ProTran: Profiling the Energy of Transformers on Embedded Platforms

Shikhar Tuli, *Student Member, IEEE,* and Niraj K. Jha, *Fellow, IEEE*

***Abstract*—Recently, automated design of efficient transformer models has attracted significant attention from both industry and academia. However, most works only focus on certain metrics while searching for the best-performing architecture. Furthermore, running traditional, complex, and large transformer models on low-compute edge platforms is a challenging task. In this work, we propose a framework to profile the hardware performance measures for a design space of transformer architectures – ProTran. Our framework can be used in conjunction with state-of-the-art neural architecture search techniques to obtain best-performing models that not only have high accuracy on the given task, but also minimize latency, energy consumption and peak power draw, which are important to most edge deployments.**

***Index Terms*—Embedded platforms, machine learning, transformers.**

I. Introduction

IN recent years, self-attention-based transformer models [1, 2] have achieved state-of-the-art results on tasks that span the natural language processing (NLP), and recently, even the computer vision domain [3]. Increasing computational power and large-scale pre-training datasets has resulted in an explosion in the architecture size of transformer models [4], much beyond the state-of-the-art convolutional neural networks (CNNs). For instance, the Megatron Tuning-NLG [4] has 530B trainable model parameters compared to only 928M trainable model parameters in BiT (which uses ResNet-152 with every hidden layer widened by a factor of four, i.e., ResNet-152x4) [5, 6]. However, such massive transformer architectures are not amenable to be run on mobile edge devices, due to a much lower compute budget and memory size.

Even if such a large model is run on these mobile devices, it would incur an extremely high latency [7]. Smaller models may have reasonable latencies, however, they may still not meet the energy or peak power budget for running on edge devices. This could be due to a limited battery size or an intermittent power supply. Thus, there is a need for profiling and benchmarking the latency, energy and peak power consumption for a diverse set of mobile-friendly transformer architectures. This would aid frameworks to leverage hardware-aware neural architecture search (NAS) [7, 8] techniques to find the optimal architecture that maximizes model accuracy while meeting latency, energy and peak power budgets.

Several works have tried to prune transformer models to reduce the number of trainable model parameters [9, 10]. Some have also proposed novel attention mechanisms to reduce the number of trainable parameters [11, 12, 13]. Others have run NAS in a design space of transformer architectural hyperparameters to obtain efficient architectures [7, 14, 15]. However, most of these works have only shown gains in the number of model parameters or the number of floating-point operations per second (FLOPs). Such works do not consider latency, energy and power consumption in their optimization loop, while searching for the optimal transformer architecture (Wang et al. [7] only consider latency for running around 2000 transformer architectures on certain edge devices, and Li et al. [16] consider only a single FPGA). Thus, there is a need to profile not only the accuracy [8], but also the latency, energy consumption and peak power draw of transformer models on various mobile devices for inclusive design in edge-AI deployments.

Nevertheless, profiling all models in vast design spaces is a challenging endeavor. Hence, in this work, we make the following contributions:

- We implement the FlexiBERT benchmarking framework [8] with its design space of diverse transformer architectures on multiple edge-AI devices, for both training and inference. We measure the latency, energy consumption and peak power draw for the transformer models in the design space. We call this profiling framework that can obtain all hardware performance measures for a design space of transformer architectures as ProTran.
- We leverage ProTran to train a surrogate model that exploits gradient-based optimization using backpropagation of inputs and heteroscedastic modelling [8, 17] to minimize the overall uncertainty in estimation of each measure in our active learning loop.
- We then leverage the surrogate models along with previously proposed performance predictors [8] to run *fast* and *efficient* hardware-aware NAS that gives equivalent performing models in terms of accuracy, but with much lower latency, energy consumption and peak power draw.

The rest of the article is organized as follows. Section II discusses the background and related work in hardware-aware NAS, pruning methods and profiling of transformer models. Section III motivates the need for the ProTran framework using a toy example in our design space.

