최적 알고리즘과 객체 검출기법/상용화 동향

2019. 3. 26

한양대학교 ERICA 스마트융합공학부
이은주
{eunju19@hanyang.ac.kr}

Key Requirements for Commercial Computer vision Usage

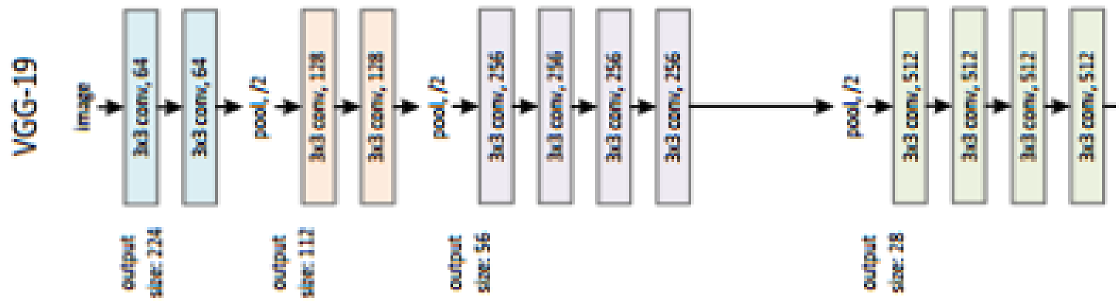
- Data-centers(Clouds)
  - Rarely safety-critical
  - Low power is nice to have
  - Real-time is preferable

- Gadgets-Smartphones, Self-driving cars, Drones, etc
  - Usually safety-critical(except smartphones)
  - Low power is must-have
  - Real-time is required

[MobileNet , JinWon Lee]

Necessities of Model Compression

- Deep networks have recently exhibited state-of-the-art performance in computer vision tasks such as image classification and object detection
- Top-performing systems usually involve very wide and deep networks, with numerous parameters.

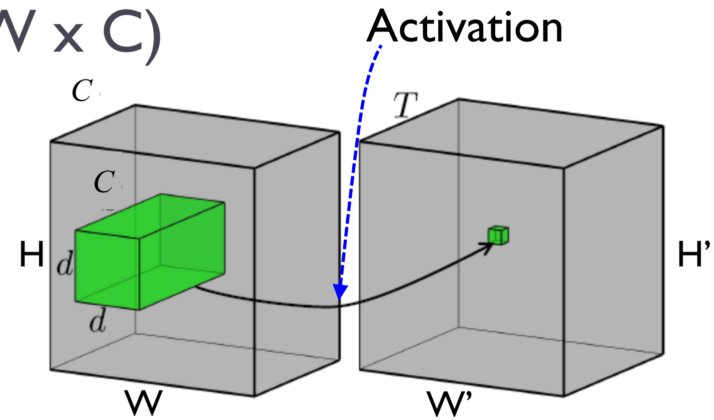


- Major drawback of such wide and deep models
  - Result in very time consuming systems at inference time,
  - Need to perform a huge number of multiplications
  - Having large amounts of parameters makes the models high memory demanding.

Deep Network Complexity

- Input: 3D tensors (map stacks) (H x W x C)
- Output: 3D tensors (H' x W'xT)
- Kernels: 4D tensors (dxdxCxT)
- Feature map size
  - $\mathrm{H}' = \frac{H-d+2z}{\Delta} + 1, \mathrm{W}' = \frac{W-d+2z}{\Delta} + 1$

    $z: zero - padding, \Delta: stride$



(a) Full convolution

[Lebedev,2015 ICLR]

- No. of parameters: dxdxCxT
- No. of operations (connections): HxWxdxdxCxT

[김용덕,Samsung S/W R&D Center]

If)patch size: 7x7, stride: 2
input size: 224(H)x224(W)x3
(RGB channel)
output size: 112 (H')x112(W')x64(T)

$$H' = W' = \frac{224 - 7 + 2 \times 3}{2} + 1 = 112.5$$

$$\#of\ params = 7 \times 7 \times 3 \times 64 = 9{,}408 \approx 9.4K$$

$$\#of\ op = 224 \times 224 \times 7 \times 7 \times 3 \times 64$$
$$= 472{,}055{,}808 \approx 472M$$

[C. Szegedy, Going deeper with convolutions, CVPR2015]

Deep Network Complexity

- Convolution Vs. Fully connected layer
  - No. of Param.: Convolution << Fully connected
  - No. of Ops.: Convolution >> Fully connected

[김용덕,Samsung S/W R&D Center]

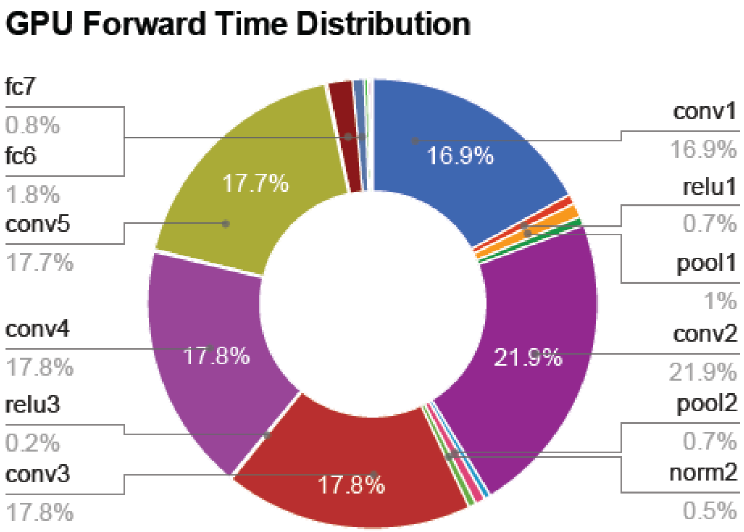
Model	Param. (M)	Conv (%)	FC (%)	Ops. (M)	Conv. (%)	FC (%)	Accuracy (Top-5, %)
AlexNet	61	3.8	96.2	725	91.9	8.1	19.97
VGG-F	99	2.2	97.8	726	87.4	12.6	44.49
VGG-M	103	6.3	93.7	1678	94.3	5.7	40.19
VGG-S	103	6.3	93.7	2640	96.3	3.7	39.38
VGG-16	138	10.6	89.4	15484	99.2	0.8	31.78
VGG-19	144	13.9	86.1	19647	99.4	0.6	31.54
GoogLeNet	<u>6.9</u>	85.1	<u>14.9</u>	<u>1566</u>	99.9	<u>0.1</u>	31.07

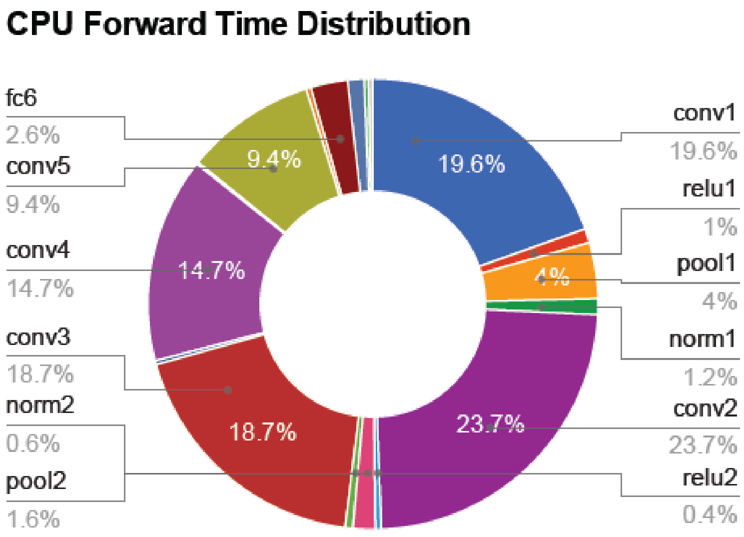
Caffe model zoo
AlexNet 8 Convolution layer 3 FC layer 6200만 weight
GoogLeNet 22 Convolution layer Transfer learning을 위해 1 FC layer 500만 weight (inception)

Deep Network Complexity

- Conv is at the heart of Deep Learning
  - Both platforms spend most computation time on dense convolution and fully connected layers
  - GPU Vs. CPU time distribution

[Jia2014 Thesis, UC Berkeley]





[batch size of 256]

We need Model Compression!

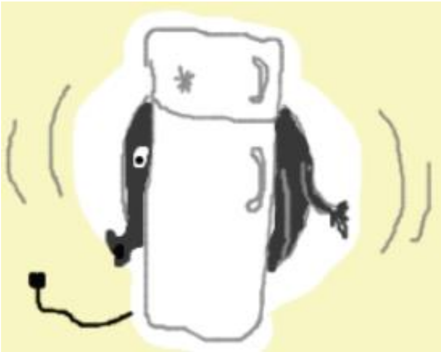
- Expanding the capability of deep learning to a wide range of applications

Wide and deep top-performing networks are not well suited for applications with memory or time limitations.

- Desirable Properties
  - Sufficiently high accuracy
  - Low computational complexity
  - Low energy usage
  - Small model size

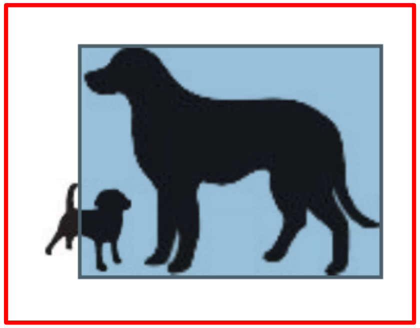
Challenges in an Embedded World

- DNNs are expensive (Inception: 3 billion flops)
  - Batteries don't last that long ~4 hrs (optimistic est.)
  - We need better libraries for numerical optimization.
- DNN models are often very large
  - We are getting better
    - AlexNet: 240MB; Inception: 6MB
  - But things are still wildly big for embedded
    - Storage, bandwidth and memory limits



Speed

Power



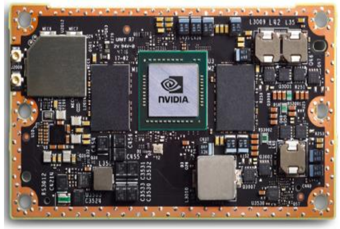
Size

[Jia2014 Thesis, UC Berkeley]

Challenges in an Embedded World

- Two approaches
  - 1) Improving HW system
    - Jetson TK1 Embedded Development Kit
    - Jetson TX1, TX2

<NVIDIA Jetson TK1>
AP: Tegra K1
GPU: NVIDIA Kepler
$192



<NVIDIA Jetson TX2>
AP: Parker Series SoC
GPU: Pascal (256 cores)
$599

AlexNet (only consider forward pass in Caffe)

