Compressing Neural Networks Structures

Spring 2018

The University of Iowa

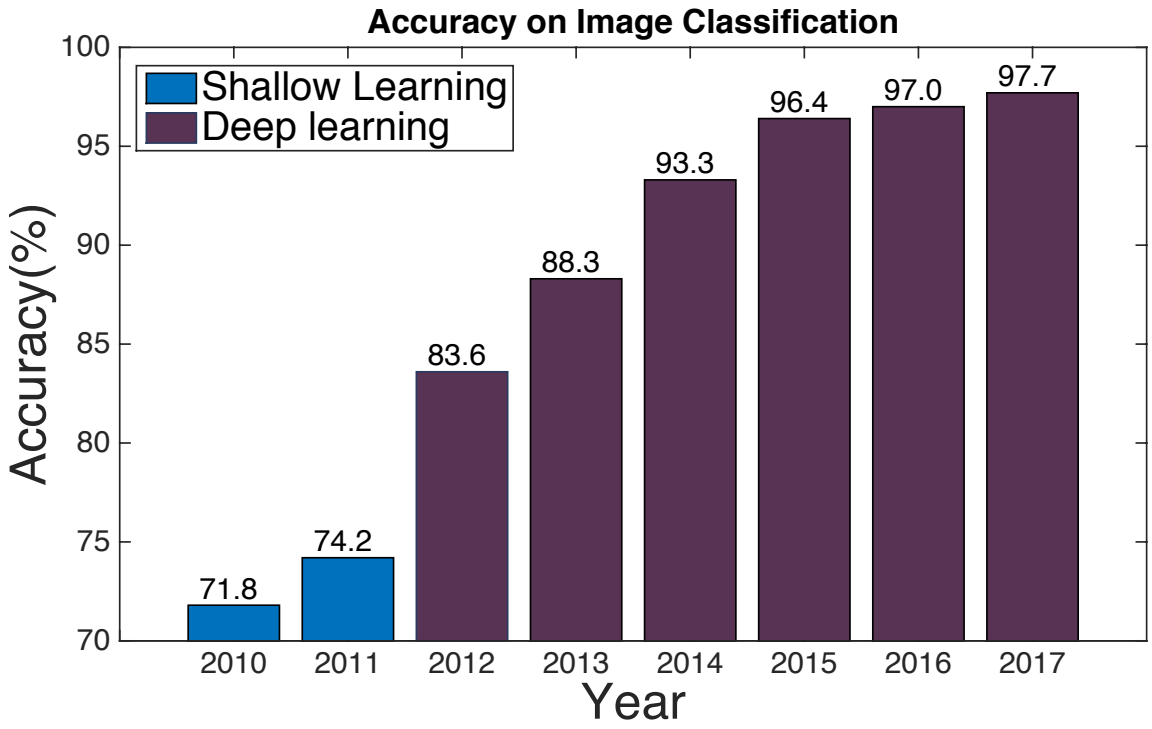
Zhe Li

Content

- **Motivation**
- Where to focus on to compress
- Different compressing methods
- SEP-Net (My Research Work)
- Experimental Results

Motivation

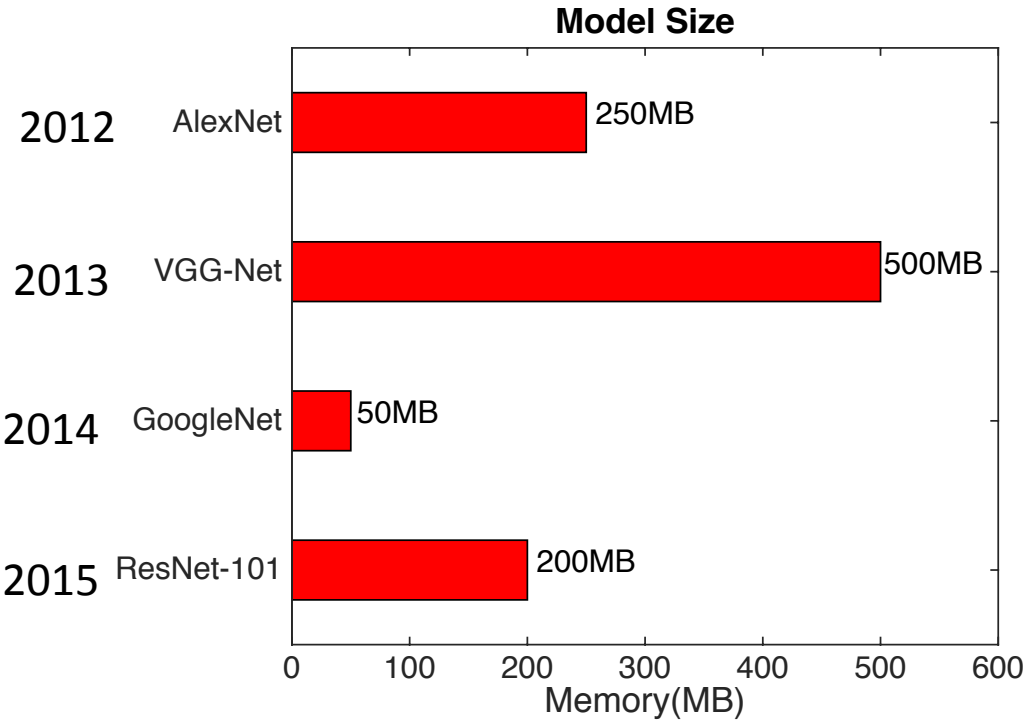
- Example: Image classification

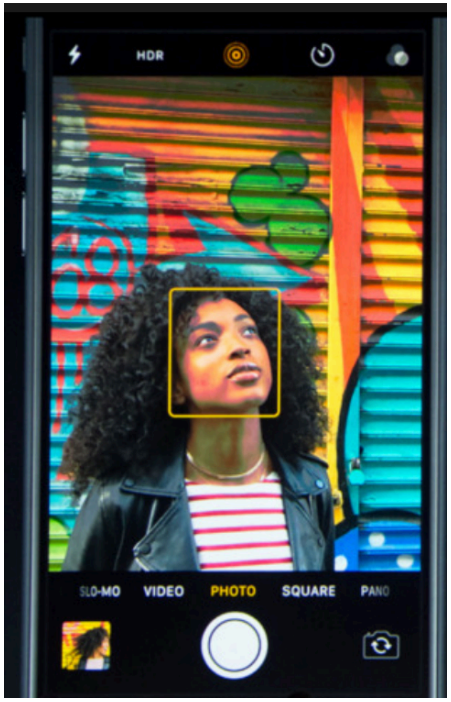


Accuracy: Test on ImageNet Benchmark Dataset (about 1.2 million images, 1000 classes)

Motivation

- Example: Image classification





Model size too large to be deployed to mobile phone or embedded system

model size = # of parameters * 4 Bytes

Motivation

- Can we reduce model size without sacrificing performance loss for mobile phone or embedded application?
  - Reducing memory usage in neural network.
- If so, how?
  - Locating where memory cost are from.
  - Applying different techniques (group-wise convolution, quantization, pruning, etc.) to reduce memory usage.

