Convolutional Neural Network

Tianchu.Zhao@uts.edu.au

Motivation: Size of data

- MNIST: 32x32x1 per image (728 pixels/weights)
- a small Image in real world: 200x200x3 (120,000 pixels/weights)
  - To train using a neural network
    - Network is complex and easy to overfit
    - Don’t have enough memory or computation power to train

Motivation: viewpoint

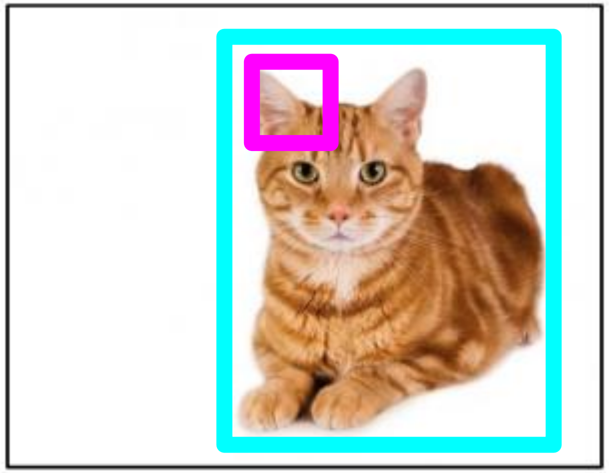
- Why object recognition is difficult
- Viewpoint: changes in view point in the image will scientifically affect the prediction performance from the standard learning method

Cat

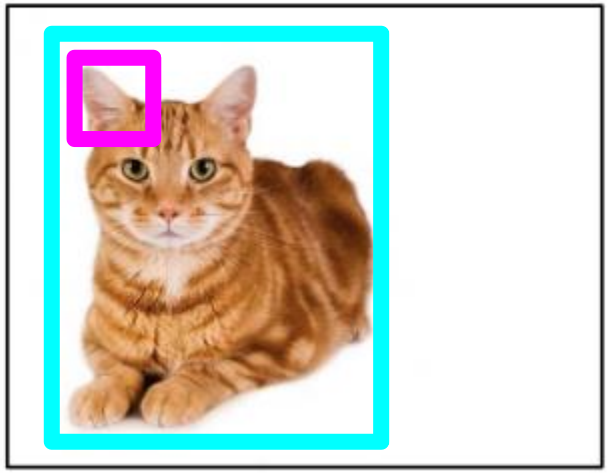
Cat

to overcome viewpoint variance

- Use redundant invariant features
- Use precise bounding box to normalise image (manual)
- Use replicated features and pooling (Convolutional Neural Network)
  - Replication reduces the number of weights that the model needs to learn



Cat



Cat

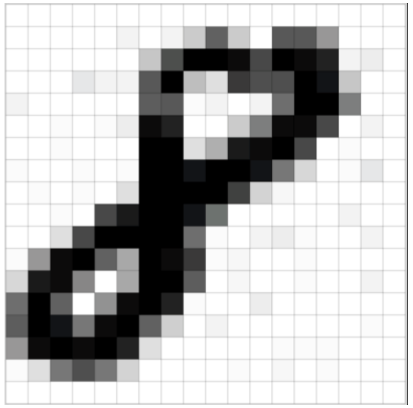
Architecture

- A simple convolutional neural network consists of the following architecture
- INPUT -> CONV -> RELU -> POOL -> FC -> OUTPUT

INPUT

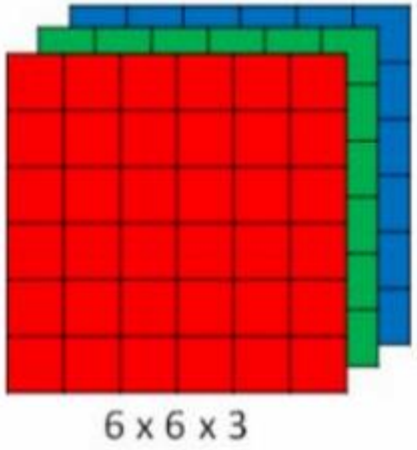
- For black and white image

  1x2-D vector



- For a colourful image

  3x2-D vector

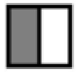


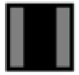
CONV (Convolutional layer)

- This layer will compute the output of neurons in the next layer that connects to a local region from the last layer by summing the product of weights from the next layer and a local region

- previously we have (horizontal edge detection)

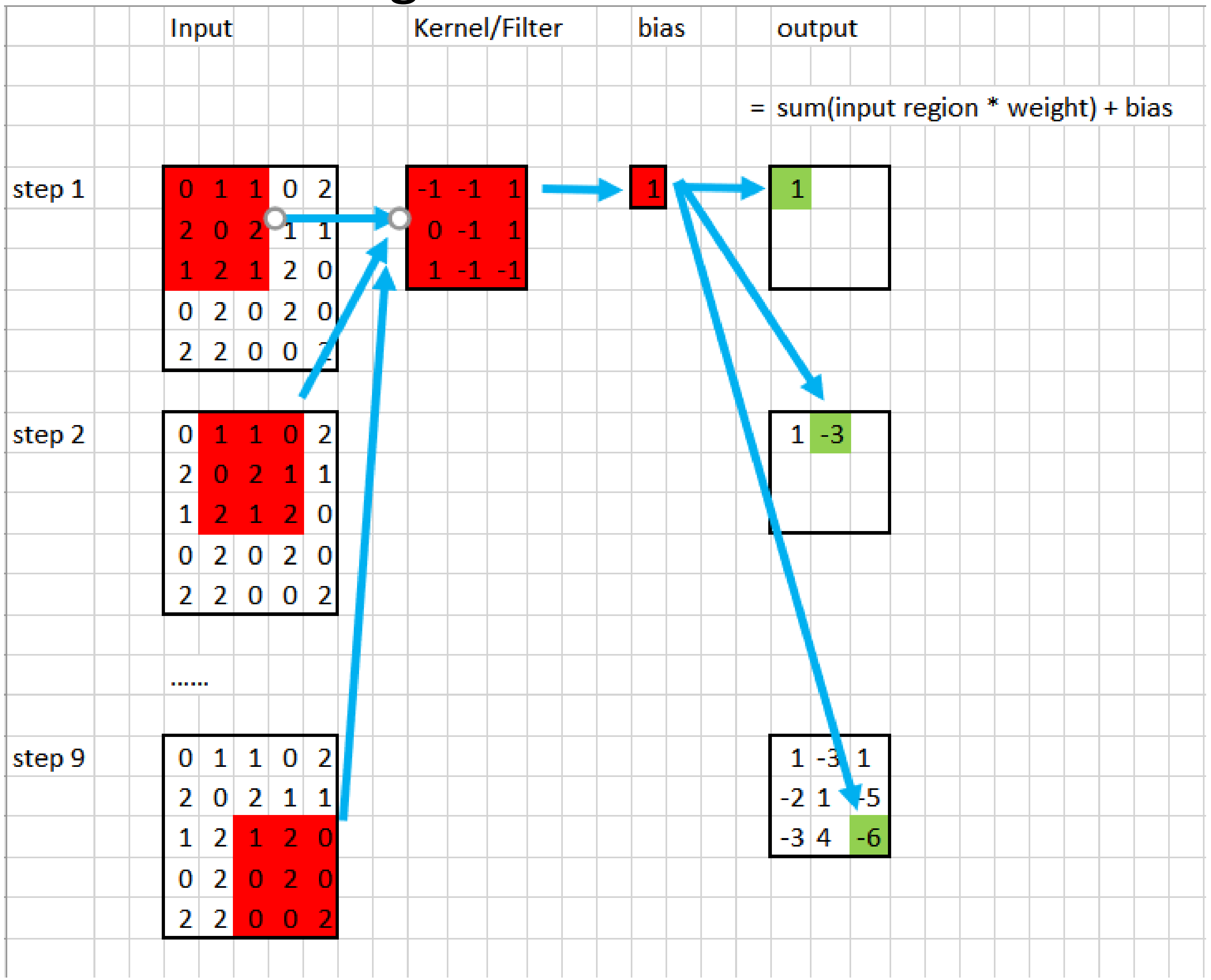
$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$