GPU	GFLOPS	Batch size	Time (ms)
Titan X	6144	128	121*
Kepler	326	1	89
		64	1808

* https://github.com/soumith/convnet-benchmarks

- Still burden to operate deep networks

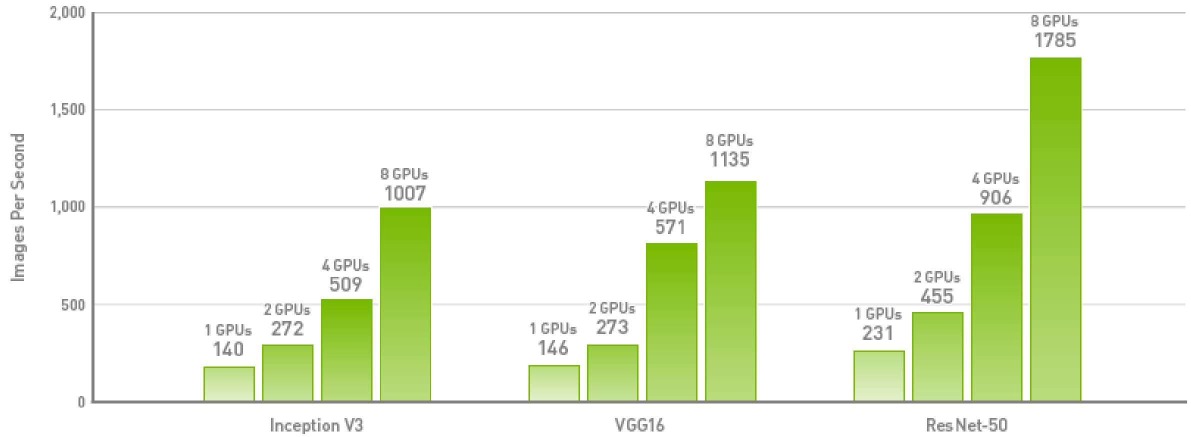
Challenges in an Embedded World

- Two approaches
  - 2) Making Tiny system (Software)
    - Caffe2 framework (NVIDIA & FaceBook)

Caffe2

A New Lightweight, Modular, and Scalable Deep Learning Framework

Caffe2 Trains Up to 7X Faster on a Single DGX-1



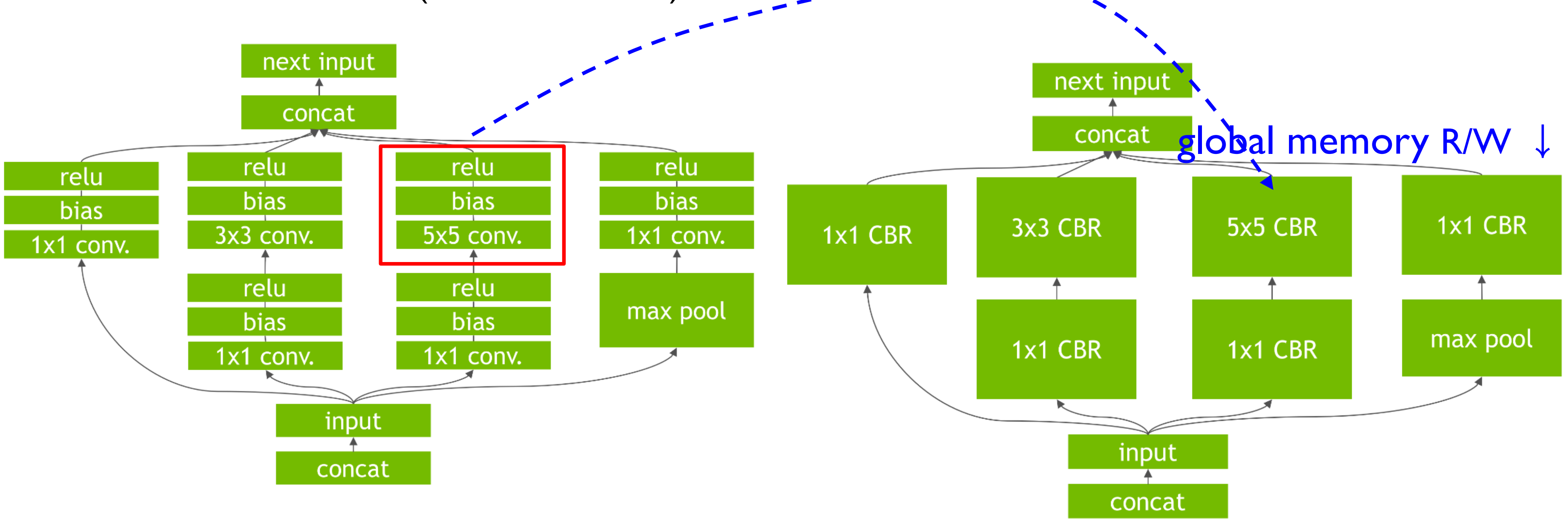
Caffe2 multi-GPU performance (images/sec) on NVIDIA DGX-1 | Networks: Inception v3, VGG16, ResNet-50 | Batch size: 64 | Number of GPUs: 1, 2, 4, 8

    - Caffe2 takes full advantage of the latest NVIDIA Deep Learning SDK libraries, cuDNN , cuBLAS and NCCL
    - Providing high-performance, multi-GPU acceleration for desktop, data centers, and embedded edge devices

[https://developer.nvidia.com/caffe2]

Challenges in an Embedded World

- Two approaches
  - 2) Making Tiny system (Software)
    - Tensor RT (NVIDIA, Caffe)

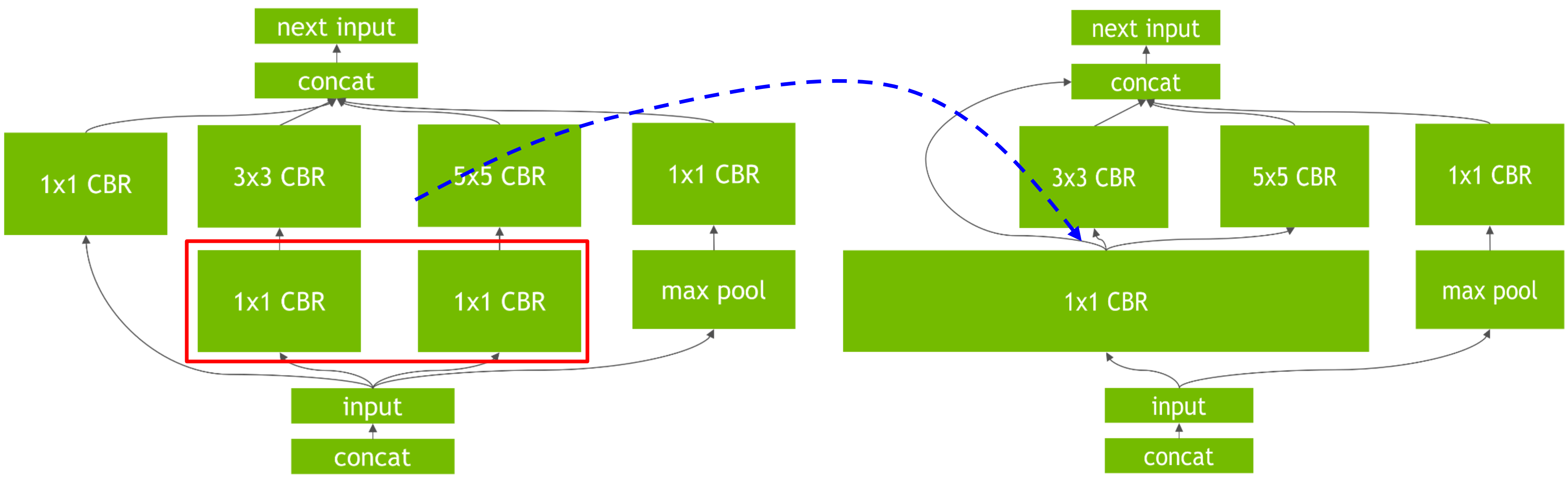


Vertical layer fusion: **memory ↑** **calculation speed ↑**

- convolution, bias, and ReLU layers are fused to form a single layer
- improves the efficiency of running Tensor RT-optimized networks on the GPU.

Challenges in an Embedded World

- Two approaches
  - 2) Making Tiny system (Software)
    - Tensor RT (NVIDIA, Caffe)

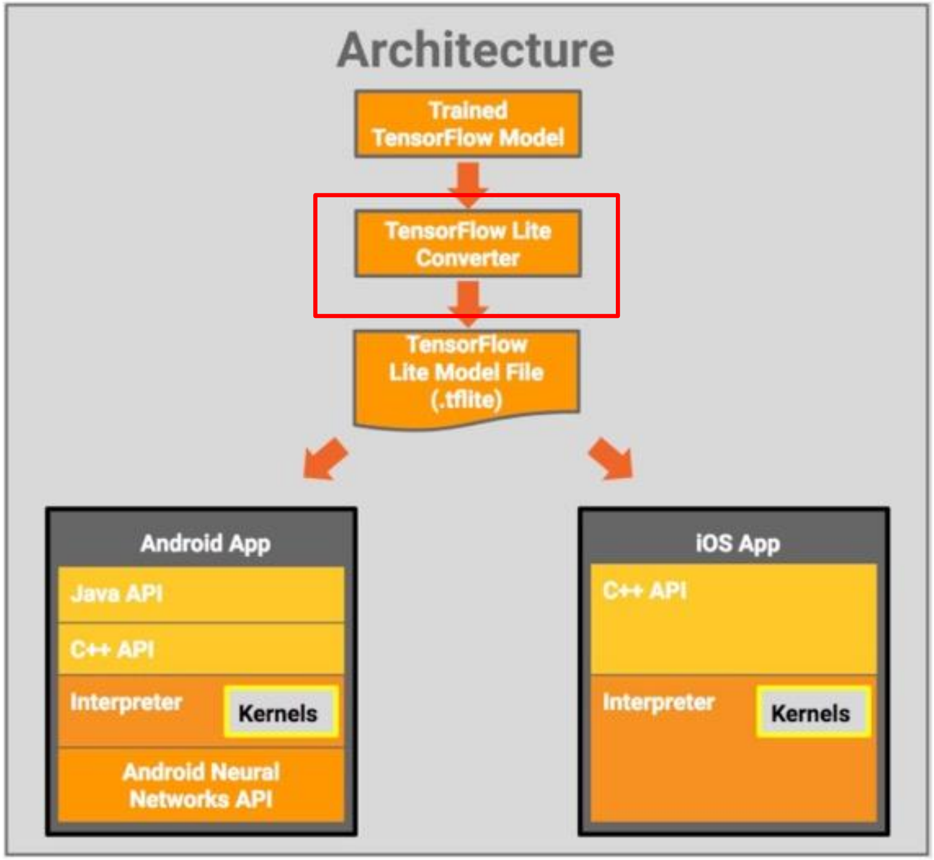


Horizontal layer fusion:
- combining layers that take the same source tensor and apply the same operations with similar parameters
- resulting in a single larger layer for higher computational efficiency

