# Predicting LLM Inference Latency: A Roofline-Driven ML Method

**Saki Imai**[*]
Northeastern University

**Rina Nakazawa**
IBM Research

**Marcelo Amaral**
IBM Research

**Sunyanan Choochotkaew**
IBM Research

**Tatsuhiro Chiba**
IBM Research

## Abstract

Recent advancements in Large Language Models (LLMs) for Generative AI have significantly increased their popularity, resulting in an exponential arise of new close and open LLM models with frequent algorithm updates. Hence, the typical approach of running and learning to define the optimal configuration starts to be unpractical due to the large combinatorial problem and shortage/cost of GPU resources, which creates the necessity for predictive performance models. Given that, we propose a new LLM performance prediction model that can be leveraged for optimal cluster management. The novelty of our approach is the combination of an analytical Roofline Model (RLM) specific for LLM implementation and based on the hardware characteristic with data from Regression Models trained with historical data. More specifically, our approach calibrates the theoretical hardware performance given from RLM with inherent runtime overhead captured by Regression Models, offering a more interpretable and accurate prediction method in cloud-based deployments. We validate our method for vLLM and Triton inference servers, demonstrating that our approach improves the $R^2$ value by $12\%$ and reduces MSE by up to $80\%$ on vLLM, and improves the $R^2$ value by $4\%$ and reduces MSE by up to $61\%$ on Triton, compared to other regression-only models.

## 1 Introduction

Improvements in large language models (LLMs) have significantly expanded their capabilities, making them paramount across various applications in recent years(1; 2; 3). However, deploying these models at scale presents challenges, particularly in optimizing their inference latency. Efficient serving of LLMs with system-level enhancements has therefore become a critical research area (4; 5; 6), driving innovations in memory management (7; 8; 9; 10), computational optimization (11; 12; 13; 14; 15), and recently, efficient cloud deployment strategies (16; 17; 18). Given the frequent introduction of new models and architectures, measuring the performance of each LLM via experiment under all possible conditions is too resource-intensive and time-consuming. Therefore, an accurate and efficient performance model that predicts LLM latency based on minimal execution parameters becomes necessary for optimal application's resource allocation strategies. Related works such as (13; 16; 19) primarily aim to improve system-level performance through optimizing scheduling, energy efficiency, and model parallelism. While effective, these methods target specific optimization strategies without a unified performance prediction framework. Our approach aligns most closely with (20), which also provides high-fidelity latency predictions. However, we build upon this by integrating theoretical modeling with real-time execution data, enabling more accurate and efficient resource allocation without extensive simulations.

[*]Work done as an intern at IBM Research

Machine Learning for Systems Workshop at (NeurIPS 2024).

The motivation for using the Roofline Model (RLM) in our work is its established ability to offer an analytical framework that balances computational throughput and memory bandwidth (21). RLM provides an interpretable performance ceiling for hardware systems, making it an ideal candidate for predicting LLM inference latency. While it has been applied successfully to hardware systems like CPUs, GPUs, and TPUs (22; 23; 24; 25), its extension to LLM inference remains relatively unexplored. However, RLM alone lacks critical runtime information, such as the variability introduced by cloud infrastructure and dynamic workloads, which leads to performance overestimation and sub-optimal predictions in LLM inference scenarios (26).

In this paper, we address these limitations by proposing a novel LLM end-to-end inference latency prediction method. Our approach combines the theoretical hardware RLM with runtime characteristics from regression model trained with historical data to create an interpretable model that reflects actual execution conditions in the cloud environments. We deploy this scheme in vLLM (27) and Triton (28), to validate proposed model effectiveness. The contributions of this work are as follows:

1. We introduce an end-to-end latency prediction method that is both theoretical and grounded in real-world execution data, resulting in a substantial increase in the $R^2$ values and significant reduction in the MSE on vLLM and Triton inference servers.
2. We validate the effectiveness of our method by evaluating it across several open-source LLMs from Hugging Face, deployed in vLLM and Triton along with a set of experiments in our own cluster.

## 2 Latency Prediction Method

The goal of our work is to train a LLM inference latency prediction model with both collected data from real-world experiments and theoretical performance data from Roofline models, which will show that can outperform the inference latency prediction of those data in isolation. Our data collection procedure is described in Section 2.2 and the method of collecting analytical inference time informed by the RLM is described in 2.3.