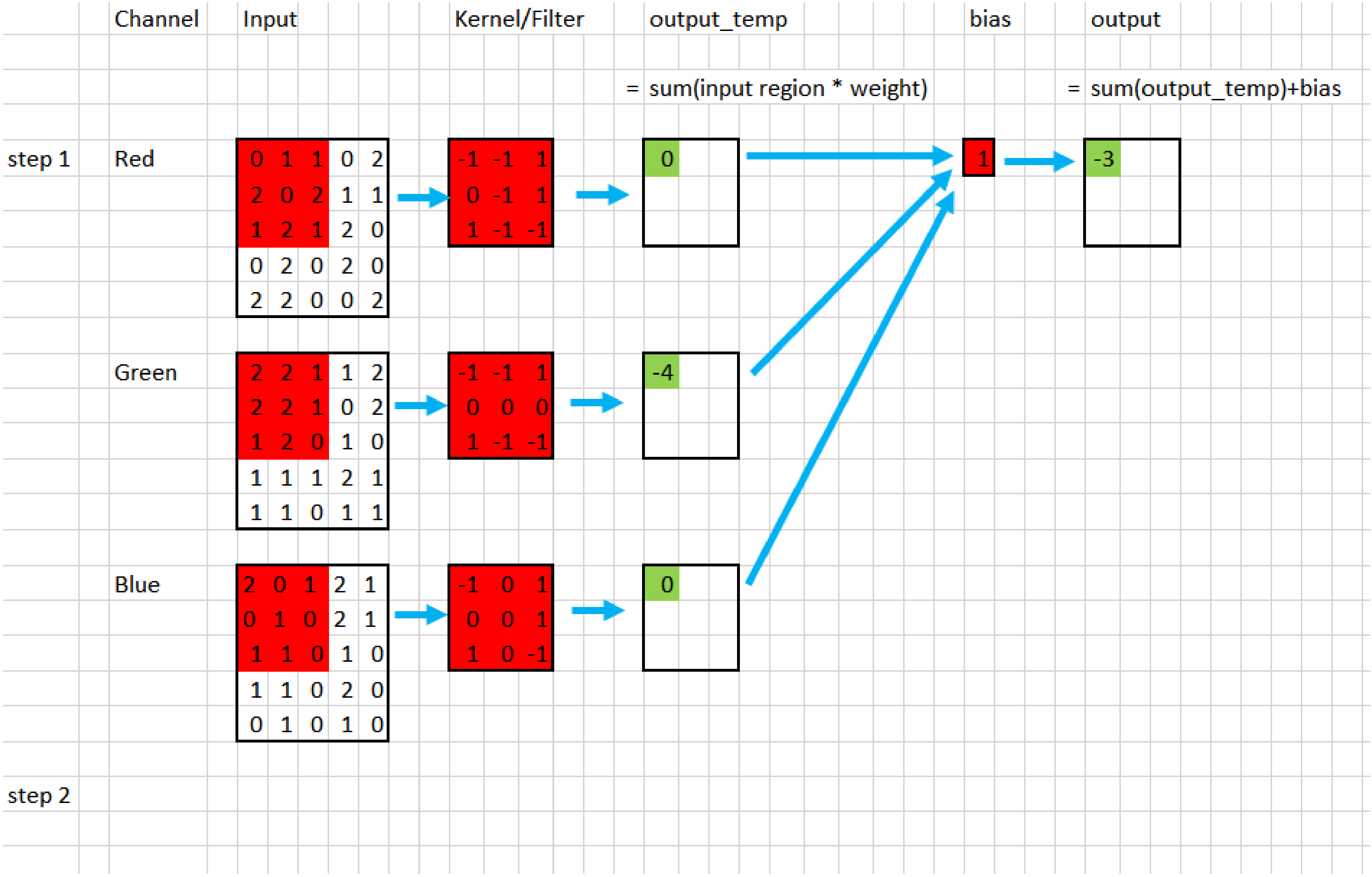


- Now instead of detect edges of the whole image, we focus on a local region

• For black and white image

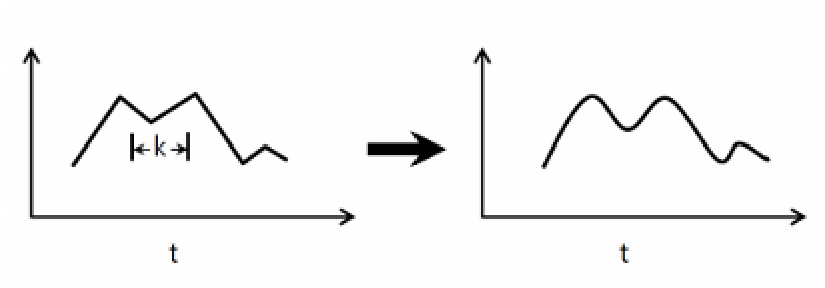


- For colourful image

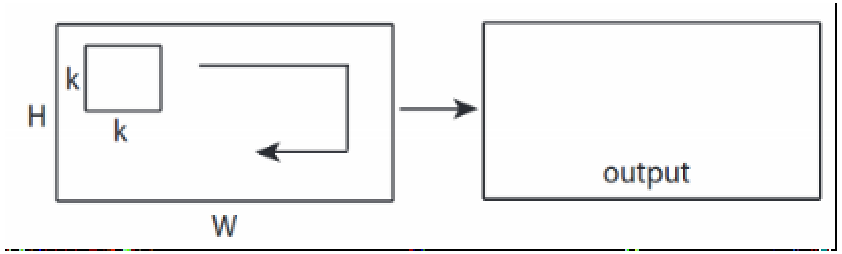


Types of convolution

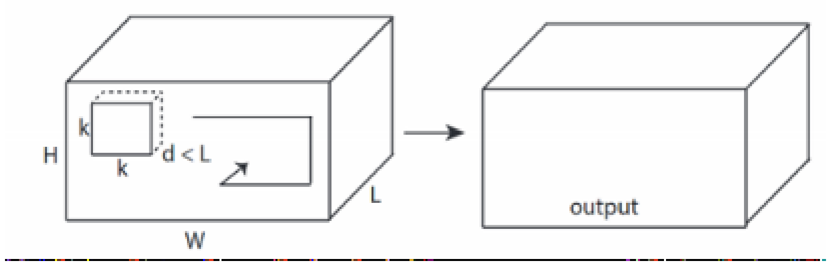
- 1D convolution



- 2d convolution



- 3d convolution



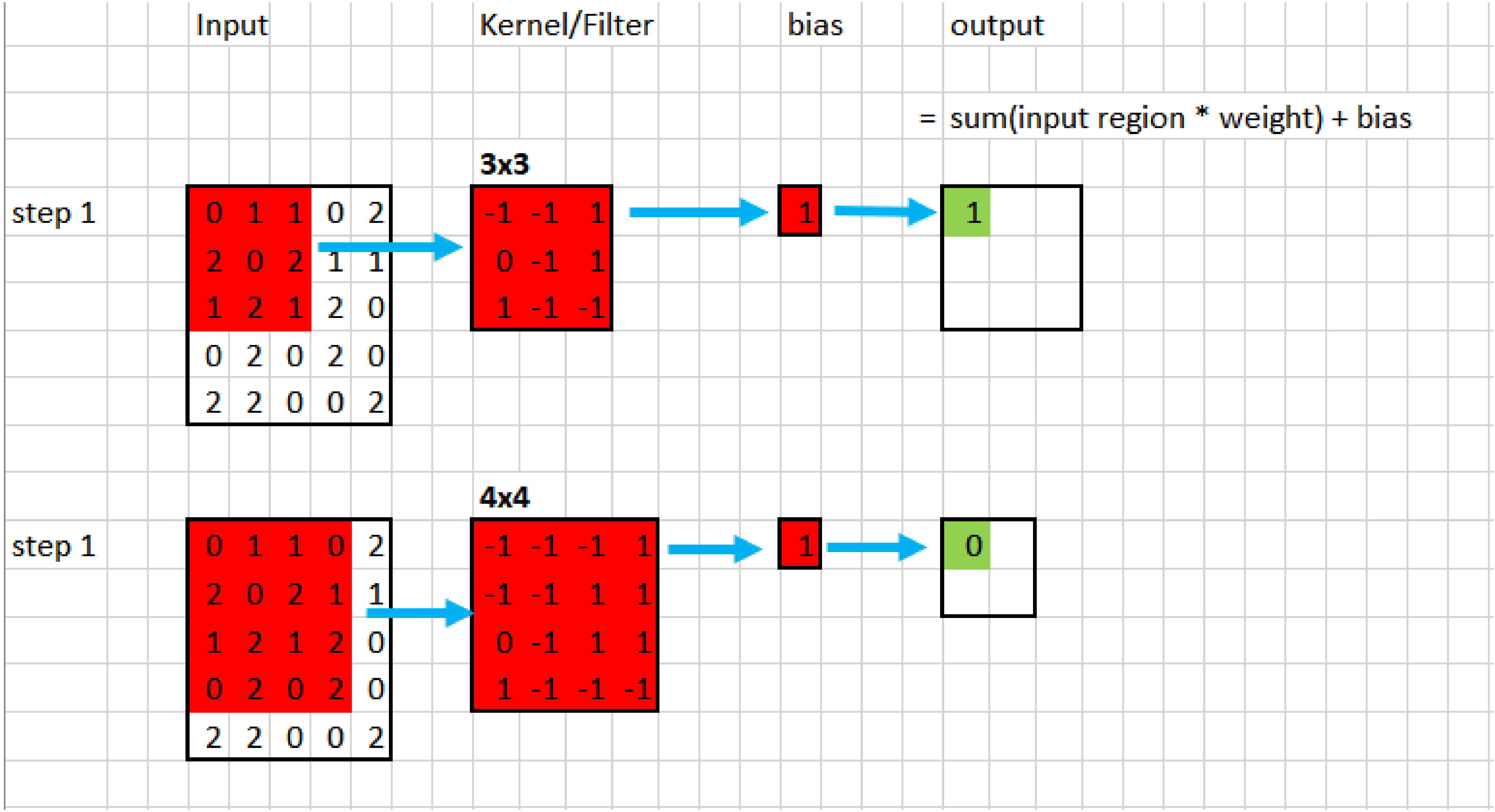
Conv – Hyperparameters

- Kernel/Filter size F
- Number of filters K
- Stride distance S
- amount of zero padding P

Conv – Kernel size, F

- Represents the spatial extent of the given connectivity
- Determines the amount of information of a local region that a kernel will extract

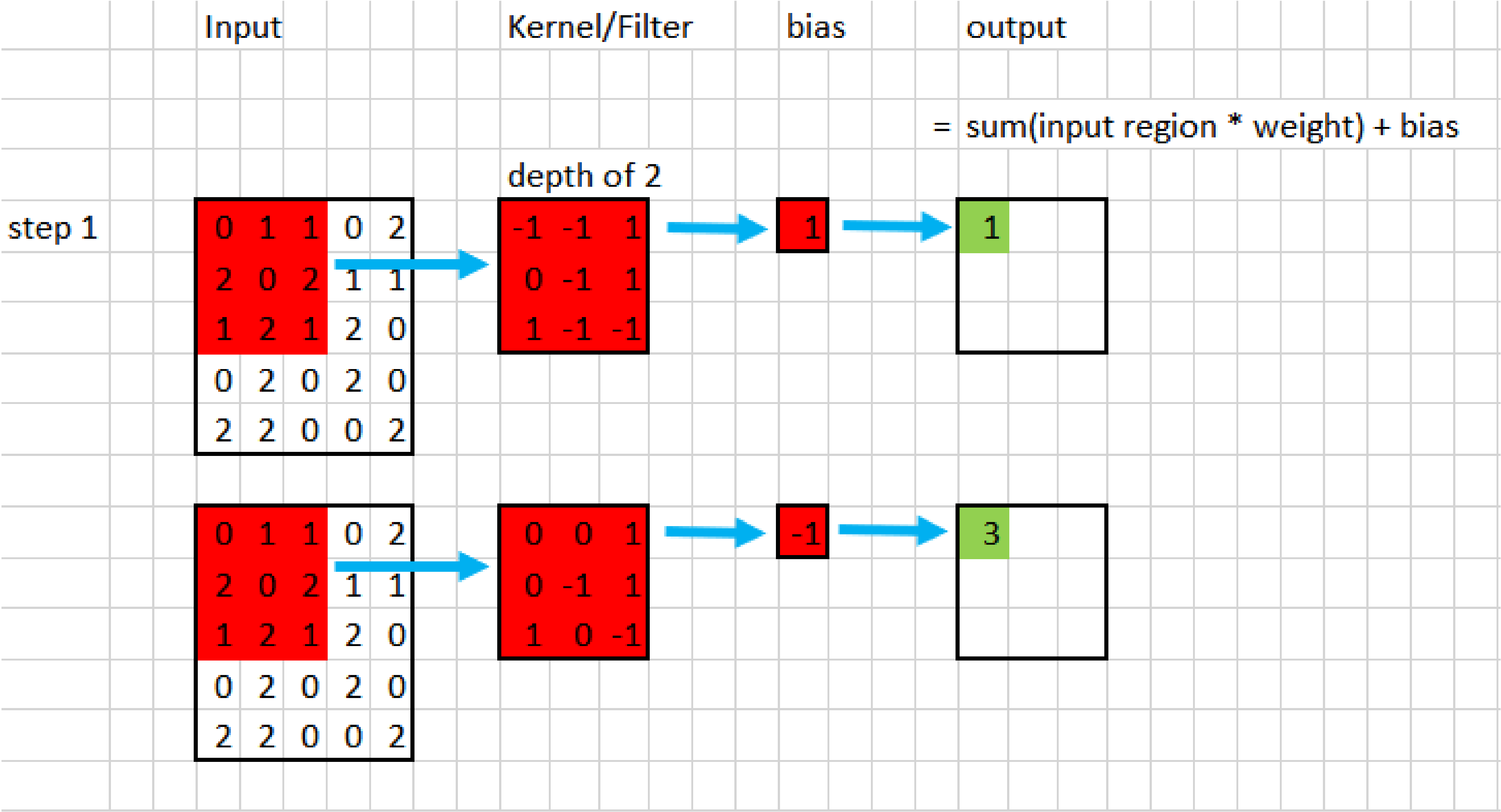
Conv – Kernel size, F



Conv – Depth, K

- represents number of kernels and each kernel will act as independent feature extractor

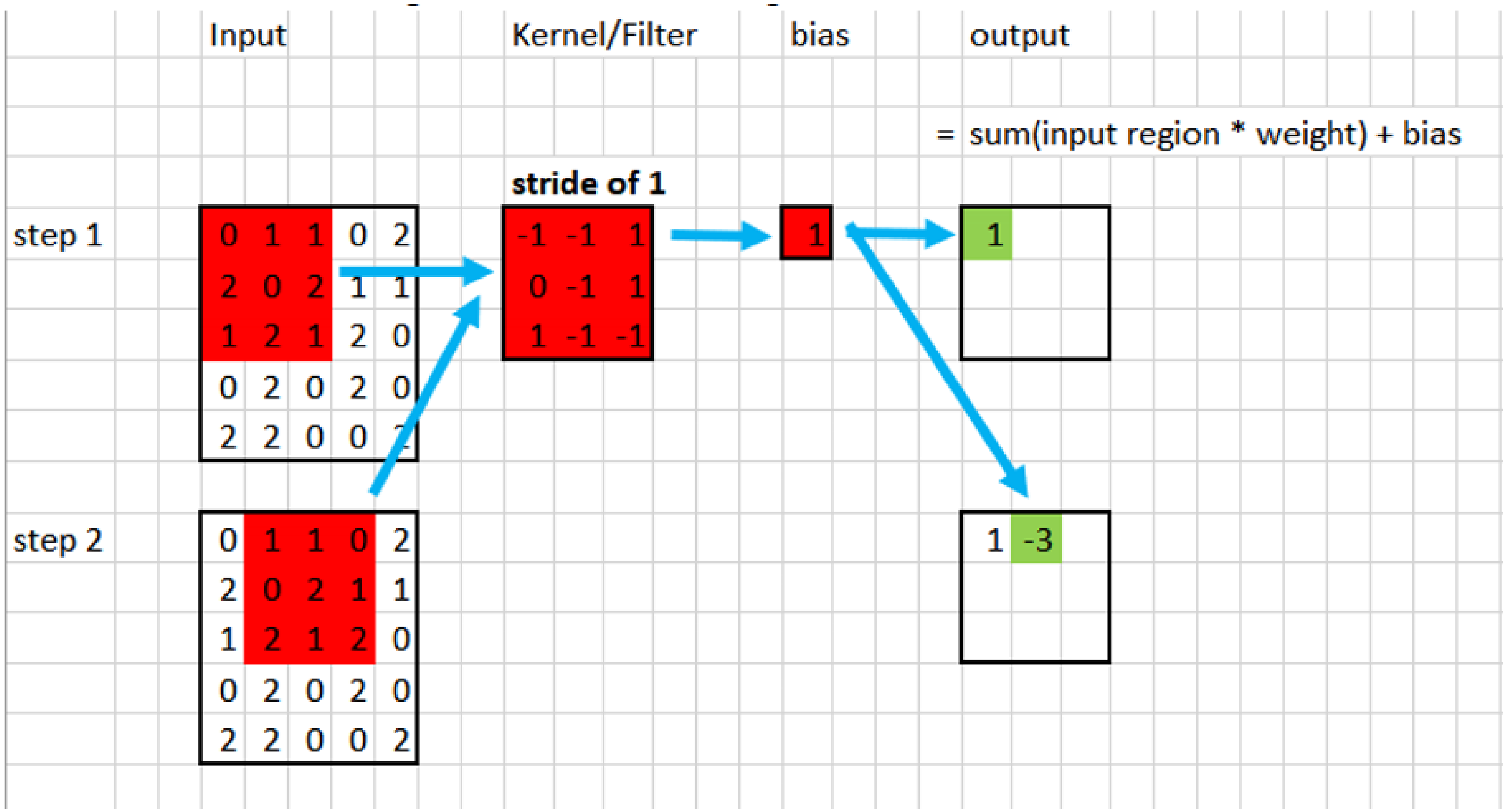
Conv – Depth, K



Input
Kernel/Filter
bias
output
= sum(input region * weight) + bias
depth of 2
step 1
0 1 1 0 2
2 0 2 1 1
1 2 1 2 0
0 2 0 2 0
2 2 0 0 2
-1 -1 1
0 -1 1
1 -1 -1
1
1
0 1 1 0 2
2 0 2 1 1
1 2 1 2 0
0 2 0 2 0
2 2 0 0 2
0 0 1
0 -1 1
1 0 -1
-1
3

