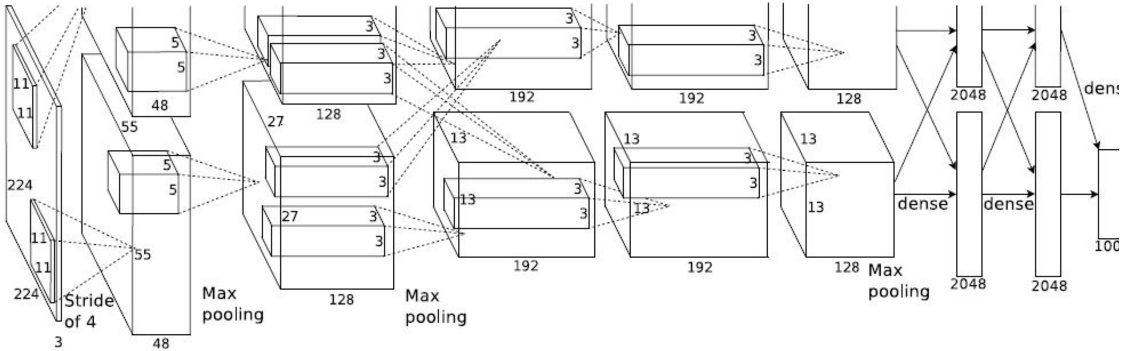
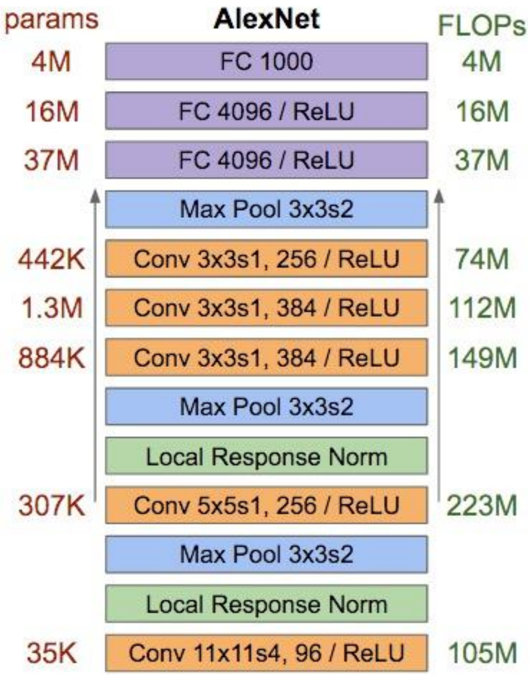
Deep Learning on Edge

Girish Varma
IIIT Hyderabad

Big Huge Neural Network!

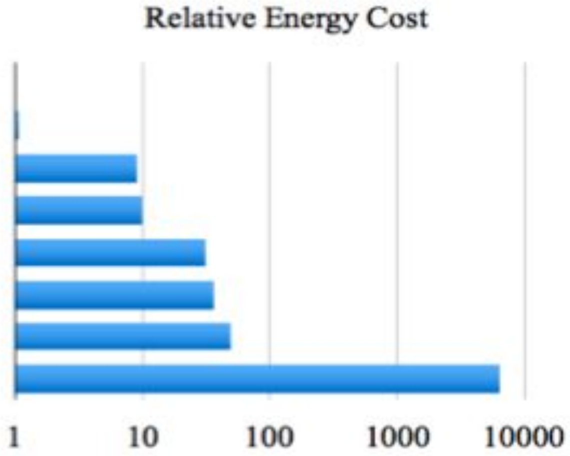


AlexNet - 60 Million Parameters = 240 MB



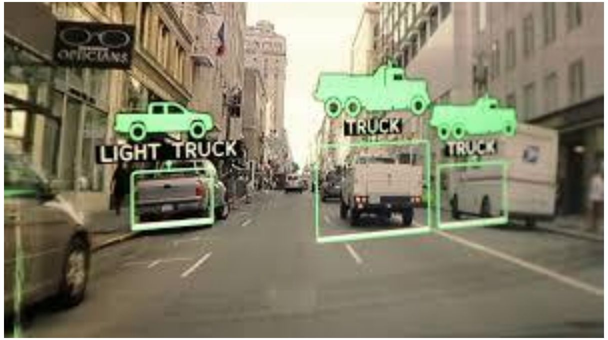
& the Humble Mobile Phone

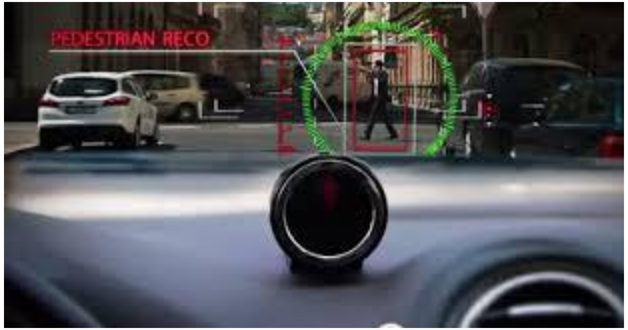
Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	**640**	**6400**



But wait! What about battery life?

Self Driving Cars!





Can we do 30 fps?

ORCAM! : Blind AID

Can we do 30 fps?

Running Model in the Cloud

1. Network Delay
2. Power Consumption
3. User Privacy

Issues on Mobile Devices

1. RAM Memory Usage
2. Running Time
3. Power Usage
4. Download / Storage size

Model Compression

What are Neural Networks made of?

- Fully Connected Layer : Matrices

- Convolutional Layer : Kernals (Tensors)

Reducing Memory Usage

1. Compressing Matrices
    a. Sparse Matrix => Special Storage formats
    b. Quantization
2. Architecture Design

PRUNING

Compressing Matrices by making them Sparse

WHY PRUNING ?

Deep Neural Networks have redundant parameters.

Such parameters have a negligible value and can be ignored.

Removing them does not affect performance.