### 2.1 Selected Models and Model Feature Extraction

In this work, we focused on LLMs that are optimized for deployment within a single GPU environment to evaluate the effectiveness of our framework while minimizing external factors. The target LLMs represent a diverse range of architectures, allowing us to evaluate the inference latency across varying architectural factors, rather than focusing solely on parameter size. The four LLMs chosen from Hugging Face are: **microsoft/phi-2** (29), **facebook/opt-1.3b** (30), **EleutherAI/pythia-1.4b** (31), and **openai-community/gpt2-xl** (32). For each LLM, we extracted 9 key features from the model's configuration file on Hugging Face (33; 34; 35; 36) (LLM feature table is in the Appendix A.1):

When training the model, we use *categorical features*: (i) model name, (ii) architecture, (iii) model type, (iv) hidden act, and *numerical features:* (v) # of parameters, (vi) hidden size, (vii) # of heads, (viii) # of layers and (ix) vocabulary size.

### 2.2 Execution Latency Data Collection Procedure

To collect LLM inference latency data, we conducted experiments on vLLM and Triton. Note that Triton supports multiple backends and we used a vLLM backend for Triton server (37). Our primary objective was to analyze the impact of user concurrency and generation length on latency. To this end, in our experiments, we varied the number of concurrent users (8, 16, 32) and generation lengths (32, 64, 128 tokens) during the inference, using a fixed generation prompt: "What is artificial intelligence?". These were single-pass, non-multiturn experiments iterated for three times for each set on an OpenShift cluster with a single NVIDIA A100 GPU.

To simulate real-world usage scenarios and capture latency data, we employed Locust (38) for load testing. Each experiment ran for 5 minutes, allowing us to capture the system's behavior under sustained load. This ensured that the hardware was fully utilized. The reported latency values are the average of these three runs, as we observed minimal variation across repetitions with identical setups. With these experiments, we collected three operational parameters and an execution latency in milliseconds: (i) number of users, (ii) generation lengths, and (iii) average content size in MB.

### 2.3 Integration of the Roofline model

The Roofline model provides an analytical approach to estimate the theoretical inference latency, which is calculated by the formula introduced in the Appendix A.2. Therefore, by using the LLM-Viewer (26), which applies the Roofline model to estimate the inference latency of a chosen LLM and hardware, based on the input length, batch size, and generation length, we collected the theoretical analytical latency data for different number of users, and generation lengths as described in 2.2. Although the LLM-Viewer tool supports only a subset of LLMs (e.g., opt-1.3b), we extended the open-source codebase to incorporate three additional LLMs of our interest. In our performance modeling in Section 3, we incorporate this analytically estimated latency as an additional feature to improve the accuracy of our execution latency predictions.

## 3 Modeling Results

In this section, we present the results of our inference latency prediction models, applying the default configurations of Random Forest (RF), XGBoost (XGB), and Linear Regression (LR) from scikit-learn (39), without any fine-tuning. We evaluate the performance of the models in two scenarios: out-of-domain (OOD) prediction, where the target LLM was excluded from the training data, and in-domain (ID) prediction, where the model was trained for a single LLM. In OOD, we assess how well our models generalize to unseen LLMs by predicting the inference latency of the LLMs that were not included in the training set. The Roofline Model (RLM) explores ID predictions and incorporates Roofline performance features to improve prediction accuracy. We report only the best results for each LLM, while the full results are provided in Appendix A.3 for reference.

Table 1: Comparison of $R^2$, MSE, and Method for different LLMs (RF, XGB, LR) across vLLM and Triton inference servers for OOD, ID, and RLM predictions.

| | | OOD | | | ID | | | RLM | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Server | Model | $R^2 \uparrow$ | MSE $\downarrow$ | Method | $R^2 \uparrow$ | MSE $\downarrow$ | Method | $R^2 \uparrow$ | MSE $\downarrow$ | Method |
| vLLM | opt | 0.903 | 11748 | LR | 0.800 | 24282 | LR | 0.974 | 3163 | LR |
| | phi | 0.908 | 18580 | XGB | 0.805 | 39215 | LR | 0.987 | 2535 | LR |
| | pythia | 0.950 | 5840 | XGB | 0.929 | 8157 | LR | 0.985 | 1687 | LR |
| | gpt2 | 0.724 | 58505 | XGB | 0.917 | 17527 | LR | 0.951 | 10384 | LR |
| Triton | opt | 0.961 | 12374 | XGB | 0.940 | 19463 | LR | 0.986 | 4538 | LR |
| | phi | 0.974 | 8878 | RF | 0.919 | 27314 | LR | 0.987 | 4392 | LR |
| | pythia | 0.859 | 41883 | LR | 0.966 | 10115 | LR | 0.995 | 1458 | LR |
| | gpt2 | 0.454 | 415931 | XGB | 0.948 | 39675 | LR | 0.965 | 26326 | LR |