Conv – Stride, S

- the distance when moving the kernels in the image



Conv – Stride, S

stride of 2

step 1

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 2 \\ 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 2 & 2 & 0 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 1 \\ 0 & -1 & 1 \\ 1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 \end{bmatrix} \quad \begin{bmatrix} 1 & \\ & \end{bmatrix}$$

step 2

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 2 \\ 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 2 & 2 & 0 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 1 \\ 0 & -1 & 1 \\ 1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ & \end{bmatrix}$$

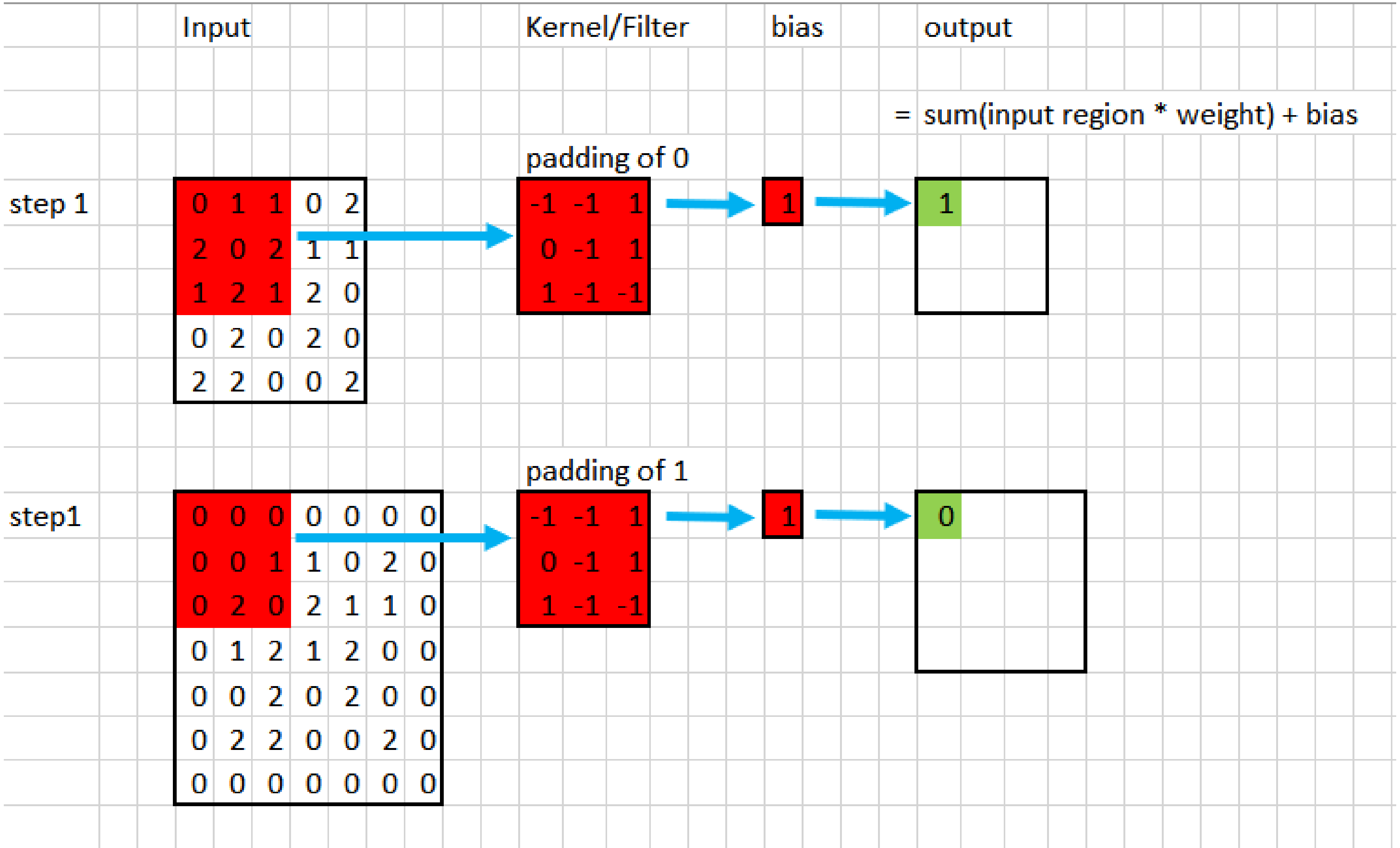
step 3

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 2 \\ 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 2 & 2 & 0 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 1 \\ 0 & -1 & 1 \\ 1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ -4 & \end{bmatrix}$$

Conv – Padding, P

- To address reduce image size after convolution
- To capture more sample from the side of image

Conv – Padding, P



Conv – Padding, P

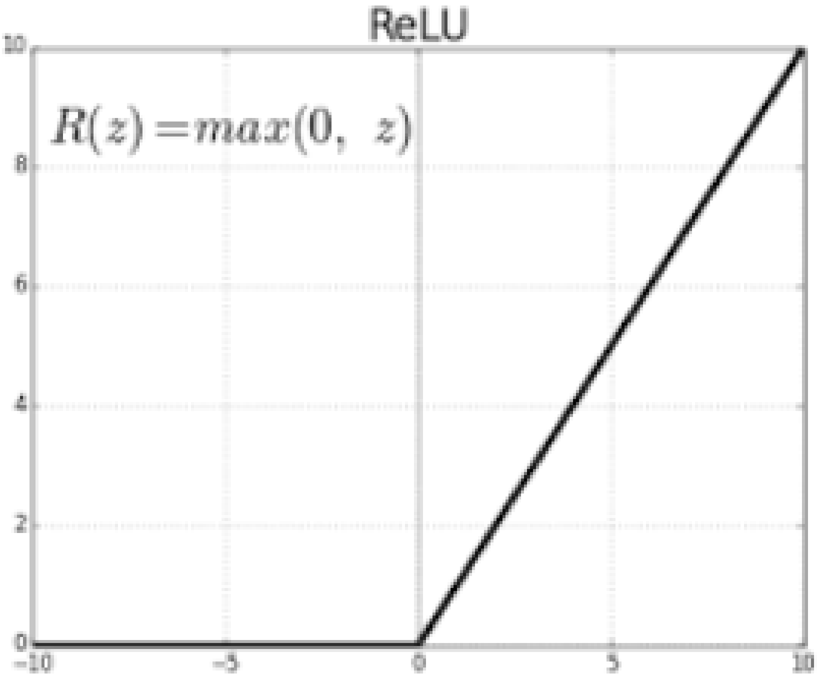
- Padding type:
- "VALID": drops the right-most columns (or bottom-most rows) if the kernel size doesn't fit to the additional pixels.
- "SAME": pads evenly left and right, this will add extra column on the right if the number of columns is odd so that every pixel is covered regardless of the kernel size.

Conv – Summary

- Given an input image of size W x H x D
- After passing through convolution with hyperparameters
  - Number of kernels, K
  - Kernel size, F
  - Stride, S
  - Amount of padding, P
- Will produce and output of dimension W_o x H_o x D_o
  - $W_o = (W – F + 2P) / S + 1$
  - $H_o = (H – F + 2P) / S + 1$
  - $D_o = K$

ReLU (rectified linear unit)