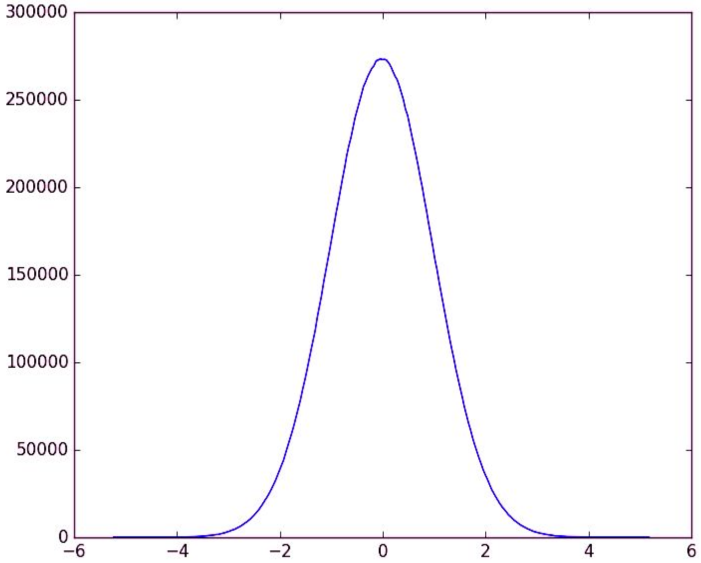
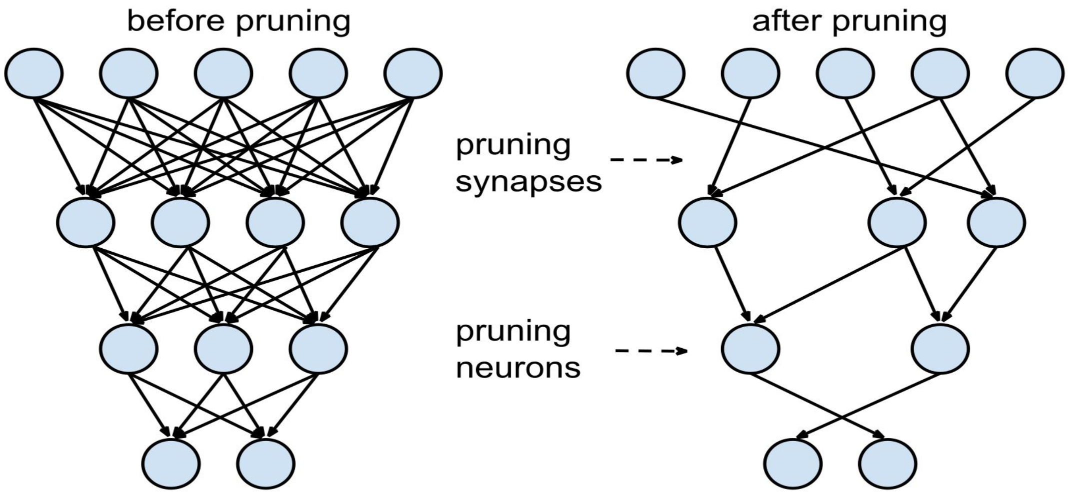


Figure: Distribution of weights after Training

Why do you need redundant parameters?
Redundant parameters are needed for training to converge to a good optima.

Optimal Brain Damage by Yann Le Cunn in 90’s
https://papers.nips.cc/paper/250-optimal-brain-damage

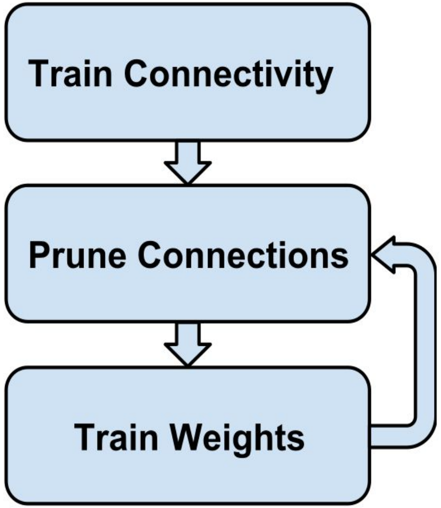
Weight Pruning

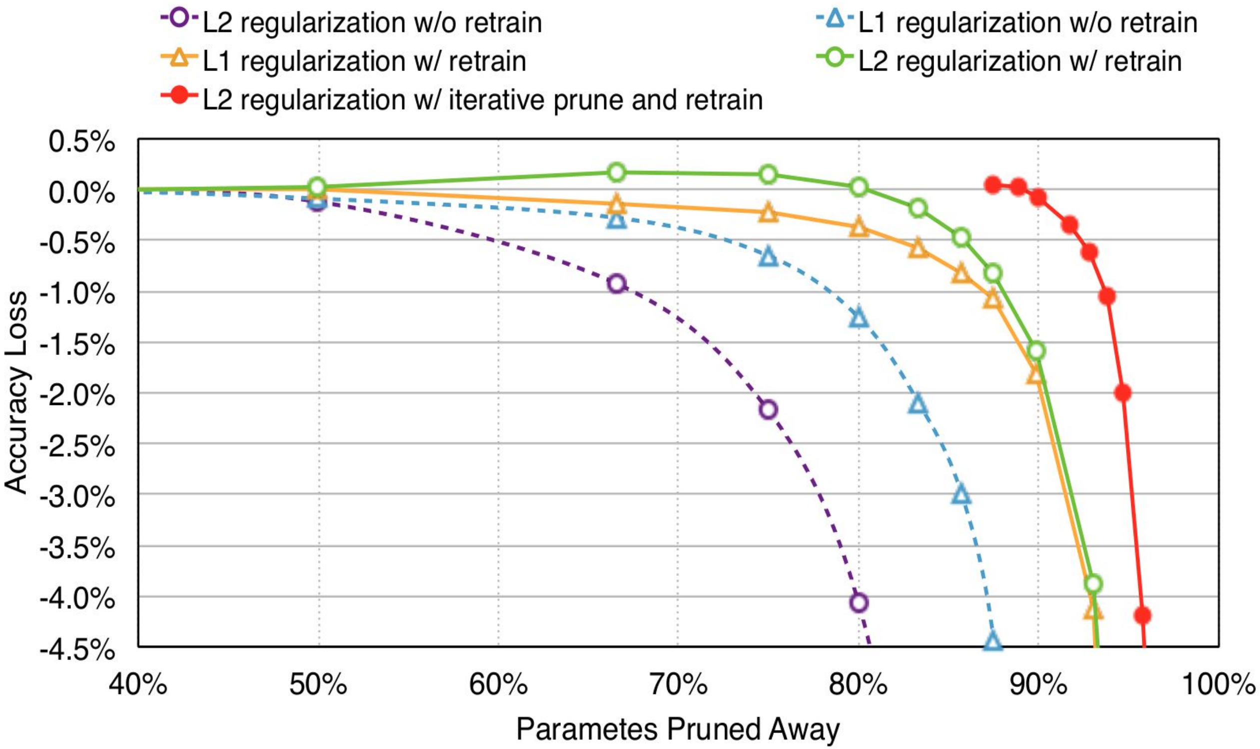


- The matrices can be made sparse. A naive method is to drop those weights which are 0 after training.
- Drop the weights below some threshold.
- Can be stored in optimized way if matrix becomes sparse.
- Sparse Matrix Multiplications are faster.