### 3.1 Out of domain prediction (OOD)

In the OOD prediction task, the models were trained on a set of 9 LLM features and 3 operational parameters (as described in 2.1 and 2.2). This configuration allowed us to evaluate how well the models generalize to unseen systems. The left most column in Table 1 shows the results of OOD prediction. We can see that the models achieve strong generalization for three out of four LLMs, with $R^2$ values exceeding 0.85. However, for gpt2 the prediction accuracy was lower with $R^2$ value of 0.724 for vLLM and 0.454 for Triton. To understand what contributed to this reduced accuracy, we compared the total number of operations (OPs) across the LLMs. Using the LLM-Viewer, we calculated the average number of OPs for all 9 configurations. The results showed that the gpt2 had the least OPs, averaging $6.4 \times 10^{12}$ which is substantially smaller than $1.04 \sim 1.61 \times 10^{13}$ for the other LLMs. This discrepancy is likely due to its small hidden size of 1600, which may explain the lower effectiveness of the OOD method.

### 3.2 In domain prediction (ID)

In this task, the models were trained and tested on data from a single LLM. Given that the LLM's architecture remained constant, we focused on the impact of the operational parameters, as described in 2.2. Each model was trained on a dataset comprising 9 configurations across varying numbers of concurrent users and generation lengths. Due to the small sample size, we used 3-fold cross validation

to evaluate our models. Despite the small dataset, all four models deployed in two different servers achieved an $R^2$ value exceeding 0.8, suggesting that the performance modeling with a limited data points are feasible. In contrast to the OOD results where the best modeling method depended on the LLM, the ID results suggest that there is a linear relationship between the operational parameters and inference latency.

### 3.3 Incorporation of the Roofline model (RLM)

By incorporating the Roofline-based analytical latency estimates as features, we were able to capture key computational and memory hardware bottlenecks that were previously unaccounted for in the model. Incorporating the analytical latency led to a significant increase in $R^2$ values for most models, with the $R^2$ score for the opt model on vLLM server increased by 17 points, and the MSE error was reduced by $87\%$, compared to the ID prediction. Note that the dataset set size remained consistent with the ID prediction, with the only change being the inclusion of the analytical inference latency.

Similar to the trend seen in ID, the results incorporating the Roofline model shows that LR was the most effective method for capturing inference latency. We believe that the LR worked well since the Roofline model addresses the two bottlenecks out of the three performance bottlenecks, which are compute bound and memory bandwidth bound (40). The LR effectively models the third bottleneck, the overhead bound, by introducing a constant into the model.

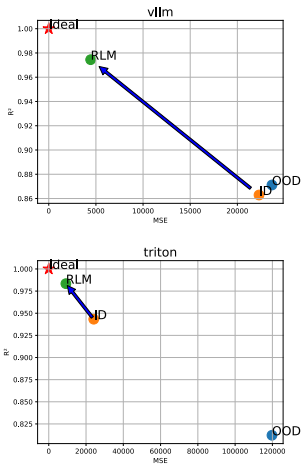


Figure 1: Scatter Plot of Average $R^2$ vs. Average MSE

**Overall performance comparison** Figure 1 shows the average $R^2$ and MSE scores across the three prediction approaches for vLLM and Triton inference servers. On average, the inclusion of the Roofline model led to a $12.93\%$ and $4.25\%$ improvement in $R^2$ compared ID prediction for vLLM and Triton, respectively. Additionally, the RLM approach resulted in MSE reductions of $80.08\%$ and $61.98\%$, relative to ID prediction for vLLM and Triton, respectively. As evident in the figure, the RLM models are substantially closer to the ideal prediction, represented by a star.

