

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy[8]

October 28, 2020

Content

Introduction

Background

Network Architecture Search (NAS)

Reinforced Learning Based Methods

Evolutionary Algorithm

Methodology

Experiment

Reference

Introduction

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

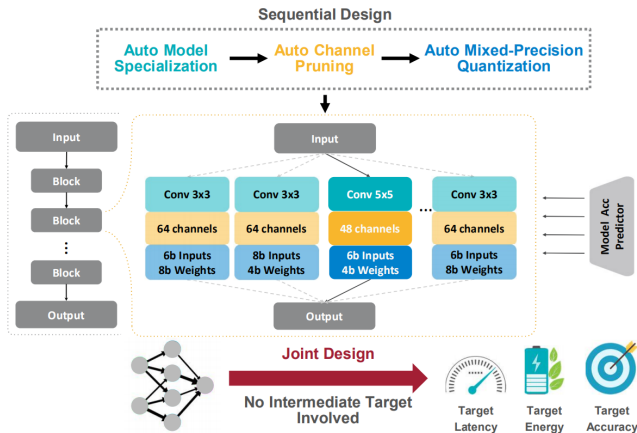


Figure: Comparison between sequential Design and Joint Design

Introduction

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

Architecture: Network Architecture Search (NAS)

Pruning: Channel Level Pruning, Automated Selection of Channel Numbers for each layer

Quantization: Mixed Precision Quantization for each layer

The joint search of multiple targets will result in the explosion of search space

$$A + B + C \longrightarrow A \times B \times C$$

Contribution

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

- ▶ A NAS-pruning-quantization joint search methodology
- ▶ A *predictor-transfer* method to *predict* the accuracy with given structure, pruning and quantization hyper-parameters

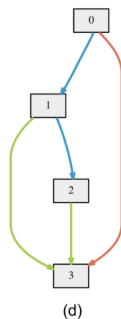
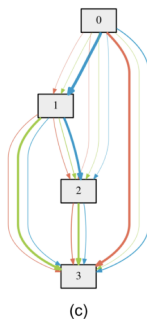
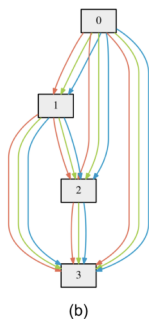
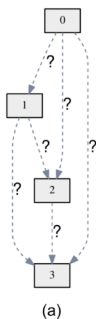
	ProxylessNAS	ChamNet	SPOS	AMC	HAQ	APQ
Hardware-aware	✓	✓	✓	✓	✓	✓
No training during search		✓	✓			✓
No evaluation during search		✓				✓
Channel pruning				✓		✓
Mixed-precision quantization			✓		✓	✓

Figure: Comparison of APQ with ProxylessNAS[2], ChamNet[3], Single Path One-Shot[4], AutoML for Model Compression[5], Hardware-Aware Automated Quantization[7]

Background

Network Architecture Search (NAS)

NAS aims at Automated search for the dedicated hand-crafted network structures, such as the residue structure, bottleneck structure. A typical NAS aims at finding the proper relationship between nodes from a variety of options, which includes convolution, pooling, identity, etc.



Background

Network Architecture Search (NAS)

The strategies used for NAS includes Reinforced Learning, Bayesian Optimization, Evolutionary Algorithm or Gradient-based methods.

The target of NAS can be described with the following equations:

$$\max_{\mathcal{A}} \quad Acc(w^*(\mathcal{A}); \mathcal{A}) \quad (1)$$

$$\text{s.t.} \quad w^*(\mathcal{A}) = \underset{w}{\operatorname{argmin}} \, loss(w; \mathcal{A}), \quad (2)$$

which is a bi-level optimization problem. Evaluating the structure \mathcal{A} involves training the parameter w , making the task extremely computation consuming.

Background

Single Path One-Shot [4]

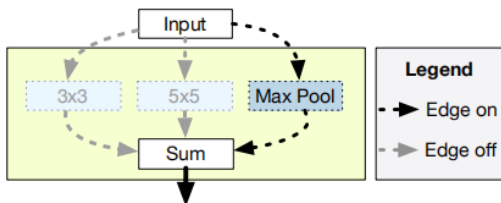


Figure: An example of evaluating the performance of Max Pool from the search space of 3×3 conv, 5×5 conv and max pool. [1]

A SuperNet is a net that contains all the choices in the search space. The SuperNet is trained with all the connections turned on. When evaluating the performance of a particular structure, then only the related connection is turned on. By cascading the blocks, the search space can be expanded exponentially.