Sparsify at Training Time

Iterative pruning and retraining



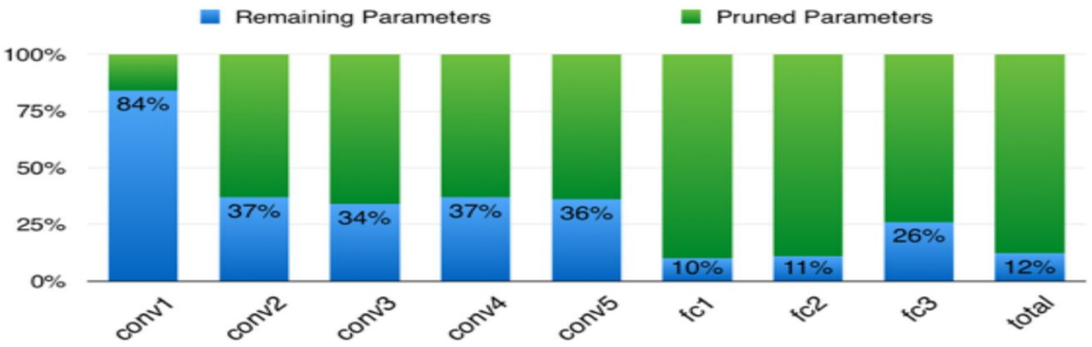


[Learning both Weights and Connections for Efficient Neural Networks]

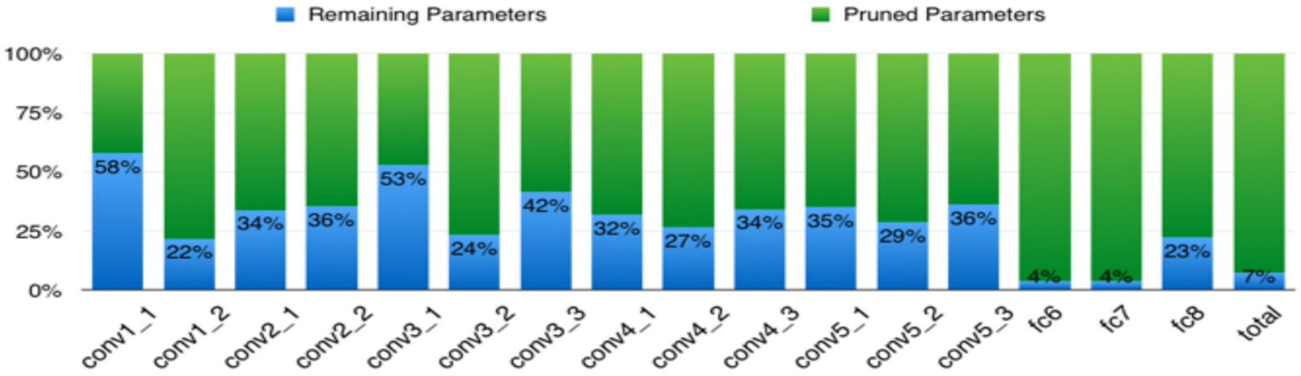
https://arxiv.org/pdf/1506.02626

by S Han - 2015 - [Cited by 233] - [Related articles]

Remaining parameters in Different Layers



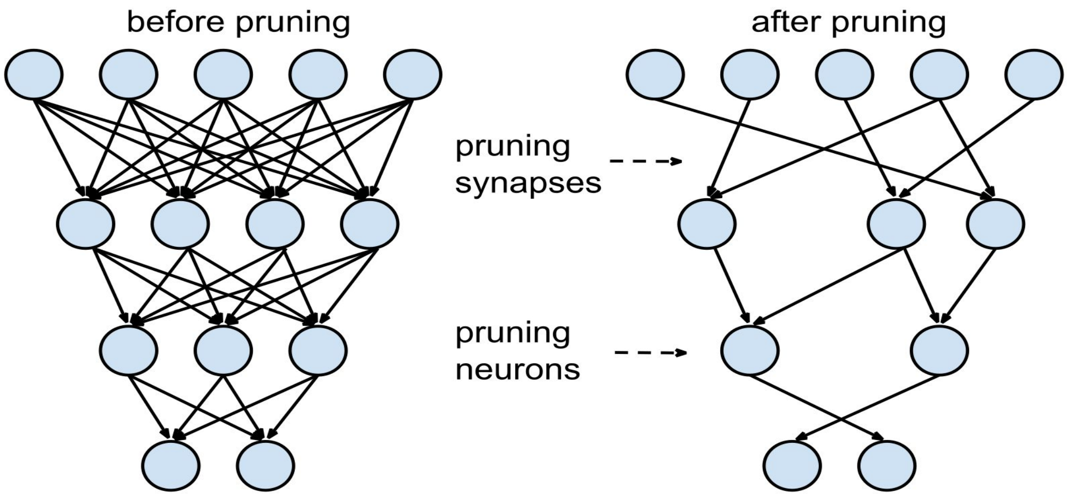
ALEXNET



VGG16

DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING Song Han, Huizi Mao, William J. Dally