Lastly, we verified that our latency inference modeling was not limited by memory bound operations and that our solution remains applicable regardless of arithmetic intensity. This is crucial because theoretically, if the operation is memory bound, performance can be captured easily by a linear regression model. To ensure this, we analyzed the operational characteristics of each model at the layer level using LLM-Viewer (26), which determined whether an operation was compute bound or memory bound based on its arithmetic intensity. We then calculated a ratio reflecting the proportion of compute-bound operations for the entire model. The average ratio across different configurations, such as sequence length and number of users, exceeded $19.5\%$ for all LLMs. This suggests that even for compute bound operations, where the performance is limited by the maximum computational throughput, the RLM model is capable of accurately predicting inference latency.

## 4 Conclusion

In this paper, we proposed a method for predicting LLM inference latency by integrating the Roofline model with LR. Our findings provide evidence that this approach significantly improves the accuracy of predictions by capturing the key performance bottlenecks and runtime overhead effects. Specifically, our proposed method led to an increase in the $R^2$ value by $12\%$ and reduces MSE by up to $80\%$ on vLLM inference server. On the Triton inference server, we observe a $4\%$ improvement in $R^2$ and a $61\%$ reduction in MSE, compared to prediction made without incorporating analytical inference time.

## References


[1] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, and X. Hu, "Harnessing the power of llms in practice: A survey on chatgpt and beyond," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 6, apr 2024. [Online]. Available: https://doi.org/10.1145/3649506

[2] M. A. K. Raiaan, M. S. H. Mukta, K. Fatema, N. M. Fahad, S. Sakib, M. M. J. Mim, J. Ahmad, M. E. Ali, and S. Azam, "A review on large language models: Architectures, applications, taxonomies, open issues and challenges," *IEEE Access*, vol. 12, pp. 26 839–26 874, 2024.

[3] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[4] B. Li, Y. Jiang, V. Gadepally, and D. Tiwari, "Llm inference serving: Survey of recent advances and opportunities," *arXiv preprint arXiv:2407.12391*, 2024.

[5] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, H. Jin, T. Chen, and Z. Jia, "Towards efficient generative large language model serving: A survey from algorithms to systems," *arXiv preprint arXiv:2312.15234*, 2023.

[6] Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, Y. Lou, L. Wang, Z. Yuan, X. Li *et al.*, "A survey on efficient inference for large language models," *arXiv preprint arXiv:2404.14294*, 2024.

[7] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, ser. SOSP '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 611–626. [Online]. Available: https://doi.org/10.1145/3600006.3613165

[8] R. Prabhu, A. Nayak, J. Mohan, R. Ramjee, and A. Panwar, "vattention: Dynamic memory management for serving llms without pagedattention," May 2024. [Online]. Available: https://www.microsoft.com/en-us/research/publication/vattention-dynamic-memory-management-for-serving-llms-without-pagedattention/

[9] H. Liu, M. Zaharia, and P. Abbeel, "Ring attention with blockwise transformers for near-infinite context," *arXiv preprint arXiv:2310.01889*, 2023.

[10] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Ré, I. Stoica, and C. Zhang, "Flexgen: high-throughput generative inference of large language models with a single gpu," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23. JMLR.org, 2023.

[11] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, "Orca: A distributed serving system for Transformer-Based generative models," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. Carlsbad, CA: USENIX Association, Jul. 2022, pp. 521–538. [Online]. Available: https://www.usenix.org/conference/osdi22/presentation/yu

[12] "Fastertransformer," https://github.com/NVIDIA/FasterTransformer.

[13] P. Patel, E. Choukse, C. Zhang, A. Shah, Goiri, S. Maleki, and R. Bianchini, "Splitwise: Efficient generative llm inference using phase splitting," in *ISCA*, June 2024. [Online]. Available: https://www.microsoft.com/en-us/research/publication/splitwise-efficient-generative-llm-inference-using-phase-splitting/

[14] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently scaling transformer inference," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 606–624, 2023.

[15] J. Shah, G. Bikshandi, Y. Zhang, V. Thakkar, P. Ramani, and T. Dao, "Flashattention-3: Fast and accurate attention with asynchrony and low-precision," *arXiv preprint arXiv:2407.08608*, 2024.

[16] T. Griggs, X. Liu, J. Yu, D. Kim, W.-L. Chiang, A. Cheung, and I. Stoica, "M\'elange: Cost efficient large language model serving by exploiting gpu heterogeneity," *arXiv preprint arXiv:2404.14527*, 2024.

