A Bayesian Approach for Multi-objective Optimization of Transformers
GitHub link

Negin Firouzian
Department of Electrical and Computer Engineering
McGill University
Montreal, Cananda
negin.firouzian@mail.mcgill.ca

1 Introduction

Transformers (Vaswani et al., 2017) has recently brought significant breakthroughs in many downstream Natural Language Processing tasks, especially with the introduction of pre-trained models such as BERT (Devlin et al., 2018). However, with the increased size and complexity of these models, deploying them on resource-constrained hardware and edge devices has become a real challenge. Designing a model that satisfies both performance metrics and hardware constraints requires searching a large design space of hypermaterepaters. As a result, it demands a lot of effort and experties to find the optimum set of hyperparameters and balance the trade-offs between accuracy metrics and resource efficiency.

To tackle this problem, automated hyperparameter optimization and Neural Architecture Search methods have been widely used to design state-of-the-art deep learning models (Shahriari et al., 2015). The key factor for the efficiency of these approaches is finding an appropriate search strategy; searching the possible hyperparameter in deep learning models is a very time-consuming task. In each evaluation, we have to evaluate the model from scratch, and for large models such as NLP Transformers, it can take up to many GPU days. Therefore it is important to find search algorithms that not only find the optimal points of the search space but also are able to converge to the optimal points with the least evaluations of the objective function. Many search strategies have been used to explore the design space of deep learning models including, evolutionary algorithms, Bayesian Optimization, reinforcement learning.
In this project, we present a multiobjective optimization method that uses a class of evolutionary algorithms, known as probabilistic model-building genetic algorithms (PMBGAs) (Pelikan, 2011), in order to jointly optimize both accuracy and latency of inference in Transformer models by finding a promising set of hyperparameters.
Our contributions in this projects are:

- **Multiobjective Optimization:** We have developed a multiobjective evolutionary optimization algorithm that is able to find the trade-offs between two conflicting objectives and generate an optimal Pareto line for the possible solutions. Multiobjective optimizers are very helpful when we want to maximize the performance of a deep learning model and at the same time satisfy the hardware constraints such as latency, power and memory footprints.

- **Non-Dominated Sorting:** In order to rank the multiples objectives in the search procedure, we have used the Non-Dominated Sorting algorithm (Deb et al., 2000). This algorithm tries to maintain a diverse frontier between multiple conflicting objectives, which results in a more effective and efficient exploration of the search space.

- **Bayesian Learning:** We have employed Bayesian Network during our search procedure in order to efficiently capture the dependencies of variables in promising solutions and generate a similar population for the next generations of search. This algorithm is inspired by Bayesian Optimization Algorithm (BOA) (Pelikan et al., 1999).

- **Domain Adaptivity:** Although this paper aims to find an optimizing algorithm for the search space of NLP transformers, this algorithm could be used in any other single objective or multiobjective task, and it is not limited to optimizing deep learning models.

2 Literature review

One common search approach in NAS is reinforcement learning. In this approach, the generation of neural architectures could be considered as the agents' actions, with the action space being the same as the search space and the RL controller is rewarded according to its accuracy metrics and hardware cost (Elsken et al., 2019). MNASNet (Tan et al., 2019), FPNet (Yang et al., 2019) and Codesign-NAS (Abdelfattah et al., 2020) are examples of papers that have used reinforcement learning as an optimization strategy for NAS.

Another widely used search technique is using evolutionary algorithms, which are the closest category to our approach. These algorithms are heuristic-based and usually start with a sample population and gradually evolve the initial population, generation by generation. In (Lin et al., 2020), authors have developed a two-stage neural architecture search; at first, they optimize the network to fit the hardware resource constraints, and then the vanilla evolution search is performed on the possible settings of neural architecture in order to maximize the accuracy. In our approach, instead of separately optimizing the objectives, we will optimize the accuracy and hardware efficiency jointly as a multiobjective problem and will try to find a Pareto optimal solution. In addition, our evolutionary algorithm is utilizing the Bayesian networks for finding the most elite population in each generation. The closest work to our approach is NSGA-Net (Lu et al., 2019). In this paper, the authors have proposed an evolutionary approach for neural architecture search, and they have used Bayesian networks in order to leverage the entire search history. Similar to our research, they use the Non-Dominated Sorting (Deb et al., 2000) approach to maintain a diverse trade-off frontier between multiple conflicting objectives and perform the efficient exploration of the search space.