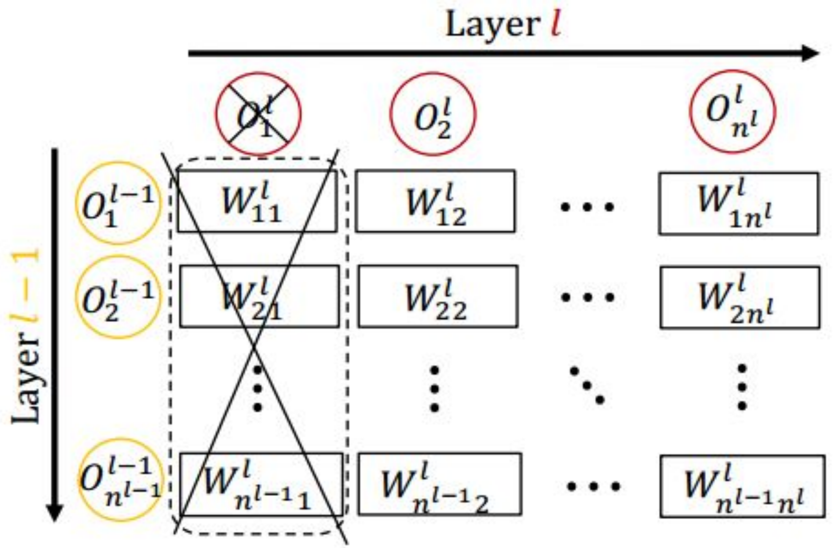
Neuron Pruning



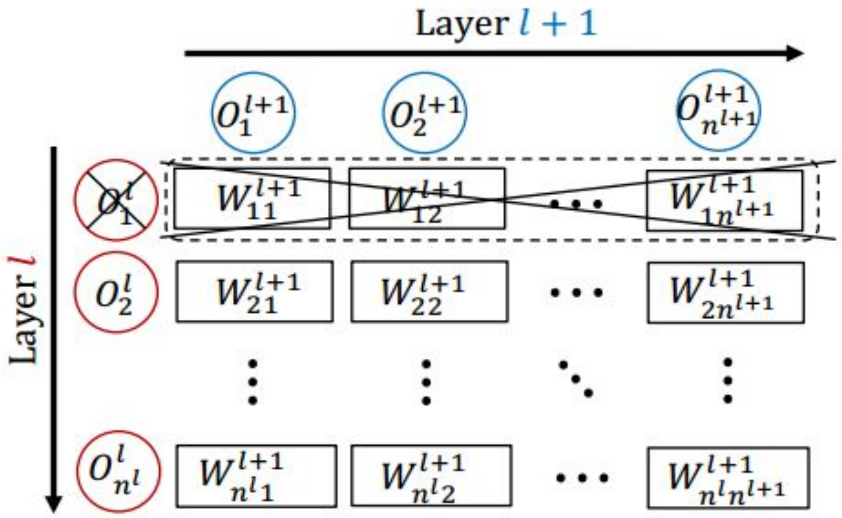
- ➔ Removing rows and columns in a weight matrix.
- ➔ Matrix multiplication will be faster improving test time.

DropNeuron: Simplifying the Structure of Deep Neural Networks Wei Pan, Hao Dong, Yike Guo

Effect of neuron pruning on weight matrices



(c) Removal of incoming connections to neuron O_1^ℓ, i.e., the group of weights in the dashed box are all zeros



(d) Removal of outgoing connections from neuron O_1^ℓ, i.e., the group of weights in the dashed box are all zeros

DropNeuron: Simplifying the Structure of Deep Neural Networks Wei Pan, Hao Dong, Yike Guo

QUANTIZATION

Binary Quantization

$$\hat{W}_{ij} = \begin{cases} 1 & \text{if } W_{ij} \geq 0, \\ -1 & \text{if } W_{ij} < 0. \end{cases}$$

Size Drop : 32X

Runtime : Much faster (7x) matrix multiplication for binary matrices.

Accuracy Drop : Classification error is about 20% on the top 5 accuracy on ILSVRC dataset.

COMPRESSING DEEP CONVOLUTIONAL NETWORKS USING VECTOR QUANTIZATION Yunchao Gong, Liu Liu , Ming Yang, Lubomir Bourdev

8-bit uniform quantization

- Divide the max and min weight values into 256 equal divisions uniformly.
- Round weights to the nearest point
- Store weights as 8 bit ints

Size Drop : 4X

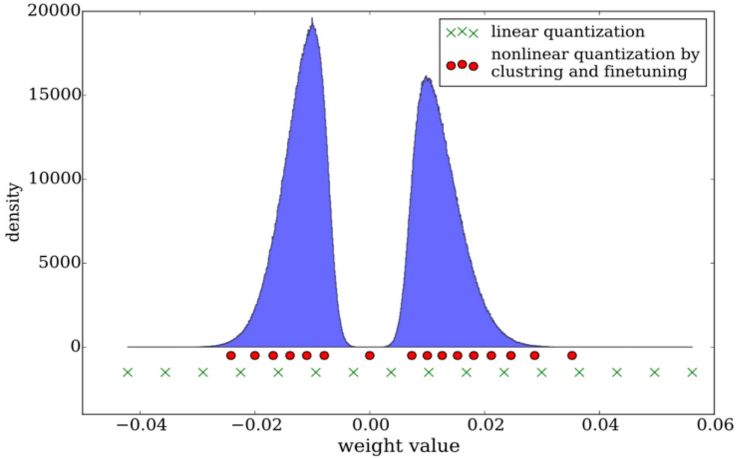
Runtime : Much faster matrix multiplication for 8 bit matrices.

Accuracy Drop : Error is acceptable for classification for non critical tasks

https://petewarden.com/2016/05/03/how-to-quantize-neural-networks-with-tensorflow/

Non Uniform Quantization/ Weight Sharing

$$\min \sum_{i}^{mn} \sum_{j}^{k} \|w_i - c_j\|_2^2,$$

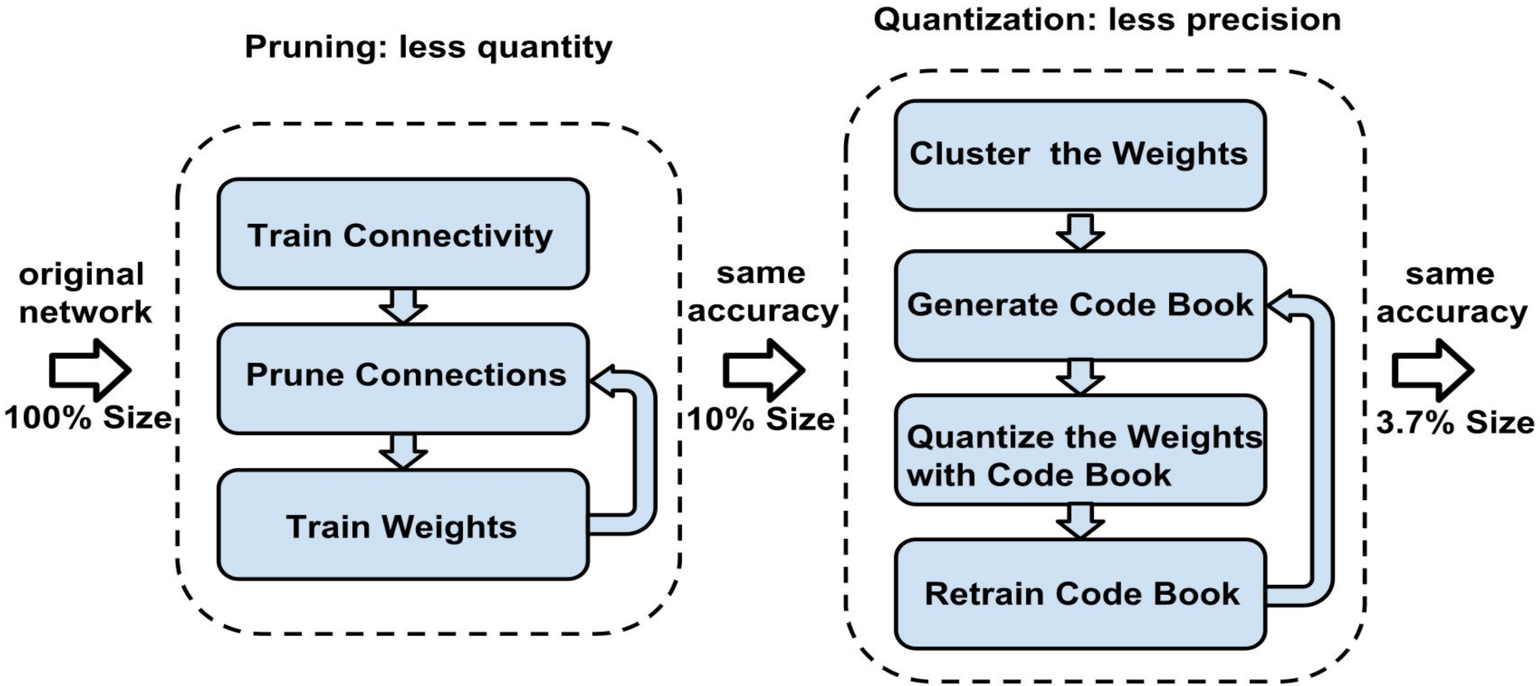


- perform k-means clustering on weights.