[17] X. Miao, C. Shi, J. Duan, X. Xi, D. Lin, B. Cui, and Z. Jia, "Spotserve: Serving generative large language models on preemptible instances," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1112–1127. [Online]. Available: https://doi.org/10.1145/3620665.3640411

[18] B. Sun, Z. Huang, H. Zhao, W. Xiao, X. Zhang, Y. Li, and W. Lin, "Llumnix: Dynamic scheduling for large language model serving," in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024.

[19] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, "Dynamollm: Designing llm inference clusters for performance and energy efficiency," *arXiv preprint arXiv:2408.00741*, 2024.

[20] A. Agrawal, N. Kedia, J. Mohan, A. Panwar, N. Kwatra, B. Gulavani, R. Ramjee, and A. Tumanov, "Vidur: A large-scale simulation framework for llm inference," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 351–366, 2024.

[21] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, apr 2009. [Online]. Available: https://doi.org/10.1145/1498765.1498785

[22] A. Ilic, F. Pratas, and L. Sousa, "Cache-aware roofline model: Upgrading the loft," *IEEE Computer Architecture Letters*, vol. 13, no. 1, pp. 21–24, 2014.

[23] C. Yang, T. Kurth, and S. Williams, "Hierarchical roofline analysis for gpus: Accelerating performance optimization for the nersc-9 perlmutter system," *Concurrency and Computation: Practice and Experience*, vol. 32, 11 2019.

[24] F. Chern, B. Hechtman, A. Davis, R. Guo, D. Majnemer, and S. Kumar, "Tpu-knn: K nearest neighbor search at peak flop/s," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 15 489–15 501. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/639d992f819c2b40387d4d5170b8ffd7-Paper-Conference.pdf

[25] Z. Jiang, L. Wang, X. Xiong, W. Gao, C. Luo, F. Tang, C. Lan, H. Li, and J. Zhan, "Hpc ai500: The methodology, tools, roofline performance models, and metrics for benchmarking hpc ai systems," *arXiv preprint arXiv:2007.00279*, 2020.

[26] Z. Yuan, Y. Shang, Y. Zhou, Z. Dong, C. Xue, B. Wu, Z. Li, Q. Gu, Y. J. Lee, Y. Yan, B. Chen, G. Sun, and K. Keutzer, "Llm inference unveiled: Survey and roofline model insights," 2024.

[27] "vLLM: Easy, Fast and Cheap LLM Serving for Everyone," https://github.com/vllm-project/vllm.git.

[28] "Triton inference server," https://github.com/triton-inference-server/server.

[29] M. Abdin, J. Aneja, S. Bubeck, C. C. T. Mendes, W. Chen, A. D. Giorno, R. Eldan, S. Gopi, S. Gunasekar, M. Javaheripi, P. Kauffmann, Y. T. Lee, Y. Li, A. Nguyen, G. de Rosa, O. Saarikivi, A. Salim, S. Shah, M. Santacroce, H. S. Behl, A. T. Kalai, X. Wang, R. Ward, P. Witte, C. Zhang, and Y. Zhang, "Phi-2: The surprising power of small language models," 2023. [Online]. Available: https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/

[30] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "Opt: Open pre-trained transformer language models," 2022.

[31] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff *et al.*, "Pythia: A suite for analyzing large language models across training and scaling," in *International Conference on Machine Learning*. PMLR, 2023, pp. 2397–2430.

[32] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[33] (2024) facebook/opt-1.3b. [Online]. Available: https://huggingface.co/facebook/opt-1.3b/blob/main/config.json

[34] (2024) microsoft/phi-2. [Online]. Available: https://huggingface.co/microsoft/phi-2/blob/main/config.json

[35] (2024) Eleutherai/pythia-1.4b. [Online]. Available: https://huggingface.co/EleutherAI/pythia-1.4b/blob/main/config.json

[36] (2024) openai-community/gpt2-xl. [Online]. Available: https://huggingface.co/openai-community/gpt2-xl/blob/main/config.json

[37] "Triton inference server vllm backend," https://github.com/triton-inference-server/vllm_backend.