Background

Reinforced Learning Based Methods

Both AMC[5] and HAQ[7] use reinforced learning strategy to compress the model size. AMC aims at reducing the channel numbers while HAQ aims at quantizing the model.

Both of them tries to find a flexible scheme that have individual parameters (channel number and quantization bit-width) for each layer and both of them transfer their discrete search space into continuous one by investigating on the prune ratio.

Background

Reinforced Learning Based Methods

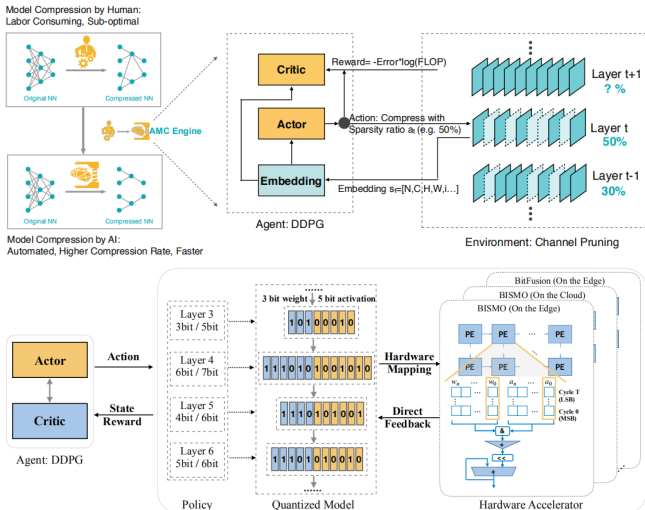


Figure: upper: AMC, lower: HAQ

Background

Reinforced Learning Based Methods

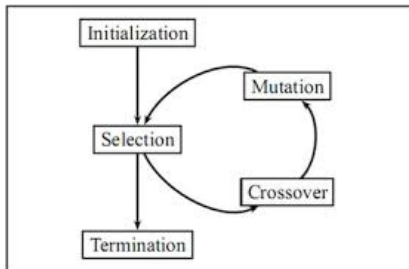
Both of them use the parameter of the layer, such as input width, height, kernel size, as the state and a continuous prune ratio $a \in (0, 1]$ as the action.

Both of them use RL model deep deterministic policy gradient (DDPG) [6], which involves a actor network which generates action for each state and a critic network to grade the action.

Background

Evolutionary Algorithm

In EA, fitter members will survive and proliferate, while unfit members will die off and not contribute to the gene pool of further generations, much like in natural selection.



Outline

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

1. Using the method of Single Path One-Shot, train a full-precision once-for-all network that covers a large search space. All the sub-networks can be extracted from it and the accuracy can be evaluated directly without retraining.
2. Train a quantization-aware precision predicting network using the data obtained above given a structure and quantization policy.
3. Search for the optimal policy using evolutionary under the constrain of a latency/energy lookup-table.

Once-for-all Network

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

Property:

- ▶ Contains a large search space including the network structure and channel numbers, but except the quantization configuration.
- ▶ The accuracy of the sub-network can be evaluated directly using the original weight trained in the once-for-all network.

Why do we except the quantization configuration from the search space?

Since adding the quantization configuration will largely expand the size of searching space ($10^{19} \rightarrow 10^{35}$), which will degrade the performance to evaluate accuracy directly.

Quantization-aware Accuracy Prediction Network

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

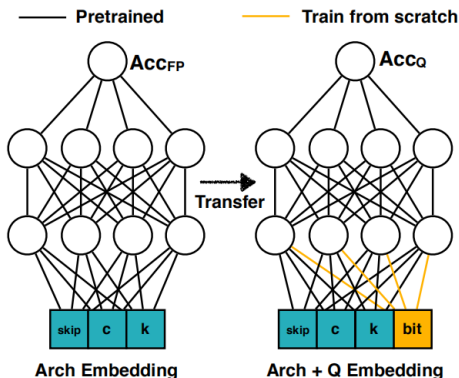


Figure: Transfer Learning of quantization-aware Accuracy Prediction Network

Quantization-aware Accuracy Prediction Network

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

Firstly, a network that predicts the full precision model accuracy is trained. This need 80K samples from the once-for-all network. After that, with 5K quantized samples, the network was transferred to learn on quantized models.

