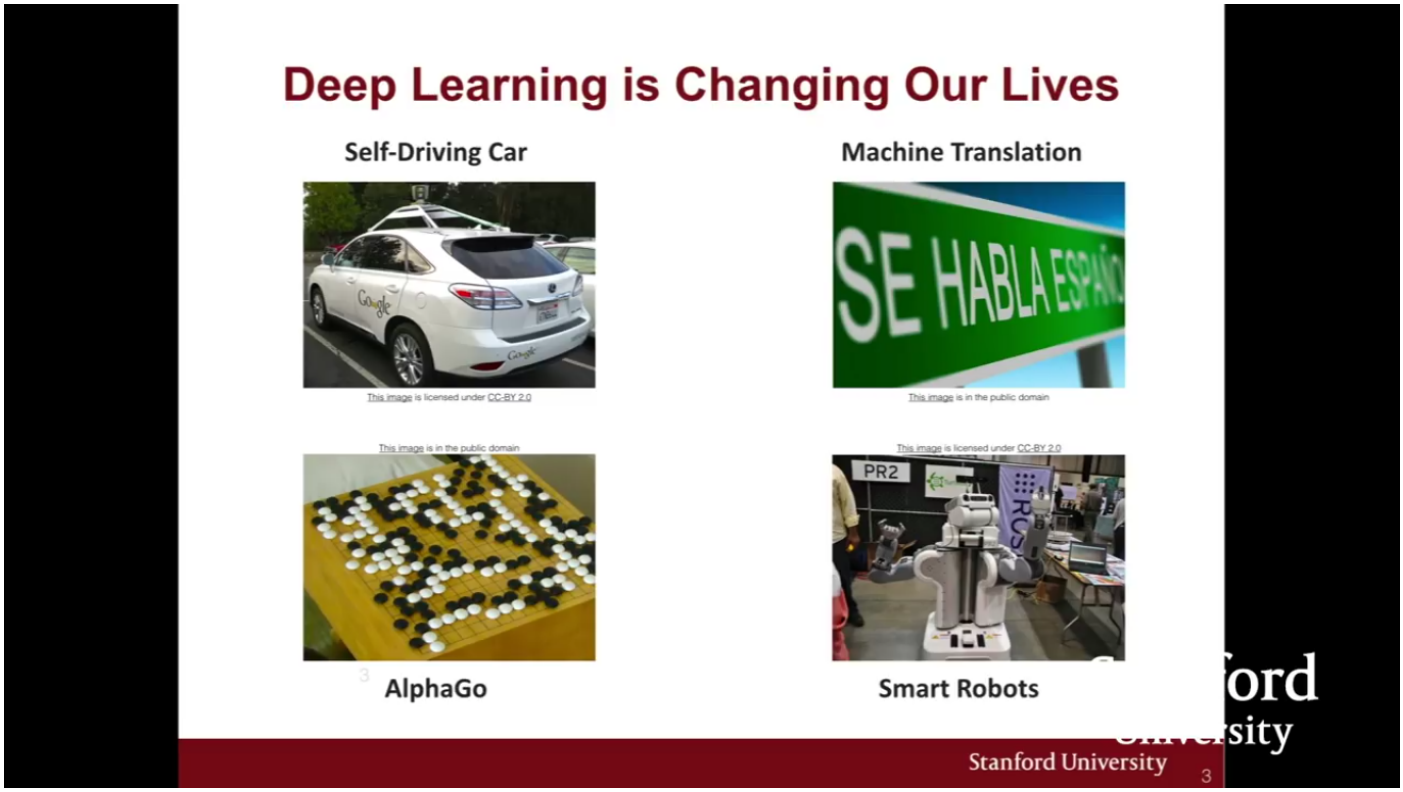
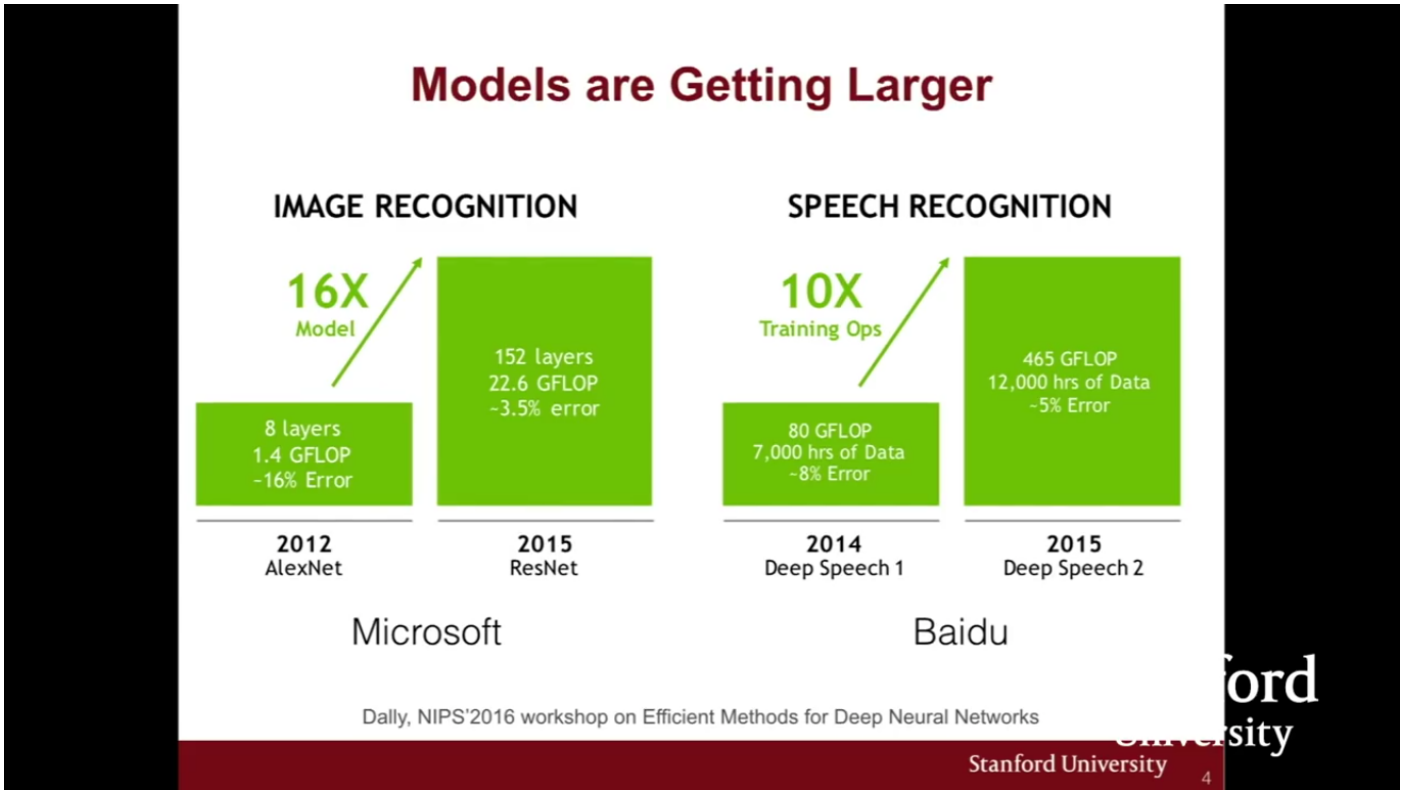
lec15

Creation Date: 17/01/2020 22:00
Last Modified Date: 22/01/2020 12:42

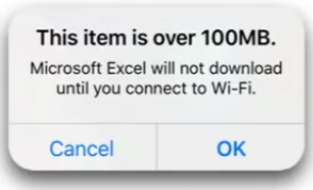
Lec 15: Efficient Methods and Hardware for Deep Learning

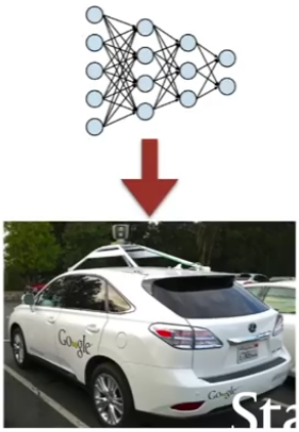




The first Challenge: Model Size

Hard to distribute large models through over-the-air update





The Second Challenge: Speed

	Error rate	Training time
ResNet18:	10.76%	2.5 days
ResNet50:	7.02%	5 days
ResNet101:	6.21%	1 week
ResNet152:	6.16%	1.5 weeks

Such long training time limits ML researcher's productivity

Training time benchmarked with fb.resnet.torch using four M40 GPUs

The Third Challenge: Energy Efficiency

This image is in the public domain

AlphaGo: 1920 CPUs and 280 GPUs,
$3000 electric bill per game

This image is in the public domain

Phone image is licensed under CC-BY 2.0

on mobile: drains battery
on data-center: increases TCO

This image is licensed under

Stanford University 7

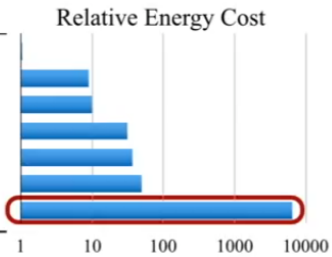
Challenges are:

- Model Size
- Speed
- Energy Efficiency

Where is the Energy Consumed?

larger model => more memory reference => more energy

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	**640**

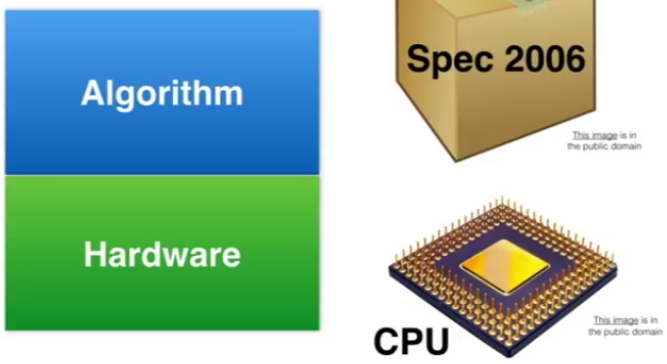


Battery images are in the public domain
Image 1, image 2, image 3, image 4

Stanford University 10

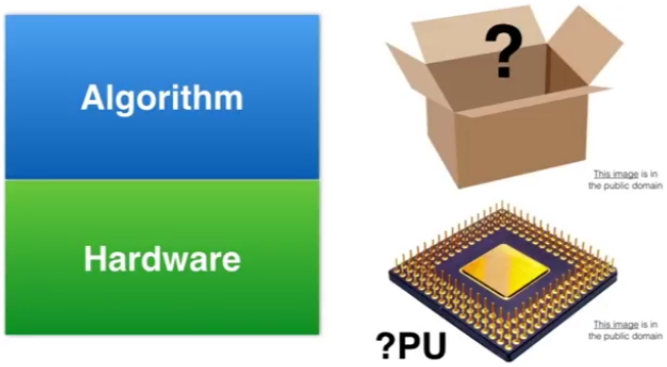
Before:

Application as a Black Box

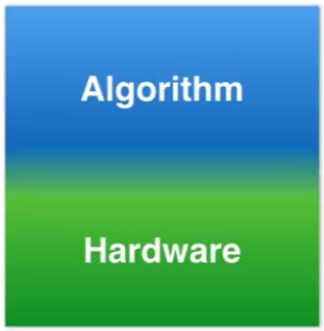


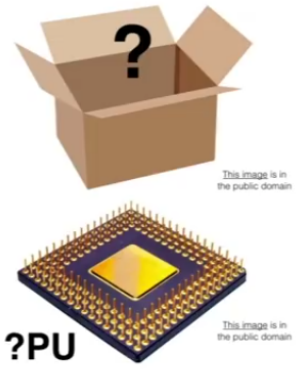
What we should do now:

Open the Box before Hardware Design



Open the Box before Hardware Design

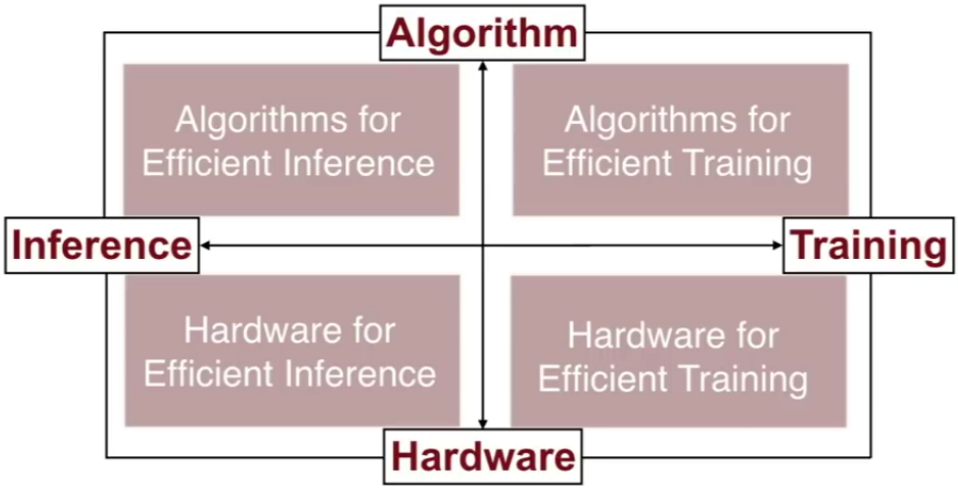


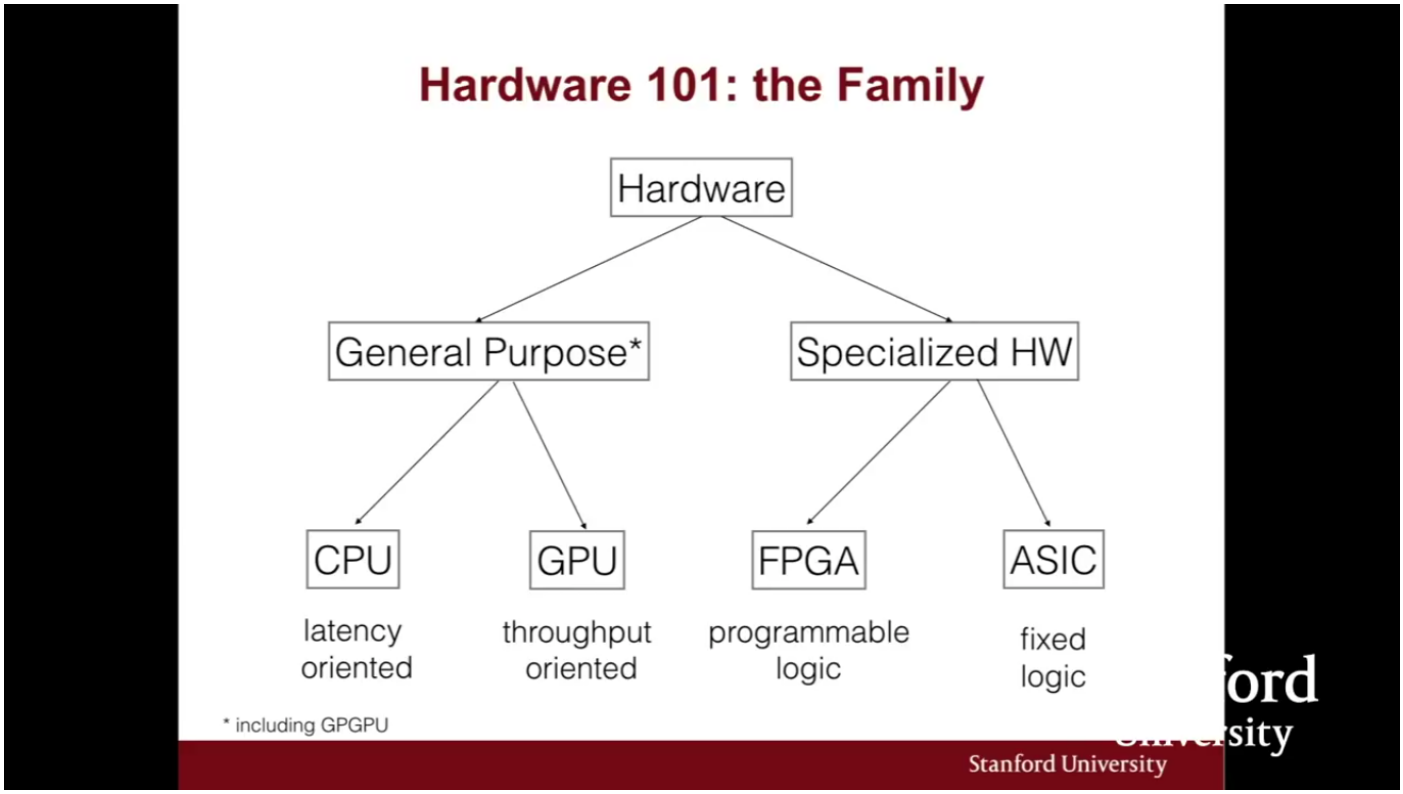


Breaks the boundary between algorithm and hardware

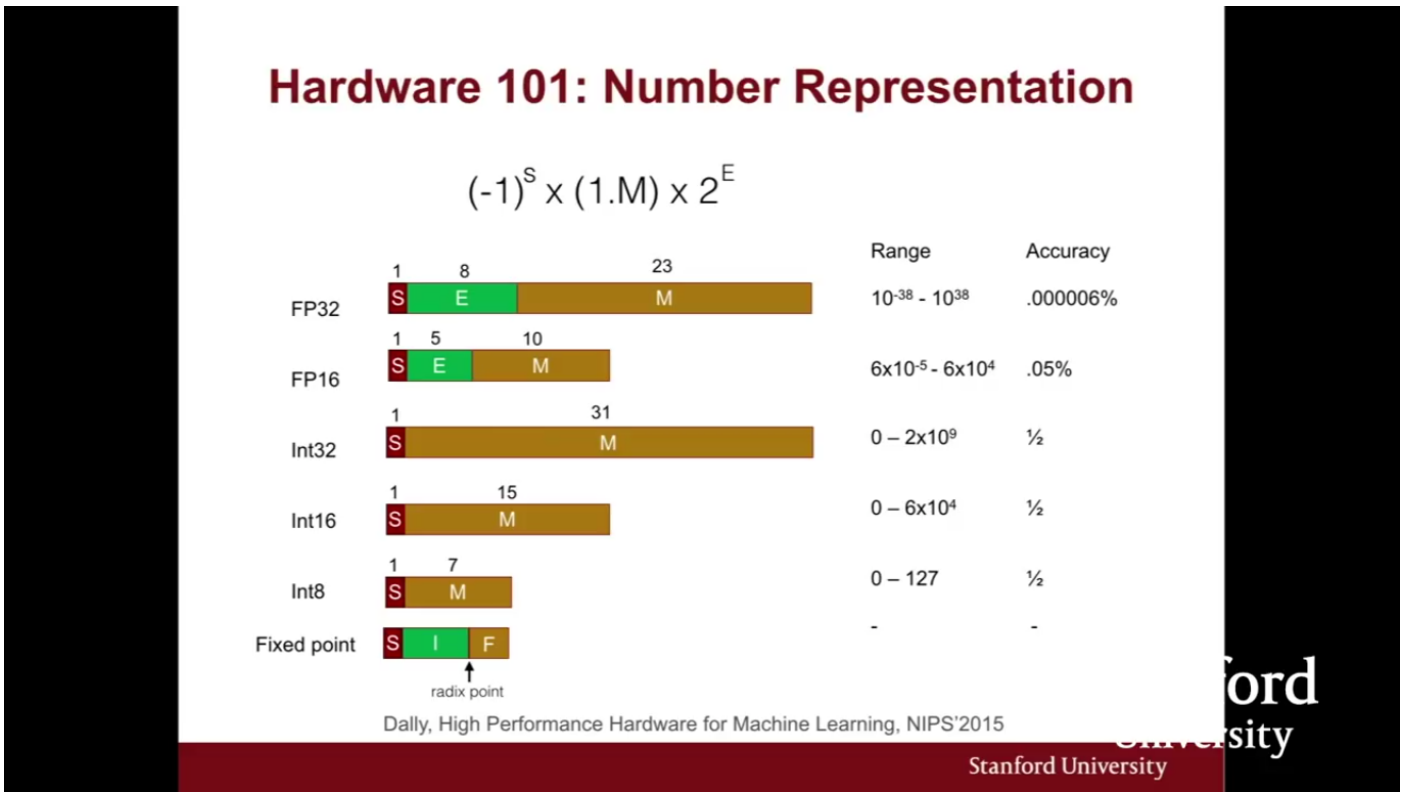
- Breaks the boundary between algorithm and hardware.

Agenda

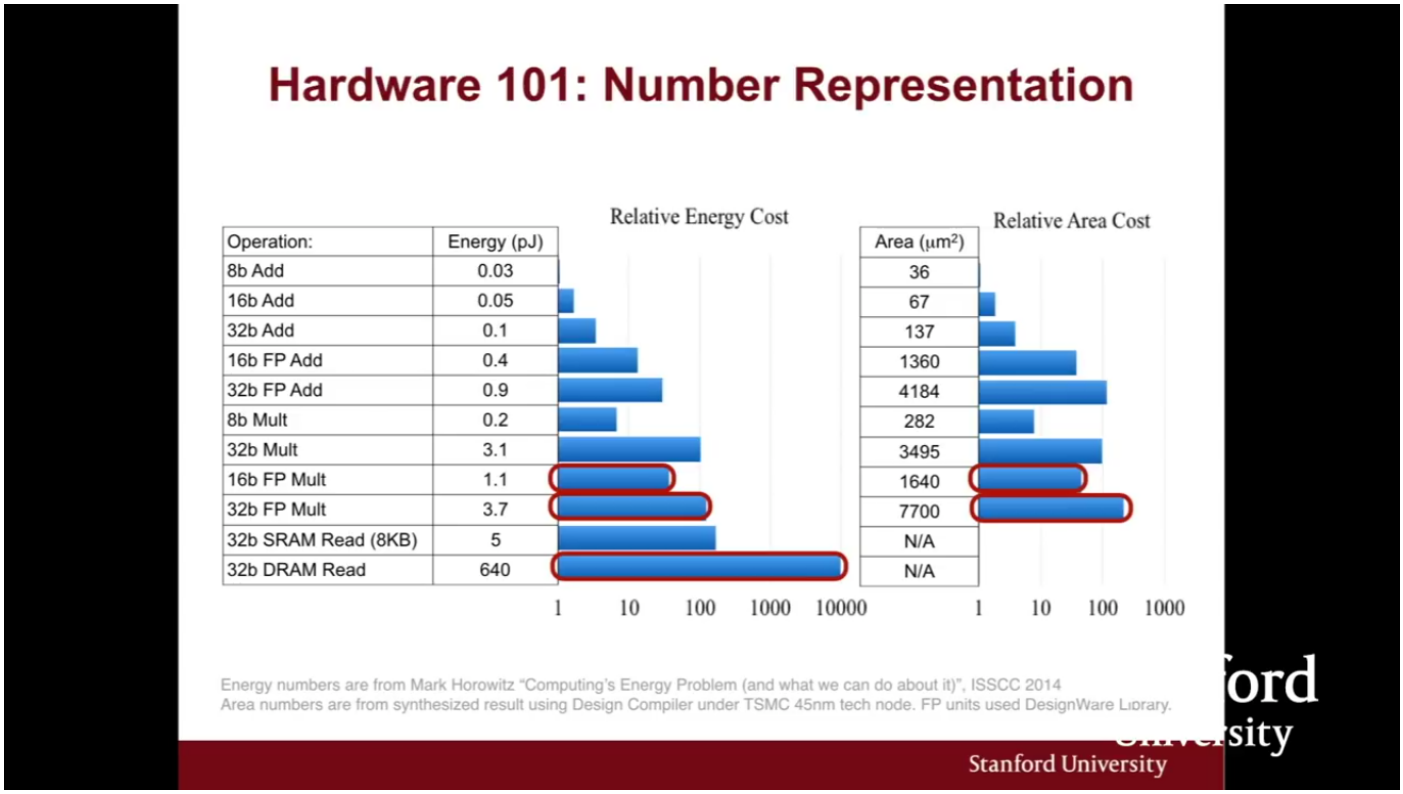


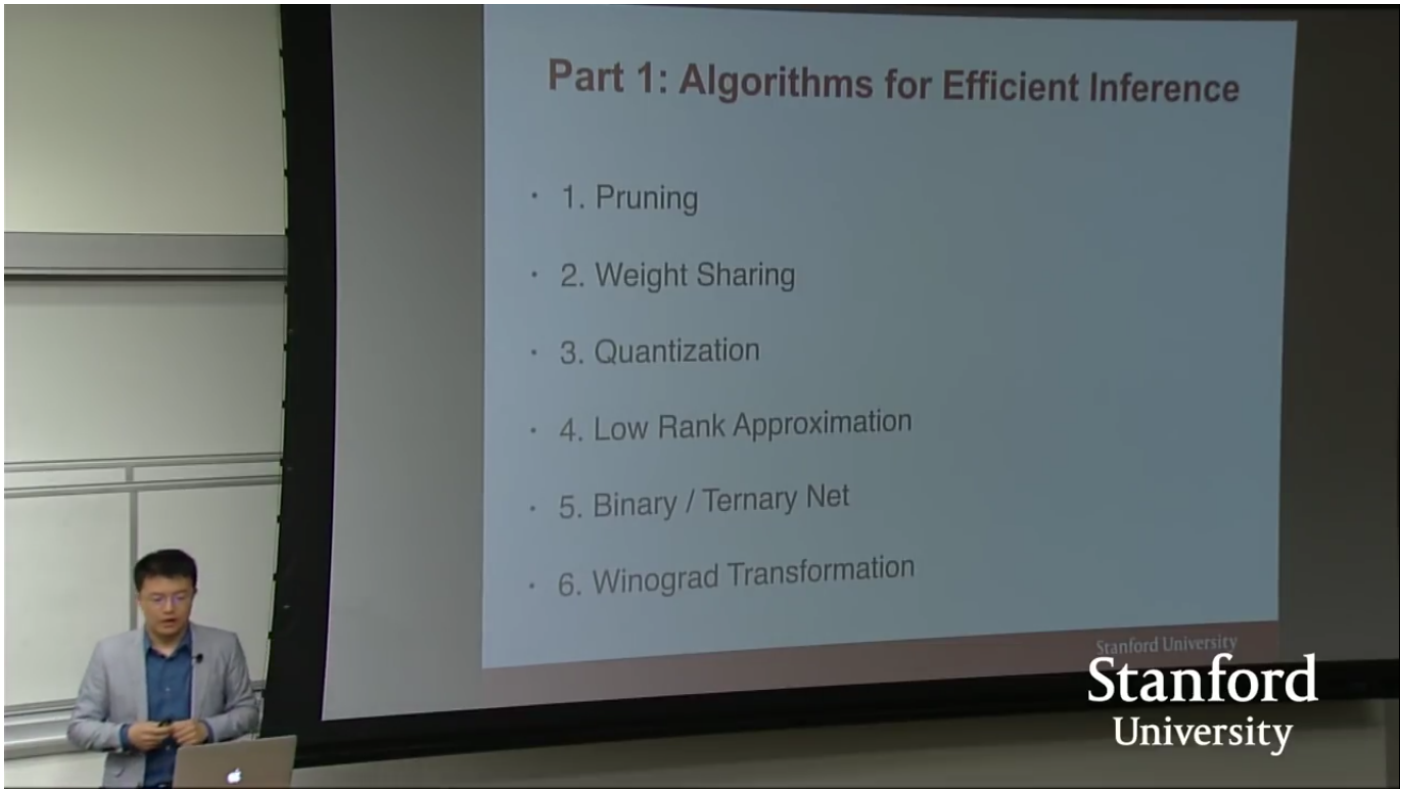


- Specialized HW is tuned for a specific kind of application or domain of applications.
- CPU is latency oriented because it is single threaded(like a big elephant)
- GPU has many small weak cores (like an army of ants)
- FPGA(Field Programmable Gate Array) is programmable and the logic can be changed. Less efficient but can we used for prototyping.
- ASIC(Application Specific Integrated Circuit) has a fixed logic for a certain application.
- **Google's TPU is a kind of ASICs**

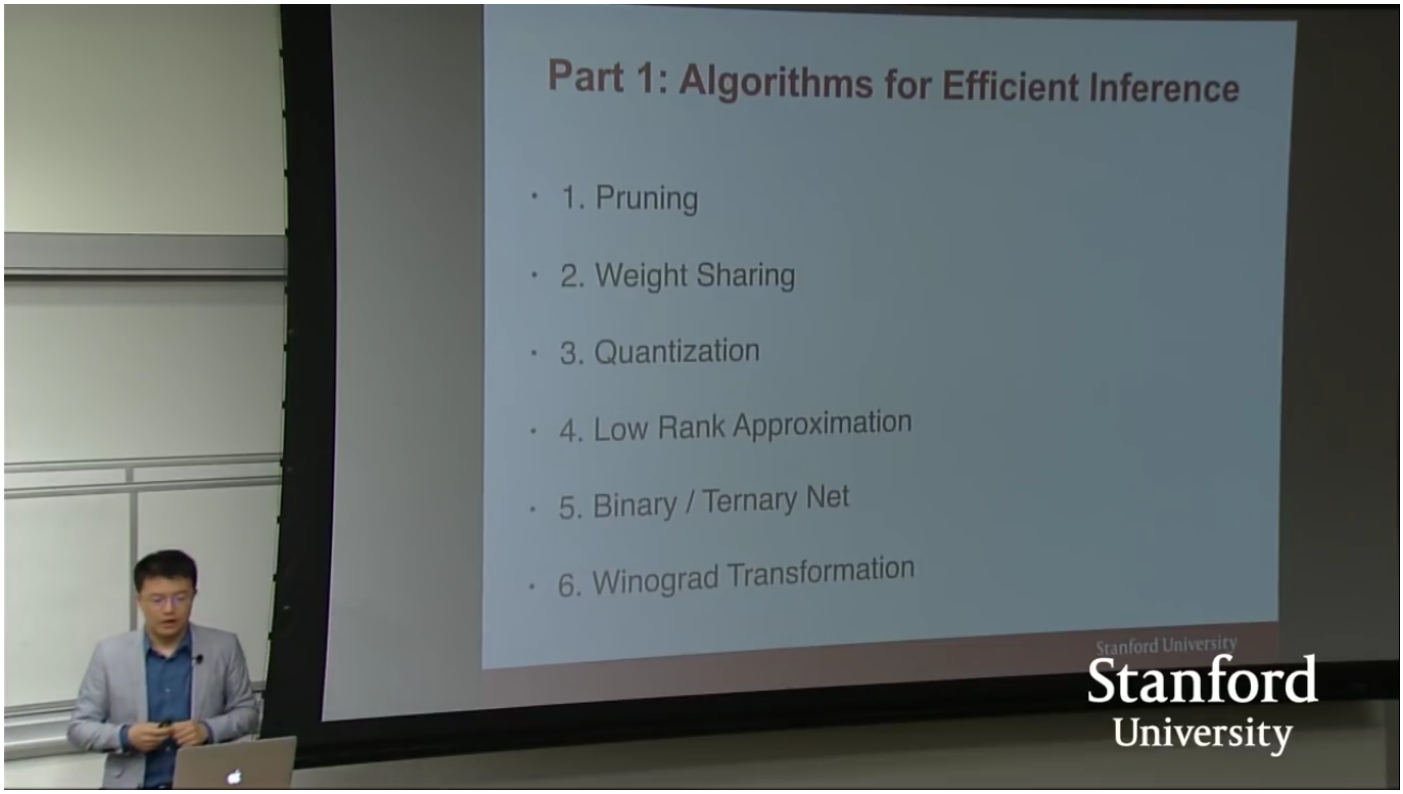


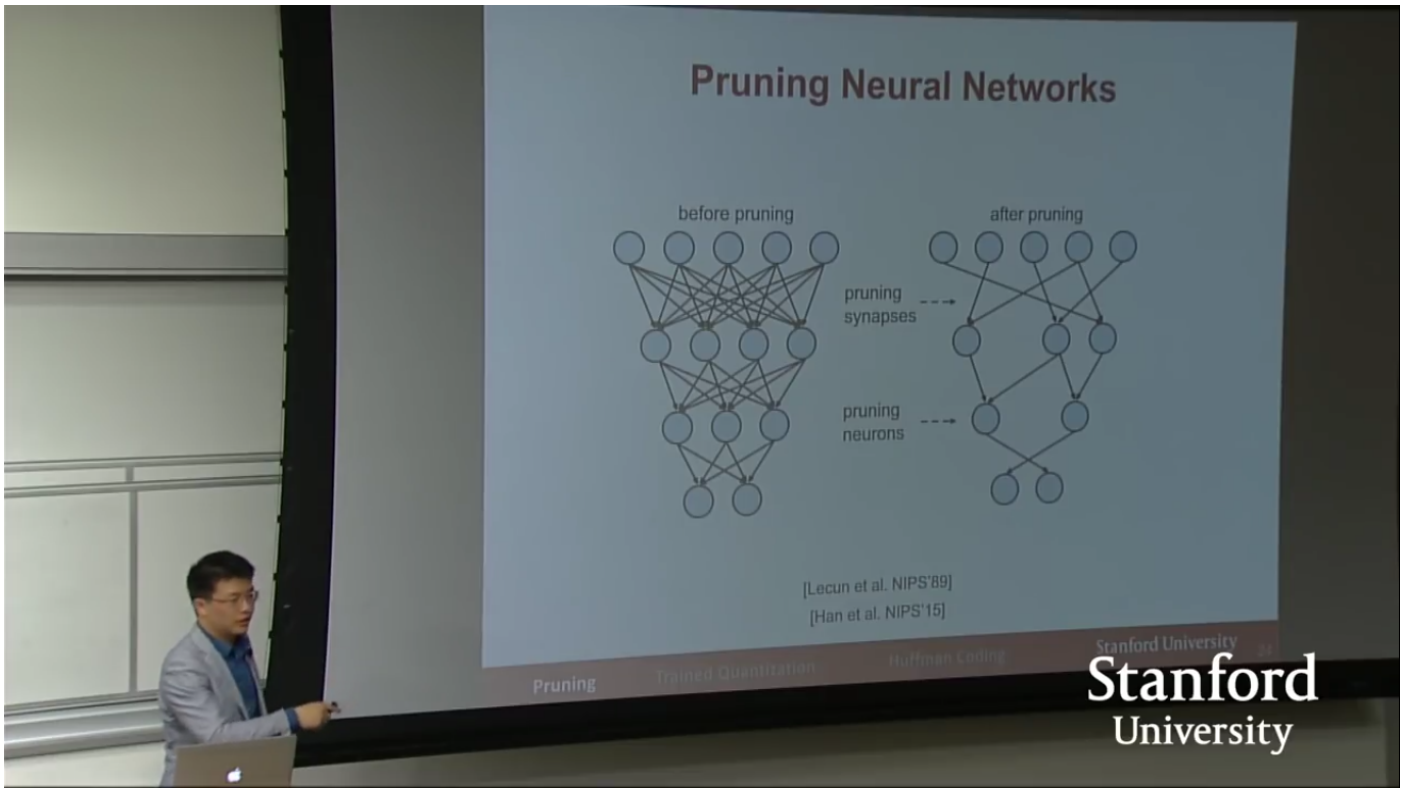
- S-Sign Bit, M-Mantissa Bits, E-Exponent Bits