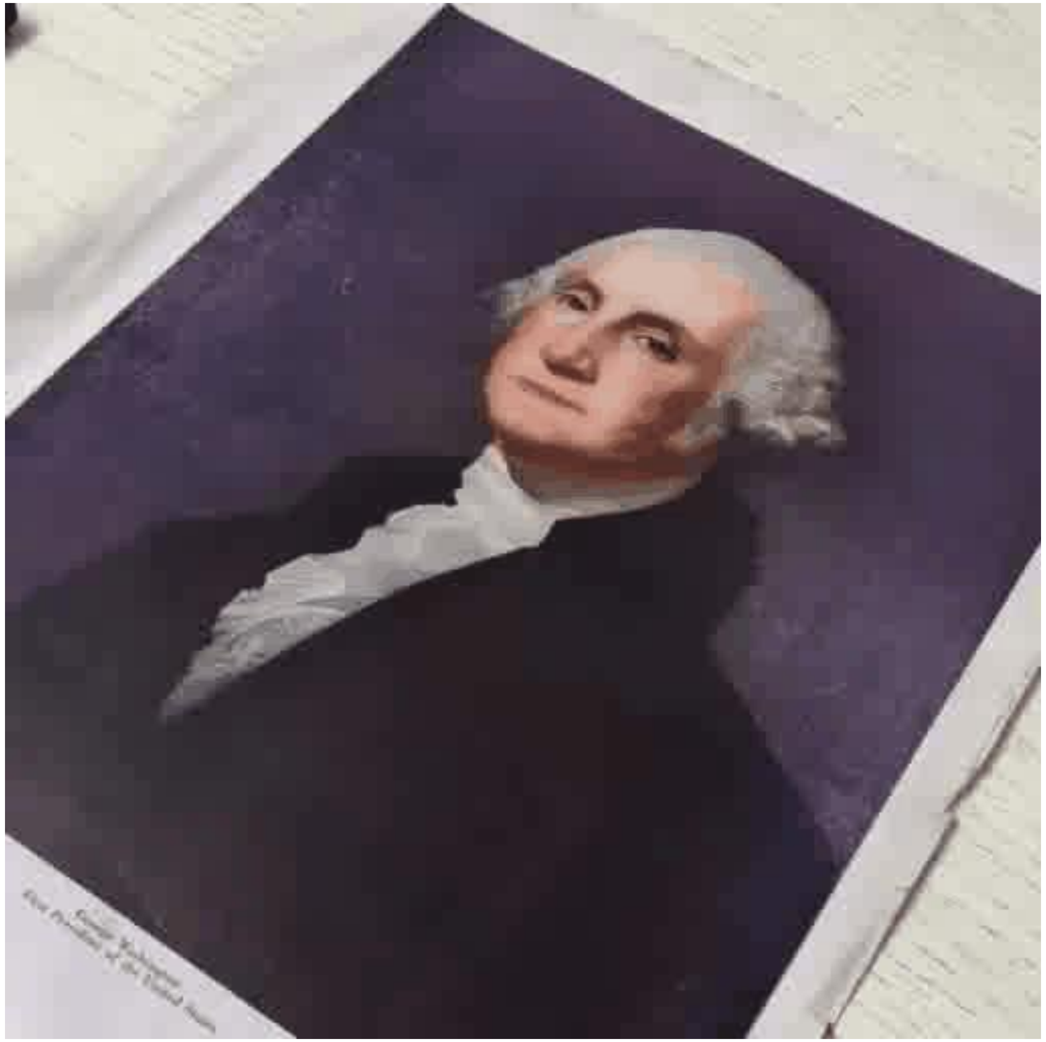
- activation layer using ReLU function



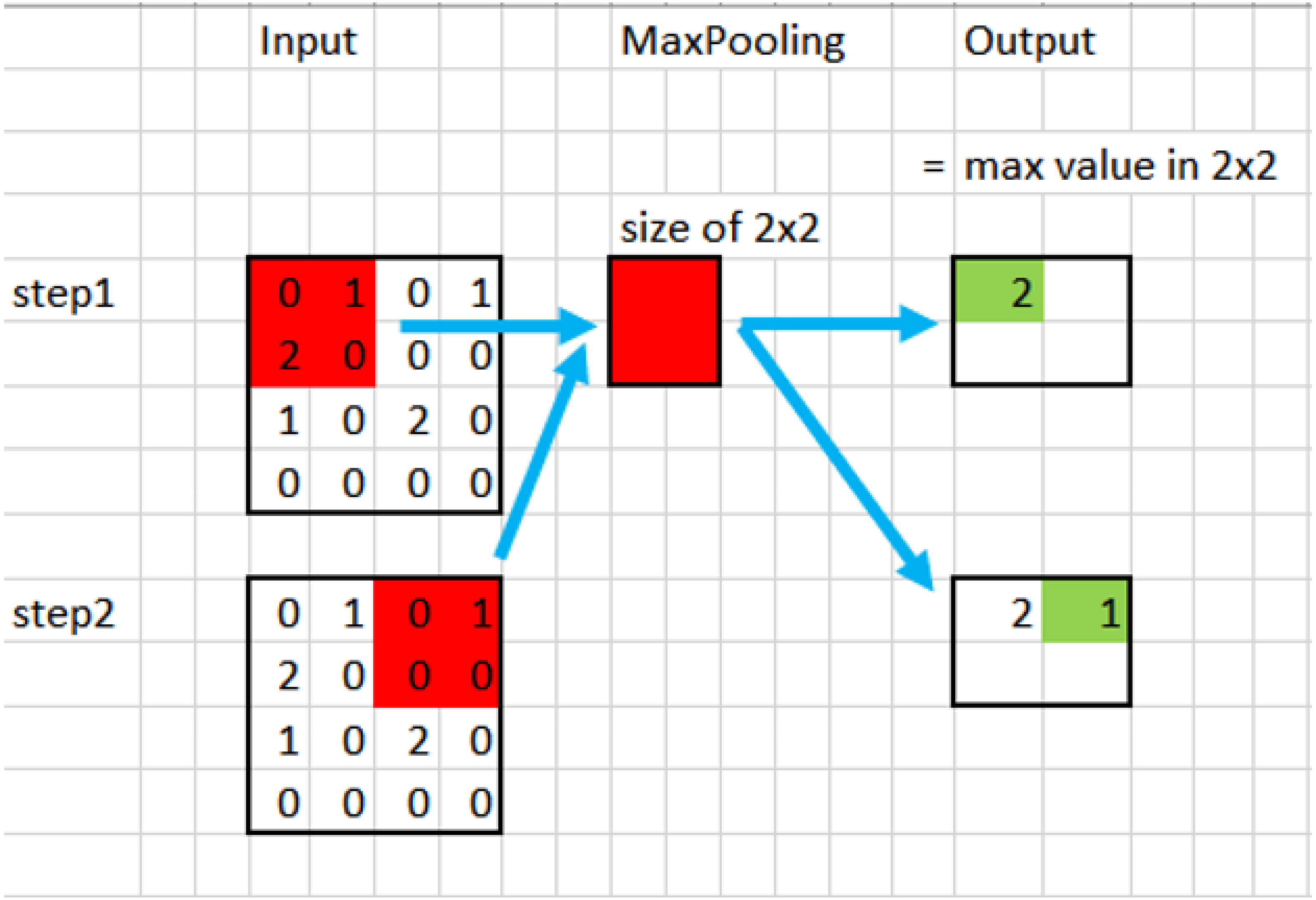
POOL (Pooling)

- This performs a down sampling operation on every kernel
- This layer doesn't have parameters

