Towards Real-Time 3D Object Detection with Pruning Search on Edge Devices

Pu Zhao, Wei Niu, Geng Yuan, Yuxuan Cai, Bin Ren, Yanzhi Wang, Xue Lin

Northeastern University
The College of William & Mary

Motivation

- Object detection (especially 3D detection to deal with LiDAR data) serves as a fundamental prerequisite for autonomous navigation
- It is essential to implement real-time 3D object detection on autonomous vehicles
- DNN based 3D object detectors usually cost tremendous memory and computation resources
- Powerful high-end GPUs usually result in significant increasing price and power consumption
- Desirable to implement real-time 3D object detection on edge devices

Contribution

- ❏ Incorporate DNN latency constraint into pruning search to satisfy a predefined real-time requirement
- ❏ Configure various pruning schemes and pruning rates for various layers, different from previous works with fixed pruning scheme for all layers
- ❏ Adopt an ensemble of neural predictors and Bayesian optimization (BO) to reduce the number of evaluated proposals, leading to less searching efforts
- ❏ We can achieve (close-to) real-time (98ms) 3D detection with PointPillars, on an off-the-shelf mobile phone with minor (or no) accuracy loss

Framework

- A generator generates various pruning proposals
- An evaluator evaluates their detection accuracy and speed performance

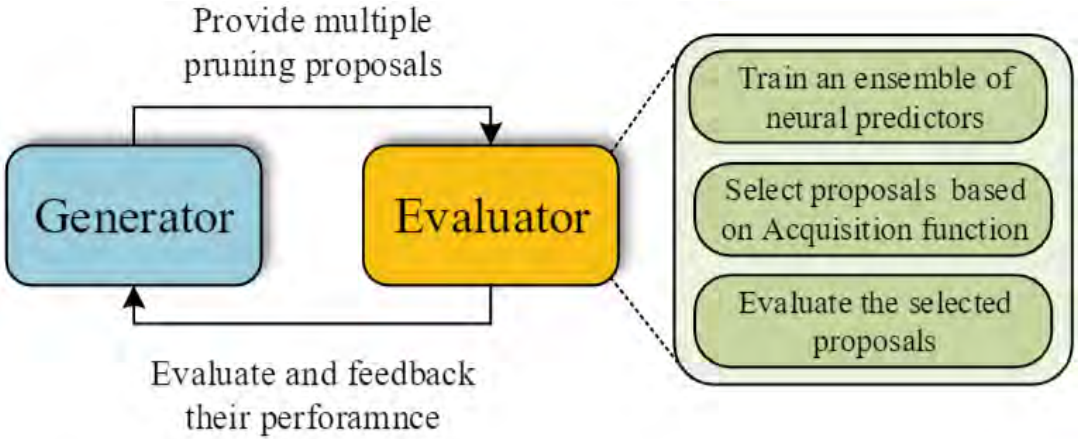


Figure 1. Automatic network pruning search framework

Generator

- The generator samples pruning proposals from the search space
  - Each pruning proposal consists of the layer-wise pruning scheme and pruning rate
  - For example, it has 20 nodes for a 10-layer DNN model
- Proposal Updating
  - mutate the best proposals in the records by randomly changing one pruning scheme or one pruning rate of one layer
- Proposal encoding
  - a binary feature for each possible node in each layer, denoting whether the node (pruning scheme or pruning rate of certain layer) is adopted or not
  - simply check which pruning scheme or rate for each layer is applied, and set the corresponding features to 1s

Table 1. Search space for each DNN layer

Pruning scheme	{Filter, Pattern-based, Block-based}
Pruning rate	{ 1×, 2×, 3×, 5×, 7×, 10×, 15× }

Evaluator

- Evaluate proposal performance
- Performance measurement (Reward)

$$m = V - \alpha \cdot \max(0, r - R),$$

  - take both accuracy and inference speed into consideration
- Adopt Bayesian Optimization (BO) to reduce evaluation efforts
  - use BO to first select a part of proposals with potential better performance
  - do not evaluate the performance of unselected proposals

Evaluator

- Adopt BO to reduce evaluation efforts
- Neural Predictor
  - a neural network repeatedly trained on the current evaluated pruning proposals with their evaluation performance as a predictor to predict the reward of unseen pruning proposals
  - Can be a simple MLP with fully connected layers
- Ensemble of Neural Predictors
  - To incorporate BO, it also needs an uncertainty estimate for the prediction
  - Train multiple neural predictors using different random weight initializations and training data orders

$$\hat{f}(g) = \frac{1}{P}\sum_{p=1}^{P} f_p(g), \text{ and } \hat{\sigma}(g) = \sqrt{\frac{\sum_{p=1}^{P}(f_p(g) - \hat{f}(g))^2}{P-1}}.$$

