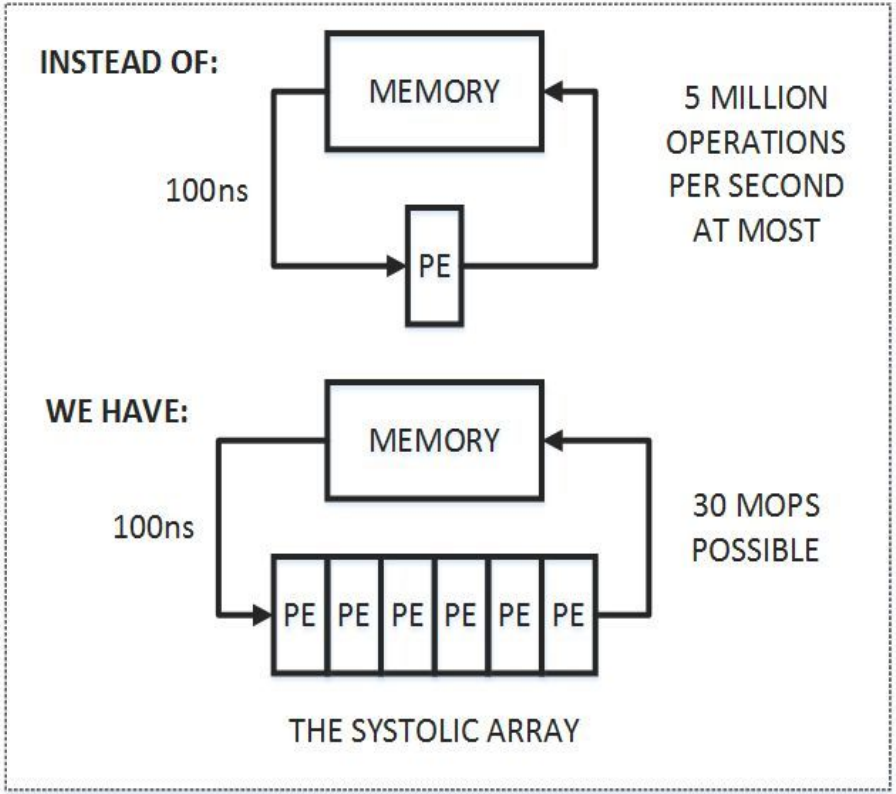
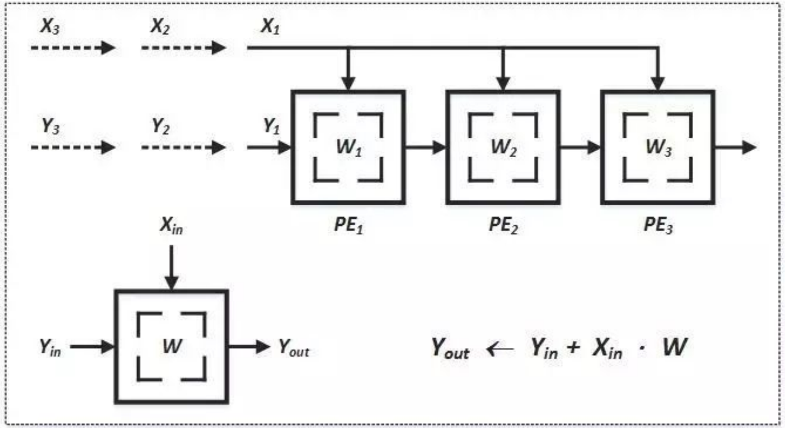
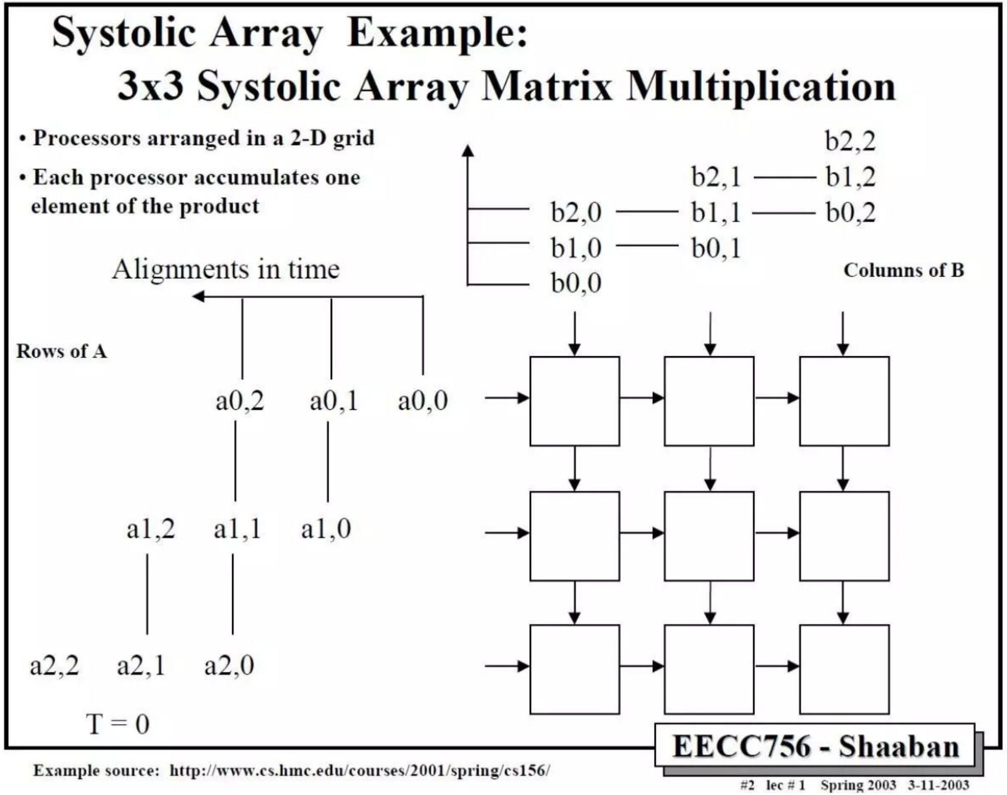
Systolic Array