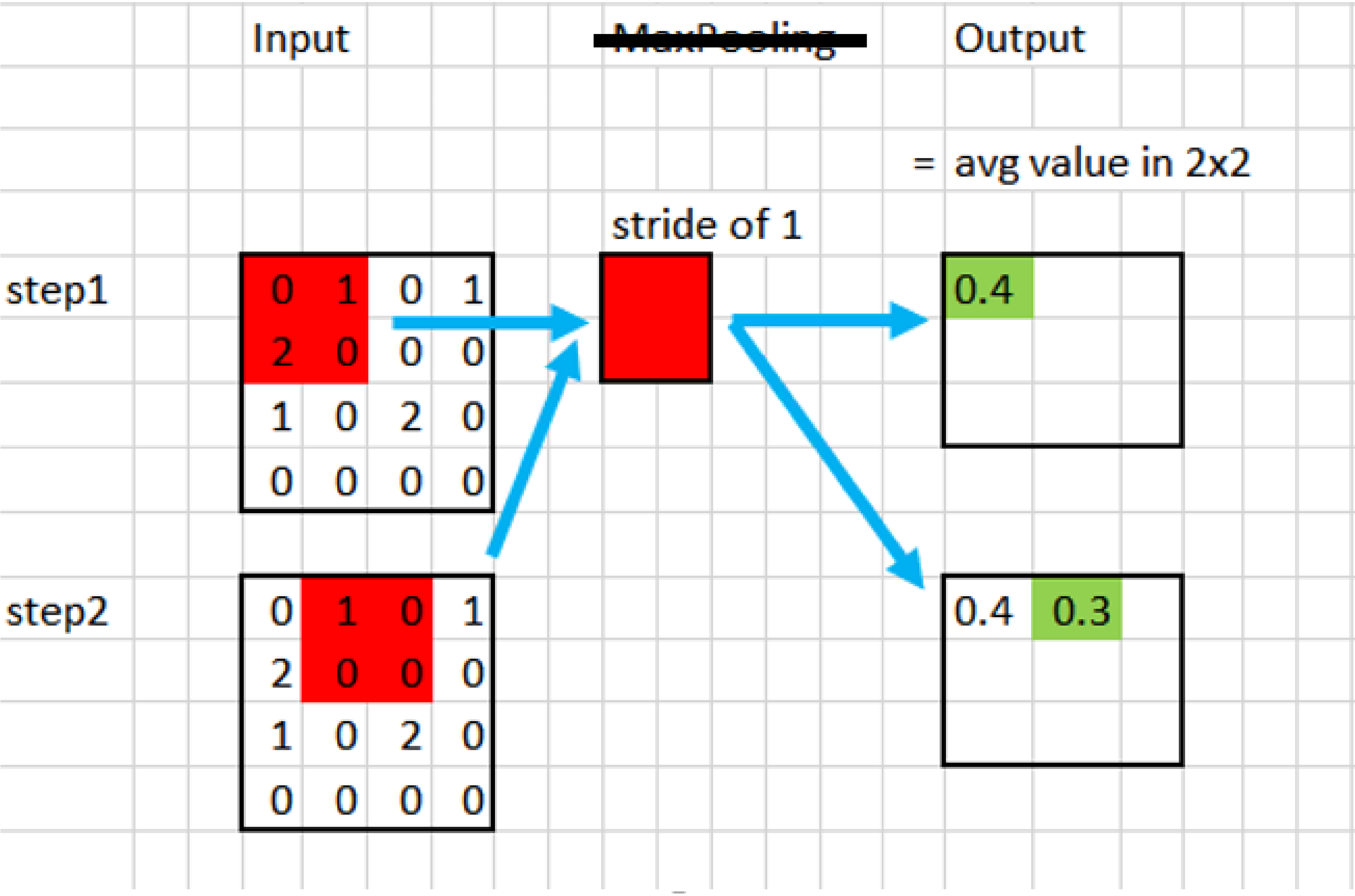
POOL - example



POOL – MaxPooling

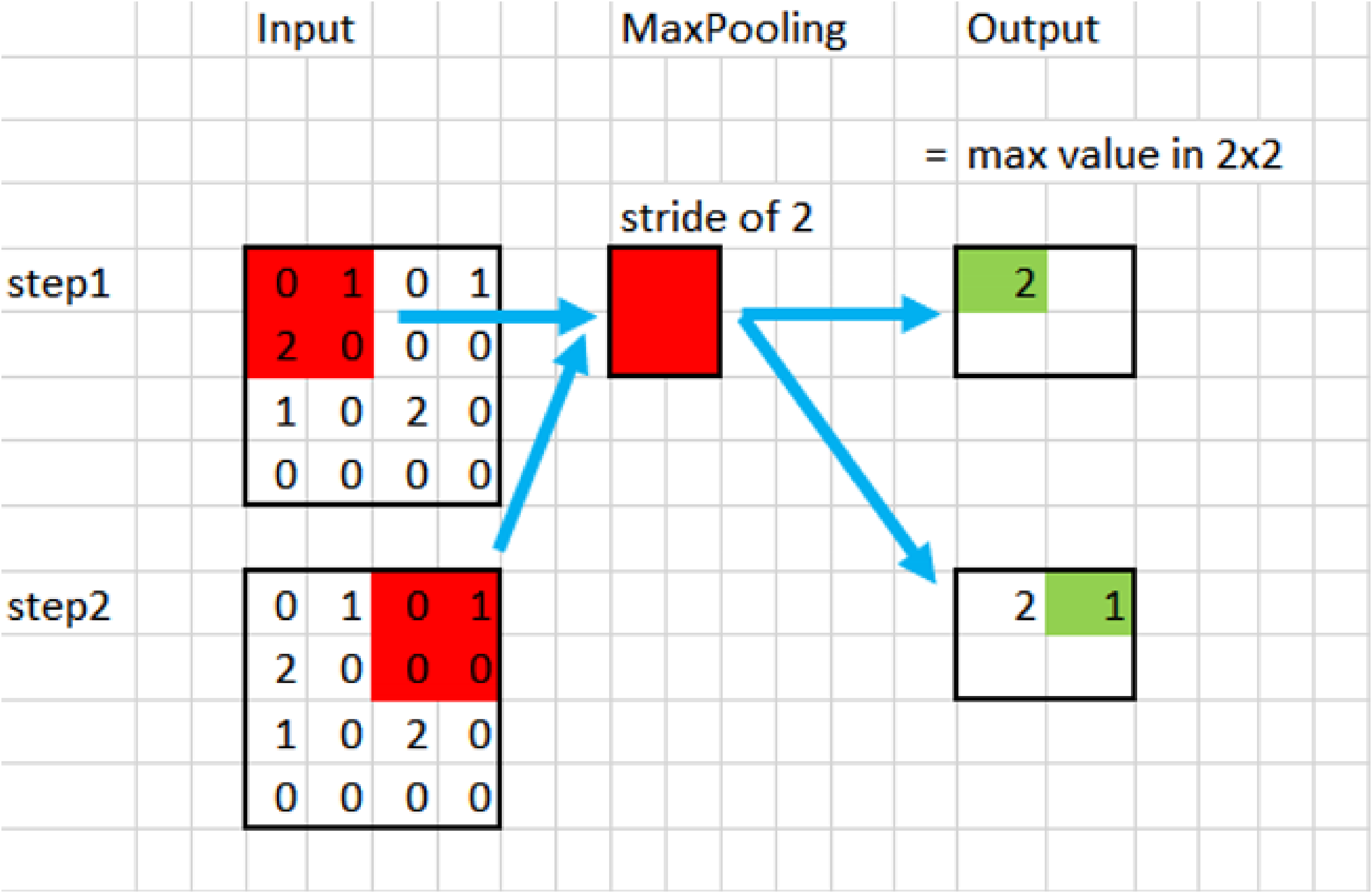


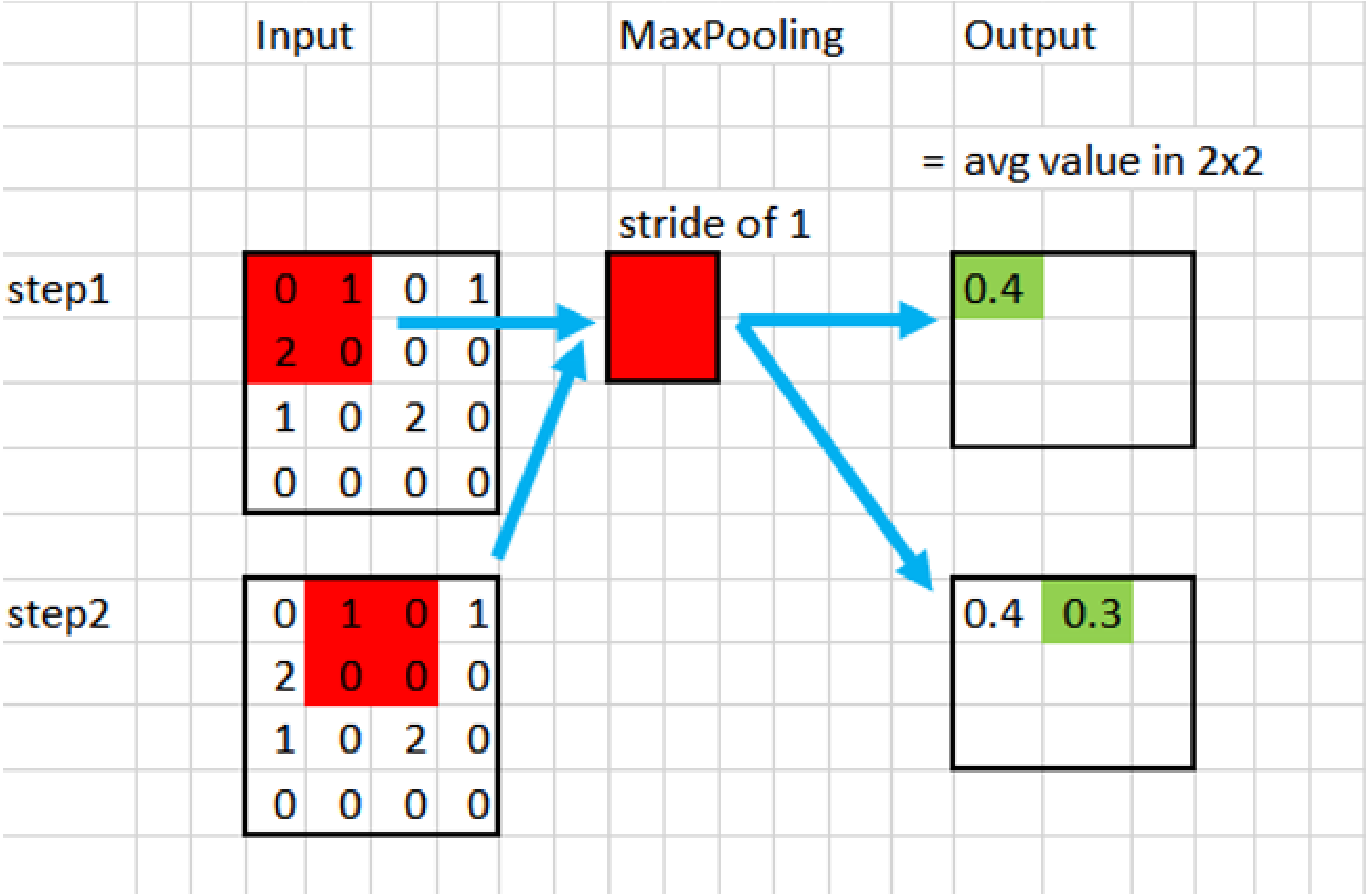
POOL - AvgPooling



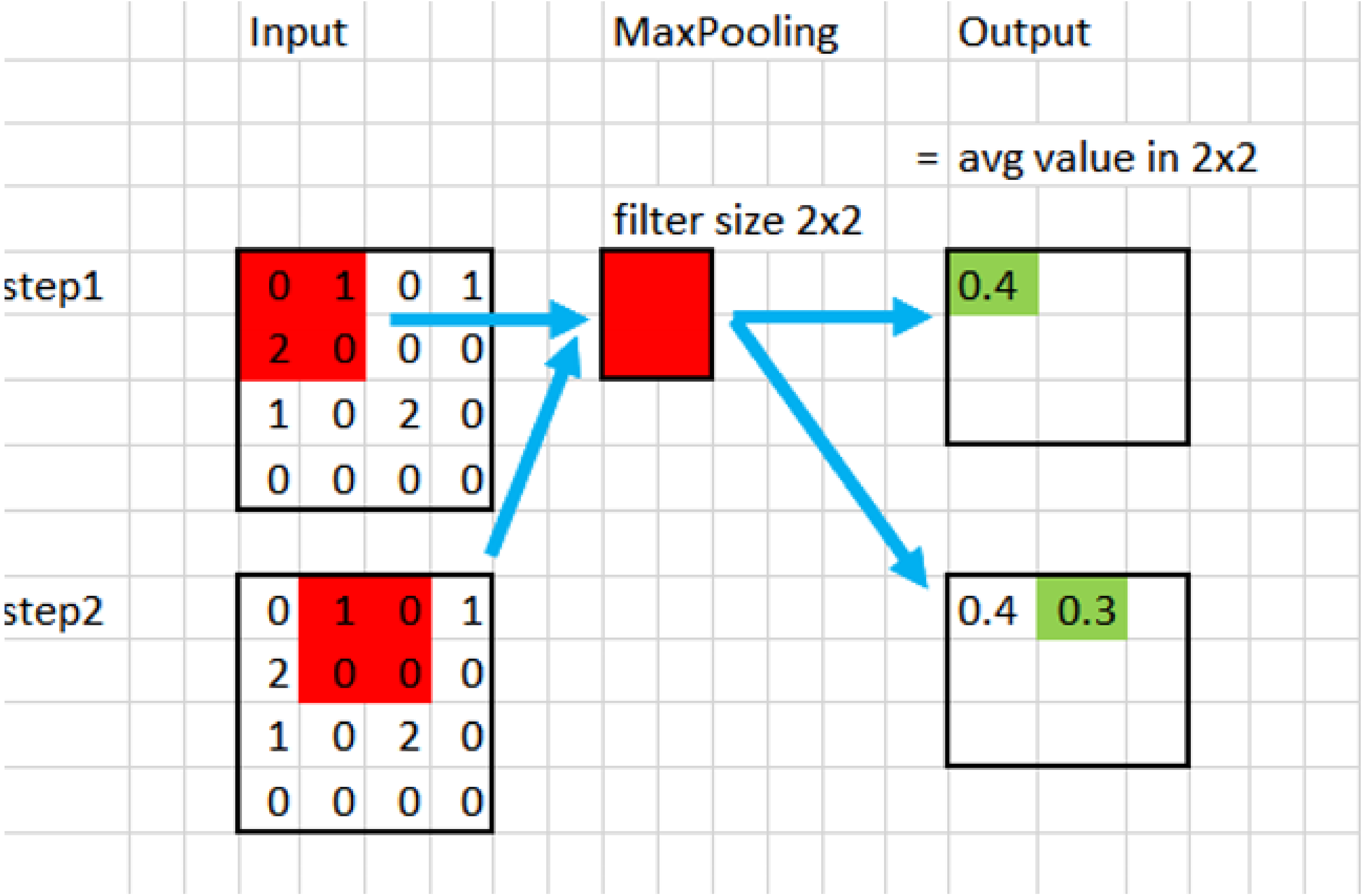
POOL, stride distance, S

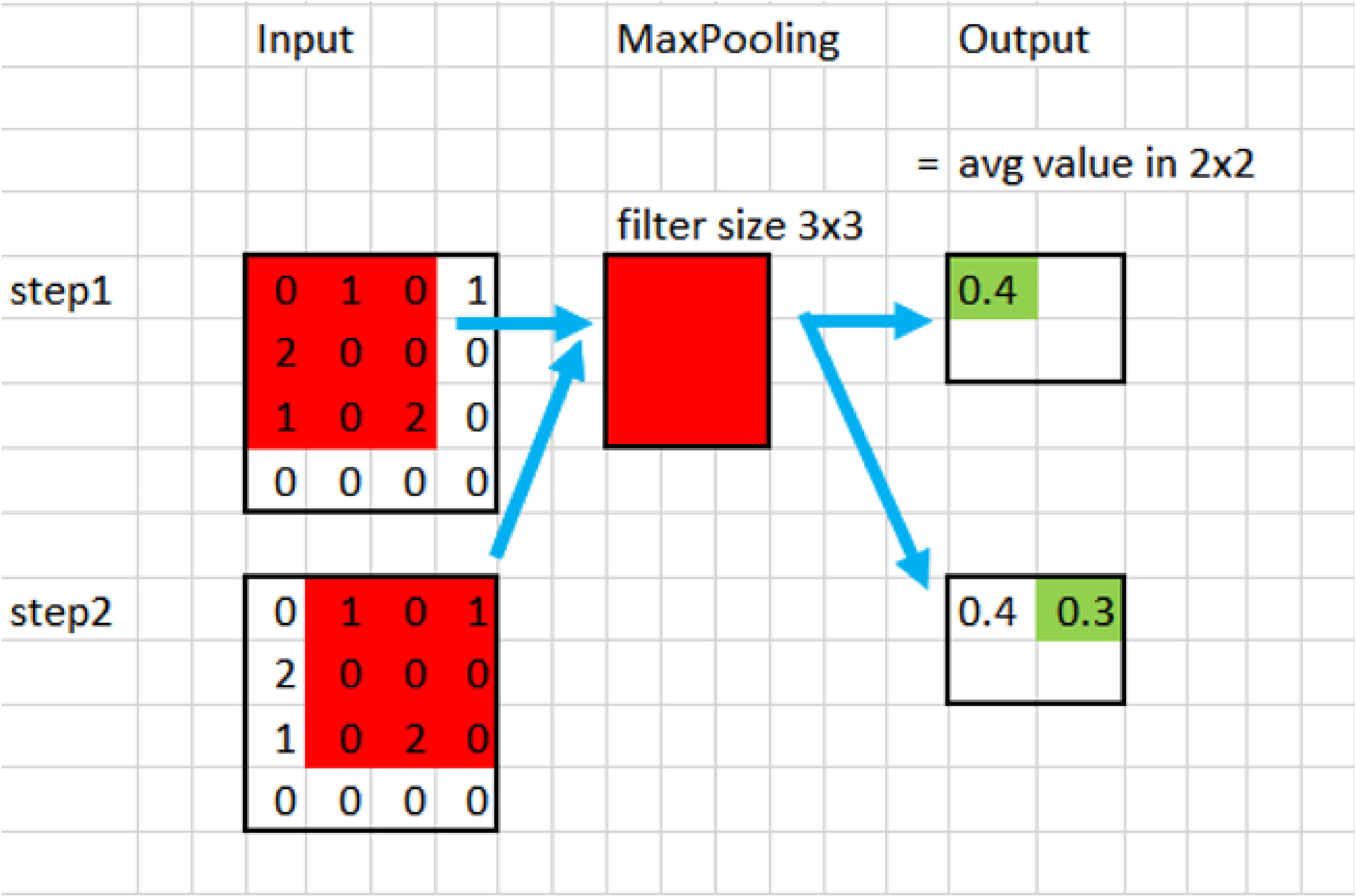
- the distance when moving kernel through the image




Input
MaxPooling
Output
= avg value in 2x2
stride of 1
step1
0 1 0 1
2 0 0 0
1 0 2 0
0 0 0 0
0.4
step2
0 1 0 1
2 0 0 0
1 0 2 0
0 0 0 0
0.4 0.3

POOL, pooling filter size, F



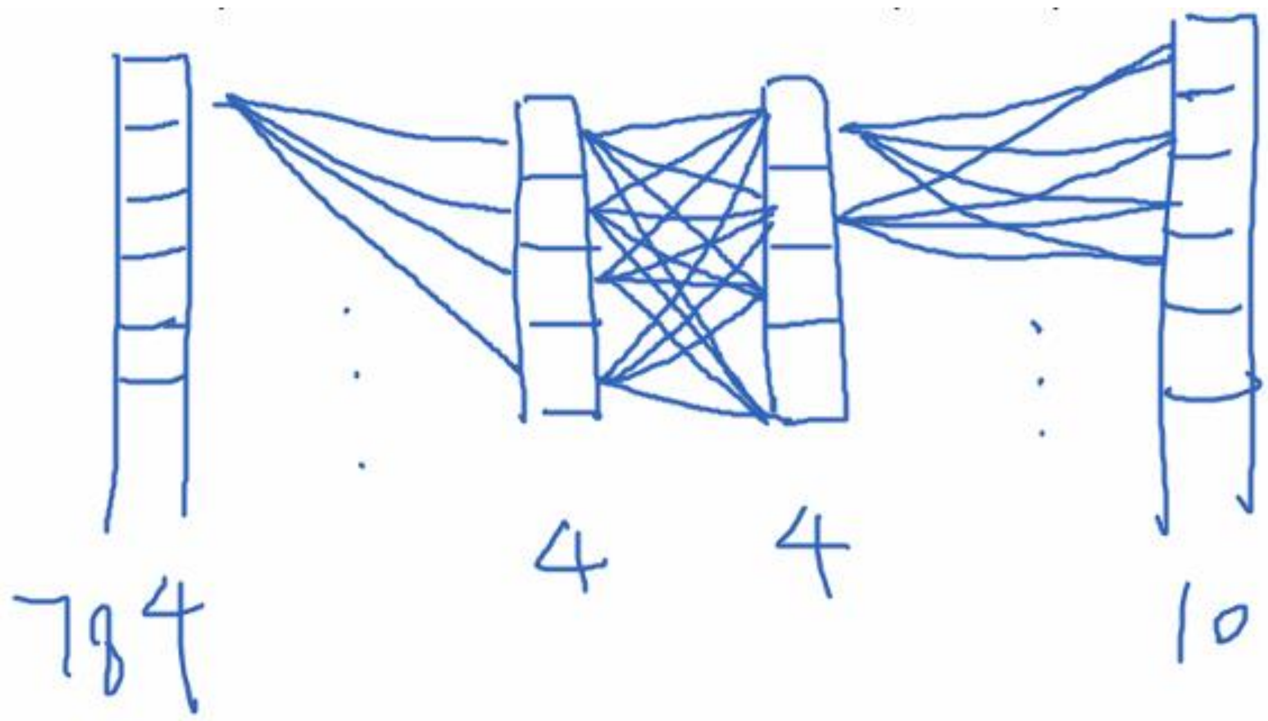

Input
MaxPooling
Output
= avg value in 2x2
filter size 3x3
step1
0 1 0 1
2 0 0 0
1 0 2 0
0 0 0 0
0.4
step2
0 1 0 1
2 0 0 0
1 0 2 0
0 0 0 0
0.4 0.3