Challenges in an Embedded World

- Two approaches
  - 2) Making Tiny system (Software)
    - TensorFlow Lite (MobileNet)

1. Pre-training Model in Cloud
2. Local fine-training with private data
3. Low complex inference in mobile!



Brief Review of Model Compression Techniques

Summarization of different approaches of different approaches for network compression

Theme Name	Description	Applications
Parameter pruning and weight sharing	• Reducing redundant parameters which are not sensitive to the performance	Convolutional layer and fully connected layer
Low-rank factorization	• Using matrix/tensor decomposition to estimate the informative parameters	Convolutional layer and fully connected layer
Transferred/compact convolutional filters	• Designing special structural convolutional filters to save parameter	Only for convolutional layer
Teacher-student model	• Training a compact neural network with distilled knowledge of a large model	Convolutional layer and fully connected layer

[Yu Cheng, IEEE Sig. Process. Mag. 2017]

Parameter pruning and sharing

- 1) Pruning
  - Explore the redundancy in the model parameters and try to remove the redundant and uncritical ones
  - Reduce network complexity and to address the over-fitting issue
  - Han et al. proposed to reduce the total number of parameters and operations in the entire network. [S. Han et al., NIPS 2015]

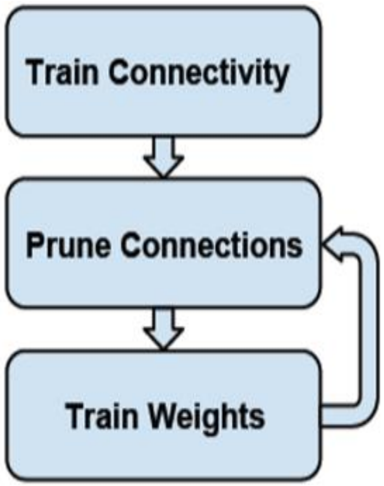
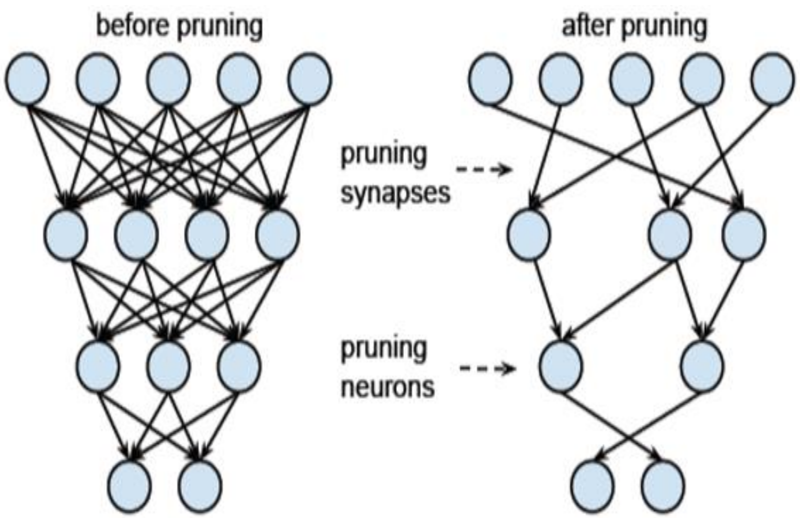


Figure 2: Three-Step Training Pipeline

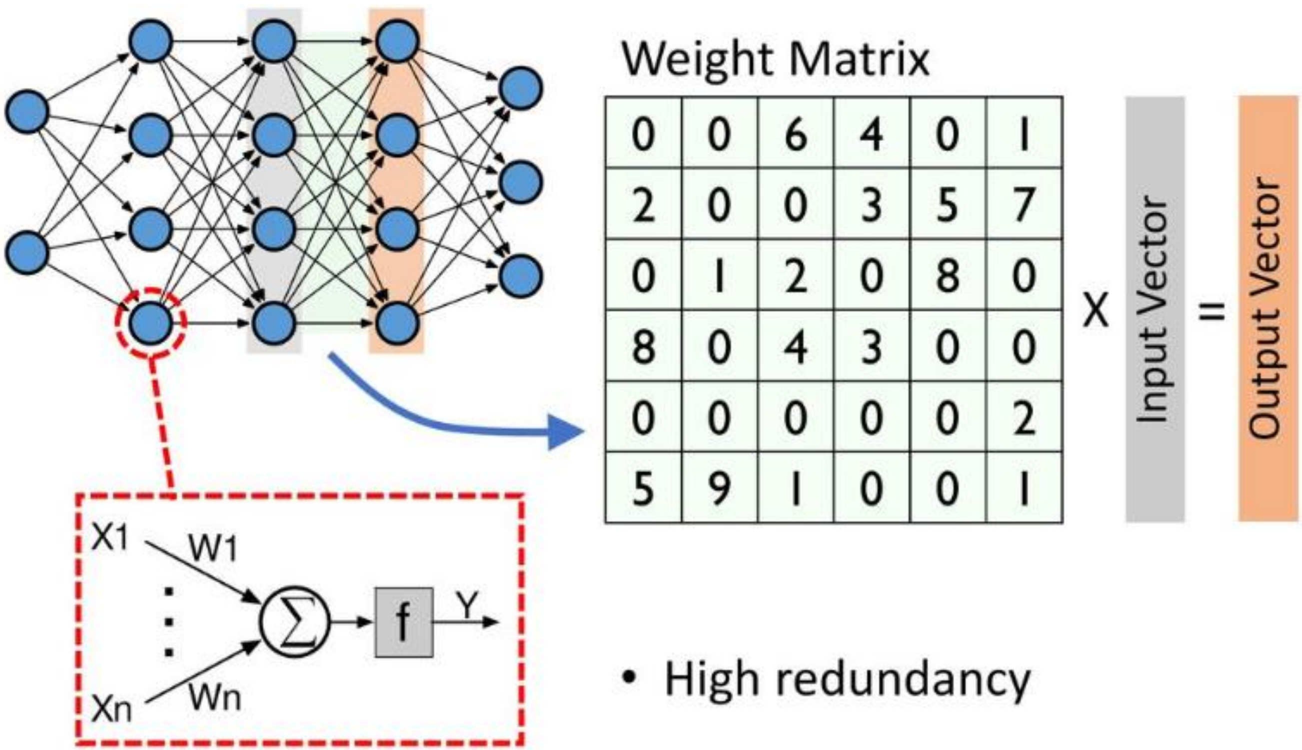


AlexNet ,VGG-16 x9 ~16 ↓

Figure 3: Synapses and neurons before and after pruning

Parameter pruning and sharing

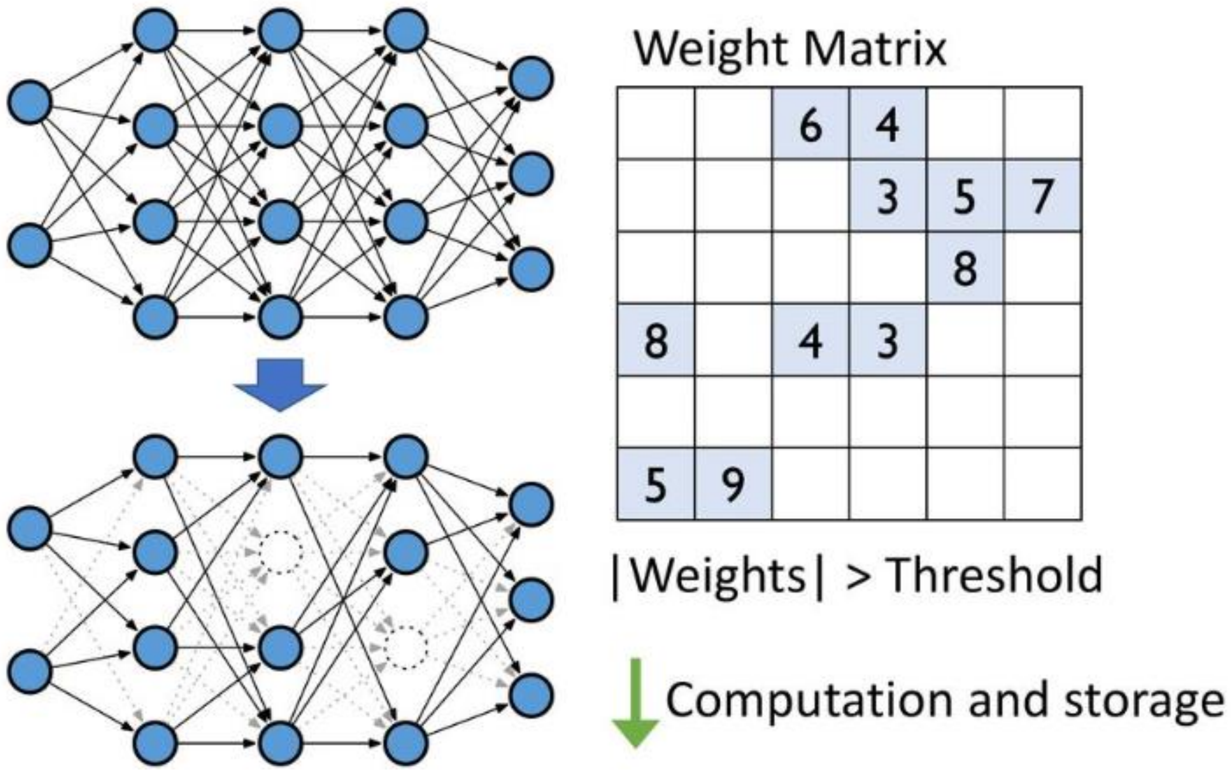
- 1) Pruning
  - **Employs a three-step process**
    - Begins by learning the connectivity via normal network training.
      - Not learning the final values of the weights



