Neural Architecture and Implementation Search

A tool for searching high-quality, target-optimized CNNs

Tejas S Shah, Dr. Callie Hao

***Abstract*—The research pertaining to Neural Architecture Search (NAS) algorithms has resulted in Neural Networks (CNNs/DNNs) out-perform the traditionally designed architectures. A step further in this direction was to generate CNNs that are task-agnostic - i.e., to generate CNNs for any downstream task [1]. However, NAS algorithms generally are target hardware agnostic and thus can require manual effort to adapt the NN to the hardware - which can be a challenging and time consuming process. This project focuses on developing a Software-Hardware Co-Design framework that can leverage the benefits of GenNAS [1] and Ansor [2] - an automatic tensor program generation framework for deep learning applications. Based on the parameters, our framework will generate CNNs optimised for a specific hardware and present a pareto-optimal chart of these CNNs based on Latency and Accuracy.**

***Index Terms*—Neural Architecture Search, Hardware-Software Co-Design**

I. Introduction

NAS generated architectures have performed well in several Computer Vision, Natural Language Processing related tasks (CNNs, RNNs) and are designed to extract architectures specifically for a particular task. However, it is observed that neural architectures are good at extracting patterns from the input data and perform well on different downstream tasks. GenNAS [1] builds on this hypothesis to generate CNNs and RNNs adopting a regression-based proxy task on downstream-task-agnostic synthetic signals for network training and evaluation. GenNAS is observed to give state-of-the-art results while achieving generalizability. Hence this makes a great NAS algorithm for our framework.

Optimally implementing a NN on a specific hardware is equally important as generating high-quality NN. However, adapting the same NN to various hardware platforms is difficult, time consuming and requires high expertise. Ansor [2] generates high-performance code without the need of any manual tuning. The input to Ansor is a Deep Learning Model (PyTorch, ONNX, TF), which is first split into small subgraphs (or Tasks). A task scheduler is utilized to allocate the time resource for optimizing many subgraphs. At each iteration, it picks a subgraph that has the most potential to increase the end-to-end performance and applies various optimizations to it.

Our goal was to develop a framework that integrates these two tools to generate high-quality, target-optimized CNNs. However, because of the way Ansor is designed, it doesn't make it a good fit for our application. This paper discusses the approach, and explains the shortcomings in detail.

II. Related Work

Zheng, et al. [3] designed a framework to automatically codesign CNN and its Hardware accelerator. The paper searches for CNNs in the NASBench search space and for the H/W Accelerator (FPGA) in CHaiDNN search space. They implemented a Reinforced Learning based algorithm to prompt the ”Controller” to generate better CNN architectures. Despite this effort, the FPGA accelerator search space is limited and the framework supports only a very limited set of FPGAs - which is difficult to work with in the contemporary world where the Hardware architectures are evolving fast. Our work leverages Ansor - which supports a multitude of H/W devices and can to-an-extent also estimate performance on custom hardware. Instead of searching for best hardware architecture, Ansor searches for best optimizations to apply for a specific hardware.

III. Methodology

The framework has two main parts: the Generator and the Optimizer-Evaluator. The Generator (GenNAS) is responsible for generating task-agnostic CNNs using NASBench201 [4]. The ’n’ best performing architectures will be passed on to the Optimizer-Evaluator (Ansor) to automatically fine-tune these CNNs and evaluate them to obtain Accuracy (from NASBench201) and Latency on the target.

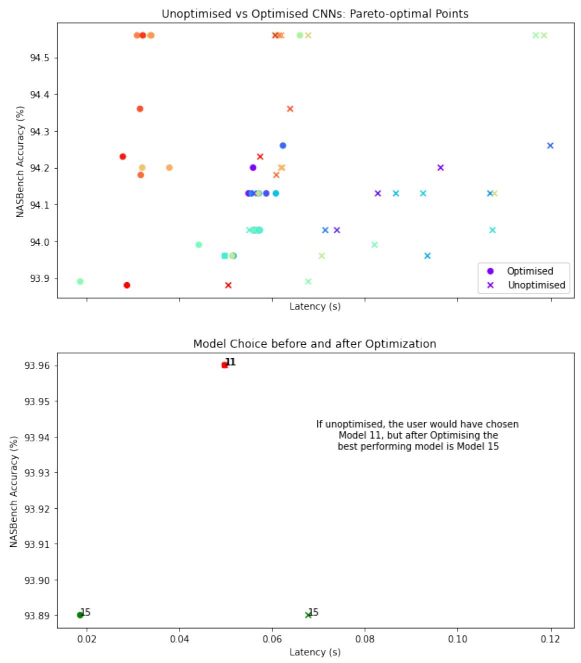

Fig. 1: Visualization generated for Analysis (Search Space: NB101, Tuning Trials: 18000, Avg. Speed-up: 1.87)

This would further be modelled into a Co-Design problem by using Ansor's latency and speed-up information for each model on a sub-graph level. The learning problem was based on the hypothesis that the best performing unoptimised model generated from GenNAS (metric: latency), is not necessarily the best model after optimisation. To validate this, 50 CNNs (20 on NASBench101 and 30 on NASBench201) were tuned and it was observed that the hypothesis was valid.

