Convolutional Neural Network (CNN)

Rowel Atienza, PhD
University of the Philippines
github.com/roatienza
2022

Convolutional Neural Network

Convolutional Neural Network (**CNN**) or **CovNet**

Closest model on how human vision works

Uses an operation called **Convolution**

Con – Latin word for *together*

Volvere – Latin word for *roll up*

Convolution operator

$$y = x * k$$

$x \in \mathbb{R}^{w \times h \times d}$ is input: d feature maps with dimensions $w \times h$

$k \in \mathbb{R}^{k \times k \times f}$ is kernel: f filters or kernels with dimensions $k \times k$

$y \in \mathbb{R}^{w \times h \times f}$ is output: f feature maps with dimensions $w \times h$ assuming sufficient padding is applied
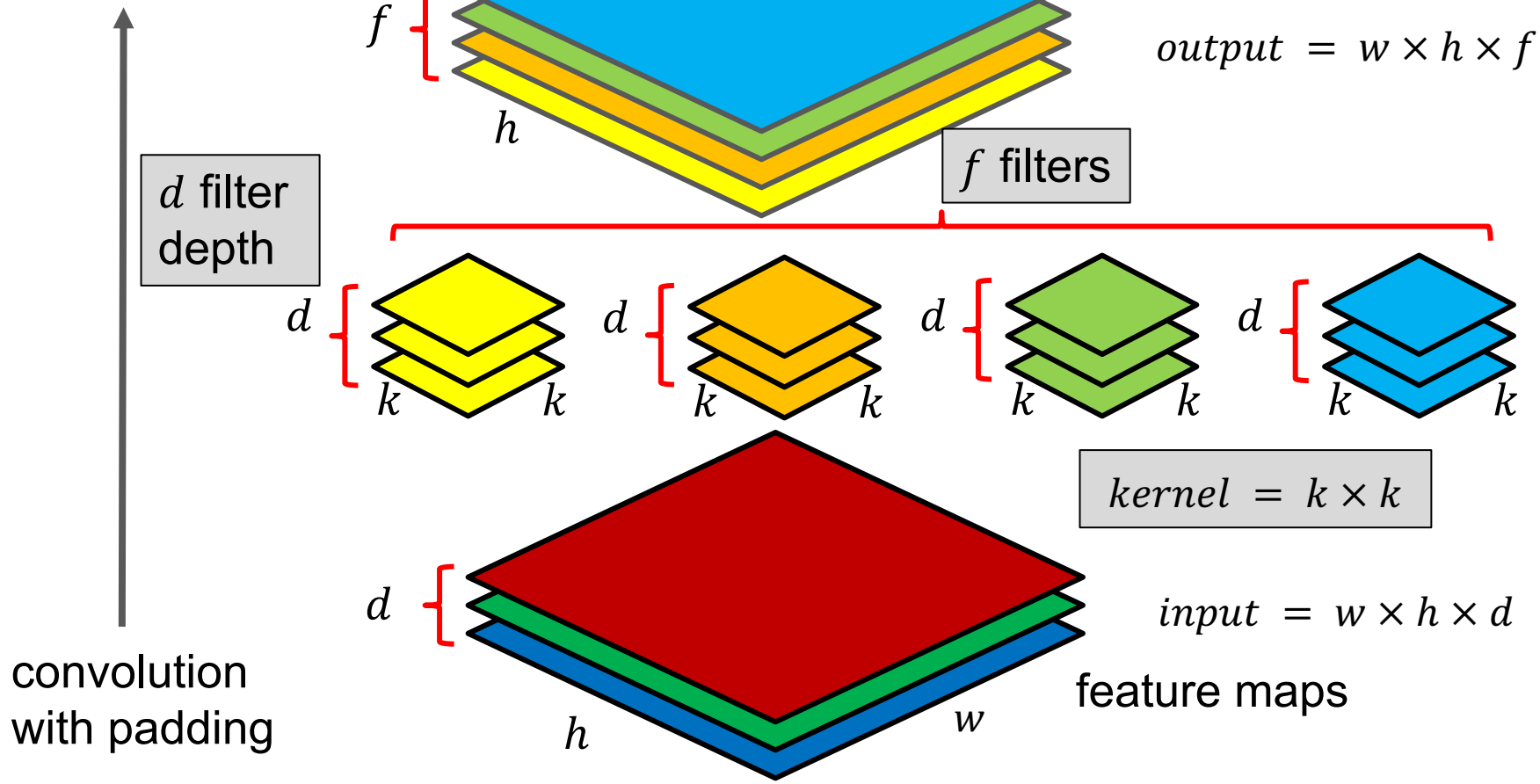
Dimensions are based on 2D inputs and square kernels