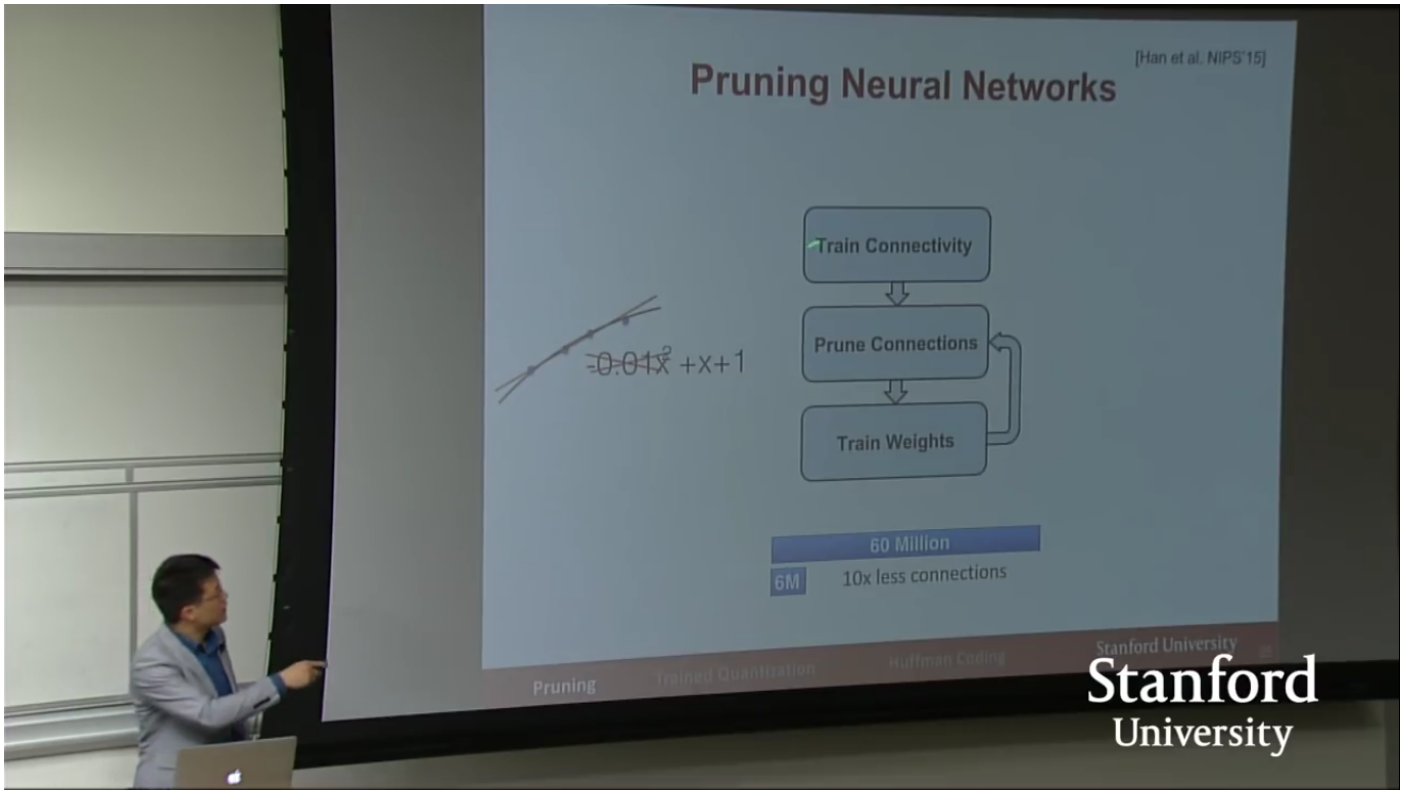


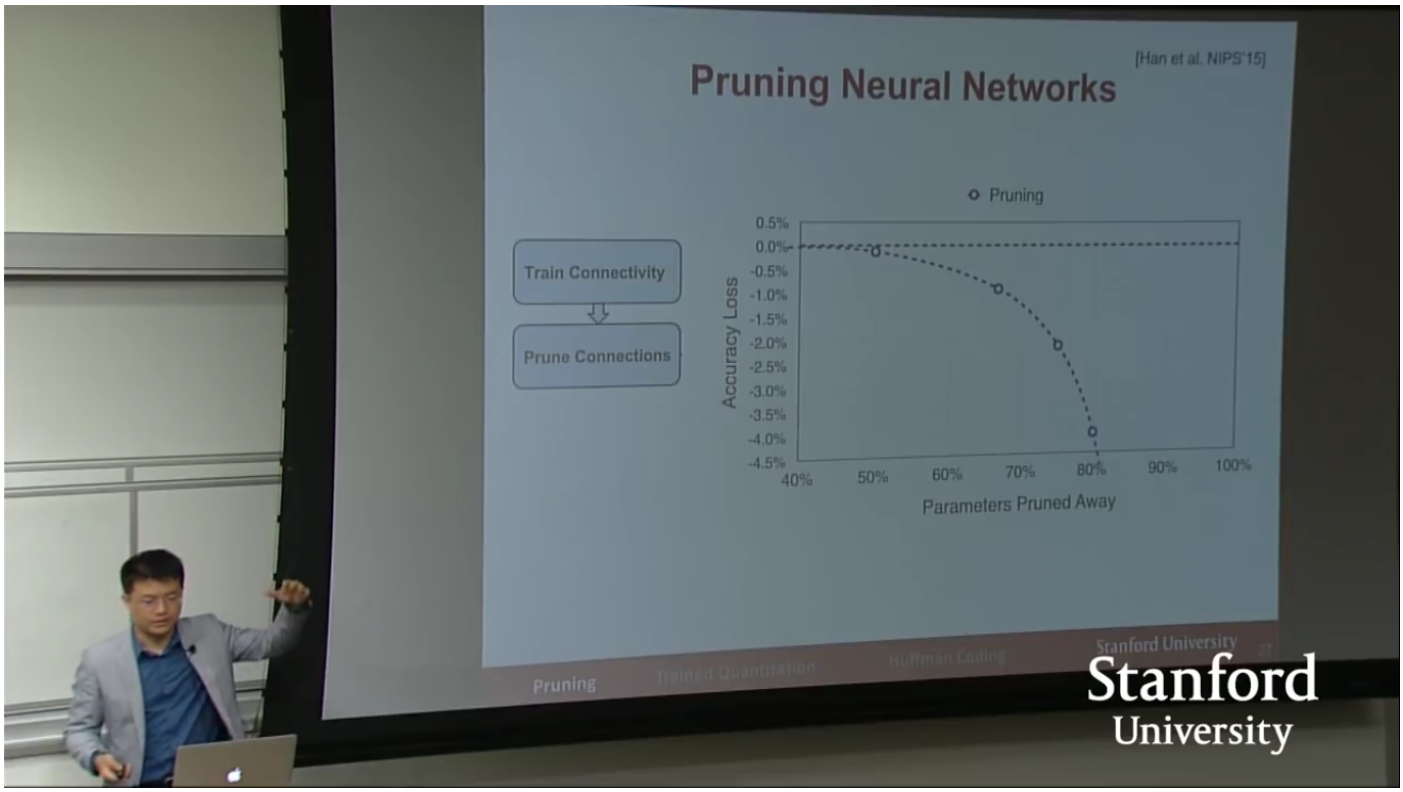


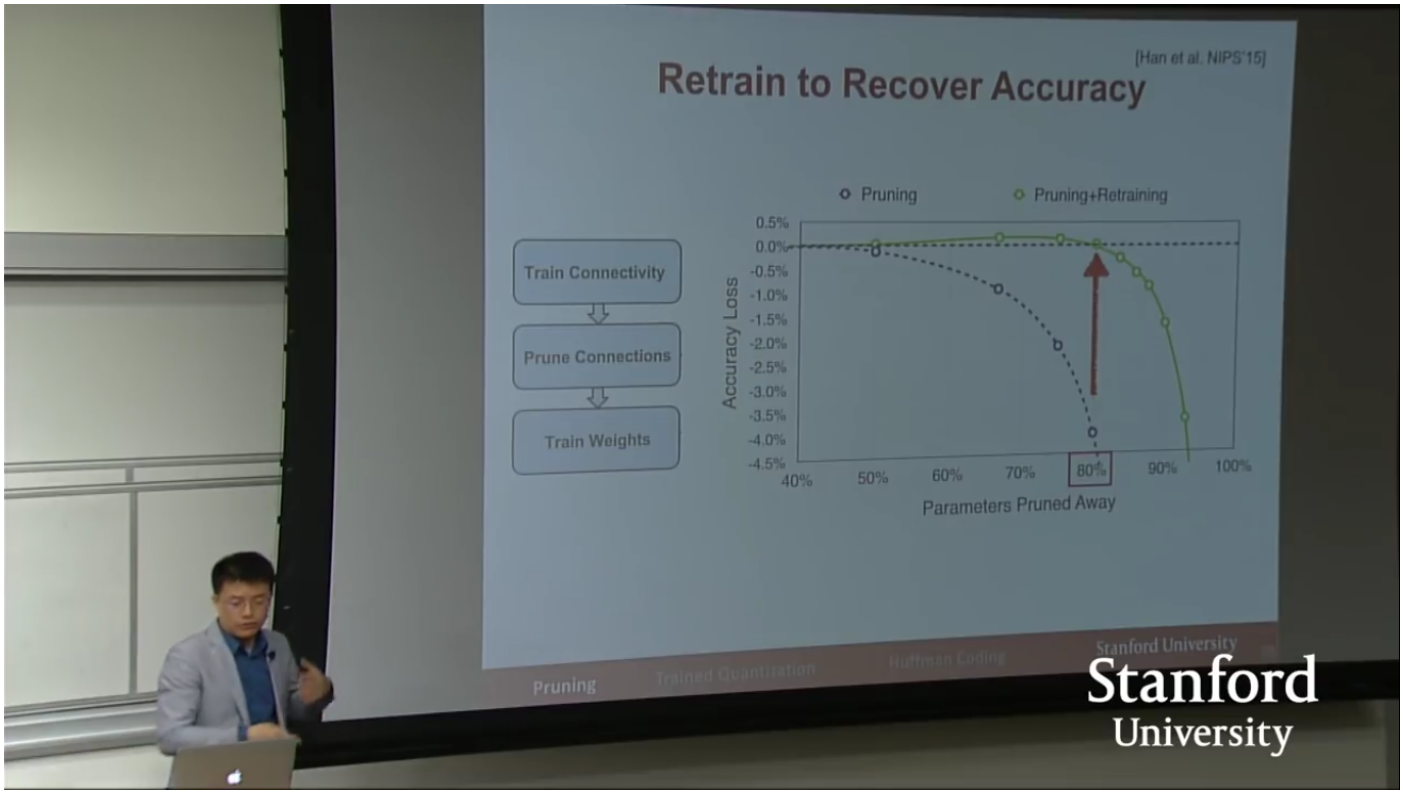
Pruning:


Part 1: Algorithms for Efficient Inference
1. Pruning
2. Weight Sharing
3. Quantization
4. Low Rank Approximation
5. Binary / Ternary Net
6. Winograd Transformation
Stanford University
Stanford
University

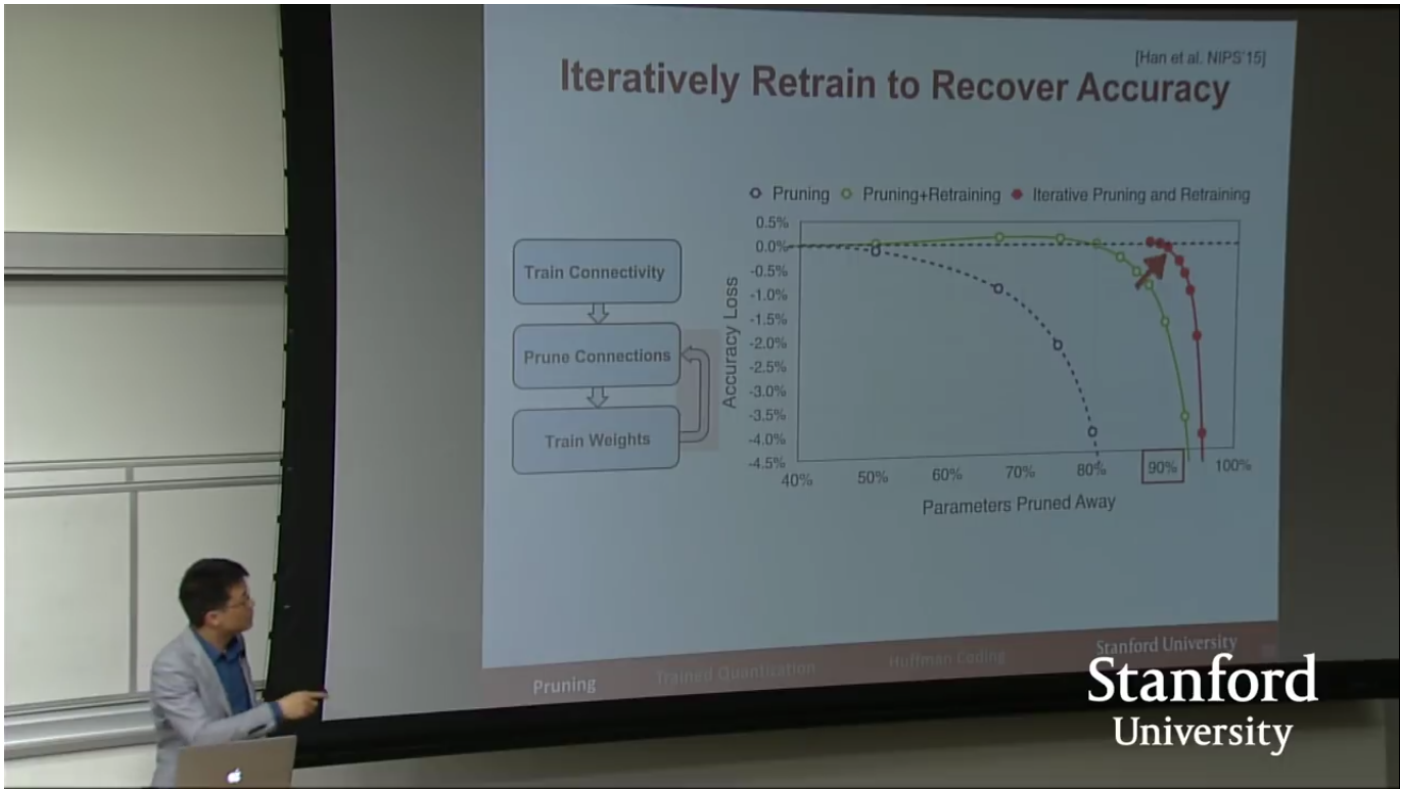

Pruning Neural Networks
before pruning
after pruning
pruning synapses
pruning neurons
[Lecun et al. NIPS'89]
[Han et al. NIPS'15]
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University


Pruning Neural Networks
[Han et al. NIPS'15]
Train Connectivity
Prune Connections
Train Weights
-0.01x² +x+1
60 Million
6M
10x less connections
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University

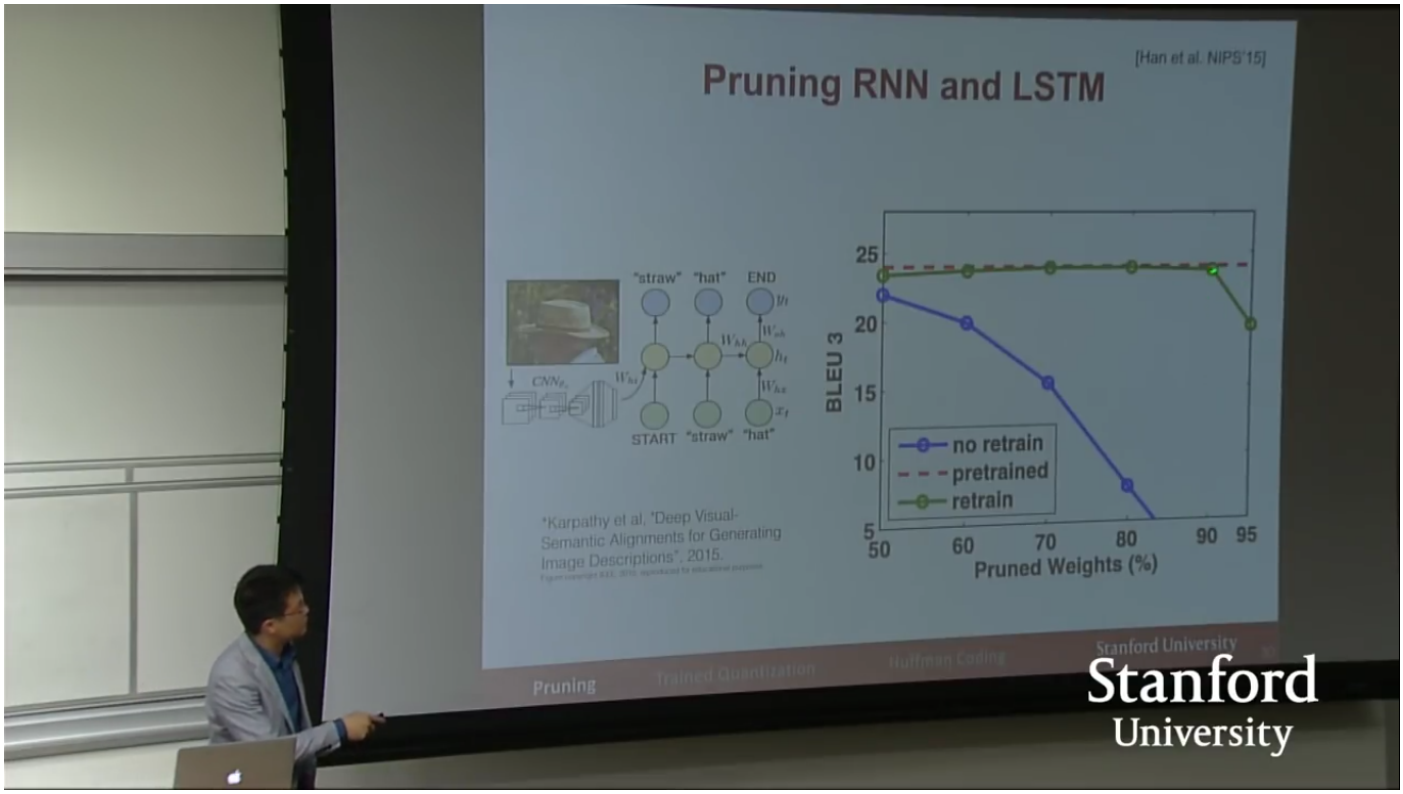

Pruning Neural Networks
[Han et al. NIPS'15]
Train Connectivity
Prune Connections
Pruning
Accuracy Loss
0.5%
0.0%
-0.5%
-1.0%
-1.5%
-2.0%
-2.5%
-3.0%
-3.5%
-4.0%
-4.5%
40%
50%
60%
70%
80%
90%
100%
Parameters Pruned Away
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University

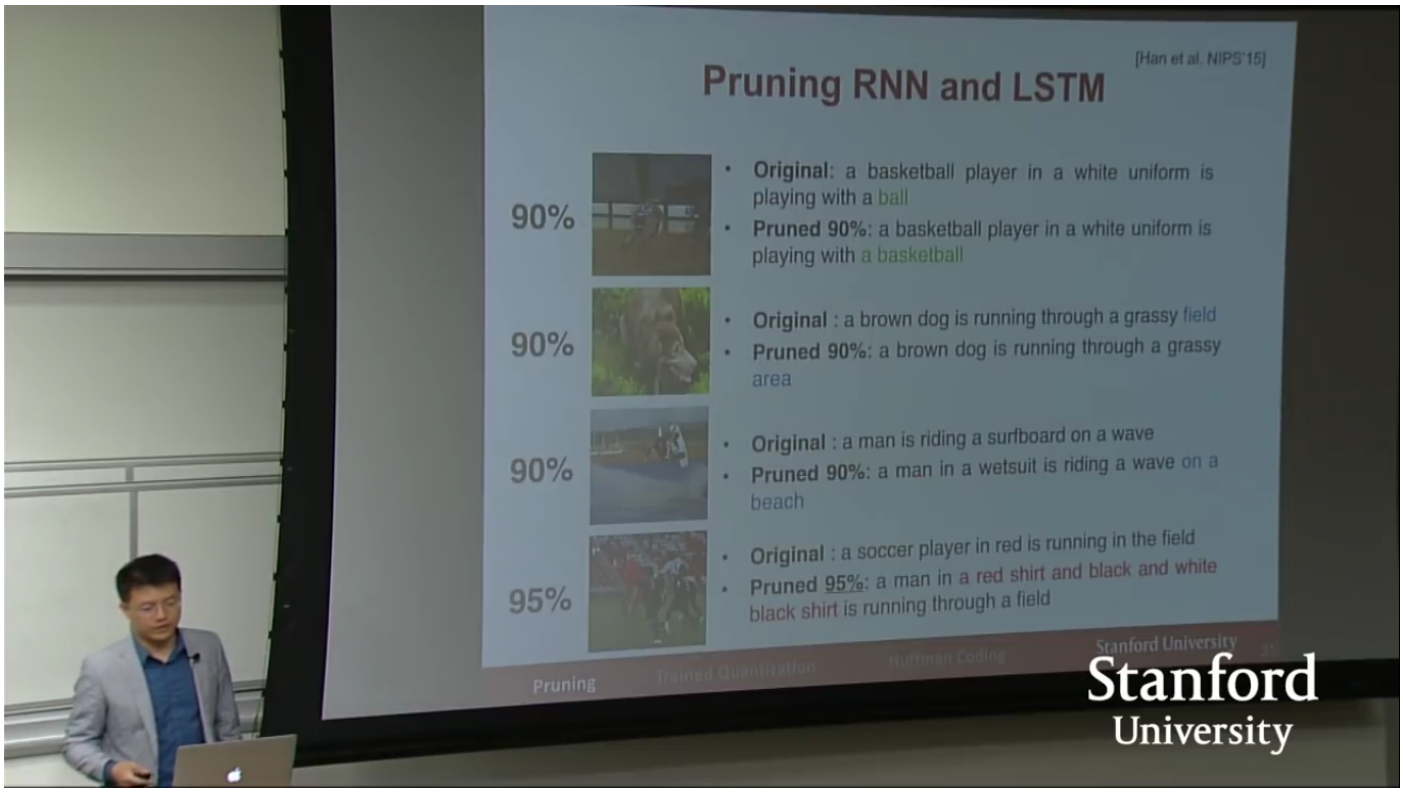


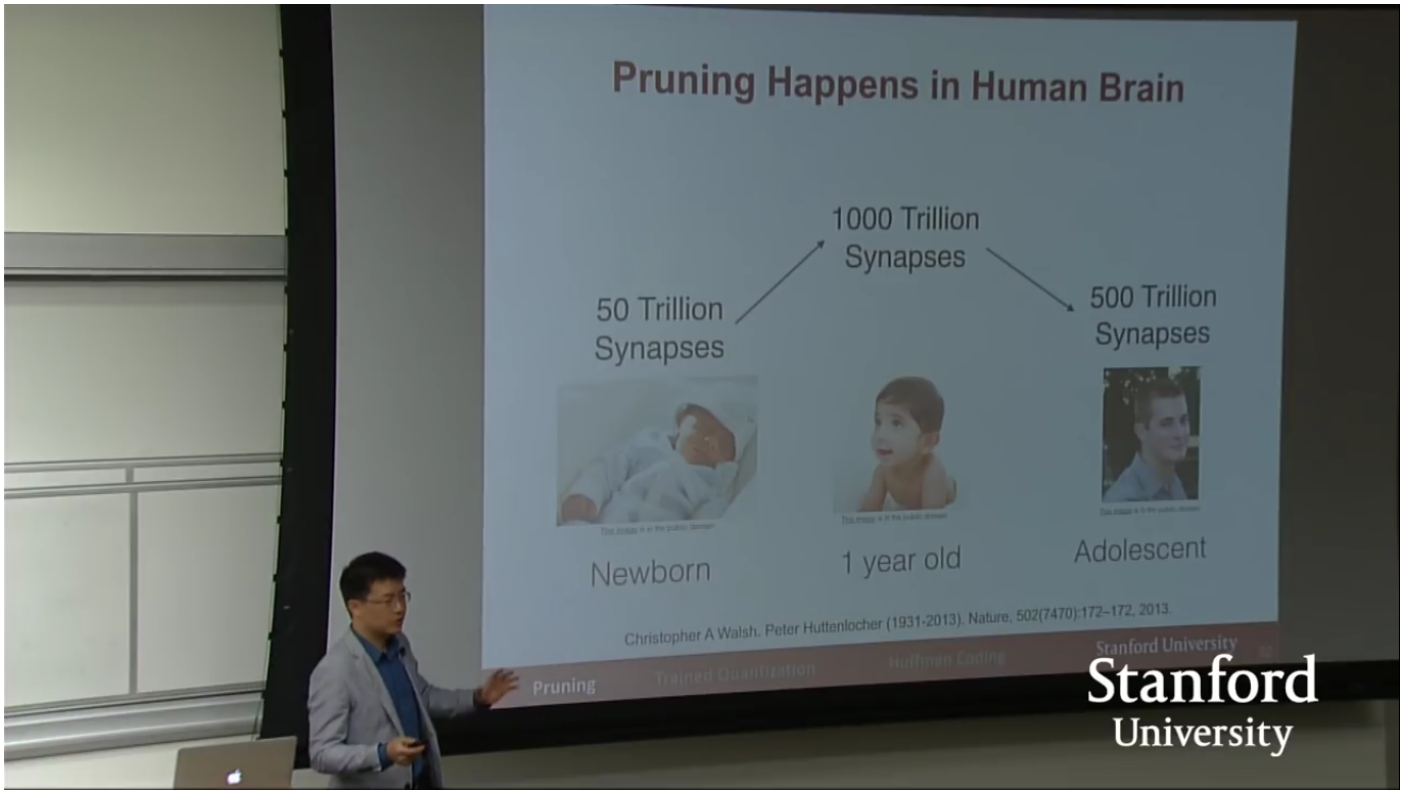
- Pruning even though reduces Accuracy until a certain threshold, **Pruning and Re-Training can help in regaining the Accuracy of the model.**

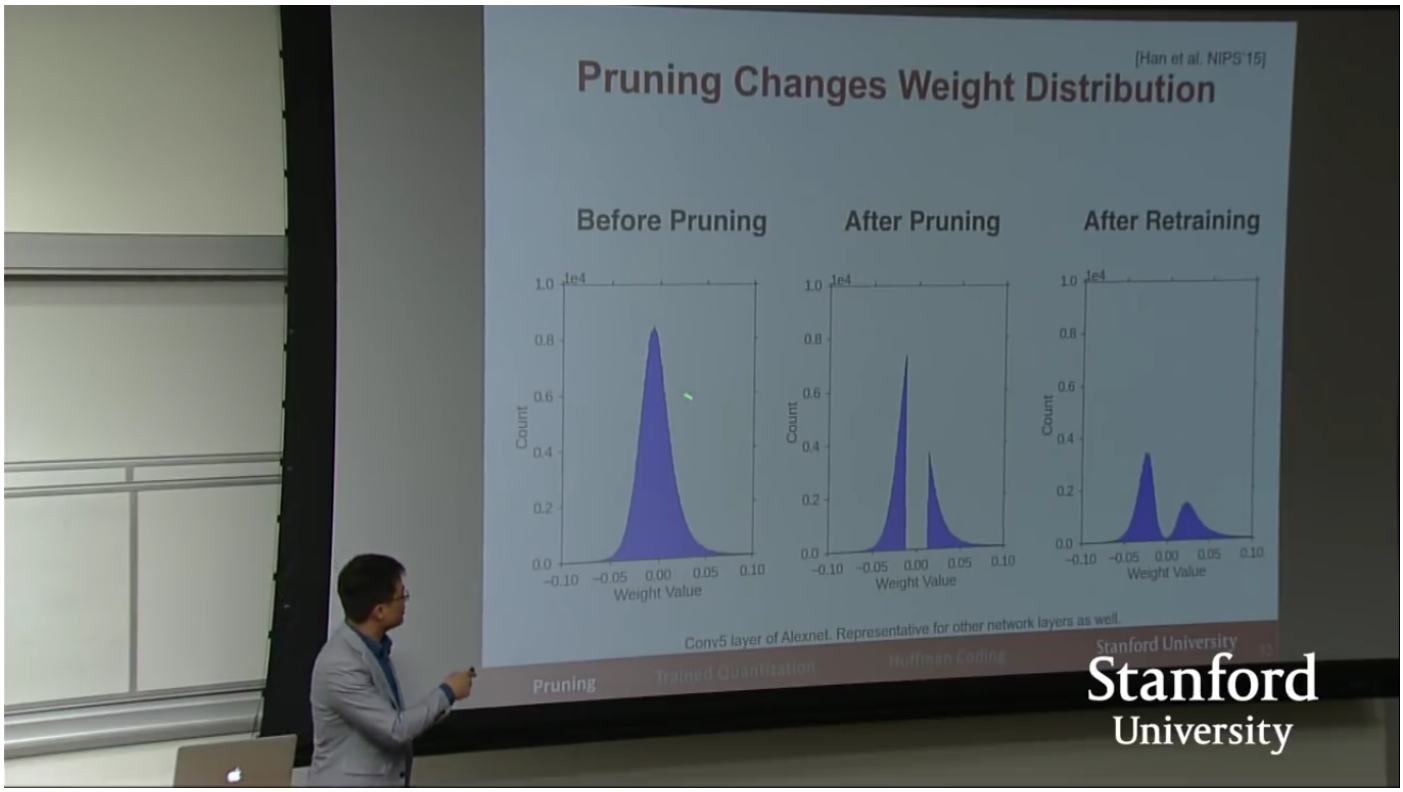


- We can notice that even at a Pruning Rate of 90% the accuracy of the model is retrained due to iterative pruning and retraining.

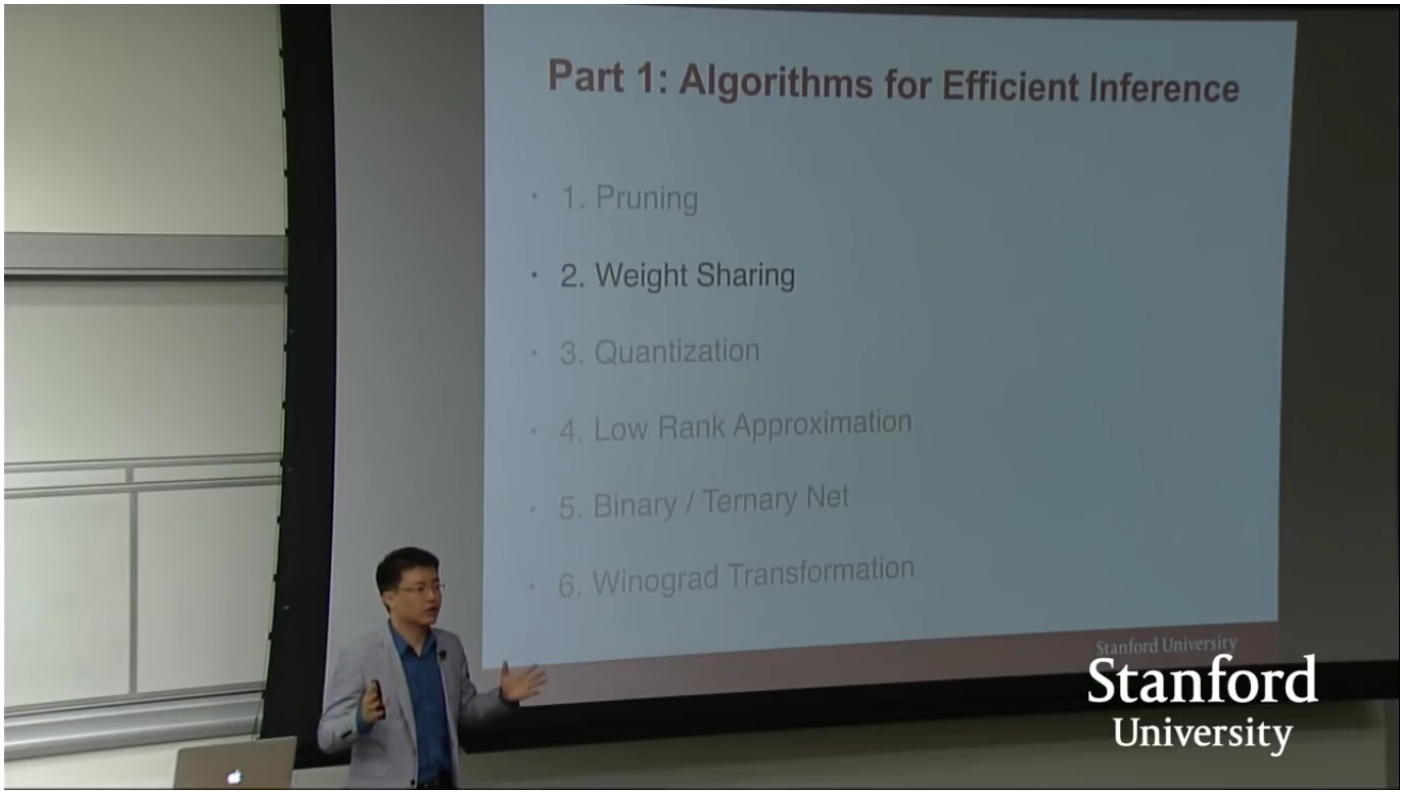

Pruning RNN and LSTM
[Han et al. NIPS'15]
"straw" "hat" END
START "straw" "hat"
*Karpathy et al, "Deep Visual-Semantic Alignments for Generating Image Descriptions", 2015.
BLEU 3
25
20
15
10
5
50 60 70 80 90 95
Pruned Weights (%)
no retrain
pretrained
retrain
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University

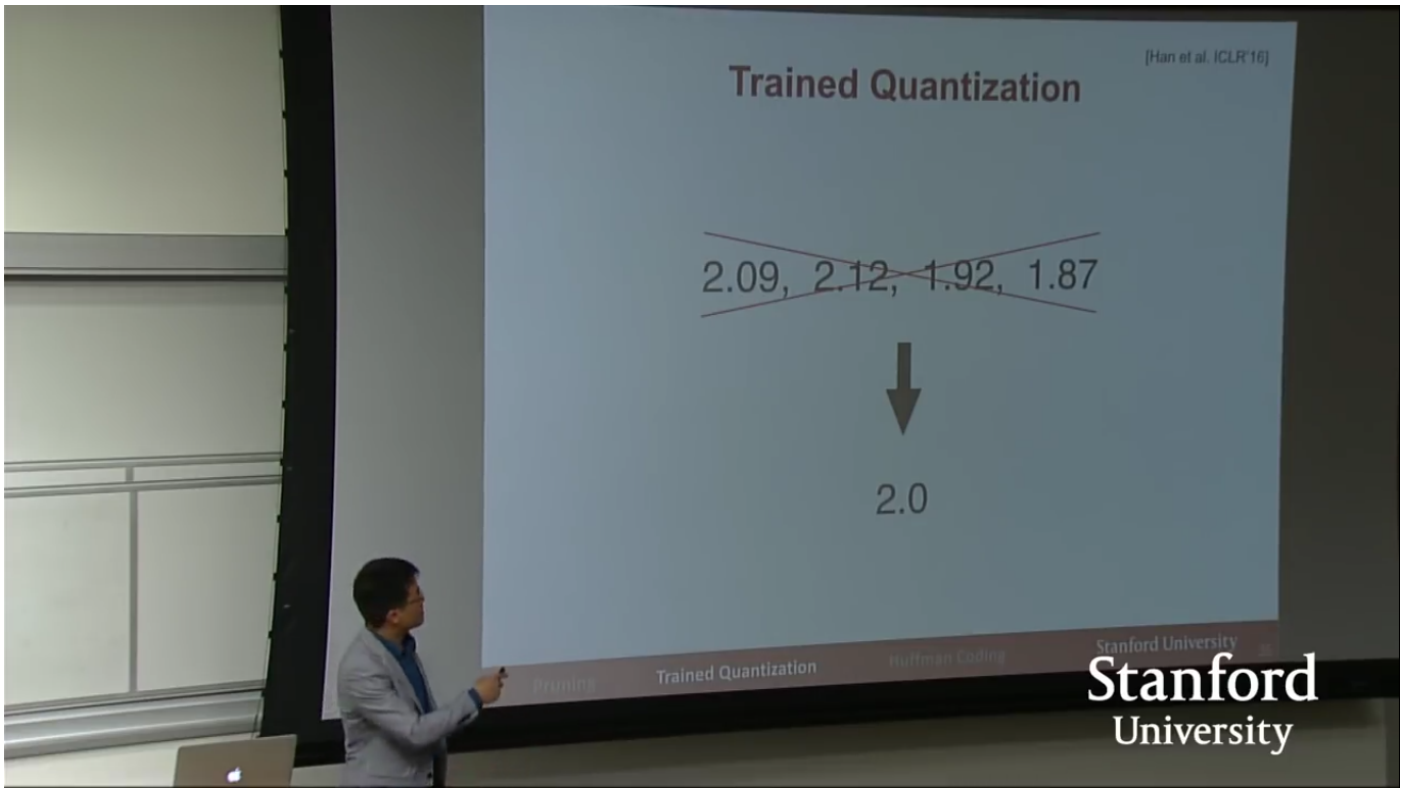

Pruning RNN and LSTM
[Han et al. NIPS'15]
90%
Original: a basketball player in a white uniform is playing with a ball
Pruned 90%: a basketball player in a white uniform is playing with a basketball
90%
Original : a brown dog is running through a grassy field
Pruned 90%: a brown dog is running through a grassy area
90%
Original : a man is riding a surfboard on a wave
Pruned 90%: a man in a wetsuit is riding a wave on a beach
95%
Original : a soccer player in red is running in the field
Pruned 95%: a man in a red shirt and black and white black shirt is running through a field
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University



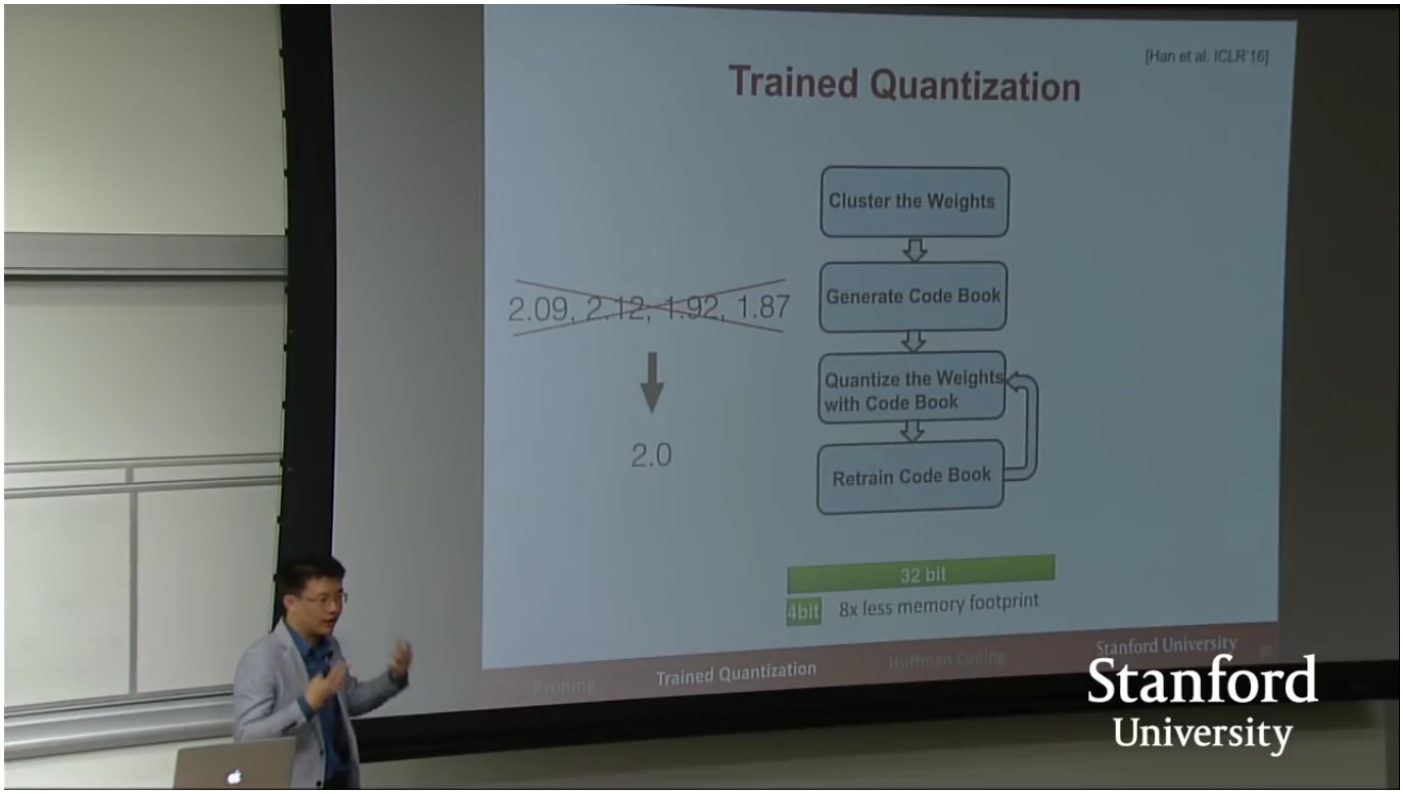


Weight Sharing:

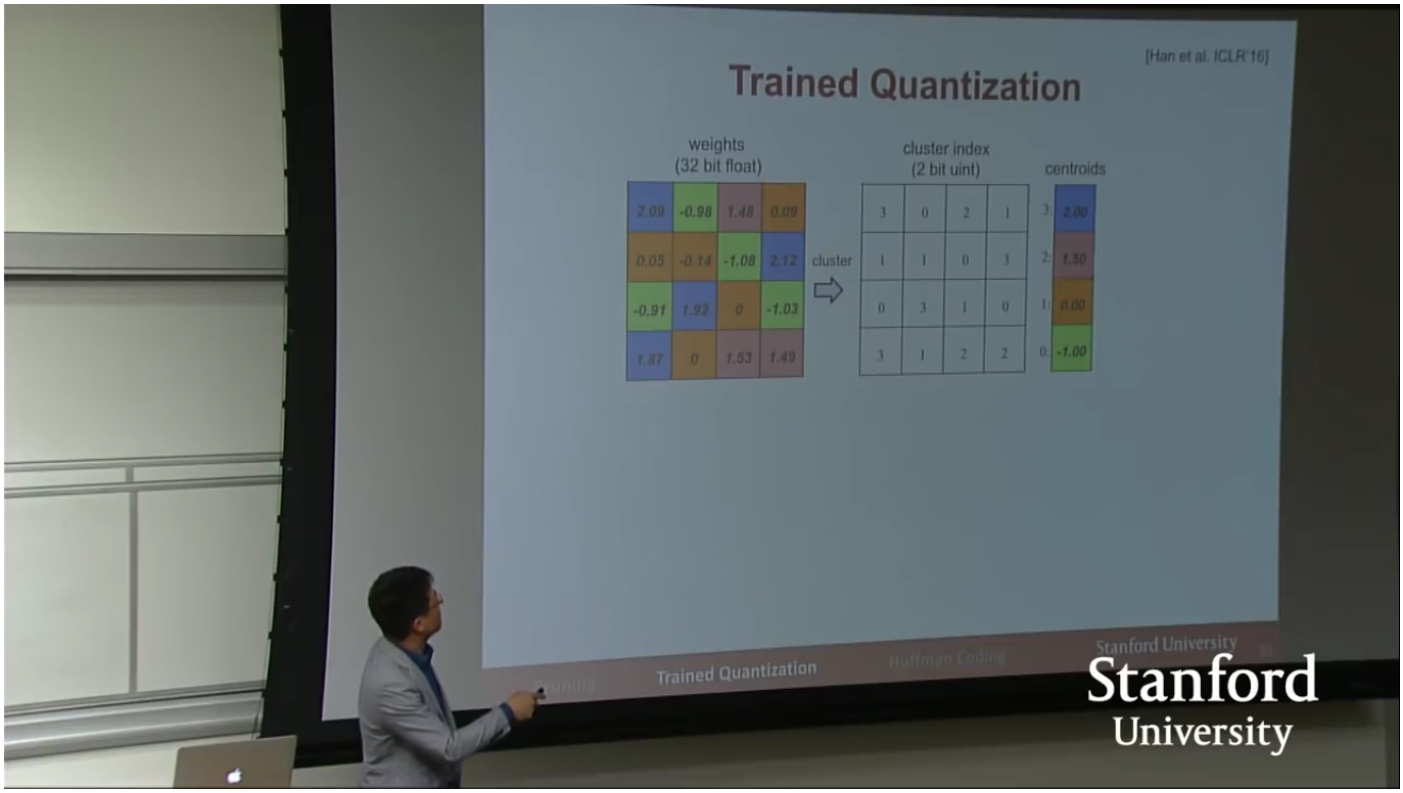




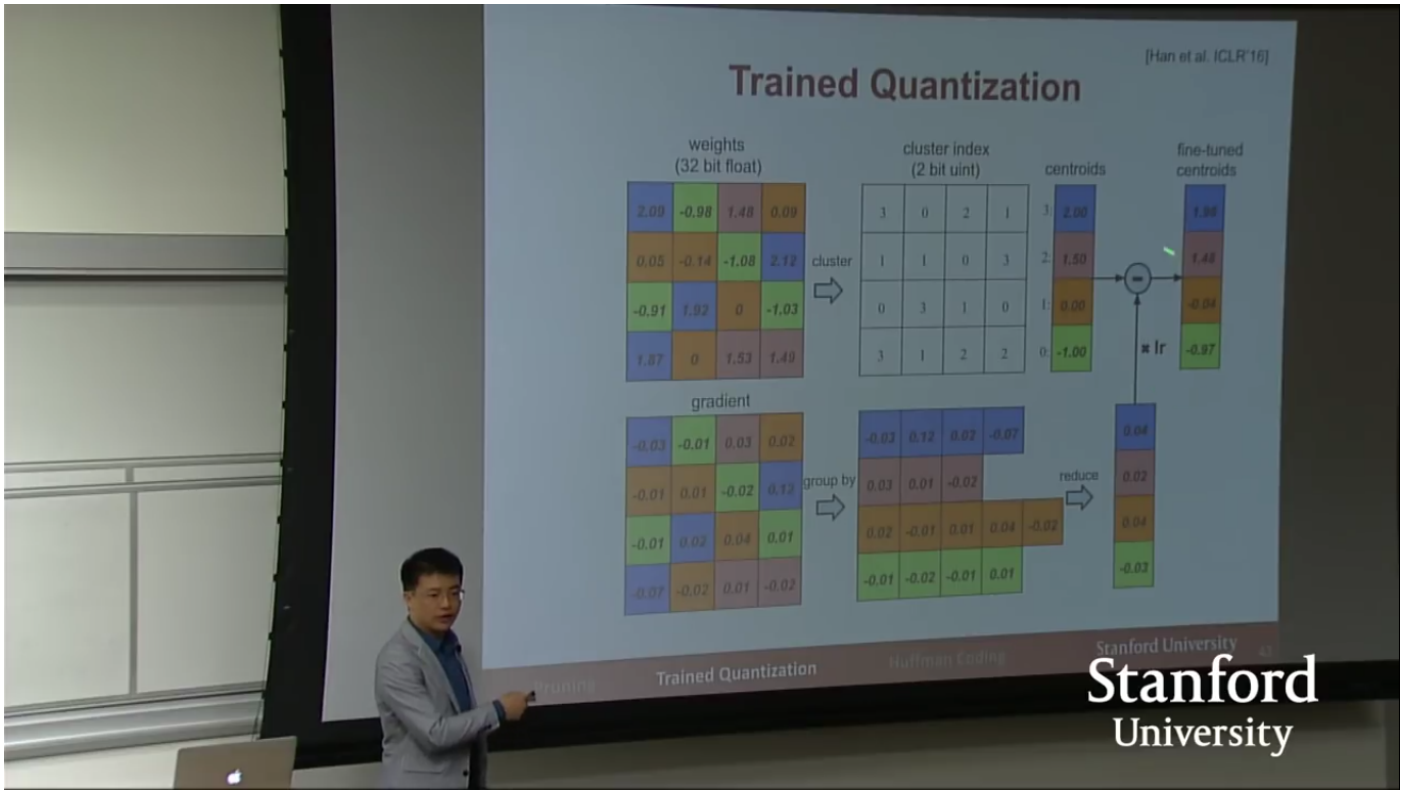
- Not all weights needs to have the exact value. We can use a single number to represent them. Too accurate a number just leads to overfitting.
- So we can use a centroid value to represent them instead of using the full precision weight.
-

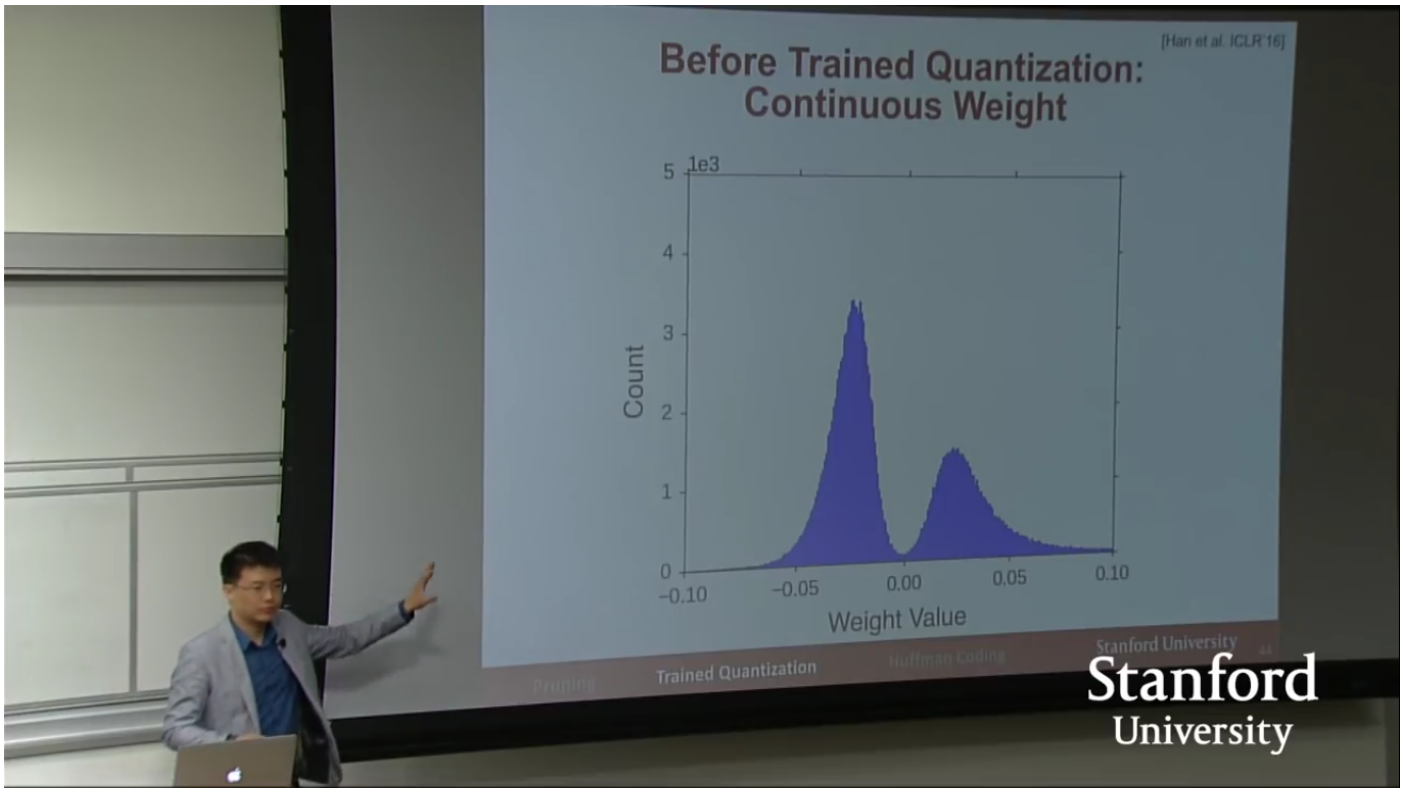


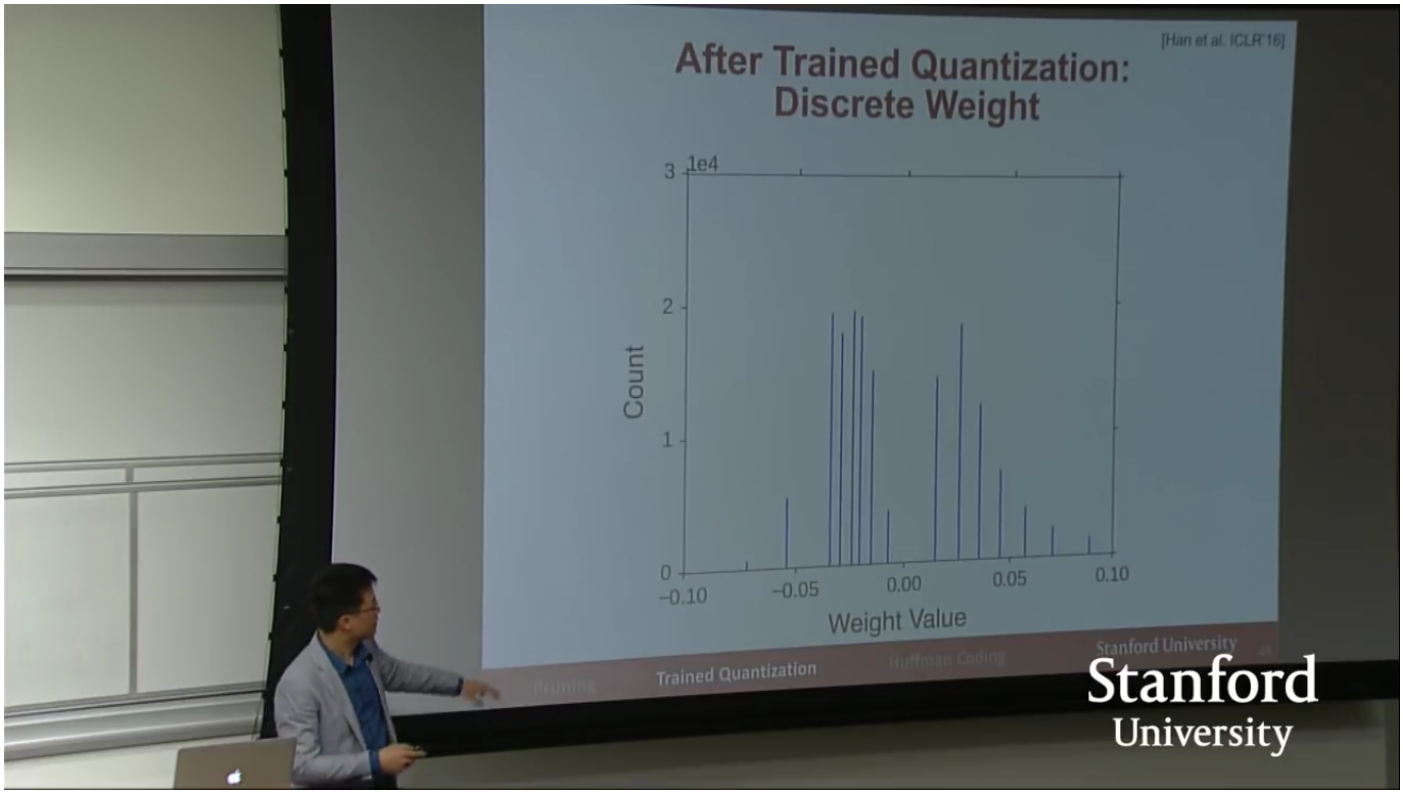
- Due to pruning, we will have less number of parameters, and due to less number of parameters we will need less number of bits to represent those parameters.



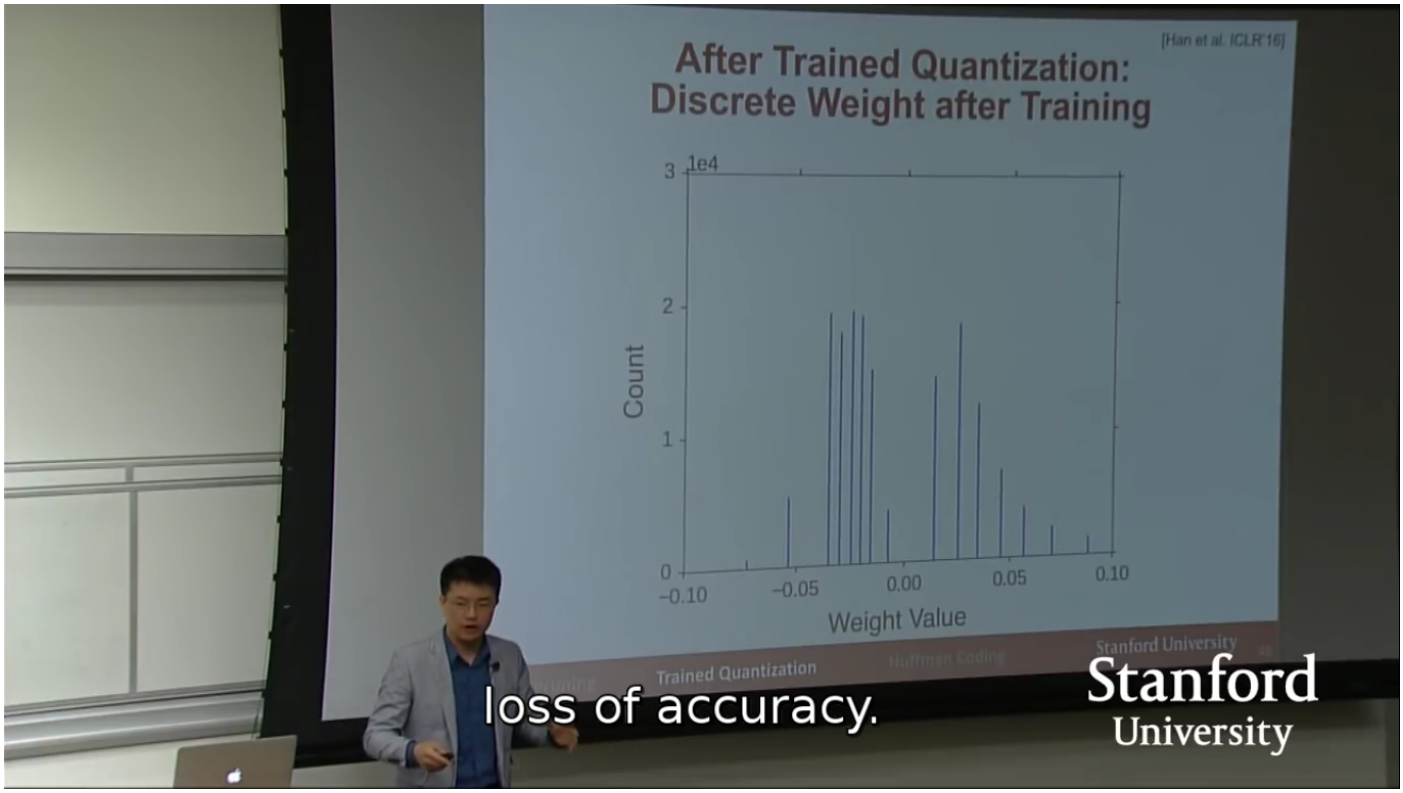
- Instead of using the 32 bit float, we are now using only 2 bit to represent the centroid values.
- But how do we train such shared weighted networks?

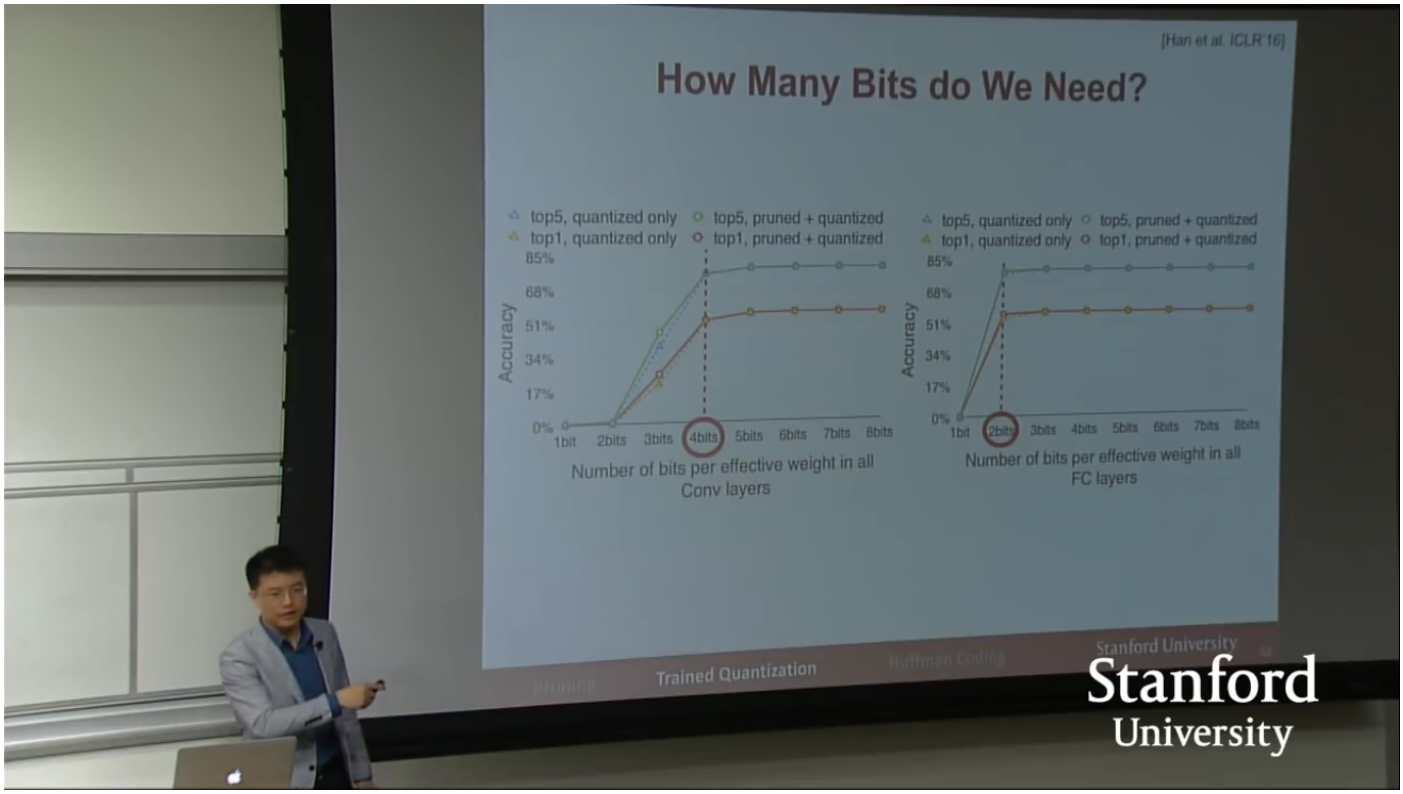

Trained Quantization
[Han et al. ICLR'16]
weights
(32 bit float)
cluster index
(2 bit uint)
centroids
fine-tuned
centroids
cluster
gradient
group by
reduce
× lr
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University

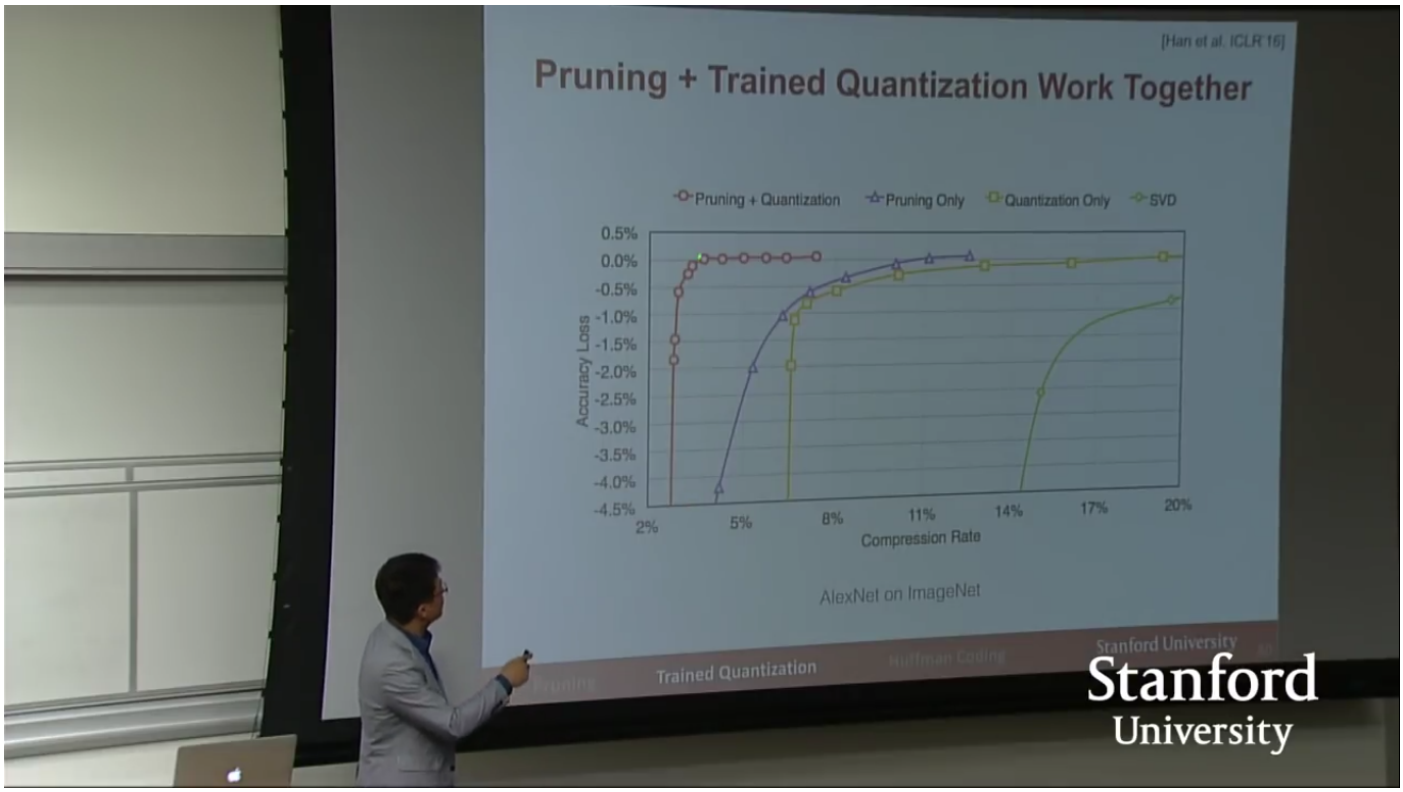

Before Trained Quantization:
Continuous Weight
[Han et al. ICLR'16]
1e3
Count
Weight Value
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University



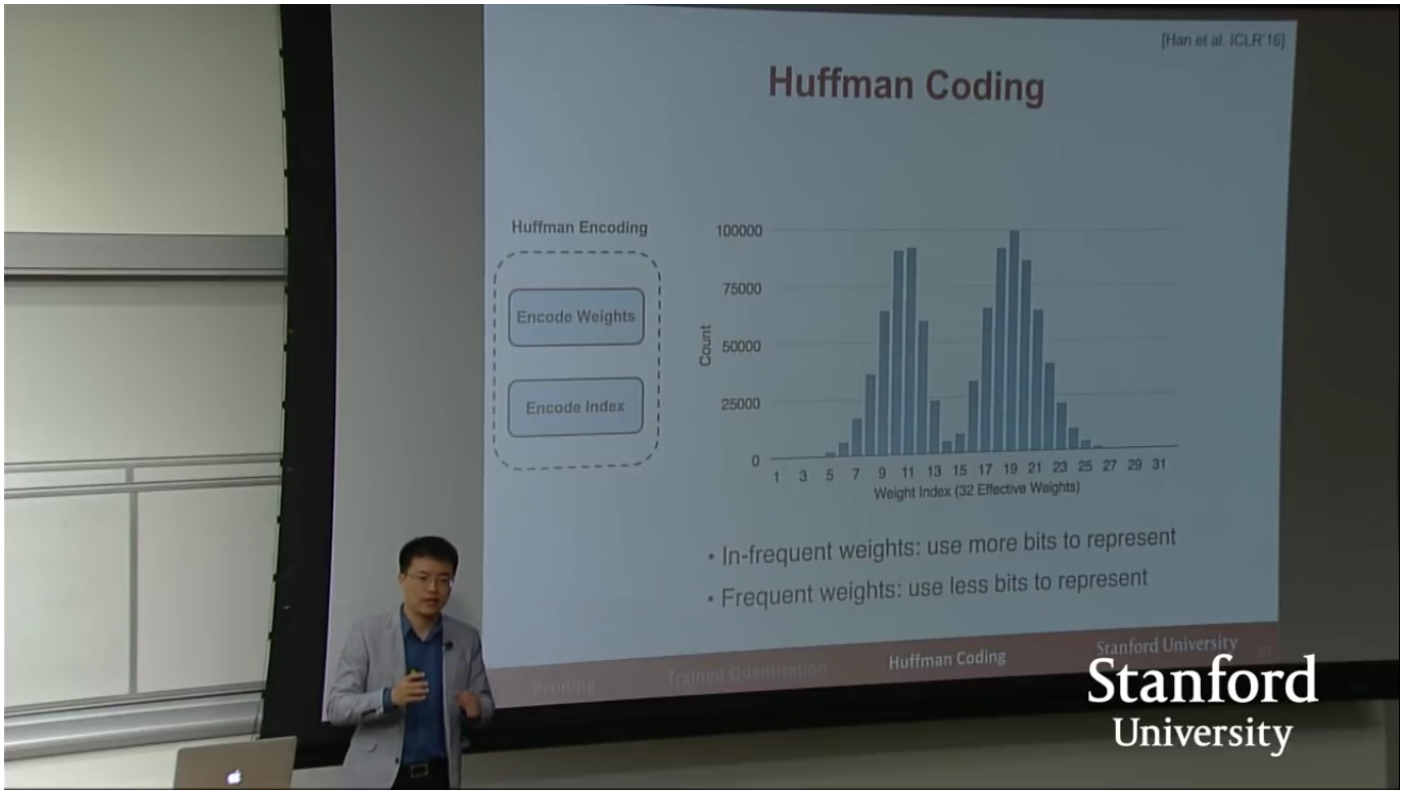
- Before we had Continuous weights and now we have Discrete weights.

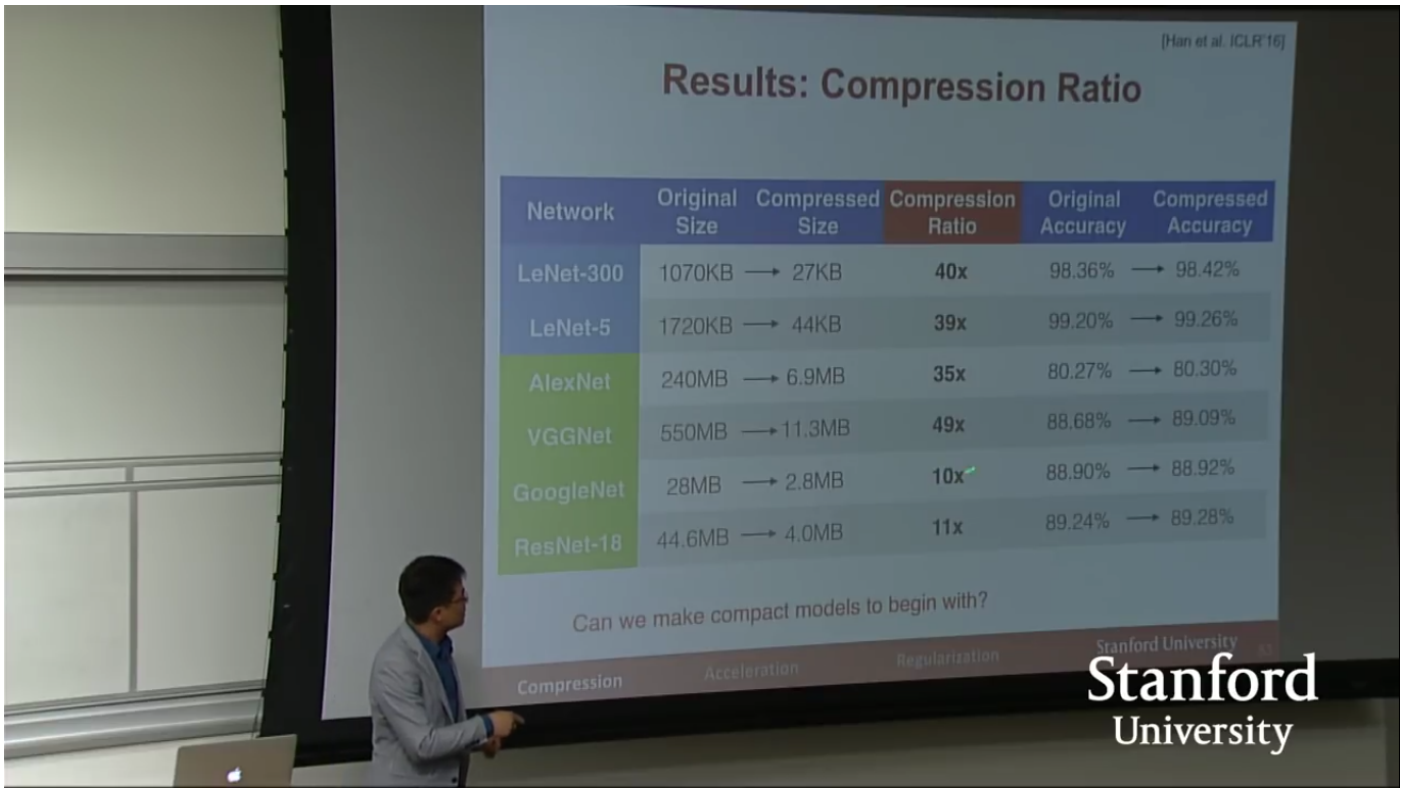


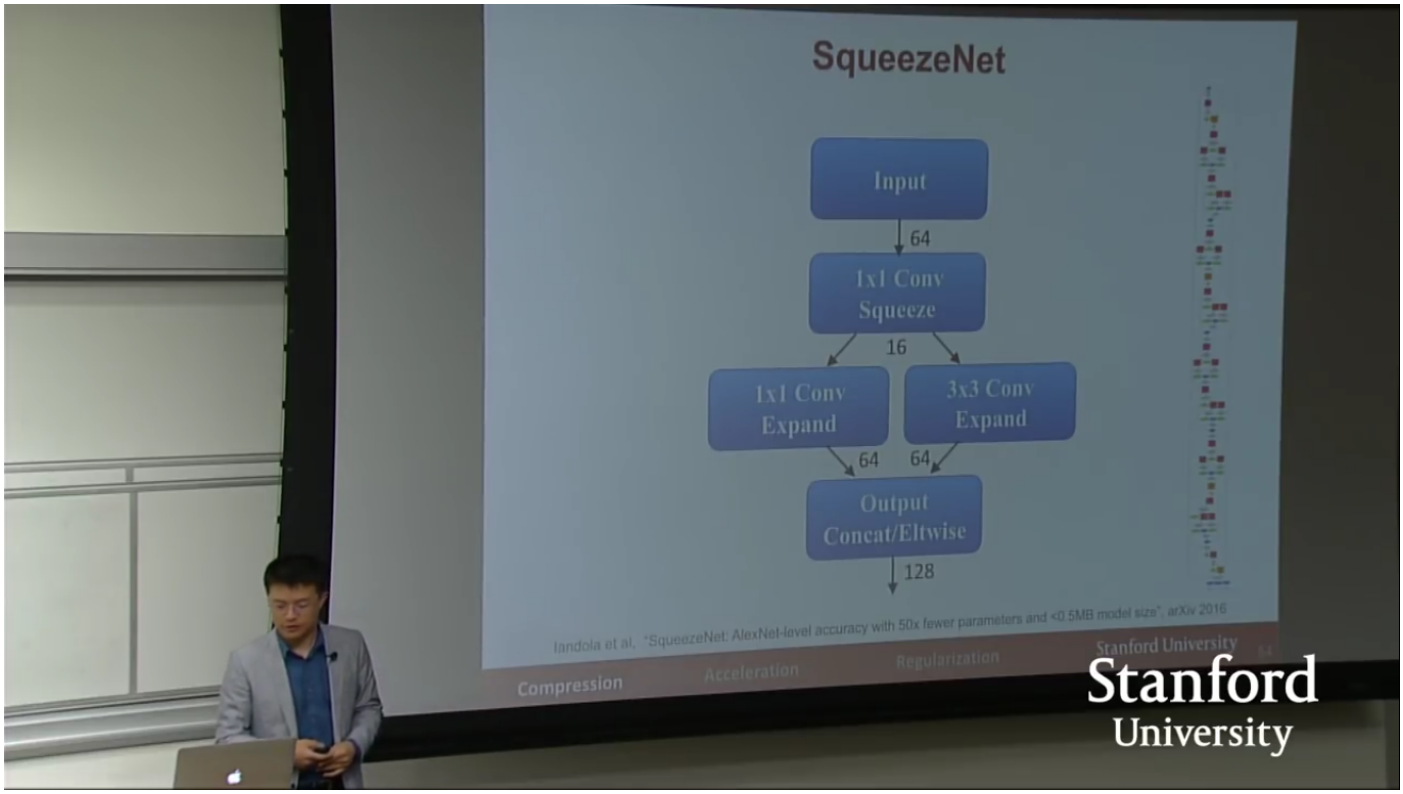


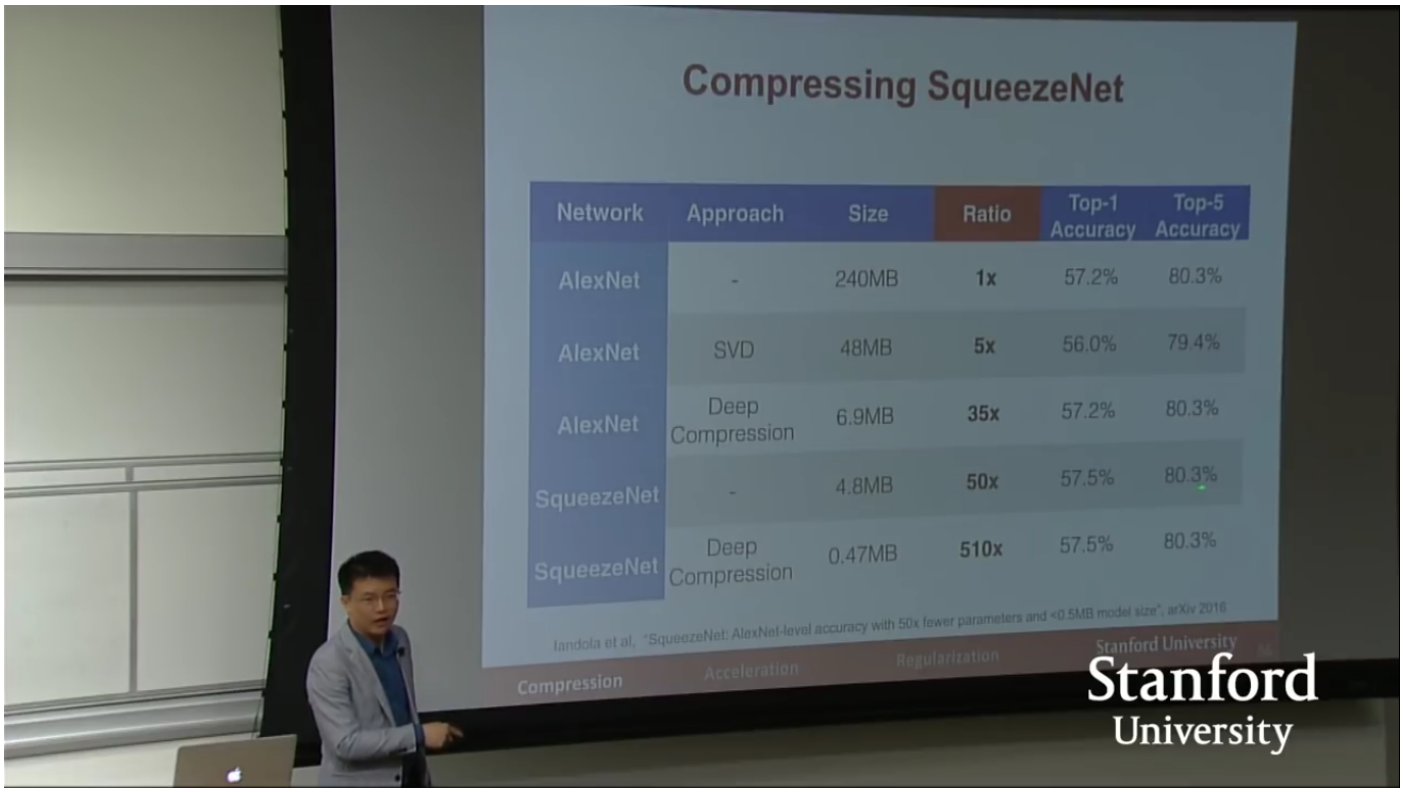


- Compression Rate(Lower the Better). We can notice that when the model is made 3% of its original size, the Accuracy Loss is not degrading.