- Need to store mapping from integers to cluster centers. We only need log (k) bits to code the clusters which results in a compression factor rate of 32/ log (k). In this case the compression rate is 4.

DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING Song Han, Huizi Mao, William J. Dally

Deep Compression by Song Han



DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING Song Han, Huizi Mao, William J. Dally

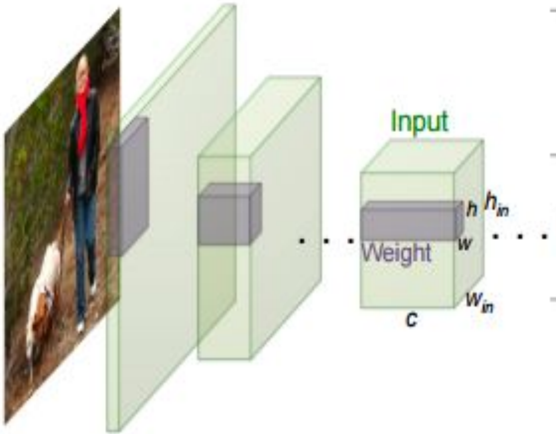
XNOR Net

- ❖ Binary Weight Networks :
  - ➢ Estimate real time weight filter using a binary filter.
  - ➢ Only the weights are binarized.
  - ➢ Convolutions are only estimated with additions and subtractions (no multiplications required due to binarization).

- ❖ XNOR Networks:
  - ➢ Binary estimation of both inputs and weights
  - ➢ Input to the convolutions are binary.
  - ➢ Binary inputs and weights ensure calculations using XNOR operations.

XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks Mohammad Rastegari , Vicente Ordonez , Joseph Redmon , Ali Farhadi

Results

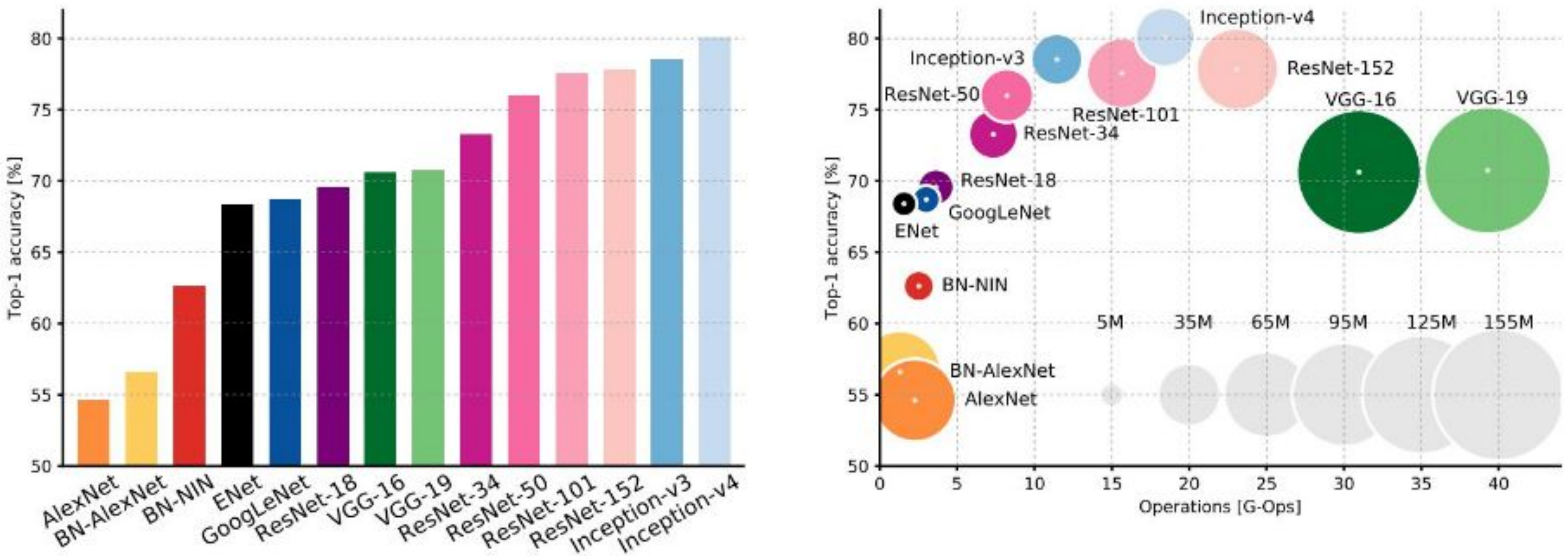


	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights: 0.12 -1.2 0.41 / -0.2 0.5 ... 0.68	+, −, ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights: 1 -1 1 / -1 1... 1	+, −	~32x	~2x	%56.8
BinaryWeight Binary Input (**XNOR-Net**)	Binary Inputs: 1 -1 ... -1 / -1 1 ... 1; Binary Weights: 1 -1 1 / -1 1... 1	XNOR, bitcount	~32x	~58x	%44.2

XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks Mohammad Rastegari , Vicente Ordonez , Joseph Redmon , Ali Farhadi

Efficient DNNs

Performance Tradeoffs



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Design small architectures

Compress scheme on **pre-trained model**

Vs

Design **small CNN architecture** from scratch
(also preserve accuracy?)

GoogLe Net

- First architecture with improved utilization of the computing resources inside the network while increasing size, both depth and width
- 22 layers deep when counting only layers with parameters
- Significantly more accurate than AlexNet
- 12 times lesser parameters than AlexNet.
- Computational cost “less than 2X compared to AlexNet”

Szegedy, Christian, et al. "Going deeper with convolutions." *CVPR*, 2015.

MobileNet from Google

Uses Depth Wise Separable Convolutions.

A formula for achieving good performance tradeoffs.

The computational cost of a depthwise separable convolution with width multiplier α is:

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F \quad (6)$$

Table 3. Resource usage for modifications to standard convolution. Note that each row is a cumulative effect adding on top of the previous row. This example is for an internal MobileNet layer with $D_K = 3$, $M = 512$, $N = 512$, $D_F = 14$.