Content

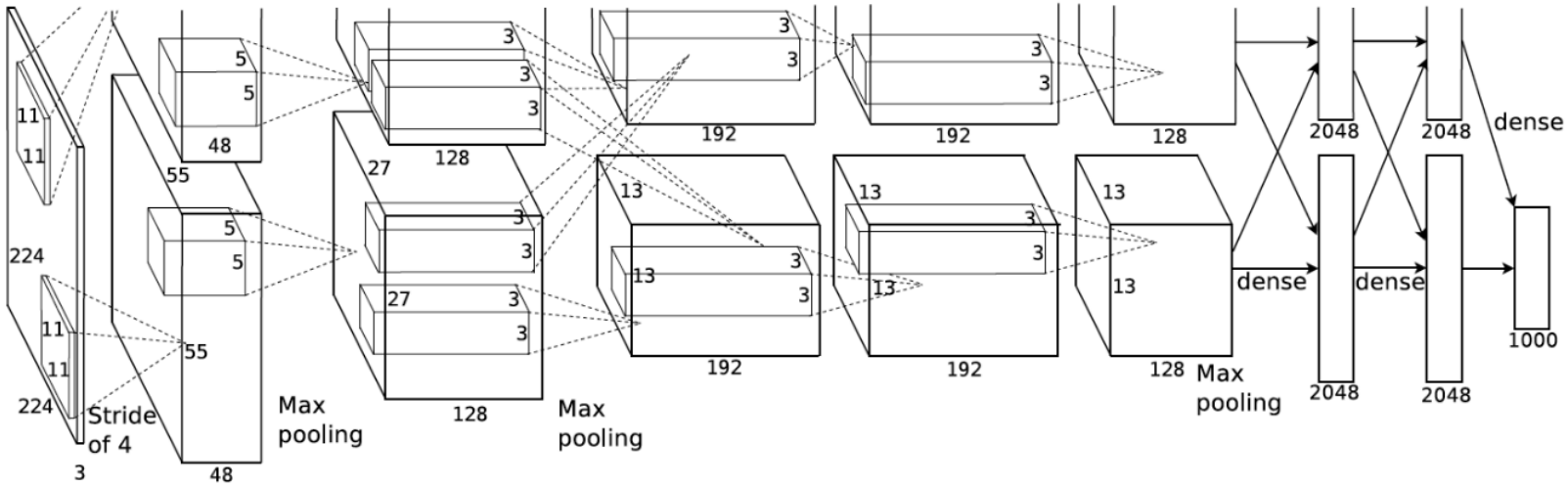
- Motivation
- **Where to focus on to compress**
- Different compressing methods
- SEP-Net (My Research Work)
- Experimental Results

Locating parameters - History View

- AlexNet (2012 ImageNet Competition Winner)
  - Breakthrough work.
  - Proposed and implemented in University of Toronto
  - Based on open source library CudaConvNet
  - Existing open source libraries such as Caffe, Tensorflow and others follows some design principle of CudaConvNet.

Locating parameters - History View

- AlexNet



- Highlights

Input: 3*224*224 size image

Output: 1000 size vector, using one index to indicate specific class

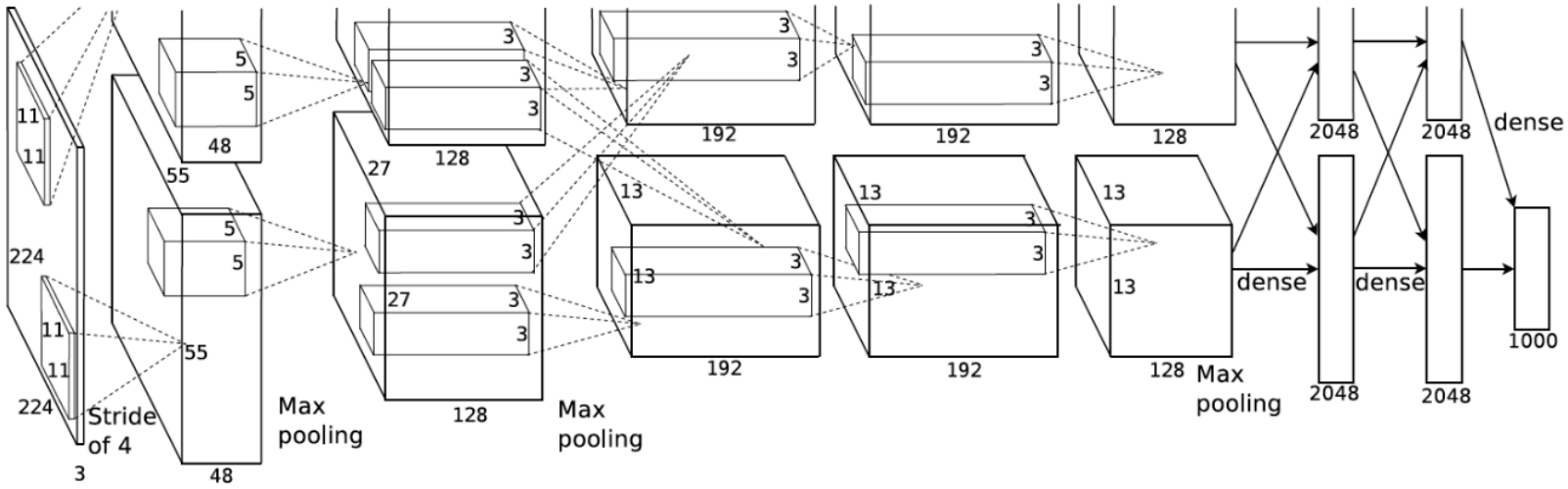
Structure: 5 convolution layers, 3 fully connected layers

Special feature: Two GPU cards are used

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Locating parameters - History View

- AlexNet



- Compute # of parameters (first convolution layer)
  - 96 filters of size 11*11*3 (3 colors of 11*11 window)
  - (11*11*3 + 1)*96 = 34944

Locating parameters - History View

- AlexNet

Parameters in Each Layer

Conv1	34.9K
Conv2	614.6K
Conv3	885.1K
Conv4	1.3M
Conv5	884.9K
FC6	37.7M
FC7	16.7M
FC8	4.1M
Total	62.3M

250MB

Layer type	# Params	Percent
Conv	3.7M	6%
FC	58.6M	94%

Questions: Which layer should we focus on for reducing # of parameters?

Locating parameters - History View

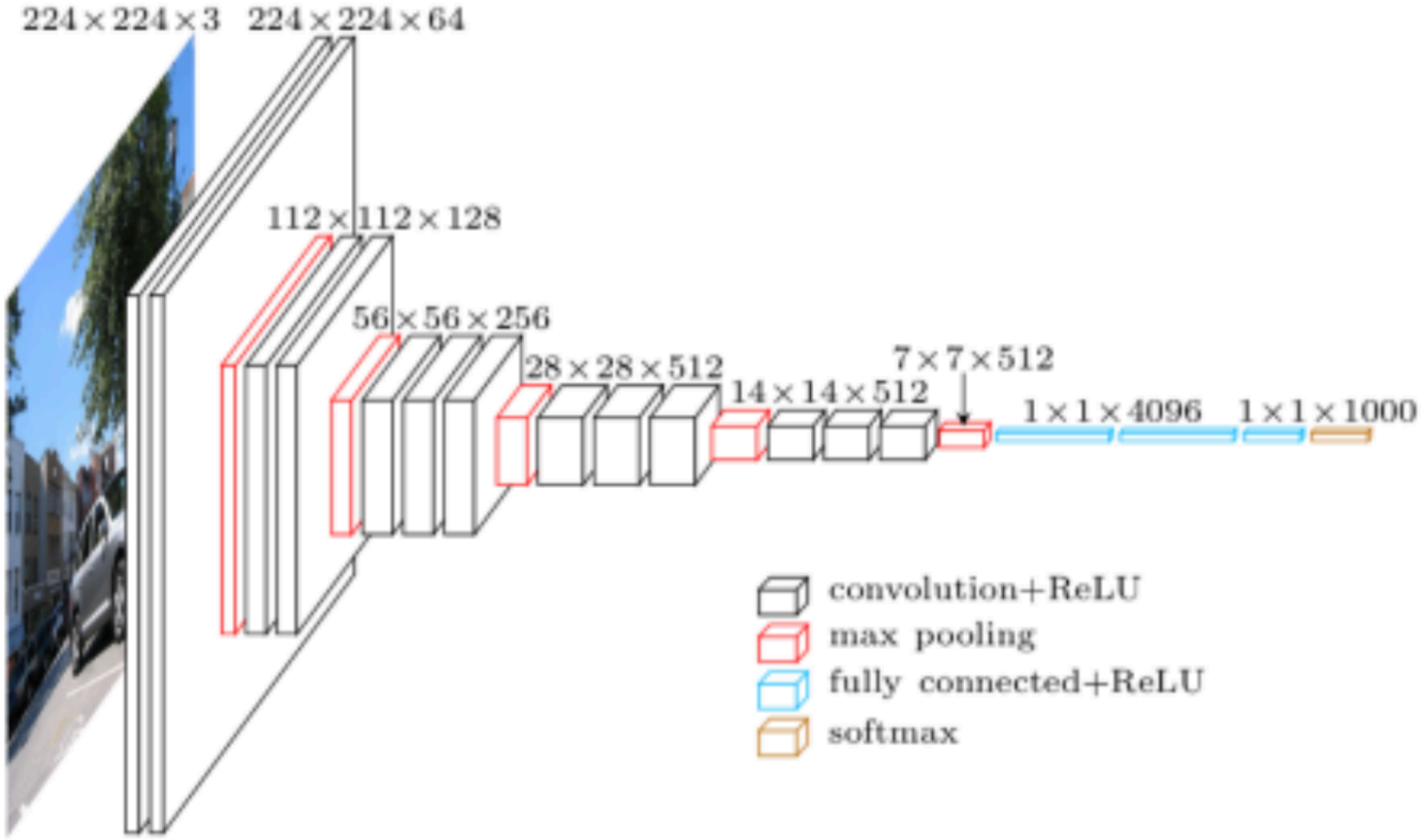
- VGG-Net (2013 ImageNet Competition Winner)