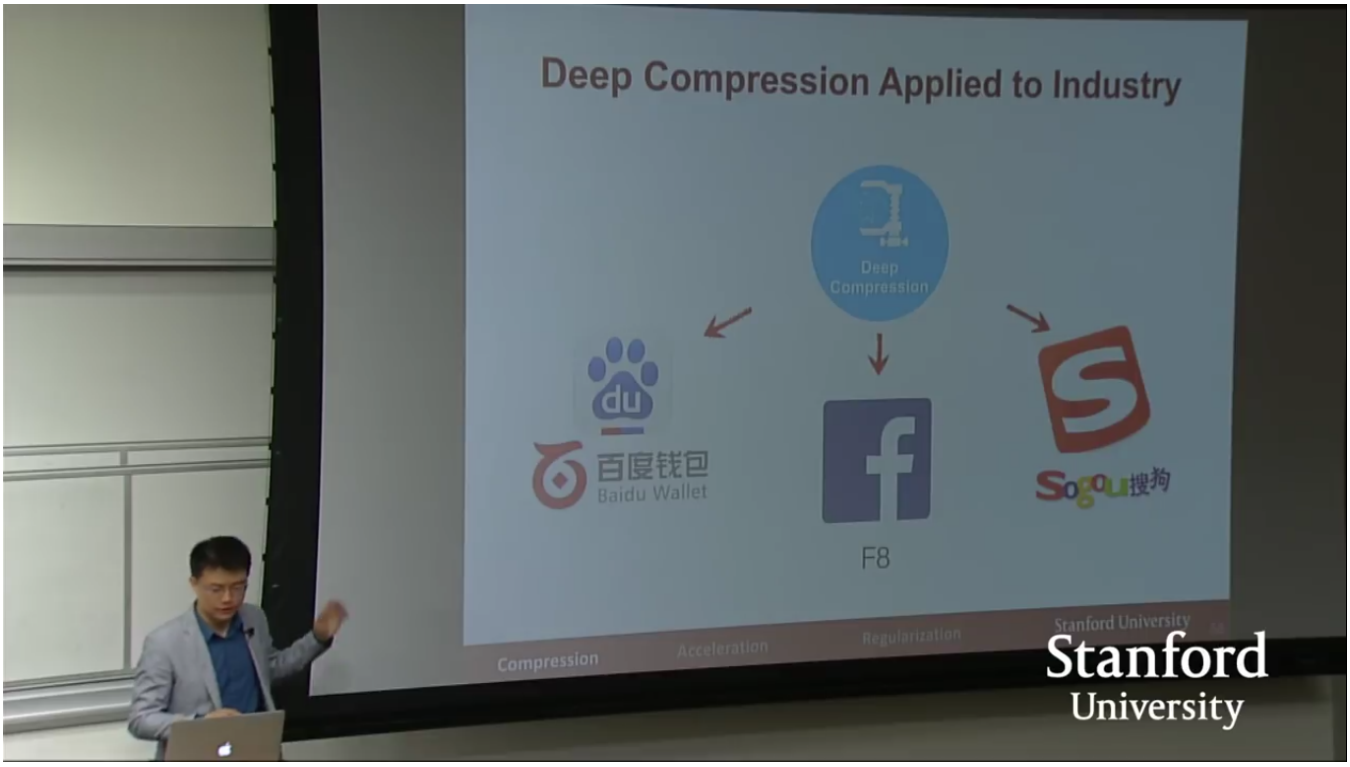

[Han et al. ICLR'16]
Huffman Coding
Huffman Encoding
Encode Weights
Encode Index
100000
75000
50000
25000
0
Count
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Weight Index (32 Effective Weights)
• In-frequent weights: use more bits to represent
• Frequent weights: use less bits to represent
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University


[Han et al. ICLR'16]
Results: Compression Ratio
Network
Original Size
Compressed Size
Compression Ratio
Original Accuracy
Compressed Accuracy
LeNet-300
1070KB → 27KB
40x
98.36% → 98.42%
LeNet-5
1720KB → 44KB
39x
99.20% → 99.26%
AlexNet
240MB → 6.9MB
35x
80.27% → 80.30%
VGGNet
550MB → 11.3MB
49x
88.68% → 89.09%
GoogleNet
28MB → 2.8MB
10x
88.90% → 88.92%
ResNet-18
44.6MB → 4.0MB
11x
89.24% → 89.28%
Can we make compact models to begin with?
Compression
Acceleration
Regularization
Stanford University
Stanford
University

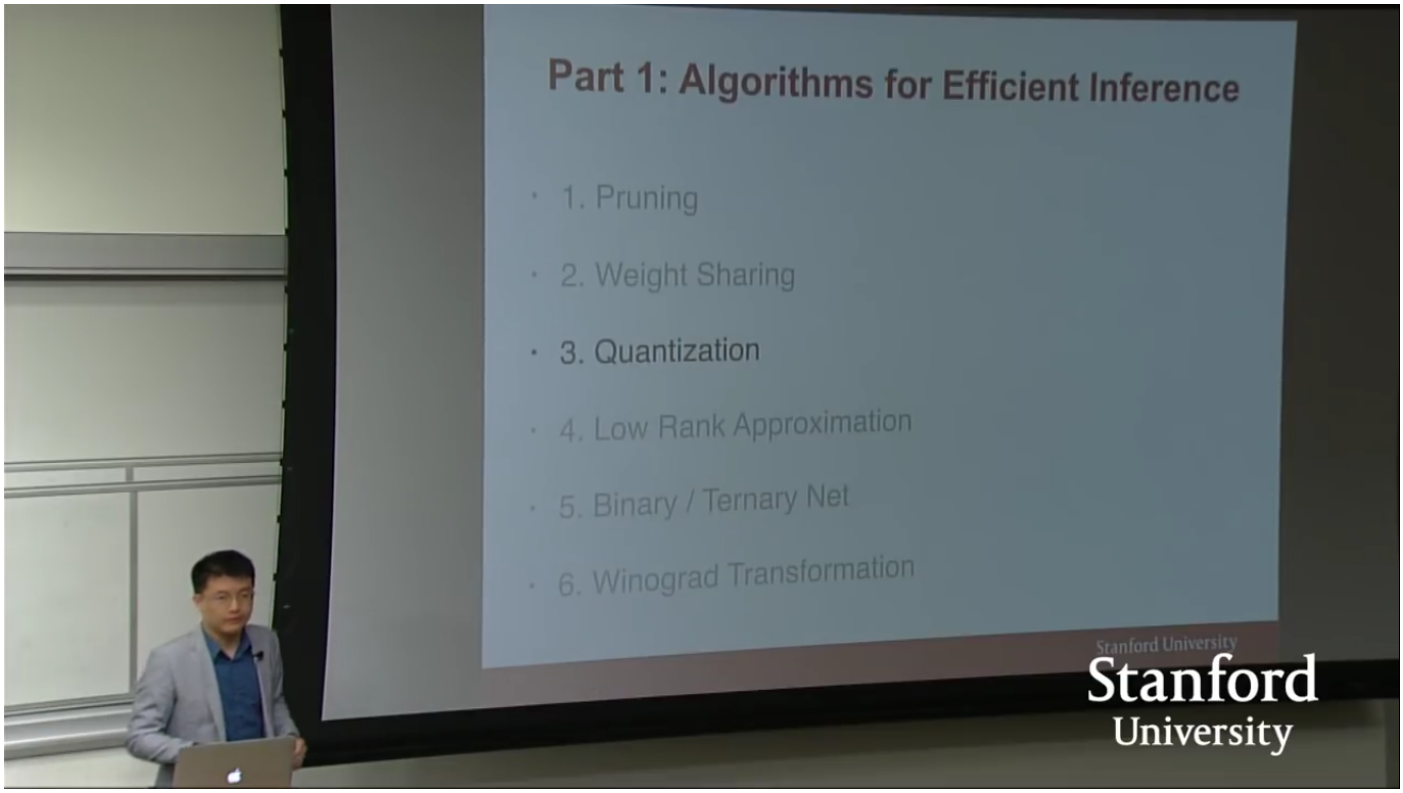




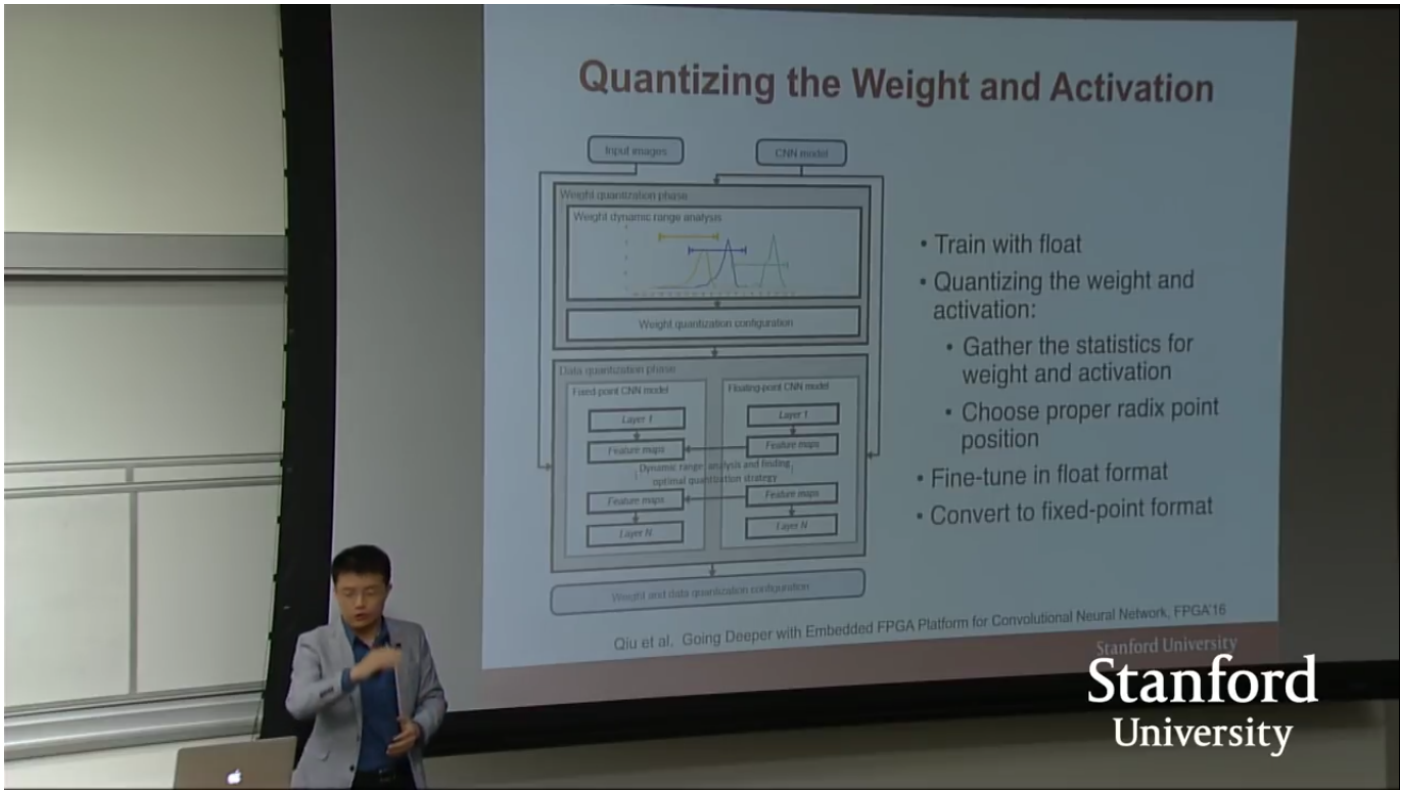
- SqueezeNet is already a small sized model and now after compression it is reducing to 0.47MB and still has the same Top-1 Accuracy as AlexNet. Due to this very small size it can even be placed in cache of SRAMs.

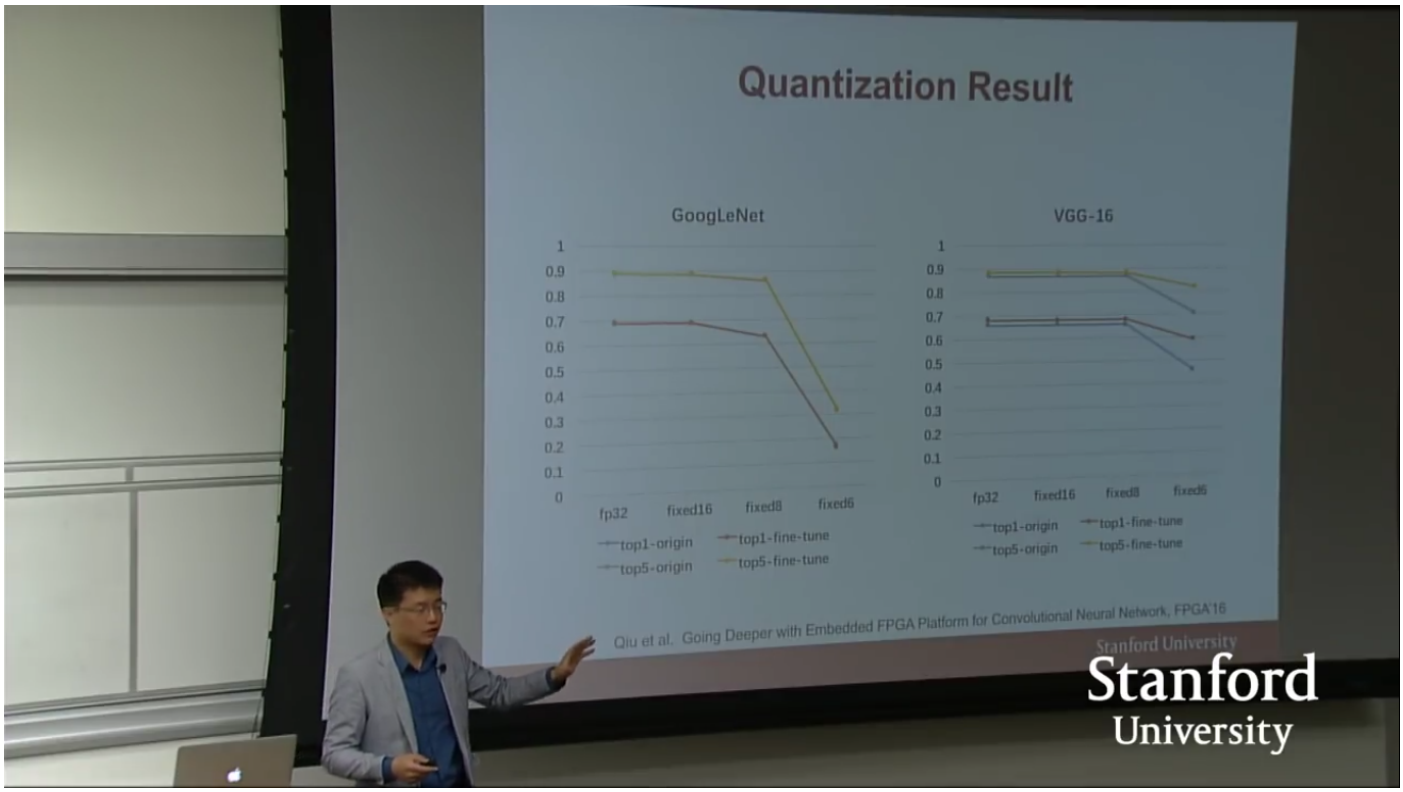
- 

Quantization:

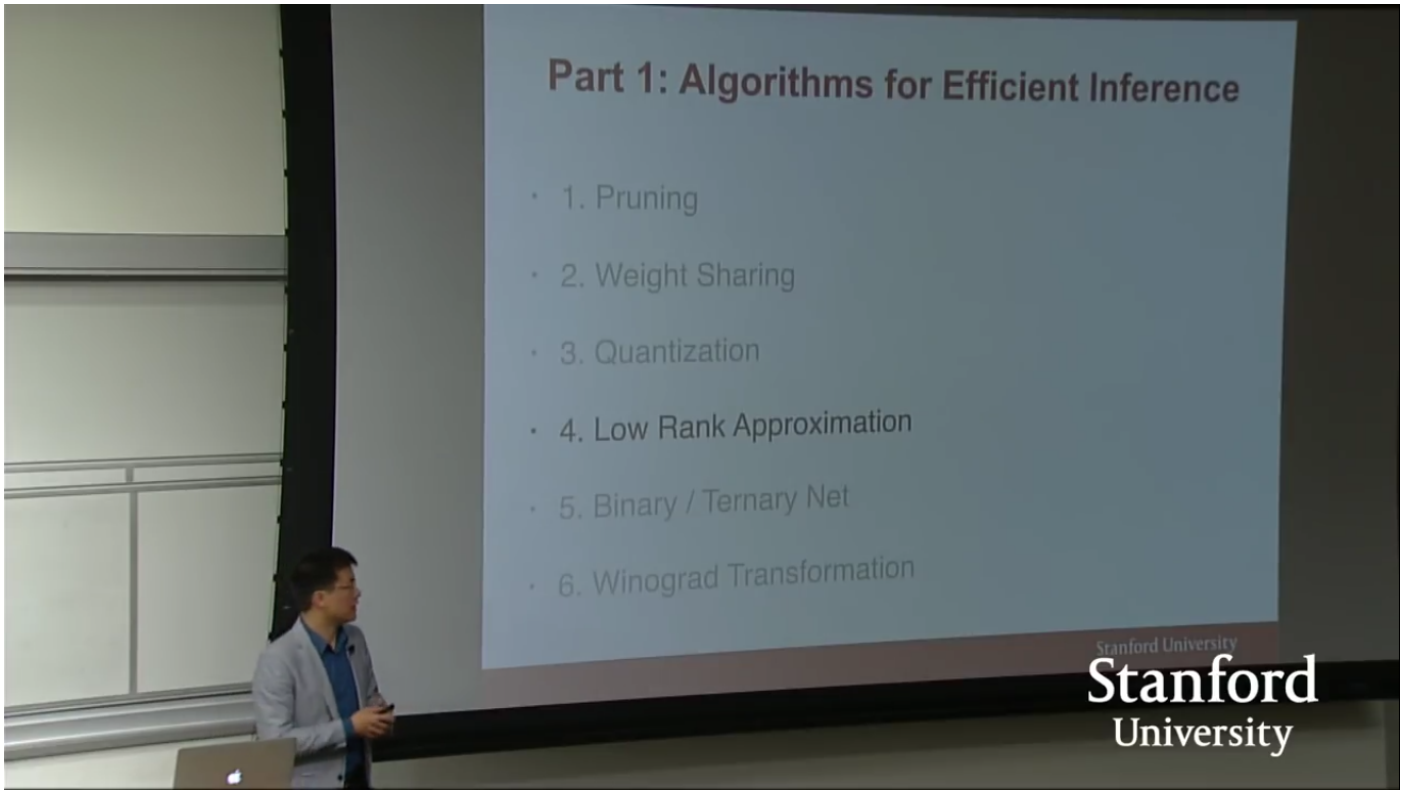


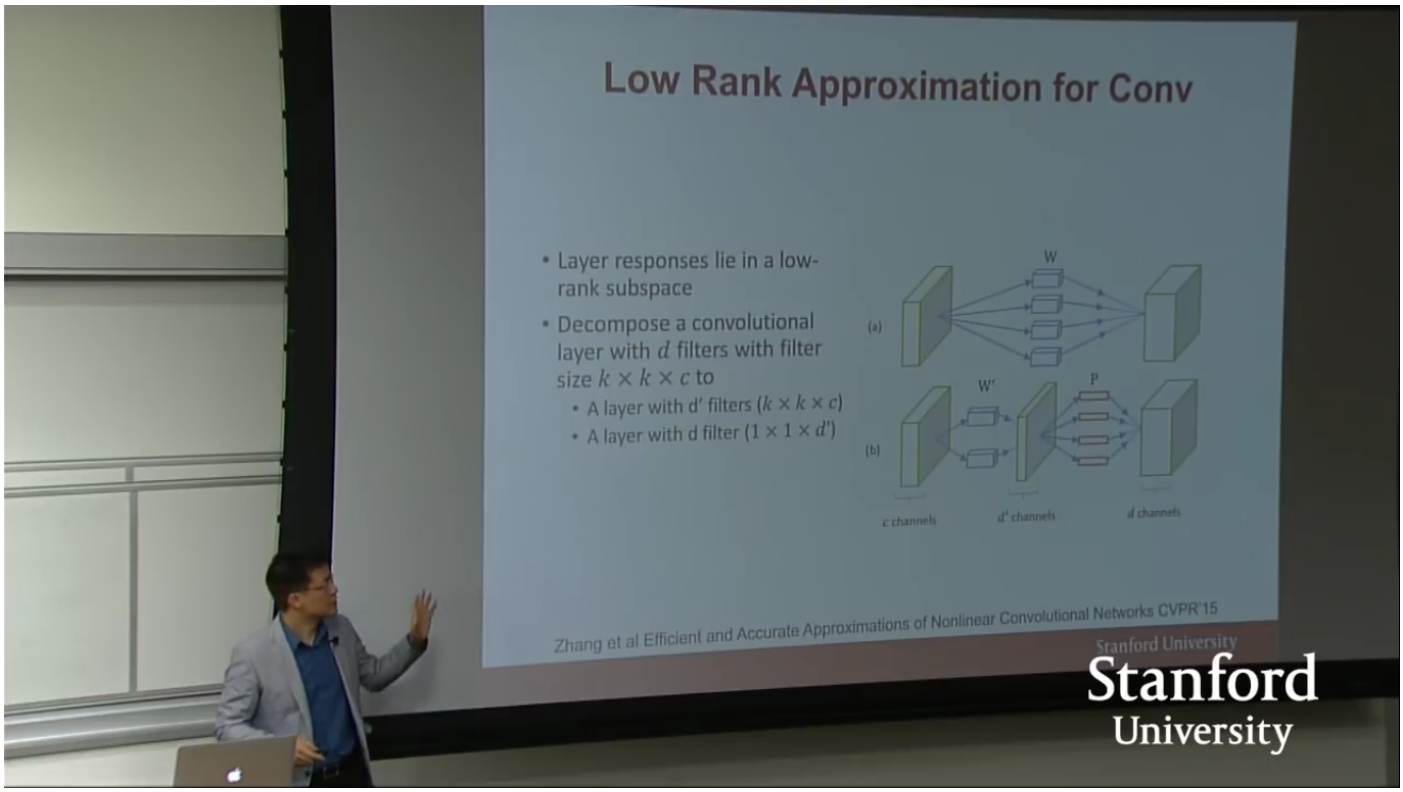
- TPUs widely use Quantization.
- In TPUs just 8 bits during inference.



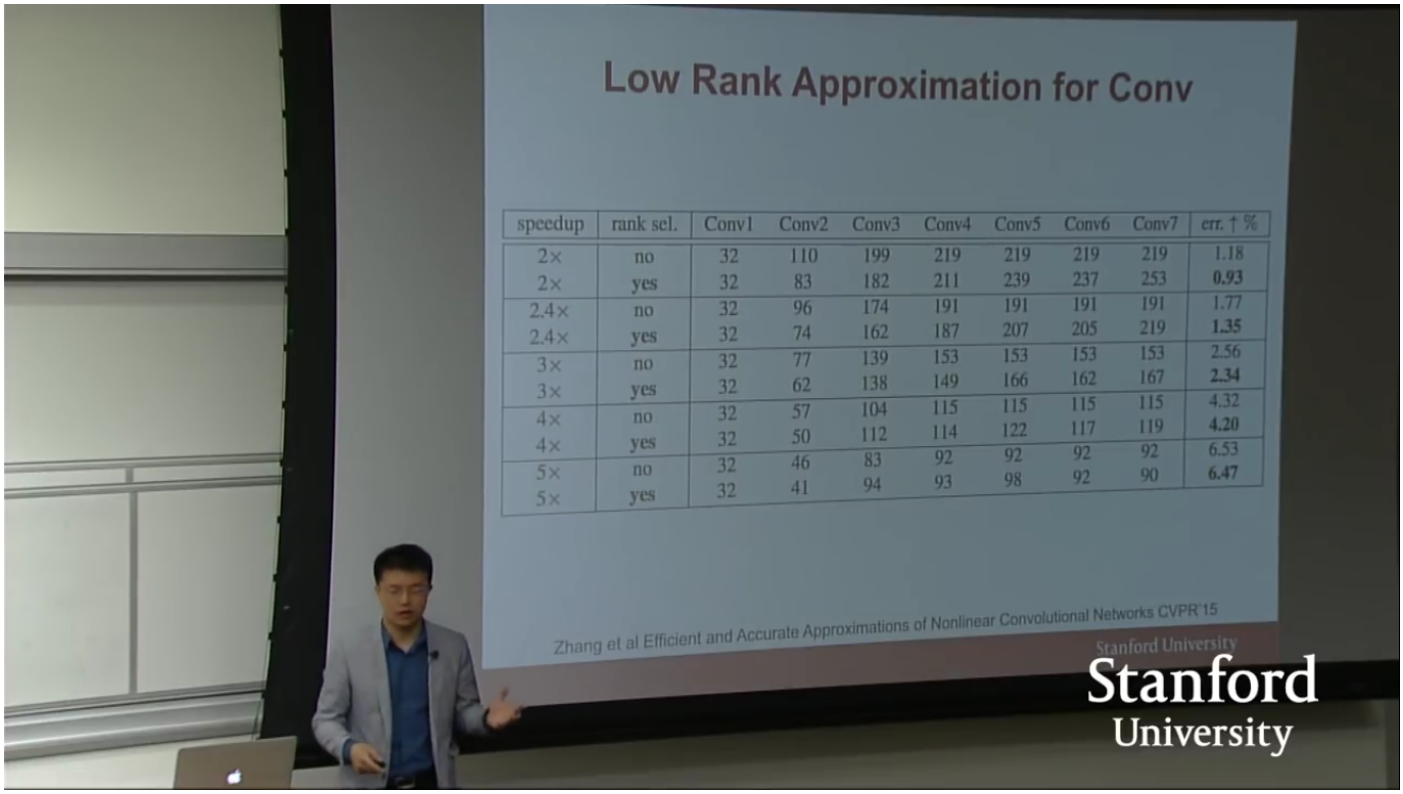


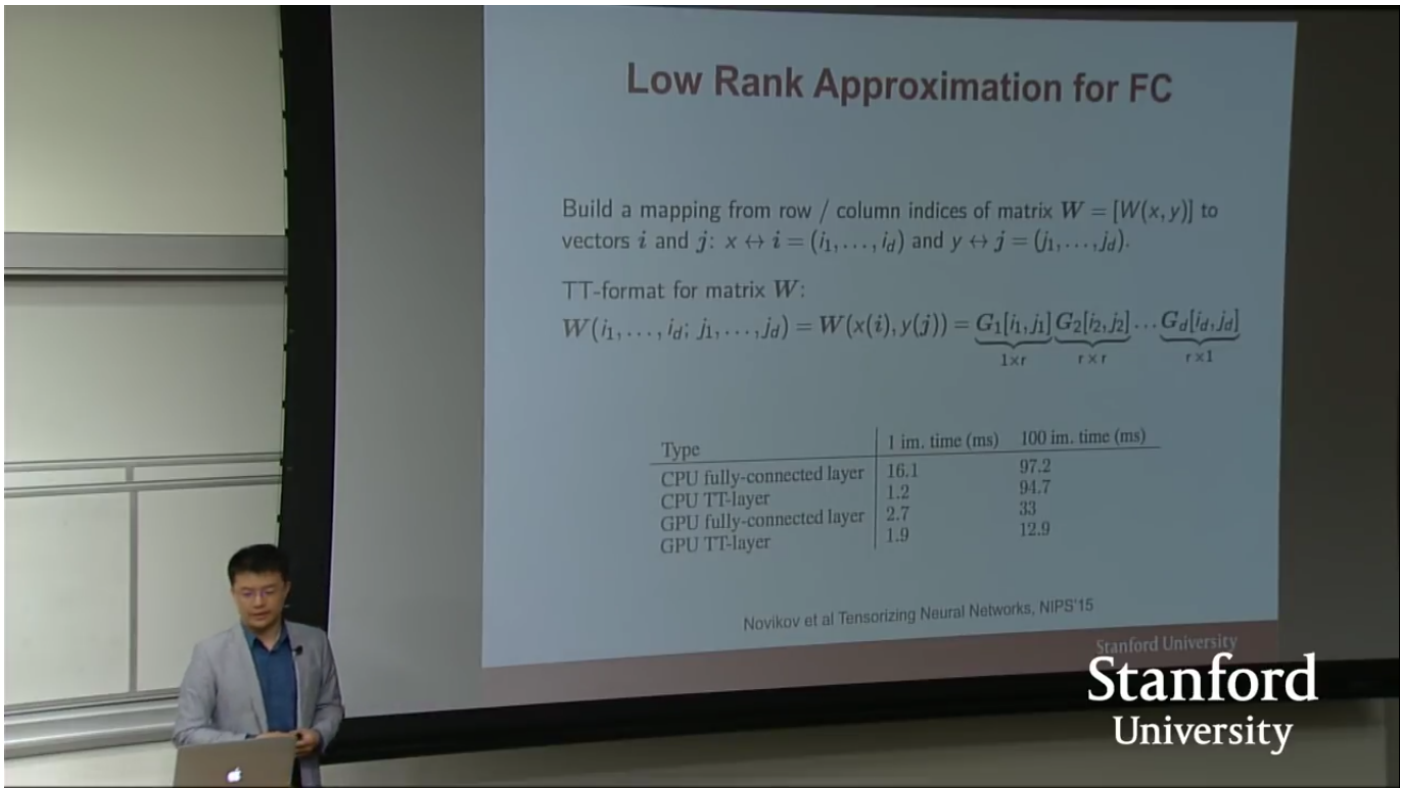
Low Rank Approximation:





- We are breaking a CONV layer into 2 CONV layers

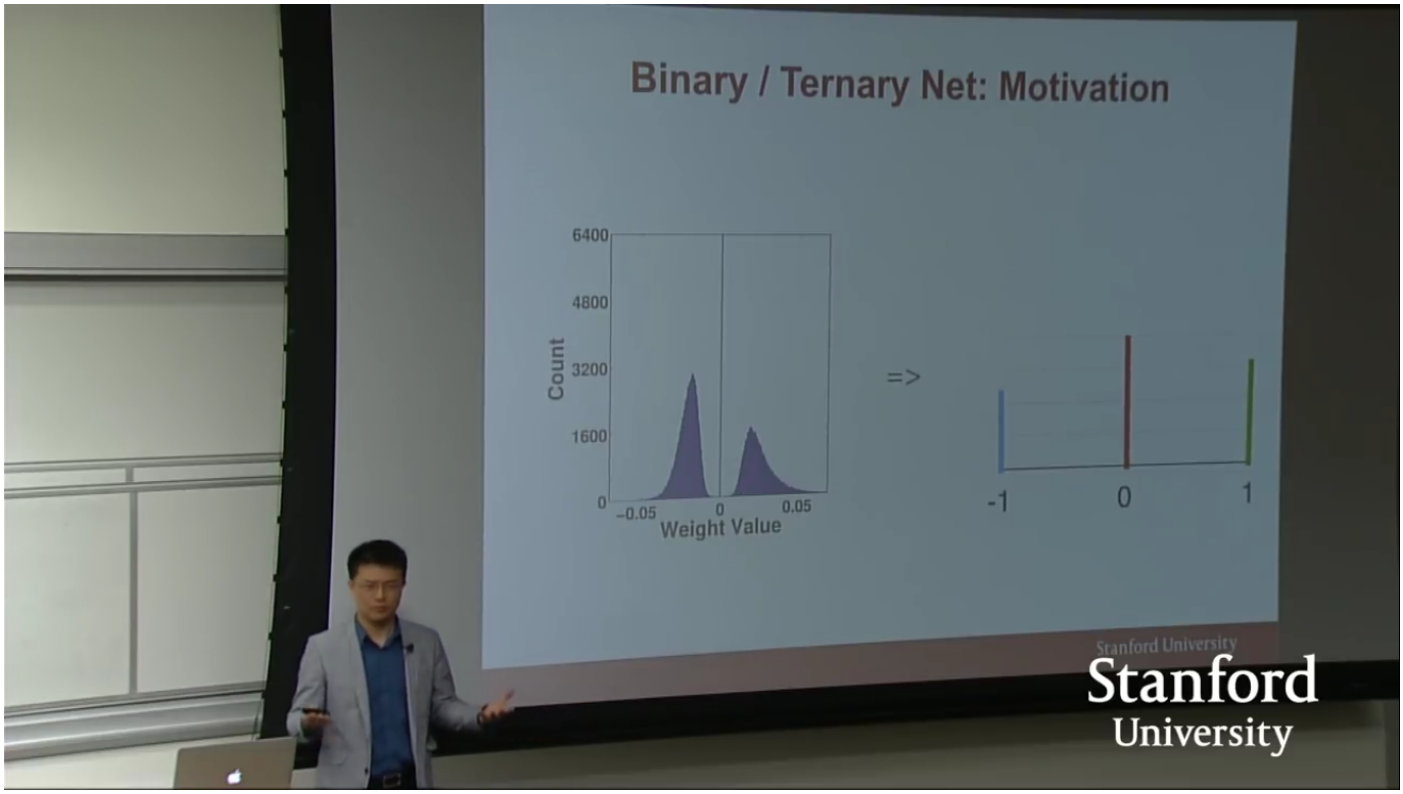


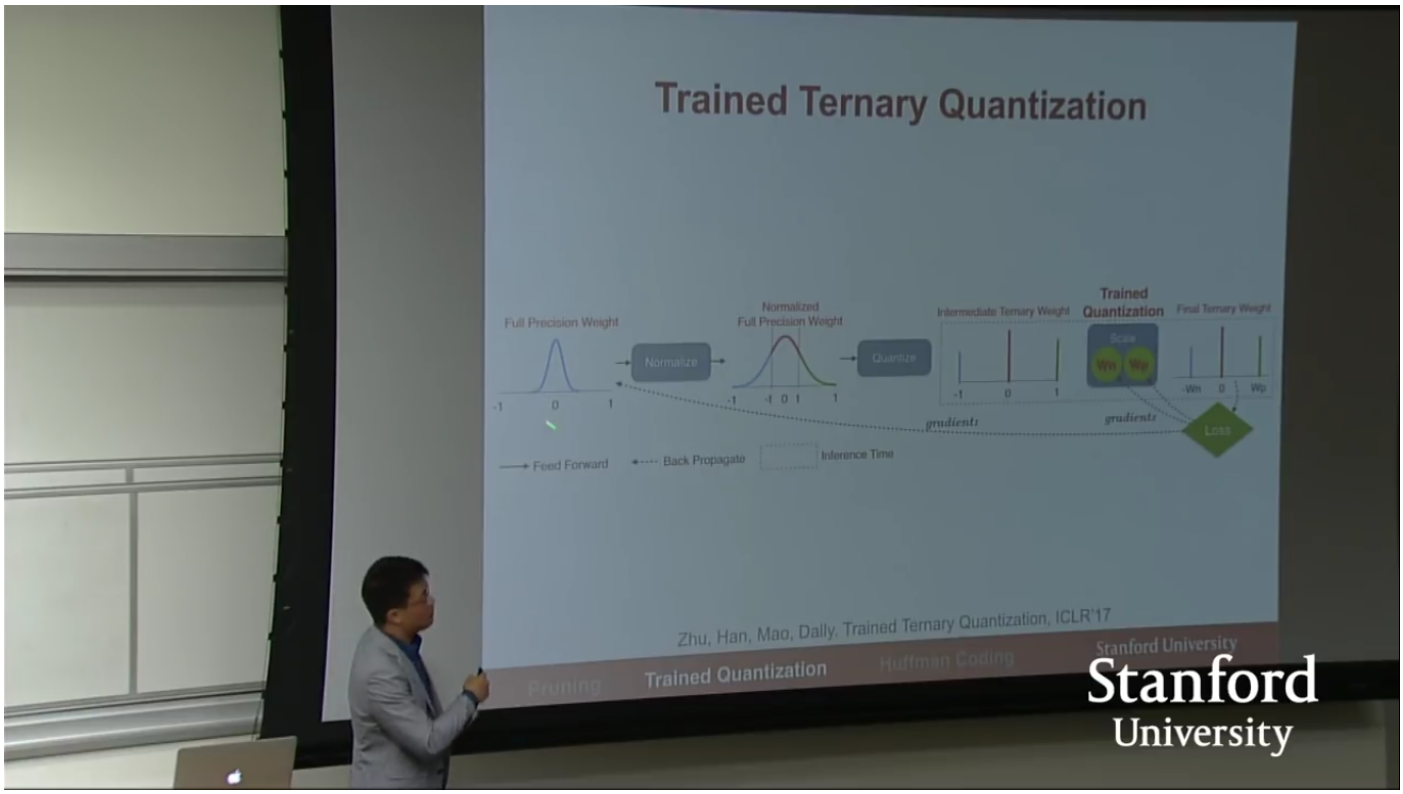


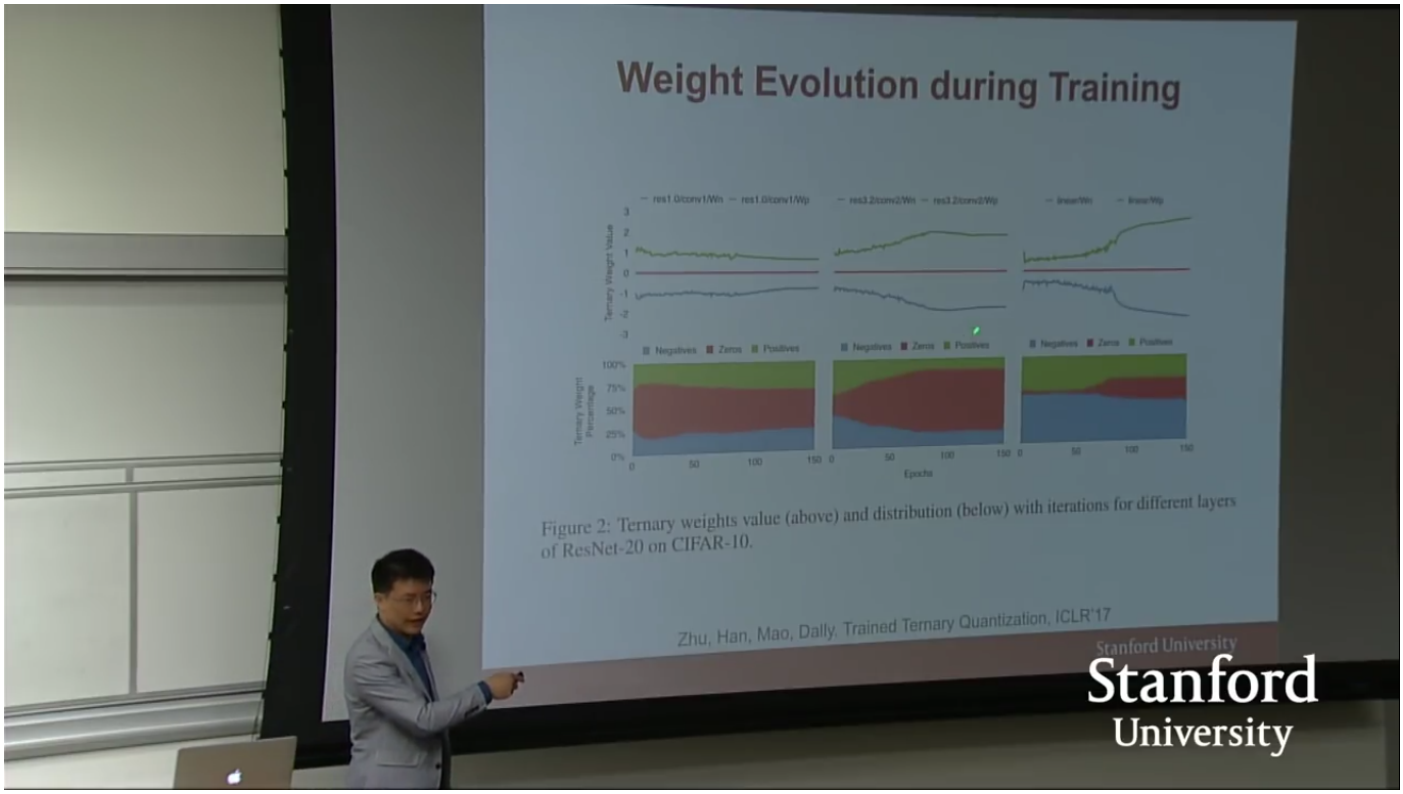
- We are using Tensor Tree to break down one fully connected layer into a tree of fully connected layers.

###Binary and Ternary Net:

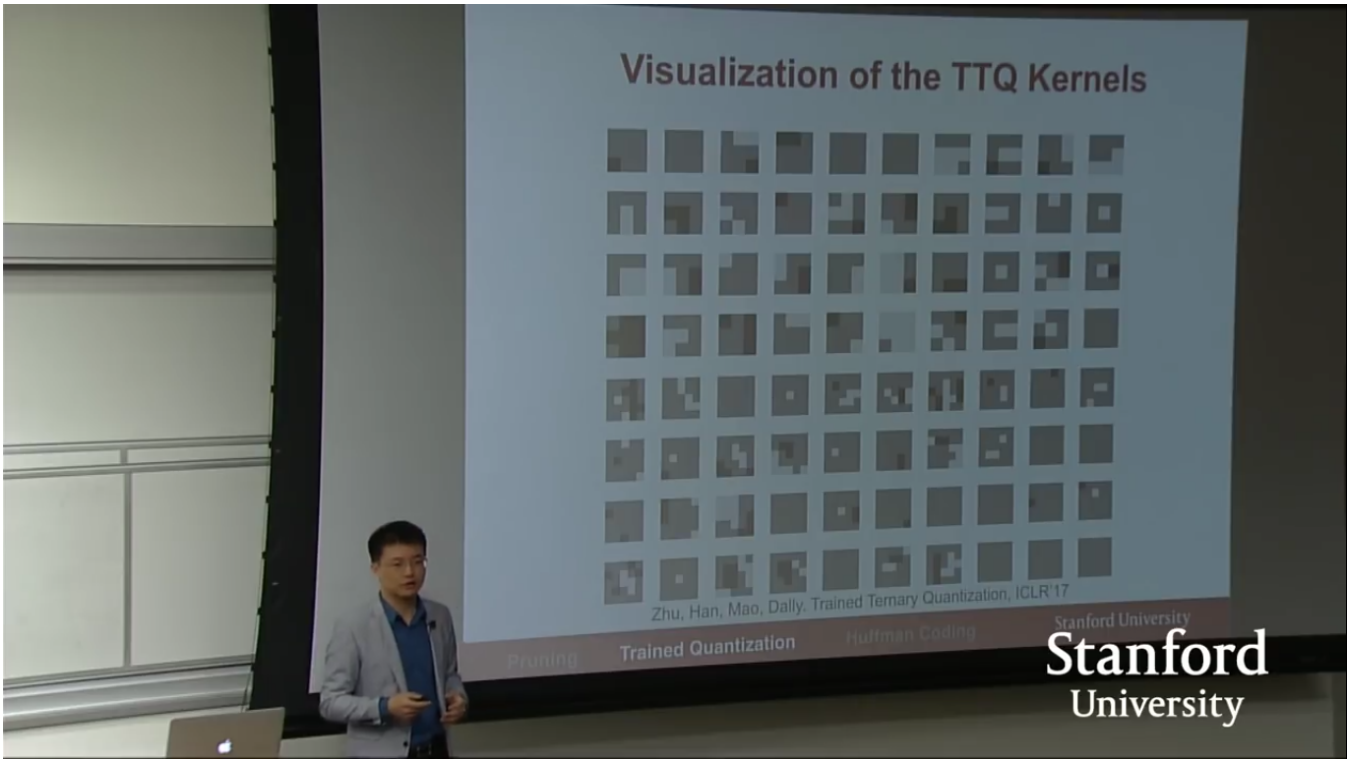
- Can we use only 2 or 3 weights to represent the Neural Network?


Binary / Ternary Net: Motivation
6400
4800
3200
1600
0
Count
−0.05
0
0.05
Weight Value
=>
-1
0
1
Stanford University
Stanford
University


Trained Ternary Quantization
Full Precision Weight
Normalize
Normalized
Full Precision Weight
Quantize
Intermediate Ternary Weight
Trained
Quantization
Scale
Wn
Wp
Final Ternary Weight
-Wn
0
Wp
gradients
gradients
Loss
Feed Forward
Back Propagate
Inference Time
Zhu, Han, Mao, Dally. Trained Ternary Quantization, ICLR'17
Pruning
Trained Quantization
Huffman Coding
Stanford University
Stanford
University

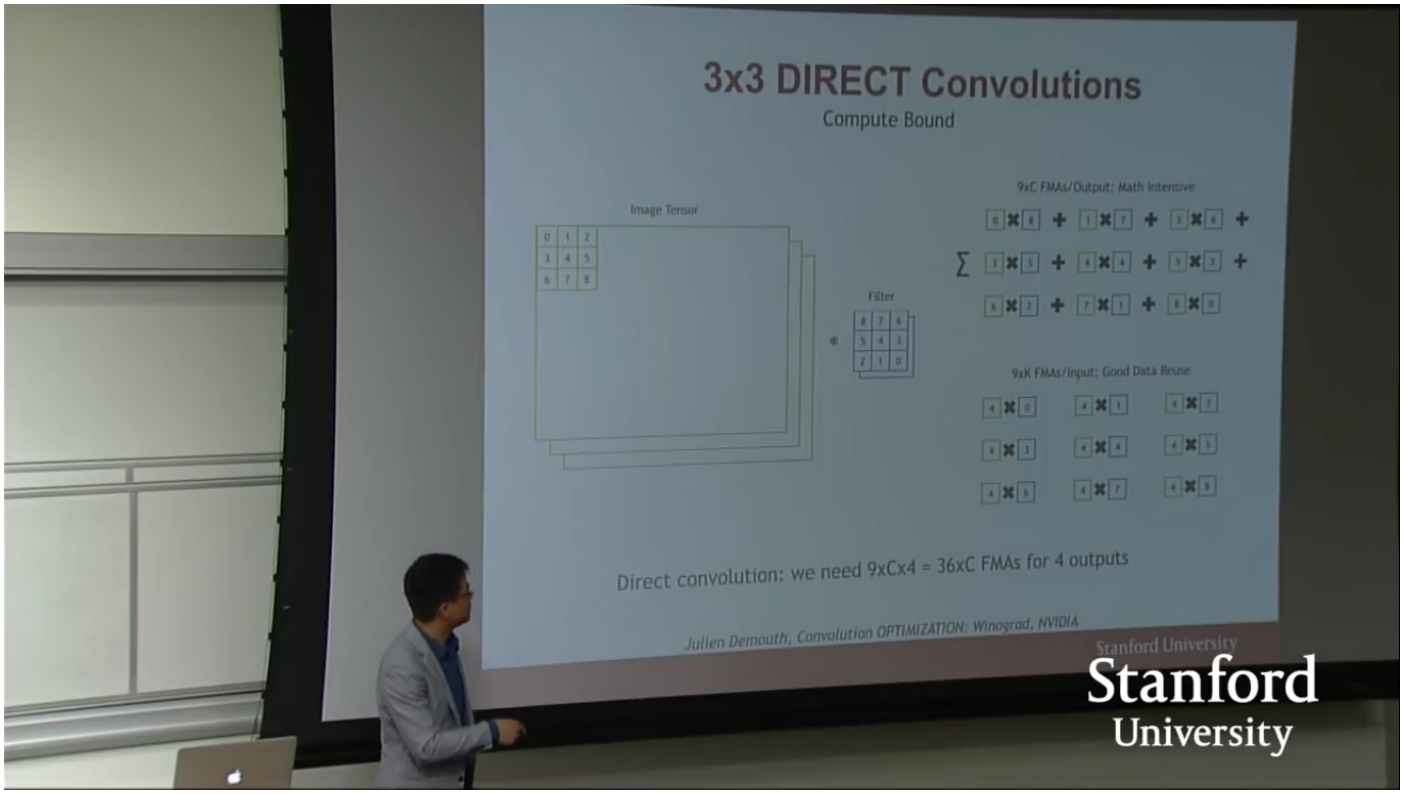


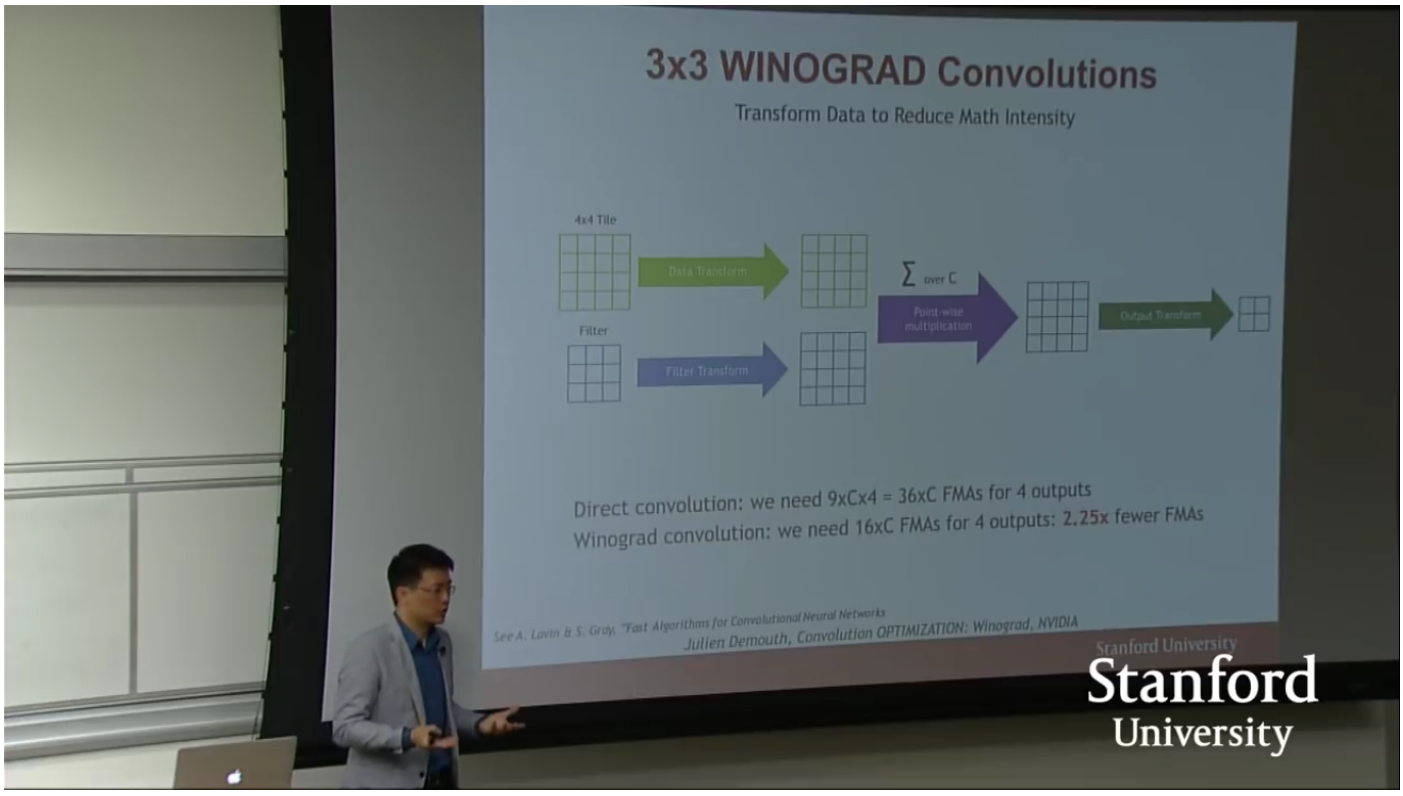
- Visualization of the Trained Ternary Quantization



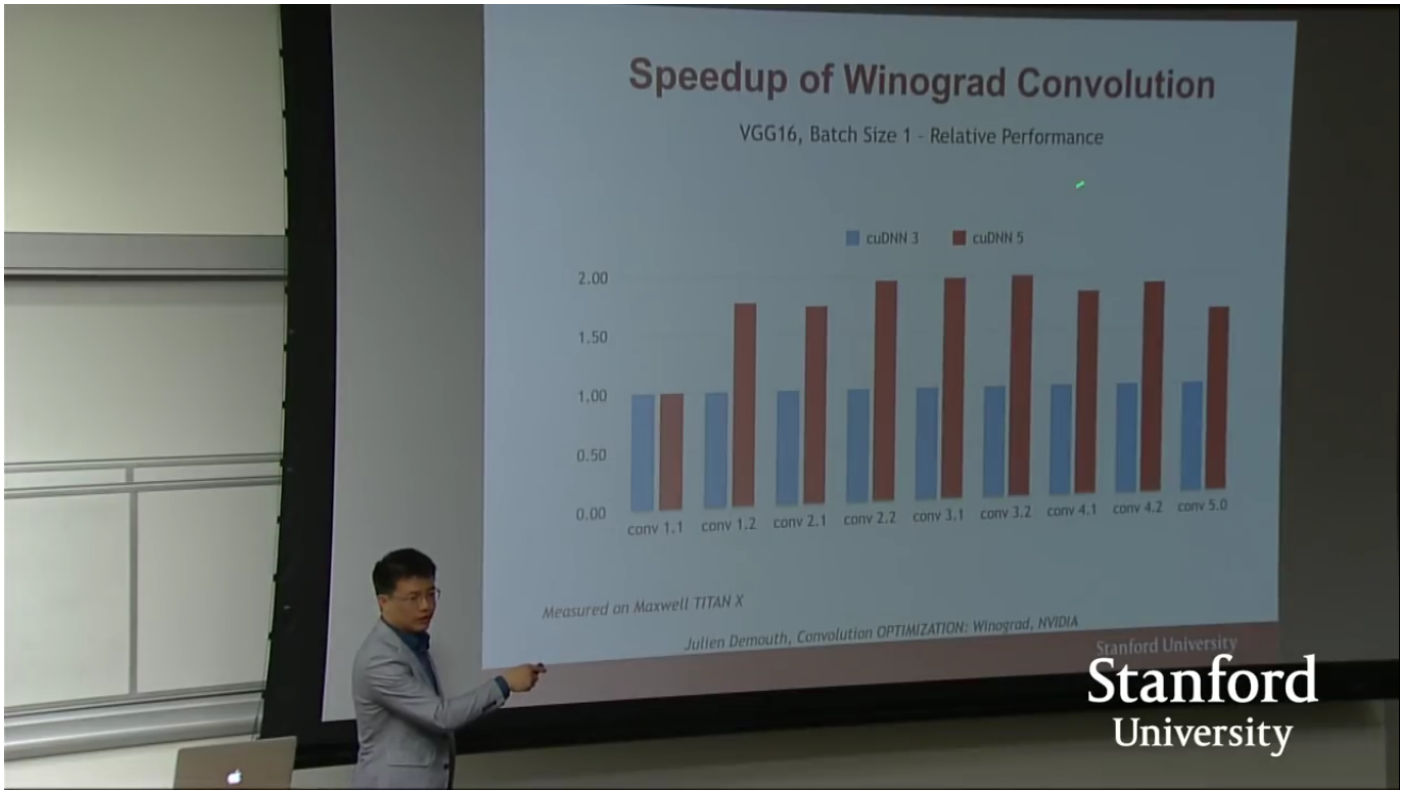
- We can find many corner detection and edge detection kernels being learnt.

Winograd Transformation:

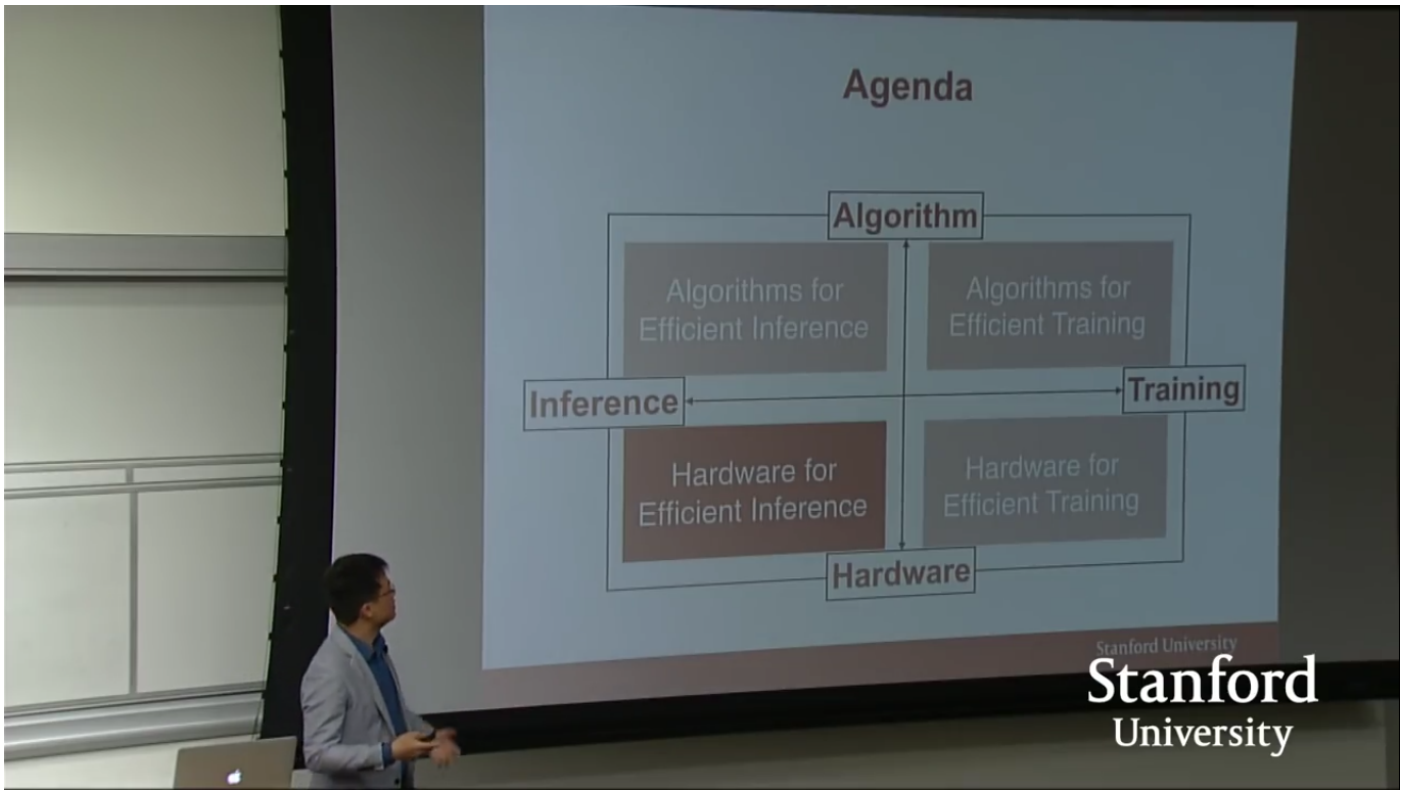


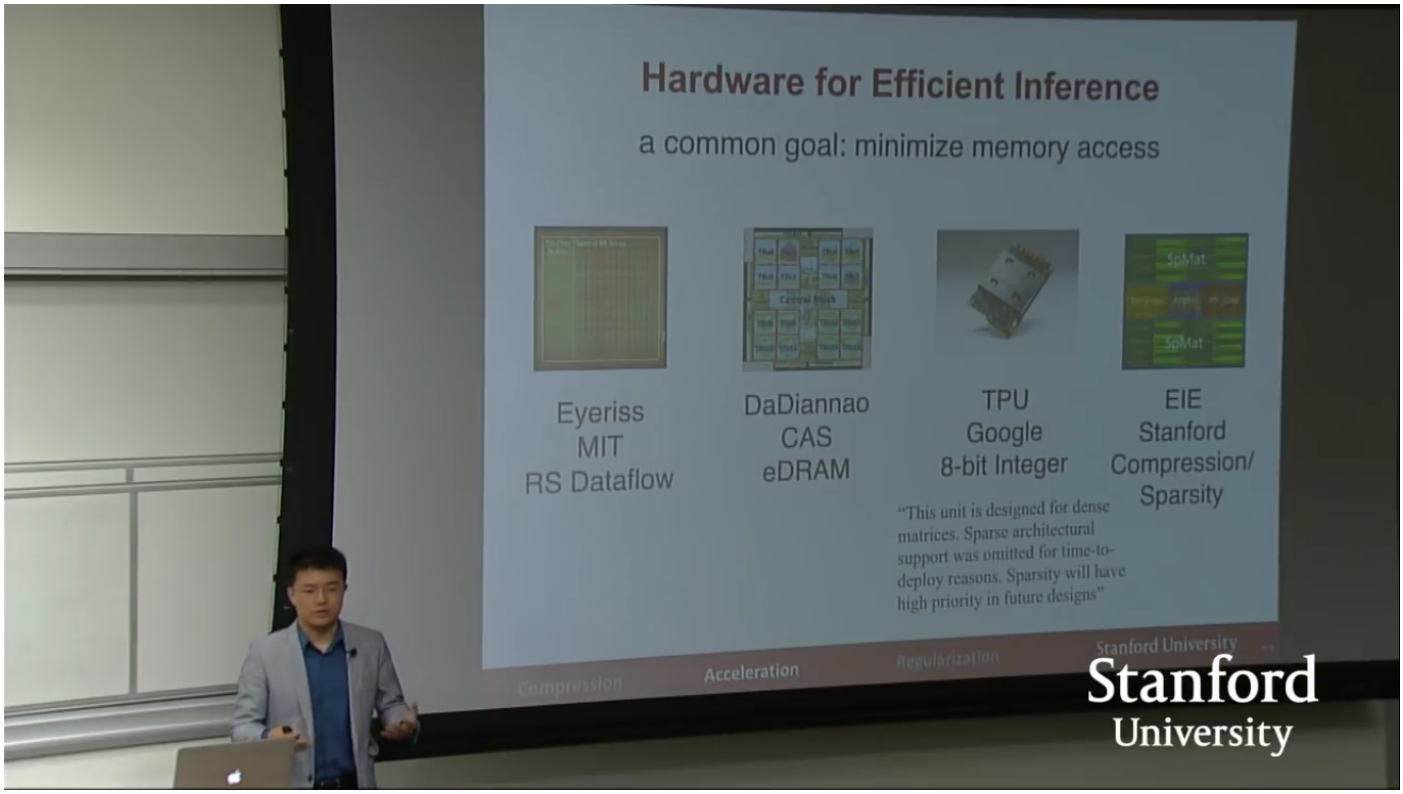


- This is not a lossy method, this is equivalent to the normal CONV operation.

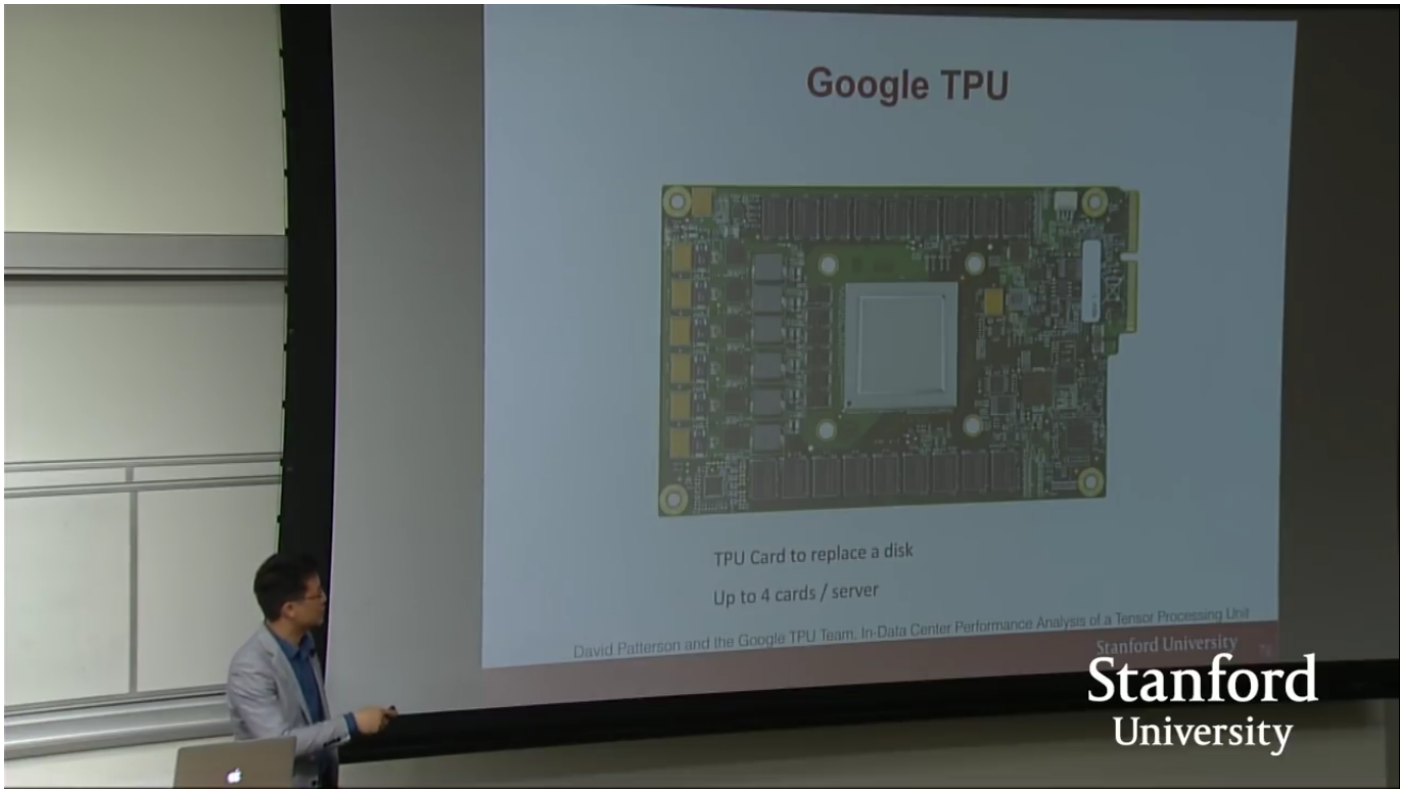


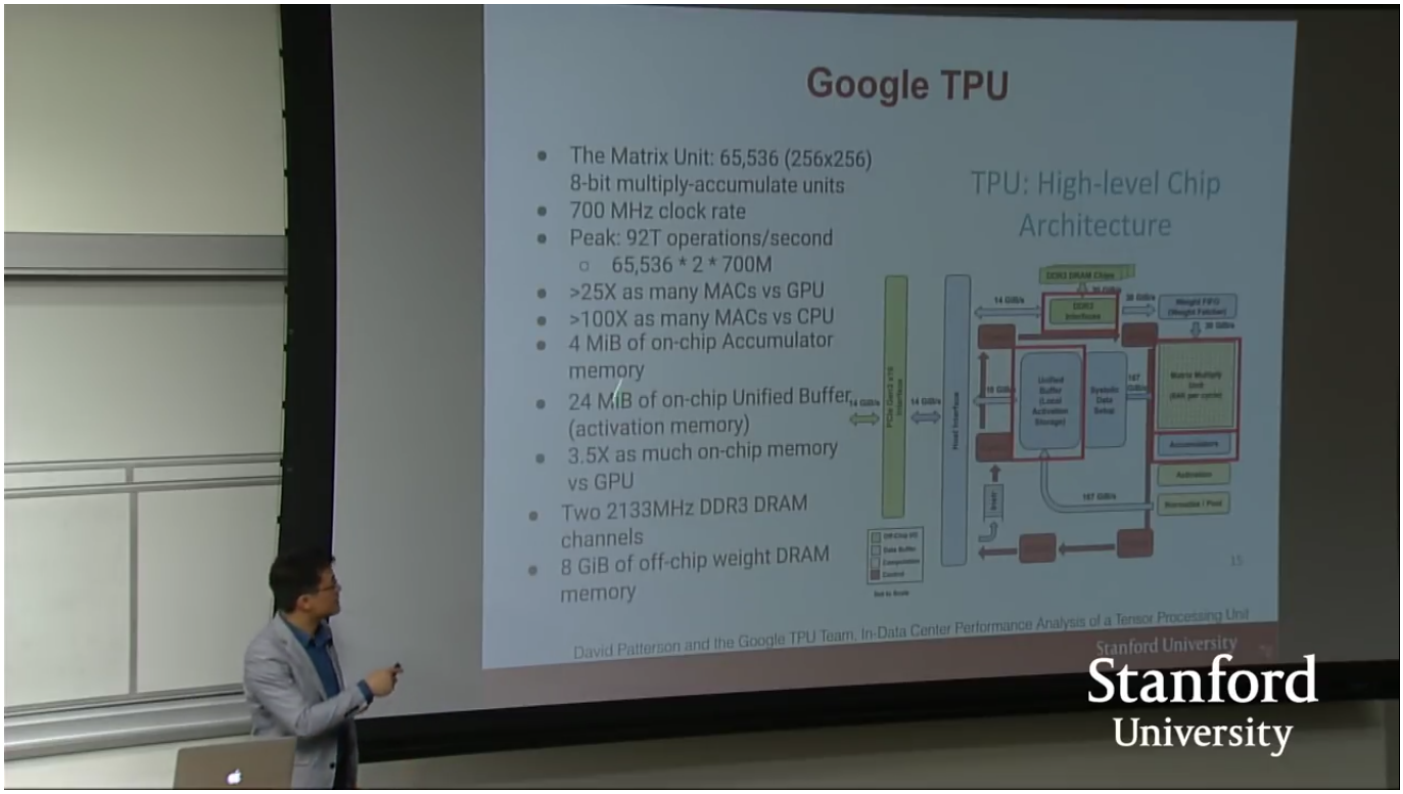
Optimal hardware for efficient inference:



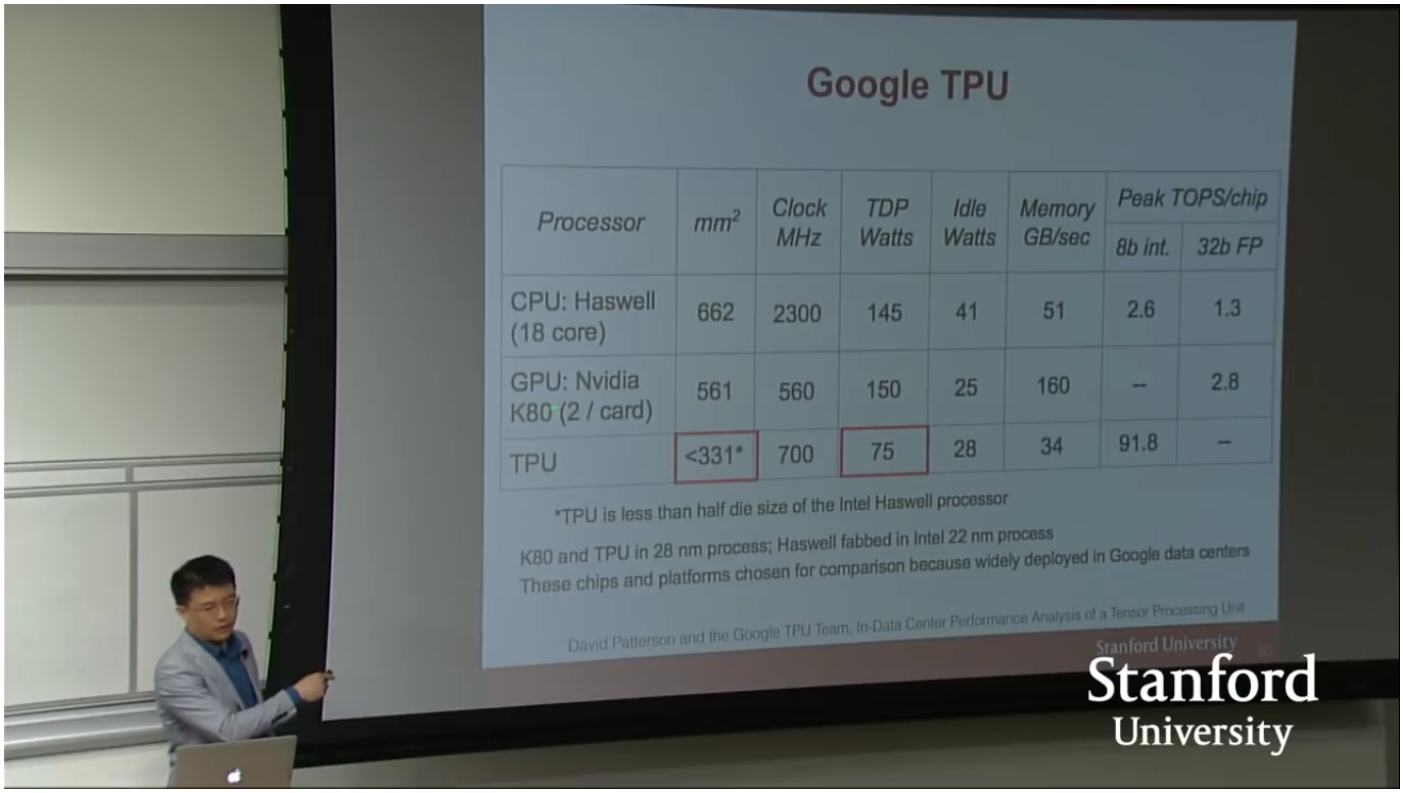


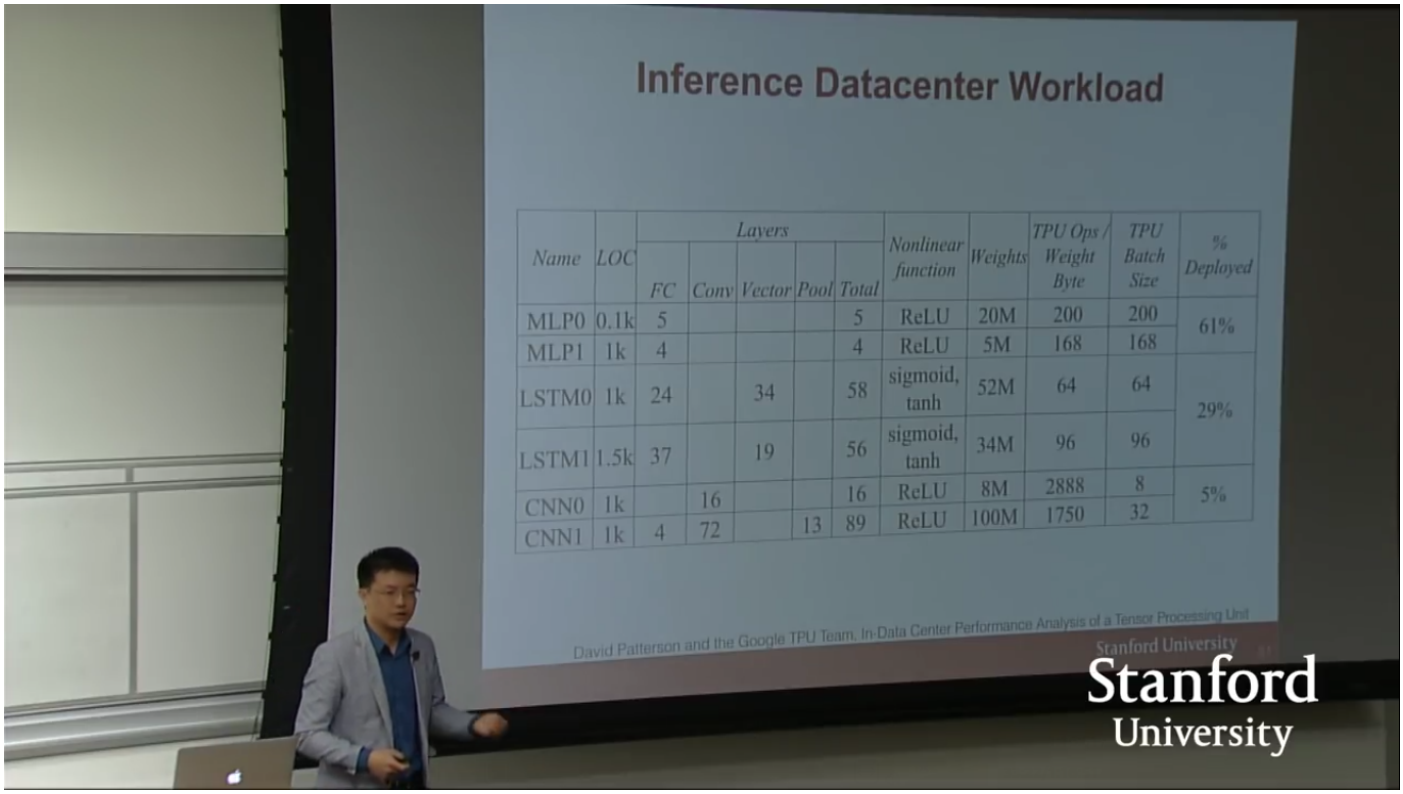
- The common goal is to minimize memory access.





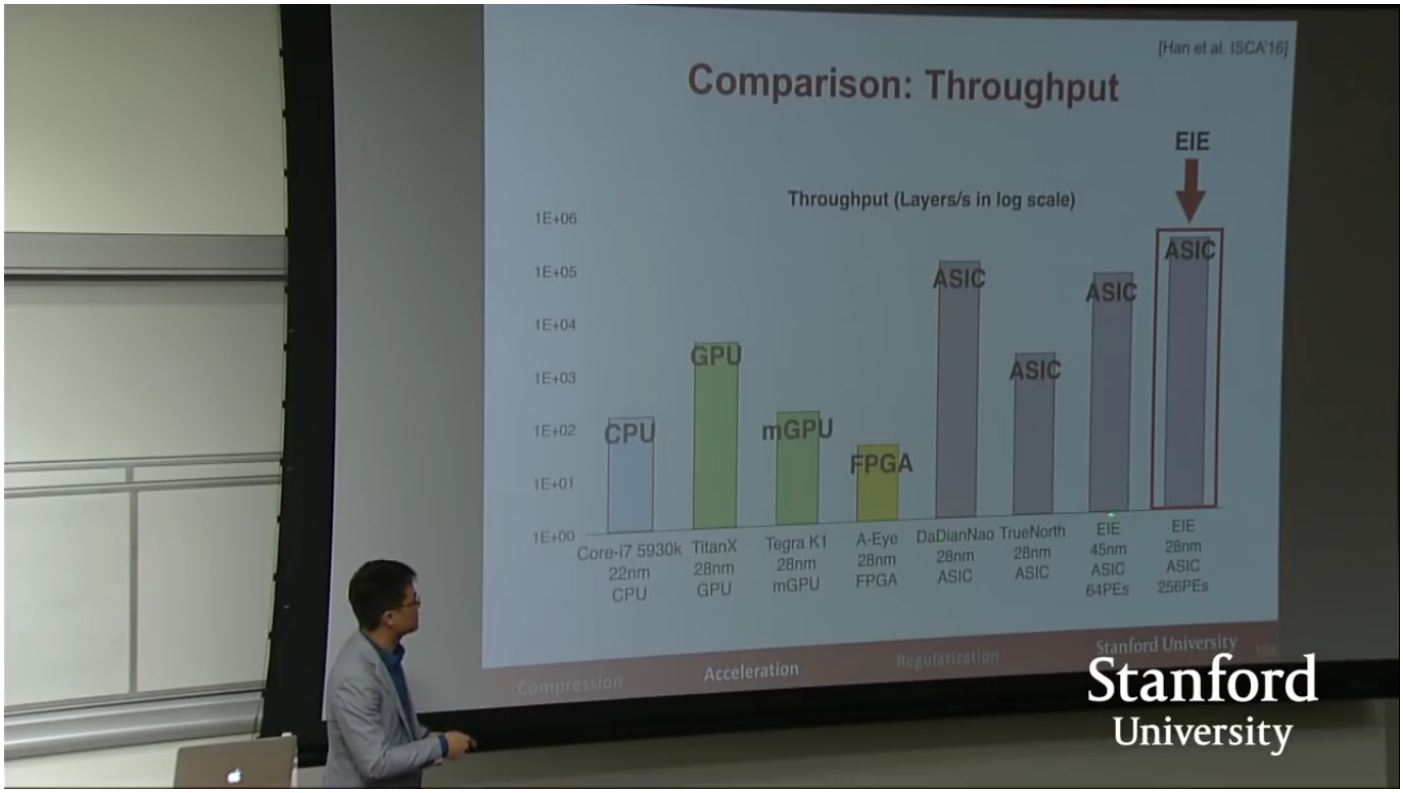
- In 1 cycle the TPU can perform 65536 multiplication - accumulation operation.



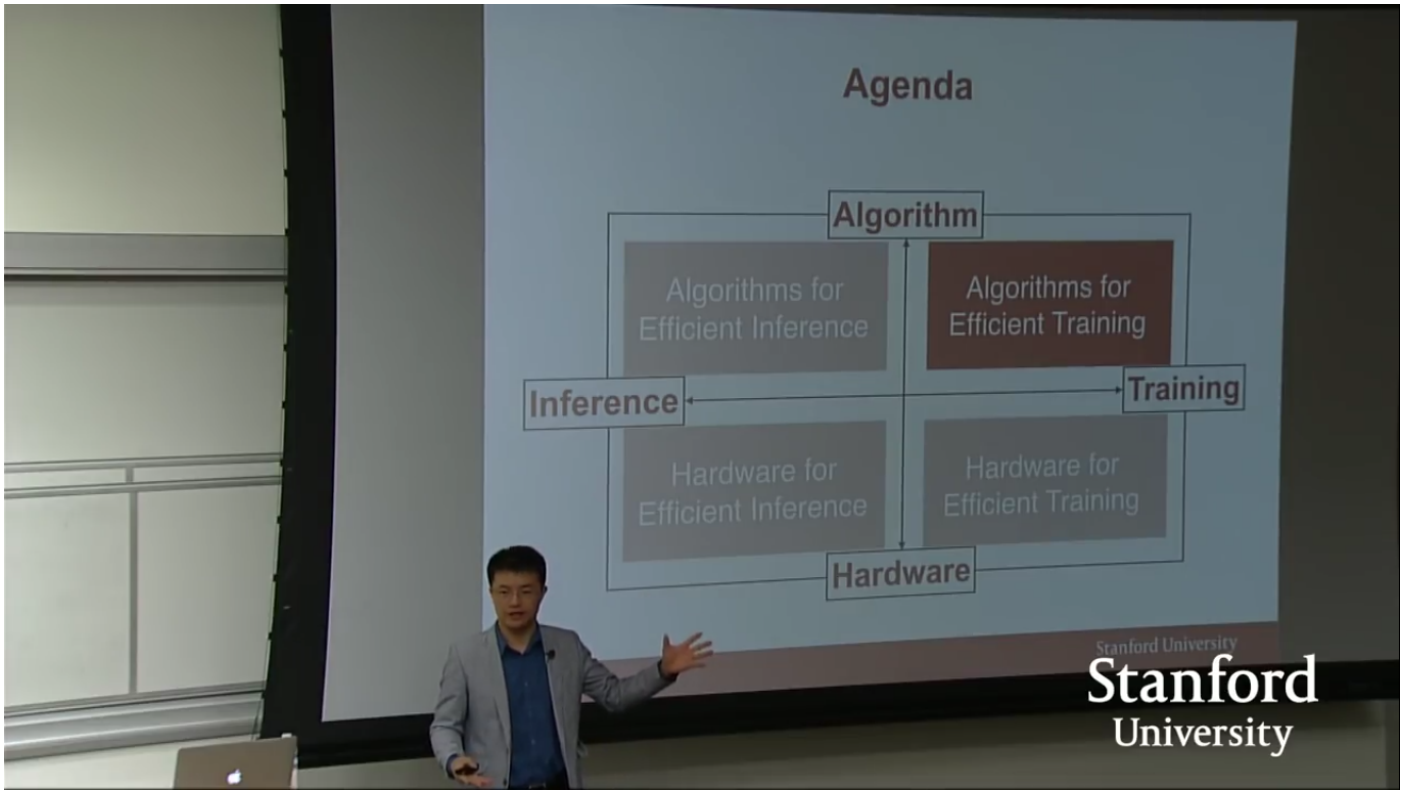


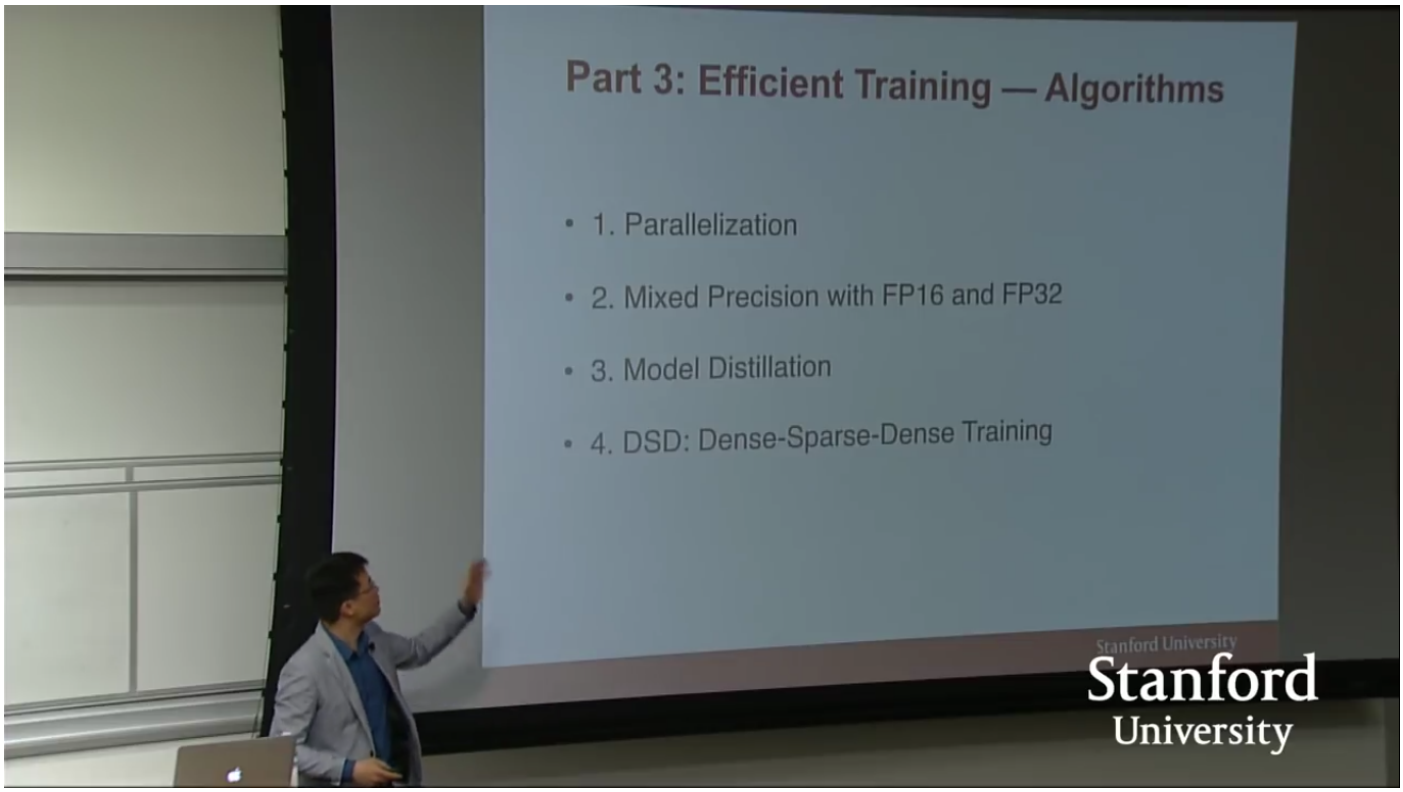
Name	LOC	Layers					Nonlinear function	Weights	TPU Ops / Weight Byte	TPU Batch Size	% Deployed
		FC	Conv	Vector	Pool	Total					
MLP0	0.1k	5				5	ReLU	20M	200	200	61%
MLP1	1k	4				4	ReLU	5M	168	168	
LSTM0	1k	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1.5k	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1k		16			16	ReLU	8M	2888	8	5%
CNN1	1k	4	72		13	89	ReLU	100M	1750	32	

- We can notice that only 5% is being used for CNN operations whereas 29% for LSTM (maybe for speech recognition), 61% for FC (maybe for ad recommendation).



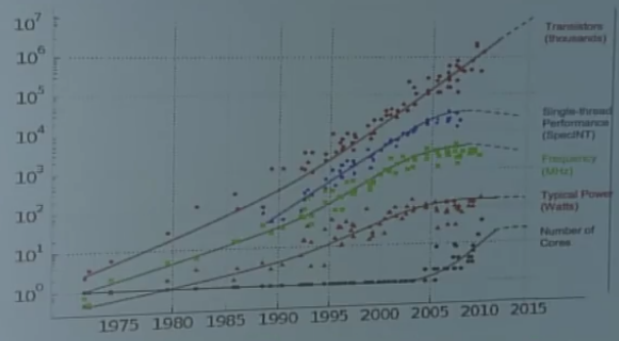
Algorithms for Efficient Training:





Parallelization:

Moore's law made CPUs 300x faster than in 1990
But its over…



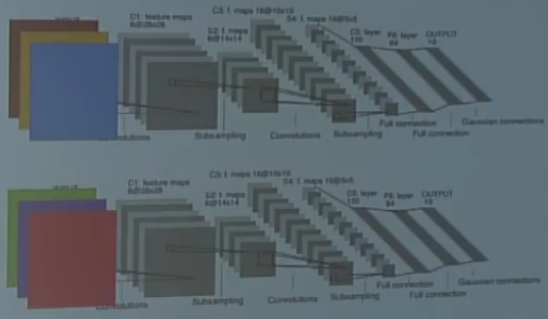
Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

C Moore, Data Processing in ExaScale-ClassComputer Systems, Salishan, April 2011

Stanford University

Stanford
University

Data Parallel – Run multiple inputs in parallel

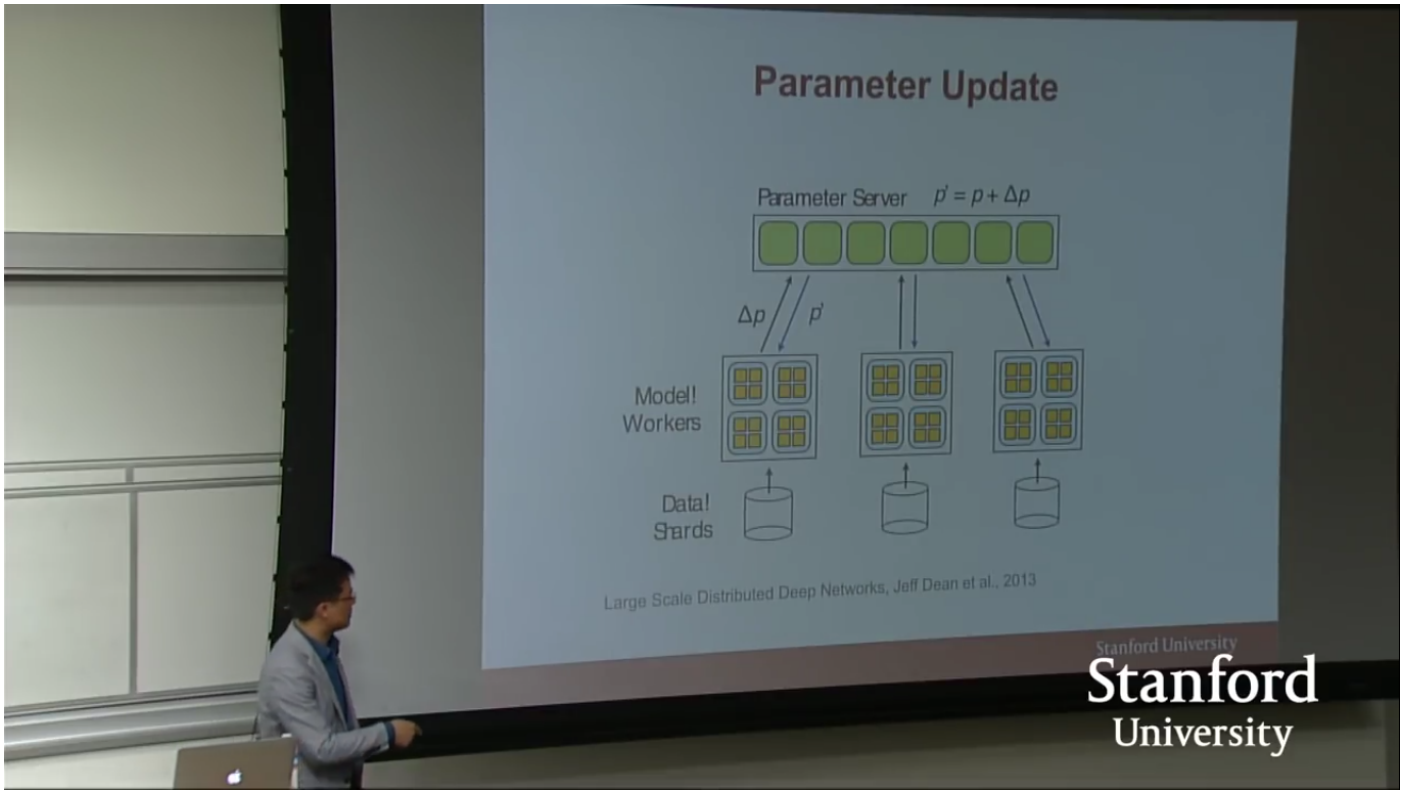


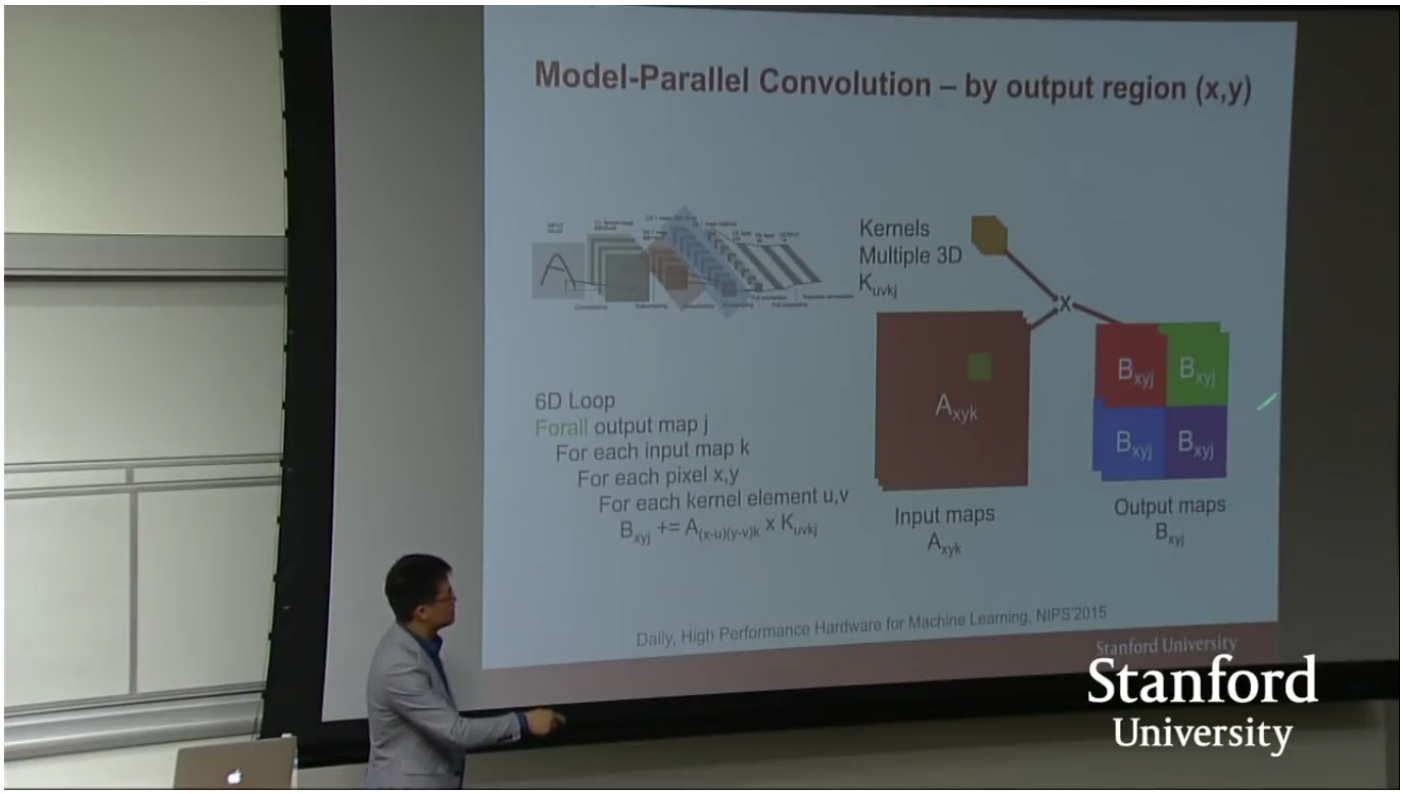
- Doesn't affect latency for one input
- Requires P-fold larger batch size
- For training requires coordinated weight update

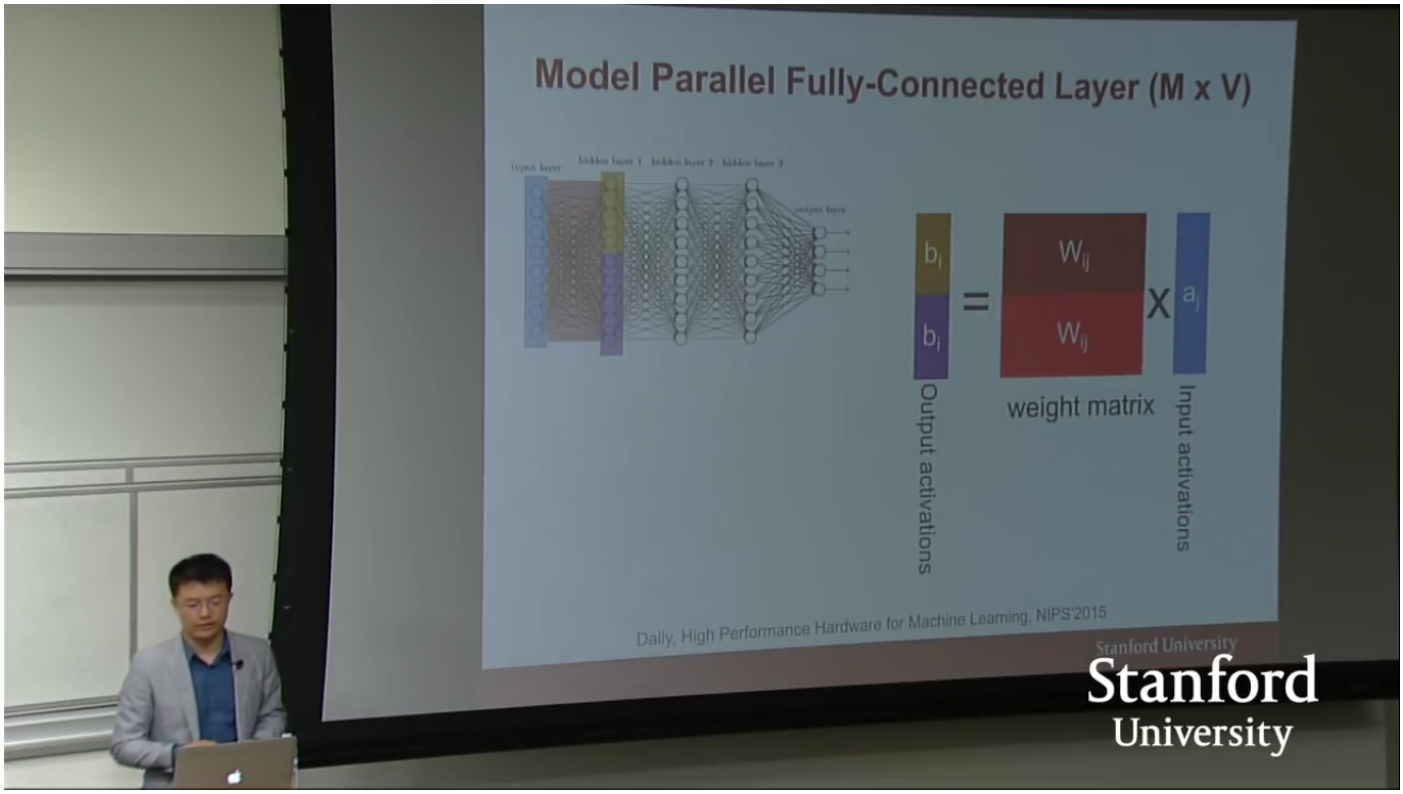
Dally, High Performance Hardware for Machine Learning, NIPS'2015

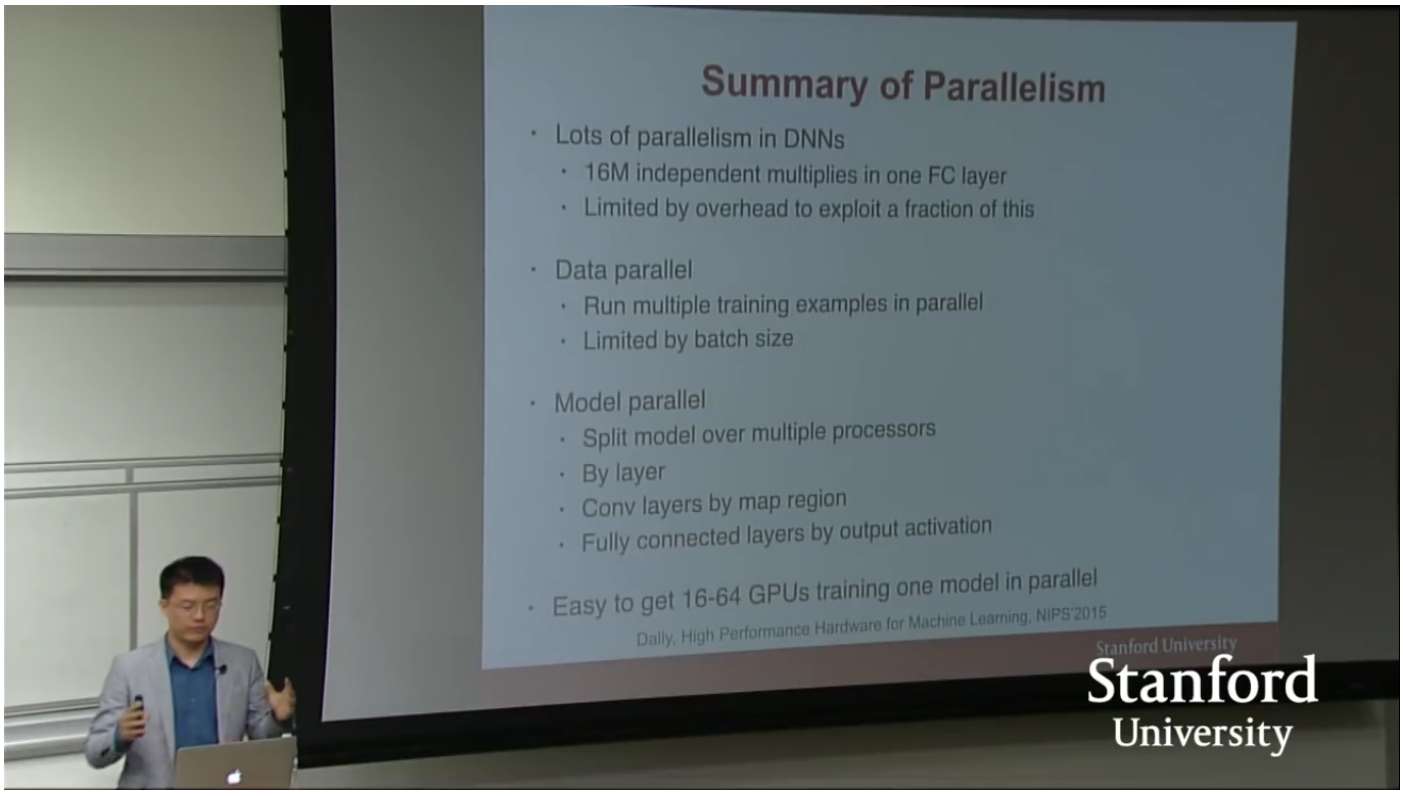
Stanford University

Stanford
University

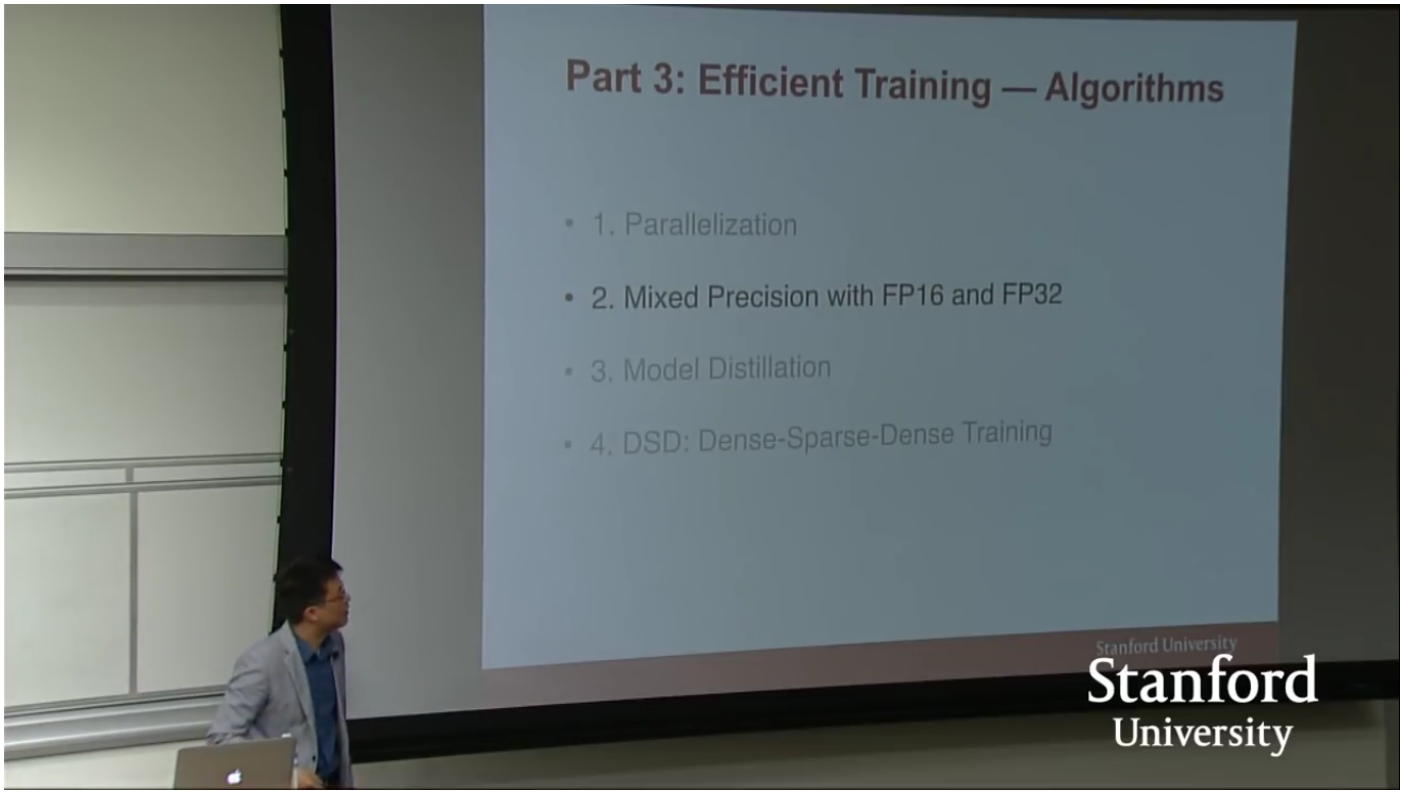

Parameter Update
Parameter Server
$p' = p + \Delta p$
Δp
p'
Model!
Workers
Data!
Shards
Large Scale Distributed Deep Networks, Jeff Dean et al., 2013
Stanford University
Stanford
University