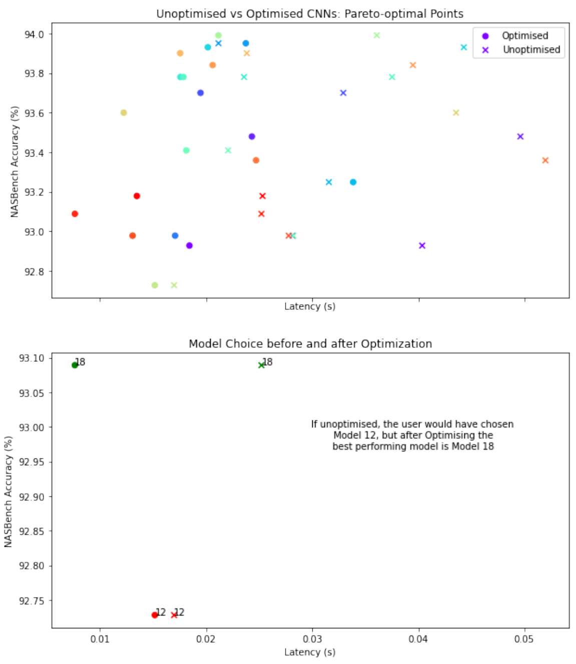

Fig. 2: Visualization generated for Analysis (Search Space: NB201, Tuning Trials: 15000, Avg. Speed-up: 1.73)

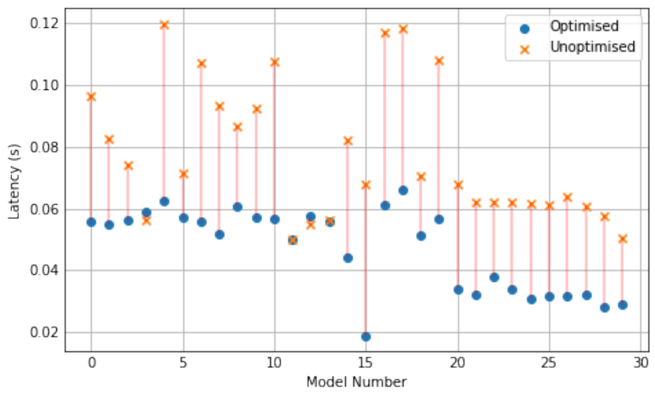

Fig. 3: Visualization showing Speed-up for each model

This visualization has two plots. The first plot marks the latencies of optimised and un-optimised models against Accuracy to generate pareto-optimal points. The second plot is generated by first picking the best un-optimised model (lowest latency) and it's corresponding optimised model. Then the best optimised model and it's corresponding un-optmised model is picked and plotted. It can clearly be observed that the presumed ”best” model (unopt.) isn't necessarily the best post optimisation.

This implied that the GenNAS could be ”trained” to generate CNNs that are more likely to be optimised the best. However, it was observed that the latencies reported by Ansor were very specific to the hardware resources available for tuning, and evaluation - meaning, we couldn't use the latencies as a metric to train GenNAS as the latencies could vary extremely depending on the available CPU cores/workload. Hence Ansor couldn't be used for this application.

IV. Results and Conclusion

1) The pipeline to generate and fine-tune CNNs for specific hardware targets was designed and tested successfully. The process to generate a batch of 16 optimized CNNs require 50 hours of 24-core CPU compute time.
2) The mean speed-up for CNNs was 1.7x for 50 CNN models for an Intel CPU. It can be ovserved that about 50 hours of compute time is required to get just 1.7x speed-up.
3) Ansor evaluates the tuned CNN on real hardware. Hence latencies for these are highly depended on available hardware at that instant. This implies that the latency information cannot be used to train GenNAS.
4) Ansor, although a great tool, does not fit our purpose. We require a tool that can give consistent latency information, and takes much lesser time than Ansor to tune the network. Since we pass a batch of CNNs, they have a lot of similar layers. Thus, a smarter version of Ansor could apply a previously found optimization to a certain subgraph of a new CNN without re-tuning for it. Such methods could speed-up this process.

References

[1] Y. Li, C. Hao, P. Li, J. Xiong, and D. Chen, “Generic neural architecture search via regression,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[2] L. Zheng, C. Jia, M. Sun, *et al.*, “Ansor: Generating {high-performance} tensor programs for deep learning,” in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 863–879.

[3] M. S. Abdelfattah, Ł. Dudziak, T. Chau, R. Lee, H. Kim, and N. D. Lane, “Best of both worlds: Automl codesign of a cnn and its hardware accelerator,” in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2020, pp. 1–6.

[4] X. Dong and Y. Yang, “Nas-bench-201: Extending the scope of reproducible neural architecture search,” *arXiv preprint arXiv:2001.00326*, 2020.