- 13 layers convolution layers
- 3 layers fully connected layers
- Researchers were focusing on improving on accuracy
  - CIFAR-10: validation accuracy of 93.56%
  - CIFAR-100: validation accuracy of 70.48%.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015

Locating parameters - History View

- VGG-Net (2013 ImageNet Competition Winner)



Locating parameters - History View

- VGG-Net (2013 ImageNet Competition Winner)

Conv1	1.7K
Conv2	36.8K
Conv3	73.7K
Conv4	147.4K
Conv5	294.9K
…	…
…	…
conv13	2.4M
FC14	102.77M
FC15	16.7M
FC16	4.1M
Total	138M

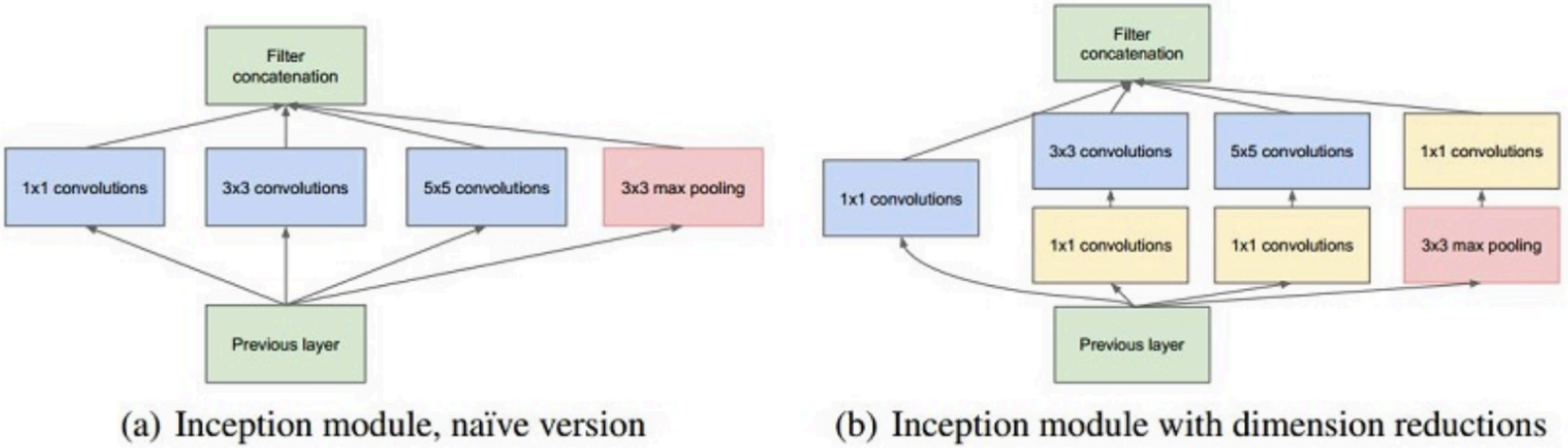
450MB

Layer type	# Params	Percent
Conv	14.4M	10.4
FC	123.6M	89.6

Questions: Which layer should we focus on for reducing # of parameters?

Locating parameters - History View

- GoogleNet (2014 ImageNet Competition Winner)

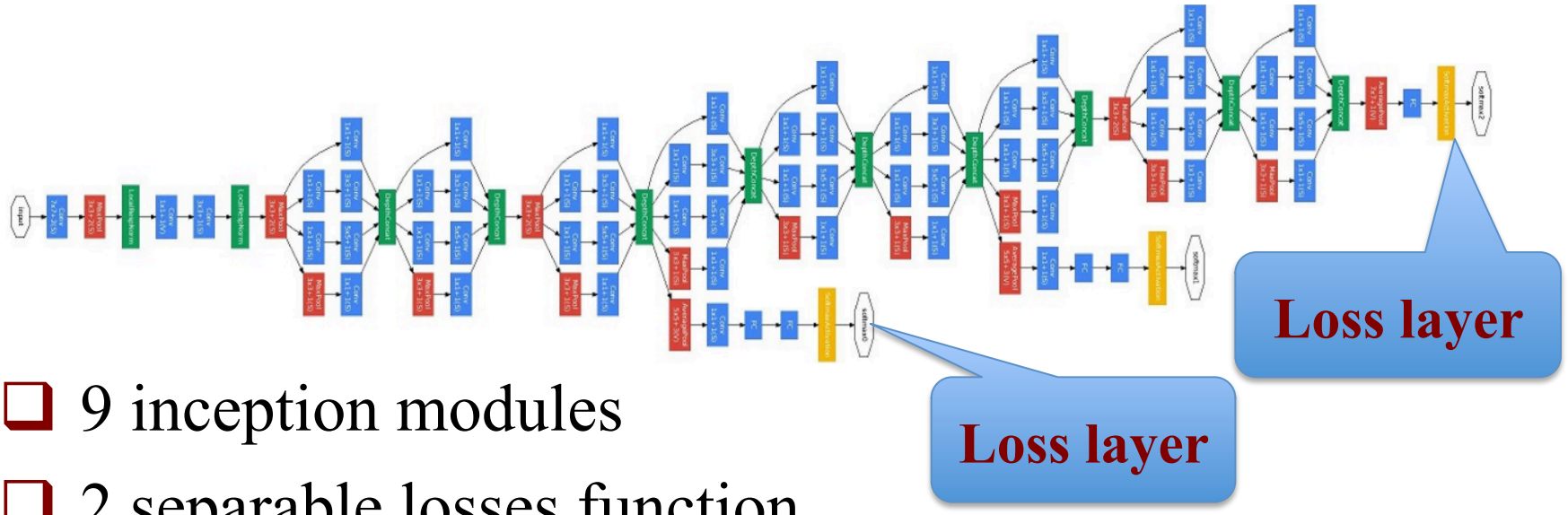


- Design base module: inception module
- Multiple path compared with previous network structure

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions.