POOL - Summary

- Given an input image of size W x H x D
- After passing through pooling with hyperparameters
  - Kernel size, F
  - Stride, S
- Will produce and output of dimension W_o x H_o x D_o
  - $W_o = (W – F) / S + 1$
  - $H_o = (H – F) / S + 1$
  - $D_o = D$

FC – Fully connected layer

- computes the class score for classification



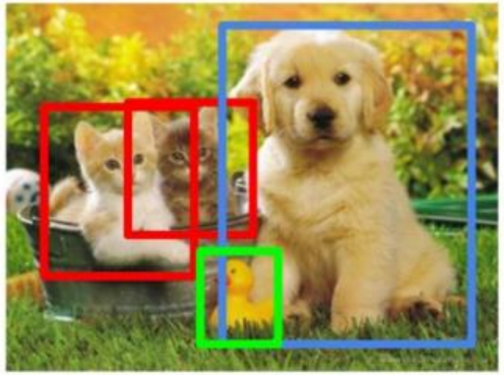
Convolutional Neural Network

- Because of parameter sharing, and sparsity of connections,
- CNN reduces the number of parameters thus reduce the potential of overfitting
- and since CNN doesn't strongly affect by the position of object in the picture, it increase the accuracy and robustness of detecting object.
- visualisation: http://scs.ryerson.ca/~aharley/vis/conv/

CNN in Computer Vision

- Classification
- Object detection
- Segmentation
- Optical Character Recognition
- Facial Recognition
- ...

CAT



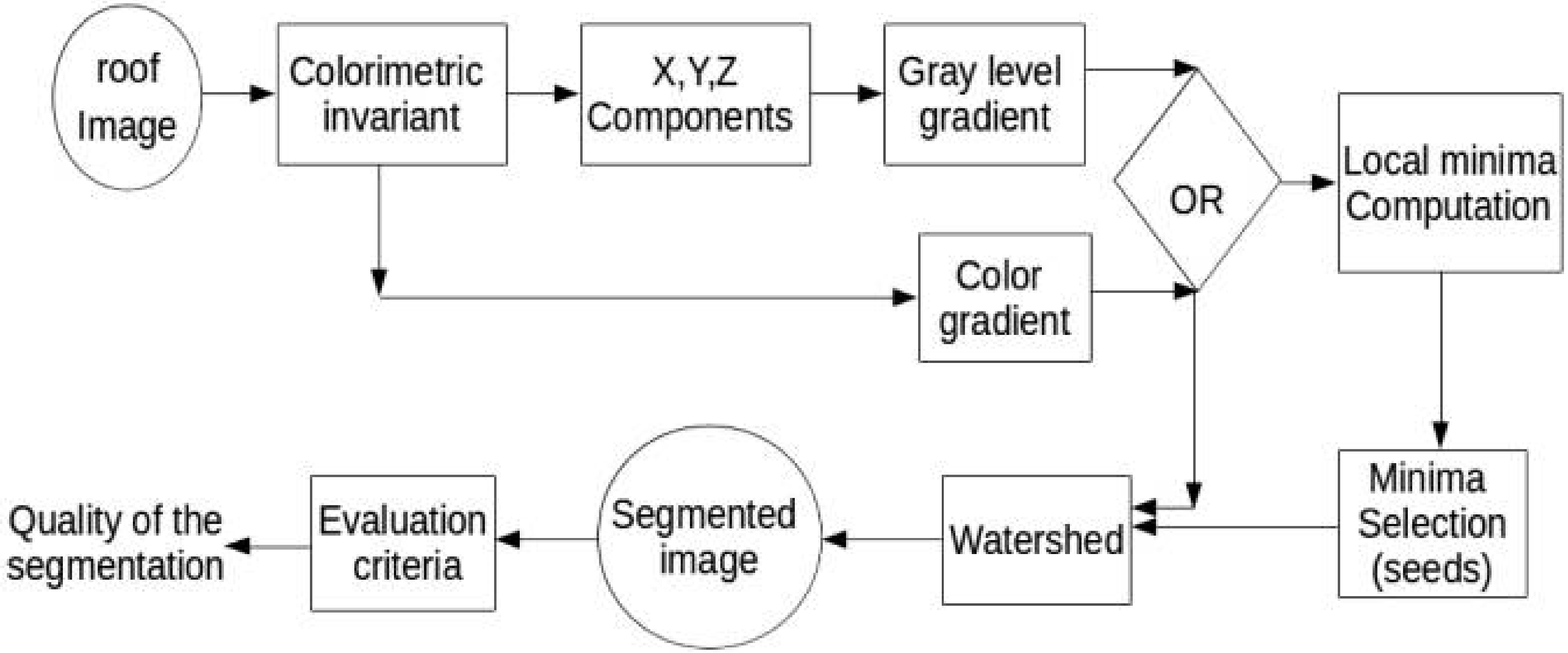
CAT, DOG, DUCK

CAT, DOG, DUCK

Traditional way

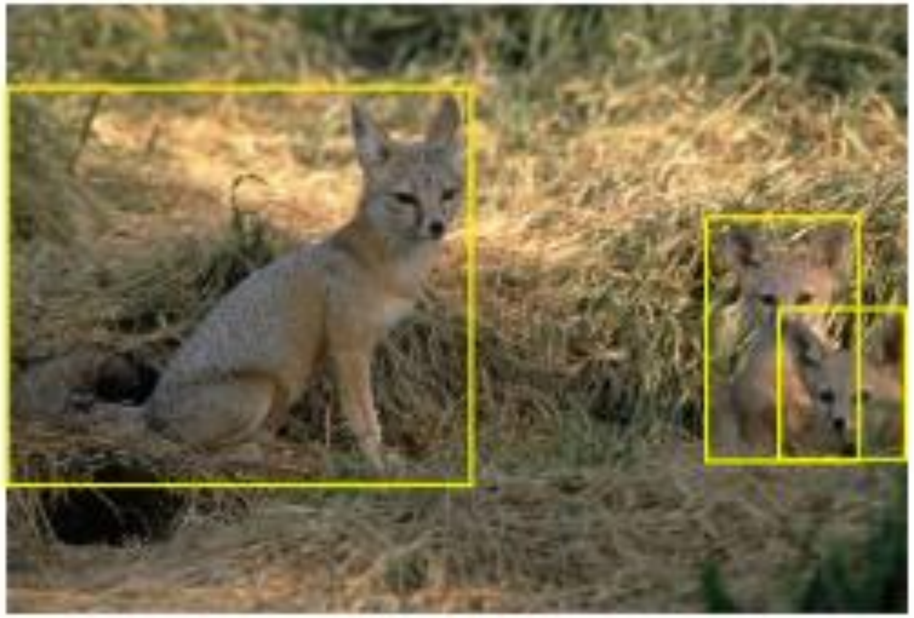
- Merabet et al, 2015. Building Roof Segmentation from Aerial Images Using a Line-and Region-Based Watershed Segmentation Technique

(a) (b) (c) (d) (e)


roof
Image
Colorimetric
invariant
X,Y,Z
Components
Gray level
gradient
OR
Local minima
Computation
Color
gradient
Minima
Selection
(seeds)
Watershed
Segmented
image
Evaluation
criteria
Quality of the
segmentation

ImageNet

- A large visual database led by Fei-Fei Li
- Contains > 14million images with hand-annotated labels



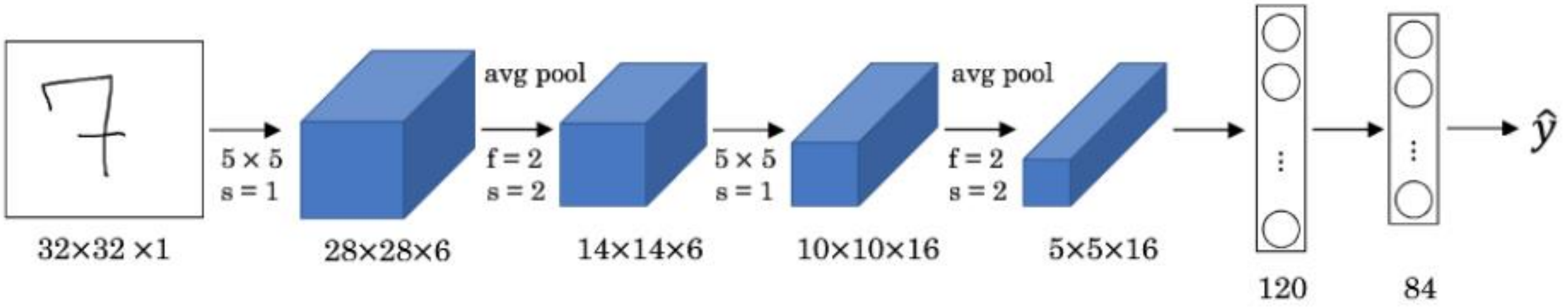
kit fox

CNN – Evolution (Classification)

- LeNet 1998
- Alexnet 2012
- ZFNet 2012
- Inception 2014
- VGG 2015
- Resnet 2015