0	0	6	4	0	1
2	0	0	3	5	7
0	1	2	0	8	0
8	0	4	3	0	0
0	0	0	0	0	2
5	9	1	0	0	1

Parameter pruning and sharing

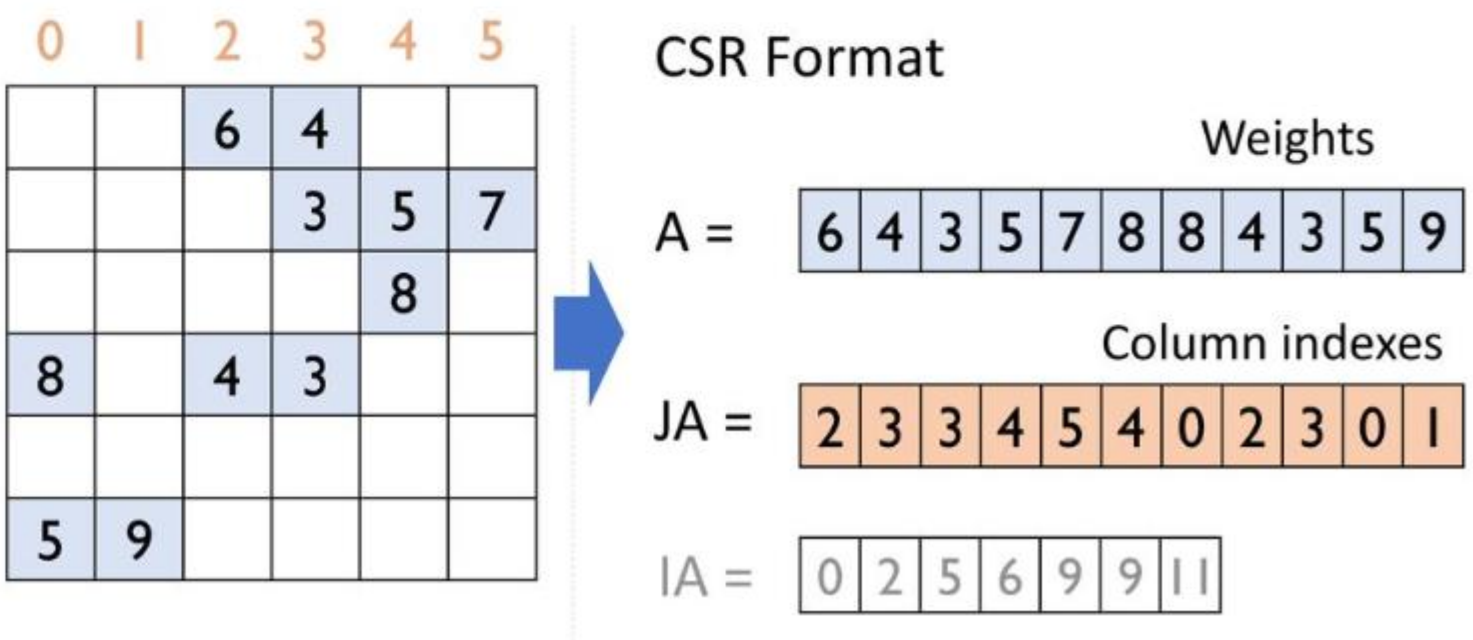
- 1) Pruning
  - **Employs a three-step process**
    - The second step is to prune the low-weight connections. All connections with weights below a threshold are removed from the network — converting a dense network into a sparse network



Parameter pruning and sharing

- 1) Pruning
  - **Employs a three-step process**
    - The final step retrains the network to learn the final weights for the remaining sparse connections.

  - But, problem?? ➔ Sparse format needs extra storage



Parameter pruning and sharing

- 1) Pruning

For Lenet-5, pruning reduces the number of weights by $12\times$ and computation by $6\times$

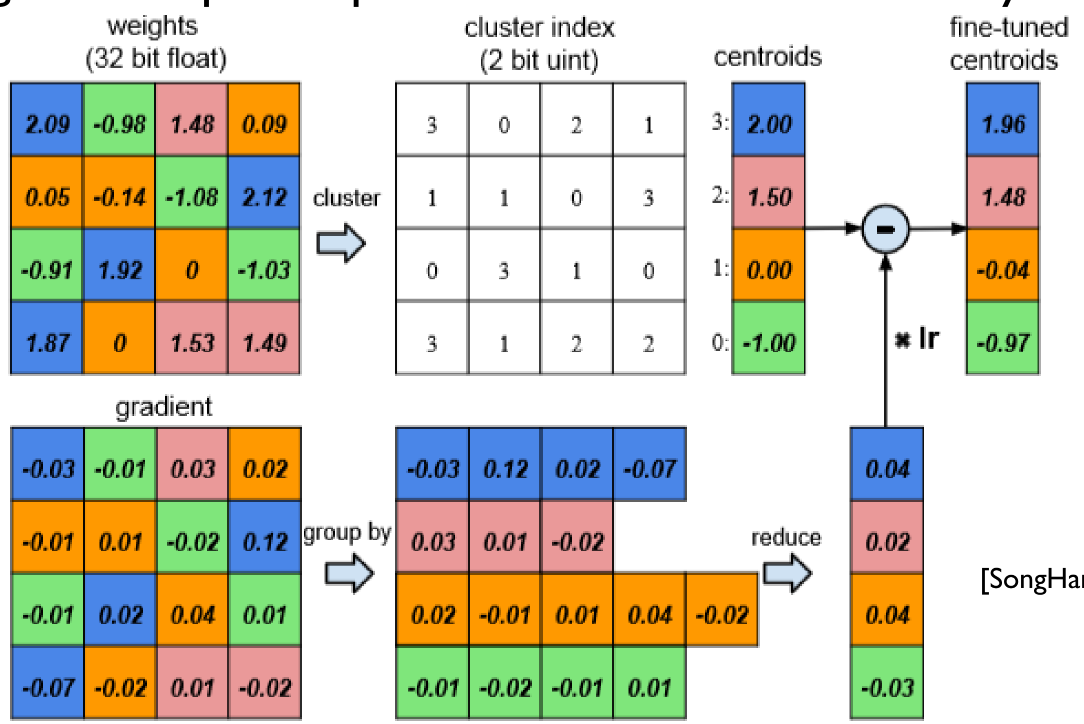
Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	0.5K	576K	82%	66%	66%
conv2	25K	3200K	72%	12%	10%
fc1	400K	800K	55%	8%	6%
fc2	5K	10K	100%	19%	10%
Total	431K	4586K	77%	**8%**	**16%**

Weights: original number of weights
FLOP: the number of floating point operations to compute that layer's activations
Act%: the average percentage of activations that are non-zero
Weights%: the percentage of non-zero weights after pruning
FLOP%: the percentage of actually required floating point operations

Parameter pruning and sharing

- 2) Quantization (Weight sharing) [Y. Gong, CoRR 2014, Y. W. Wu, CVPR 2016, V. Vanhoucke, NIPS 2011]
  - Applied k-means scalar quantization to reduce the number of bits required to represent each weight.
  - Store by having multiple connections share the same weight.
  - Result in significant speed-up with minimal loss of accuracy.

AlexNet
Quantize to 8-bits (256 shared weights) for each CONV layers, and 4-bits (16 shared weights) for each FC layer without any loss of accuracy.



[SongHan, ICLR 2016]

Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom).

Parameter pruning and sharing

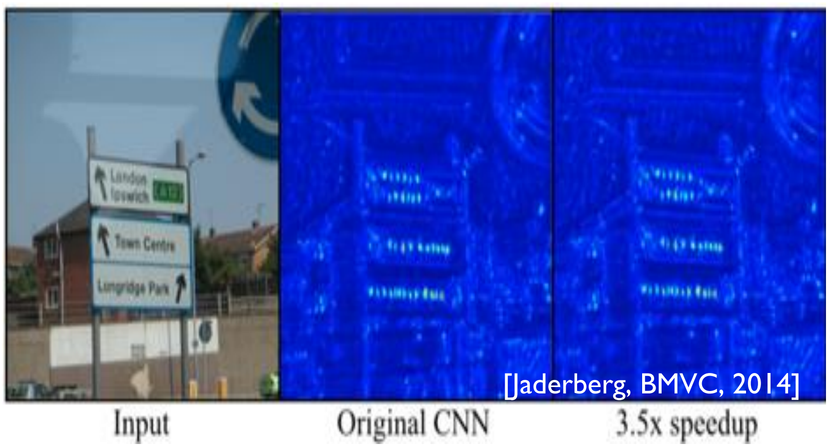
- 3) Pruning-Quantization [F. Tung, G. Mori, CVPR 2018]
  - Combine network pruning and weight quantization in a single operation and learn the pruned network structure and quantized weights together.
  - The full-precision weights are fine-tuned during training, and discarded after training is complete.

Layer	p	b	Original	Compressed	Rate
conv1	0.21	8	140 KB	35 KB	4×
conv2	0.36	6	1.2 MB	204 KB	6×
conv3	0.43	4	3.5 MB	395 KB	9×
conv4	0.32	4	2.7 MB	321 KB	8×
conv5	0.31	3	1.8 MB	174 KB	10×
fc6	0.96	3	151.0 MB	1.80 MB	84×
fc7	0.95	3	67.1 MB	969 KB	69×
fc8	0.74	3	16.4 MB	876 KB	19×
Overall			**243.9 MB**	**4.8 MB**	**51×**

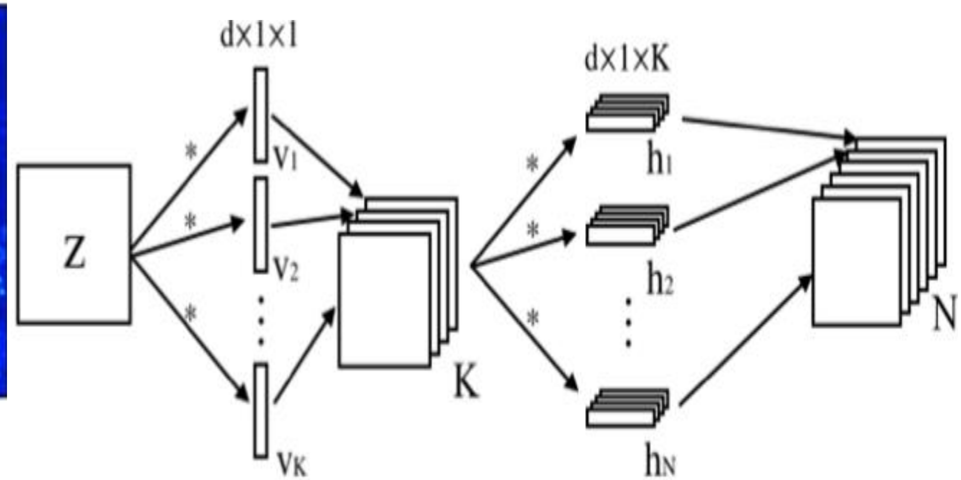
AlexNet on ImageNet (p: pruning rate, b: bits per weight).
Original top-1 accuracy: 57.2%.
Compressed top-1 accuracy: 57.9%.