Locating parameters - History View

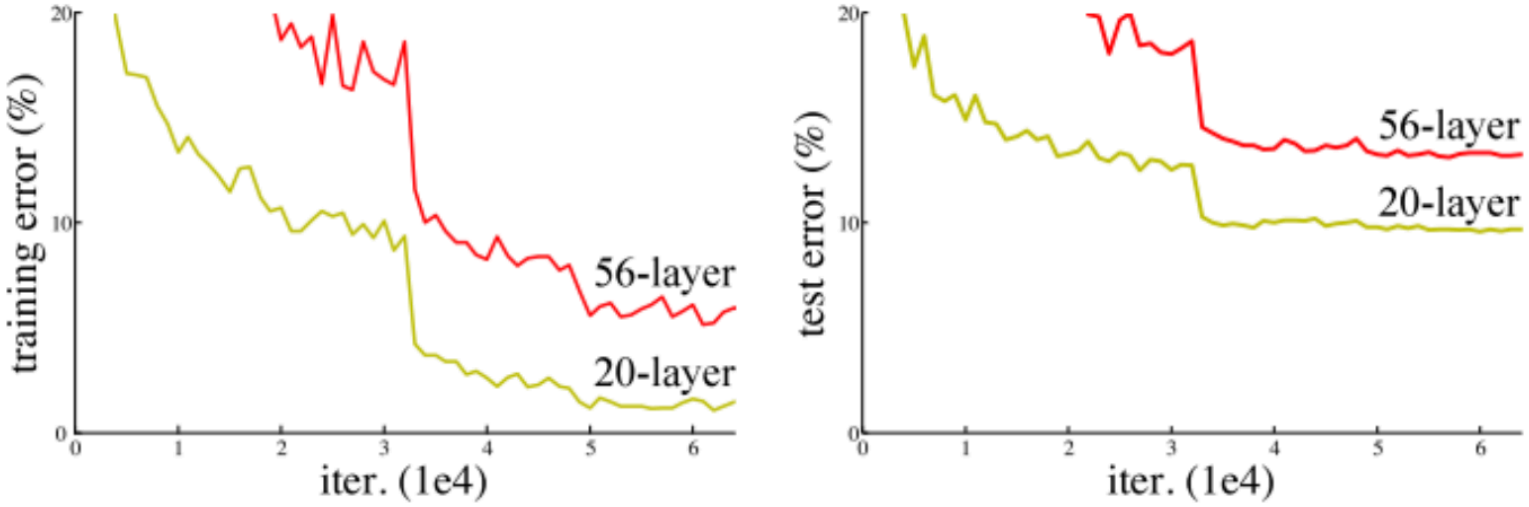
- GoogleNet (2014 ImageNet Competition Winner)



- 9 inception modules
- 2 separable losses function
- Remove fully connected layers
- Model size reduce to 50MB

Locating parameters - History View

- ResNet (2015 ImageNet Competition Winner)



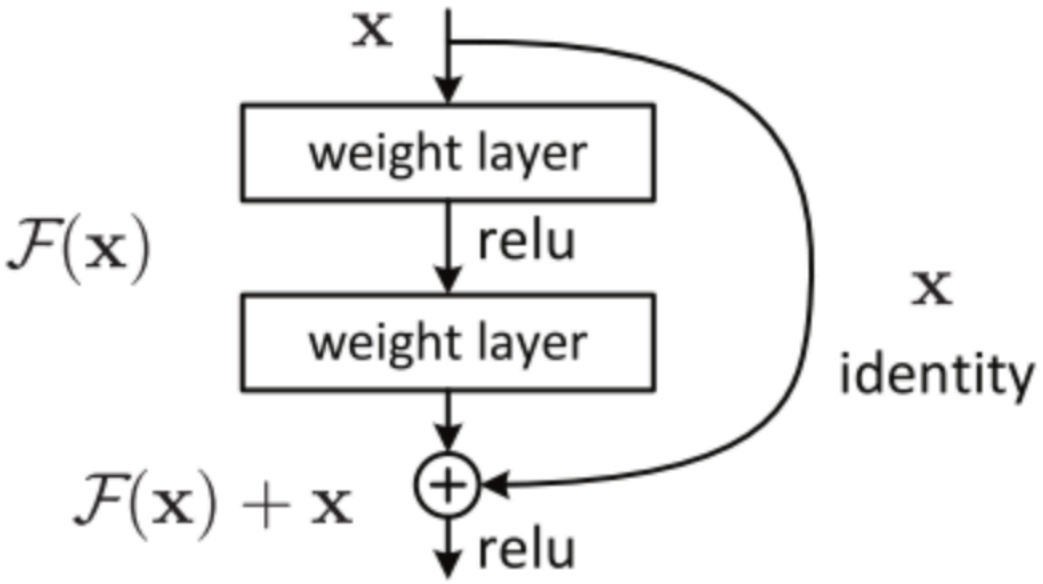
More layers are not necessary to give better performance!

?

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016

Locating parameters - History View

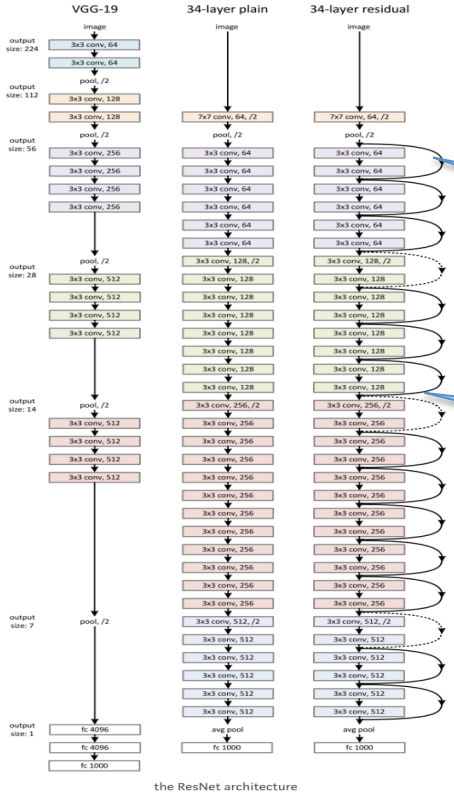
- ResNet (2015 ImageNet Competition Winner)
  - Designed so-called residual block



a residual block

Locating parameters - History View

- ResNet (2015 ImageNet Competition Winner)
  - 152 convolutional layers and no fully connected layer



Skip layer

96.6% test accuracy, 200MB Model size

Summary from History

- AlexNet → VGG-Net → GoogleNet → ResNet
  - Test accuracy are increasing to plateau (96.6%)
  - Reducing model size by removing fully connected layer
  - Still need to reduce model size but without losing performance

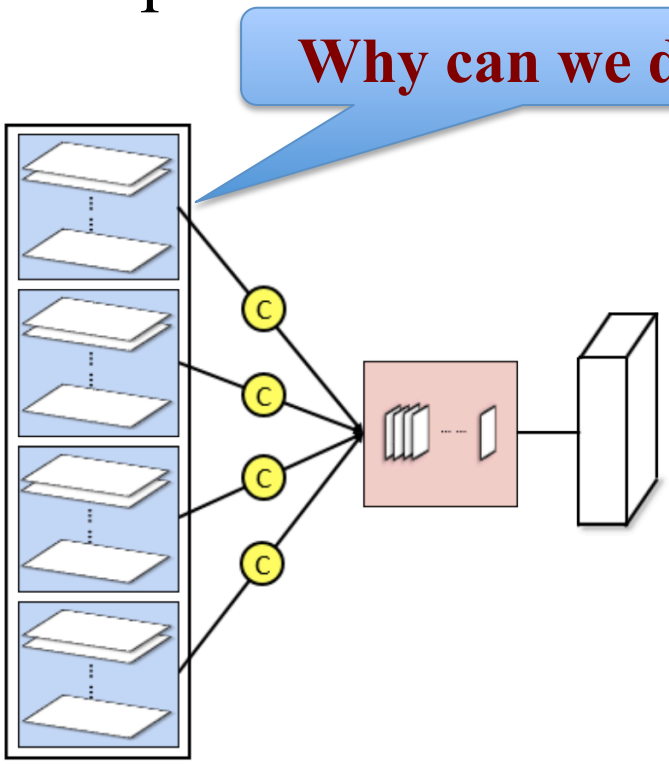
Questions: Which layer should we focus on for reducing # of parameters?

