

Neural Networks II

Instructor: Siting Liu

School of Information Science and Technology (SIST), ShanghaiTech University

Shanghai Institute of Semiconductor Innovation (SISI)

<https://faculty.sist.shanghaitech.edu.cn/liust/EE219/EE219.html>



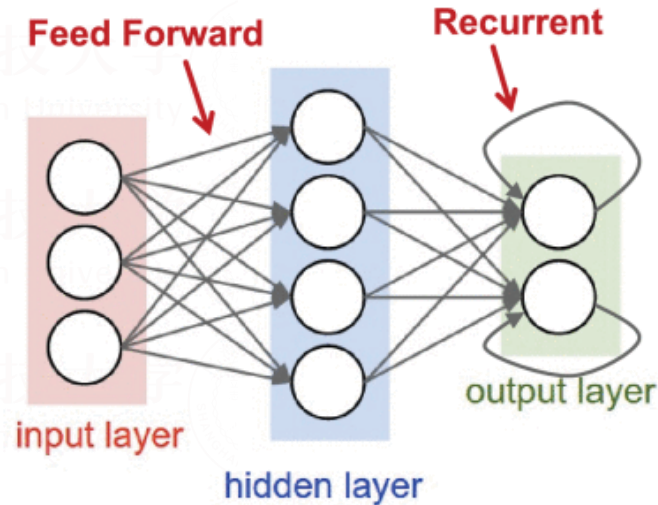
Computer Architecture - EE219
faculty.sist.shanghaitech.edu.cn

立志成才 报国裕民

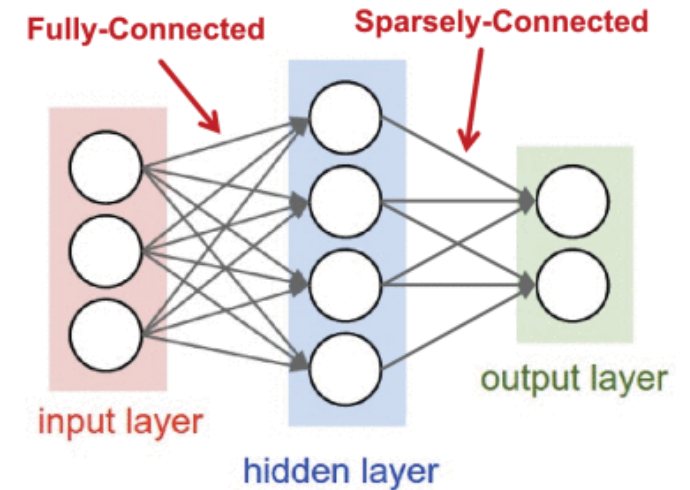


Deep Neural Networks

- Fully connected layers
 - Usually used as the last few layers
- Recurrent/feedback
- Sparsely-connected
 - E.g. Conv. layer
- Fewer parameters



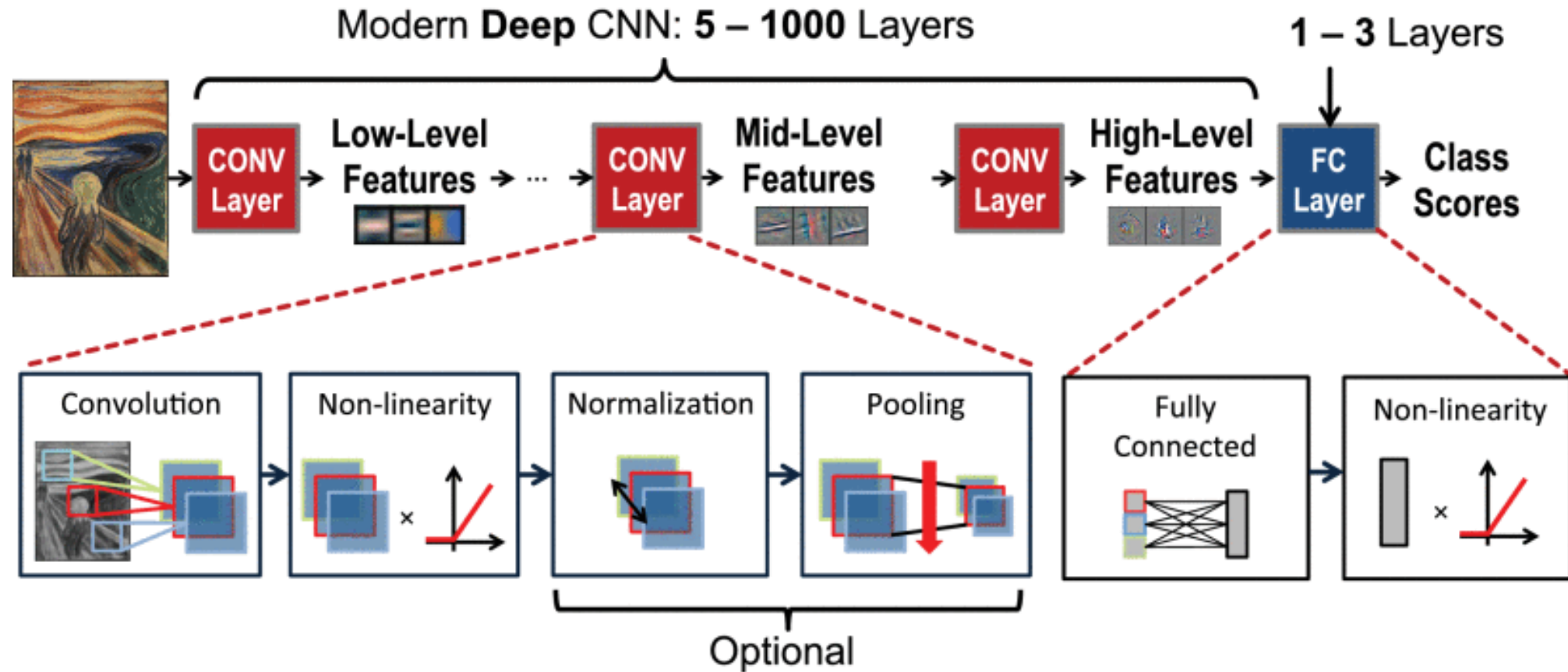
(a) Feedforward versus feedback (re-current) networks



(b) Fully connected versus sparse

V. Sze, Y. -H. Chen, T. -J. Yang and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," in Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, Dec. 2017.

Deep Neural Network Example

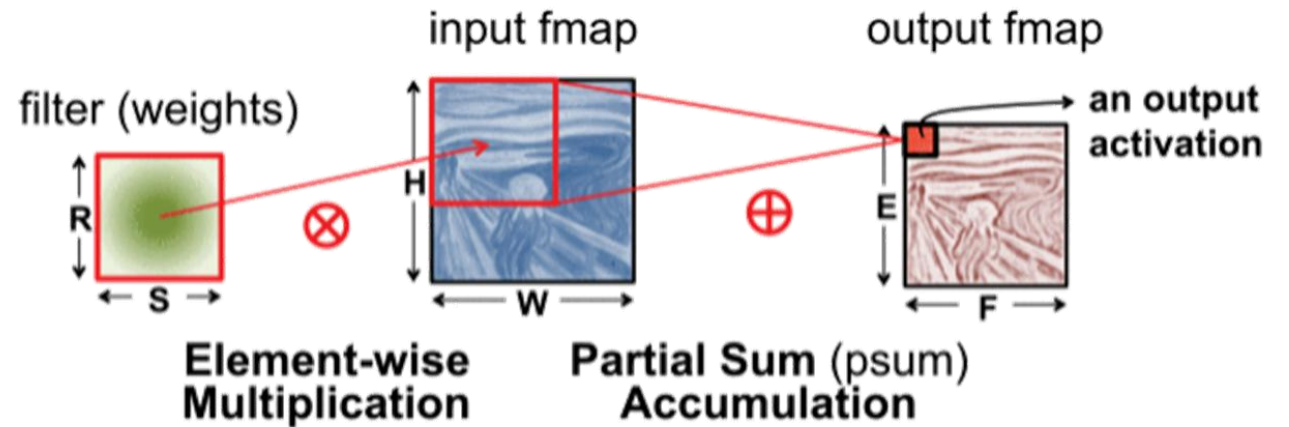


V. Sze, Y. -H. Chen, T. -J. Yang and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," in Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, Dec. 2017.

Convolution

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)`

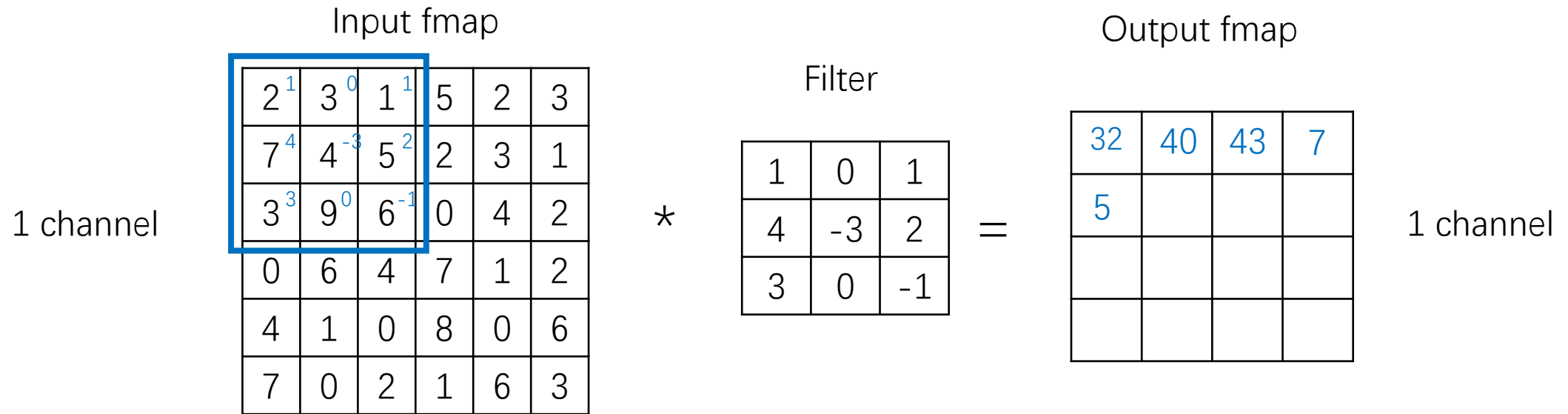
- Conv. (sparsely-connected layers with shared weights)
 - **Feature maps**
 - Channels
 - **Filters/kernels (weights)**



(a) 2-D convolution in traditional image processing

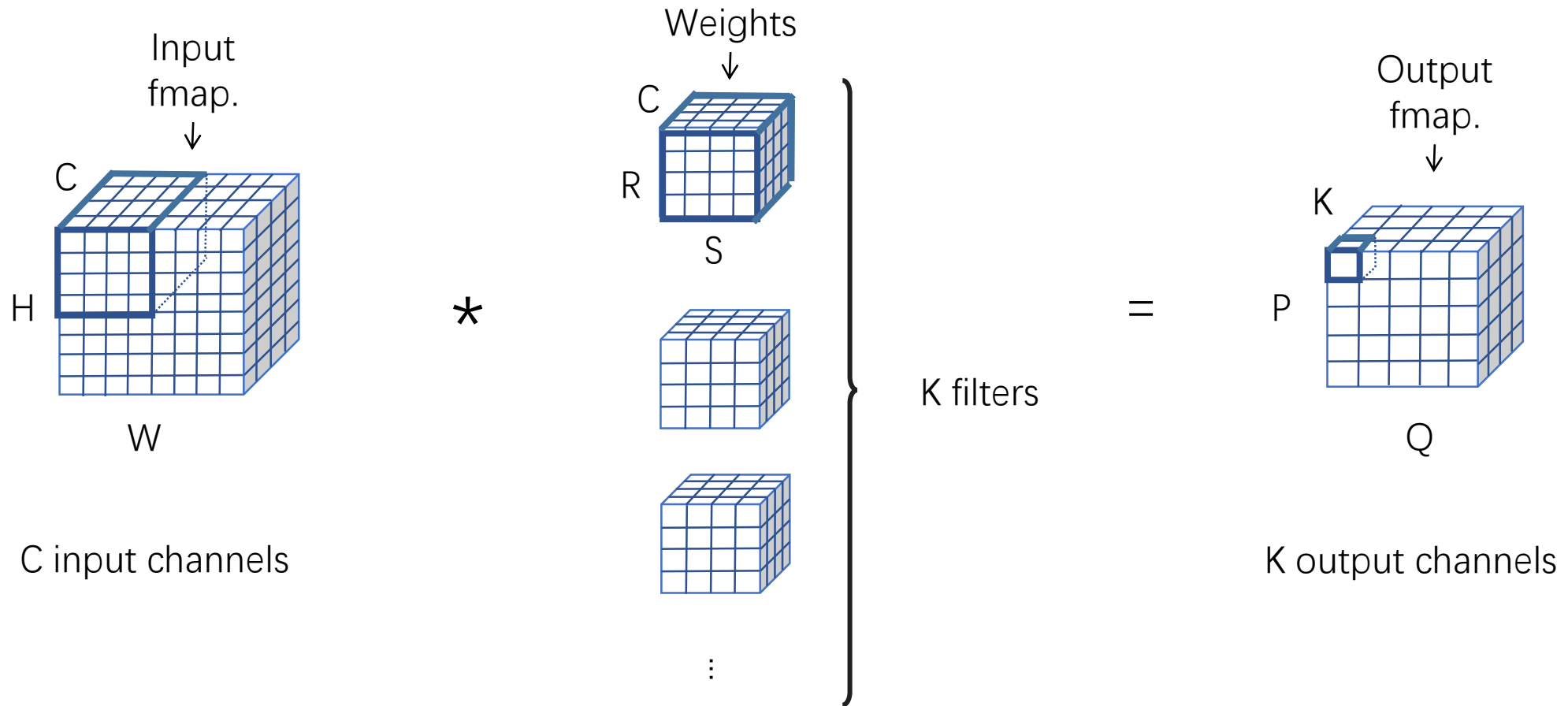
Basic 2D Convolution

- Recognize objects by computing the similarity between the feature map and the filter