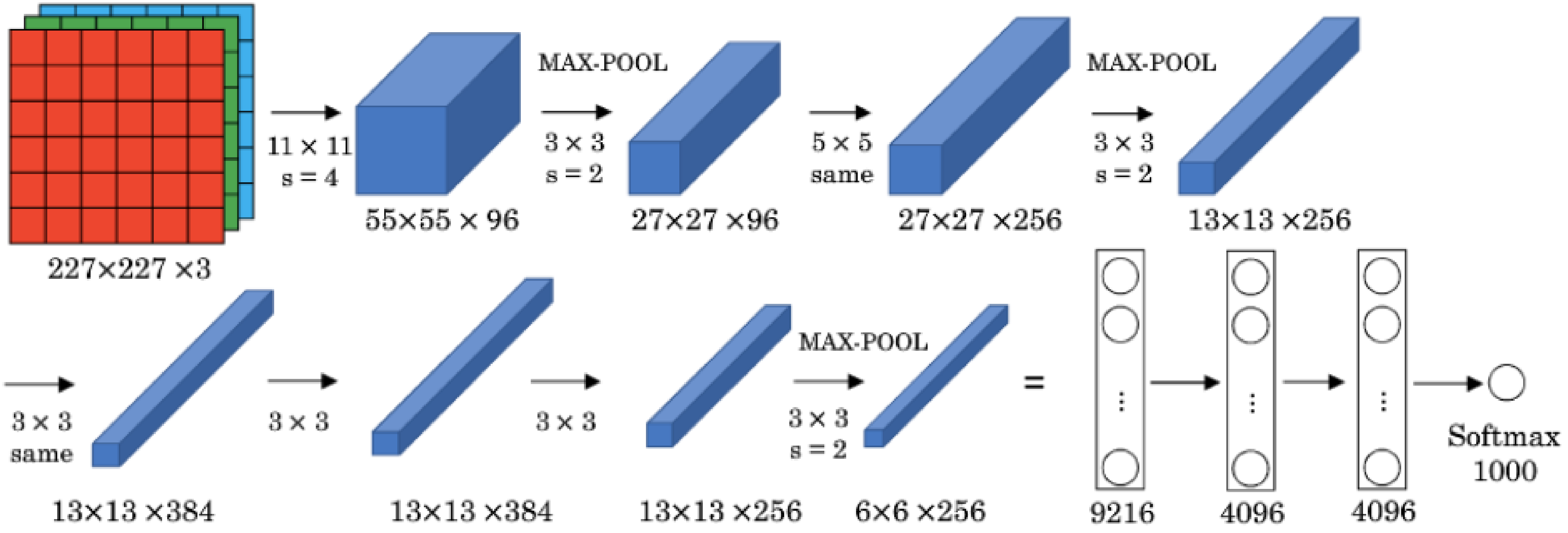
LeNet

- LeCun et.al., 1998. Gradient-based learning applied to document recognition



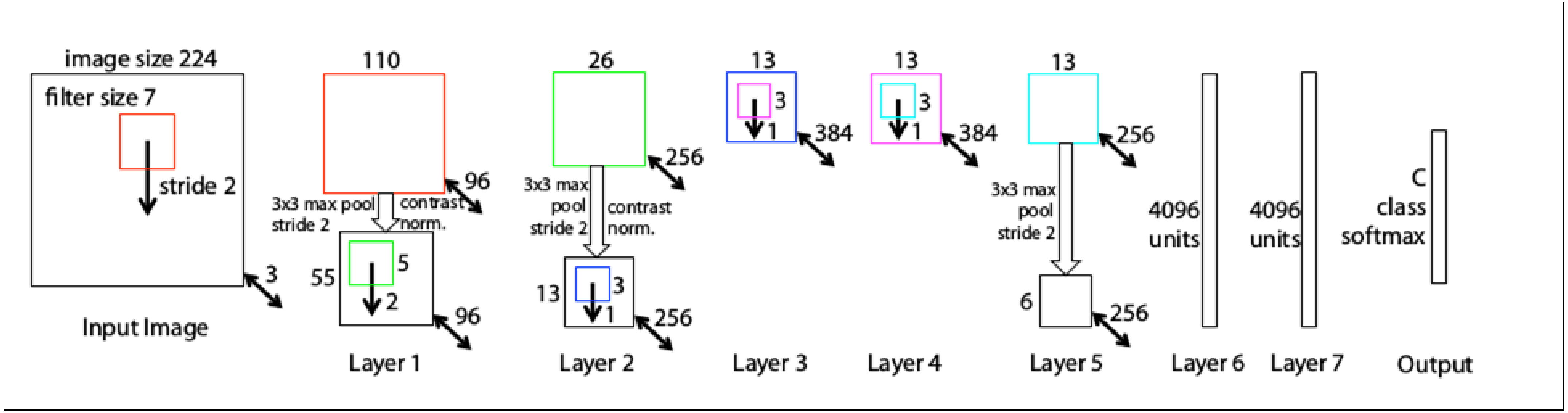
Alexnet

- Krizhevsky et al.,2012. ImageNet classification with deep convolutional neural networks



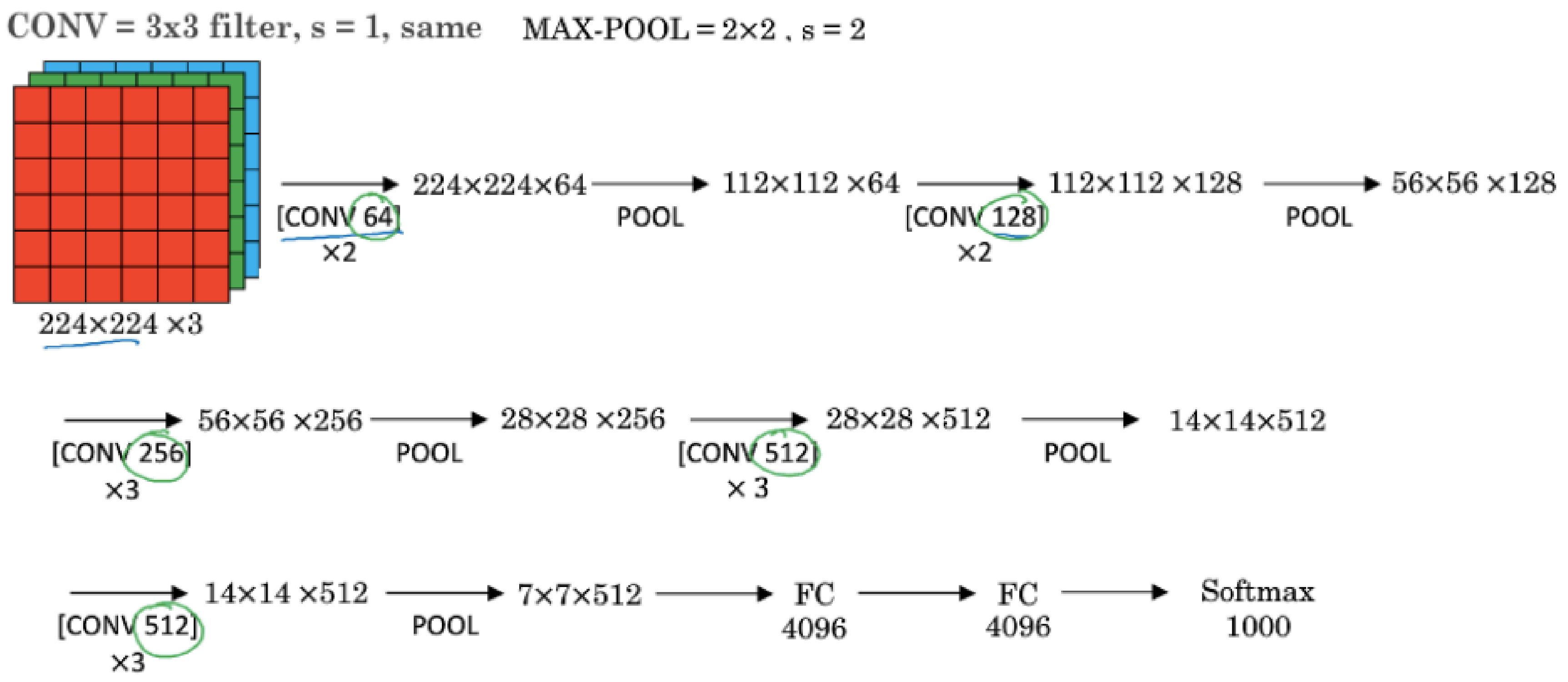
ZFNet

- Zeiler and Fergus, 2013. Visualizing and Understanding Convolutional Networks



VGG

- Simonvan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition。



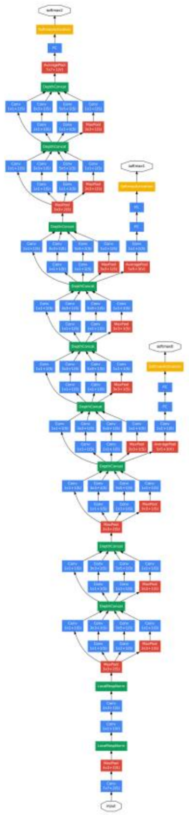
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

GoogLeNet/Inception

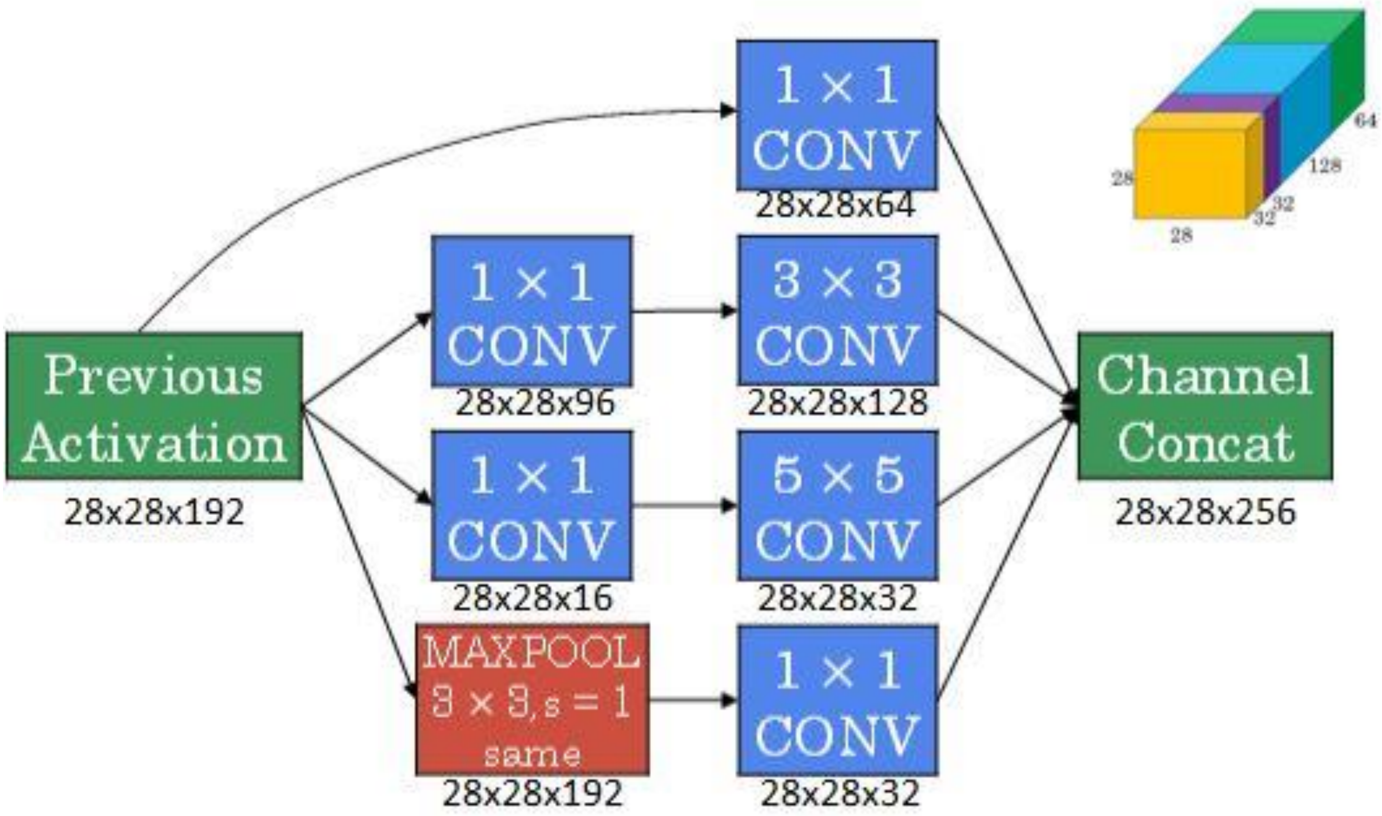
- Szegedy et al., 2014, Going Deeper with Convolutions