Content

- Motivation
- Where to focus on to compress
- **Compressing for Convolutional layer**
- SEP-Net (My Research Work)
- Experimental Results

Targeting on convolution layer

- Group-wise convolution



Take an example:

Input feature map: 16 x 48 x 48

Output feature map: 32 x 48 x 48

Filter size: 3 x 3

padding and stride: 1, 1

Original convolution:

16 X 11 X 11 X 32

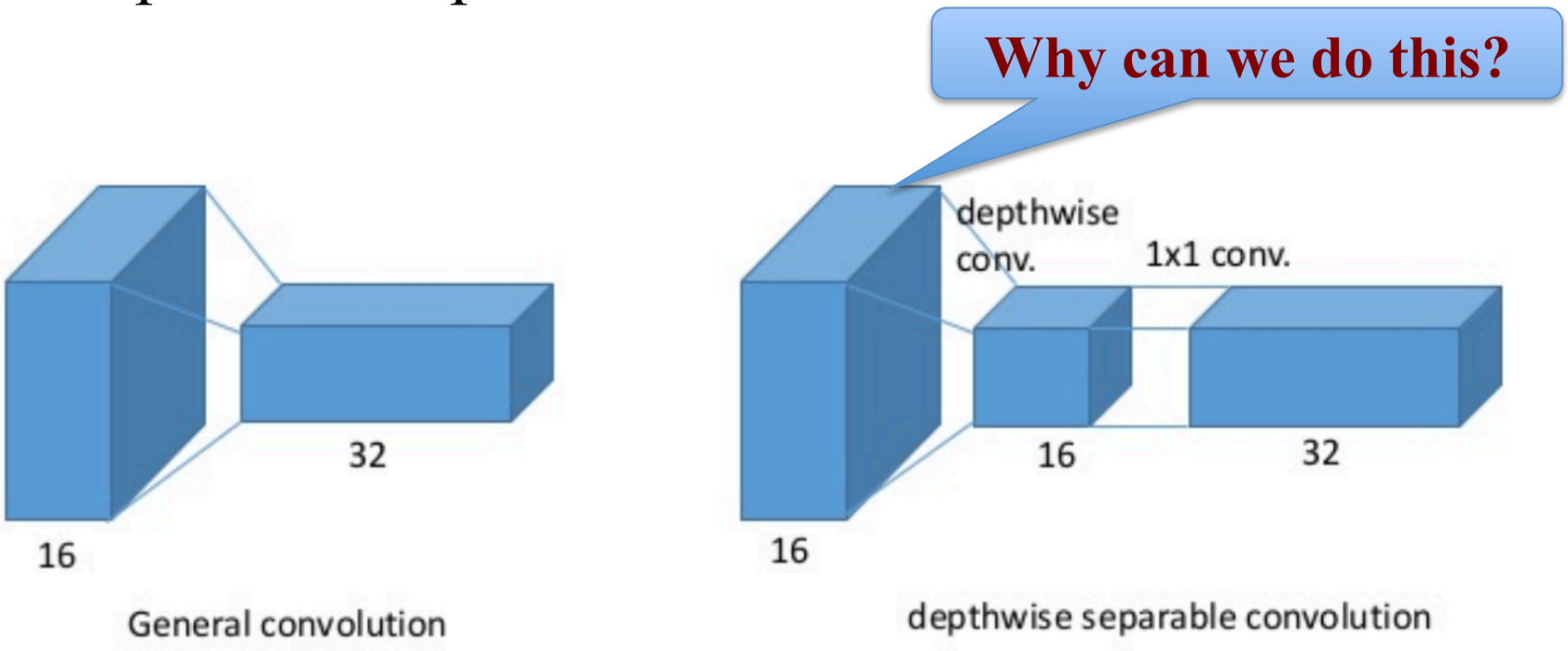
Group wise convolution: (4 group)

16/4 x 11 x 11 x 32

Reducing # of parameters in convolution k times (k group convolution)