Low-rank factorization

- Reducing the convolution layer would improve the compression rate as well as the overall speedup.
- Ideas based on [tensor decomposition]() is derived by the intuition that there is a significant amount of redundancy in 4D tensor(convolution filter)
  - Particularly promising way to remove the redundancy.



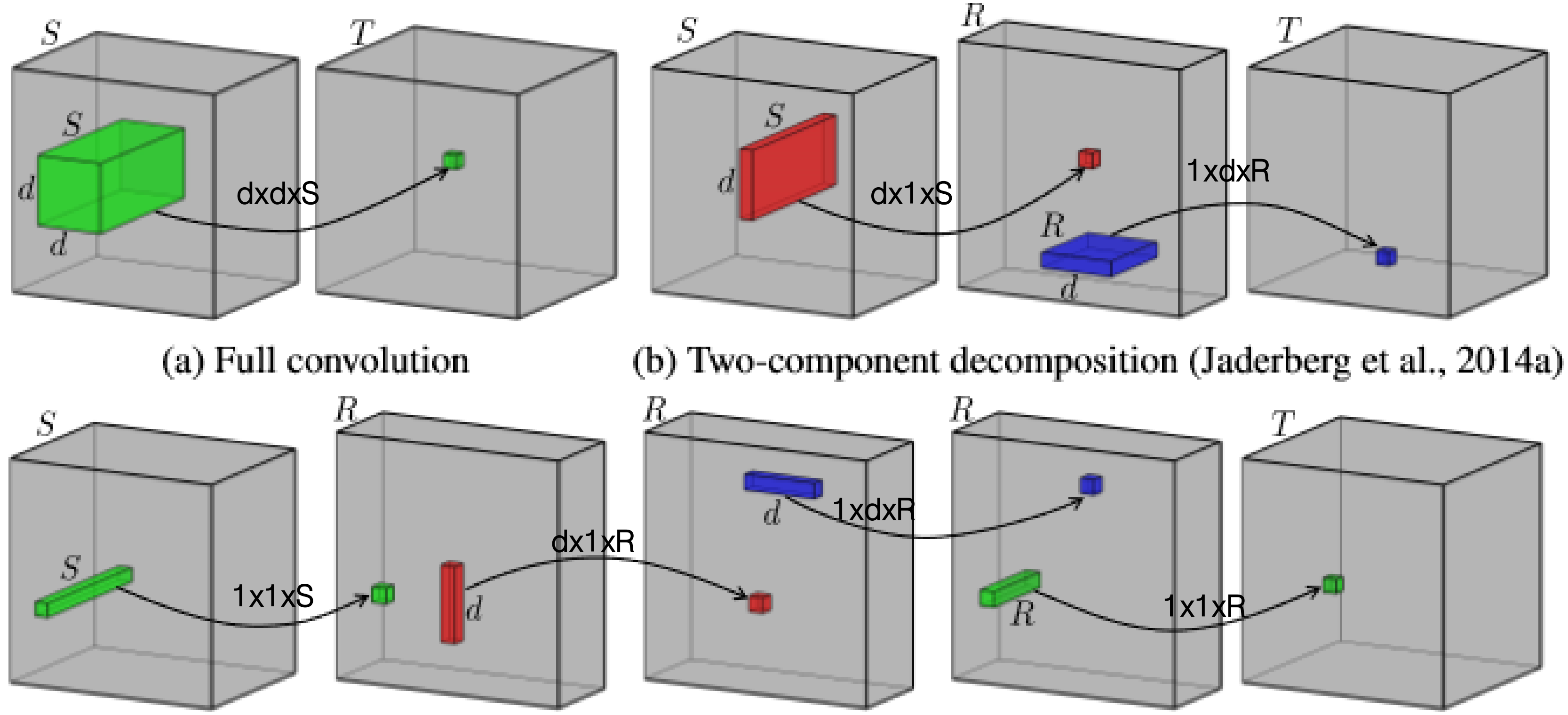
Original convolutional layer
(single-channel input)



Low-rank Scheme 2.
(single-channel input)

Low-rank factorization

Multi-channel input / CP Decomposition



CANDECOMP/PARAFAC decomposition

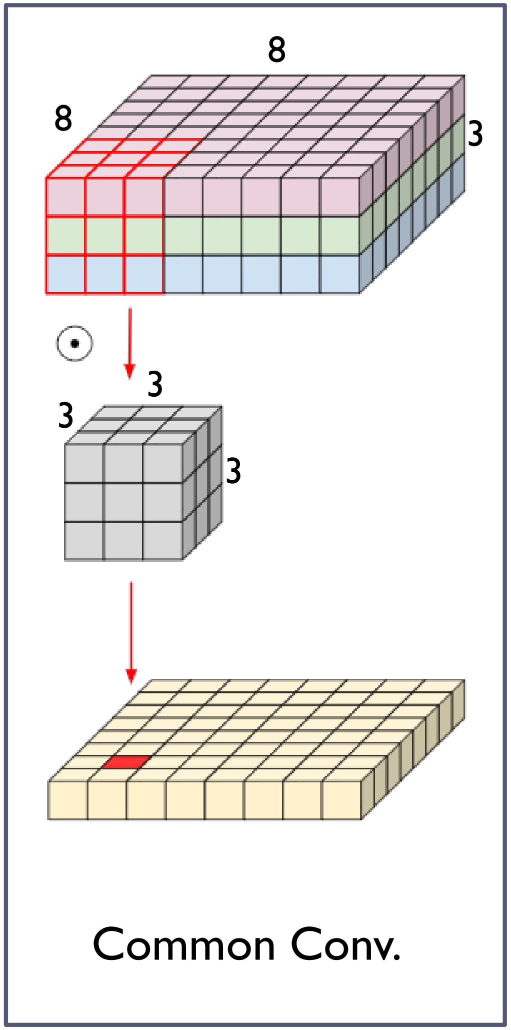
[Lebedev 2015 ICLR]

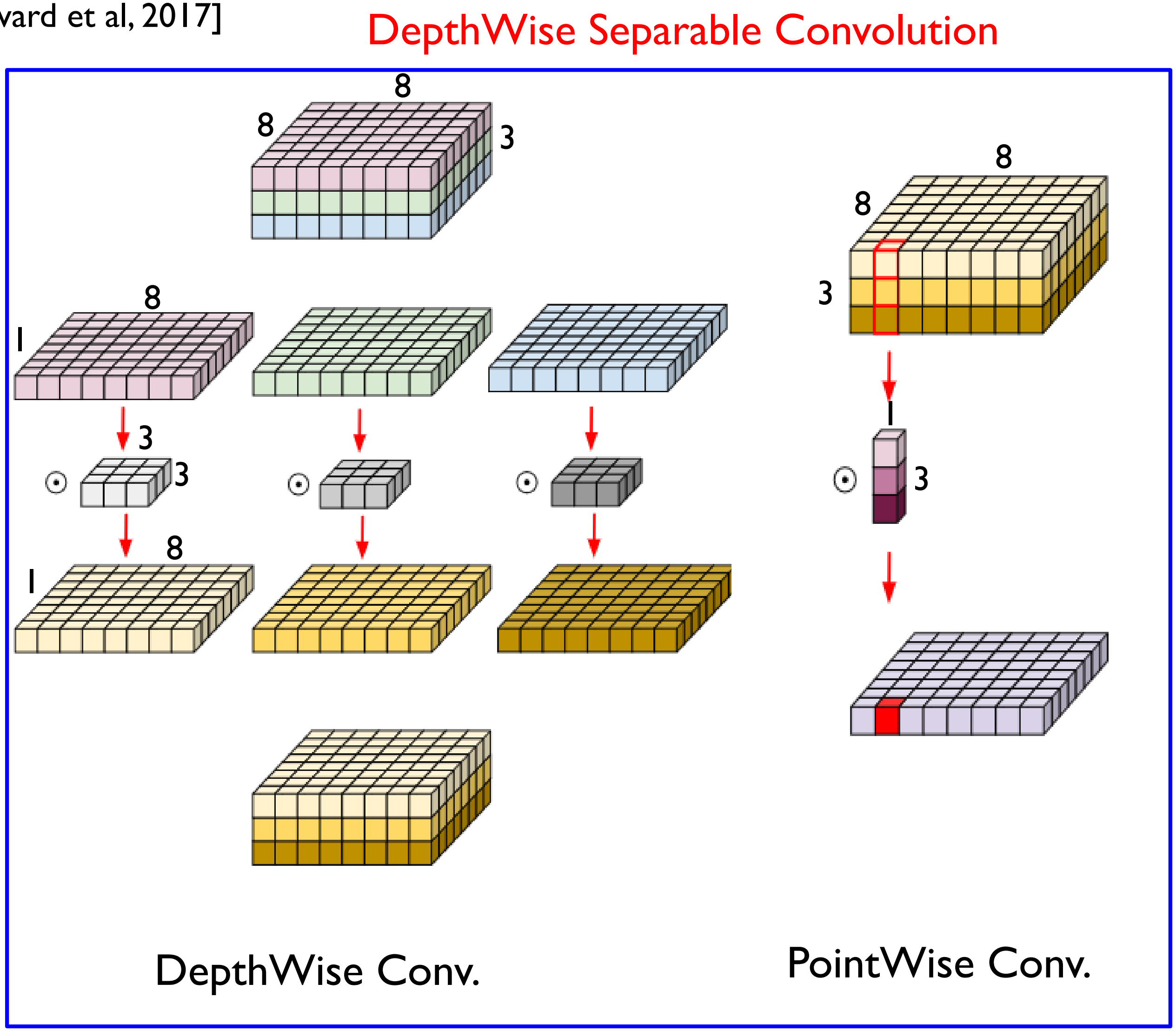
- Retraining required for convergence

FLOPs, Memory x2~4 ↓ (Loss accuracy ~1%)

Low-rank factorization

MobileNet [A. G. Howard et al, 2017]