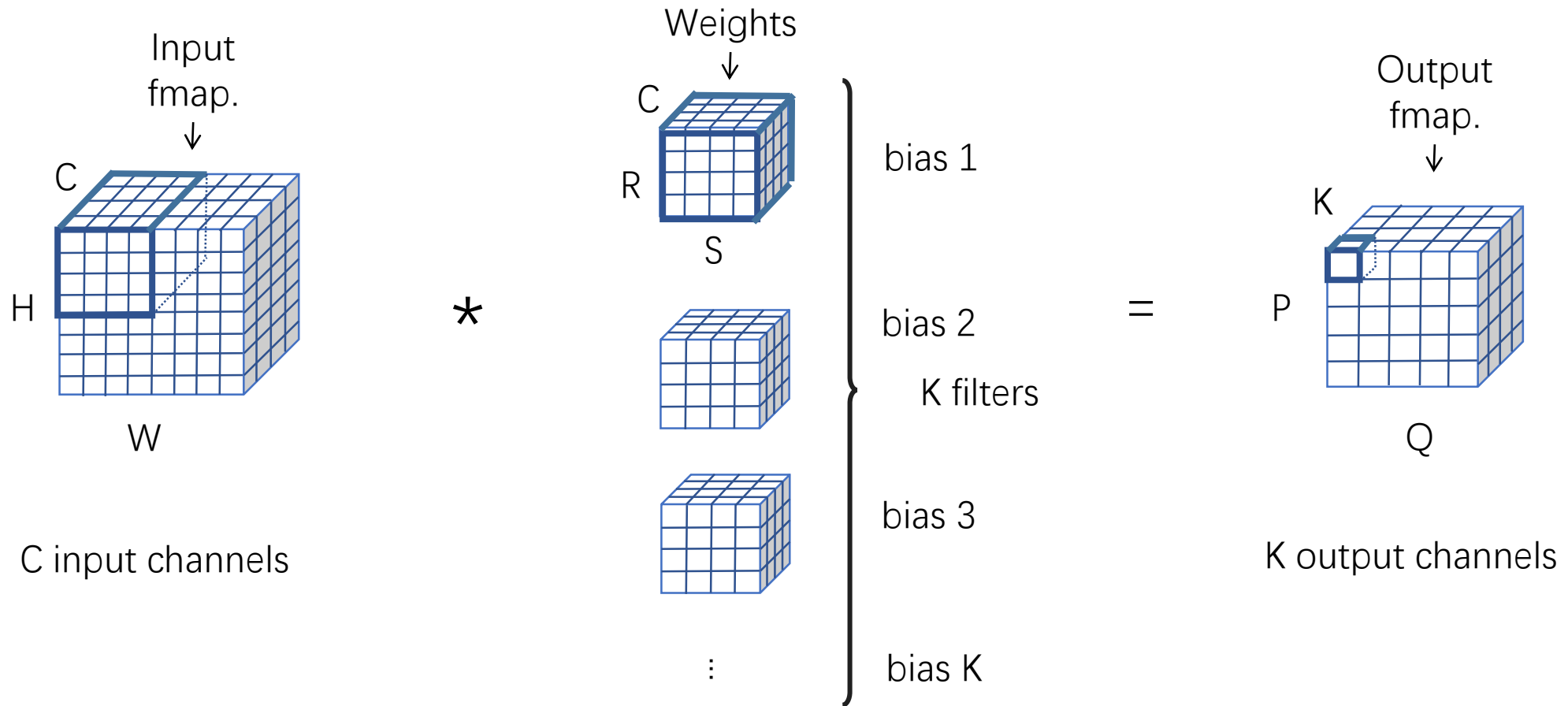
Convolution with 3D Inputs

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)`



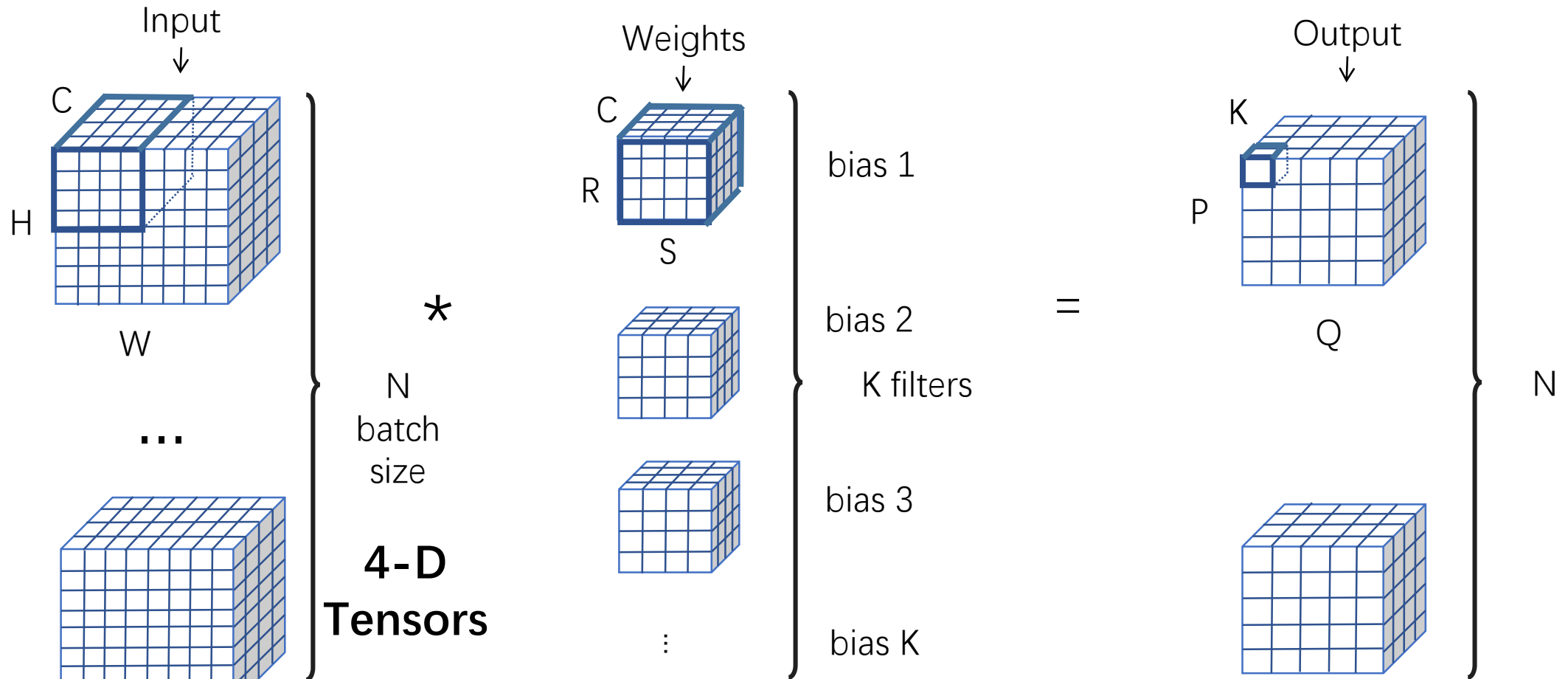
Convolution with Biases

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)`



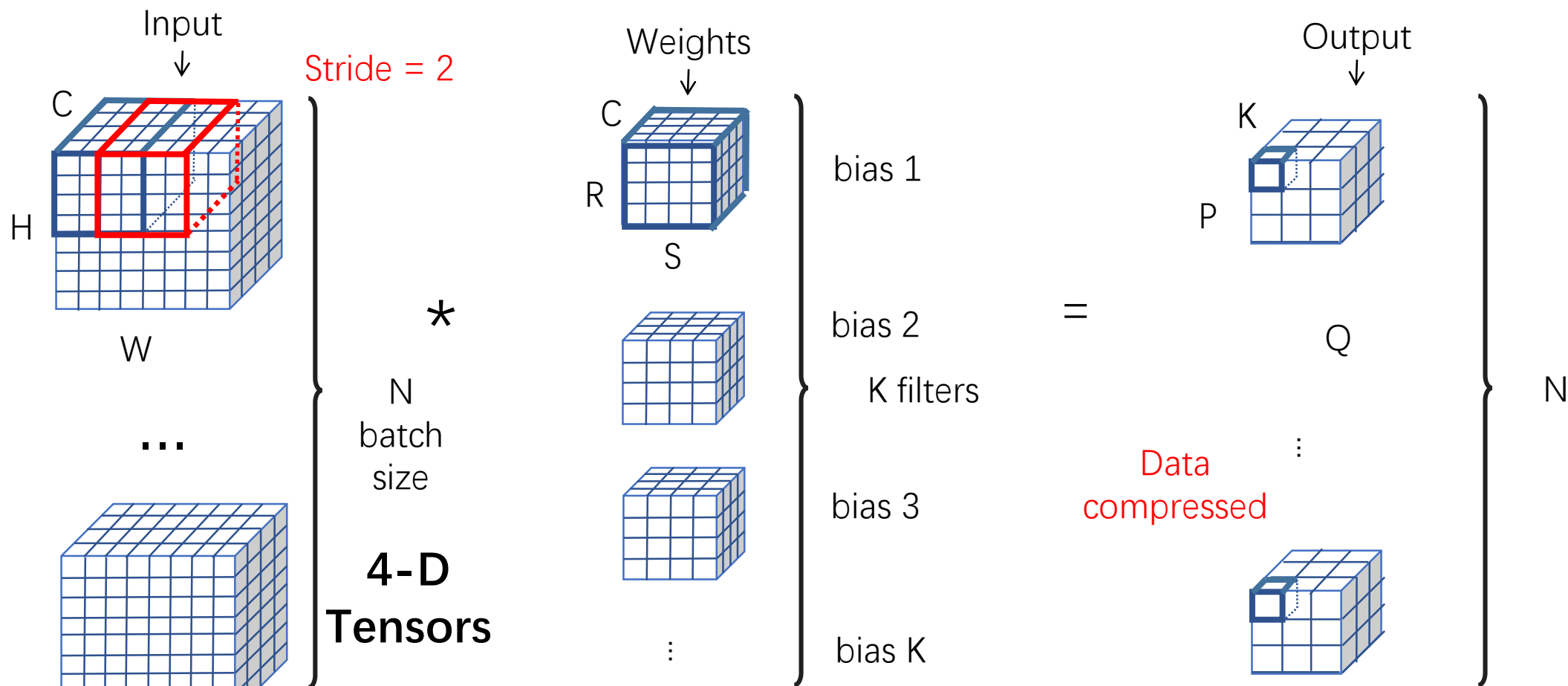
Batched Convolution

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)`



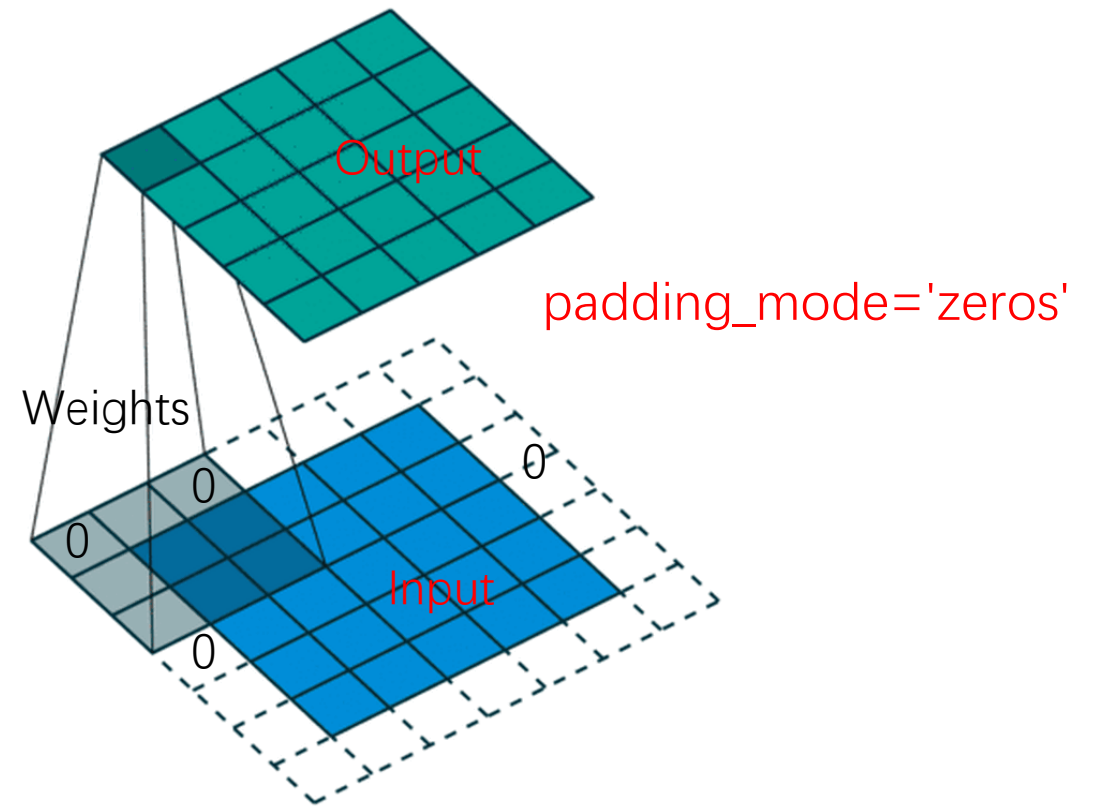
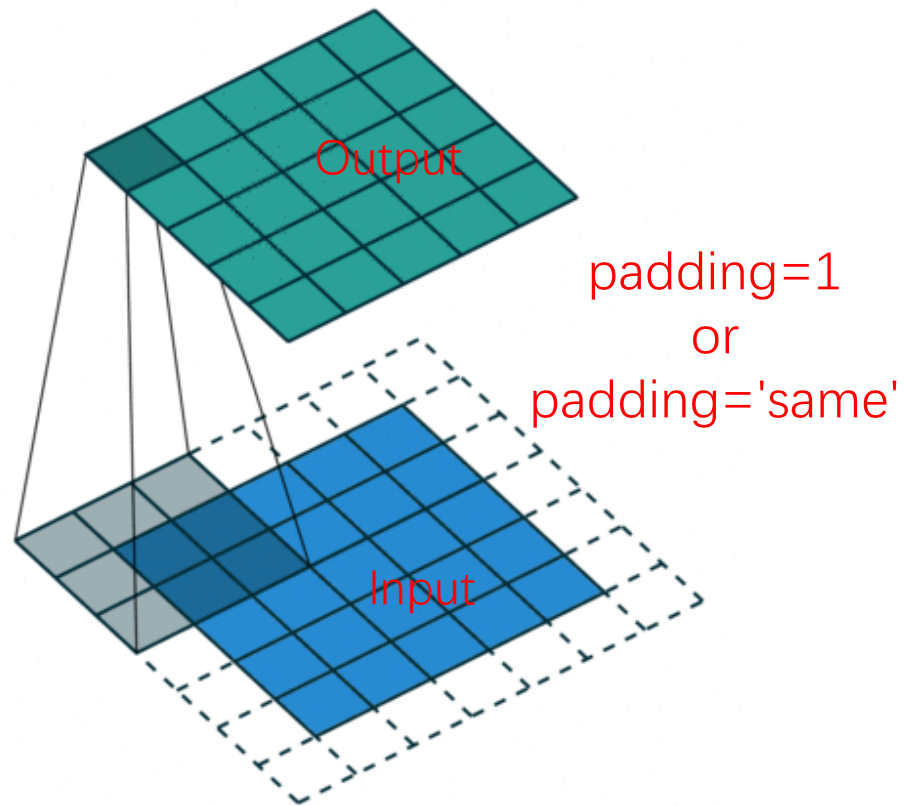
Convolution with Strides

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=2, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)`



Convolution with Padding

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=2, padding=1('same'), dilation=1, groups=1,  
bias=True, padding_mode='zeros', device=None, dtype=None)
```



Convolution with Padding (Cont'd)

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=2, padding=1('same'), dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)`

padding_mode='zeros'

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