Targeting on convolution layer

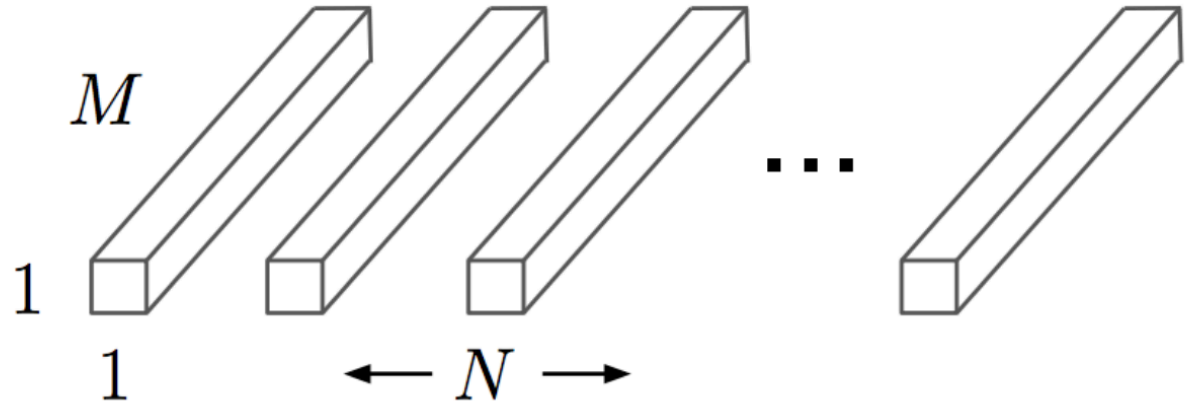
- Depth-wise separable convolution



https://www.slideshare.net/DongWonShin4/depthwise-separable-convolution

Targeting on convolution layer

- Point wise convolution

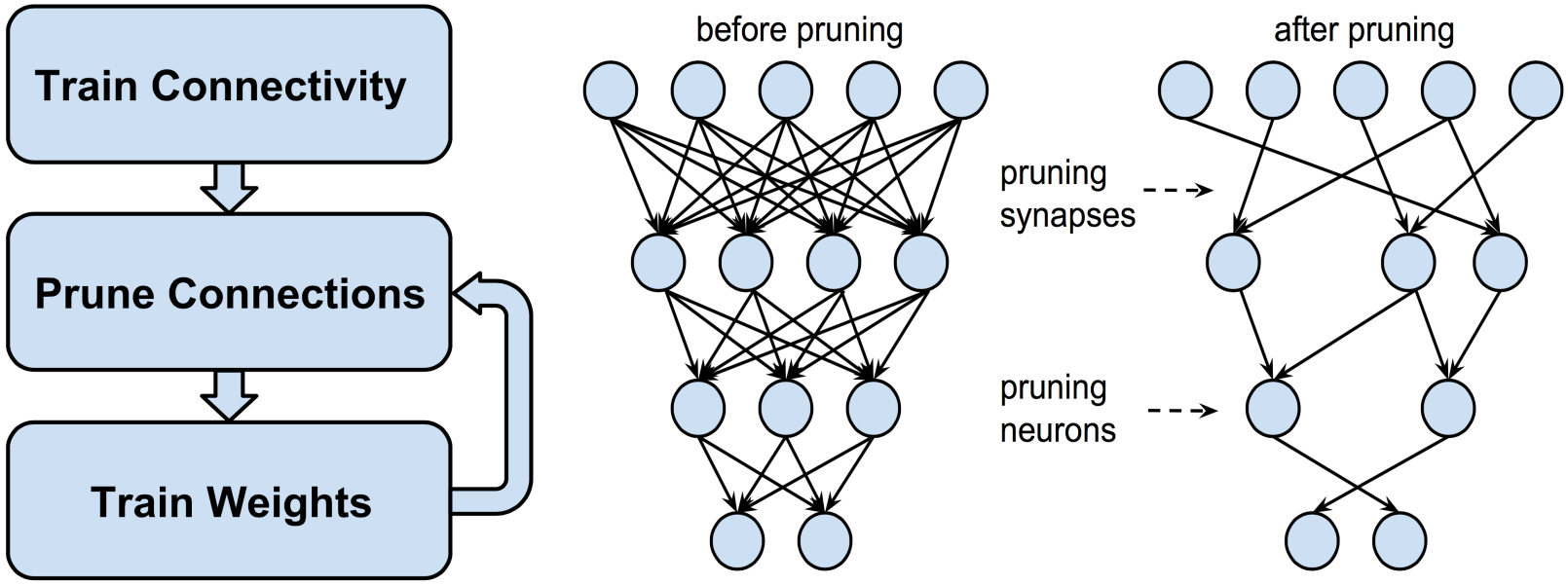


(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

1 x 1 filters serve as data transformation

Weight Pruning

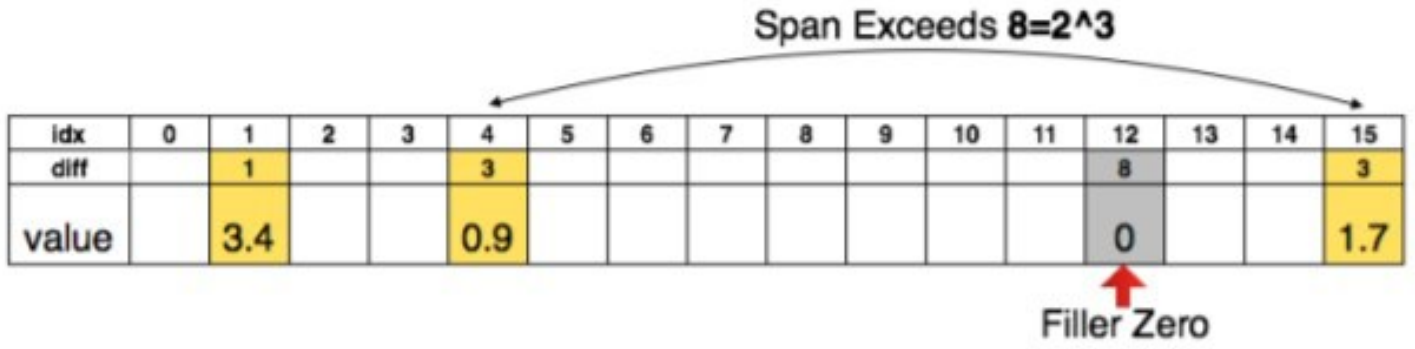
- Magnitude-based Method: Iterative Pruning + Retraining



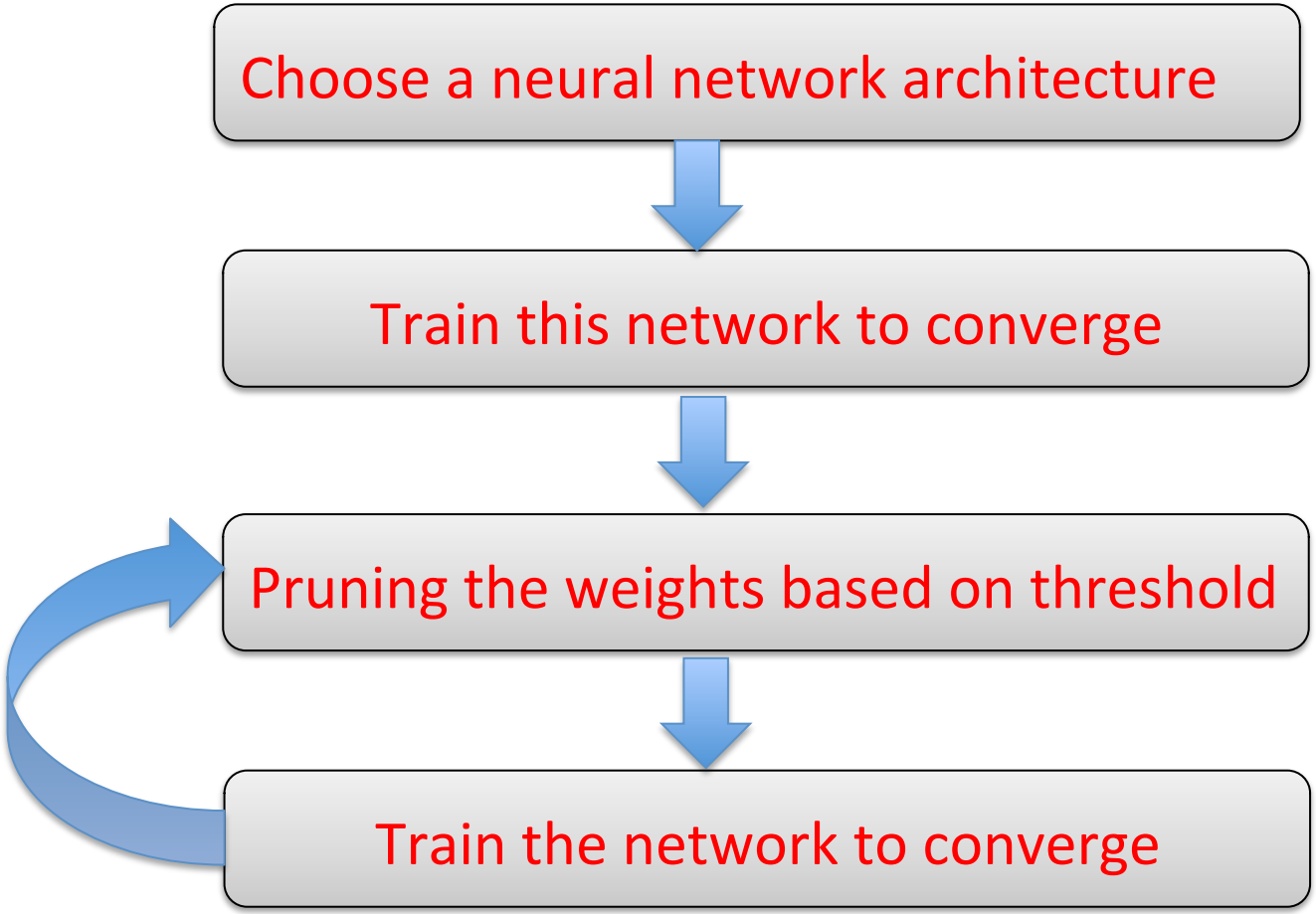
Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

Weight Pruning Implementation

- Before pruning: w_{ij} in W[i, j]
- After pruning: w_{ij} in list $L_i = \{ (j, w_{ij}) : j \geq 0, w_{ij} > \varepsilon \}$
- To reduce the memory, j can be replaced by a three-bits integer to store the next available position.
- Example： Instead of L = { (1, 3.4), (4, 0.9), (15, 1.7) }, we use M = { (1, 3.4), (3, 0.9), (8, 0), (3, 1.7) }



Weight Pruning



Weight Pruning

- Experiment Results

Network	Top-1 Error	Top-5 Error	# of Params	Compression
LeNet-300-100 Ref	1.64%	-	267K	12X
LeNet-300-100 Pruned	1.59%	-	22K	
LeNet-5 Ref	0.80%	-	431K	12X
LeNet-5 Pruned	0.77%	-	36K	
AlexNet Ref	42.78%	19.73%	61M	9X
AlexNet Pruned	42.77%	19.67%	6.7M	
VGG-16 Ref	31.50%	11.32%	138M	13X
VGG-16 Pruned	31.34%	10.88%	10.3M	