II. Background and Related Work

This section introduces the relevant background and related works in the fields of hardware-aware NAS, pruning methods and profiling of transformer architectures.

This work was supported by NSF Grant No. —. S. Tuli and N. K. Jha are with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ, 08544, USA (e-mail: {stuli, jha}@princeton.edu).

Manuscript received —; revised —.

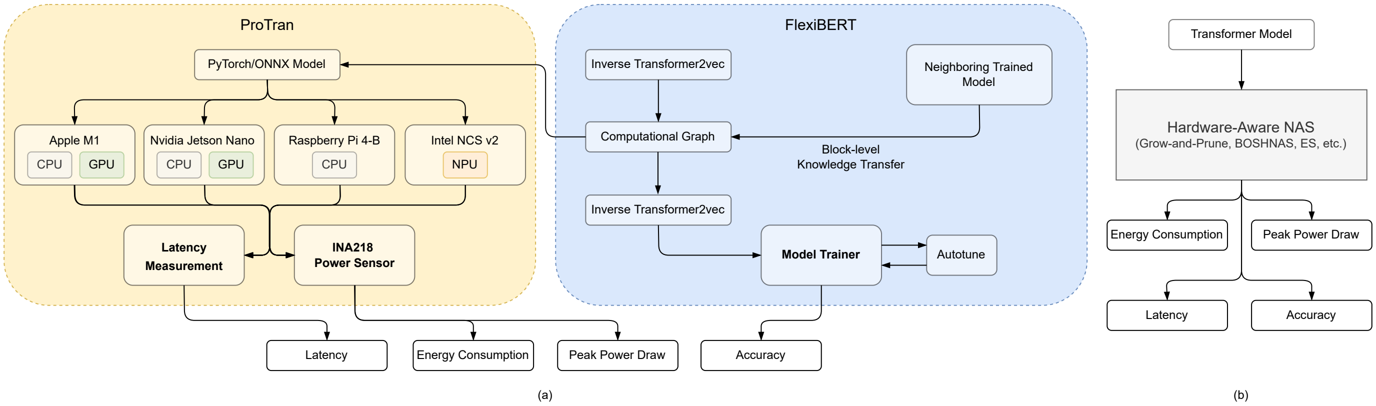

Fig. 1: Overview of the ProTran framework: (a) ProTran being used in conjunction with FlexiBERT for modeling accuracy along with latency, energy consumption, and peak power draw for different embedded platforms (hardware performance measures), (b) hardware-aware NAS exploiting ProTran and FlexiBERT for simultaneous optimization of accuracy with the hardware performance measures.

A. *Transformer Architectures*

Various transformer architectures have been proposed in the past. BERT is one of the most popular transformer architectures that is widely used for language modeling [2]. Its variants leverage mechanisms other than the vanilla self-attention [18] to either optimize performance or reduce model size and complexity. These include – RoBERTa [19] that implements robust pre-training techniques, ConvBERT [20] that uses one-dimensional convolutional operations, MobileBERT [21] that uses bottleneck structures and multiple feed-forward stacks, SqueezeBERT [13] that uses grouped convolution operations to approximate the feed-forward stack, etc. Further, architectures like FNet [11] and LinFormer [12] use the Fourier transform or a low-rank approximation of the self-attention operation to aid efficiency and reduce the number of model parameters.

Many works have tried to device design spaces in order to search for optimal architectural design decisions in a unified manner. For instance, SchuBERT proposes a design space of transformer architectures [22]. However, this work does not consider different types of attention operations and only has *homogeneous* models in its design space. In this work, we leverage FlexiBERT, a state-of-the-art benchmarking framework for diverse transformer architectures, which incorporates most popularly used attention operations in a design space of *heterogeneous* and *flexible* transformer architectures [8]. We model the latency, energy and peak power for transformer architectures on a diverse set of embedded platforms, thanks to the FlexiBERT framework.