[38] "Locust," https://github.com/locustio/locust.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] H. He, "Making deep learning go brrrr from first principles," 2022. [Online]. Available: https://horace.io/brrr_intro.html

[41] Z. Zheng, X. Ren, F. Xue, Y. Luo, X. Jiang, and Y. You, "Response length perception and sequence scheduling: An llm-empowered llm inference pipeline," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 65 517–65 530. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/ce7ff3405c782f761fac7f849b41ae9a-Paper-Conference.pdf

[42] H. Qiu, W. Mao, A. Patke, S. Cui, S. Jha, C. Wang, H. Franke, Z. T. Kalbarczyk, T. Başar, and R. K. Iyer, "Efficient interactive llm serving with proxy model-based sequence length prediction," *arXiv preprint arXiv:2404.08509*, 2024.

[43] K. Cheng, W. Hu, Z. Wang, P. Du, J. Li, and S. Zhang, "Enabling efficient batch serving for lmaas via generation length prediction," *arXiv preprint arXiv:2406.04785*, 2024.

# A Appendix / supplemental material

## A.1 Model Characteristics from Hugging Face

Table 2: Model Specifications

| Model | Hidden Size | Hidden Act | Model Type | N Head | N Layer | Vocab Size | Architecture | # of Params |
|---|---|---|---|---|---|---|---|---|
| opt-1.3b | 2048 | relu | opt | 32 | 24 | 50272 | OPTForCausalLM | 1.3B |
| phi-2 | 2560 | gelu_new | phi | 32 | 32 | 51200 | PhiForCausalLM | 2.7B |
| pythia-1.4b | 2048 | gelu | gpt_neox | 16 | 24 | 50304 | GPTNeoXForCausalLM | 1.4B |
| gpt2-xl | 1600 | gelu_new | gpt2 | 25 | 48 | 50257 | GPT2LMHeadModel | 1.6B |

## A.2 The Roofline Details Explanation

The Roofline model provides a framework for analyzing performance by relating the computational capability of hardware (in terms of operations per second) to the bandwidth of memory access. It allows us to estimate the upper bound of performance, given a specific Arithmetic Intensity (AI), which is the ratio of computational work to memory access.

The ratio of operations (OPs) to memory access determines AI:

$$\text{AI} = \frac{\text{OPs}}{\text{memory\_access}} \tag{1}$$

where OPs refers to the number of operations required for a given task and memory_access refers to the volume of data transfers (e.g., fetching weights and activations from memory)

The performance is limited either by the bandwidth of memory access or by the maximum achievable computational throughput (max_OPS)

$$\text{performance} = \begin{cases} \text{AI} \times \text{bandwidth}, & \text{if AI} < \frac{\text{max\_OPS}}{\text{bandwidth}} \\ \text{max\_OPS}, & \text{otherwise} \end{cases} \tag{2}$$

where bandwidth refers to the rate of data transfer between memory and computation (in bytes/second) and max_OPS refers to the peak computational throughput of the hardware (in operations per second)

Once the performance has been determined based on the Roofline model, the inference time for each operation or layer can be calculated as:

$$\text{inference\_time} = \frac{\text{OPs}}{\text{performance}} \tag{3}$$

The total inference time for an LLM consists of two phases: prefill and decode phase. The total inference time can be computed as the sum of the inference times across both phases, as follows:

$$\begin{aligned} \text{total\_inference\_time} = &\left( \sum_{l=1}^{L_{\text{prefill}}} \text{inference\_time}_l^{\text{prefill}} \right) \\ &+ \sum_{i=\text{prompt\_len}}^{\text{prompt\_len}+\text{gen\_len}} \left( \sum_{l=1}^{L_{\text{decode}}} \text{inference\_time}_l^{\text{decode}} \right) \end{aligned} \tag{4}$$

With the Roofline model, we can calculate an analytical inference latency of LLMs by considering both the computational intensity and the memory bandwidth limitations of the hardware.

