual Code SLM for Desktop

Kamalkumar Rathinasamy[1] **Balaji A J**[1] **Ankush Kumar**[1]
Rajab Ali Mondal[1] **Harshini K**[1] **Sreenivasa Raghavan K S**[1]
Gagan Gayari[1] **Swayam Singh**[*] **Mohammed Rafee Tarafdar**[1]

{kamalkumar_r, balaji_jayaram, ankush.kumar06,
rajab.mondal, harshini.k04, sreenivasa.seshadri,
gagan.gayari, mohammed.tarafdar}@infosys.com, singhswayam08@gmail.com

[1]Infosys Limited

Abstract

This paper presents NT-Java-1.1B[2], an open-source specialized code language model built on StarCoderBase-1.1B[3], designed for coding tasks in Java programming. NT-Java-1.1B achieves state-of-the-art performance, surpassing its base model and majority of other models of similar size on MultiPL-E (Cassano et al., 2022 [1]) Java code benchmark. While there have been studies on extending large, generic pre-trained models to improve proficiency in specific programming languages like Python, similar investigations on small code models for other programming languages are lacking. Large code models require specialized hardware like GPUs for inference, highlighting the need for research into building small code models that can be deployed on developer desktops. This paper addresses this research gap by focusing on the development of a small Java code model, NT-Java-1.1B, and its quantized versions, which performs comparably to open models around 1.1B on MultiPL-E Java code benchmarks, making them ideal for desktop deployment. This paper establishes the foundation for specialized models across languages and sizes for a family of NT Models.

1 Introduction

The state-of-the-art code models, capable of understanding and generating code in numerous programming languages, are revolutionizing the way enterprises approach software development. With the ability to understand and generate code across a vast array of programming languages, these code models offer a significant boost in productivity. However, the one-size-fits-all approach of these generic multi-lingual code models often falls short in meeting the nuanced requirements of project-level coding tasks in an enterprise, which tend to be language-specific. This has led to the development of Narrow Transformers (NTs), specialized models further trained on a particular programming language, offering a more efficient solution for enterprises. These NTs are designed to optimize performance for a specific programming language, balancing the trade-offs between model size, inferencing cost, and operational throughput. As demand for tailored solutions grows, we can expect a surge in NT development, providing the precision and efficiency required by enterprise projects.

[*]Affiliated to: University of Allahabad
[2]https://huggingface.co/infosys/NT-Java-1.1B
[3]https://huggingface.co/bigcode/starcoderbase-1b

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

However, in practice, the substantial economic cost associated with training and fine-tuning large code models renders language model experiments prohibitively expensive for most researchers and organizations. Additionally, deploying these massive models in everyday scenarios, such as on personal computers, proves either inefficient or unfeasible. These challenges emphasize the importance of shifting focus to explore Narrow Transformer approach on powerful yet smaller code language models (code SLMs). Consequently, we developed a Narrow Transformer for Java within a smaller parameter range (i.e., 1.1B), suitable for desktop deployment and democratizing code model experiments.