padding_mode='reflect'

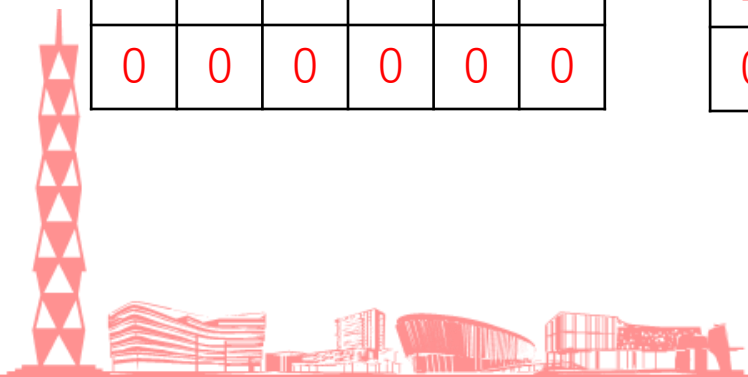
4	6	4	7	1	7
6	9	6	0	4	0
4	6	4	7	1	7
0	1	0	8	0	8
2	0	2	1	6	1
0	1	0	8	0	8

padding_mode='replicate'

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

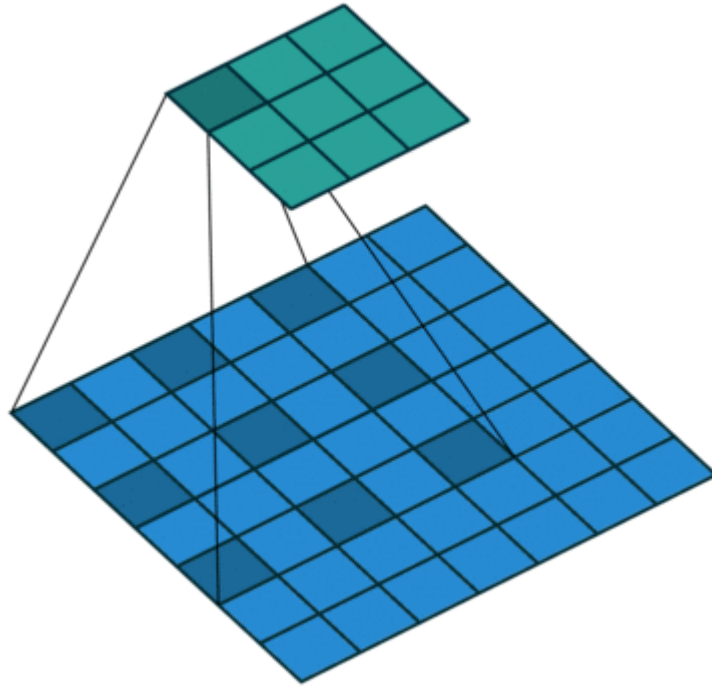
padding_mode='circular'

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



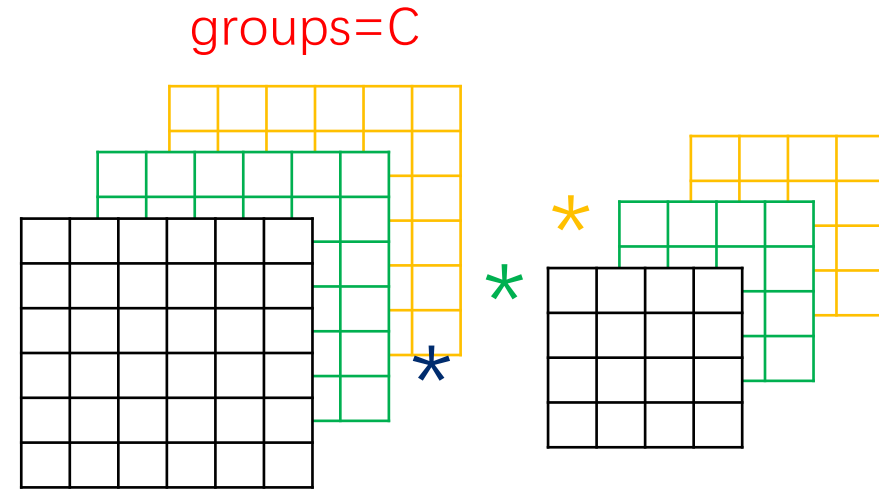
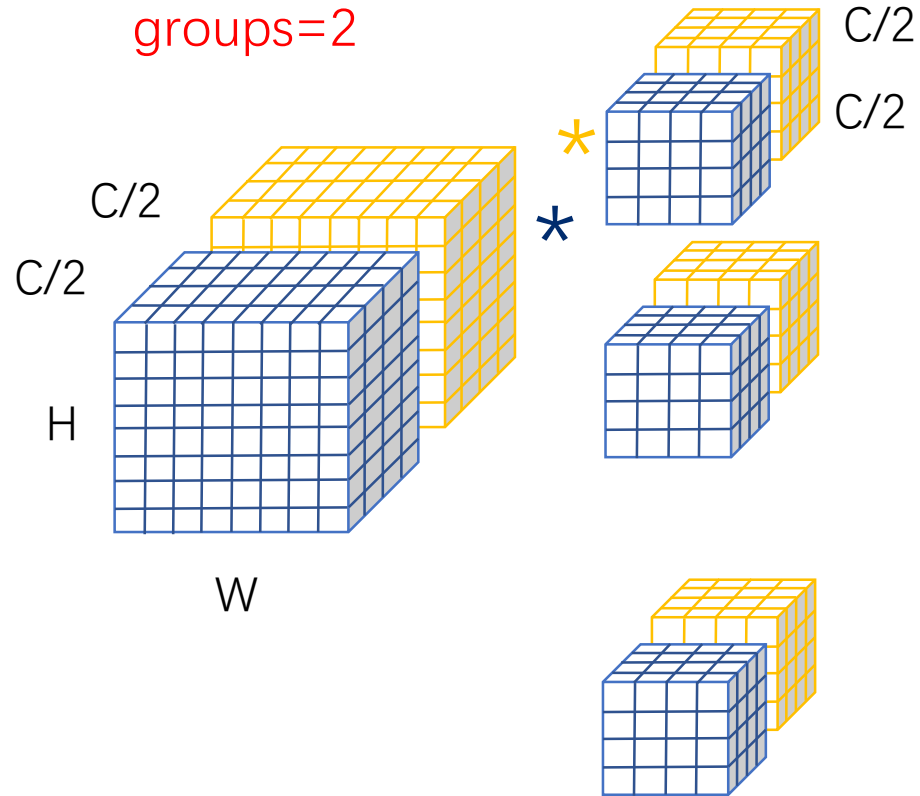
Convolution with Dilation

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=2, padding=0, dilation=2, groups=1,  
bias=True, padding_mode='zeros', device=None, dtype=None)
```



Grouped Convolution

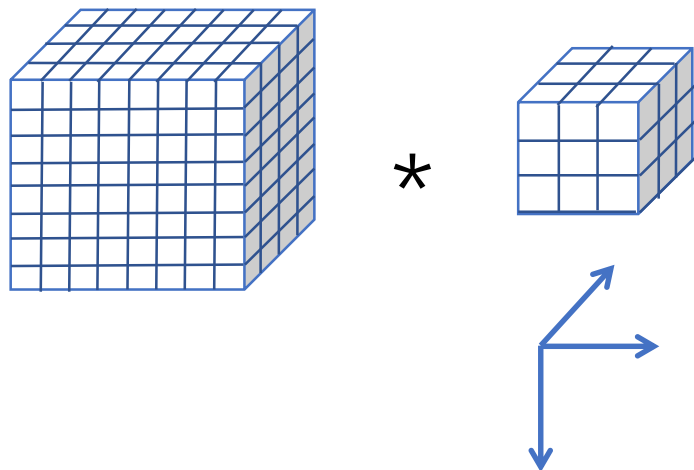
`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=2, padding=0, dilation=2, groups=N, bias=True, padding_mode='zeros', device=None, dtype=None)`



Depthwise separable layer,
widely used in light/efficient CNNs

3D Convolution

- Conv3d



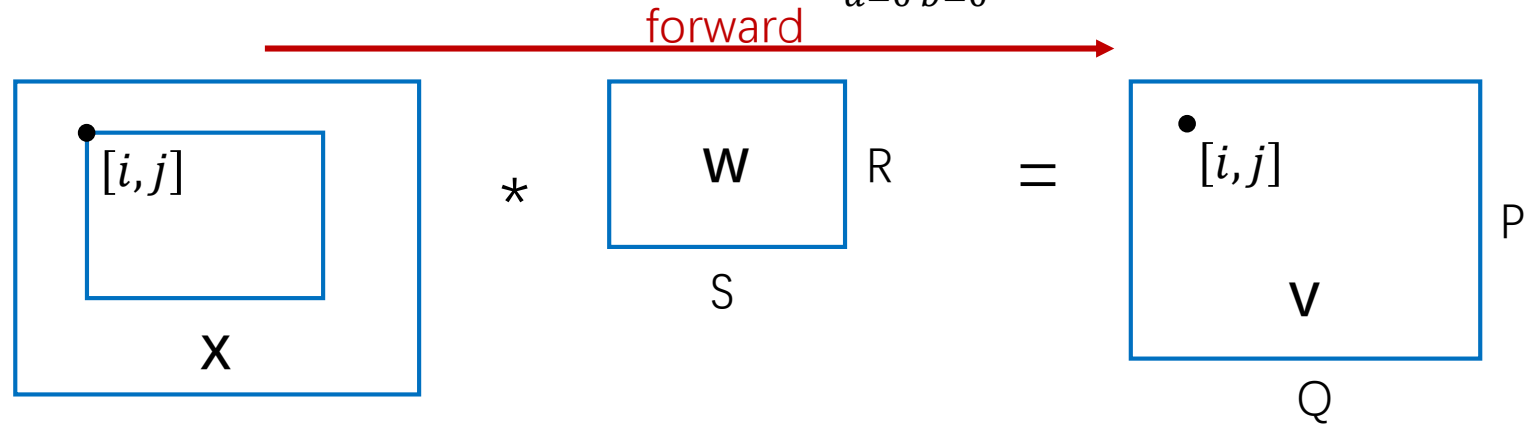
The sizes of channel do not have to be the same for 3D convolution.