Systolic Array





Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

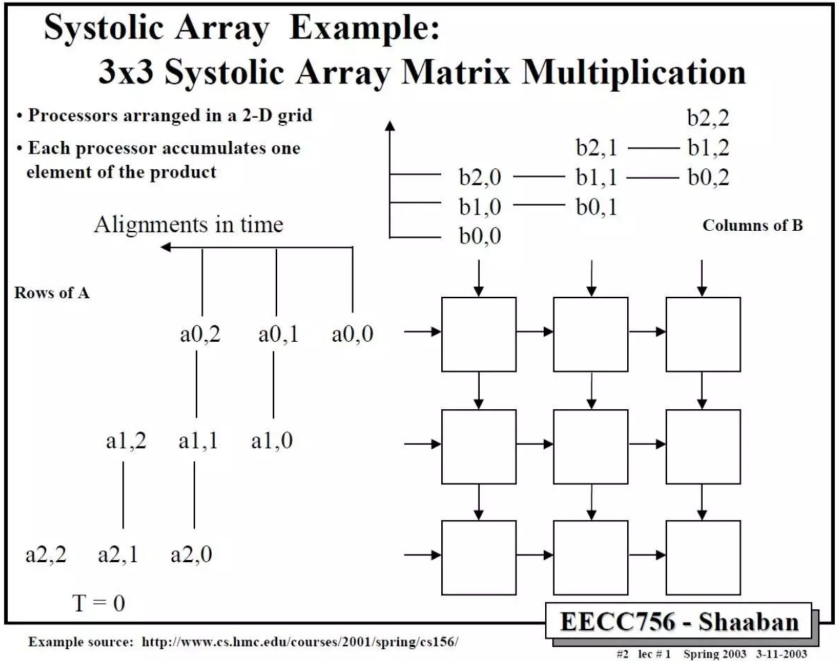
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

EECC756 - Shaaban

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

#2 lec # 1 Spring 2003 3-11-2003

Systolic Array



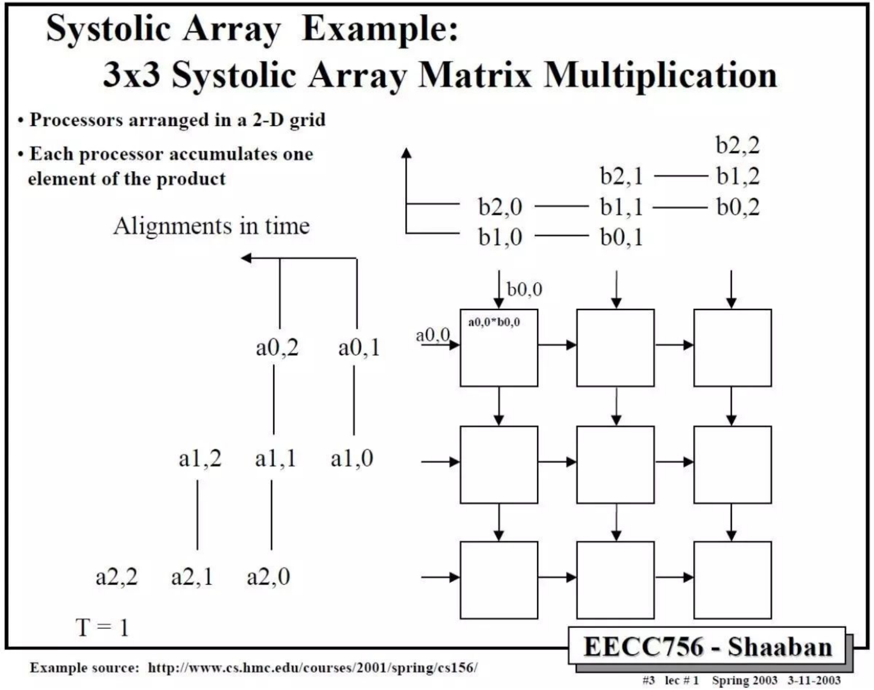
Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