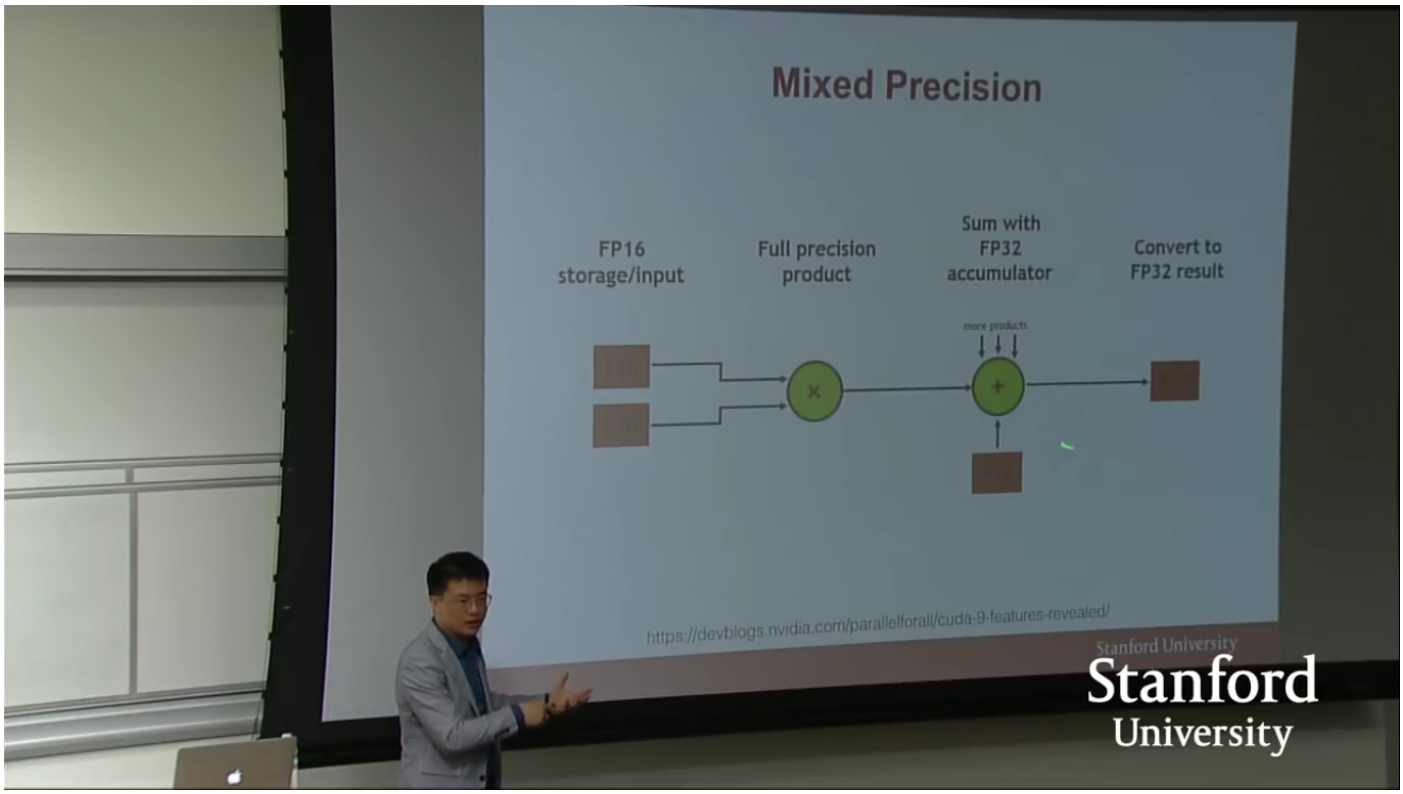

Model-Parallel Convolution – by output region (x,y)
Kernels
Multiple 3D
K_{uvkj}
6D Loop
Forall output map j
For each input map k
For each pixel x,y
For each kernel element u,v
B_{xyj} += $A_{(x-u)(y-v)k}$ x K_{uvkj}
A_{xyk}
Input maps
A_{xyk}
B_{xyj}
Output maps
B_{xyj}
Dally, High Performance Hardware for Machine Learning, NIPS'2015
Stanford University
Stanford
University

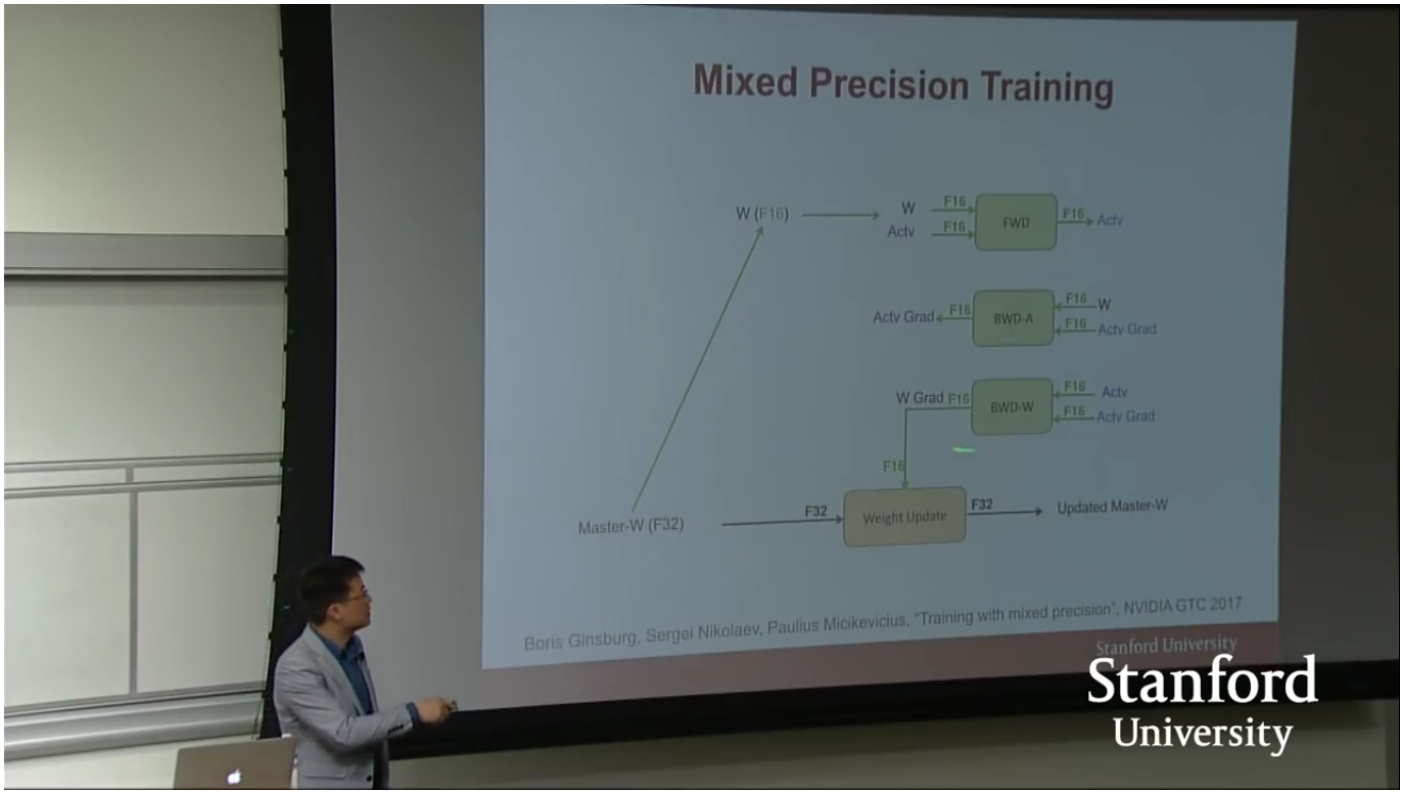




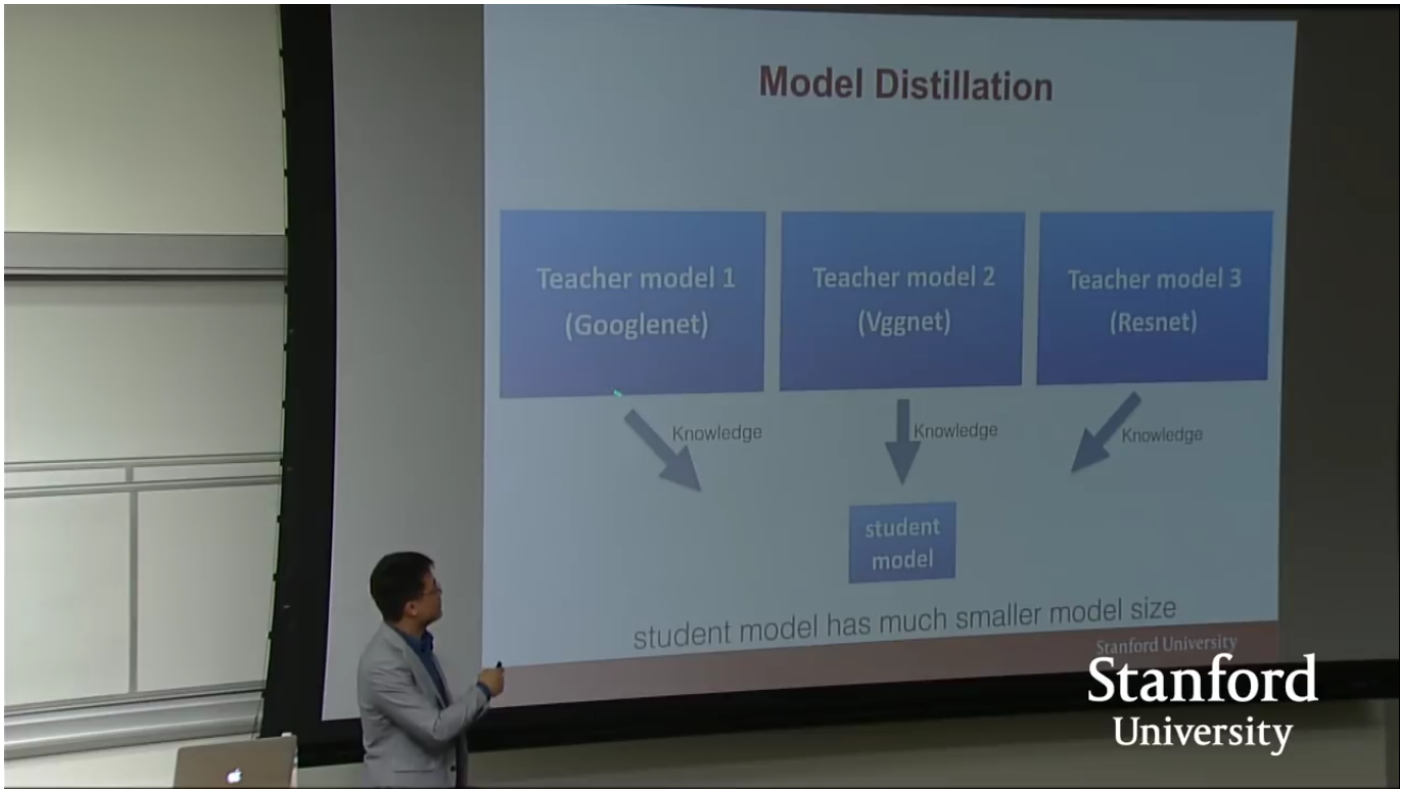
Mixed Precision with FP-16 and FP-32


Part 3: Efficient Training — Algorithms
1. Parallelization
2. Mixed Precision with FP16 and FP32
3. Model Distillation
4. DSD: Dense-Sparse-Dense Training
Stanford University
Stanford
University

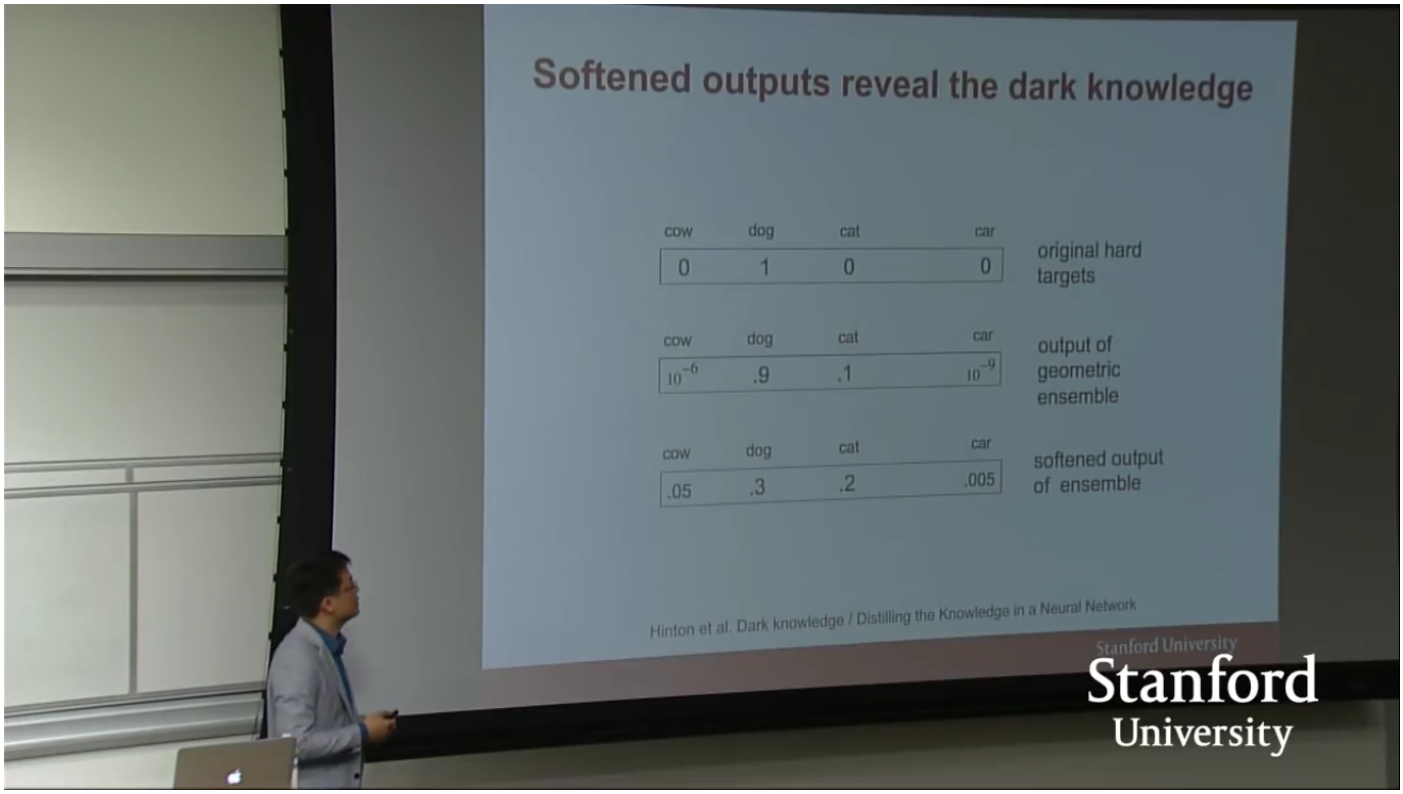

Mixed Precision
FP16 storage/input
Full precision product
Sum with FP32 accumulator
Convert to FP32 result
more products
https://devblogs.nvidia.com/parallelforall/cuda-9-features-revealed/
Stanford University
Stanford
University

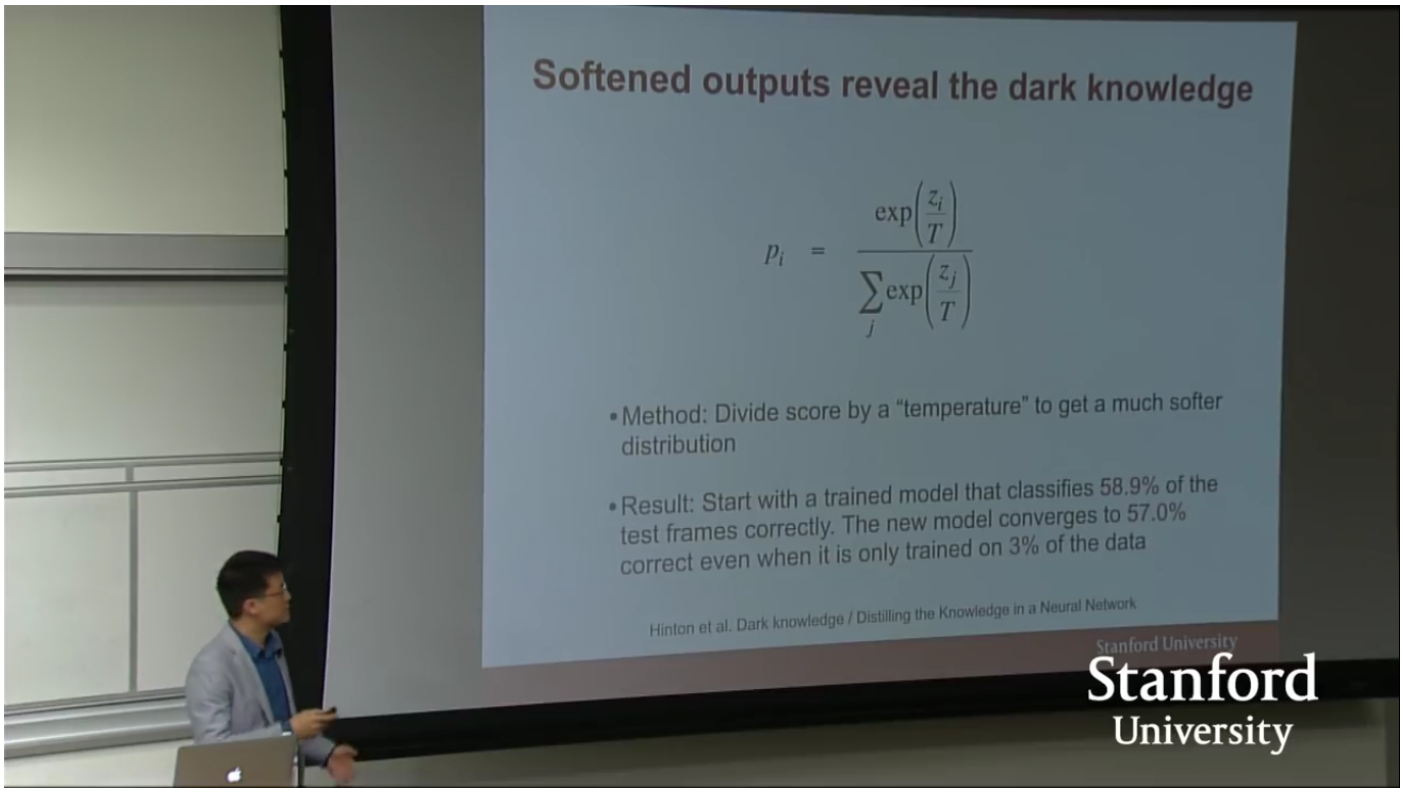


Model Distillation:



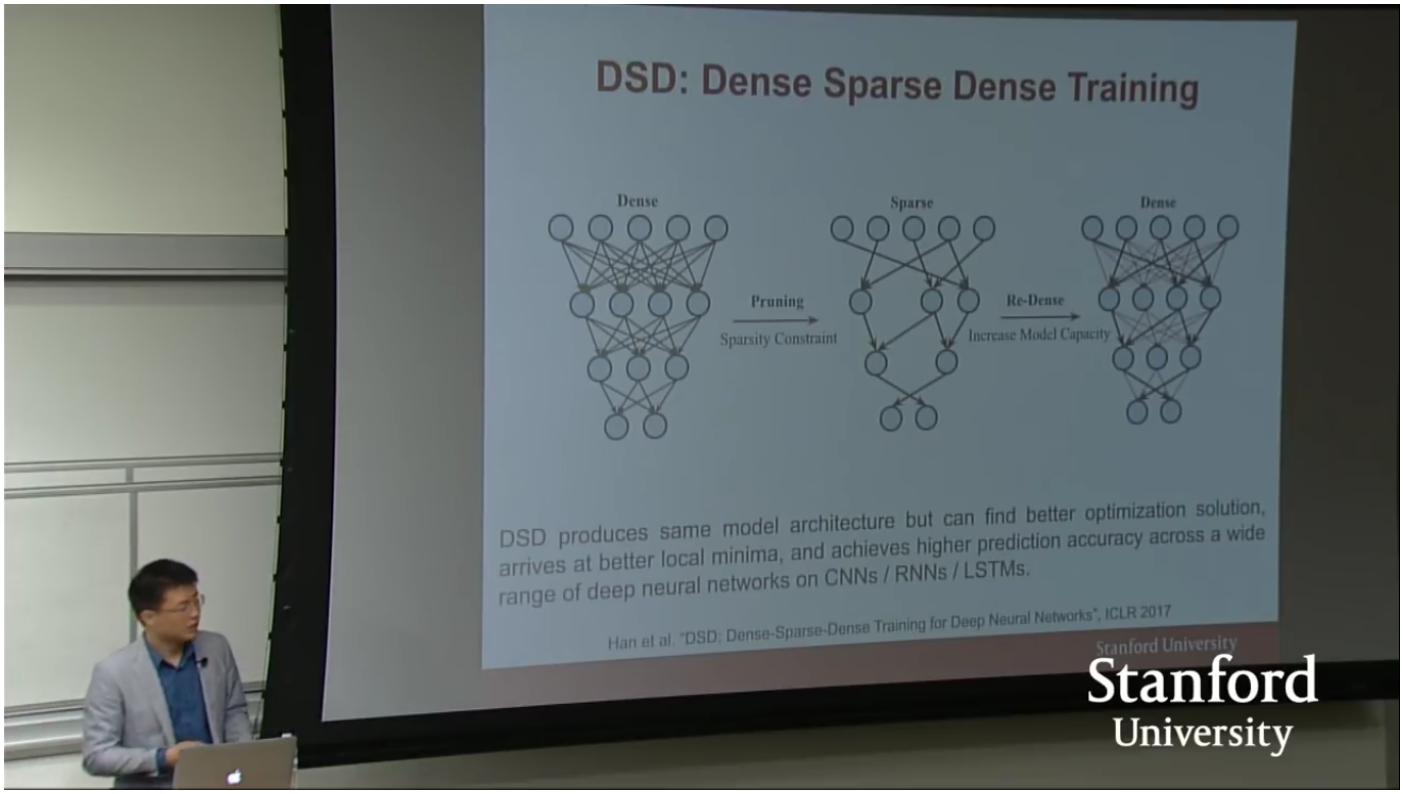
- The idea is to use better performing large parent networks and teach a small student network which is competitive to the parent networks.



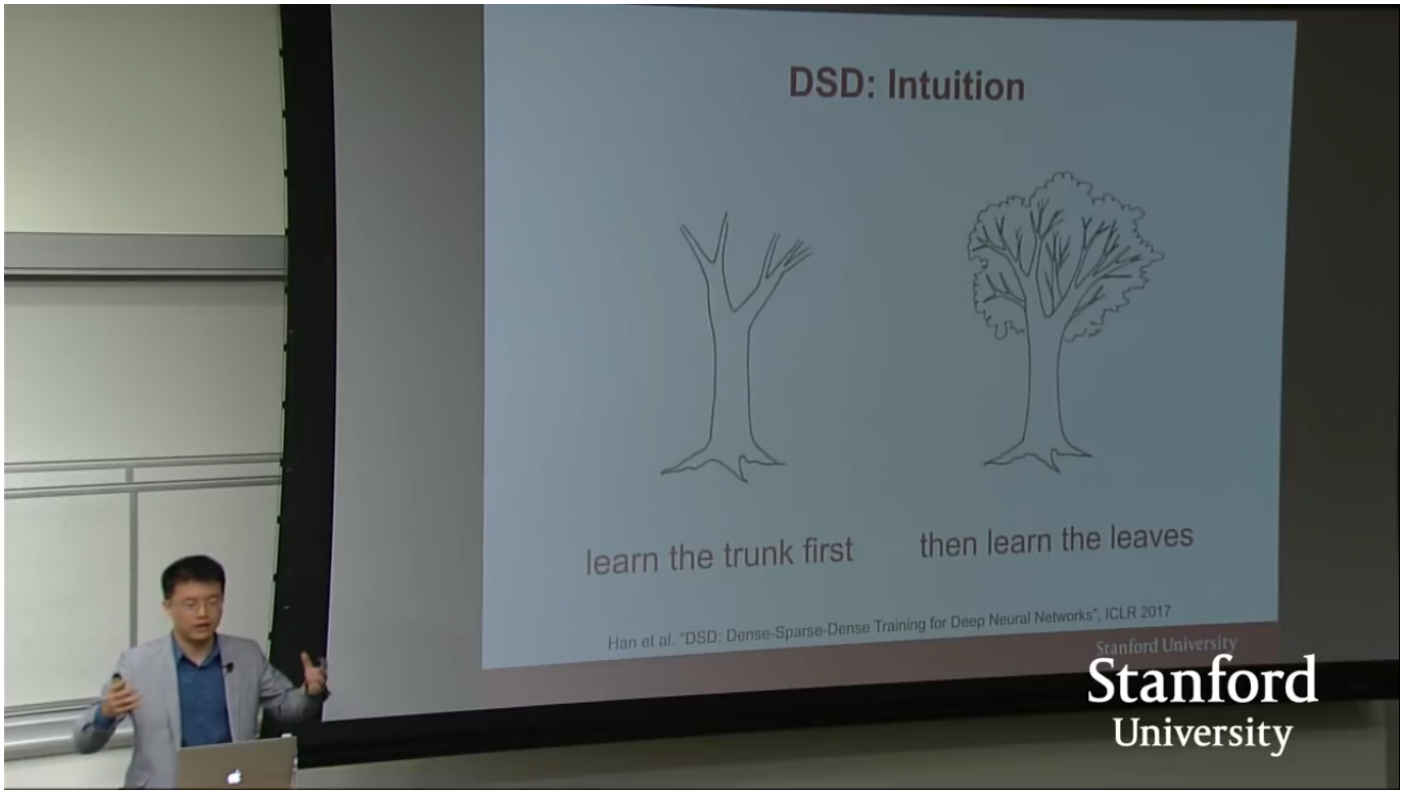


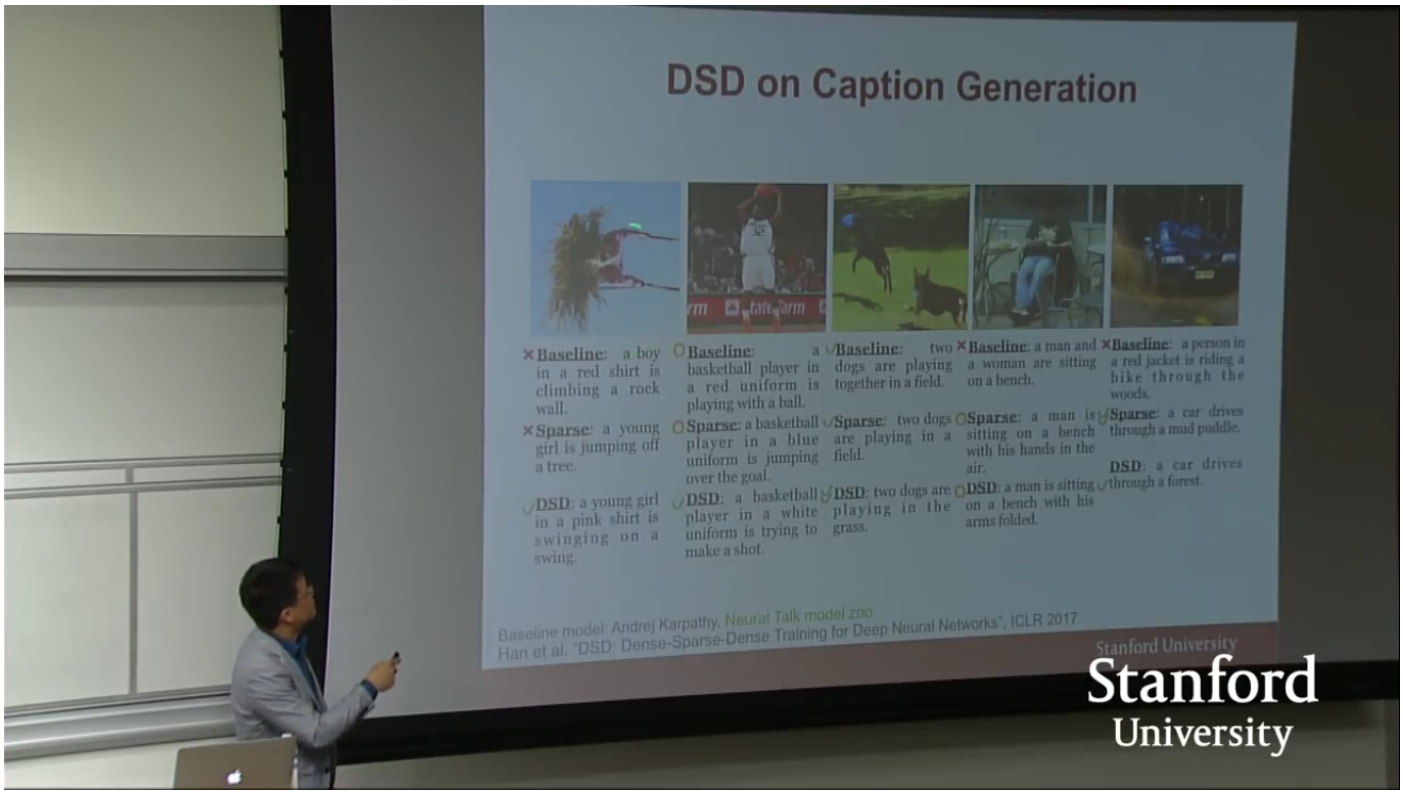
- We control the "softness" by the parameter T (Temperature)

Dense - Sparse - Dense Training

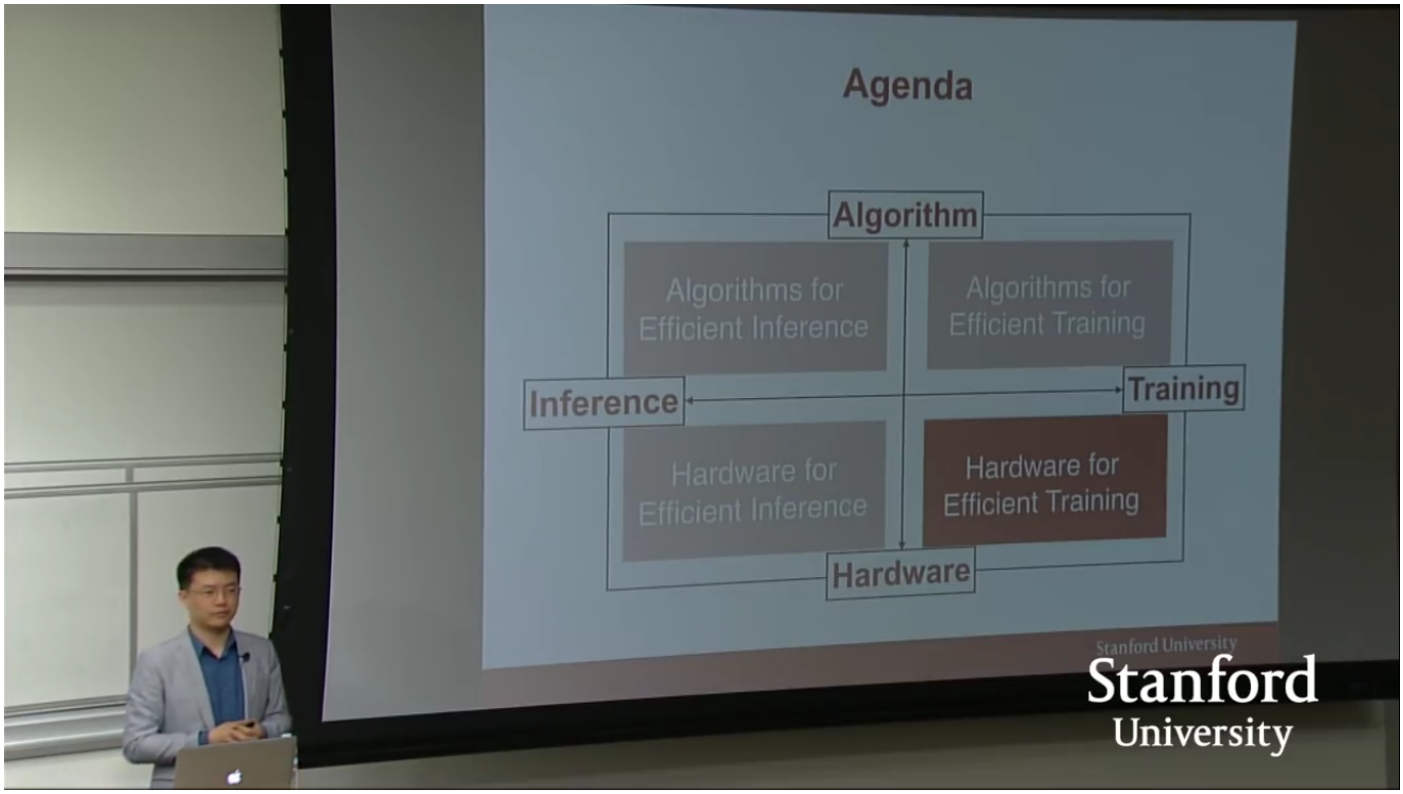


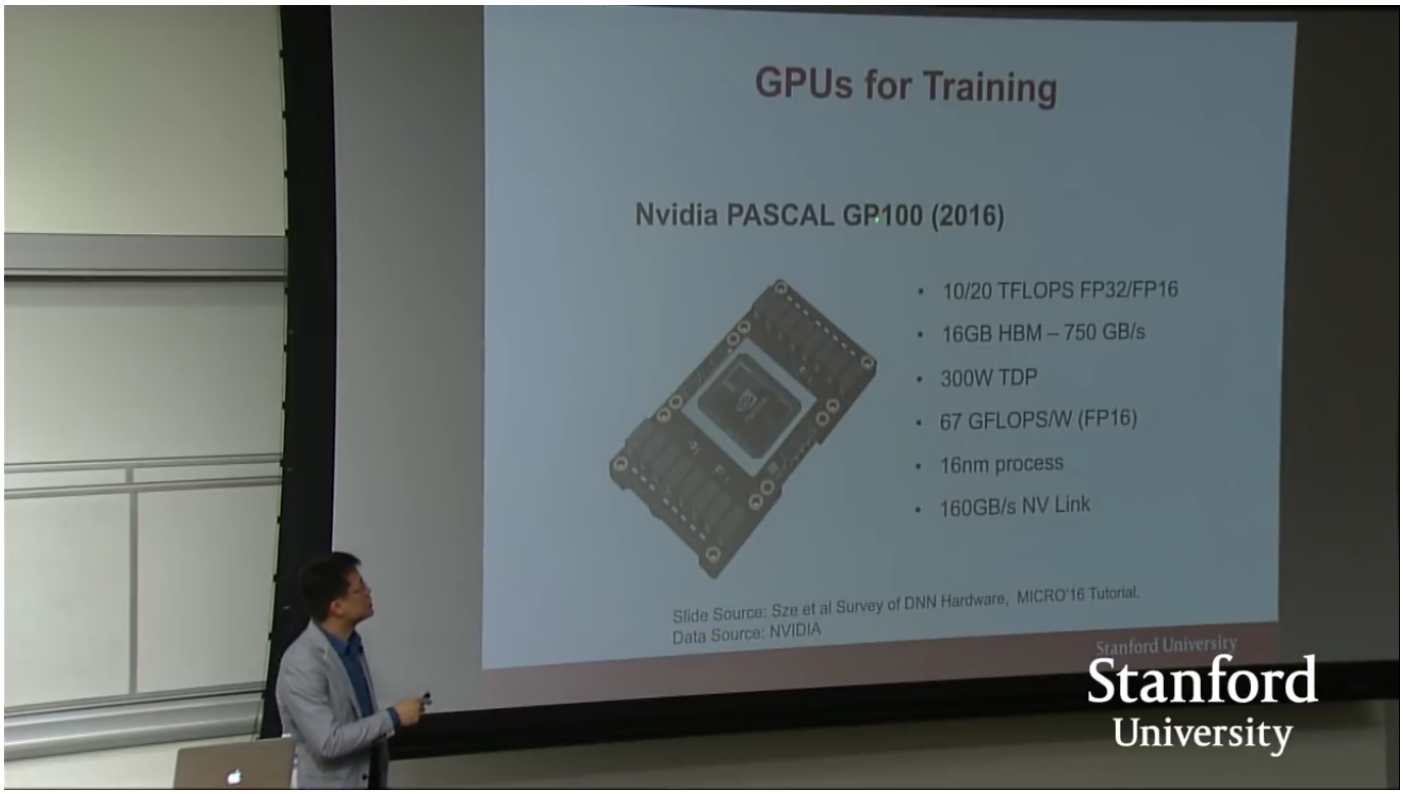
- DSD produces same model architecture but can find better optimization solution, arrives at better local minima, and achieves higher prediction accuracy over a wide range of deep neural networks on CNNs, RNNs and LSTMs.