All of the aforementioned works on neural network architecture search are mainly focused on CNNs and computer vision applications. To this date and to our knowledge, the only work that is trying to search for efficient transformers for NLP tasks is HAT (Wang et al., 2020). In this work, the authors have developed a latency predictor model for transformers and then conduct an evolutionary search with a latency constraint to find an efficient model for specific target hardware. Since they have created a latency predictor model for transformers, there are no recurrent train and test times, and therefore the neural architecture search part does not result in long GPU hours, and they had performed a simple evolutionary search on the predictor model to find the best solution.Other works on the optimization of transformers include NAT (Guo et al., 2019) which is evaluated on CIFAR-10 and ImageNet datasets, not the NLP task. And also The evolved Transformer (So et al., 2019), where they performed a NAS to find a better Transformer architecture but they haven't taken the hardware constraints into account.

Therefore our work is the first attempt at optimizing the NLP Transformers with respect to two objectives of performance and hardware efficiency.

3 Proposed Approach

3.1 Encoding

The first step in developing an optimization algorithm is finding an encoding that could present the input variable properly. After examing possible encoding choices, we ended up using binary encoding. It is also more convenient to learn the structure of the Bayesian network for binary variables. For each input variable, we encoded the possible choices to unique binary codes based on their order.

$$X \in \{4, 8, 16, 32, 64, 128, 256, 512\}$$

$$X_{encoded} \in \{000, 001, 010, 011, 100, 101, 110, 111\}$$

After generating codes for each variable, we concatenated all of the encoded variables as one single binary input for the Bayesian network and optimization algorithms.

In order to keep track of the original values of variables, we developed a data structure called *EncoderDecoder* that stores both original values and encoded values of variables. During the search process, we have to encode and decode the variables multiple times. In that way, each time that we need to navigate from original values to binary or vice versa, we simply query the *EncoderDecoder* instance without doing extra calculations.

3.2 Learning Bayesian network

For effectively utilizing the Bayesian network in our search algorithm, it is important to learn a Bayesian network that could properly encode the dependencies between variables and their structure. In each iteration of the search, we need to properly capture the distribution of promising solutions in order to produce a similar population for

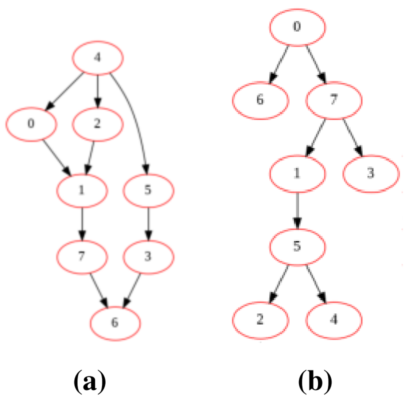

Figure 1: Bayesian network constructed on final generation of search by **(a)** Greedy heuristic and **(b)** Chow-liu methods

the next generation. Since the dataset of search variables is relatively small compared to large machine learning tasks, we decided to use the score-based method with the Bayesian score to generate more reliable structures. Three structure search algorithms, including (1) local search, (2) greedy heuristic and (3) Chow-Liu Trees, were examined to learn the Bayesian network structure. The local search method was very slow compared to other options. However, both greedy heuristic and Chow-Liu methods produced efficient structures; therefore, we decided to evaluate both methods on our search algorithm. Figure 1 shows the constructed Bayesian network in the last iteration of Transformer optimization, using Greedy and Chow-liu methods.

Algorithm 1 Optimization Algorithm

```
1:  $t \leftarrow 0$
2:  randomly generate initial population P(0)
3:  while not done do
4:      evaluate P(t);
5:      rank members of P(t) and select population
        of promising solutions, S(t);
6:      build Bayesian network B(t) with on S(t);
7:      sample new off-springs, O(t) from B(t);
8:      incorporate O(t) into P(t) and create the new
        population P(t+1);
9:      $t \leftarrow t + 1$
10: end while=0
```

