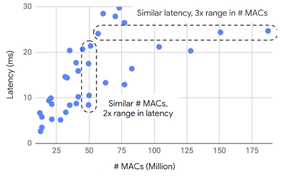
NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications

Tien-Ju Yang[1], Andrew Howard[2], Bo Chen[2], Xiao Zhang[2], Alec Go[2], Mark Sandler[2], Vivienne Sze[1], and Hartwig Adam[2]

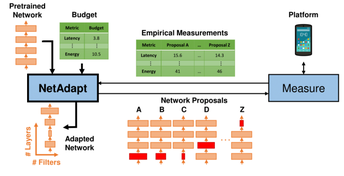
[1]Massachusetts Institute of Technology, [2]Google Inc.

Introduction

- Common problems of network simplification methods:
  - Require **manually** choosing the simplified architecture
  - Guided by **indirect metrics** (e.g., # of MACs), which may not be good approximations to the direct metrics (e.g., latency)
- NetAdapt
  - **Automatically and progressively** finds the simplified architecture
  - Incorporates **direct metrics** in the network simplification process by using real measurements from the **target platform**
  - **Improves** the accuracy-latency trade-offs of highly efficient neural networks, **MobileNet V1/V2**



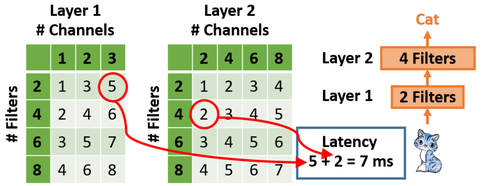
Latency vs. # of MACs (from Google OVIC)



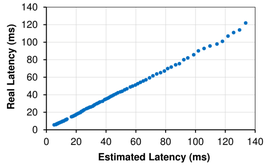
NetAdapt

Fast Resource Consumption Estimation

- Taking measurements can be slow and difficult to parallelize due to the limited number of available devices
- The network latency can be estimated by the sum of the latency of each layer



Fast Resource Consumption Estimation



Real Latency vs. Estimated Latency

NetAdapt Algorithm

- Problem formulation:

$$\max_{Net} Acc(Net) \ \ subject\ to\ Res_j(Net) \le Bud_j, j = 1, \cdots, m$$

Break into a set of simpler problems and solve iteratively

$$\max_{Net_i} Acc(Net_i) \ \ subject\ to\ Res_j(Net_i) \le Res_j(Net_{i-1}) - \Delta R_{i,j}, j = 1, \cdots, m$$

*Acc: accuracy function, Res: resource evaluation function, ΔR: resource reduction, Bud: given budget

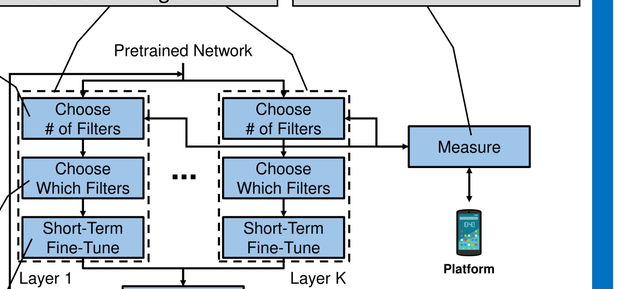
- Algorithm:

Generate multiple network proposals in each iteration: Remove filters from a single block

Use fast resource consumption estimation

Determine **how many** filters to remove to meet the constraint

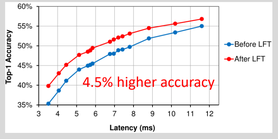
Determine **which filters** to remove: Remove the filters with the smallest l2-norm



Select the network proposal with the highest accuracy

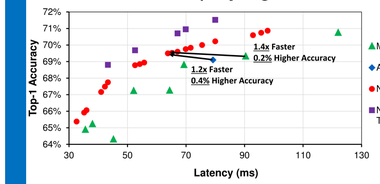
Fine-tune the network with a few iterations before selection; Critical for small networks

Fine-tune the network until convergence

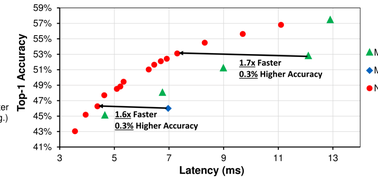


Experiment Results

- NetAdapt is effective on both small and large MobileNet V1 and both MobileNet V1 and V2
  - Small MobileNet V1 is much harder to simplify
- We use the same training configuration for all the networks
  - The training configuration can have a large impact on the accuracy
- Results of simplifying MobileNet V1 on a mobile CPU



Large MobileNet V1



Small MobileNet V1

* Multipliers: A. Howard et al., arXiv 2017; MorphNet: A. Gordon et al, CVPR 2018; ADC: Y. He et al., arXiv 2018
* Better training configuration: add dropout and label smoothing
* Single large core of Pixel 1 CPU

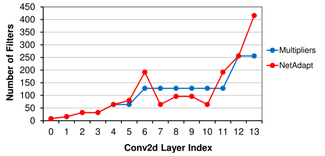
- Early results of simplifying MobileNet V2 on a mobile CPU

Network	Top-1 Accuracy (%)		Latency (ms)	
75% MobileNetV2 (224) [18]	69.8	(+0)	64.5	(100%)
NetAdapt (Similar Latency)	70.9	(+1.1)	63.6	(99%)
NetAdapt (Similar Accuracy)	70.2	(+0.4)	55.5	(86%)

- Using direct metrics (e.g., latency) is important

Network	Top-1 Accuracy (%)		# of MACs ($\times 10^6$)		Latency (ms)	
25% MobileNetV1 (128) [9]	45.1	(+0)	13.6	(100%)	4.65	(100%)
MorphNet [5]	46.0	(+0.9)	15.0	(110%)	6.52	(140%)
NetAdapt	46.3	(+1.2)	11.0	(81%)	6.01	(129%)
75% MobileNetV1 (224) [9]	68.8	(+0)	325.4	(100%)	69.3	(100%)
ADC [8]	69.1	(+0.3)	304.2	(93%)	79.2	(114%)
NetAdapt	69.1	(+0.3)	284.3	(87%)	74.9	(108%)

- The discovered architecture preserves more filters when
  - the resolution is decreased.
  - the layer is followed by the final fully-connected layer.



Small MobileNet V1