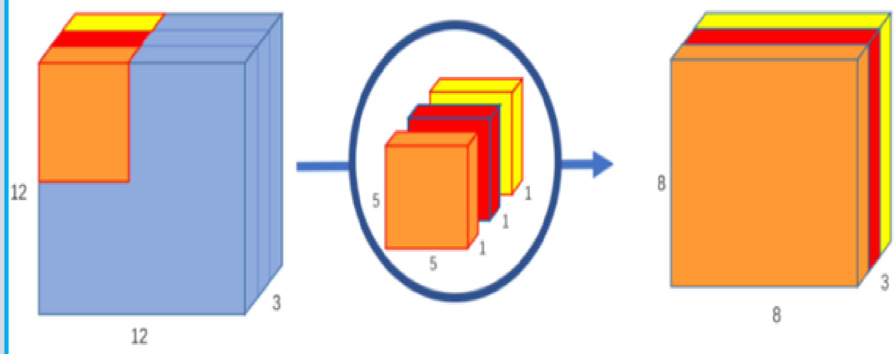




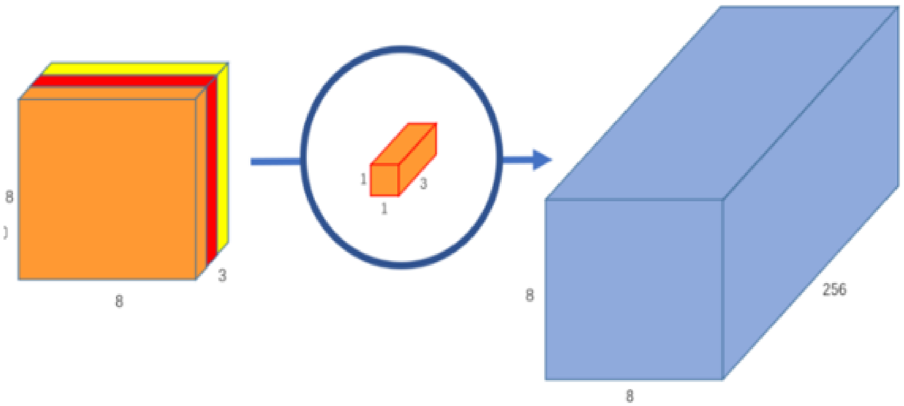
Low-rank factorization

MobileNet [A. G. Howard et al, 2017]

- Comparison of # Params & Ops



DepthWise Conv.



PointWise Conv.

Parms: 5x5x3

Ops: (12x12)x(5x5x3)

#Params: 1x1x3x256

#Ops: (8x8)x(1x1x3)x256

Common Conv.

#Total Params: 5x5x3x256=19,200

#Total Ops: (12x12)x(5x5x3)x256=2,764,800≈2M

MobileNet Conv.

Total Params: (5x5x3)+(1x1x3x256) = 843 (22.8 times ↓)

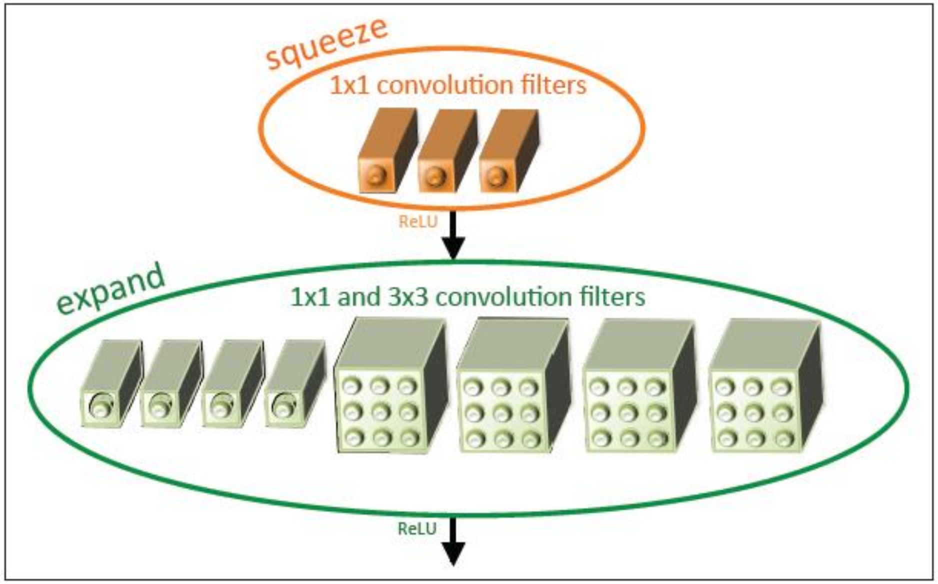
Total Ops: {(12x12)x(5x5x3)}+{(8x8)x(1x1x3)x256}=55,952 (29.9 ↓)

Low-rank factorization

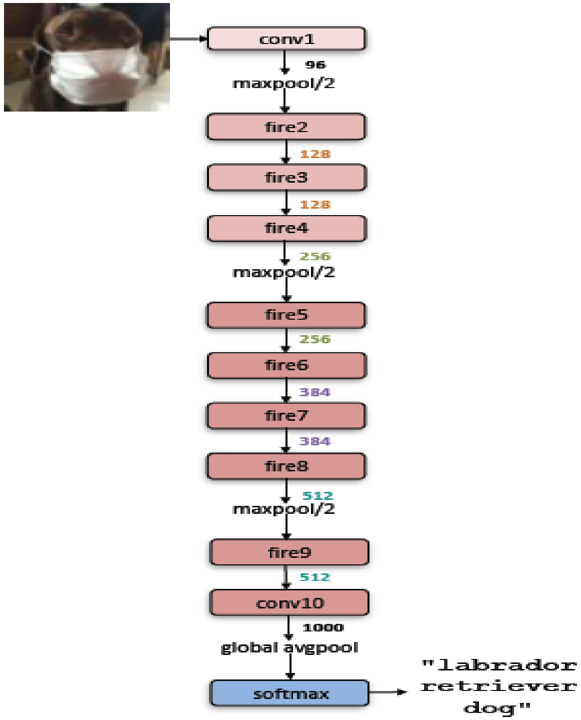
SqueezeNet [Forrest N. et al, arXiv 2016] AlexNet speed x50 ↓ model size < 5MB

- Architectural design strategies

1. Replace 3x3 filters with 1x1 filters (# of params $\frac{1}{9}$ ↓)
2. Decrease the number of input channels
3. Downsample late in the network so that convolution layers have large activation

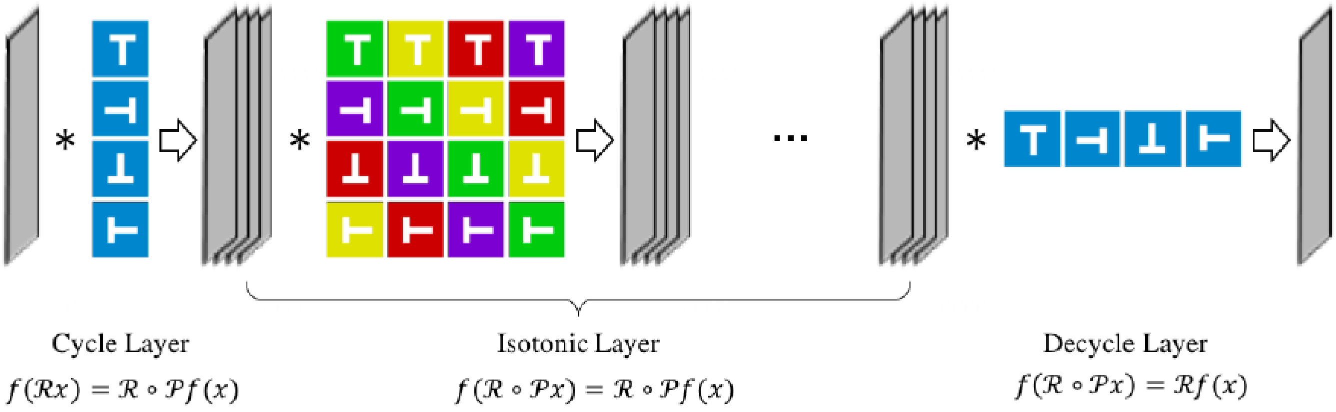


Fire Module = squeeze layer + expand layer



Transferred/compact convolutional filters

- Transferred convolutional filters (weights) [T. S. Cohen and M. Welling, arXiv2016]
  - Transforming the filters by the transform $T(\cdot)$, and then passing it through the network or layer to compress the whole network models.
  - Deep CNNs also benefit from using a large set of convolutional filters by applying certain transform $T(\cdot)$ to a small set of base filters since it acts as a regularizer for the model.
  - Ex)



rotated the original filters with angle $\theta \in \{0, 90, 180, 270\}$.

[J. Li et al. arxiv1705.08623v2, 2018]

Transferred/compact convolutional filters

- Transferred convolutional filters (weights)
  - Pros: Achieve reduction in parameters with little or no drop in classification accuracy.

COMPARISONS OF DIFFERENT APPROACHES BASED ON TRANSFERED CONVOLUTIONAL FILTERS ON CIFAR-10 AND CIFAR-100. (Top 5 error)

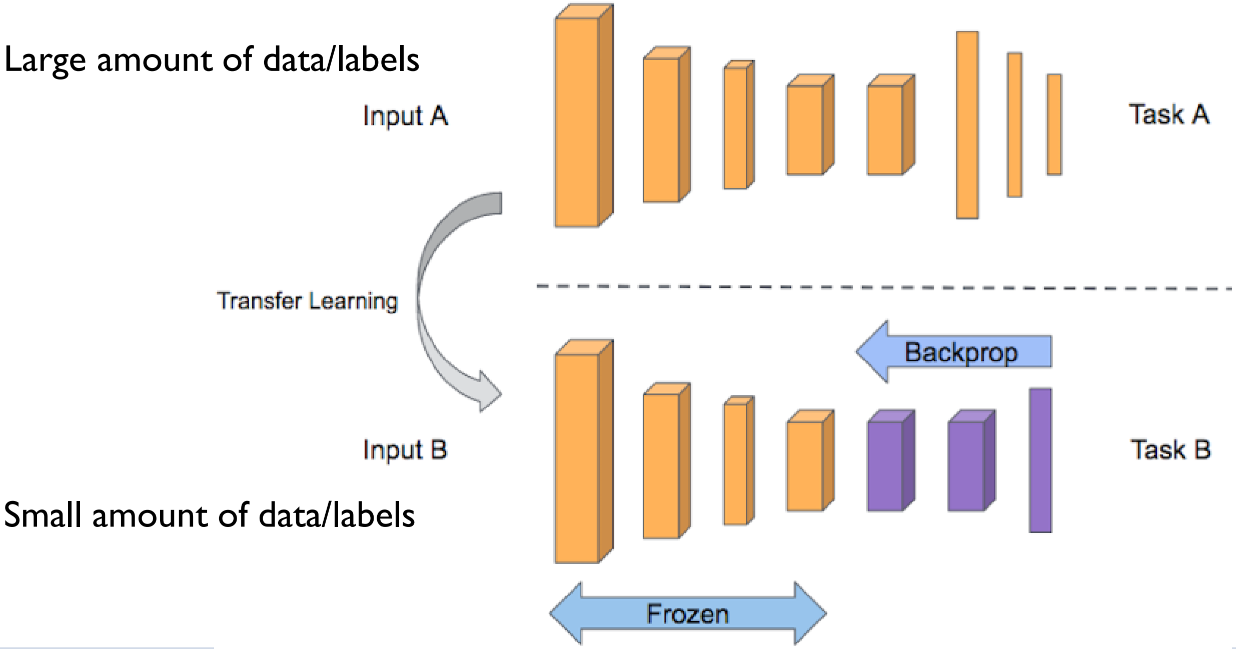
Model	CIFAR-100	CIFAR-10	Compression Rate
VGG-16	34.26%	9.85%	1.
MBA [45]	33.66%	9.76%	2.
CRELU [44]	34.57%	9.92%	2.
CIRC [42]	35.15%	10.23%	4.
DCNN [43]	33.57%	9.65%	1.62

  - Con:
    - Competitive performance for wide/flat architectures (like VGGNet) but not narrow/special ones (like GoogleNet, Residual Net)
    - Transfer assumptions sometimes are too strong to guide the algorithm, making the results unstable on some datasets.