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

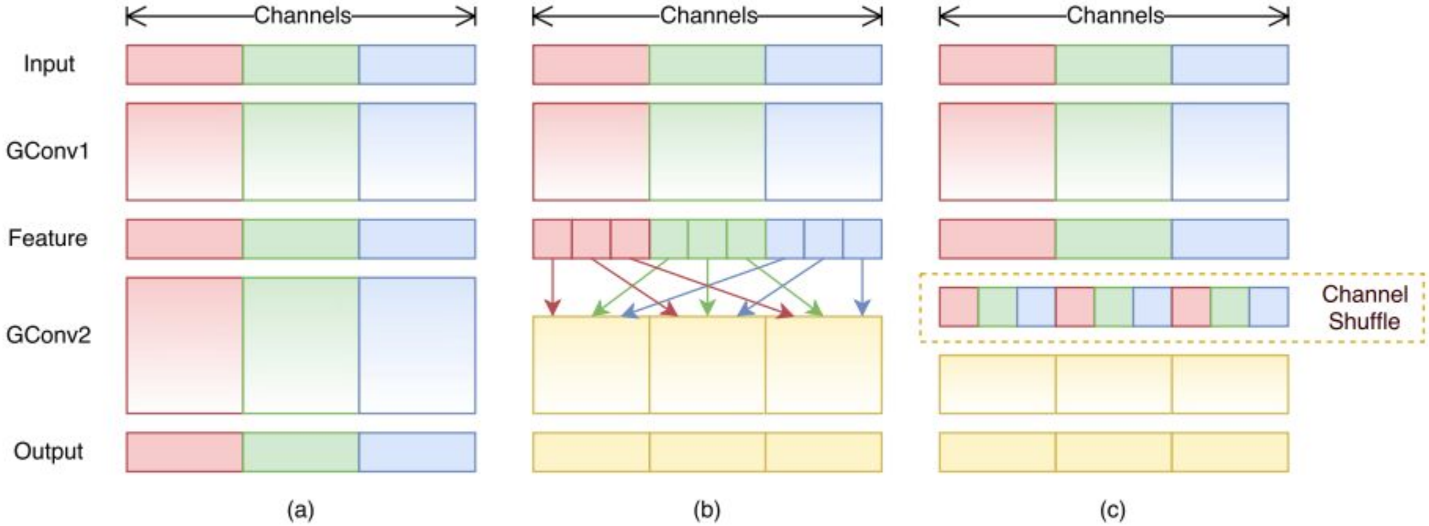
Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

ShuffleNet



Channels
Channels
Channels
Input
GConv1
Feature
GConv2
Output
Channel
Shuffle
(a)
(b)
(c)

ShuffleNet

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet $2\times$ ($g = 3$)	524	**26.3**	3.1
ShuffleNet $2\times$ (with *SE*[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet $1.5\times$ ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet $1\times$ ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet $0.5\times$ ($g = 4$)	38	**41.6**	7.8
ShuffleNet $0.5\times$ (shallow, $g = 3$)	40	42.8	6.6

Table 5. ShuffleNet vs. MobileNet [12] on ImageNet Classification

MobileNetV2 in 2018

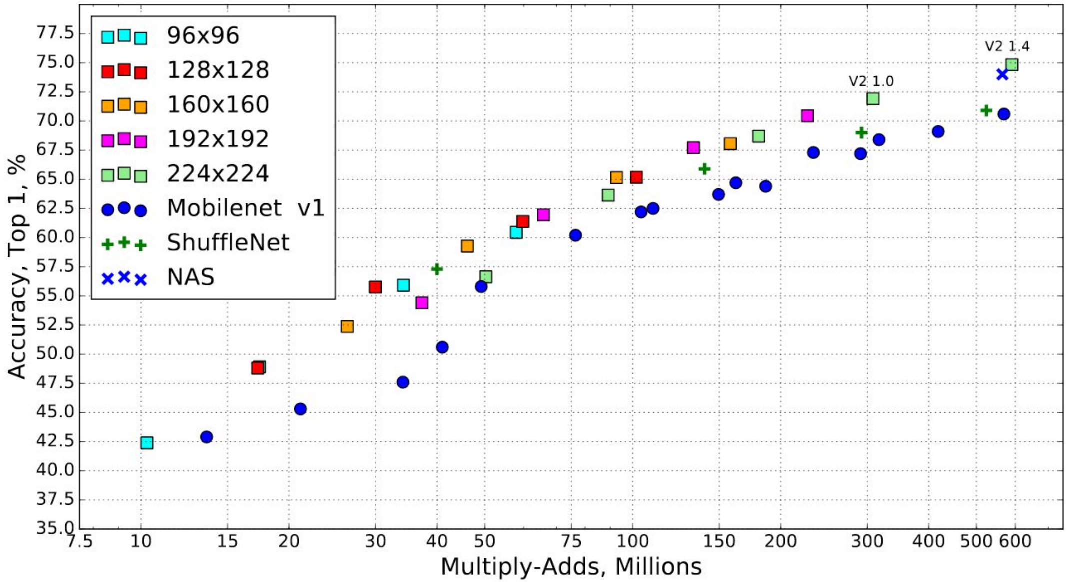
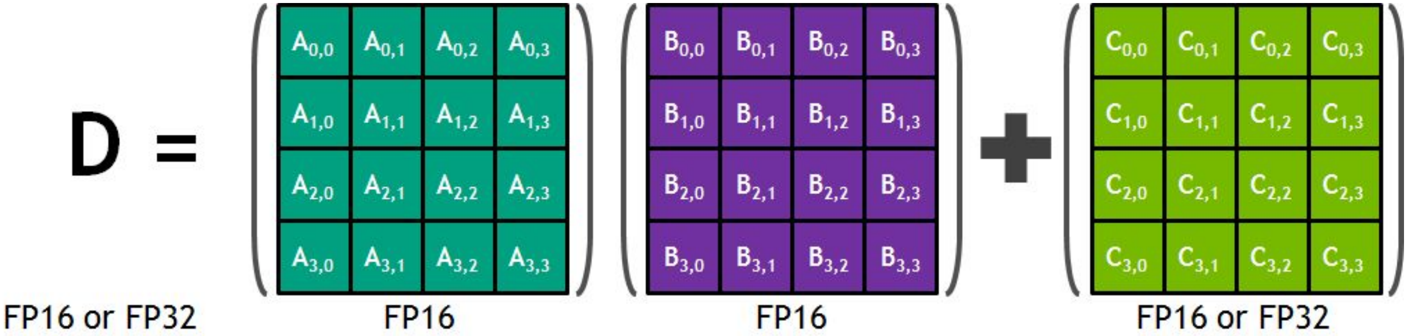