B. *Hardware-Aware NAS*

NAS basically searches for the architecture which attains the best accuracy on a specified dataset. However, NAS alone is hardly of much use if the best performing transformer cannot be run on the hardware at hand (or does not meet the hardware performance constraints). Hence, recent works have focused on hardware-aware NAS which directs the architecture search for a target platform. ChamNet proposed accuracy and resource (latency and energy) predictors, and leveraged GP-BO to find the optimal CNN architecture for a given platform [23]. Some works have also proposed *simultaneous* co-design of the hardware and the software design decisions [24, 25, 26]. However, these works are only limited to CNN design spaces.

HAT [7], a recent framework for hardware-aware NAS, trains a large transformer model first and then uses latency feedback to obtain a sub-model for the given hardware platform. However, all sub-models are *homogeneous* and have constant dimensionality on each encoder layer. Further, this work uses a static training recipe, which may not be optimal for every sub-model. Lastly, the design space is highly restricted, which has been shown to lead to marginal gains [26]. Instead, other NAS techniques can be leveraged for superior and efficient search of the optimal model in a diverse set of transformer architectures [8, 23, 27]. Fig. 1 shows how ProTran leverages the FlexiBERT framework (along with a hardware-aware NAS pipeline) to implement block-level grow-and-prune of transformer architectures to obtain the best-performing model in terms of accuracy, latency, energy and peak power consumption.

C. *Energy Profiling of Transformer Models*

Profiling the energy consumption of any ML model is a challenging task. This is because, extracting the energy consumed only by the processes of training or running inference for an ML model is non-trivial. Further, if the design space is large, running training or inference for each model could take drastically long times. Nevertheless, many previous works have profiled the energy consumption of ML architectures. ChamNet trains predictors for the energy consumption for various CNNs in its design space, on different hardware platforms under various latency constraints [23]. FTRANS shows the energy and power consumption for different transformer architectures on an FPGA [16]. However, no NAS approach on transformer architectures has accounted for energy and peak power consumption in the past [7, 8]. Thus, there is a need

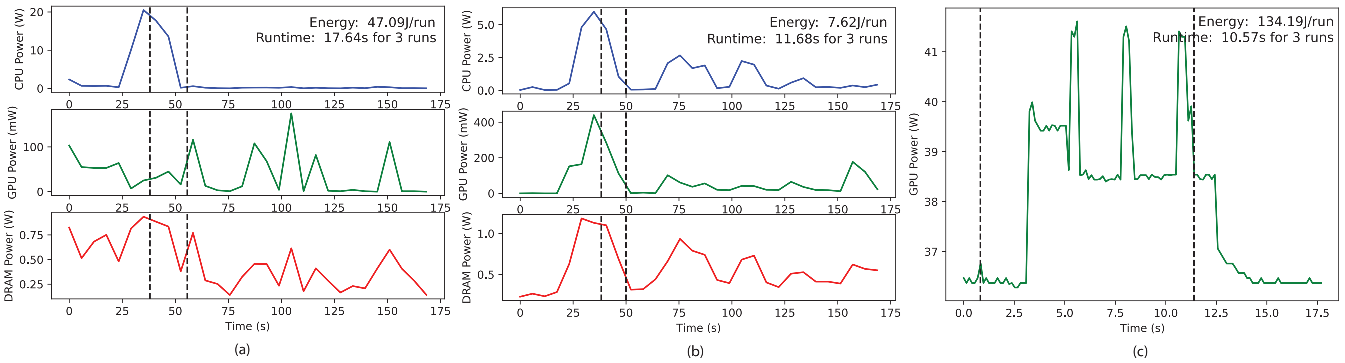

Fig. 2: Power consumption from different sources (CPU, GPU or DRAM) for different platforms: (a) Apple M1 SoC on 8-core CPU, (b) Apple M1 SoC on 8-core GPU, (c) Nvidia A100 GPU. One run corresponds to a full pass of running inference of the BERT-Tiny [28] model on the SST-2 [29] task for the entire dataset.

for light-weight surrogate models for energy and peak power estimation of a diverse set of transformer architectures on various edge-AI platforms. This would not only aid energy-aware NAS for transformer models, but also efficient co-design for optimal edge deployments.