Convolution Backprop.

- 2D convolution with
 - stride=1
 - padding=0
- Given $\partial L / \partial \mathbf{V}$ to compute
 - $\partial L / \partial \mathbf{W}$, i.e., $\partial L / \partial \mathbf{V} \times \partial \mathbf{V} / \partial \mathbf{W}$
 - $\partial L / \partial \mathbf{X}$, i.e., $\partial L / \partial \mathbf{V} \times \partial \mathbf{V} / \partial \mathbf{X}$

$$V[i, j] = \sum_{a=0}^{s-1} \sum_{b=0}^{r-1} x[i+a, j+b] \times w[a, b]$$



$$\begin{array}{ccc} \frac{\partial L}{\partial \mathbf{x}_{0,0}} & \frac{\partial L}{\partial \mathbf{x}_{0,1}} & \dots \\ \frac{\partial L}{\partial \mathbf{x}_{1,0}} & & \ddots \end{array}$$

$$\begin{array}{ccc} \frac{\partial L}{\partial \mathbf{w}_{0,0}} & \frac{\partial L}{\partial \mathbf{w}_{0,1}} & \dots \\ \frac{\partial L}{\partial \mathbf{w}_{1,0}} & & \ddots \end{array} \quad R$$

S

$$\begin{array}{ccc} \frac{\partial L}{\partial \mathbf{v}_{0,0}} & \frac{\partial L}{\partial \mathbf{v}_{0,1}} & \dots & \frac{\partial L}{\partial \mathbf{v}_{0,Q-1}} \\ \frac{\partial L}{\partial \mathbf{v}_{1,0}} & & \ddots & \end{array} \quad P$$

Q

Convolution Backprop. (Cont'd)

$$\begin{array}{ccc} \mathbf{x}_{0,0} & \mathbf{x}_{0,1} & \dots \\ \mathbf{x}_{1,0} & & \\ & \ddots & \mathbf{x}_{k,l} \end{array}$$

$$\begin{array}{ccc} \frac{\partial L}{\partial \mathbf{w}_{0,0}} & \frac{\partial L}{\partial \mathbf{w}_{0,1}} & \dots \\ \frac{\partial L}{\partial \mathbf{w}_{1,0}} & & \ddots \end{array}$$

S

R

$$\begin{array}{ccc} \frac{\partial L}{\partial \mathbf{v}_{0,0}} & \frac{\partial L}{\partial \mathbf{v}_{0,1}} & \dots & \frac{\partial L}{\partial \mathbf{v}_{0,Q-1}} \\ \frac{\partial L}{\partial \mathbf{v}_{1,0}} & & \ddots & \end{array}$$

P

- For the weights (stride = 1, padding = 0)

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}_{0,0}} &= \frac{\partial L}{\partial \mathbf{v}_{0,0}} \frac{\partial \mathbf{v}_{0,0}}{\partial \mathbf{w}_{0,0}} + \frac{\partial L}{\partial \mathbf{v}_{0,1}} \frac{\partial \mathbf{v}_{0,1}}{\partial \mathbf{w}_{0,0}} + \frac{\partial L}{\partial \mathbf{v}_{0,2}} \frac{\partial \mathbf{v}_{0,2}}{\partial \mathbf{w}_{0,0}} + \dots + \frac{\partial L}{\partial \mathbf{v}_{P-1,Q-1}} \frac{\partial \mathbf{v}_{P-1,Q-1}}{\partial \mathbf{w}_{0,0}} \\ &= \frac{\partial L}{\partial \mathbf{v}_{0,0}} \mathbf{x}_{0,0} + \frac{\partial L}{\partial \mathbf{v}_{0,1}} \mathbf{x}_{0,1} + \frac{\partial L}{\partial \mathbf{v}_{0,2}} \mathbf{x}_{0,2} + \dots + \frac{\partial L}{\partial \mathbf{v}_{P-1,Q-1}} \mathbf{x}_{P-1,Q-1} \\ &= \mathbf{x} * \frac{\partial L}{\partial \mathbf{v}} \end{aligned}$$

Also a convolution!!!

The Other Components in Convolution Layers

- Pooling
 - Usually used to reduce data
 - `torch.nn.functional.max_pool2d(input, kernel_size, stride=None, padding=0, dilation=1, ceil_mode=False, return_indices=False)`
 - `torch.nn.quantized.functional.avg_pool2d(input, kernel_size, stride=None, padding=0, ceil_mode=False, count_include_pad=True, divisor_override=None)`

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

2*2 maxpool
stride = 2

9	7
2	8

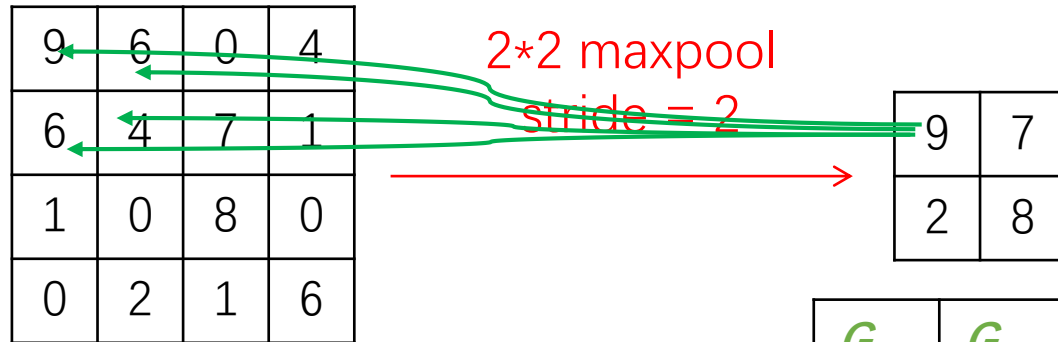
8	6	0	4
6	4	7	1
2	0	8	0
0	2	2	6

2*2 avgpool
stride = 2

7	3
1	4



Pooling Backprop.



$G_{0,0}$	$G_{0,1}$
$G_{1,0}$	$G_{1,1}$

$G_{0,0}$	0
0	0

0	0
$G_{0,1}$	0

0	0
0	$G_{1,0}$

$G_{1,1}$	0
0	0

8	6	0	4
6	4	7	1
2	0	8	0
0	2	2	6

2*2 avgpool
stride = 2

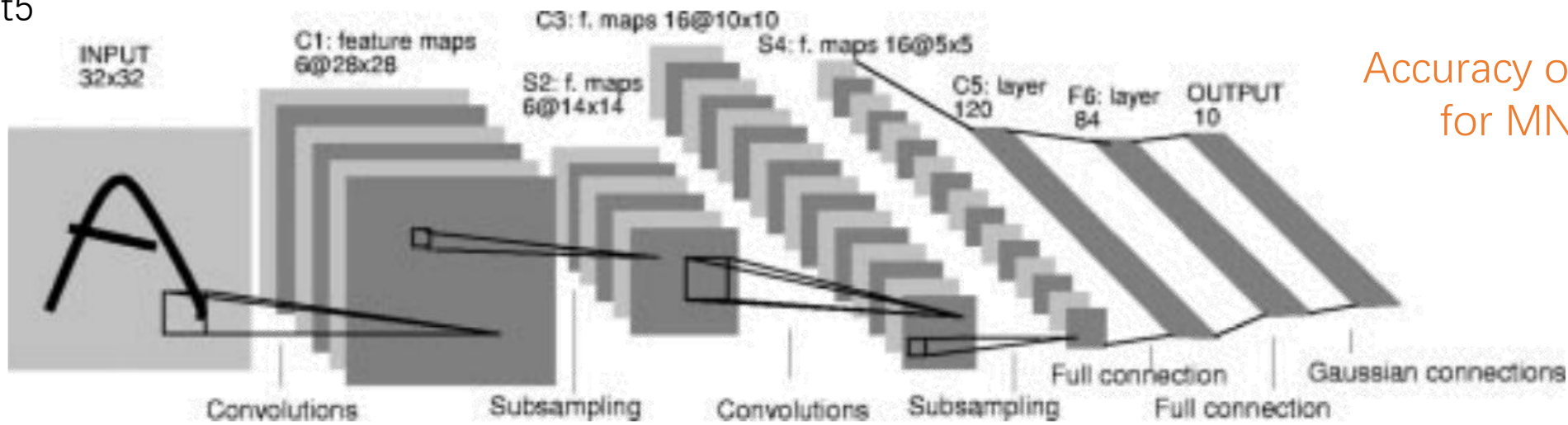
7	3
1	4

$G_{0,0}$	$G_{0,1}$
$G_{1,0}$	$G_{1,1}$

$G_{0,0}/4$	$G_{0,0}/4$	$G_{0,1}/4$	$G_{0,1}/4$
$G_{0,0}/4$	$G_{0,0}/4$	$G_{0,1}/4$	$G_{0,1}/4$
$G_{1,0}/4$	$G_{1,0}/4$	$G_{1,1}/4$	$G_{1,1}/4$
$G_{1,0}/4$	$G_{1,0}/4$	$G_{1,1}/4$	$G_{1,1}/4$

The Evolution of CNN Models

- LeNet5



Accuracy over 99%
for MNIST

6 5*5 filters
156 trainable
parameters

122,304
connections

2*2 average
pooling with
sigmoidal func.
(with a trainable
factor and bias)

12 trainable
parameters

5,880
connections

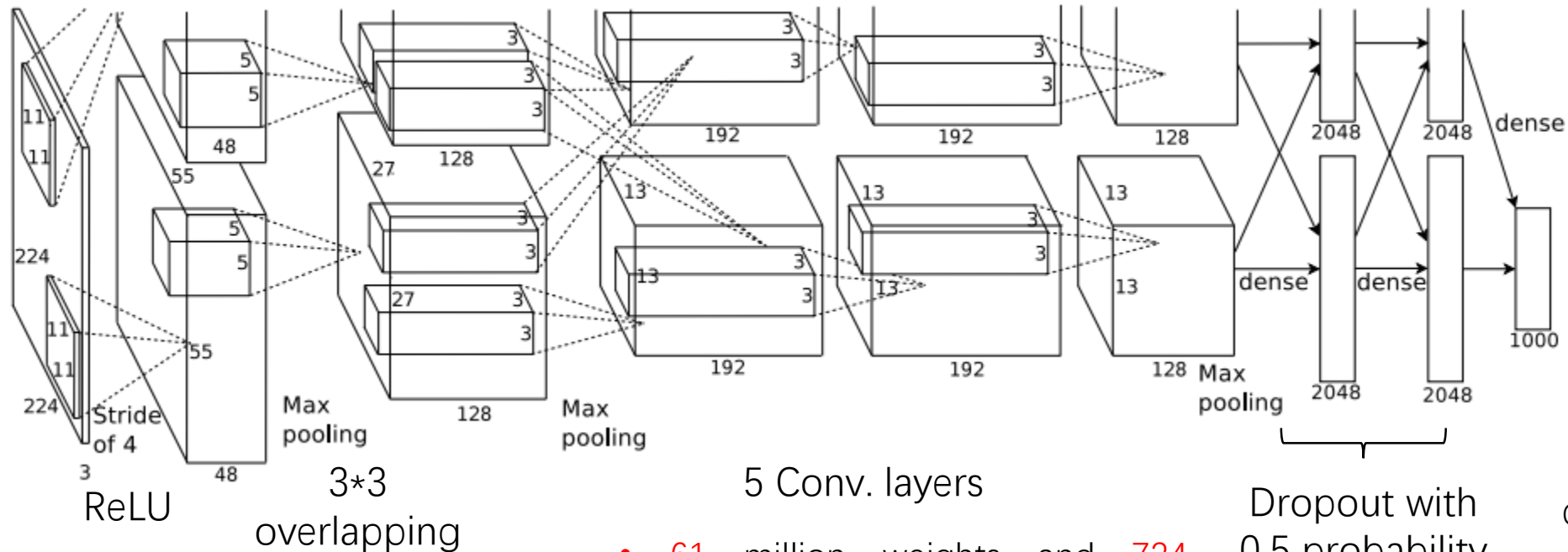
Table 1 Each Column Indicates Which Feature Map in S2 Are Combined by the Units in a Particular Feature Map of C3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