EECC756 - Shaaban

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

#2 lec # 1 Spring 2003 3-11-2003



Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

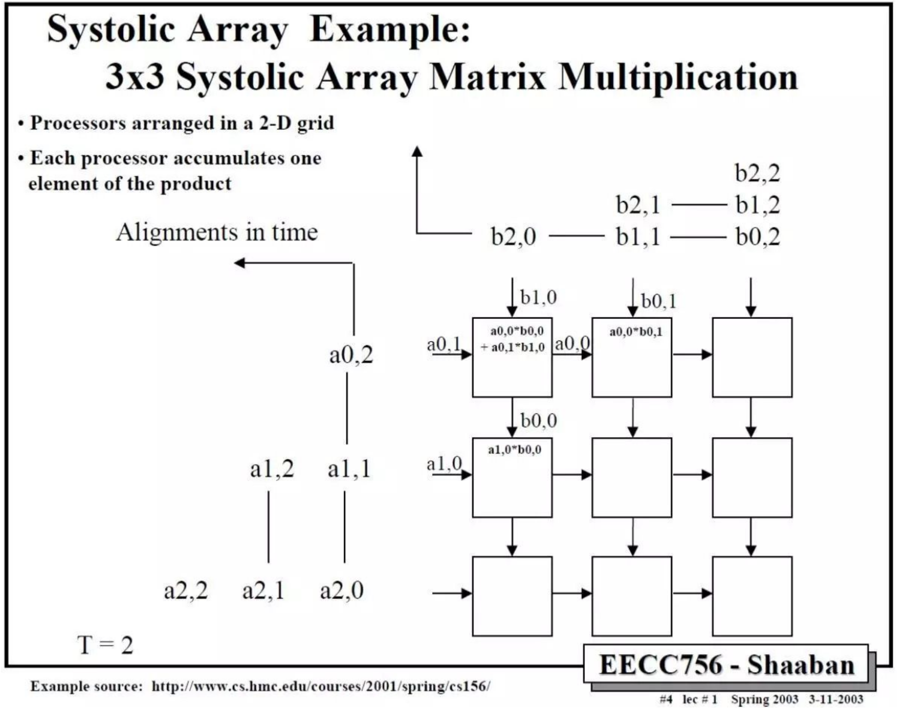
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

EECC756 - Shaaban

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

#3 lec # 1 Spring 2003 3-11-2003

Systolic Array



Systolic Array Example:

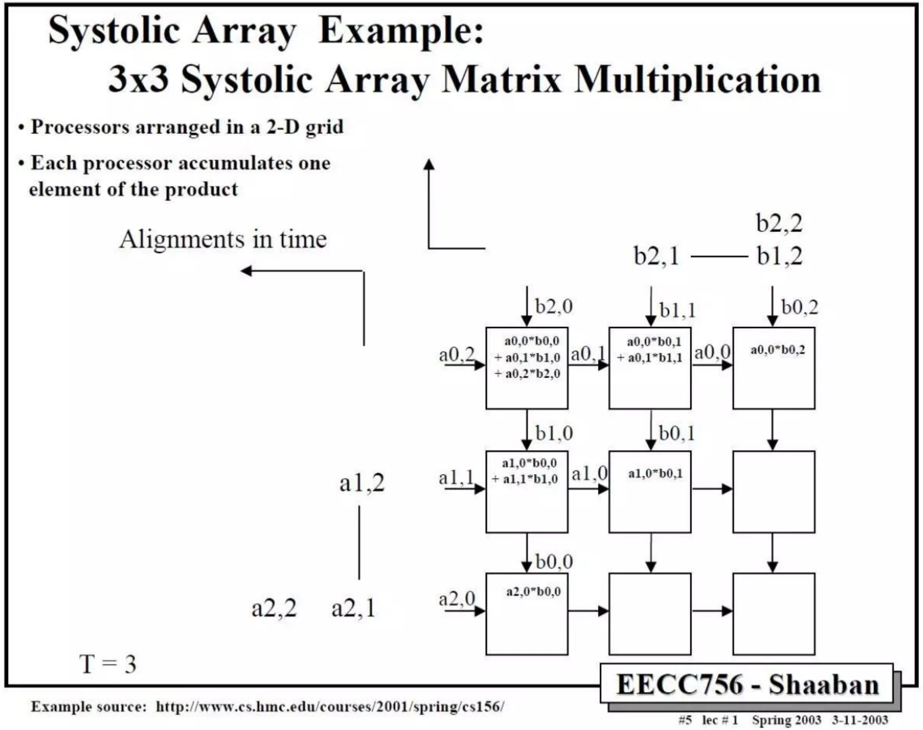
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

EECC756 - Shaaban

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

#4 lec # 1 Spring 2003 3-11-2003



Systolic Array Example:

3x3 Systolic Array Matrix Multiplication

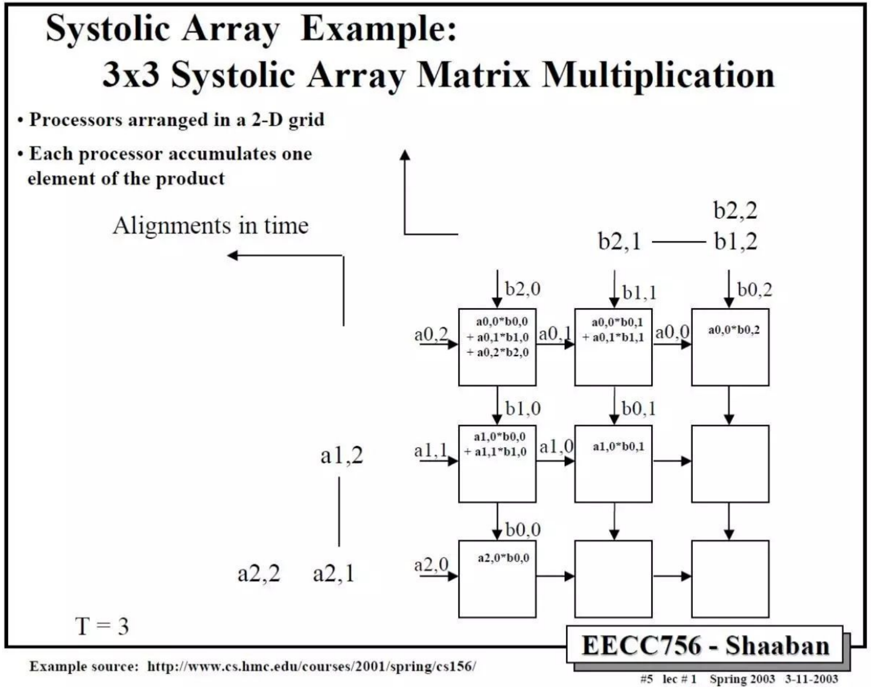
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

EECC756 - Shaaban

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

#5 lec # 1 Spring 2003 3-11-2003

Systolic Array



Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

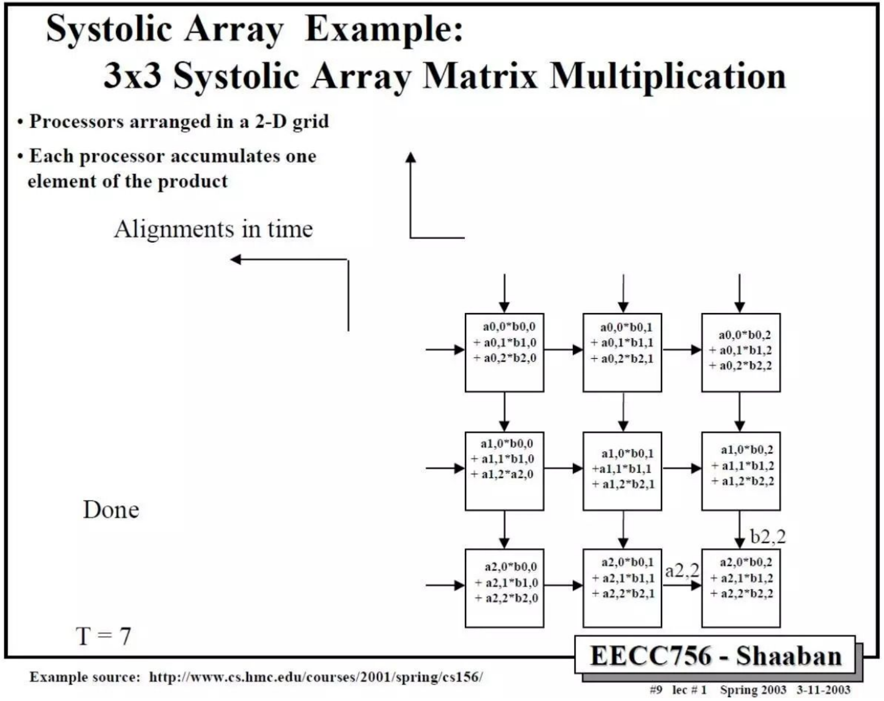
- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