### A.3 Other results and insights

#### A.3.1 Out of domain prediction results

Table 3: Comparison of $R^2$ and MSE values for different models (RF, XGB, LR) across vLLM and Triton inference servers for out-of-domain prediction

| | | RF | | XGB | | LR | |
|---|---|---|---|---|---|---|---|
| Server | Models | $R^2 \uparrow$ | MSE $\downarrow$ | $R^2 \uparrow$ | MSE $\downarrow$ | $R^2 \uparrow$ | MSE $\downarrow$ |
| vLLM | opt-1.3b | 0.5634 | 53002.1716 | 0.5443 | 55316.8348 | **0.9032** | 11748.1579 |
| | phi-2 | 0.8196 | 36213.6210 | **0.9075** | 18579.6461 | -5.8300e+19 | 1.1700e+25 |
| | pythia-1.4b | 0.7550 | 28312.9495 | **0.9495** | 5839.8439 | 0.9301 | 8074.4466 |
| | gpt2-xl | 0.6646 | 71209.9622 | **0.7244** | 58505.1173 | 0.4514 | 116453.0214 |
| Triton | opt-1.3b | 0.9501 | 16218.3881 | **0.9619** | 12374.4824 | 0.7337 | 86492.8719 |
| | phi-2 | **0.9737** | 8878.4013 | 0.0139 | 332776.3625 | -5.4100e+20 | 1.8300e+26 |
| | pythia-1.4b | 0.7385 | 77716.5185 | 0.8148 | 55042.1667 | **0.8591** | 41883.3045 |
| | gpt2-xl | 0.4037 | 454333.7655 | **0.4541** | 415930.6822 | 0.0316 | 737786.5246 |

#### A.3.2 In domain prediction results

Table 4: Comparison of $R^2$ and MSE values for different models (RF, XGB, LR) for vLLM inference server for in-domain prediction

| | | RF | | XGB | | LR | |
|---|---|---|---|---|---|---|---|
| Server | Models | $R^2 \uparrow$ | MSE $\downarrow$ | $R^2 \uparrow$ | MSE $\downarrow$ | $R^2 \uparrow$ | MSE $\downarrow$ |
| vLLM | opt-1.3b | 0.6918 | 37411.8021 | 0.4699 | 64352.5602 | **0.7999** | 24282.4593 |
| | phi-2 | 0.7129 | 57648.6761 | 0.7060 | 59023.7664 | **0.8047** | 39214.7970 |
| | pythia-1.4b | 0.6957 | 35171.0402 | 0.7044 | 34163.1720 | **0.9294** | 8156.7246 |
| | gpt2-xl | 0.6549 | 73258.4546 | 0.6734 | 69330.7897 | **0.9174** | 17527.0093 |
| Triton | opt-1.3b | 0.7543 | 79799.7556 | 0.7764 | 72625.4403 | **0.9401** | 19462.7767 |
| | phi-2 | 0.7560 | 82356.7808 | 0.7977 | 68281.4014 | **0.9191** | 27313.6654 |
| | pythia-1.4b | 0.8127 | 55658.6728 | 0.8958 | 30984.7781 | **0.9660** | 10115.2553 |
| | gpt2-xl | 0.7747 | 171658.8409 | 0.8822 | 89769.1954 | **0.9479** | 39674.8095 |

#### A.3.3 Prediction with the Roofline Model

Table 5: Comparison of $R^2$ and MSE values for different models (RF, XGB, LR) for vLLM and Triton inference servers using the Roofline Model

| | | RF | | XGB | | LR | |
|---|---|---|---|---|---|---|---|
| Server | Models | $R^2 \uparrow$ | MSE $\downarrow$ | $R^2 \uparrow$ | MSE $\downarrow$ | $R^2 \uparrow$ | MSE $\downarrow$ |
| vLLM | opt-1.3b | 0.6687 | 40213.3071 | 0.7380 | 31809.8520 | **0.9739** | 3162.5134 |
| | phi-2 | 0.7121 | 57808.1003 | 0.7127 | 57684.2802 | **0.9874** | 2535.0774 |
| | pythia-1.4b | 0.7146 | 32989.3233 | 0.7044 | 34160.2992 | **0.9854** | 1686.5280 |
| | gpt2-xl | 0.6781 | 68339.6833 | 0.7422 | 54729.6965 | **0.9511** | 10384.2784 |
| Triton | opt-1.3b | 0.7608 | 77701.6086 | 0.8115 | 61211.0672 | **0.9860** | 4537.5362 |
| | phi-2 | 0.7723 | 76857.4284 | 0.7976 | 68305.0763 | **0.9870** | 4391.8022 |
| | pythia-1.4b | 0.8087 | 56873.4912 | 0.8997 | 29808.2705 | **0.9951** | 1458.1791 |
| | gpt2-xl | 0.7813 | 166590.8172 | 0.9059 | 71672.1134 | **0.9654** | 26326.0134 |