III. Motivation

This section motivates the need for rigorous profiling of transformer architectures on diverse edge-AI platforms and the development of light-weight surrogate models for various hardware performance measures.

A. Energy Reduction on Mobile Platforms

Fig. 2 plots the power consumption for running model inference for BERT-Tiny [28] on the SST-2 task [29], considering different hardware platforms. Figs. 2(a) and (b) show the central processing unit (CPU), graphics processing unit (GPU) and dynamic random access memory (DRAM) power consumption on an iPad (with the Apple M1 SoC) [30]. These power-draw profiles can be compared against that of a regular GPU in Fig. 2(c) (we use Nvidia A100 for the baseline GPU). These figures show that the Apple M1 SoC has much lower power consumption throughout its operation when compared to the A100 GPU.

As can be seen from Fig. 3, the Apple M1 SoC on its integrated GPU outperforms the traditional A100 GPU in terms of energy and peak power consumption, and is also close in terms of latency. We see 17.61× reduction in energy consumption, and 6.56× reduction in peak power draw, while having only 10.58% higher latency per run, for the Apple M1 SoC running on its integrated GPU when compared to the Nvidia A100 GPU. This motivates the need for profiling the latency, energy and peak power consumption of various transformer architectures on a diverse set of embedded platforms. This would aid efficient design of transformer architectural design decisions for different edge deployments.

B. Challenges and Proposed Solutions

The example shown in Fig. 2 only runs inference of a small transformer architecture. For larger architectures, and even for training, much more memory is often required. Such large models (e.g. BERT-Large) may not fit into the memory of the integrated CPU or GPU. To counter this, gradient accumulation over multiple subsets of the mini-batch could be exploited, however, at the cost of increased latency. A challenge that remains is to efficiently search for optimal transformer architectures that can fit into the memory while also not compromising on accuracy, or the need for gradient accumulation that hurts latency. For this, different approaches can be experimented, including NAS, grow-and-prune, etc. Further, quantization and reduction of the activations inside the model could be explored to alleviate memory bottlenecks [7].

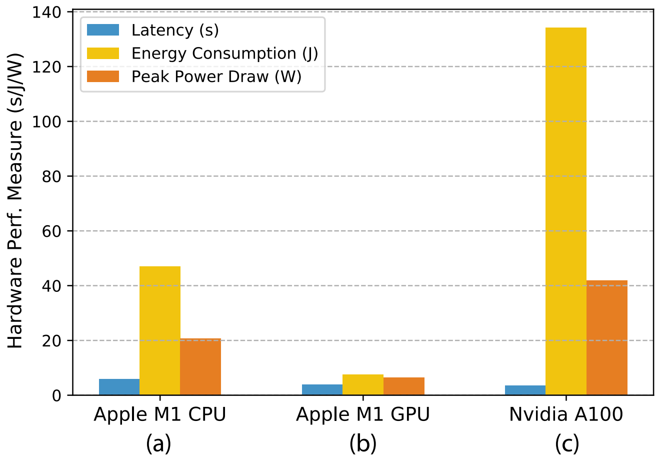

Fig. 3: Gains in different hardware peformance measures of the Apple M1 SoC compared to the Nvidia A100 GPU.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Int. Conf. Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019, pp. 4171–4186.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learning Representations*, 2021.

[4] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zheng, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro, "Using DeepSpeed and Megatron to train Megatron-Turing NLG 530b, A large-scale generative language model," *CoRR*, vol. abs/2201.11990, 2022.

[5] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big Transfer (BiT): General visual representation learning," in *Proc. European Conference on Computer Vision*, 2020, pp. 491–507.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[7] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, and S. Han, "HAT: Hardware-aware transformers for efficient natural language processing," in *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7675–7688.