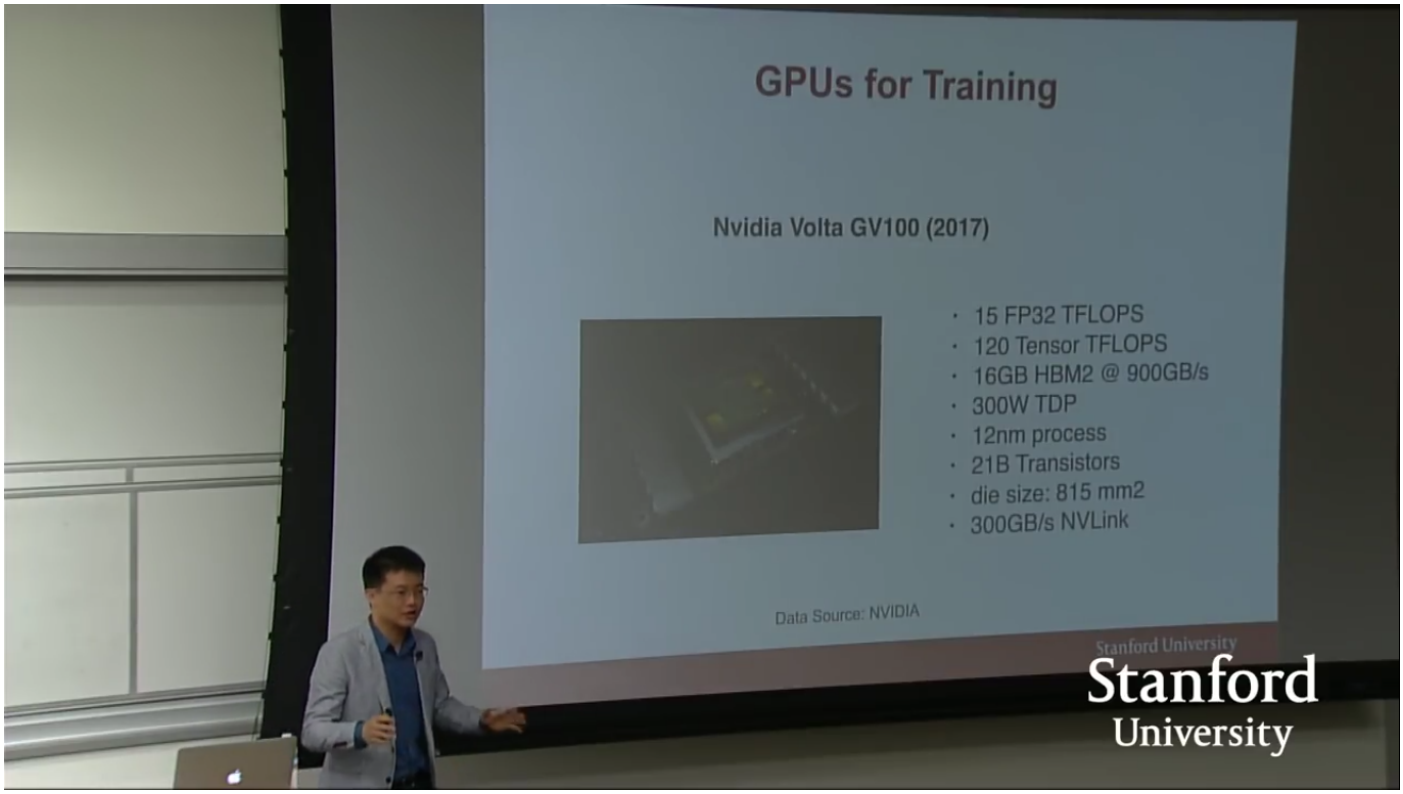


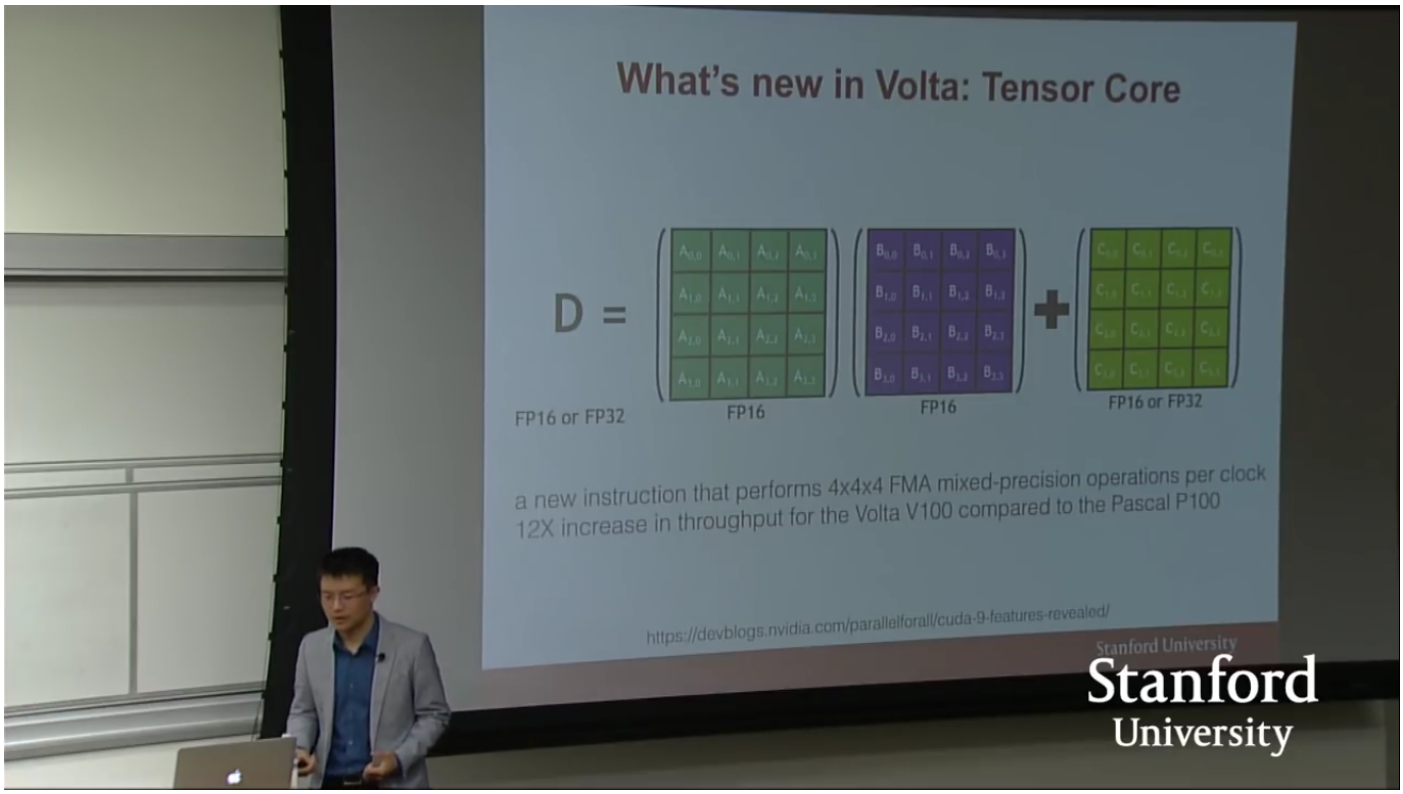


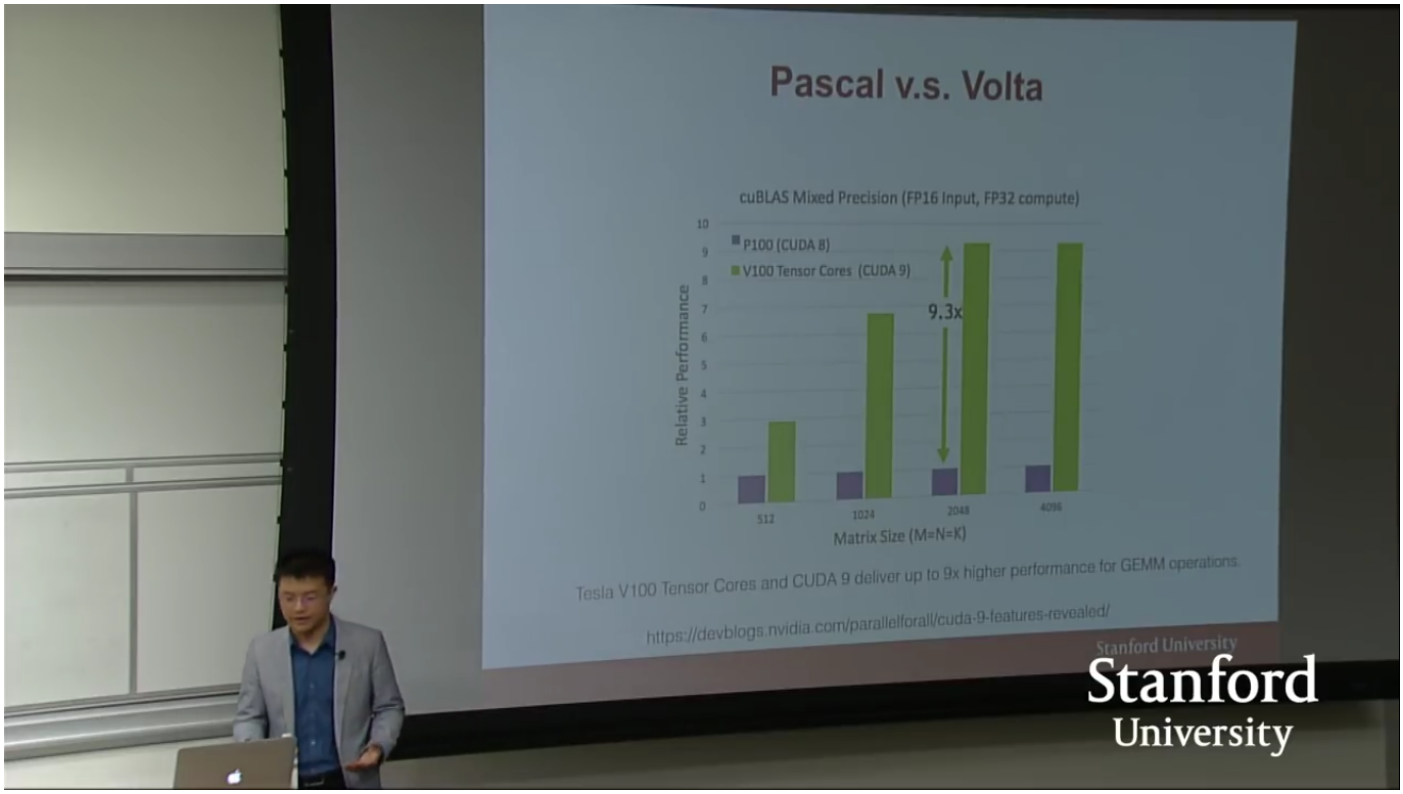
Hardware for Efficient Training:

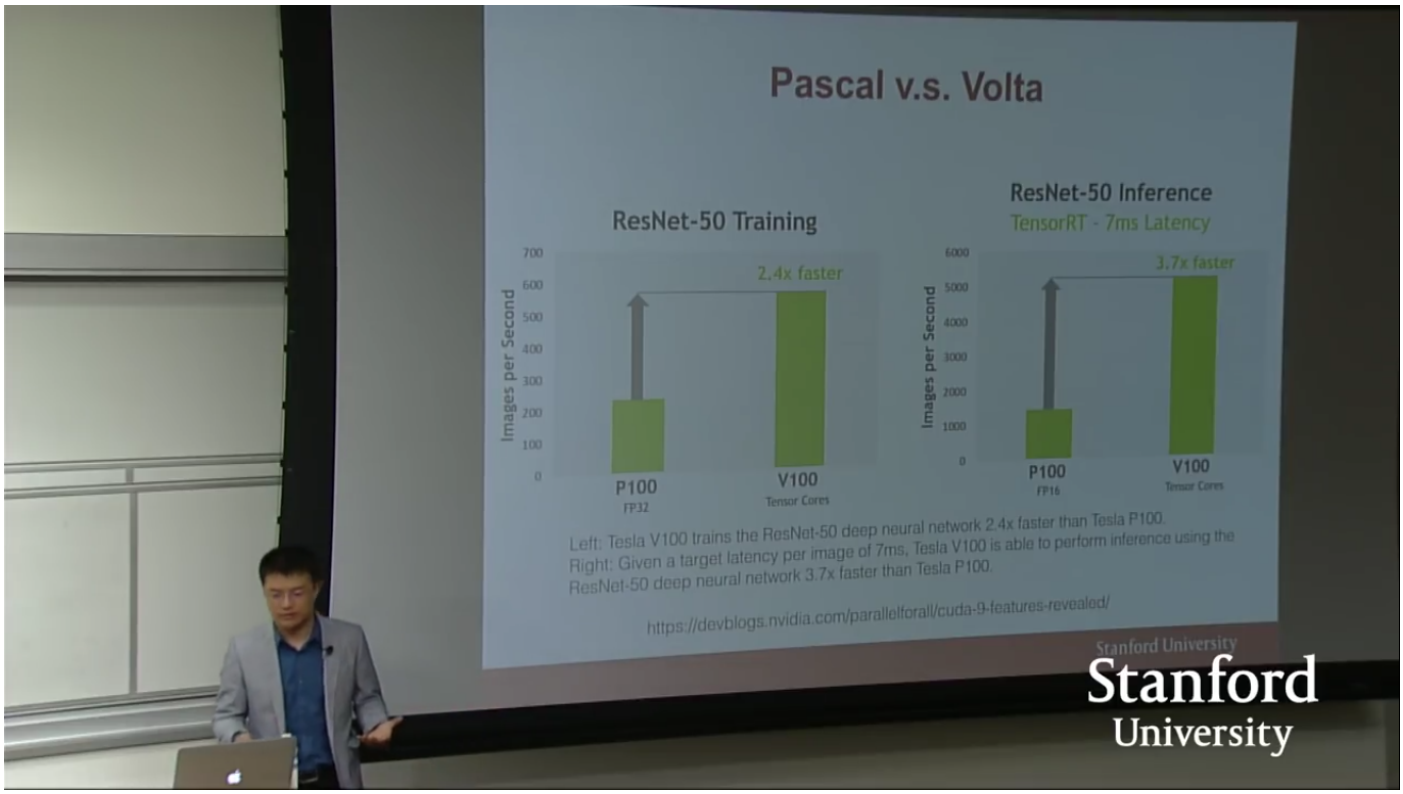

Agenda
Algorithm
Algorithms for Efficient Inference
Algorithms for Efficient Training
Inference
Training
Hardware for Efficient Inference
Hardware for Efficient Training
Hardware
Stanford University
Stanford
University

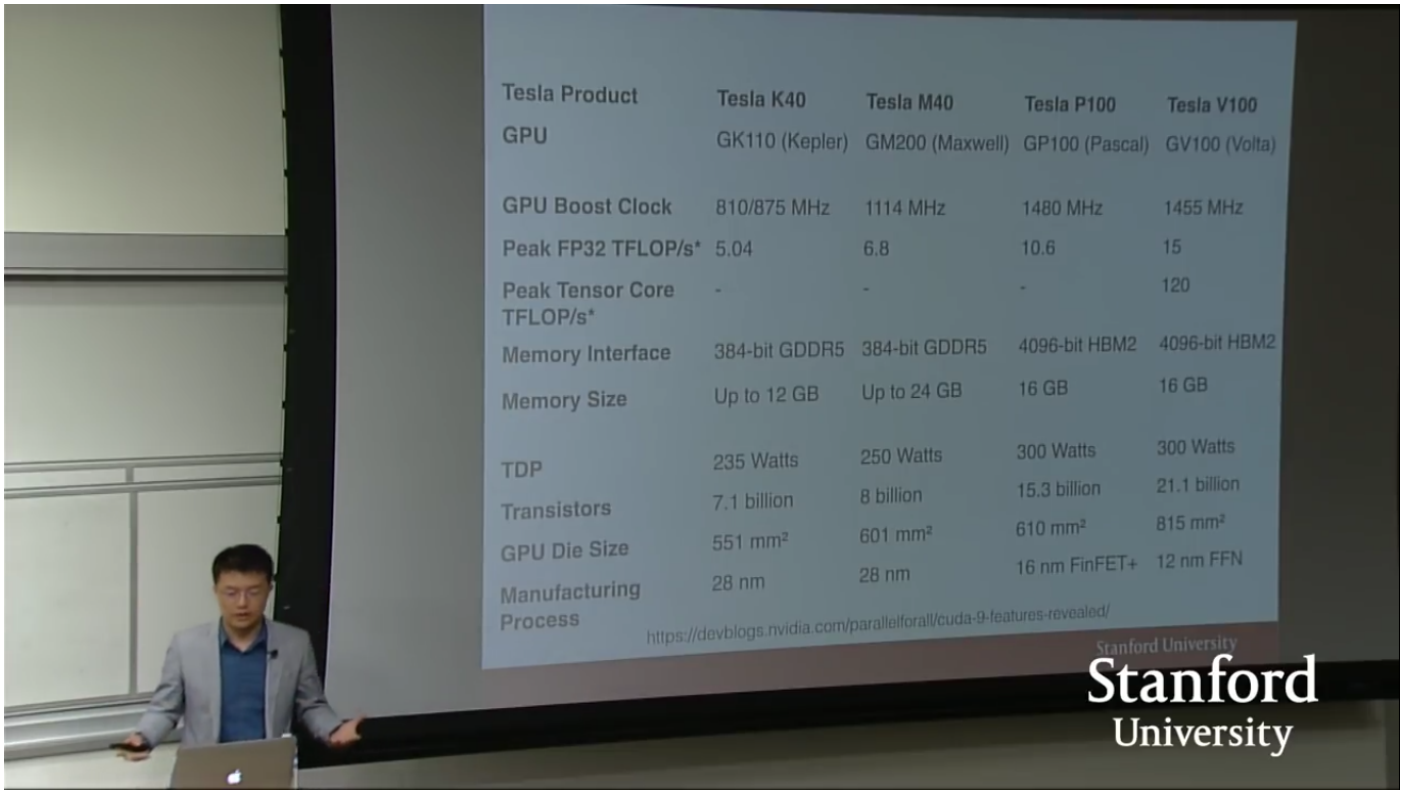

GPUs for Training
Nvidia PASCAL GP100 (2016)
10/20 TFLOPS FP32/FP16
16GB HBM – 750 GB/s
300W TDP
67 GFLOPS/W (FP16)
16nm process
160GB/s NV Link
Slide Source: Sze et al Survey of DNN Hardware, MICRO'16 Tutorial.
Data Source: NVIDIA
Stanford University
Stanford
University

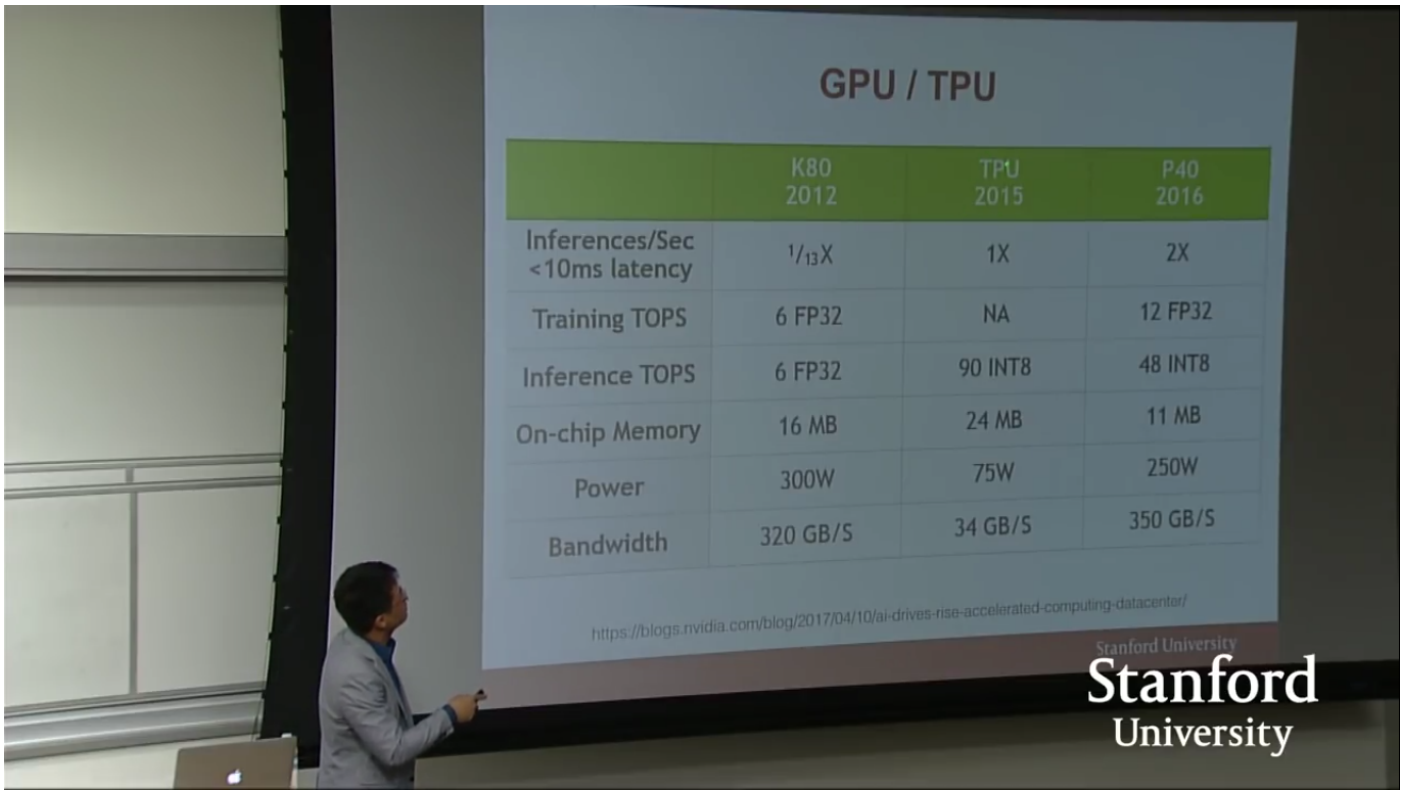

GPUs for Training
Nvidia Volta GV100 (2017)
15 FP32 TFLOPS
120 Tensor TFLOPS
16GB HBM2 @ 900GB/s
300W TDP
12nm process
21B Transistors
die size: 815 mm2
300GB/s NVLink
Data Source: NVIDIA
Stanford University
Stanford
University

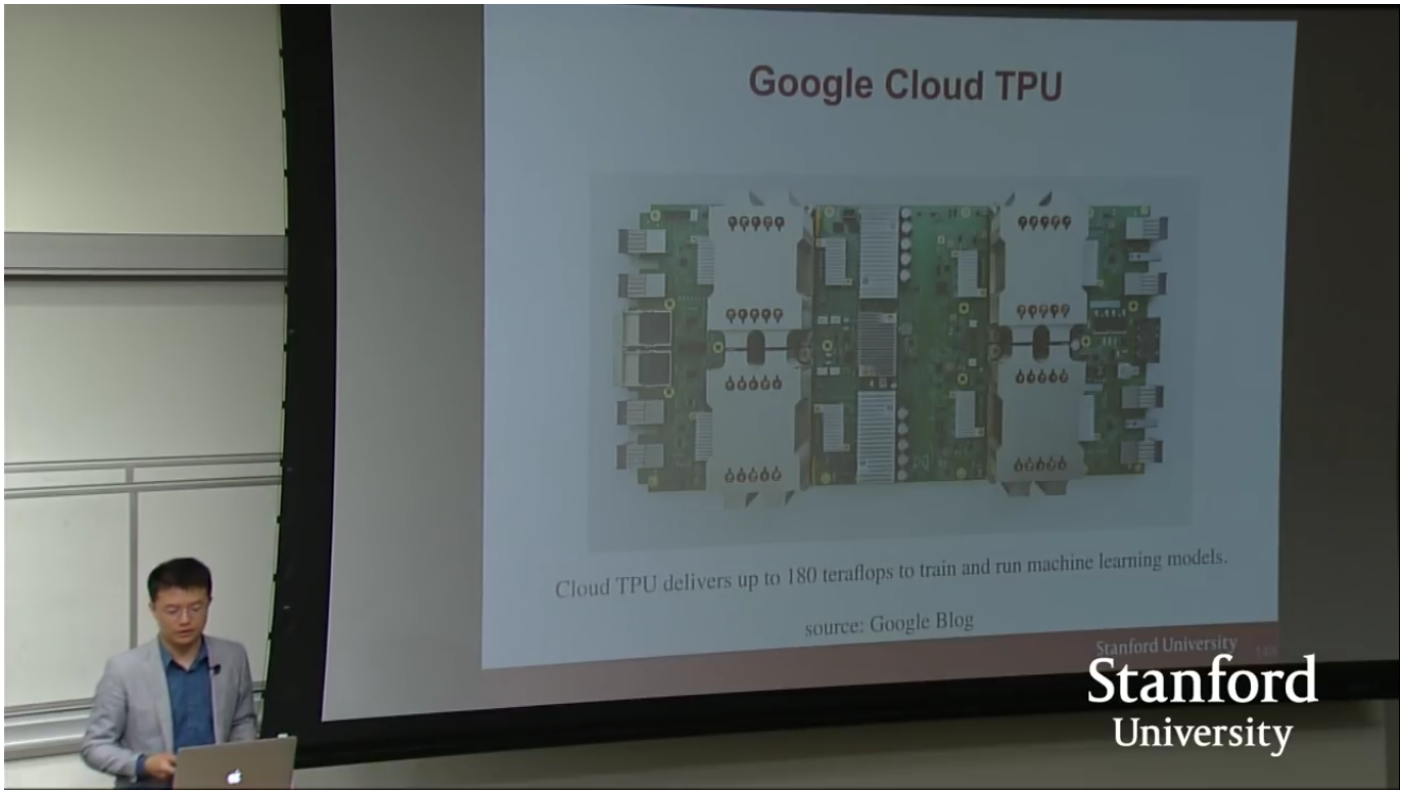

What's new in Volta: Tensor Core
D =
FP16 or FP32
FP16
FP16
FP16 or FP32
a new instruction that performs 4x4x4 FMA mixed-precision operations per clock
12X increase in throughput for the Volta V100 compared to the Pascal P100
https://devblogs.nvidia.com/parallelforall/cuda-9-features-revealed/
Stanford University
Stanford
University

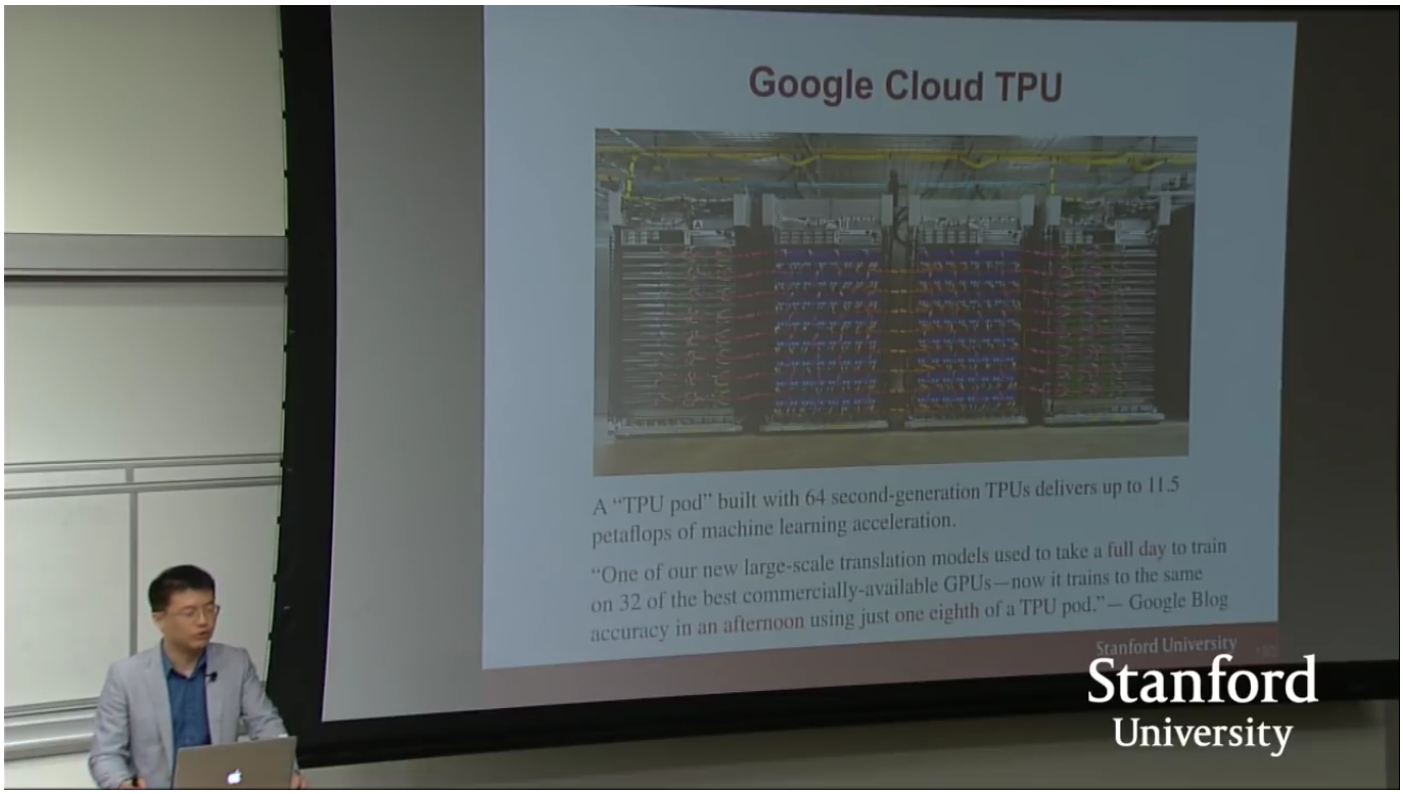

Pascal v.s. Volta
cuBLAS Mixed Precision (FP16 Input, FP32 compute)
P100 (CUDA 8)
V100 Tensor Cores (CUDA 9)
9.3x
Relative Performance
Matrix Size (M=N=K)
512
1024
2048
4096
Tesla V100 Tensor Cores and CUDA 9 deliver up to 9x higher performance for GEMM operations.
https://devblogs.nvidia.com/parallelforall/cuda-9-features-revealed/
Stanford University
Stanford
University

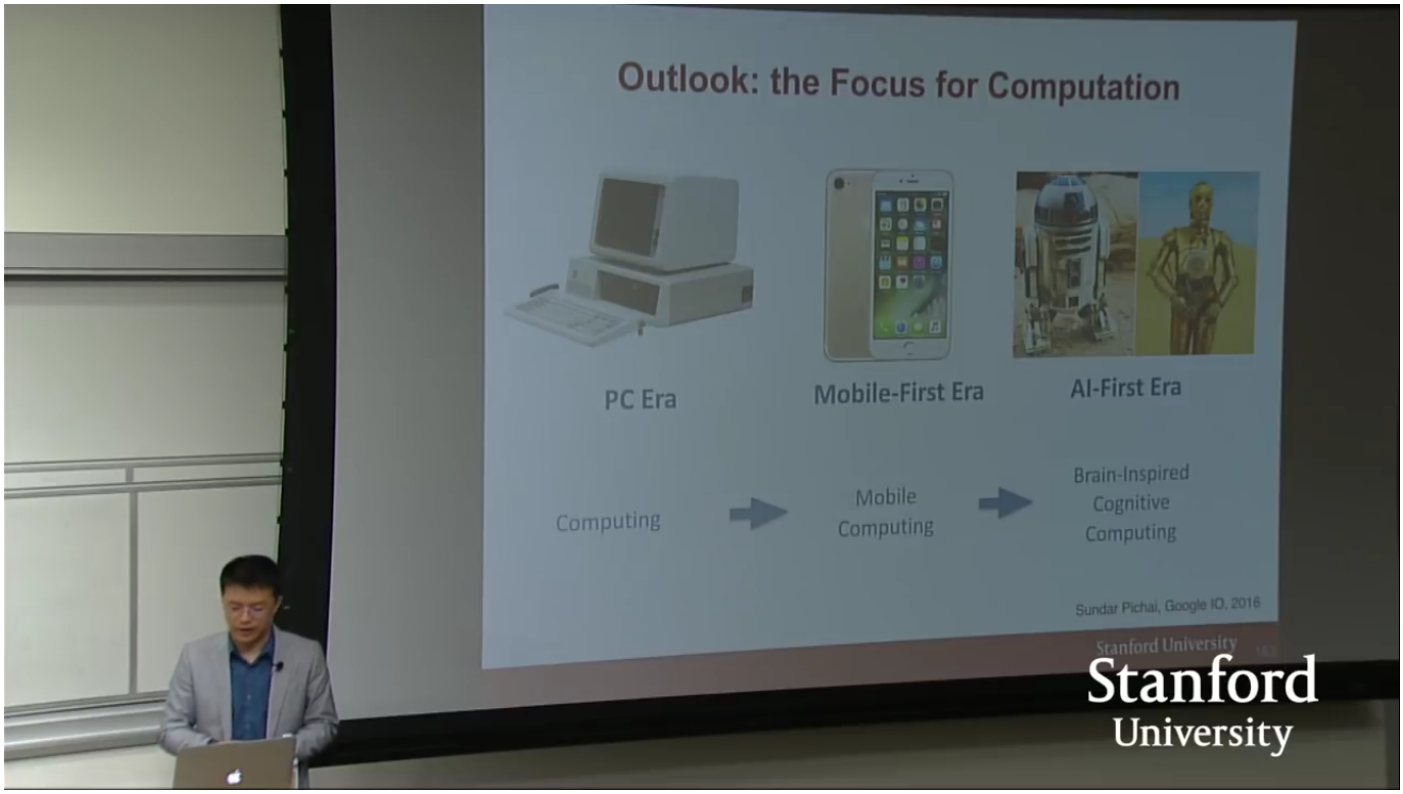

Pascal v.s. Volta
ResNet-50 Training
2.4x faster
Images per Second
P100
FP32
V100
Tensor Cores
ResNet-50 Inference
TensorRT - 7ms Latency
3.7x faster
Images per Second
P100
FP16
V100
Tensor Cores
Left: Tesla V100 trains the ResNet-50 deep neural network 2.4x faster than Tesla P100.
Right: Given a target latency per image of 7ms, Tesla V100 is able to perform inference using the ResNet-50 deep neural network 3.7x faster than Tesla P100.
https://devblogs.nvidia.com/parallelforall/cuda-9-features-revealed/
Stanford University
Stanford
University


Tesla Product | Tesla K40 | Tesla M40 | Tesla P100 | Tesla V100
GPU | GK110 (Kepler) | GM200 (Maxwell) | GP100 (Pascal) | GV100 (Volta)
GPU Boost Clock | 810/875 MHz | 1114 MHz | 1480 MHz | 1455 MHz
Peak FP32 TFLOP/s* | 5.04 | 6.8 | 10.6 | 15
Peak Tensor Core TFLOP/s* | - | - | - | 120
Memory Interface | 384-bit GDDR5 | 384-bit GDDR5 | 4096-bit HBM2 | 4096-bit HBM2
Memory Size | Up to 12 GB | Up to 24 GB | 16 GB | 16 GB
TDP | 235 Watts | 250 Watts | 300 Watts | 300 Watts
Transistors | 7.1 billion | 8 billion | 15.3 billion | 21.1 billion
GPU Die Size | 551 mm² | 601 mm² | 610 mm² | 815 mm²
Manufacturing Process | 28 nm | 28 nm | 16 nm FinFET+ | 12 nm FFN
https://devblogs.nvidia.com/parallelforall/cuda-9-features-revealed/
Stanford University
Stanford
University


GPU / TPU
K80 2012 | TPU 2015 | P40 2016
Inferences/Sec <10ms latency | 1/13X | 1X | 2X
Training TOPS | 6 FP32 | NA | 12 FP32
Inference TOPS | 6 FP32 | 90 INT8 | 48 INT8
On-chip Memory | 16 MB | 24 MB | 11 MB
Power | 300W | 75W | 250W
Bandwidth | 320 GB/S | 34 GB/S | 350 GB/S
https://blogs.nvidia.com/blog/2017/04/10/ai-drives-rise-accelerated-computing-datacenter/
Stanford University
Stanford
University


Google Cloud TPU
Cloud TPU delivers up to 180 teraflops to train and run machine learning models.
source: Google Blog
Stanford University
Stanford
University


Google Cloud TPU
A "TPU pod" built with 64 second-generation TPUs delivers up to 11.5 petaflops of machine learning acceleration.
"One of our new large-scale translation models used to take a full day to train on 32 of the best commercially-available GPUs—now it trains to the same accuracy in an afternoon using just one eighth of a TPU pod."— Google Blog
Stanford University
Stanford
University


Outlook: the Focus for Computation
PC Era
Mobile-First Era
AI-First Era
Computing
Mobile Computing
Brain-Inspired Cognitive Computing
Sundar Pichai, Google IO, 2016
Stanford University
Stanford University