[S. Diel, et al. ICML2016]

Teacher-Student Model

- Exploiting knowledge transfer (KT) to compress model was first proposed by Caruana et al. [C. Buciluˇa, ACM 2006]
  - Trained a compressed/ensemble model (Task B)of strong classifiers (Task A) with a small amount of data/labels
  - Reproduced the output of the original larger network.
- Limit.:The work is limited to shallow models.



Teacher-Student Model

- Knowledge Distillation (KD) [G. E. Hinton, CoRR 2015]
  - Compress deep and wide networks into shallower ones
  - Compressed model mimicked the function learned by the complex model.
  - Shift knowledge from a large teacher model into a small one by learning the class distributions output via softened softmax.

  - Basic idea: trains a student network, from the softened output of an ensemble of wider networks, teacher network.
  - The idea is to allow the student network to capture not only the information provided by the true labels, but also the finer structure learned by the teacher network

Teacher-Student Model

- Knowledge Distillation (KD) - terms
  - Hard target: a labeled data A with provided 0/1 labels
  - Soft target: a labeled data A with provided 0~1 class probability

  - P_T = softmax (a_T): soft output (output probability) of teacher pre-softmax output (a_T)

$$\mathrm{P}_{\mathrm{T}}^{\tau} = \mathrm{softmax}\left(\frac{\mathbf{a}_T}{\tau}\right) \quad = \quad P_T^{\tau} = \frac{\exp(\frac{aT_i}{\tau})}{\sum_j \exp(\frac{aT_j}{\tau})}$$

  - P_S = softmax(a_S): soft output (output probability,) of the student's pre-softmax output (a_S)

$$\mathrm{P}_{\mathrm{S}}^{\tau} = \mathrm{softmax}\left(\frac{\mathbf{a}_S}{\tau}\right) \quad = P_S^{\tau} = \frac{\exp(\frac{aS_i}{\tau})}{\sum_j \exp(\frac{aS_j}{\tau})}$$

Teacher-Student Model

- Knowledge Distillation (KD)
  - Soften softmax
    - 1) more informative than the original 0/1 class labels
    - 2) impart the relationship between different classes

dog

An example of hard and soft targets

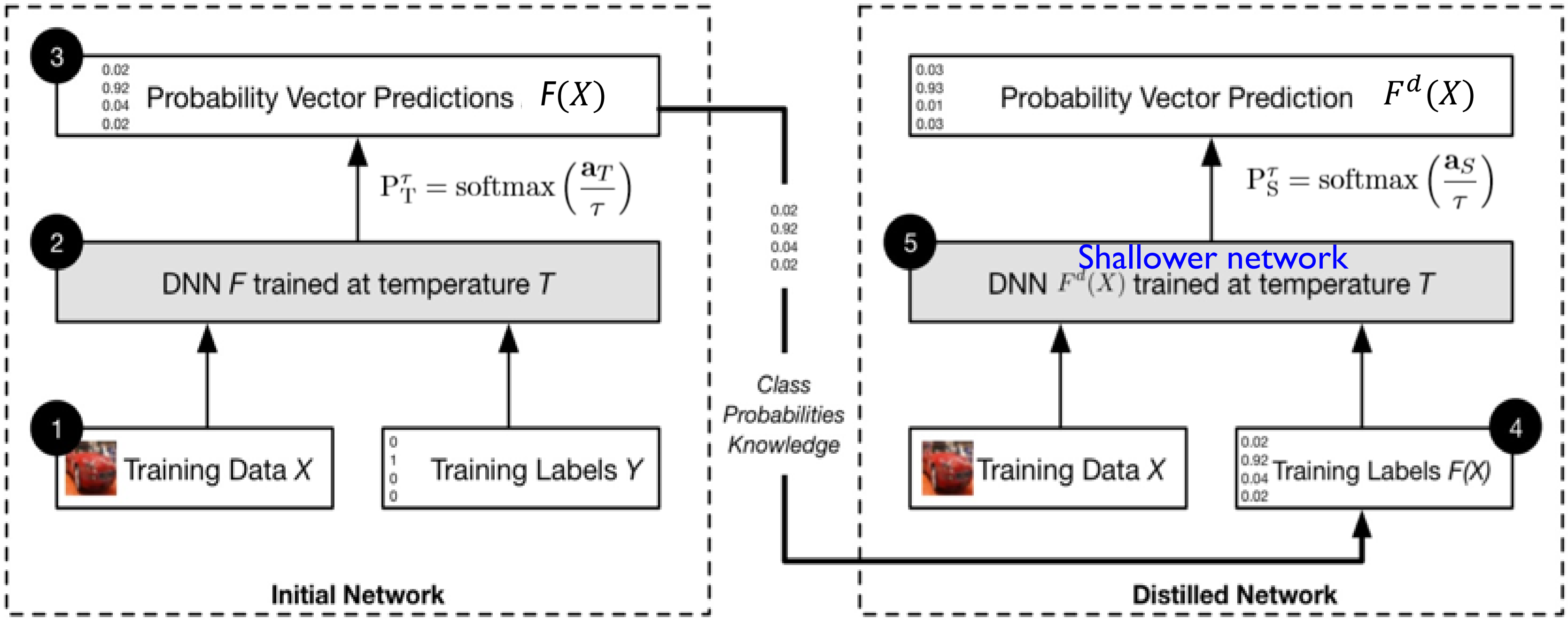
cow	dog	cat	car	
0	1	0	0	original hard targets
10^{-6}	.9	.1	10^{-9}	output of geometric ensemble
.05	.3	.2	.005	softened output of ensemble

Softened outputs reveal the dark knowledge in the ensemble.

Problem ? Overfitting ?

Teacher-Student Model

- Knowledge Distillation (KD)
  - Apply soft target distribution from the result of model Teacher to train model Student
    - Ability to pass the knowledge learned in model Teacher to the model Student



Teacher-Student Model

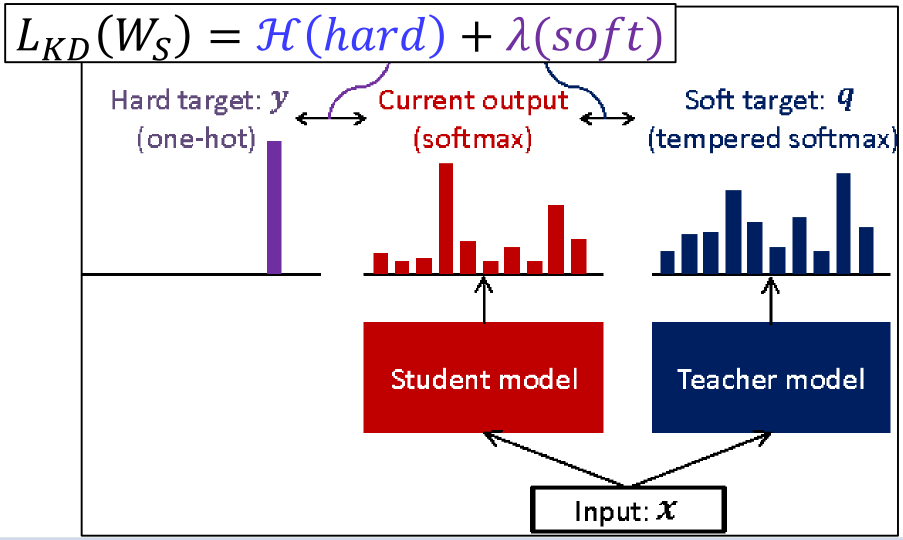
- Knowledge Distillation (KD)
  - Student network is then trained to optimize the following loss function

$$\boxed{\mathcal{L}_{KD}(\mathbf{W_S}) = \mathcal{H}(\mathbf{y_{true}}, \mathrm{P_S}) + \lambda\mathcal{H}(\mathrm{P_T^\tau}, \mathrm{P_S^\tau}),}$$

H refers to the cross-entropy and λ is a tunable parameter to balance both cross-entropies

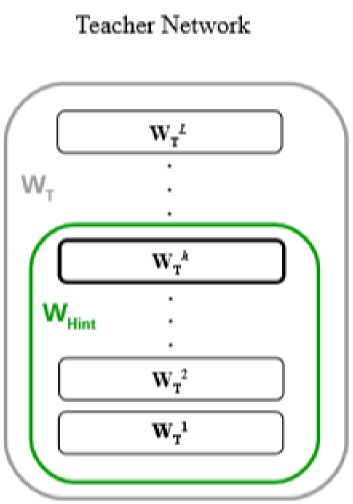
$\mathcal{H}(\mathbf{y_{true}}, \mathrm{P_S})$: traditional cross-entropy between the output of a (student) network and labels

$\mathcal{H}(\mathrm{P_T^\tau}, \mathrm{P_S^\tau})$: enforces the student network to learn from the softened output of the teacher network