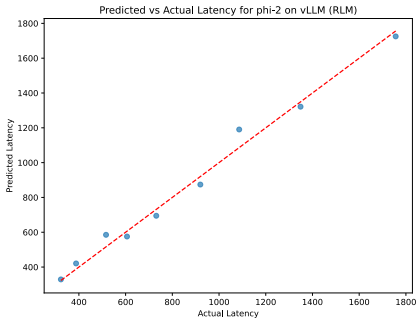


Figure 2: Predicted vs Actual Latency for phi-2 on vLLM

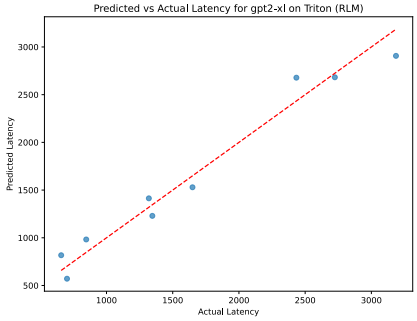


Figure 3: Predicted vs Actual Latency for gpt2-xl on Triton

### A.3.4 Predicted vs Actual Latency plots with the Roofline Model

## A.4 Limitation

Our work fixed the generation length from LLM to 32, 64 and 128 tokens for the purpose of modeling the inference latency. In practical application, however, the generation length is not predetermined, and varies depending on the nature of the task and input. To accurately model the inference latency in real world scenarios, we must consider predicting the generation length based on the input prompt (41; 42; 43).

Another limitation is that our study did not incorporate inflight batching for Triton Inference Server. Inflight batching can aggregate multiple requests into a single batch for more efficient processing and this could reduce latency.

## A.5 Future work

Our focus was on evaluating the effectiveness of our approach, so we limited our study to smaller-sized language models, though larger 7B models could also be run on a single A100 GPU. Future work should extend this analysis to include these larger models, as they are increasingly common in practical applications and offer additional insights into the latency scaling with model size. Furthermore, testing across different hardware configurations, such as more powerful or specialized GPUs, TPUs, or Multi-Instance GPUs, could provide deeper insights into scalability and performance for different inference strategies.

Additionally, future work could examine the impact of tensor parallelism and alternative deployment strategies on inference latency. Exploring different optimization strategies could provide a more robust understanding of how to achieve efficient, low-latency inference in diverse settings.

## 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: We believe that our abstract and introduction of the paper clearly delivers the contributions, scope and motivation of the problem.

   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: Due to the tight page constraint, we have addressed the limitation of our paper in the Appendix, along with potential future work to mitigate these limitations.

   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 theoretical inference latency derivation using the Roofline model is provided in the Appendix, as it is not part of our original contribution.

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: All the LLMs and servers we tested are open-source and we have provided a detailed methodology in the paper to ensure reproducibility.

   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 will release the code along with detailed instructions for reproducibility after the paper is accepted.

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: We have described the dataset and training details for each prediction methods in the paper. The accompanying code provides full details on our modeling and prediction methods.

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: [NA]

Justification: Our work did not include any confidence tests or experiments to demonstrate statistical significance.

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: As indicated in the paper, we conducted all experiments on a single NVIDIA A100 GPU within an OpenShift cluster, with each experiment running for 5 minutes.

   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: Our work did not involve any human subjects, and the datasets used are publicly available on Hugging Face. We believe that sharing the latency dataset collected from our experiments poses no risk of harm and does not lead to discrimination against any specific population.

   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 proposed method is expected to have a positive social impact on environmental sustainability through the optimization of cloud resource usage. However, incorrect latency predictions could result in negative social consequences. To address this, implementing safeguards and an iterative logic to continuously improve prediction accuracy is essential to mitigate potential drawbacks.

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: [NA]

Justification: We do not anticipate any direct risks associated with our work. While LLMs are generally susceptible to generating harmful content or spreading misinformation, these concerns fall outside the scope of our work.

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: We cited all the papers, code, data, and models used in our work and included links in the paper to access these resources for review.

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: [NA]

    Justification: We will release the code after the paper is accepted.

    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: Our work did not involve crowdsourcing or research involving human subjects.

    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: Our work did not involve crowdsourcing or research involving human subjects.

Guidelines:

- 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.