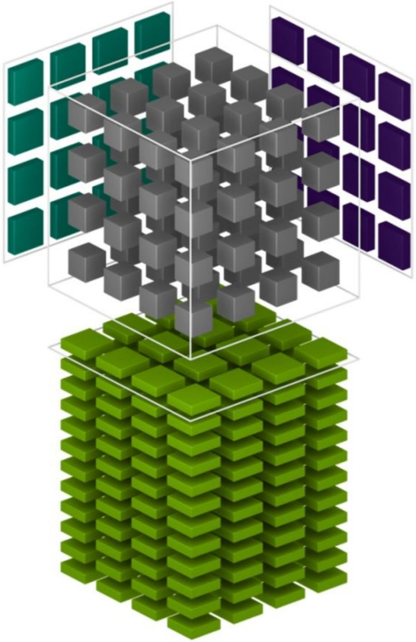
Figure 5: Performance curve of MobileNetV2 vs MobileNetV1, ShuffleNet, NAS. For our networks we use multipliers 0.35, 0.5, 0.75, 1.0 for all resolutions, and additional 1.4 for for 224. Best viewed in color.

Compilers & Hardware Processors

Processors that operate on Matrices

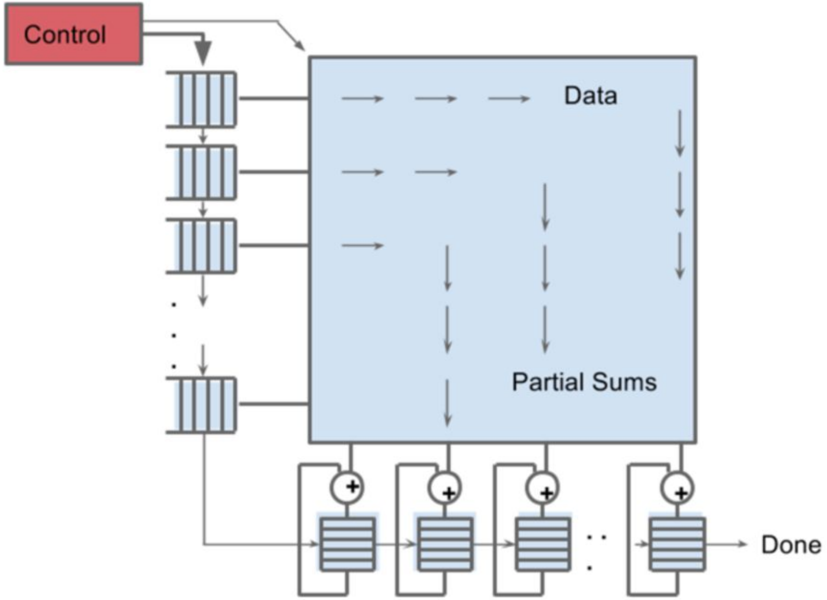


Nvidia Tensor Cores



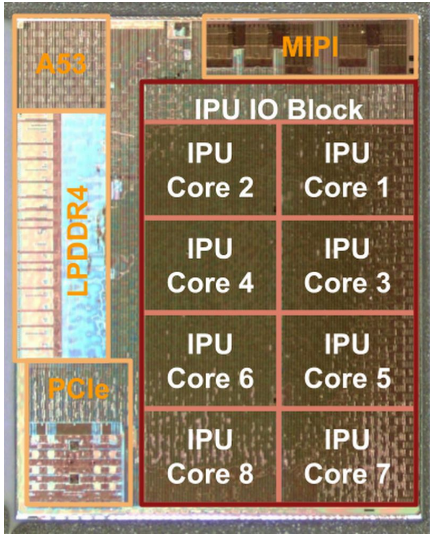
https://www.youtube.com/watch?v=7HUbfJ9ke3A

Google TPU



Pixel Visual Core

Machine learning and HDR on Pixel 2



https://www.youtube.com/watch?v=Gk7FWH12WLI

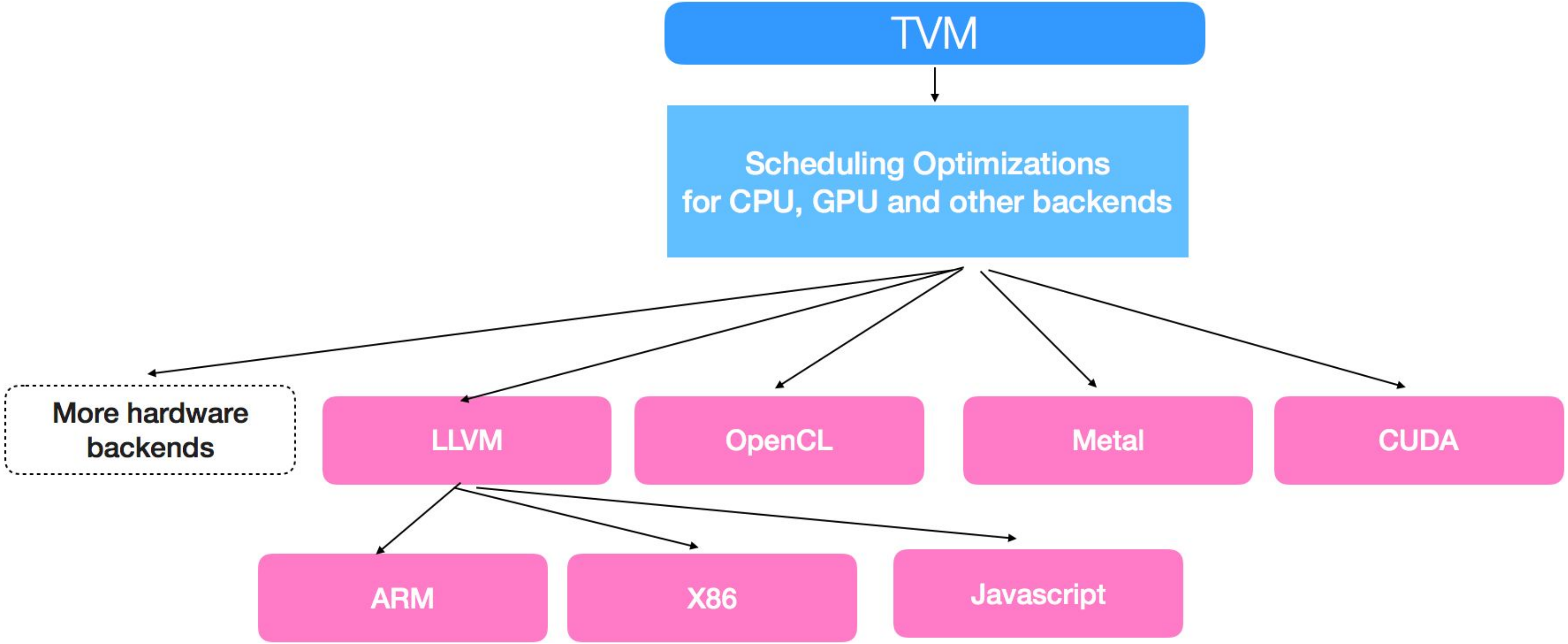
Android Neural Network API

Android 8.1 ships with neural network APIs.