[8] Anonymous, "FlexiBERT: Expanding the flexibility of transformer architectures and exploring a heterogeneous design space," in *Proc. Int. Conf. on Machine Learning*, 2022, in review.

[9] M. A. Gordon, K. Duh, and N. Andrews, "Compressing BERT: studying the effects of weight pruning on transfer learning," *CoRR*, vol. abs/2002.08307, 2020.

[10] Z. Yan, H. Wang, D. Guo, and S. Han, "MicroNet for efficient language modeling," in *Proc. Int. Conf. Neural Information Processing Systems 2019 Competition and Demonstration Track*, vol. 123, 08–14 Dec 2020, pp. 215–231.

[11] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontañón, "FNet: Mixing tokens with Fourier transforms," *CoRR*, vol. abs/2105.03824, 2021.

[12] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *CoRR*, vol. abs/2006.04768, 2020.

[13] F. Iandola, A. Shaw, R. Krishna, and K. Keutzer, "SqueezeBERT: What can computer vision teach NLP about efficient neural networks?" in *Proc. SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Online: Association for Computational Linguistics, Nov. 2020, pp. 124–135.

[14] J. Xu, X. Tan, R. Luo, K. Song, J. Li, T. Qin, and T.-Y. Liu, "NAS-BERT: Task-agnostic and adaptive-size BERT compression with neural architecture search," in *Proc. 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, p. 1933–1943.

[15] Y. Yin, C. Chen, L. Shang, X. Jiang, X. Chen, and Q. Liu, "AutoTinyBERT: Automatic hyper-parameter optimization for efficient pre-trained language models," in *Proc. 59th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Aug. 2021, pp. 5146–5157.

[16] B. Li, S. Pandey, H. Fang, Y. Lyv, J. Li, J. Chen, M. Xie, L. Wan, H. Liu, and C. Ding, "FTRANS: Energy-efficient acceleration of transformers using fpga," in *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, p. 175–180.

[17] S. Tuli, S. R. Poojara, S. N. Srirama, G. Casale, and N. R. Jennings, "COSCO: Container orchestration using co-simulation and gradient based optimization for fog computing environments," *IEEE Trans. Parallel and Distributed Systems*, vol. 33, no. 1, pp. 101–116, 2021.

[18] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 2, 2018, pp. 464–468.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019.

[20] Z.-H. Jiang, W. Yu, D. Zhou, Y. Chen, J. Feng, and S. Yan, "ConvBERT: Improving BERT with span-based dynamic convolution," in *Proc. Int. Conf. Neural Information Processing Systems*, vol. 33, 2020, pp. 12 837–12 848.

[21] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: A compact task-agnostic BERT for resource-limited devices," in *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020, pp. 2158–2170.

[22] A. Khetan and Z. Karnin, "schuBERT: Optimizing elements of BERT," in *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020, pp. 2807–2818.

[23] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia, P. Vajda, M. Uyttendaele, and N. K. Jha, "Chamnet: Towards efficient network design through platform-aware model adaptation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

[24] Y. Lin, M. Yang, and S. Han, "NAAS: Neural accelerator architecture search," *CoRR*, vol. abs/2105.13258, 2021.

[25] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUNet: Tiny deep learning on IoT devices," in *Proc. Int. Conf. Neural Information Processing Systems*, vol. 33, 2020, pp. 11 711–11 722.

[26] S. Tuli, C. H. Li, R. Sharma, and N. K. Jha, "CODEBench: A neural architecture and hardware accelerator co-design framework," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2022, in review.

[27] X. Dai, H. Yin, and N. K. Jha, "NeST: A neural network synthesis tool based on a grow-and-prune paradigm," *IEEE Trans. Computers*, vol. 68, no. 10, pp. 1487–1497, 2019.

[28] I. Turc, M. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: The impact of student initialization on knowledge distillation," *CoRR*, vol. abs/1908.08962, 2019.

[29] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proc. EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Nov. 2018, pp. 353–355.

[30] Apple. (2020) Apple unleashes M1. [Online]. Available: https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/