T = 3

EECC756 - Shaaban

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

#5 lec # 1 Spring 2003 3-11-2003



Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

T = 7

EECC756 - Shaaban

Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

#9 lec # 1 Spring 2003 3-11-2003

1. AutoPruner: An End-to-End Trainable Filter Pruning Method for Efficient Deep Model Inference (Nanjing University)

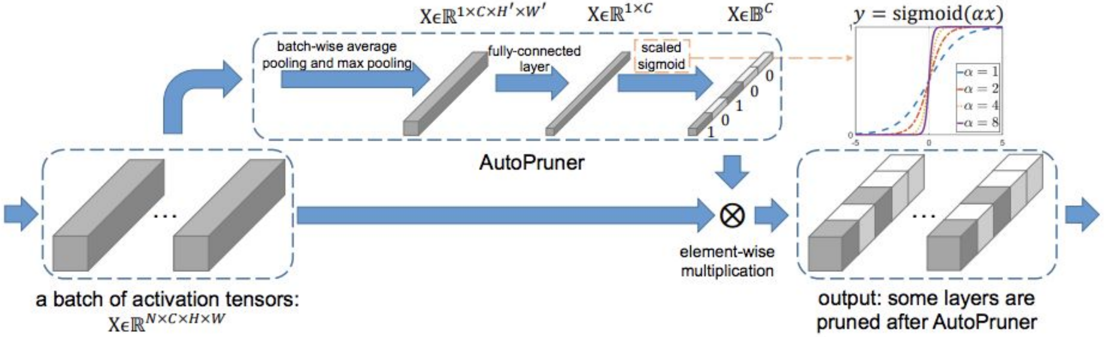
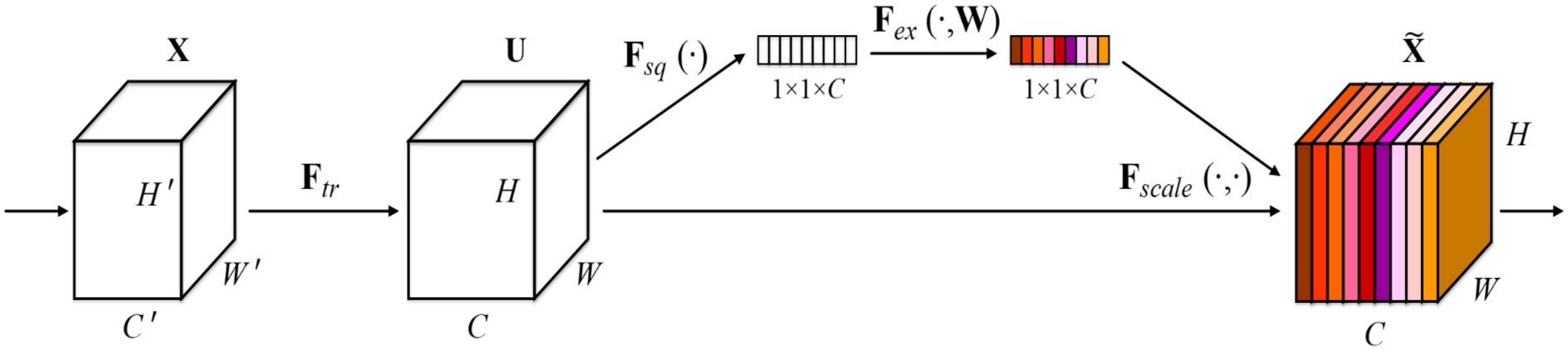

$X \epsilon \mathbb{R}^{1 \times C \times H' \times W'}$
$X \epsilon \mathbb{R}^{1 \times C}$
$X \epsilon \mathbb{B}^{C}$
$y = \text{sigmoid}(\alpha x)$
batch-wise average pooling and max pooling
fully-connected layer
scaled sigmoid
0
0
1
0
1
$\alpha = 1$
$\alpha = 2$
$\alpha = 4$
$\alpha = 8$
1
0
-5
0
5
AutoPruner
...
element-wise multiplication
a batch of activation tensors: $X \epsilon \mathbb{R}^{N \times C \times H \times W}$
output: some layers are pruned after AutoPruner

Figure 1: Framework of the proposed AutoPruner layer. Given a mini-batch of activation tensors, we use a new batch-wise average pooling and a usual max pooling to generate a single tensor. Then, this tensor is projected into a C-dimensional vector via a fully-connected layer, where C is the number of channels. Finally, a novel scaled sigmoid function is used to obtain an approximate binary output. By gradually increase the value of α in the scaled sigmoid, the output of AutoPruner gradually becomes a C-dimensional binary code. Then, all the filters and channels corresponding to the zeros index values will be pruned to obtain a smaller and faster network.