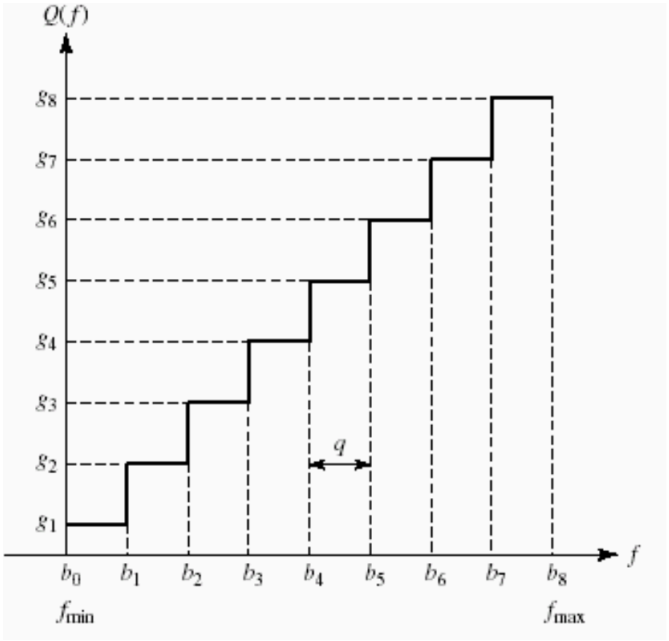
Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

Content

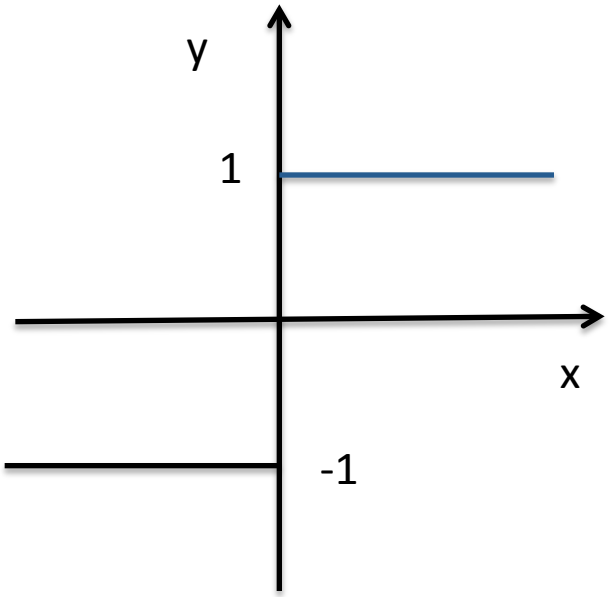
- Motivation
- Where to focus on to compress
- Compressing for Convolutional layer
- **SEP-Net (My Research Work)**
- Experimental Results

Targeting on convolution layer

- Quantization



General case



Quantization we used for simple purpose

Pattern Binarization (Quantization)

- k x k (k > 1) filters serve as spatial pattern extraction.
- 1 x 1 filters serve as data transformation.
- Reduced memory usage of model dramatically.

-0.0219	0.0408	-0.0547
-0.0855	0.0478	-0.0510
-0.0105	0.0924	-0.0126

-1	1	-1
-1	1	-1
-1	1	-1

A trained 3 x 3 filter from GoogleNet (left) and its binarized filter (right)

Pattern Binarization (Quantization)

-0.0219	0.0408	-0.0547
-0.0855	0.0478	-0.0510
-0.0105	0.0924	-0.0126

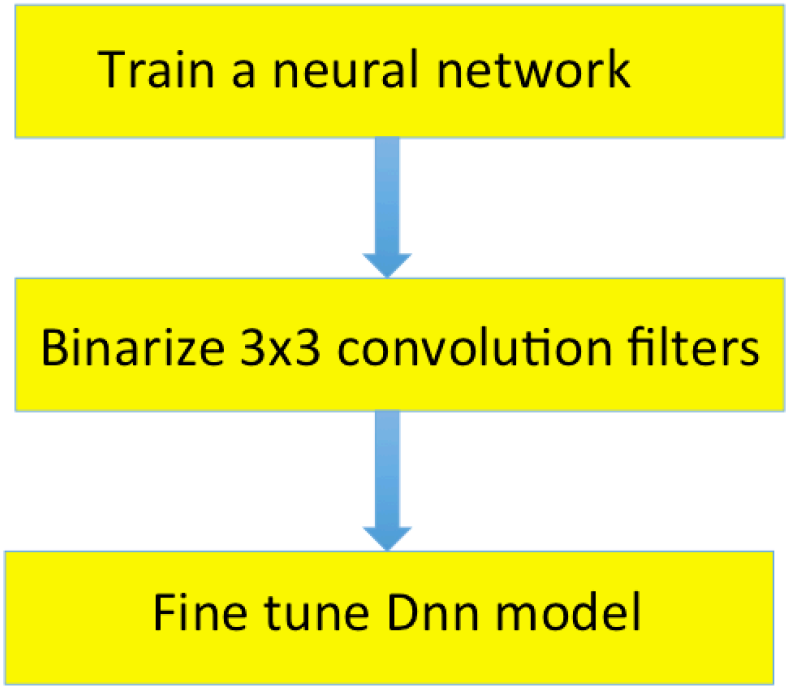
-1	1	-1
-1	1	-1
-1	1	-1

How much can we reduce memory usage by the above ?

Before: 9 parameters → 9 x 4 Bytes
After: 9 parameters → 9 bits

How to use pattern binarization?

- Easily adopted to any successful networks structures such as GoogleNet, ResNet as following procedure:

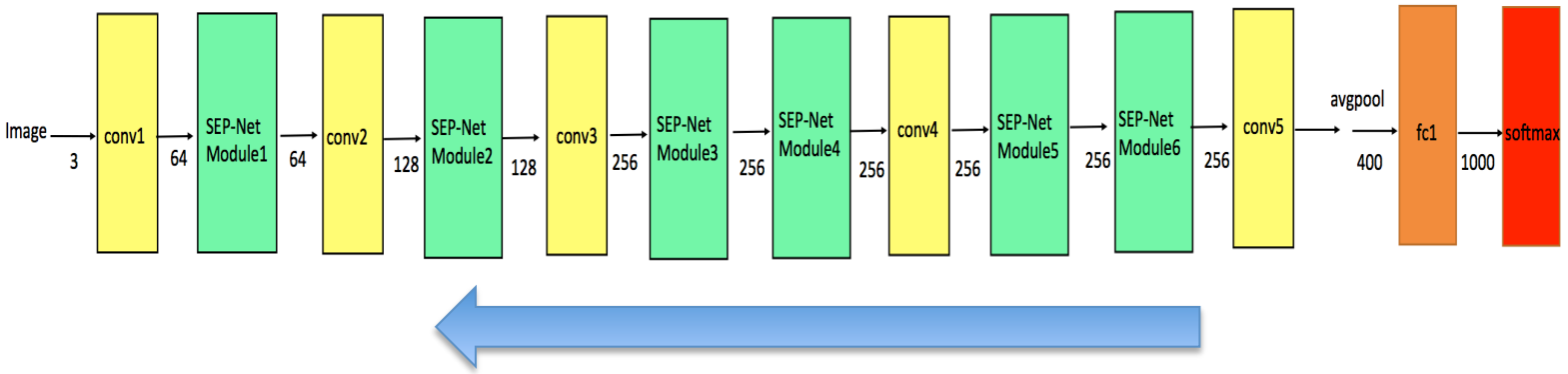


Implementation

- Keep binarization filter same during fine-tuning.