2 Related Work

Codex-12B (Chen et al., 2021 [2]) was built by extending pre-training of GPT, with 159 GB of unique Python files, from public software repositories hosted on GitHub. Codex exhibits its highest proficiency in Python; however, it also demonstrates competence in over twelve additional programming languages. CodeGen-Mono-350M/2.7B/6.1B/16.1B (Nijkamp et al., 2023 [3]) were built by further pretraining CodeGen-Multi-350M/2.7B/6.1B/16.1B with the mono-lingual dataset BIGPYTHON that contains public, non-personal, permissively licensed Python code from GitHub. CodeGen-Mono outperformed CodeGen-Multi on Python as per the HumanEval benchmark. StarCoder-15.5B (Li et al., 2023 [4]) was built by extending pre-training of StarCoderBase-15.5B (which was trained with multi-lingual datasets comprising code from 80+ programming languages) with a Python subset of 35B tokens from the StarCoderBase training data. StarCoder outperformed StarCoderBase on Python as per the HumanEval benchmark. In the evaluation of StarCoder and StarCoderBase on 19 programming languages with MultiPL-E datasets, StarCoder outperformed StarCoderBase on Python, underperformed on 9 programming languages, and despite being further trained only on Python, it still outperformed StarCoderBase on 9 other programming languages. CodeLlama-PYTHON-7B/13B/34B/70B (Baptiste et al., 2023 [5]) were built by extending pre-training of CodeLlama-7B/13B/34B/70B (which were trained on 500B tokens of code data, except CodeLlama-70B, which was trained on 1T tokens) on 100B tokens of python heavy dataset. CodeLlama-PYTHON outclasses CodeLlama on Python on MultiPL-E benchmarks, but it is not consistent on rest of the languages. While extending pretraining of multi-lingual code models with a specific language dataset doesn't guarantee improvement in performance in other languages, it guarantees improvement in that language. Enterprises are adopting these generic or Python-trained multi-lingual models to enhance coding tasks, with AI-mature enterprises fine-tuning them using their own codebases. However, if a pre-trained model is already specialized in the required language, further training on the project's codebase yields better results. Given Java's widespread use in enterprise projects, this paper illustrates the development of such a pre-trained code model specialized on Java.

Small Language Models (SLMs) will shift the AI community's focus in enterprise and consumer solutions due to their ability to run on personal devices without a GPU, enabling large-scale deployment while maintaining data privacy and security. Significant examples in the present scenario of code SLMs include SantaCoder-1.1B (Ben Allal et al., 2023 [6]), Phi-1 (Gunasekar et al., 2023 [7]), DeciCoder-1B[4], StarCoderBase-1.1B, WizardCoder-1B-V1.0 (Luo et al., 2023 [8]), DeepSeek-Coder-1b-base (Guo et al., 2024 [9]) and Refact-1.6B[5]. All these state-of-the-art models around 1B size are multi-lingual code models, indicating that no considerable work has been done towards extending training of multi-lingual code SLMs in building language-specific code SLMs.

3 Datasets

The foundation model for our experiment was StarCoderBase-1.1B. Enterprise projects shortlist candidate models for coding tasks based on factors like licensing and training data. Using any dataset beyond StarCoderBase for extending its pretraining would complicate the adoption of NT-Java-1.1B due to licensing concerns. Therefore, we used a subset of StarCoderData[6], the curated dataset from The Stack v1[7] used to train StarCoderBase, to build NT-Java-1.1B.

[4]`https://huggingface.co/Deci/DeciCoder-1b`
[5]`https://huggingface.co/smallcloudai/Refact-1_6B-fim`
[6]`https://huggingface.co/datasets/bigcode/starcoderdata`
[7]`https://huggingface.co/datasets/bigcode/the-stack`

The Java dataset from StarCoderData was used for training NT-Java-1.1B. The Java dataset is around 22B tokens.

4 Model Training

4.1 Data Preprocessing

For data preprocessing, we employed the Megatron-LM framework for data preprocessing. The NT-Java-1.1B employs the StarCoderBase GPT2BPETokenizer with a 49,152-token vocabulary, without additions. The Java dataset (87 parquet files) was merged into one file and processed through Megatron to generate .bin and .idx files for training. The pre-processing also tokenizes and appends an <EOD> token to each Java sample.

4.2 Model Architecture

NT-Java-1.1B, similar to StarCoderBase-1.1B, is a decoder-only Transformer model with Multi-Query Attention (Shazeer, 2019 [10]), which uses FlashAttention. This speeds up the attention computation and reduces the training time of the model. The hyper-parameters for the architecture can be found in Table 1.

Table 1: Model architecture of NT-Java-1.1B.