2. Squeeze-and-Excitation Networks (Momenta University of Oxford)



3 Learning Efficient Convolutional Networks through Network Slimming (Tsinghua & Intel Lab China)

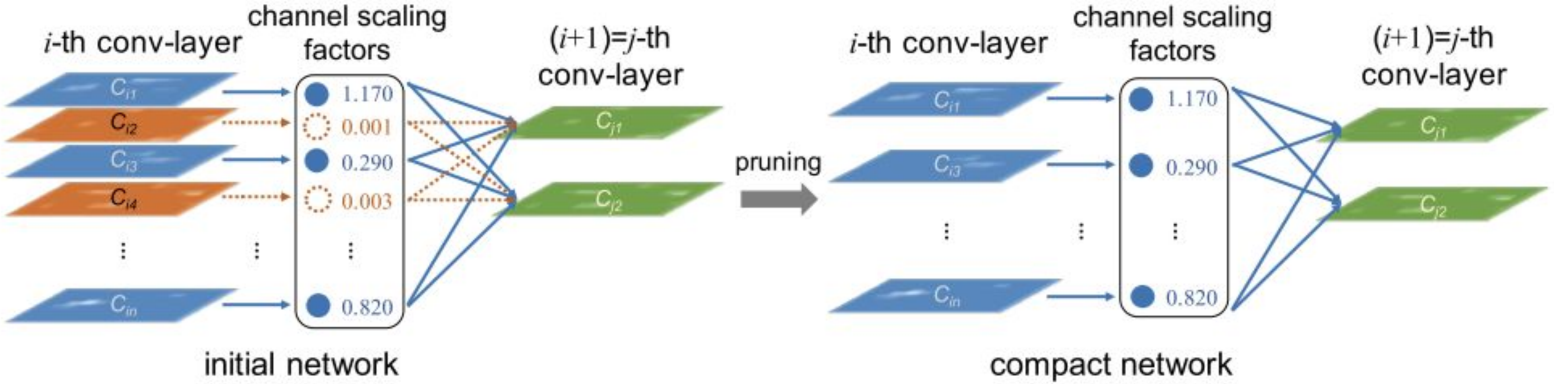
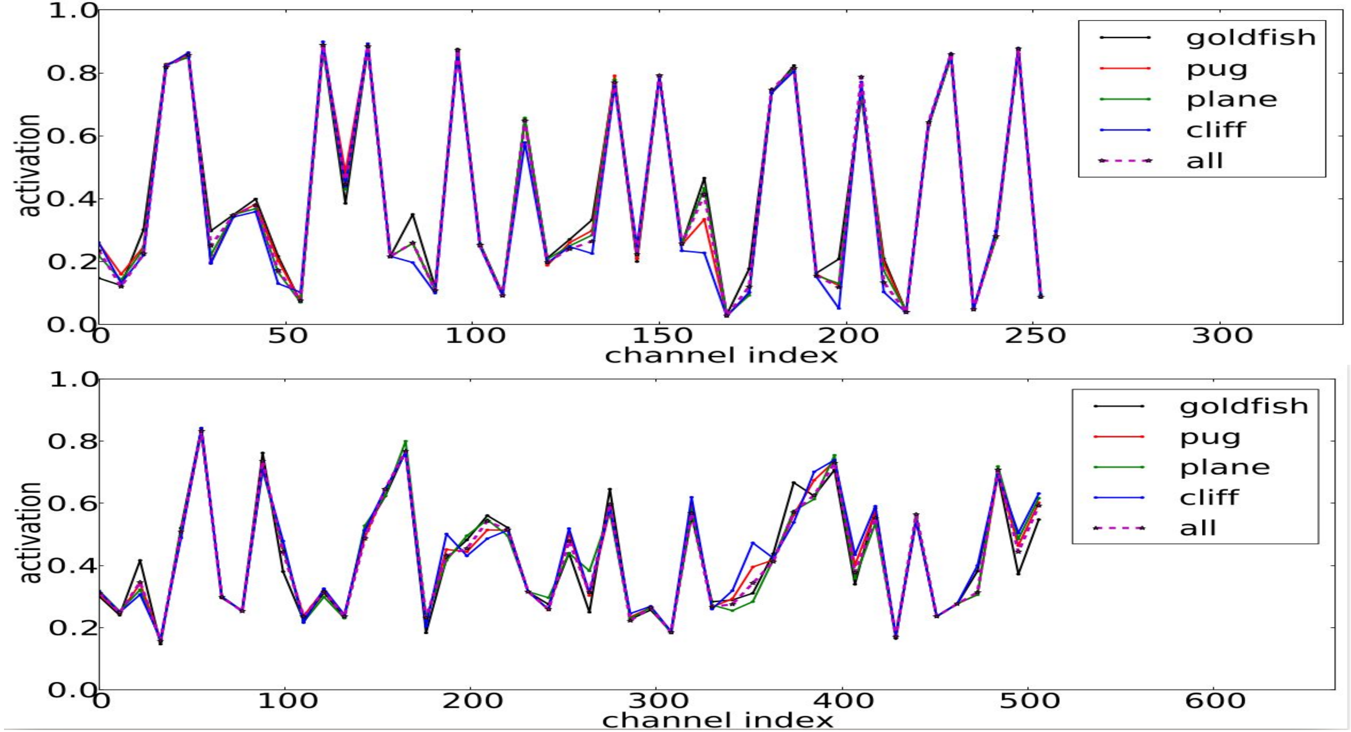