AlexNet

- The first CNN to win the ImageNet Challenge in 2012



- 2 GTX 580 GPUs, each with 3GB memory, training 5~6 days for roughly 90 cycles with 1.2 million images

- 61 million weights and 724 million MACs for one image
- Around 85% top-5 accuracy on ImageNet

Dropout with 0.5 probability
3 FC layers

Dropout downside
converge slower

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

AlexNet (Cont'd)

- The first CNN to win the ImageNet Challenge in 2012

ReLU, improve training convergence speed

LRN (obsolete)

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$

Dropout (useful technique preventing overfitting)

Manually randomly (with probability of 0.5 in **AlexNet**) set the outputs of neurons to 0

Mini-batch backprop/training without updating the weights connected to the selected neurons

Restore the original NN structure and take another mini-batch

Repeat 1~3 until convergence

For fully-connected (FC) layers

Top 1: 37.5% Top 5: 17.0% error rate ImageNet

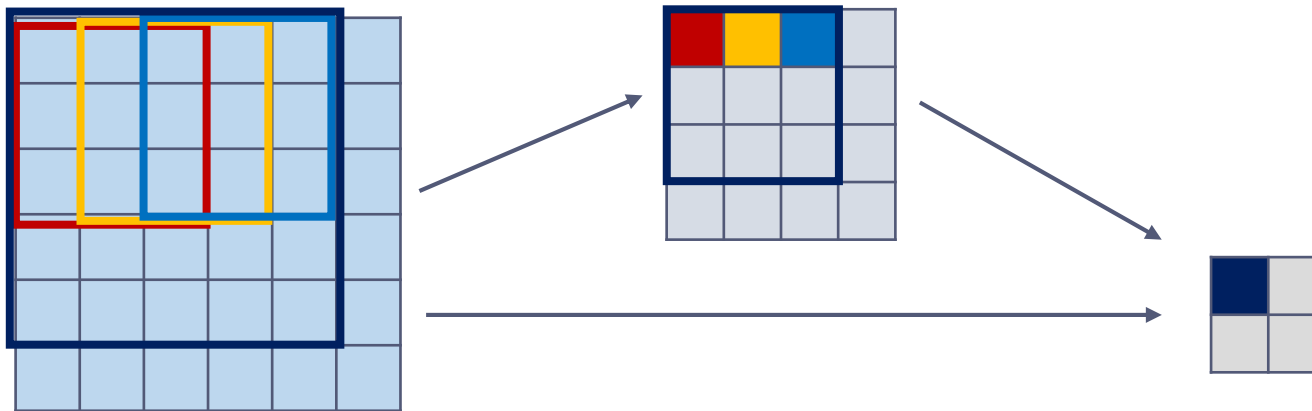
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

立志成才 报国强民

VGG (from Google)

- VGG from Google (receptive field)
 - Pretrain the first and last few layers, the optimized parameters are used as the initialization value of a deeper DNN
 - Smaller convolution filter but deeper structure
 - top1: 24.7%, top5: 7.5% error rate

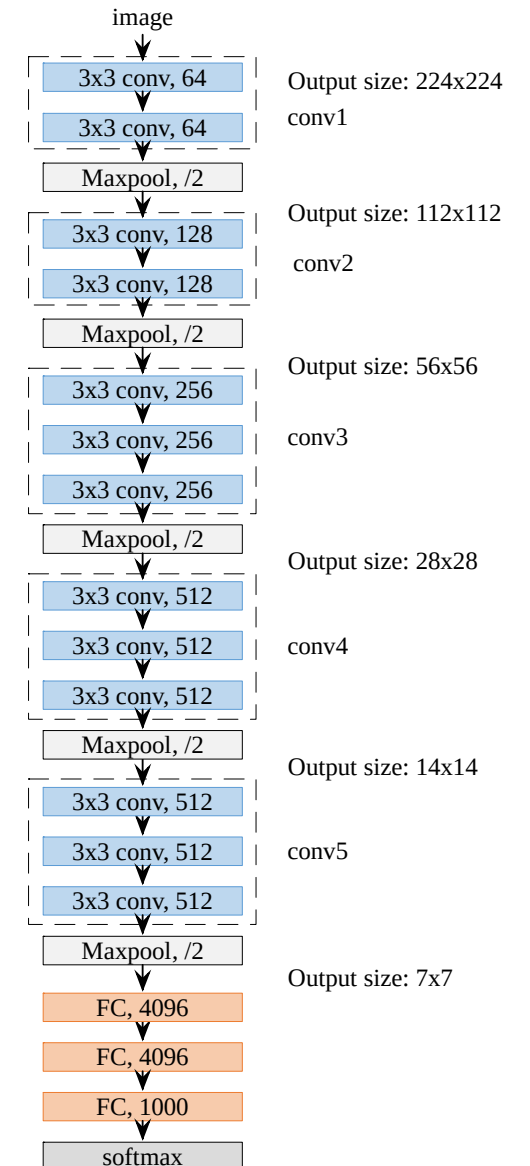
Receptive field



- Trained with Titan GPU for multiple weeks

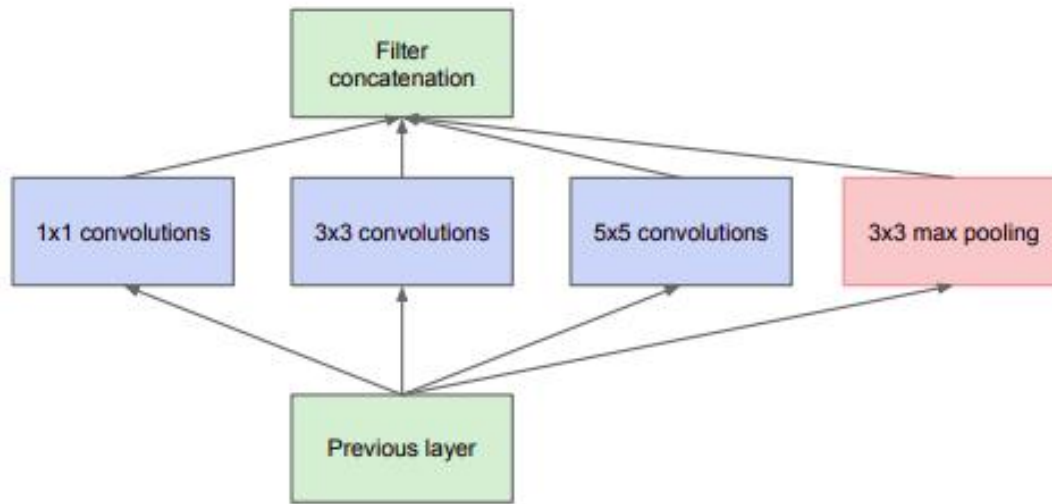
Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

VGG-16

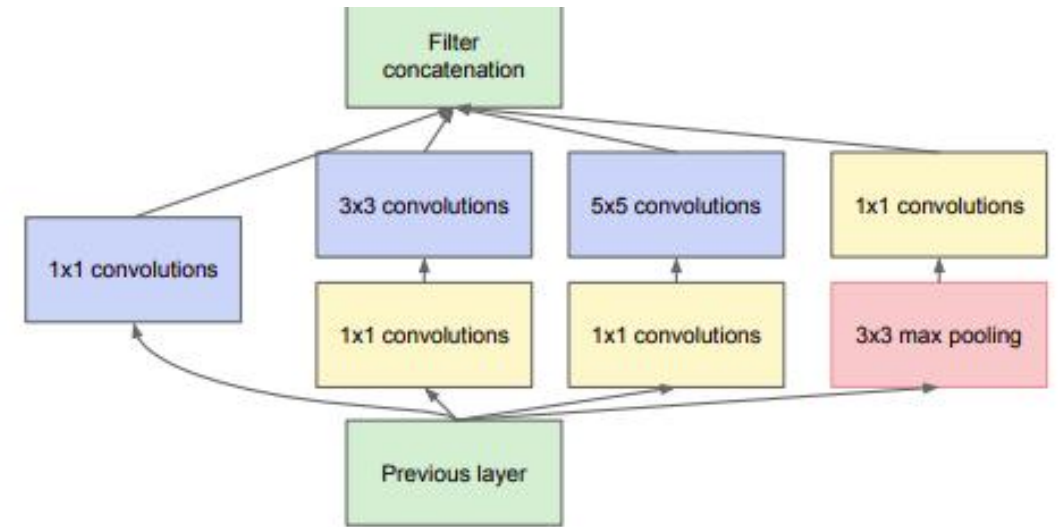


Inception (from Google)

- AKA GoogLeNet



(a) Inception module, naïve version



(b) Inception module with dimension reductions

Inception-v1(GoogLeNet): Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions, CVPR 2015:1-9.

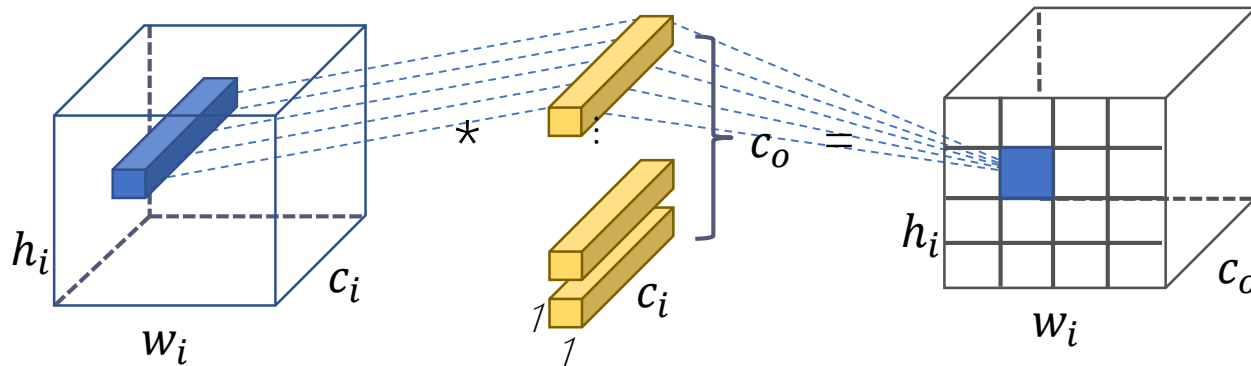
BN-Inception: Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. ICML, 2015:448-456.

Inception-v2, Inception-v3: Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[C]// CVPR, 2016:2818-2826.

Inception-v4: Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, AAAI'2017.

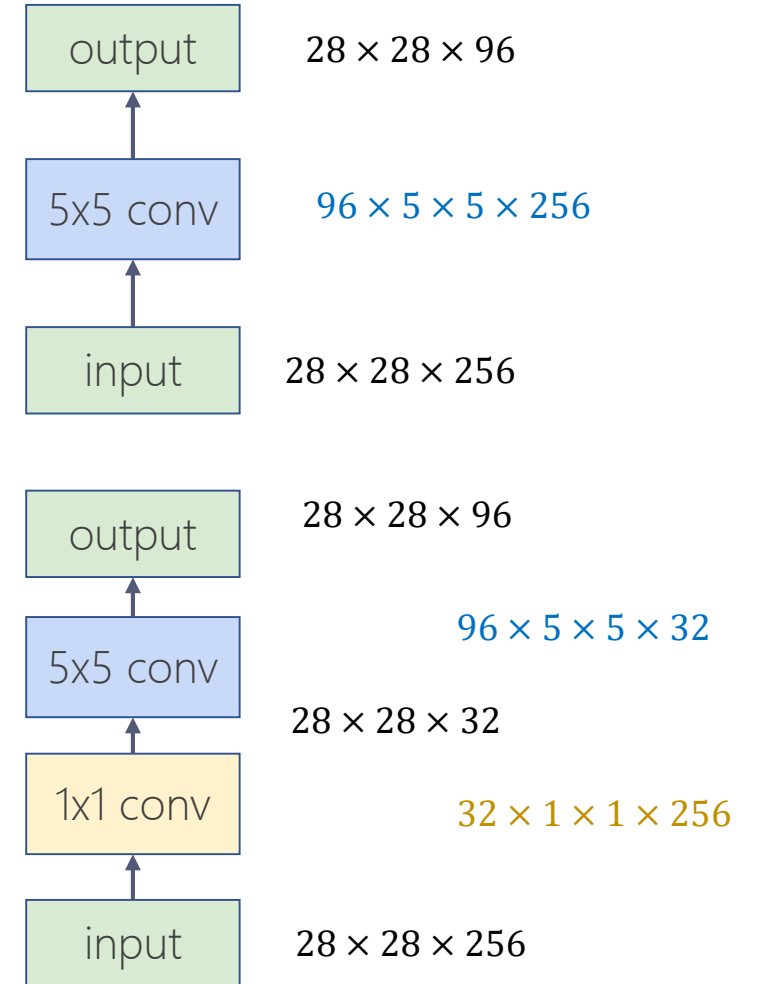
1*1 Convolution

- AKA GoogLeNet



MAC

Memory



Batch Normalization

- Batch Normalization
 - Before activation function and after conv.
 - For mini-batch containing m samples

$$\hat{\mathbf{v}} = \frac{\mathbf{V} - \boldsymbol{\mu}}{\boldsymbol{\delta}}$$

- During training

$$\boldsymbol{\mu}_{\mathcal{B}} = \frac{1}{m} \sum_i \mathbf{v}_{i,:}$$

$$\boldsymbol{\delta}_{\mathcal{B}} = \sqrt{\epsilon + \frac{1}{m} \sum_i (\mathbf{v}_i - \boldsymbol{\mu})^2}$$

- Meanwhile

$$\boldsymbol{\mu}_{\text{running}} = \alpha \boldsymbol{\mu}_{\text{running}} + (1 - \alpha) \boldsymbol{\mu}_{\mathcal{B}}$$

$$\boldsymbol{\delta}_{\text{running}} = \alpha \boldsymbol{\delta}_{\text{running}} + (1 - \alpha) \boldsymbol{\delta}_{\mathcal{B}}$$

- During inference
Using running params.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\};$

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad \text{Batch size} \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

For Conv. layers, it is normalized w.r.t. each channel of the feature maps to maintain spatial property (A.K.A. spatial batch normalization).