CNN Parameters: Kernel or Filter + Bias

In MLP, we learn weights and biases

In CNN, we learn the weights and bias of a kernel

CNN

feature maps

$output = w \times h \times f$

f filters

d filter depth

$kernel = k \times k$

$input = w \times h \times d$

feature maps

convolution with padding

5

A CNN is made of multiple convolutional layers

The deeper the network, the more representations the network learns

$f_1(\boldsymbol{x})$

$f_2(f_1)$

$f_3(f_2)$

conv

conv

conv

...

feature maps

feature maps

feature maps
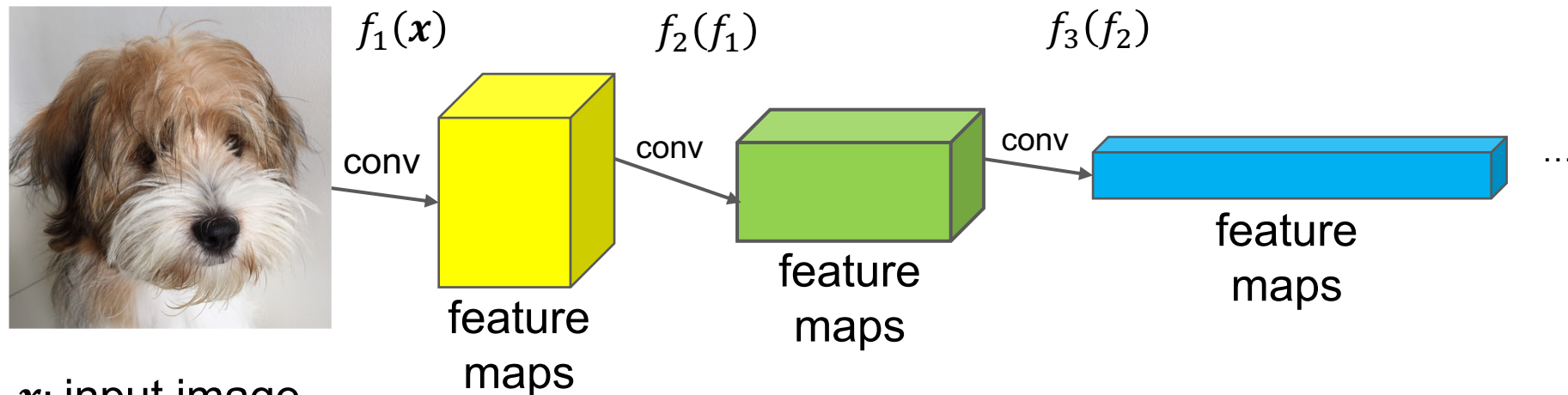
$\boldsymbol{x}$: input image or input feature

$$\boldsymbol{y} = f(\boldsymbol{x}) \approx f_n \circ f_{n-1} \circ f_{n-2} \circ \cdots \circ f_1(\boldsymbol{x})$$

CNN models can also be used as function approximators

# Reasons why CNN models are effective and efficient

Sparse Interaction - kernel as a feature detector is small compared to the input image thus requires few interaction only

Parameter Sharing - same set of parameters used for more than 1 function in the model

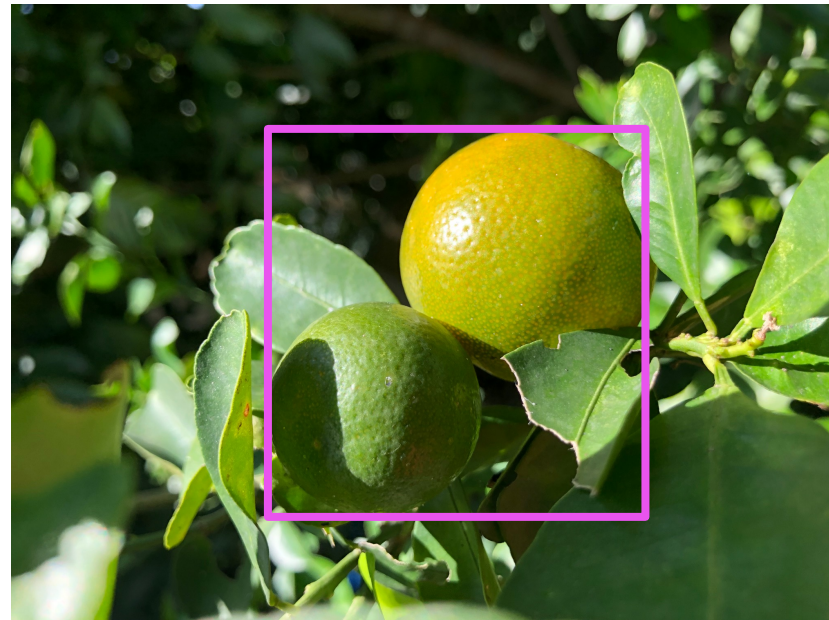**Receptive Field** - the input units that affect the output unit