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

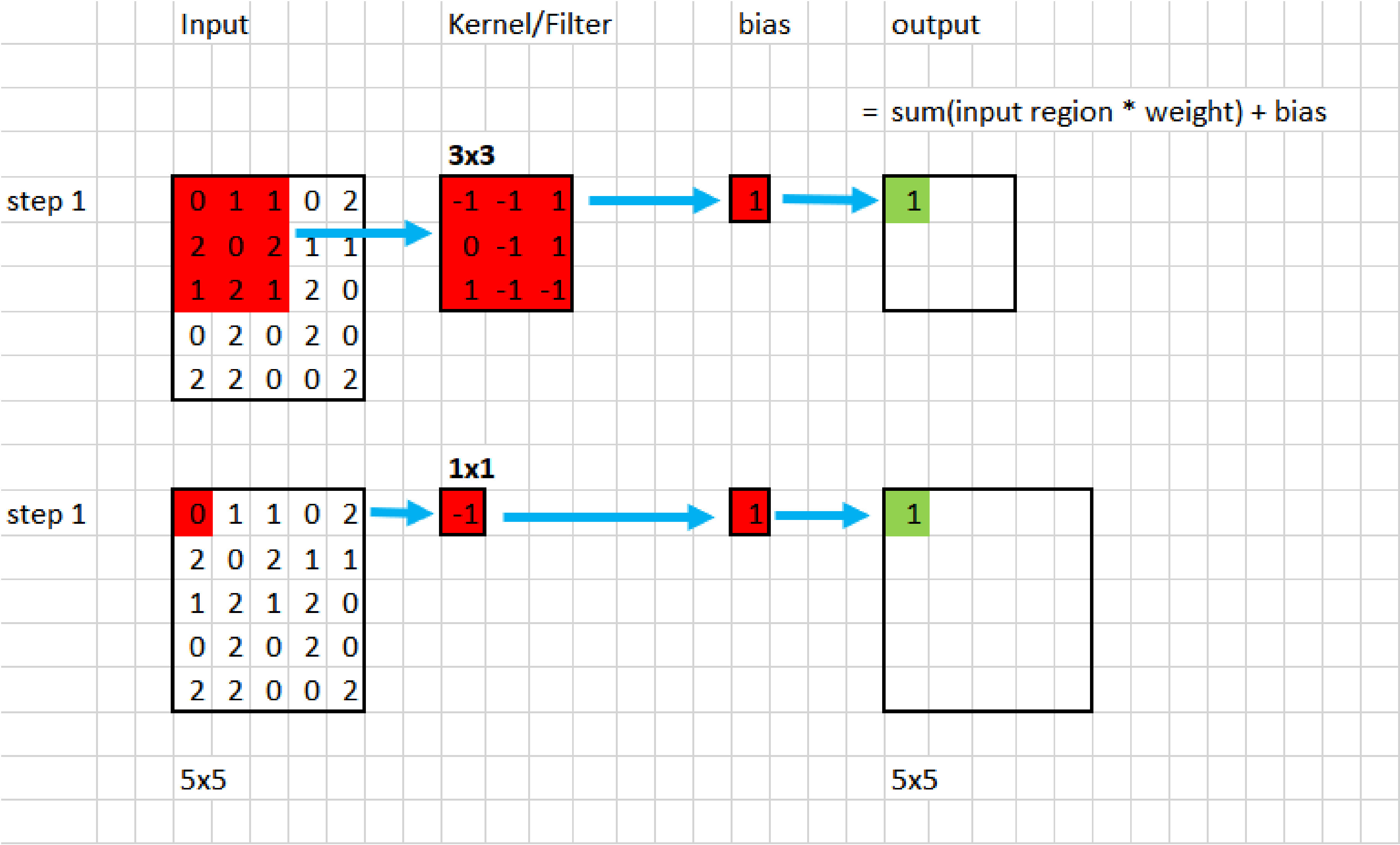


- zoom in

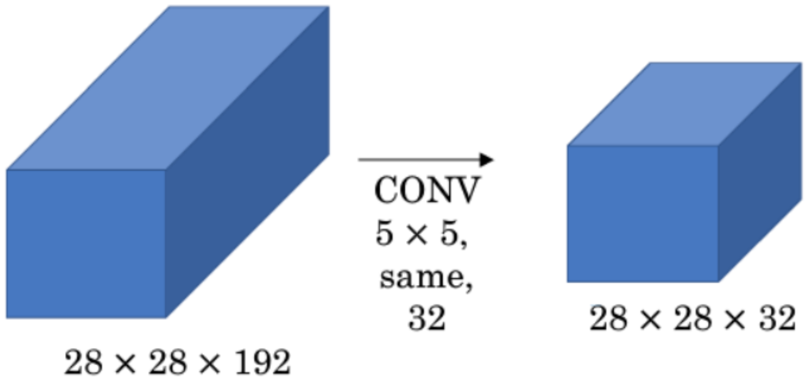


1 × 1
CONV
28x28x64
Previous
Activation
28x28x192
1 × 1
CONV
28x28x96
3 × 3
CONV
28x28x128
1 × 1
CONV
28x28x16
5 × 5
CONV
28x28x32
MAXPOOL
3 × 3, s = 1
same
28x28x192
1 × 1
CONV
28x28x32
Channel
Concat
28x28x256
64
128
28
32
32
28

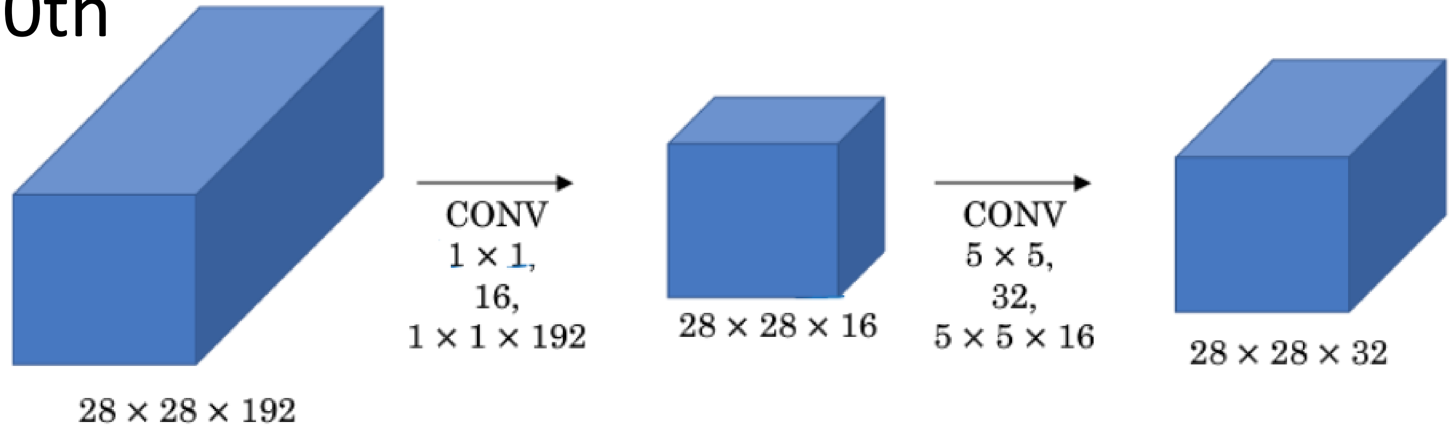
1x1 Convolution



- reduce number of parameters e.g.
- originally we have 28*28*192*5*5*32 = 120,422,400 weights



- after using 1x1 conv we have
- we have 28*28*192*1*1*16+28*28*16*5*5*32=12,443,648 about 1/10th



Resnet

- He et al, 2015. Deep Residual Learning for Image Recognition

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

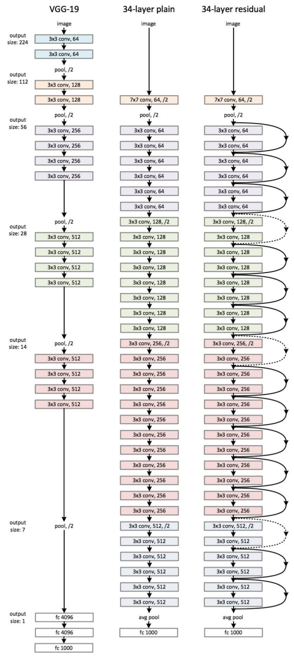
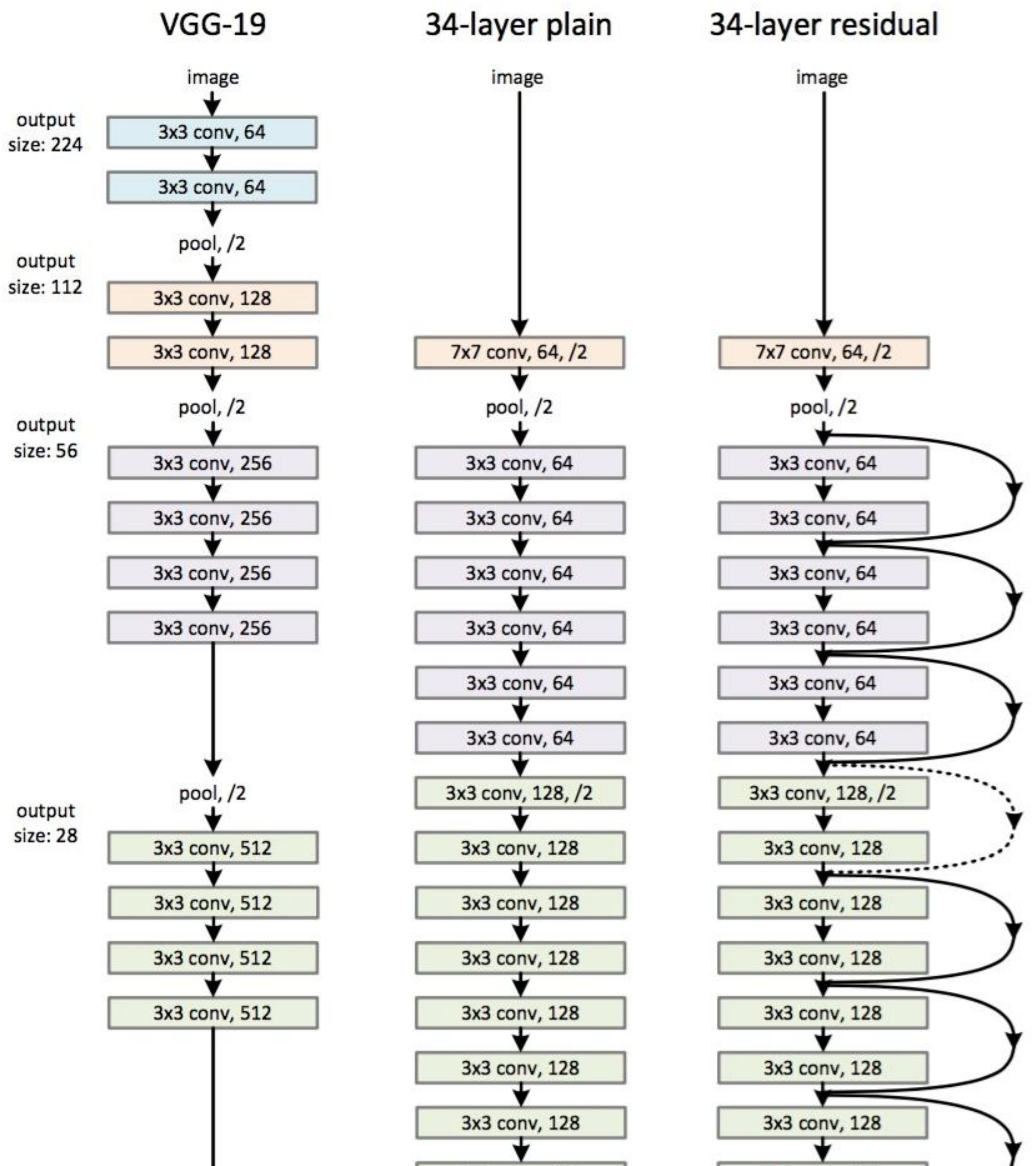
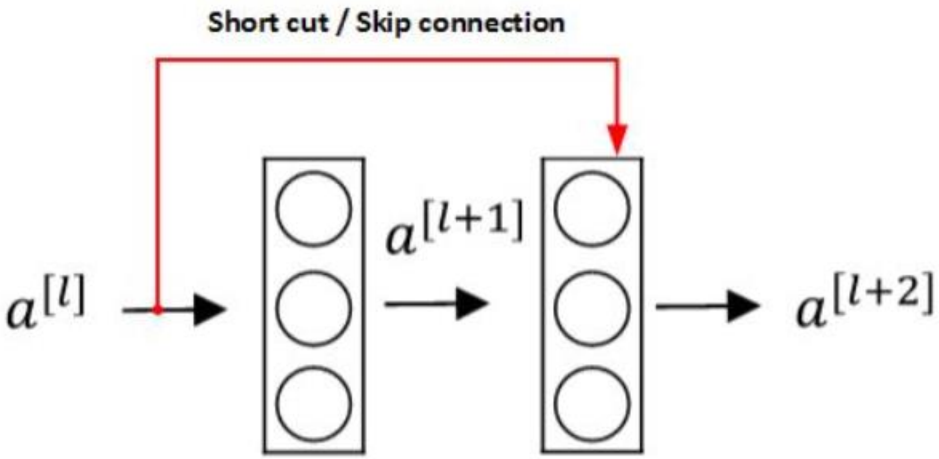


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128

Resnet – Skip connection



$$
\begin{aligned}
a^{[l+2]} &= g(z^{[l+2]} + a^{[l]}) \\
&= g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})
\end{aligned}
$$

when vanishing gradient happens
this becomes

$$a^{[l+2]} = g(a^{[l]}) = ReLU(a^{[l]}) = a^{[l]}$$

Grand Plan (besides the lecture material)

- (✓) Neural Network Foundation (update to ppt, tbd)
- (✓) Neural Network Components (gradient descent, hyperparameter, regularization) (update to ppt, tbd)
- (✓) Convolution Neural Network
- (☐) Recurrent Neural Network
- (☐) Generative Adversarial Network (+ unsupervised + symmetric nn)
- (☐) Reinforcement Learning
- (☐) Vote/Choice from students or Big data
- (☐) Vote/Choice from students or TimeSerise/Natural Language Processing

(optional) You can select a topic from the github readme page (https://github.com/tczhao/ada2018tut),
or propose a topic, or a paper/article that you want to go through,
send it to Tianchu.Zhao@uts.edu.au
I will randomly pick a few (or the most popular) and through them in the w9?/10/w11 tutorials