<https://arxiv.org/abs/1502.03167>

BN Backprop.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

- Given $\frac{\partial L}{\partial y_i}$, to compute $\frac{\partial L}{\partial \gamma}, \frac{\partial L}{\partial \beta}, \frac{\partial L}{\partial x_i}$

$$\frac{\partial L}{\partial \gamma} = \sum_i \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial \gamma} = \sum_i \frac{\partial L}{\partial y_i} \hat{x}_i$$

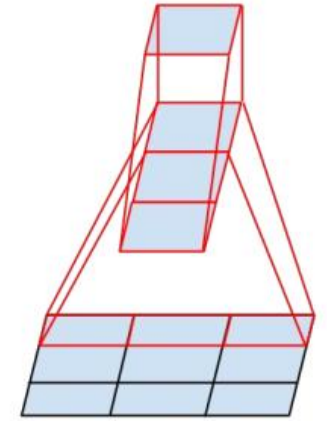
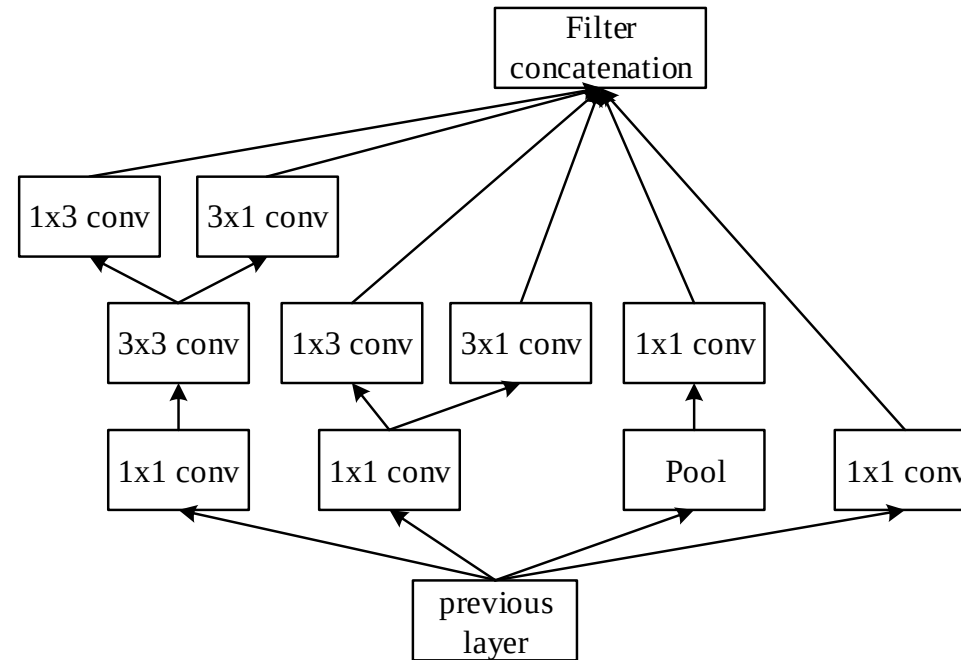
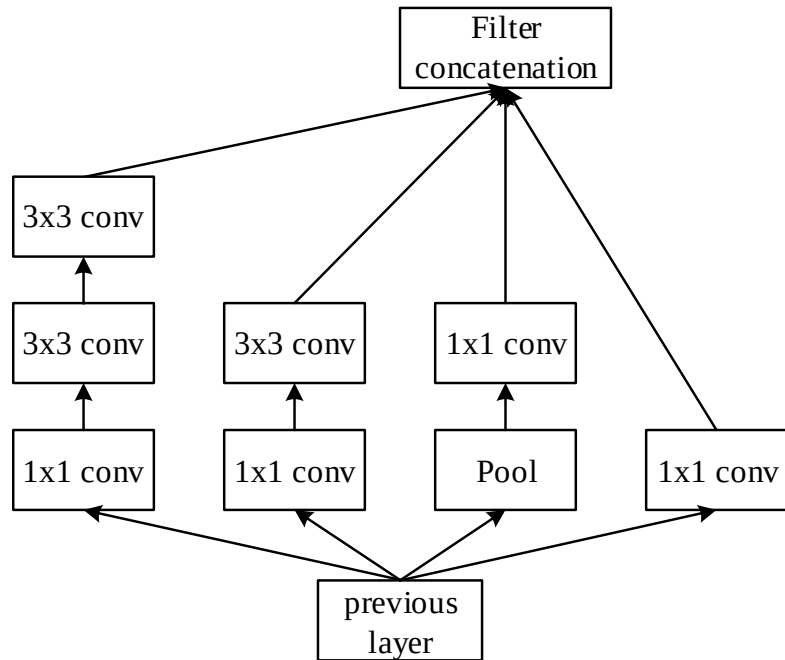
$$\frac{\partial L}{\partial \beta} = \sum_i \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial \beta} = \sum_i \frac{\partial L}{\partial y_i}$$

$$\frac{\partial L}{\partial x_i} =$$



Inception V3

- Break 3*3 convs. into 1*3 & 3*1 convs.



ResNet

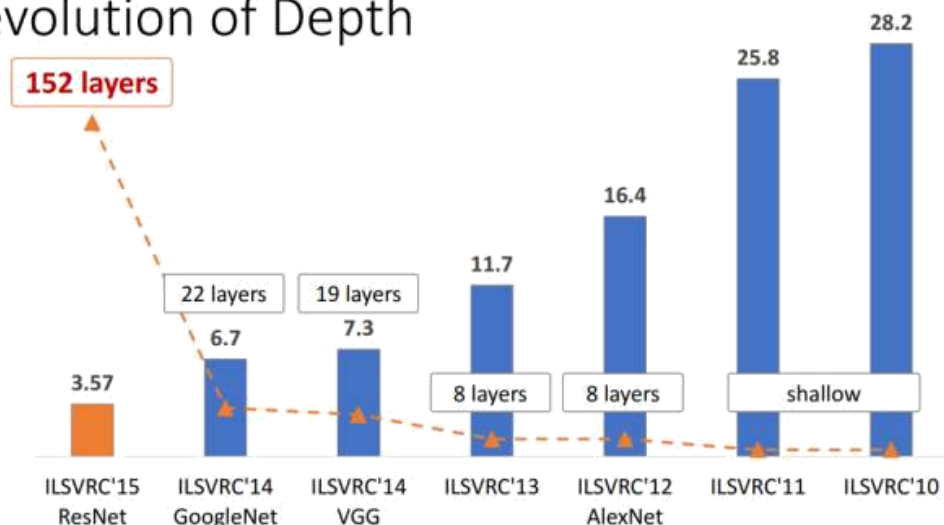
- ImageNet, top5: 3.57% (ResNet-152)

ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**

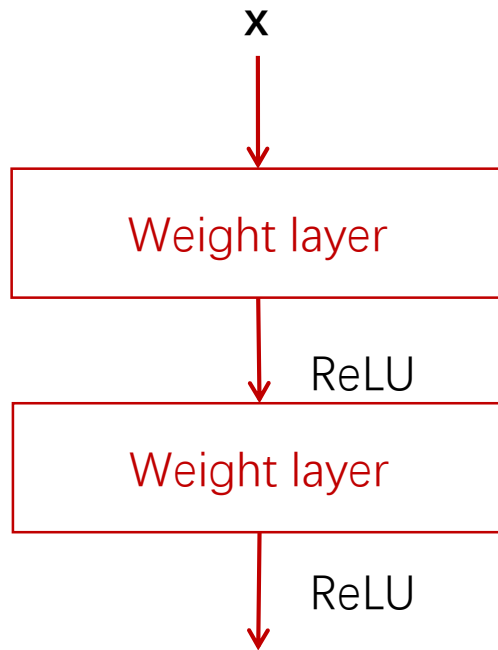
- ImageNet Classification: “Ultra-deep” 152-layer nets
- ImageNet Detection: 16% better than 2nd
- ImageNet Localization: 27% better than 2nd
- COCO Detection: 11% better than 2nd
- COCO Segmentation: 12% better than 2nd

Revolution of Depth

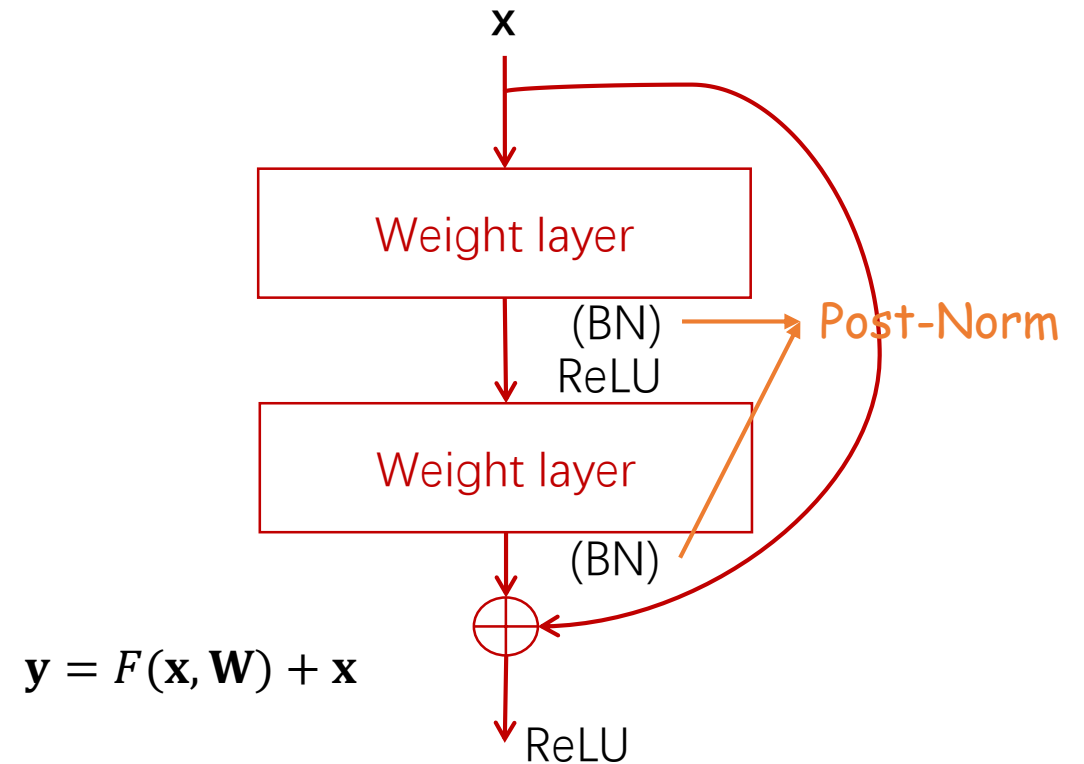


He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Residual Connection



Plain net.