The deeper the network, the larger is the receptive field

Receptive Field

Output Unit

Increasing depth

Receptive Field
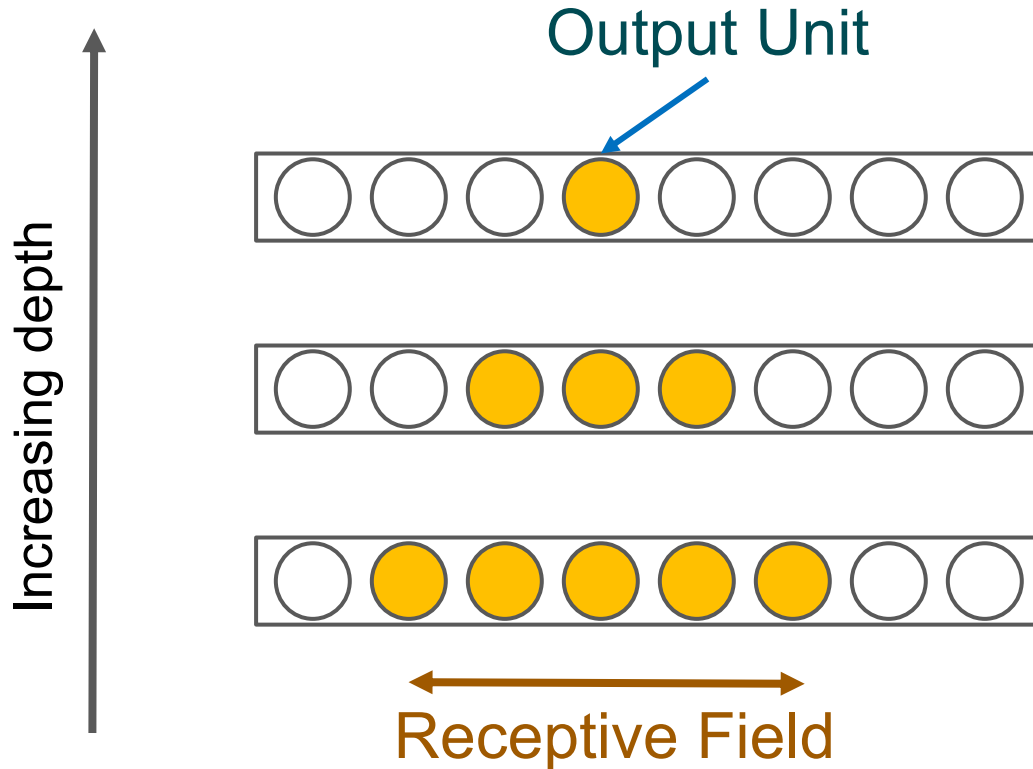
Assume:
Side view
3×3 kernel

CNN models are scale invariant

Due to max pooling, CNN models detect the same object regardless of scale

CNN models are translation equivariant

Due to kernel convolution, CNN models detect the same object even if it moves in the image

CNN models are not rotation invariant

No object detected

No built-in operators to make CNN models rotation invariant.
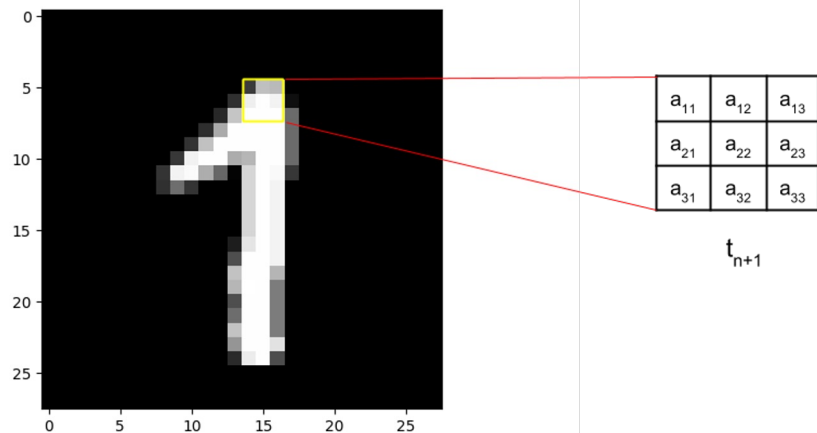*Solution*: Train the model with rotated objects.