3.3 Optimization algorithm

The overall procedure of our proposed optimization algorithm is available in Algorithm 1. This algorithm is inspired by Bayesian Optimization Algorithm (BOA) (Pelikan et al., 1999) but has few modifications from the original algorithm. The algorithm starts by generating a uniformly random population of solutions. This population would be a small portion of the whole search space, in our case around 10%. After sampling a random population, the optimization loop starts by evaluating and ranking the population in the current timestamp. Our algorithm could optimize multiobjective problems. In these problems instead of a single solution, there are sets of optimal solutions known as fronts. These fronts are sets of solutions in which every member of the set does not dominate other members. the most optimal front is known as the Pareto front and finding the most optimal Pareto front is the goal of multi-objective optimization. We have used the Non-Dominated Sorting algorithm (Deb et al., 2000) to find the front lines of the population and select the solutions in the Pareto front line of the search space as the optimization result. Figure 2 shows the final ranked fronts of Transformer optimization problem. After sorting the population, we select the top 50% promising solutions and learn a Bayesian network on them. This network is then used to generate a new offspring based on the learned distributions. In the next step, new samples are incorporated into the promising solutions that were selected in the previous step. The ratio of offsprings is 0.7 of the whole population size. After creating the next generation of population, we repeat the whole procedure of evaluating, ranking, learning and sampling again and again until we reach the stopping criteria.

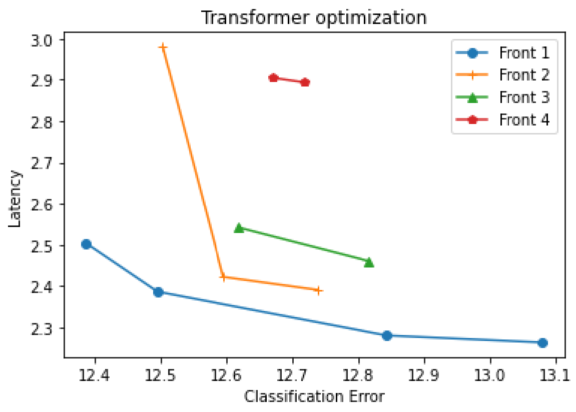

Figure 2: Front lines generated by the optimization algorithm on Transformer function and using greedy heuristic for learning the Bayesian network.

4 Experiments and results

We first evaluated our optimizing algorithm on Binh and Korn function (Binh and Korn, 1997) as a benchmark and fine-tuned our algorithm on that functions. Since evaluations in Transformers are expensive and long, we wanted to fully optimize the search algorithm and then test the final algorithm on the Transformer optimization problem. The parameter to tune for the optimization algorithm was the number of iterations, the ratio of new samples and old population recombination and population size. After examining different settings, we concluded that with only four iterations and population size of 0.1SEARCH_SPACE_SIZE, the algorithm could find promising solutions close to the true Pareto front line. Figure 3 shows the Pareto front generate by our algorithm and NSGA II (Deb et al., 2000), which is the multiobjective version of the vanilla genetic algorithm. According to this figure our algorithm has been able to generate on par and in some points better solutions compared to the widely used genetic algorithm.

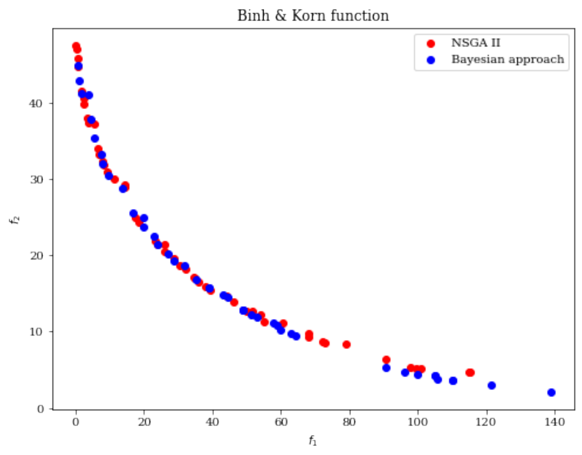

Figure 3: Comparison of NSGA II and our Bayesian approach on Binh & Korn benchmark.