Hyperparameter	NT-Java
Hidden size	2048
Intermediate size	8192
Max. position embeddings	8192
Num. of attention heads	16
Num. of hidden layers	24
Attention	Multi-query
Num. of parameters	$\approx$ 1.1B

4.3 Training Details

NT-Java-1.1B was trained using the Megatron-LM Framework[8]. The training began with StarCoderBase-1.1B, serving as the initial checkpoint, to build its Java variant. In our experiments, we utilized a context length of 8192 tokens for tasks involving the Next token prediction and the Fill-in-the-Middle (FIM) (M Bavarian, 2022 [11]) objective. The PyTorch Distributed framework was employed, with data parallelism strategy. We chose bf16 precision and the Adam optimizer (Kingma & Ba, 2015 [12]) with $\beta 1 = 0.9$, $\beta 2 = 0.95$, and $\epsilon = 10^{-8}$, along with a weight decay of 0.1.

Experimental Settings

In this study, we delve into the impact of extending pretraining of StarCoderBase-1.1B for Java using two key objectives: Next token prediction and Fill-in-the-Middle.

Experiment 1 - Next token prediction objective: We conducted training over 100,000 steps (equivalent to 5 epochs) with a batch size of 1 million tokens. The learning rate commenced at $4{\times}10^{-4}$ and underwent cosine decay, reaching a minimum of $4{\times}10^{-6}$ with 1,000 iterations of linear warmup. A global batch size of 180 facilitated the training process, which spanned 12 days. Model checkpoints were saved every 1,000 steps for subsequent evaluation.

Experiment 2 - Fill-in-the-Middle: We repeated Experiment 1 along with FIM training objective. The FIM rate was set to 50%. The FIM dataset was evenly split into two components, SPM (Suffix-Prefix-Middle) and PSM (Prefix-Suffix-Middle).

Observation from Experiment 1 & 2: Without FIM training objective, the model's infilling capability diminished significantly, with FIM scores approaching nearly zero (Table 2), despite the

[8] `https://github.com/Infosys/Megatron-LM#nt-java-11b-extending-pretraining`

base model's inherent infilling capability. While training with FIM objective, we observed a minor decrease in MultiPL-E metrics (approximately 0.7%) compared to the model trained without FIM objective, but the model retained its proficiency in infilling tasks. The comparative performance of the models throughout the training are illustrated in Figure 1.

Table 2: Experimental results with and without FIM.

Model	FIM	HumanEval-FIM (Java)	MultiPL-E (Java)
NT-Java-1.1B (Experiment 1)	No	0.01	19.6
NT-Java-1.1B (Experiment 2)	Yes	0.67	18.9

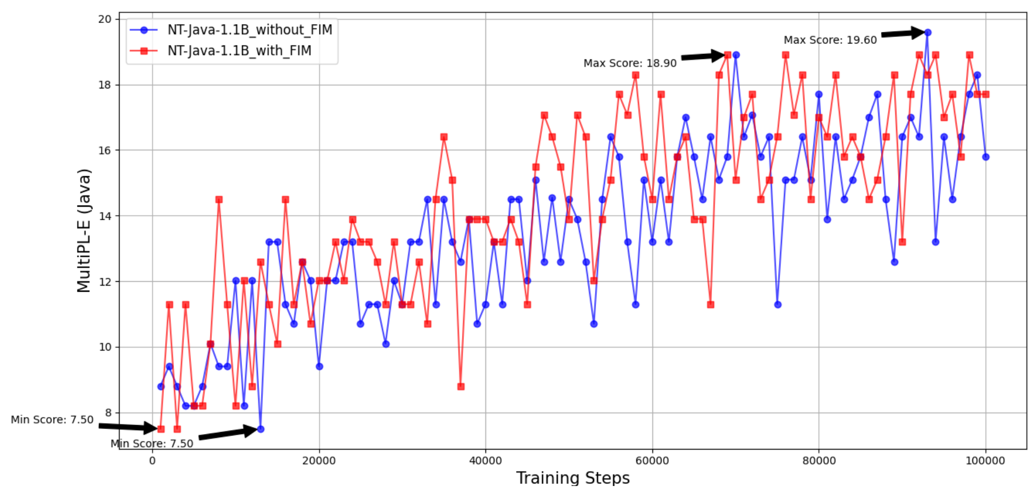

Figure 1: MultiPL-E Scores of NT-Java-1.1B trained with and without FIM.