CNN Ops

Convolution

Activation

Padding

Pooling

Strides

Convolution

t_n

t_{n+1}
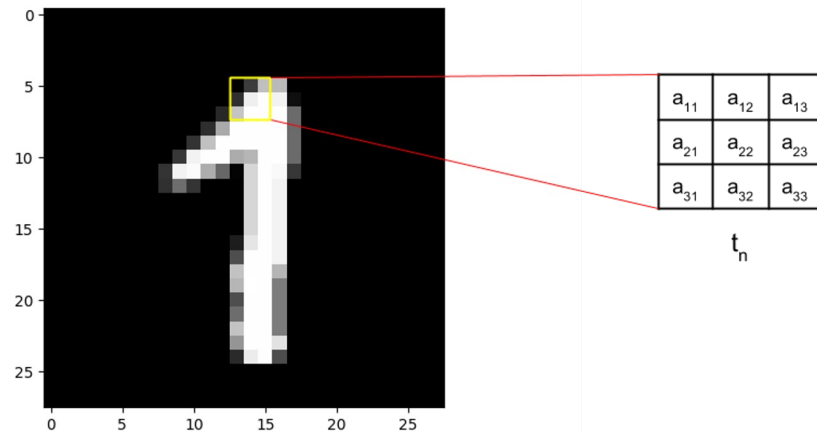
CNN: Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{11} = aw + bx + ey + fz$

Convolution No Padding (Valid)

X

a	**b**	**c**	d
e	**f**	**g**	h
i	j	k	l
m	n	o	p

*

K

w	**x**
y	**z**

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{12} = bw + cx + fy + gz$

Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{13} = cw + dx + gy + hz$

Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{21} = ew + fx + iy + jz$

Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{22} = fw + gx + jy + kz$

Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$$y_{23} = gw + hx + ky + lz$$

Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{31} = iw + jx + my + nz$

Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{32} = jw + kx + ny + oz$

Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{33} = kw + lx + oy + pz$

Activation Function - ReLU

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$f(\cdot)$ →

R

r_{11}	r_{12}	r_{13}
r_{21}	r_{22}	r_{23}
r_{31}	r_{32}	r_{33}

$R = \text{ReLU}(y)$
$r_{ij} = \text{ReLU}(y_{ij})$
Element-wise ReLU
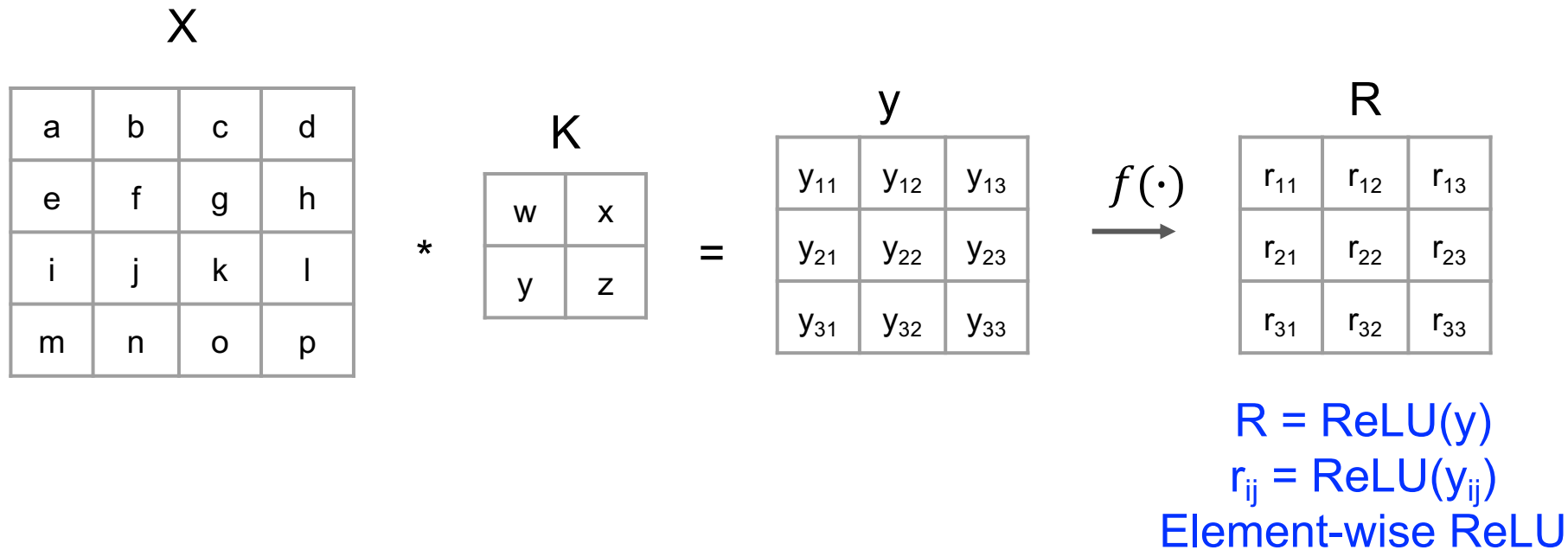
Downsampling - Pooling (eg MaxPooling)