After developing a reliable optimization algorithm, we finally employed Transformers optimization as the input problem. In this problem, the objectives were classification error (100 - accuracy) and latency of inference on the test set. The goal was to minimize both of the objectives and find the Pareto front line of the problem. Our search space includes:

$$EmbeddingSizes \in \{2, 4, 6, 8, 10, 16, 32, 64\}$$

$$HeadNumbers \in \{1, 2, 3, 4\}$$

$$IntermediateSize \in \{2, 4, 6, 8, 10, 16, 32, 64\}$$

Due to our limited GPU resources and time, we weren't able to explore a bigger search space, but the results were still representative in our selected small dataset. The total possibilities of the search space were 256 combinations. We only selected 30 random samples as the initial population, and half the top-ranked population was passed to the search algorithm. We repeated our search procedure for four generations, and we were able to find the optimal frontier solution by only evaluating 90 samples from the Transformer function. This is 2.8x faster than the simple local search. We could even further speed up the search process by saving the results of objective functions in each evaluation so that we would not need to query the objective function for the repeated solutions.

As mentioned earlier we employed both greedy heuristic and Chow-Liu methods for constructing the Bayesian network. Figure 4 shows the Pareto optimal front generated by both of these methods. Except for one point, the greedy approach has produced better solutions in the search space compared to Chow-Liu. This is because the Bayesian structures learned by the Chow-Liu method are more approximate and therefore the convergence speed of the optimization algorithm would be slower compared to the greedy method.

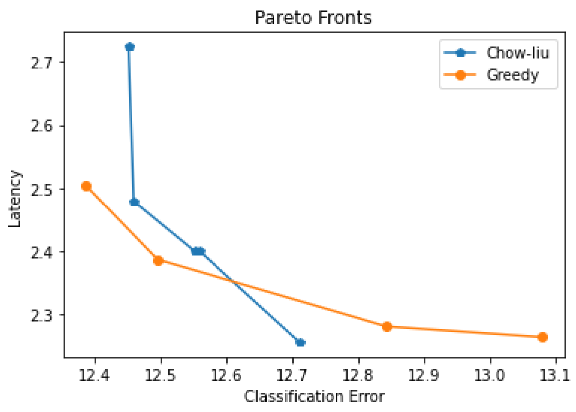

Figure 4: Comparison of the effect of two Bayesian structure search methods on the Pareto front of search algorithm.

5 Conclusion and future works

In this project, we presented an evolutionary optimization algorithm that utilized graphical models, namely Bayesian networks, to guide the search through the design search space. Although our final algorithm is evaluated on Transformers, the algorithm themself cannot be limited to this task and can be employed on different optimization tasks. Our algorithms have been able to successfully find the trad-offs between conflicting objectives and find the Paret front set for the given problems. Therefore this algorithm could be useful in hardware-aware neural architecture search tasks, where performance is not the only objective and we have to optimize the accuracy and hardware constraints jointly.

As future work, this algorithm could be tested on larger optimization benchmarks with more variables to evaluate its limits and strengths. In addition, we can further increase the search space of the transformer by adding extra variables and increasing their range so that we could explore wider choices of architectures in this model. Other recombination techniques in the process of creating the new generation can also be explored, such as mixing both bayesian network sampling and mutation in each search iteration.

References

Mohamed S Abdelfattah, Łukasz Dudziak, Thomas Chau, Royson Lee, Hyeji Kim, and Nicholas D Lane. 2020. Best of both worlds: Automl codesign of a cnn and its hardware accelerator. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE.

To Thanh Binh and Ulrich Korn. 1997. Mobes: A multiobjective evolution strategy for constrained optimization problems. In *The third international conference on genetic algorithms (Mendel 97)*, volume 25, page 27. Citeseer.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International conference on parallel problem solving from nature*, pages 849–858. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. 2019. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21.

Yong Guo, Yin Zheng, Mingkui Tan, Qi Chen, Jian Chen, Peilin Zhao, and Junzhou Huang. 2019. Nat: Neural architecture transformer for accurate and compact architectures. *arXiv preprint arXiv:1910.14488*.

Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. 2020. Mcunet: Tiny deep learning on iot devices. *arXiv preprint arXiv:2007.10319*.

Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 419–427.

Martin Pelikan. 2011. Probabilistic model-building genetic algorithms. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 913–940.

Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. 1999. Boa: The bayesian optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference GECCO-99*, volume 1, pages 525–532. Citeseer.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. HAT: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7675–7688, Online. Association for Computational Linguistics.

Yang Yang, Chao Wang, Lei Gong, and Xuehai Zhou. 2019. Fpnet: Customized convolutional neural network for fpga platforms. In *2019 International Conference on Field-Programmable Technology (ICFPT)*, pages 399–402. IEEE.