How ?

Forward to compute loss



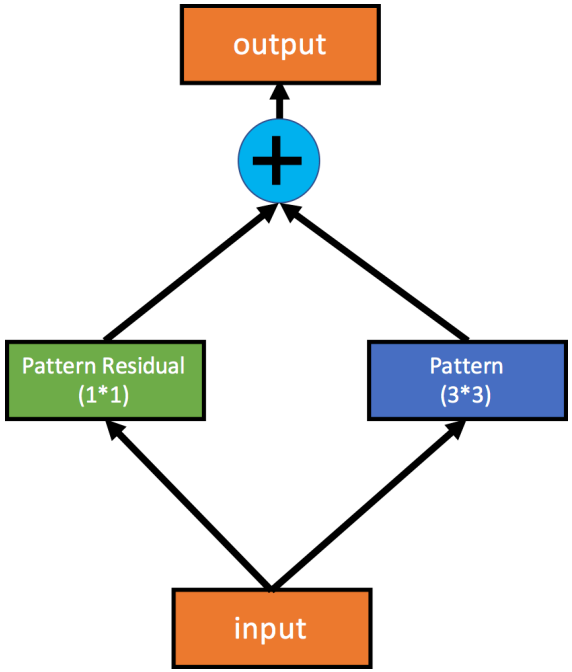
Backward to update gradient

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell(prediction, truth)$$

set learning rate as 0

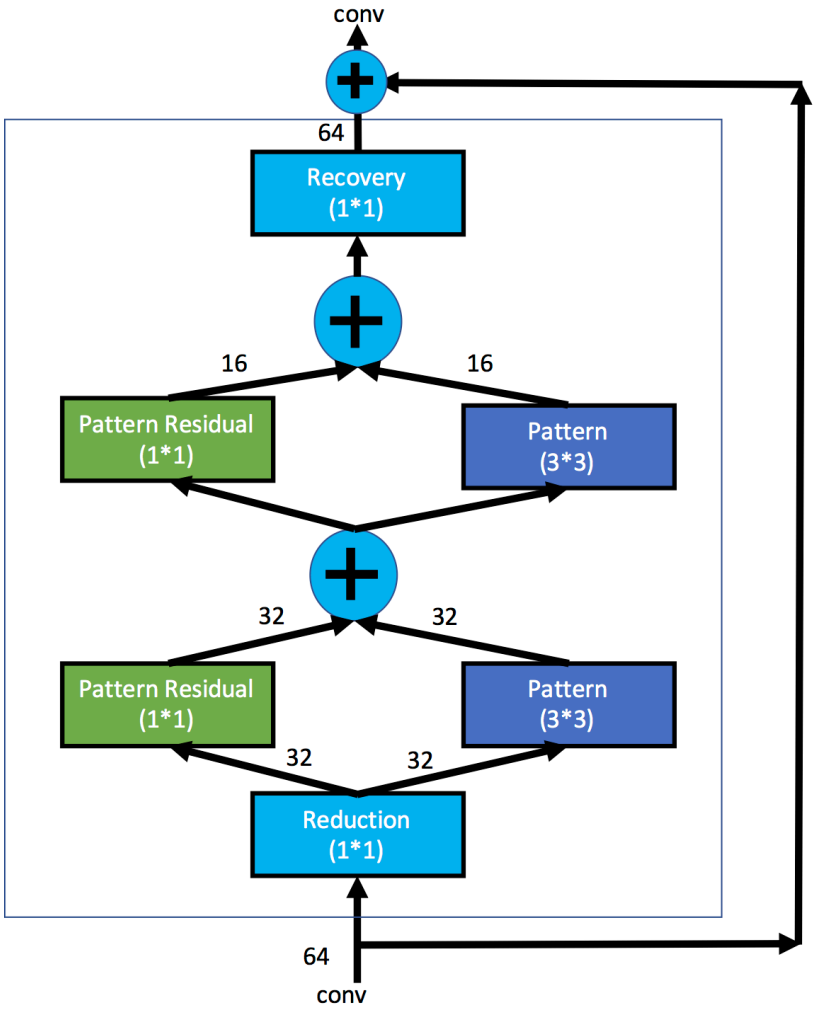
Pattern Residual Block

- Features maps from 1 x 1 and k x k convolutions are added together.
- Adding 1 x 1 convolutions offsets the change incurred by the binarization.



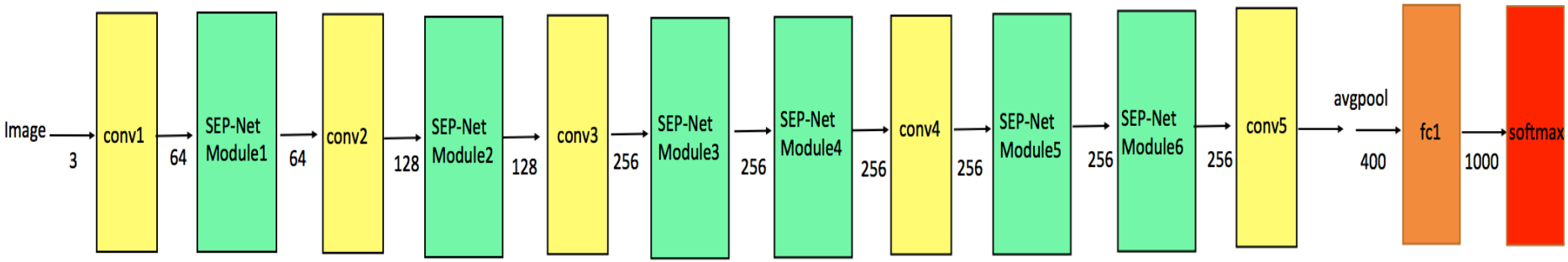
SEP-Net Module

- 1 x 1 convolution layers: dimension reduction.
- 2 PRB blocks with different output channels.
- 1 x 1 convolution layer: dimension recovery.
- Skip connection.



The Proposed SEP-Net Structure

- Proposed SEP-Net for mobile/embedded devices.
- The designed SEP-Net has 1.3M parameters (5.2MB)



Content

- Motivation
- Where to focus on to compress
- Compressing for Convolutional layer
- SEP-Net (My Research Work)
- **Experimental Results**

Experimental Results

- 1000 classes image recognition

model	Original Model		After Binarization	
	#Parameter	Top1-Top5	#Parameter	Top1-Top5
GoolgeNet	6.99M	0.6865 0.8891	4.43M	0.6797 0.8827
C-Inception	5.10M	0.6480 0.8630	2.43M	0.6400 0.8550

Experimental Results

- The designed SEP-Net

Model	Parameter Number	Size (bytes)	Top-1 Acc
MobileNet	1.3M	5.2MB	0.637
	2.6M	10.4MB	0.684
SEP-Net-R	1.3M (small)	5.2MB	0.658
	1.7M (large)	6.7MB	0.667
-	-	-	-
SqueezeNet	1.2M	4.8MB	0.604
MobileNet	1.3M	5.2MB	0.637
SEP-Net-R (small)	**1.3M**	**5.2MB**	**0.658**
SEP-Net-B (small)	1.1M	4.2MB	0.637
SEP-Net-BQ (small)	1.1M	1.3MB	0.635

SEP-Net-R: SEP-Net with raw valued weights
SEP-Net-B: SEP-Net with pattern binarization
SEP-Net-BQ: SEP=Net with pattern binarization and other weights quantized using linear quantization with 8 bits

Summary

- From AlexNet to VGG-Net, GoogleNet, ResNet, test accuracy is steadily increasing to plateau
- Reducing model size by removing FC layers
- Further Reducing model size by targeting on convolution layer through group-wise convolution, depth separable convolution, quantization, pruning etc.