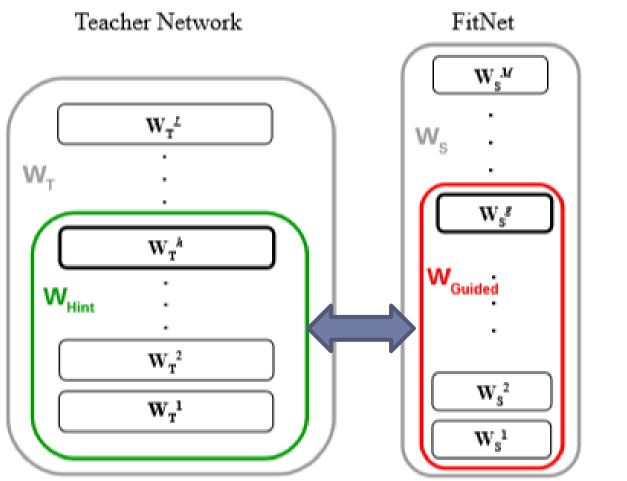
Teacher-Student Model

- FitNets: Hints for thin deep nets [A. Romero, ICLR 2015]
  - Proposed an approach to train thin but deeper networks than their teacher, called FitNets
  - Create a model that was 10 times more efficient, with fewer multiplication times to inference as the number of parameters decreased.
  - Key idea: train student to resemble only intermediate hidden layers (hints) rather than whole structure

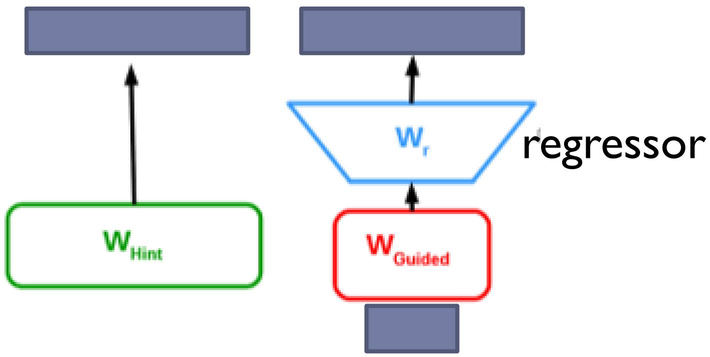


Teacher-Student Model

- FitNets: Hints for thin deep nets [A. Romero, ICLR 2015]
  - Want the guided layer to be able to predict the output of the hint layer.
  - Problem: hard to make a student similar because it is thinner than a teacher.
  - Solution: add a regressor to the guided layer, expand the dimension
    ➔ output matches the size of the hint layer.



(a) Teacher and Student Networks



Teacher-Student Model

- FitNets: Hints for thin deep nets [A. Romero, ICLR 2015]
  - The regressor parameters by minimizing the following loss function:

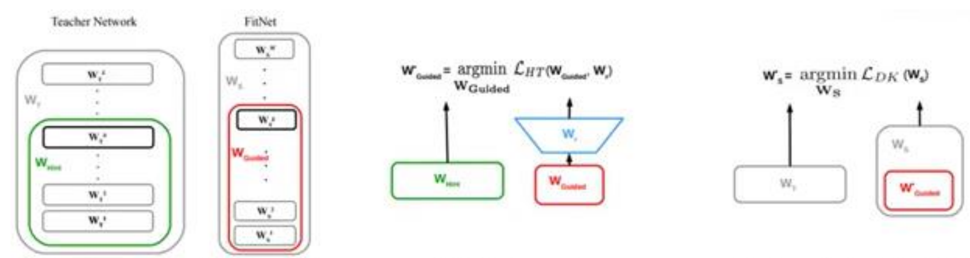
Regressor **Weight for Regressor**

$$\mathcal{L}_{HT}(\mathbf{W_{Guided}}, \mathbf{W_r}) = \frac{1}{2}||\underbrace{u_h(\mathbf{x}; \mathbf{W_{Hint}})}_{\text{Teacher}} - r(\underbrace{v_g(\mathbf{x}; \mathbf{W_{Guided}}); \mathbf{W_r}}_{\text{Student}})||^2,$$

u_h and v_g : the teacher/student deep nested functions

r: regressor function on top of the guided layer with parameters Wr

Input: $\mathbf{W_S}, \mathbf{W_T}, g, h$
Output: $\mathbf{W_S^*}$
$\boxed{\mathbf{W_{Hint}}} \leftarrow \{\mathbf{W_T}^1, \ldots, \mathbf{W_T}^h\}$
$\boxed{\mathbf{W_{Guided}}} \leftarrow \{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\}$
Intialize $\boxed{\mathbf{W_r}}$ to small random values
$\boxed{\mathbf{W^*_{Guided}}} \leftarrow \underset{\mathbf{W_{Guided}}}{\text{argmin}} \; \mathcal{L}_{HT}(\mathbf{W_{Guided}}, \mathbf{W_r})$
$\{\mathbf{W_S}^1, \ldots, \mathbf{W_S}^g\} \leftarrow \{\mathbf{W_{Guided}}^{*1}, \ldots, \mathbf{W_{Guided}}^{*g}\}$
$\mathbf{W_S^*} \leftarrow \underset{\mathbf{W_S}}{\text{argmin}} \; \mathcal{L}_{KD}(\mathbf{W_S})$



(a) Teacher and Student Networks (b) Hints Training (c) Knowledge Distillation

Teacher-Student Model

- FitNets: Hints for thin deep nets [A. Romero, ICLR 2015]
  - FitNet outperforms the teacher model, reducing the number of parameters by a factor of 3
  - Provides near state-of-the-art performance.

Algorithm	# params	Accuracy
Compression		
FitNet	~2.5M	**64.96%**
Teacher	~9M	63.54%
State-of-the-art methods		
Maxout		61.43%
Network in Network		64.32%
Deeply-Supervised Networks		**65.43%**

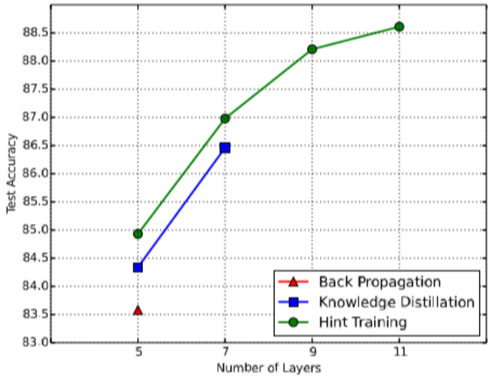
Table 2: Accuracy on CIFAR-100

32 X 32 50k training 10k testing # of parameter 28 ↓

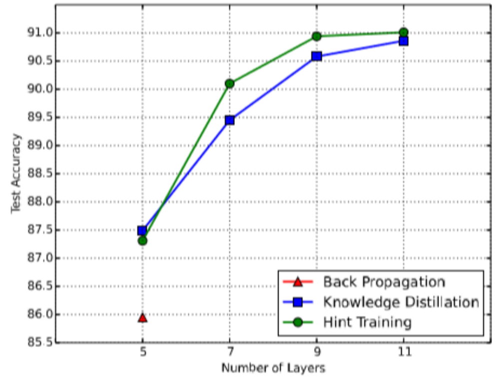
Teacher-Student Model

- FitNets: Hints for thin deep nets [A. Romero, ICLR 2015]
  - Deep models have better performances than shallower ones given a fixed computational budget

Deep is better



(a) 30M Multiplications



(b) 107M Multiplications

Shallow-Deep VS. Wide-Deep

  - Accuracy/Speed Trade-off on CIFAR-10

Network	# layers	# params	# mult	Acc	Speed-up	Compression rate
Teacher	5	~9M	~725M	90.18%	1	1
FitNet 1	11	~250K	~30M	89.01%	**13.36**	**36**
FitNet 2	11	~862K	~108M	91.06%	4.64	10.44
FitNet 3	13	~1.6M	~392M	91.10%	1.37	5.62
FitNet 4	19	~2.5M	~382M	**91.61%**	1.52	3.60

Teacher-Student Model

- Procs & Cons of KD based approaches (including FitNets)
  - Pros: KD-based Approaches can make deeper models shallower or deeper & thinner and help significantly reduce the computational cost.
  - Cons:
    - KD can only be applied to classification tasks with softmax loss function, which hinders its usage.
    - The model assumptions sometimes are too strict to make the performance competitive with other type of approaches.
    - Still applied to small scale models

Discussion

- A major problem to hinder the extension of deep CNNs.
  - Hardware constraints in various of small platforms (e.g., mobile, robotic, self-driving car)
  - In terms of Compression, we have to study
    - How to make full use of the limited computational source available
    - How to design special compression methods for such platforms

Discussion

- How to choose the proper approaches ?
  - Depending on the applications and requirements
  - 1) Need compacted models from pre-trained models
    - Either pruning & sharing or low rank factorization based methods.
  - 2) Need end-to-end solutions
    - Low rank and transferred convolutional filters
  - 3) For applications in some specific domains, methods with human prior
    - Medical images classification, transferred convolutional filters should work well as medical images (like organ) do have the rotation transformation property
  - 4) stable model accuracy
    - Pruning & sharing could give reasonable compression rate while not hurt the accuracy
  - 5) Involves small/medium size datasets
    - Knowledge distillation approaches (Teacher student)

[Yu Cheng et al., 2017]

Discussion

- How to choose the proper approaches ?
  - 6) Makes senses to combine two or three of them to maximize the compression/speed up rates.
    - E.g) like object detection
      - Compress the convolutional layers with low rank factorization
      - Fully connected layers with a pruning method.

[Yu Cheng et al., 2017]

References

- C. Szegedy, Going deeper with convolutions, CVPR2015
- 김용덕, compressing CNN for mobile device, Samsung S/W R&D Center
- Lebedev, SPEEDING-UP CONVOLUTIONAL NEURAL NETWORKS,2015 ICLR
- Yu Cheng, A Survey of Model Compression and Acceleration for Deep Neural Networks, IEEE SIGNAL PROCESSING MAGAZINE, 2017
- S. Dieleman, J. De Fauw, and K. Kavukcuoglu, “Exploiting cyclic symmetry in convolutional neural networks,” in Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ser. ICML’16, 2016.
- T. S. Cohen and M. Welling, “Group equivariant convolutional networks,” arXiv preprint arXiv:1602.07576, 2016.
- G. E. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” CoRR, vol. abs/1503.02531, 2015
- A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “Fitnets: Hints for thin deep nets,” CoRR, vol. abs/1412.6550, 2014.
- A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in NIPS, 2012.
- Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in Proceedings of the IEEE, 1998, pp. 2278–2324.
- K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv:1512.03385, 2015

References

- M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, 2015, pp. 3123–3131.
- M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," CoRR, vol. abs/1602.02830, 2016
- M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in ECCV, 2016.
- J. Li et al. Deep Rotation Equivariant Network, arxiv1705.08623v2, 2018

Thanks