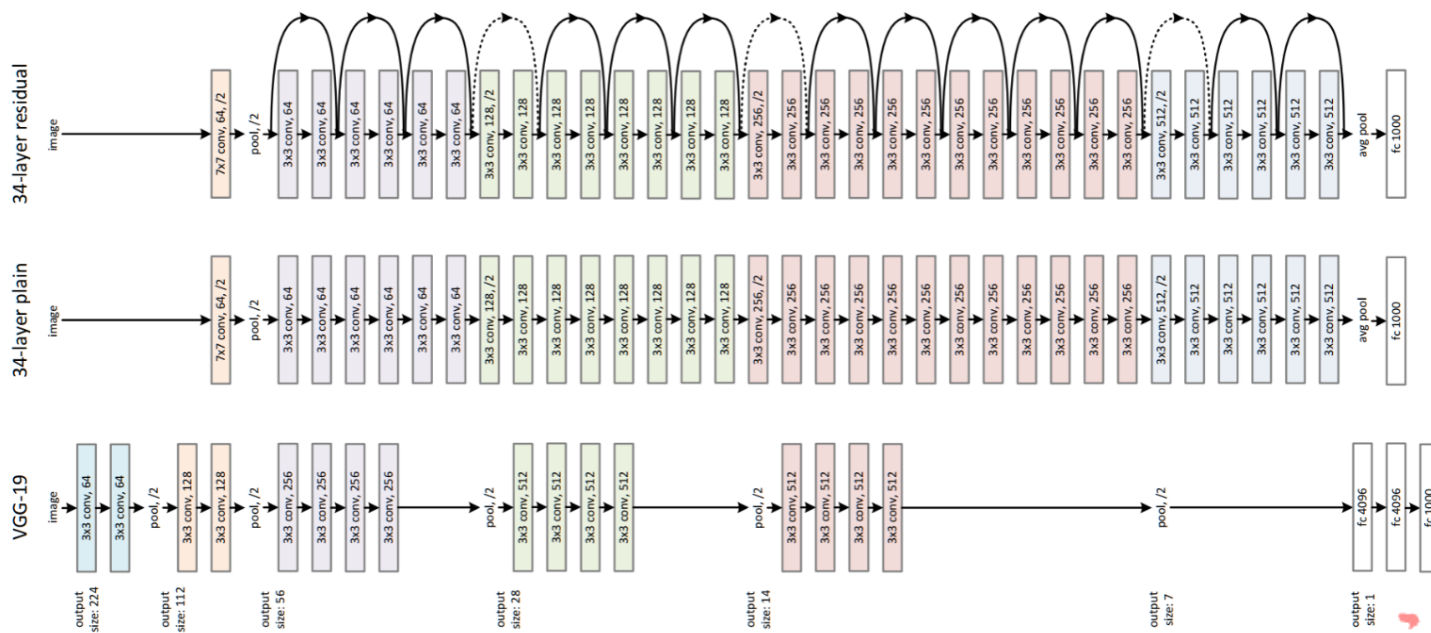


Residual learning block.

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

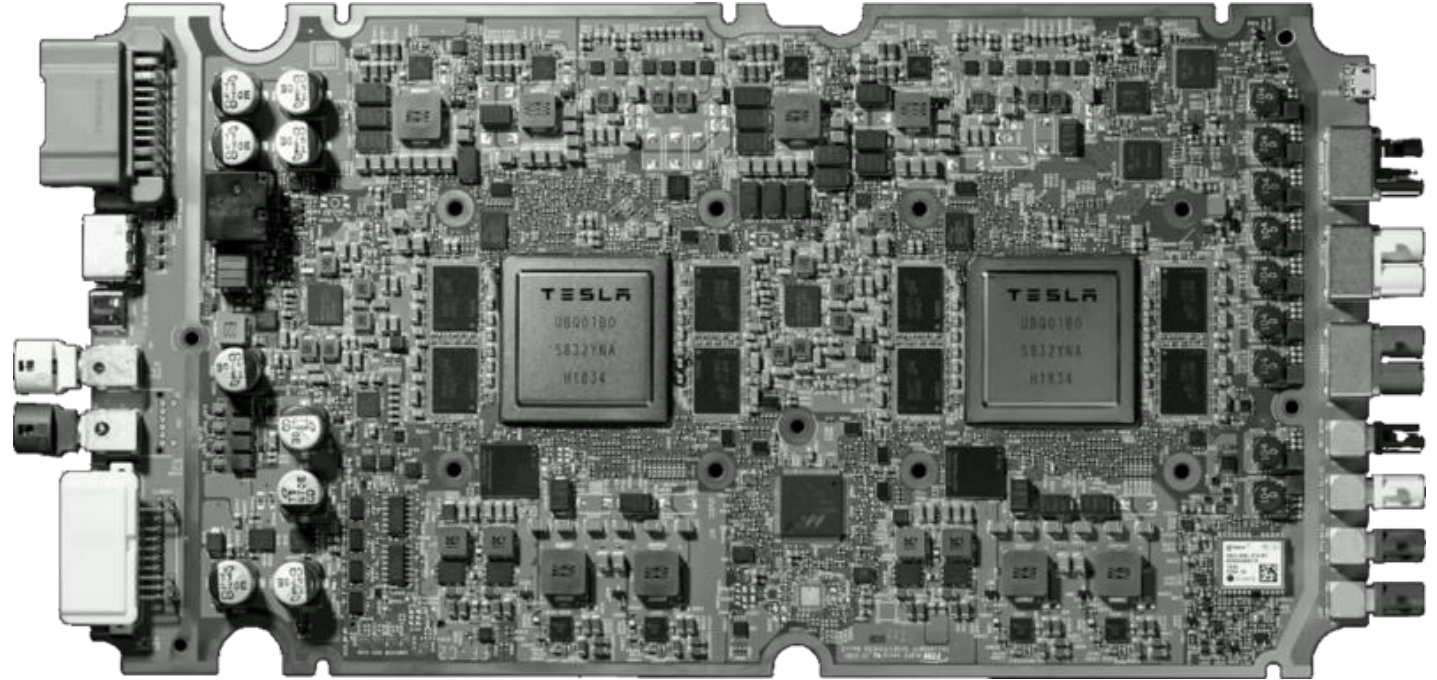
Residual Connection (Cont'd)

- Add shortcut connections
 - Identity shortcuts added directly when input and output are of the same dimensions (solid lines)
 - When dimensions change (dotted lines)
 - Extra zero padding (no extra params.)
 - Linear transformation (1*1 conv. with extra params.)
- BN after each convolution and before activation
- Comparison with VGG-16
 - 11.3 billion FLOPs vs. 15.3/19.6 billion FLOPs (VGG16/19)



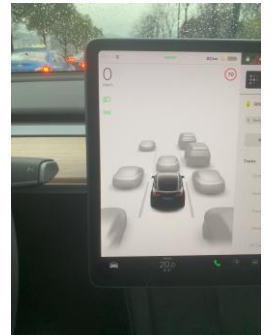
From CNN to Real Applications

- Object detection
 - Autonomous driving
 - Surveillance
 -



[https://en.wikichip.org/wiki/tesla_\(car_company\)/fsd_chip#Operation](https://en.wikichip.org/wiki/tesla_(car_company)/fsd_chip#Operation)

Han, Jianhua, et al. "SODA10M: A Large-Scale 2D Self/Semi-Supervised Object Detection Dataset for Autonomous Driving." Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track. 2021.



立志成才 报国裕民

Industrial

- Foreign
 - NXP
 - Intel/Mobileye
 - TI
 - ARM
 - Infineon
 - Qualcomm
 - Maxim
 - NVIDIA Xavier/Orin
 - Renesas
 - Xilinx (AMD)
 - Tesla FSD
 - Aisin
 - OmniVision
 - etc.
- Domestic
 - Horizon Robotics (Journey series)
 - Huawei
 - Ambarella (安霸 CV series)
 - Black sesame (invested by Xiaomi)
 - SemiDrive
 - Westwell
 - Leapmotor (零跑)
 - etc.

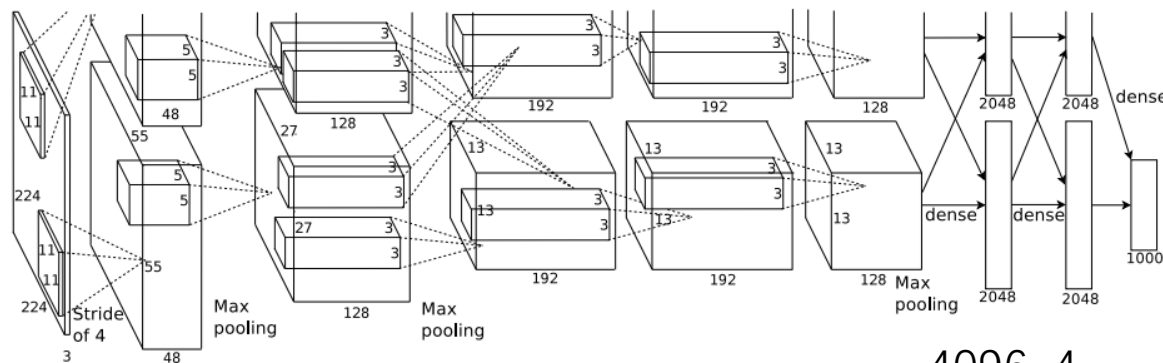


Classification vs. localization

Input	single and big object
Output	label & bounding box (Cat) & (x, y, w, h)



CAT



Target Label
Cat

CE Loss

Class Scores

Cat: 0.9

Dog: 0.05

...

4096-1000
FC layer

Target Box
(x, y, w, h)

L2/MSE Loss

4096-4
FC layer

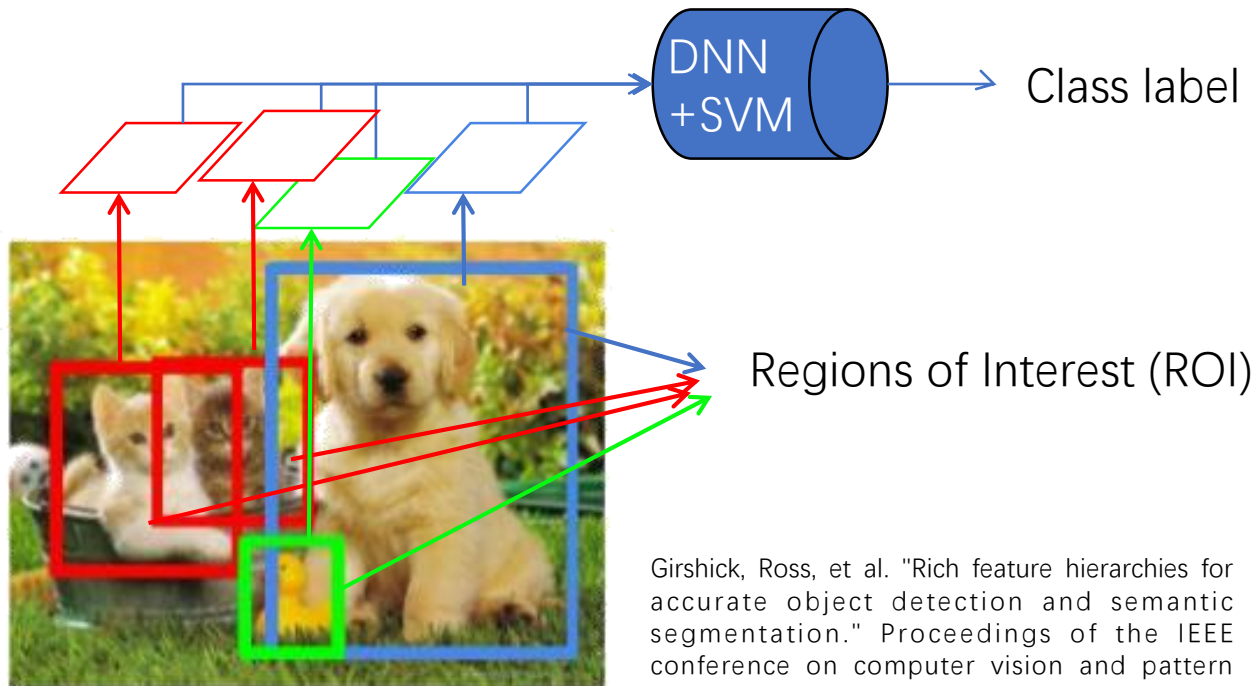
Box Coordinates
(x', y', w', h')

Total
Loss

(Weighted
sum)

Object Detection

Input	Images with multiple objects
Output	Labels & bounding boxes



Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

- R-CNN: region-based CNN
- **Selective search** gives 2000 regions (conventional CV)
- Bounding box regression
 - Start with selective search result (p_x, p_y, p_h, p_w)
 - One possible solution: define transform (t_x, t_y, t_h, t_w)
 - Output box (b_x, b_y, b_h, b_w)