Search with Evolutionary Algorithm

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

Algorithm 1: APQ framework

Input: Pretrained once-for-all network \mathcal{S} , evolution round $iterMax$, population size N , mutation rate $prob$, architecture constraints C .

- 1 Use \mathcal{S} to generate FP32 model dataset \mathcal{D}_{FP} $\langle arch, acc \rangle$ and quantized model dataset \mathcal{D}_{MP} $\langle quantization\ policy, arch, acc \rangle$.
- 2 Use \mathcal{D}_{FP} to train a full precision (FP) accuracy predictor \mathcal{M}_{FP} .
- 3 Use \mathcal{D}_{MP} and \mathcal{M}_{FP} (pretrained weight to transfer) to train a mixed precision (MP) accuracy predictor \mathcal{M}_{MP} .
- 4 Randomly generate initial population \mathcal{P} $\langle quantization\ policy, arch \rangle$ with size N satisfying C .
- 5 **for** $i = 1 \dots iterMax$ **do**
- 6 Use \mathcal{M}_{MP} to predict accuracy for candidates in \mathcal{P} and update Top_k with the candidates having Top-k highest accuracy.
- 7 $\mathcal{P}_{crossover} = Crossover(Top_k, N/2, C)$
- 8 $\mathcal{P}_{mutation} = Mutation(Top_k, N/2, prob, C)$
- 9 $\mathcal{P} = Top_k \cup \mathcal{P}_{crossover} \cup \mathcal{P}_{mutation}$

Output: Candidates with best accuracy Top_k .





Experiment

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy




Model	ImageNet Top1 (%)	Latency (ms)	Energy (mJ)	BitOps (G)	Design cost (GPU hours)	CO ₂ e (marginal)	Cloud compute cost (marginal)
MobileNetV2 - 8bit	71.8	9.10	12.46	19.2	-	-	-
ProxylessNAS - 8bit	74.2	13.14	14.12	19.5	200N	56.72	\$148 – \$496
ProxylessNAS + AMC - 8bit	73.3	9.77	10.53	15.0	204N	57.85	\$151 – \$506
MobileNetV2 + HAQ	71.9	8.93	11.82	-	96N	27.23	\$71 – \$238
ProxylessNAS + AMC + HAQ	71.8	8.45	8.84	-	300N	85.08	\$222 – \$744
DNAS [38]	74.0	-	-	57.3	40N	11.34	\$30 – \$99
Single Path One-Shot [8]	74.6	-	-	51.9	288 + 24N	6.81	\$18 – \$60
Ours-A (w/o transfer)	72.1	8.85	11.79	13.2	2400 + 0.5N	0.14	\$0.4 – \$1.2
Ours-B (w/ transfer)	74.1	8.40	12.18	16.5	2400 + 0.5N	0.14	\$0.4 – \$1.2
Ours-C (w/ transfer)	75.1	12.17	14.14	23.6	2400 + 0.5N	0.14	\$0.4 – \$1.2

Figure: The experiment result of APQ, compared with other benchmarks. N means the cost of deploy once.

Reference I

-  Gabriel Bender et al. “Understanding and simplifying one-shot architecture search”. In: *International Conference on Machine Learning*. 2018, pp. 550–559.
-  Han Cai, Ligeng Zhu, and Song Han. “Proxylessnas: Direct neural architecture search on target task and hardware”. In: *arXiv preprint arXiv:1812.00332* (2018).
-  Xiaoliang Dai et al. “Chamnet: Towards efficient network design through platform-aware model adaptation”. In: *Proceedings of the IEEE Conference on computer vision and pattern recognition*. 2019, pp. 11398–11407.
-  Zichao Guo et al. “Single path one-shot neural architecture search with uniform sampling”. In: *arXiv preprint arXiv:1904.00420* (2019).

Reference II

-  Yihui He et al. “Amc: Automl for model compression and acceleration on mobile devices”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 784–800.
-  Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).
-  Kuan Wang et al. “Haq: Hardware-aware automated quantization with mixed precision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 8612–8620.
-  Tianzhe Wang et al. “APQ: Joint Search for Network Architecture, Pruning and Quantization Policy”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2078–2087.