Evaluator

- Adopt BO to reduce evaluation efforts
- Select proposals with BO
  - Obtain the acquisition function value for proposals in the pool
    - Use upper confidence bound (UCB) as the acquisition function

$$\phi_{\mathrm{UCB}}(g) = \hat{f}(g) + \beta\hat{\sigma}(g)$$

  - Select a small part of proposals with largest acquisition function values
- Evaluate the performance of selected proposals
  - uses magnitude based pruning framework (with two steps including pruning and retraining) to perform the actual pruning
  - test its speed performance on mobile devices with compiler optimization supporting various pruning schemes and rates

Evaluator

- Adopt BO to reduce evaluation efforts

Algorithm 1 Evaluation with predictor ensemble & BO

Input: Observation data $\mathcal{D}$, BO batch size B, BO acquisition function $\phi(\cdot)$
Output: The best pruning proposal g
for steps **do**
  Generate a pool of candidate pruning proposals $\mathcal{G}_c$;
  Train an ensemble of neural predictors with $\mathcal{D}$;
  Select $\{\hat{g}^i\}_{i=1}^{B} = \arg\max_{g \in \mathcal{G}_c} \phi(g)$;
  Evaluate the proposal and obtain reward $\{r^i\}_{i=1}^{B}$ of $\{\hat{g}^i\}_{i=1}^{B}$;
  $\mathcal{D} \leftarrow \mathcal{D} \cup (\{\hat{g}^i\}_{i=1}^{B}, \{r^i\}_{i=1}^{B})$;
end for

Experiments

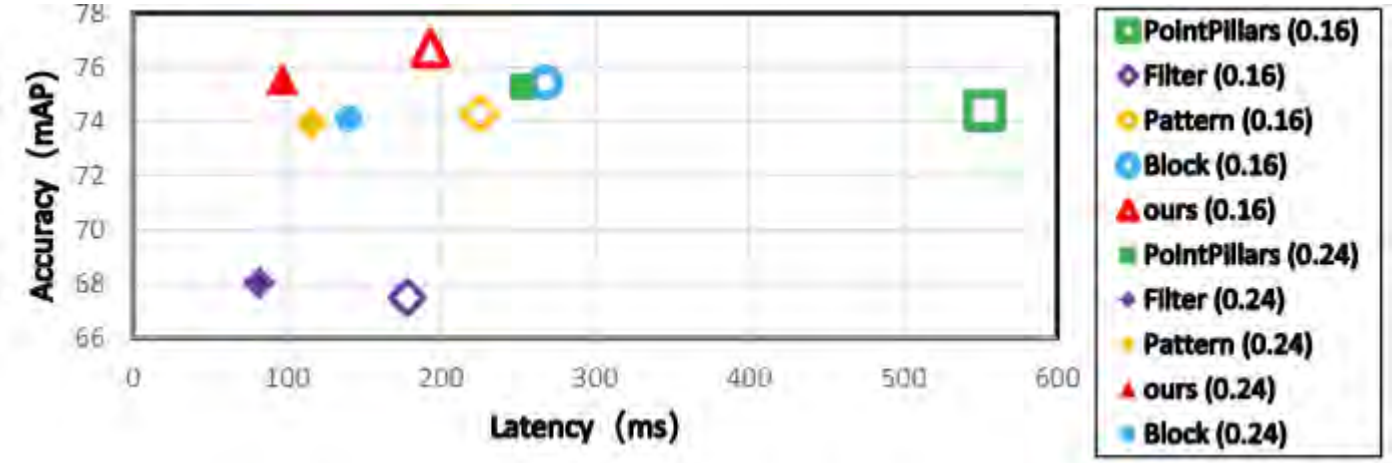
- 3D detection and employ the PointPillars as starting point
- test on KITTI dataset

Table 2. Comparison of various pruning methods for PointPillars

Methods (grid size)	Para. #	Comp. # (MACs)	Speed (ms)	Car 3D detection		
				Easy	Moderate	Hard
PointPillars (0.16)	5.8M	60G	553	85.16	74.39	69.42
Filter (0.16)	1.1M	10.8G	178	80.63	67.51	65.28
Pattern (0.16)	1.1M	10.7G	225	83.64	74.30	68.42
Block (0.16)	1.1M	10.7G	268	82.86	75.43	69.71
Ours (0.16)	1.1M	10.7G	193	**85.52**	**76.69**	**70.10**
PointPillars (0.24)	5.4M	28G	253	84.24	75.28	68.46
Filter (0.24)	0.8M	4.0G	82	81.36	68.06	65.77
Pattern (0.24)	0.8M	3.9G	116	82.16	73.93	68.25
Block (0.24)	0.8M	4.0G	140	83.69	74.09	68.06
Ours (0.24)	0.8M	3.9G	98	**85.38**	**75.72**	**68.53**

Experiments

- ❏ 3D detection and employ the PointPillars as starting point
- ❏ test on KITTI dataset



- ❏ Our method only needs 98ms to process one frame on mobile devices with the highest accuracy