Uses GPU in mobiles efficiently.

Developments in Programming Languages

TVMLang: A new high level programming language/compiler for machine



Architecture Search

Neural Network Designing
Neural Networks!

Neural Architecture Search

All previous methods required a neural network to be designed by humans.

In NAS, a machine learning algorithm does a heuristic search over neural network designs to get optimized network.

https://www.youtube.com/watch?v=YNLC0wJSHxI

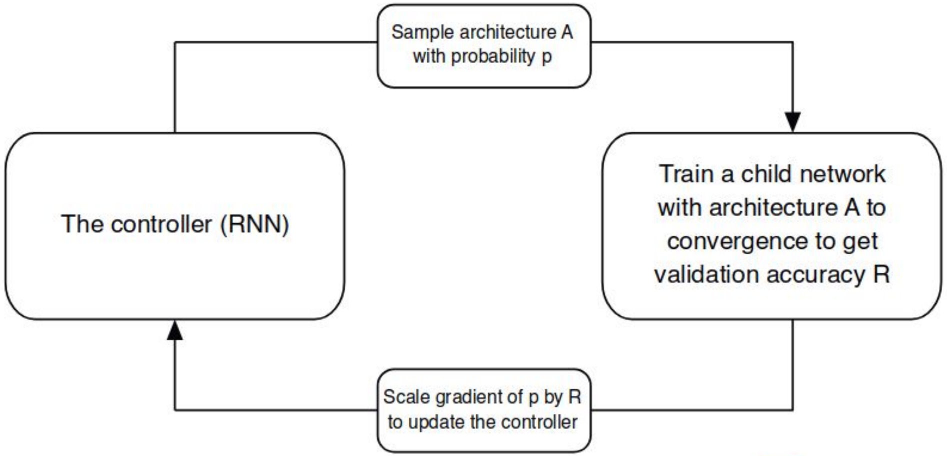
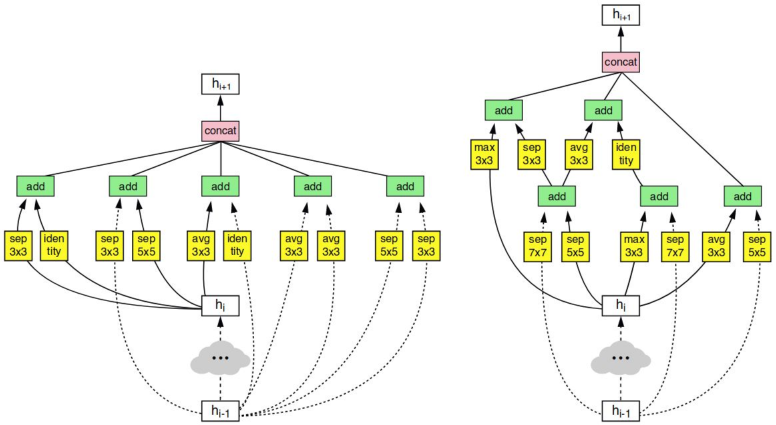
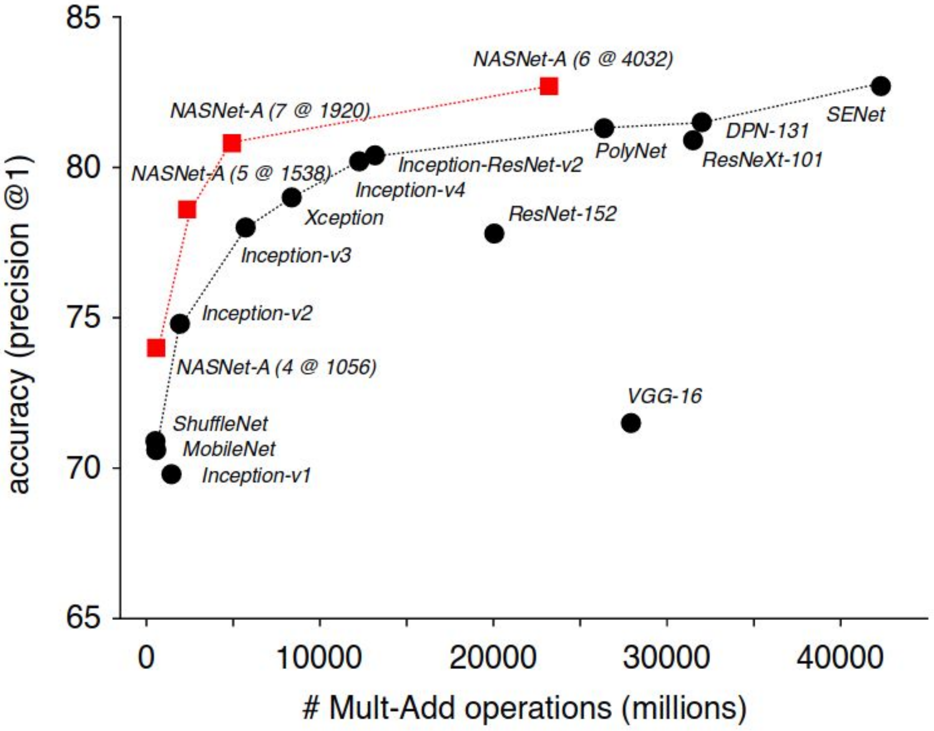


Figure 1. Overview of Neural Architecture Search [70]. A controller RNN predicts architecture A from a search space with probability p. A child network with architecture A is trained to convergence achieving accuracy R. Scale the gradients of p by R to update the RNN controller.

Neural Architecture Search





Product in 2018

Google Cloud Platform

Search CONSOLE

Why Google | Products | Solutions | Launcher | Pricing | Customers | Documentation | Support | Partners | CONTACT SALES

CLOUD AUTOML ALPHA

Train high quality custom machine learning models with minimum effort and machine learning expertise

REQUEST ACCESS

Train Custom Machine Learning Models

Cloud AutoML is a suite of Machine Learning products that enables developers with limited machine learning expertise to train high quality models by leveraging Google's state of the art transfer learning, and Neural Architecture Search technology.

AutoML Vision is the first product to be released. It is a simple, secure and flexible ML service that lets you train custom vision models for your own use cases. Soon, Cloud AutoML will release other services for all other major fields of AI.

Thanks!