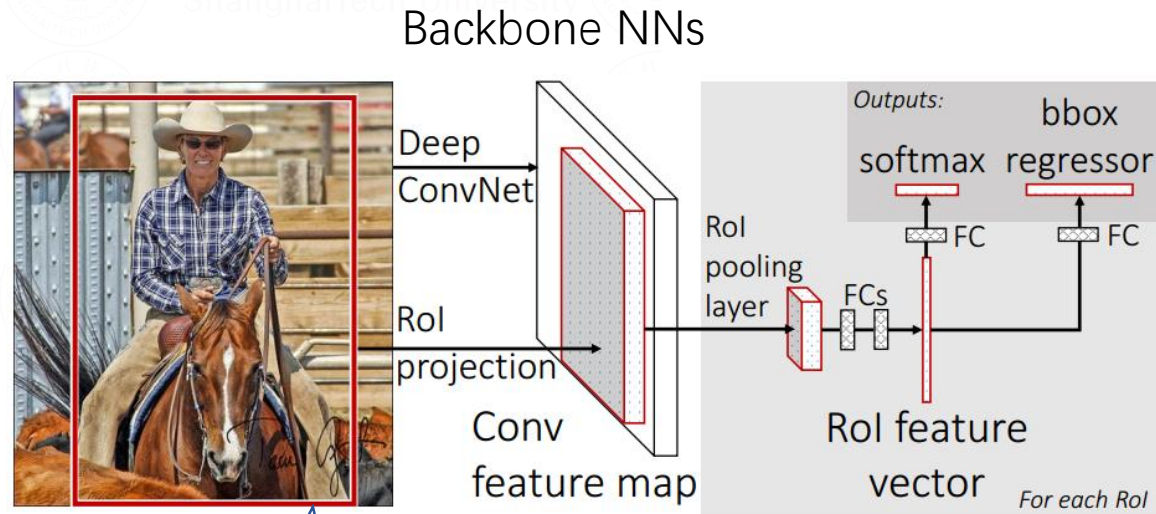
Use formula:

$$b_x = p_x + p_w t_x$$
$$b_w = p_w \exp(t_w)$$

$$b_y = p_y + p_h t_y$$
$$b_h = p_h \exp(t_h)$$

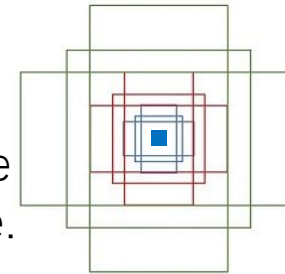
Fast & Faster R-CNN

- Fast R-CNN



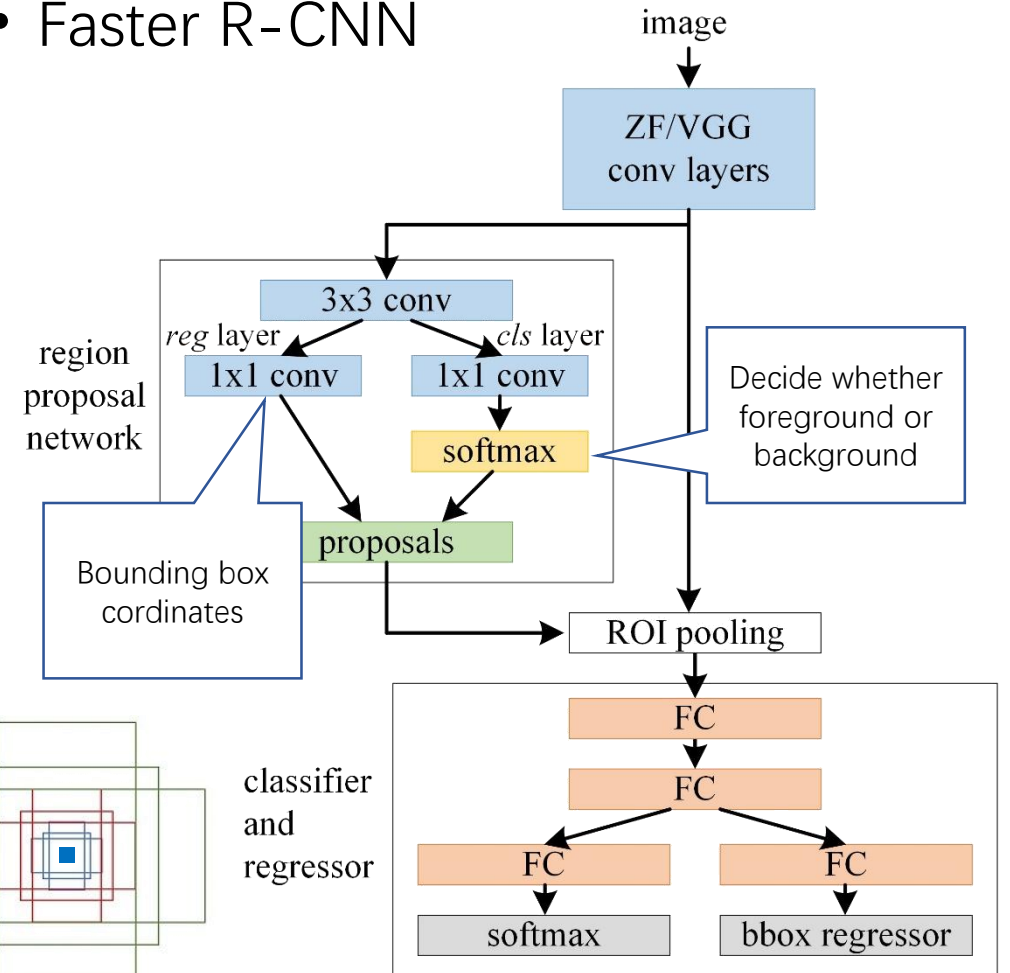
Selective search

Anchor box:
Three different sizes with three different shape.



classifier and regressor

- Faster R-CNN



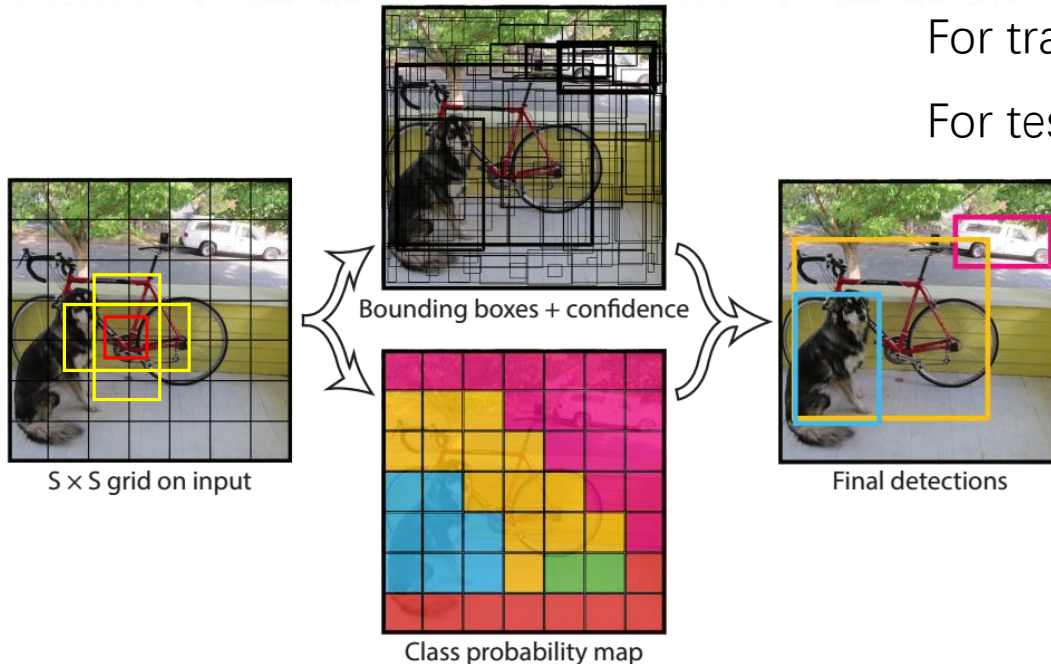
Girshick, Ross. "Fast R-CNN." Proceedings of the IEEE International Conference on Computer Vision. 2015.

Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).

立志成才 报国裕民

You Look Only Once (YOLO)

- One-stage object detection (YOLO)
 - Split into $S \times S$ cells, for each cell, assign B bounding boxes.
 - For each bounding box, predict position $\{x, y, w, h\}$, confidence and class (20 classes for PASCAL VOC dataset, plus 1 class for background)



For training: confidence = $\text{Pr}(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$

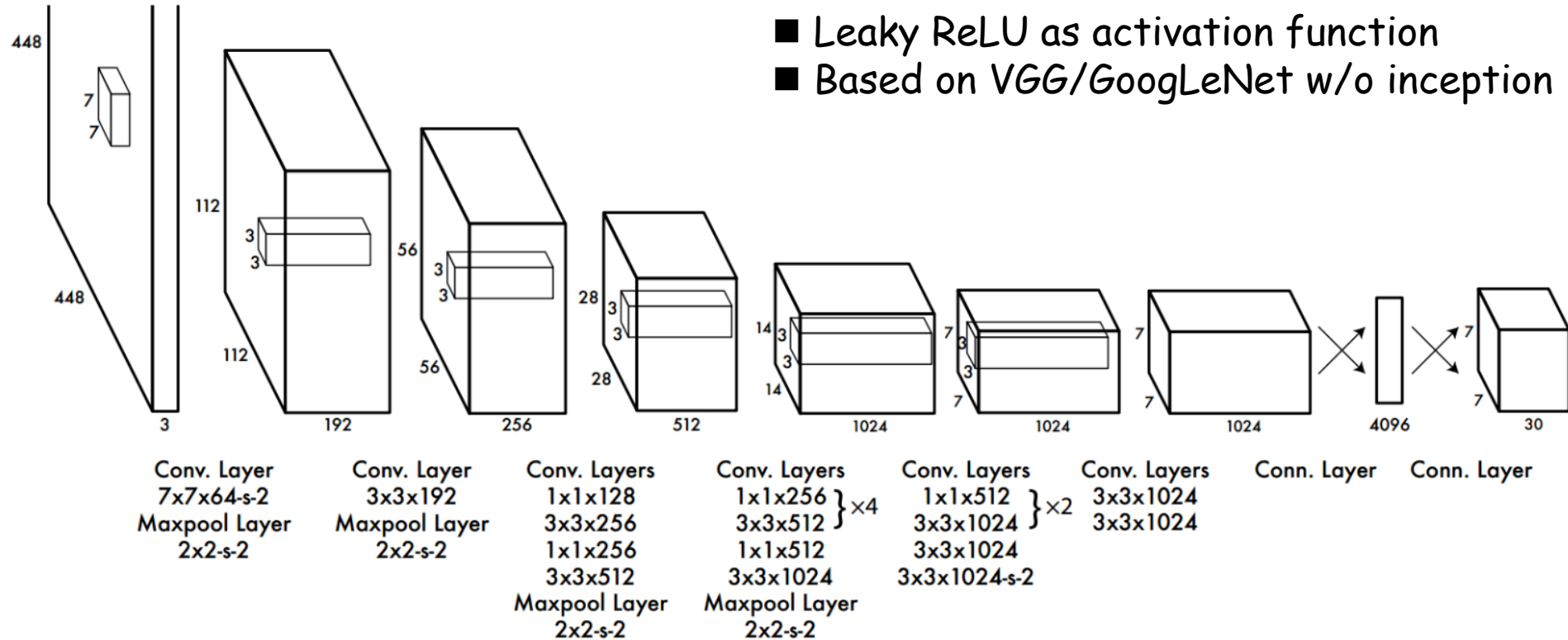
For test: $\text{Pr}(\text{Class}_i | \text{Object}) * \text{Pr}(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \text{Pr}(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$
 $\{0, 1\}$

- Input: image
- Output: $S * S * (B * (4 + \text{Confidence}) + \# \text{ Class}))$ vector/tensor

YOLO(v1): Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection. 2015.

立志成才 报国裕民

YOLO



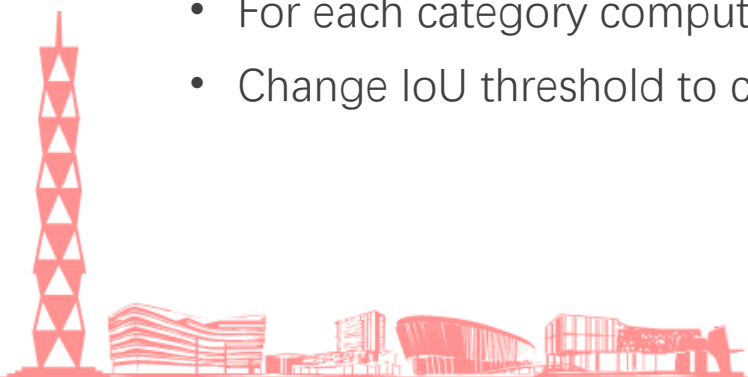
YOLO(v1): Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection. 2015.

Evaluation of Object Detection

- Mean average precision (mAP)
 - Run object detection for all the (test) images

Result \ Label	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	True Negative

- $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$
- $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
 - For each category compute AP area under Precision vs. Recall Curve as mAP
 - Change IoU threshold to compute mAP for COCO dataset



mAP

- AP for one class: Change the confidence ($\text{IoU} \times \text{Prob}(\text{class})$) of threshold, compute recall & precision
- True number of samples in this class: 10
- Average over n classes to compute mAP

Objects detected and their confidence

Thres.	Confidence	True Label	Precision	Recall
	0.98	1	1	0.1
	0.94	1	1	0.2
	0.85	1	1	0.3
	0.77	0	3/4	0.3
	0.59	1	4/5	0.4
	0.43	1	5/6	0.5
	0.25	0	5/7	0.5
	0.11	1	6/8	0.6