Experiment 2.1 - Fill-in-the-Middle: We extended training from Experiment 2 for 20,000 steps (1 epoch) more as the evaluation scores were in an upward trend. The learning rate commenced at 4×10^{-6} and underwent cosine decay, reaching a minimum of 4×10^{-7} with 1,000 iterations of linear warmup. We did not intend to continue further training as the model converged with no significant decrease in loss.

4.4 Post Training

The NT-Java-1.1B model, with bf16 precision, is 2.27 GB in size. To create more compact models for desktop use without major accuracy loss, quantized versions of model in GGUF[9] format were developed for CPU-based frameworks like Ollama[10], GPT4ALL[11], and LM Studio[12]. The quantized versions of the models (NT-Java-1.1B-GGUF[13]), ranging from 2-bit to 8-bit, reduce the model size from 511 MB to 1.32 GB.

4.5 Compute

NT-Java-1.1B was trained with 6 A100 80 GB GPUs on a single-node GPU cluster. The training process remained stable overall, with only a few restarts.

[9] https://github.com/ggerganov/ggml/blob/master/docs/gguf.md
[10] https://github.com/ollama/ollama
[11] https://github.com/nomic-ai/gpt4all
[12] https://github.com/lmstudio-ai
[13] https://huggingface.co/infosys/NT-Java-1.1B-GGUF

5 Evaluation

This section presents evaluation of our proposed coding SLM to assess its capabilities in code generation and infilling tasks.

5.1 MultiPL-E

In our initial assessment, we evaluated the NT-Java-1.1B model (Experiment 2.1) on Java code generation tasks using the MultiPL-E benchmark and the BigCode Eval Harness[14], adhering to Big Code Models Leaderboard[15] norms. NT-Java-1.1B achieved a higher pass@1 score than its base model and 3B variant, as shown in Table 3.

Table 3: Pass@1 results on MultiPL-E.

Model	Java
StarCoderBase-1.1B	14.2
StarCoderBase-3B	19.25
NT-Java-1.1B	**20.2**

5.2 Fill-in-the-Middle Benchmark

Subsequently, we evaluated the model's performance on single-line code infilling using the SantaCoder benchmark, which measures 'line exact match' accuracy on Java code within HumanEval solutions. Our model showed results comparable to StarCoderBase-1.1B, as detailed in Table 4.

Table 4: HumanEval-FIM scores.

Model	Java
StarCoderBase-1.1B	0.71
NT-Java-1.1B	**0.67**

5.3 Computational Capabilities

Furthermore, we also assessed the model's efficiency and resource utilization. As shown in Table 5, NT-Java quantized models strike an optimal balance between accuracy and resource use, making them ideal for resource-constrained environments. The MultiPL-E scores for the quantized variants were computed using the 'load in 4-bit' and 'load in 8-bit' parameters in the BigCode Eval Harness.

Table 5: Accuracy and resource utilization.

Model	Pass@1 (Java)	Size (GB)
StarCoderBase-1.1B	14.2	≈ 2.27
NT-Java-1.1B_Q4	**15.1**	**0.76**
NT-Java-1.1B_Q8	**17.7**	**1.23**
StarCoderBase-3B	19.25	≈ 6.1
NT-Java-1.1B	**20.2**	**2.27**

6 Limitations

NT-Java-1.1B is currently limited to Java and does not support other programming languages, which necessitates the development of separate models for each language. To ensure a fair comparison, we have focused on evaluating the model's performance against models of similar sizes (1.1B & 3B) in same family (StarCoderBase) only.