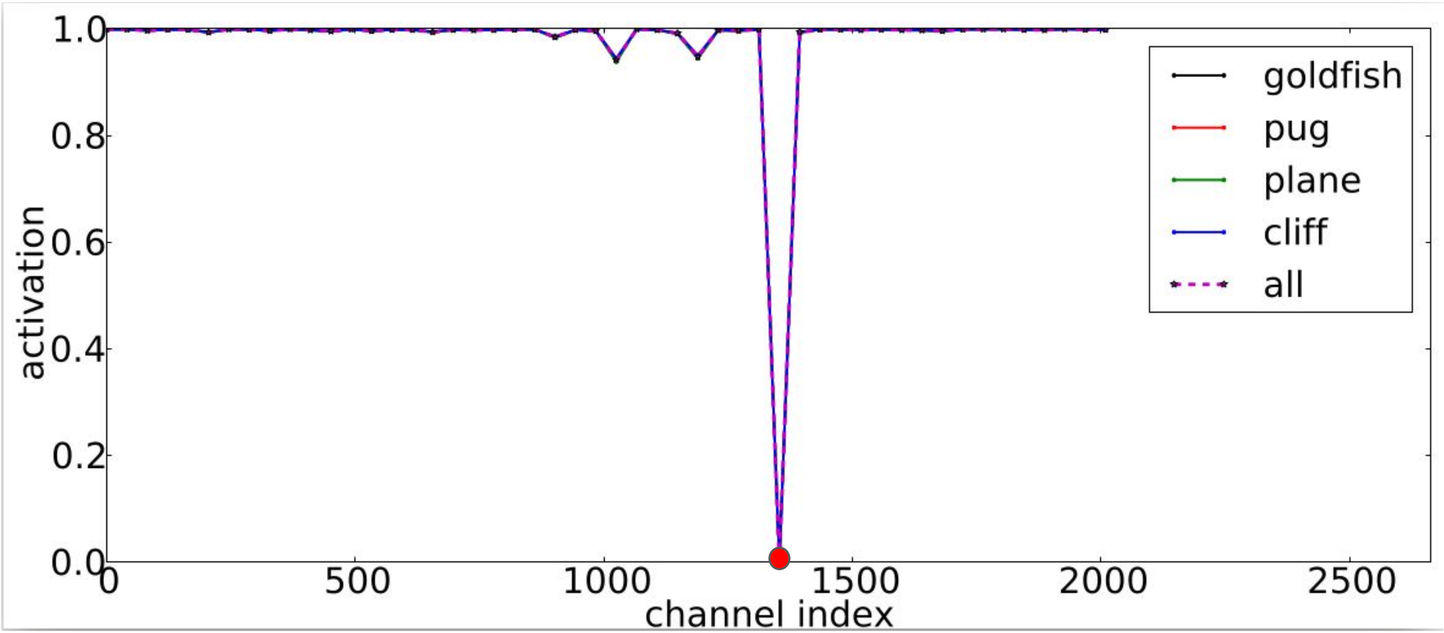
Figure 1: We associate a scaling factor (reused from a batch normalization layer) with each channel in convolutional layers. Sparsity regularization is imposed on these scaling factors during training to automatically identify unimportant channels. The channels with small scaling factor values (in orange color) will be pruned (left side). After pruning, we obtain compact models (right side), which are then fine-tuned to achieve comparable (or even higher) accuracy as normally trained full network.