[14] `https://github.com/bigcode-project/bigcode-evaluation-harness`

[15] `https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard`

7 Conclusion

In this technical report, we outlined the rationale and training approach used to develop NT-Java-1.1B, a small language model trained specifically on Java code. We evaluated NT-Java-1.1B on coding tasks and found it to be competitive with, or outperforming, other similarly sized models for Java programming.

This study demonstrates the successful achievement of its objective of enhancing the efficiency of a code SLM for a particular programming language by training it further with a subset of its dataset for that language. While the research employed the StarCoderBase-1.1B model and its Java language dataset, other SLMs and their associated programming language datasets can yield comparable experimental outcomes.

The release of NT-Java-1.1B and its variants aims to democratize code foundation models, making them accessible for deployment in memory-constrained environments such as developer desktops and laptops. By adhering to the principles of the OpenRAIL-M[16] and by open-sourcing the corresponding scripts on GitHub, we hope to enable both the research and developer communities to experiment and adopt code SLMs.

Acknowledgments and Disclosure of Funding

We extend our gratitude[17] to S Reka for her support in distributed training, and to Gokul Ayyappan, Mayank Singh, Pranavi Suvvari, Puneeth Sai Charan Raghavendra Gunturu for their contributions to data analysis. We appreciate Jaideep Valani's insights on SLMs, and Jaskirat Singh Sodhi's efforts in evaluations. We thank Infosys[18] for providing the compute resources.

References

[1] Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. Multipl-e: A scalable and extensible approach to benchmarking neural code generation. August 2022.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[3] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[4] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier,

[16] https://bigscience.huggingface.co/blog/bigscience-openrail-m

[17] s_reka@infosys.com, gokul.ayyappan@infosys.com, mayank.singh28@infosys.com, pranavi.suvvari@infosys.com, puneethsaicharan.g@infosys.com, jaideep.valani@infosys.com, jaskirat.s@infosys.com

[18] https://www.infosys.com

João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Moustafa-Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you! *CoRR*, abs/2305.06161, 2023.

[5] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023.

[6] Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Muñoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, Logesh Kumar Umapathi, Carolyn Jane Anderson, Yangtian Zi, Joel Lamy-Poirier, Hailey Schoelkopf, Sergey Troshin, Dmitry Abulkhanov, Manuel Romero, Michael Lappert, Francesco De Toni, Bernardo García del Río, Qian Liu, Shamik Bose, Urvashi Bhattacharyya, Terry Yue Zhuo, Ian Yu, Paulo Villegas, Marco Zocca, Sourab Mangrulkar, David Lansky, Huu Nguyen, Danish Contractor, Luis Villa, Jia Li, Dzmitry Bahdanau, Yacine Jernite, Sean Hughes, Daniel Fried, Arjun Guha, Harm de Vries, and Leandro von Werra. Santacoder: don't reach for the stars! *CoRR*, abs/2301.03988, 2023.

[7] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *CoRR*, abs/2306.11644, 2023.

[8] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *CoRR*, abs/2306.08568, 2023.

[9] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. *CoRR*, abs/2401.14196, 2024.

[10] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150, 2019.

[11] Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. July 2022.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction provides the reason and scope of the research.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to limitations section as those are discussed there.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The related work section outlines the underlying assumptions, while the evaluation section serves to validate these assumptions.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: As the complete training information is being discussed in model training section, it can be reproduced.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have open-sourced the data and code . Currently not added as reference to maintain anonymity.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Experimental setting are provided in training details subsection under model training section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: As the experiments are performed, statistical information of the experiments is provided in experimental settings subsection.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to compute subsection in model training section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research has not scraped any data and no crowdsourcing is done. The information of datasets is provided for reference to validate the process followed by the original owners.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The introduction section also discusses the impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We have shortlisted base model and datasets that adheres to responsible practices with necessary safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Owner of assets are properly credited in references and footnotes.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have open-sourced all assets, but they are not currently referenced to maintain anonymity.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing was done during the data collection.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines: This study does not involve humans.

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.