First Layers



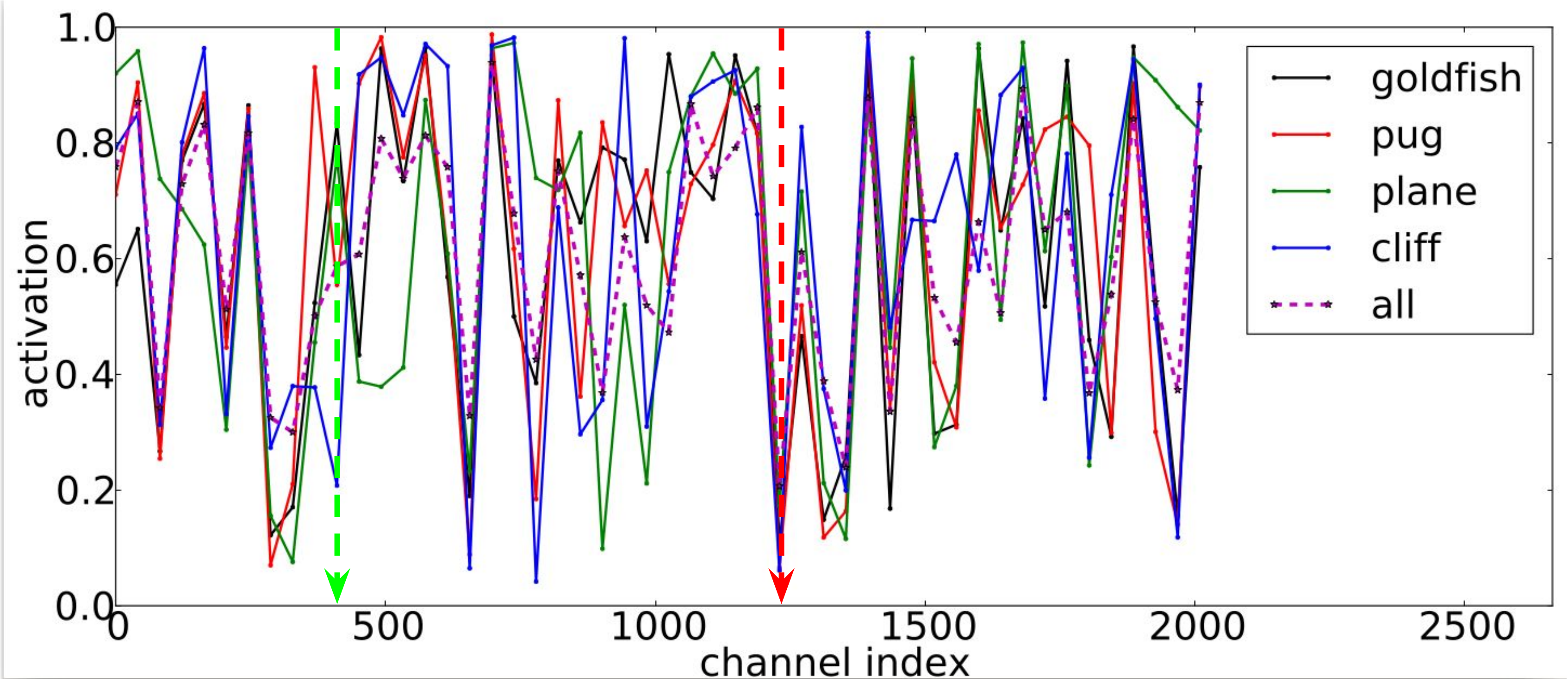
1.　不同类，差别不大　2. 有些Channel总是很接近于零，所以可以被去掉

Some Layers



像这一层，标红点的那个channel，基本上断定可以被去掉。

Last Layers



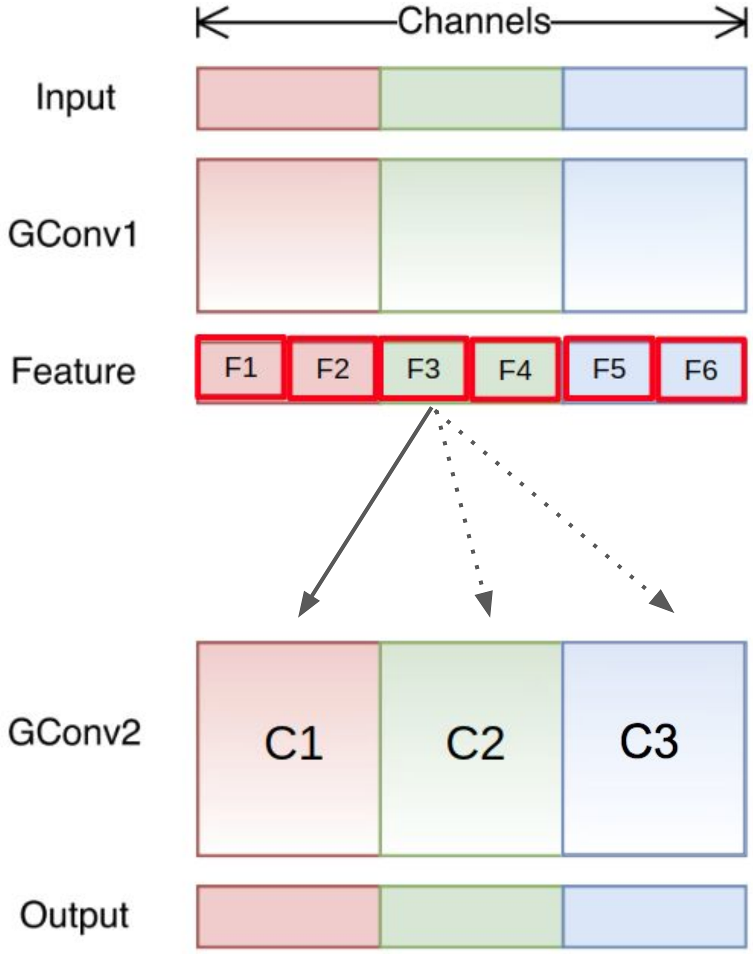
1. 标绿的那个Channel，不能别永久拿掉。（对某些类别有用，但是对某些没用）
2. 标红的那个Channel，对所有的类别都没有用，可以拿掉

SE-Net Summary

1. 可以把 SE－Net 看作是一个 dynamic pruning 的例子。feature map ---> scaling factor ---> exciting or squeeze (pruning) 。也就是说，每次 pruning 与否是由这次 forward 产生的 feature map 决定的。
2. 对于某个 Channel，所有的 sample 都对他 squeeze，那么就从 dynamic pruning 升格为 normal pruning

从 Pruning 角度理解 SE－Net

Shuffle Net



1. Shuffle Net 也是一种 Pruning：以 F3 为例，如果它被 shuffle 到 group 1，那么相当于它对于 group 2&3 的 kernel 是不可见的，也就是说：对于 group 2&3 内的 kernel来说，F3被 pruning 掉了。
2. 把这个概念进一步抽象化：普通 Pruning，如果把 F3 pruned，那么 C1、C2、C3 都不可见 F3；Shuffle Net 其实引入了一种 Part－Pruning 的概念，即：F3 对 C2、C3 是 pruning 的，但是对 C1 是 preserved 的。
3. 所以，Shuffle Net 在 *手工地* 设计 shuffle 顺序时，其实就是在手工地寻找最佳的 pruning 组合。