R

r_{11}	r_{12}	r_{13}
r_{21}	r_{22}	r_{23}
r_{31}	r_{32}	r_{33}

P

p_{11}

=

$p_{11} = \max(r_{11}, r_{12}, r_{21}, r_{22})$

Why MaxPool? To Increase the Receptive Field

Before MaxPool
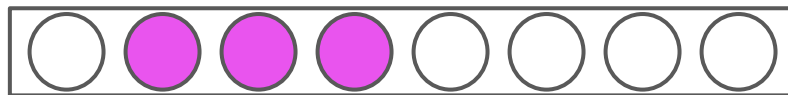
Assume:
Side view
3×3 kernel

Receptive Field is 3

Why MaxPool? To Increase the Receptive Field

After 2x2 MaxPool
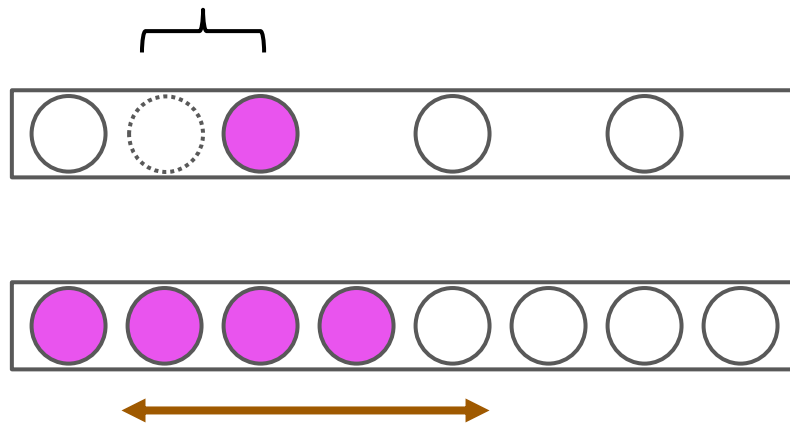
Assume:
Side view
3×3 kernel

Receptive Field is now 4

Downsampling using MaxPooling (MP)

X

MP(
a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p
) =

y

y_{11}	y_{12}
y_{21}	y_{22}

$y_{11} = \max(a, b, e, f)$

X

MP(
a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p
) =

y

y_{11}	y_{12}
y_{21}	y_{22}

$y_{12} = \max(c, d, g, h)$

X

MP(
a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p
) =

y

y_{11}	y_{12}
y_{21}	y_{22}

$y_{21} = \max(i, j, m, n)$

X

MP(
a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p
) =

y

y_{11}	y_{12}
y_{21}	y_{22}

$y_{22} = \max(k, l, o, p)$

27
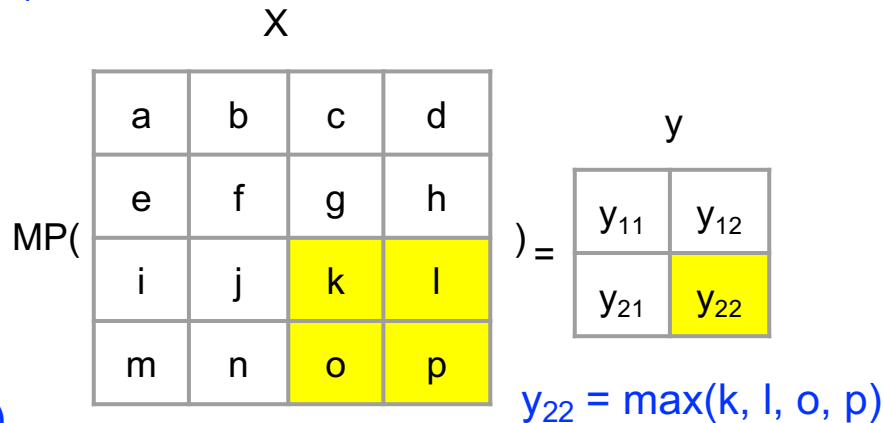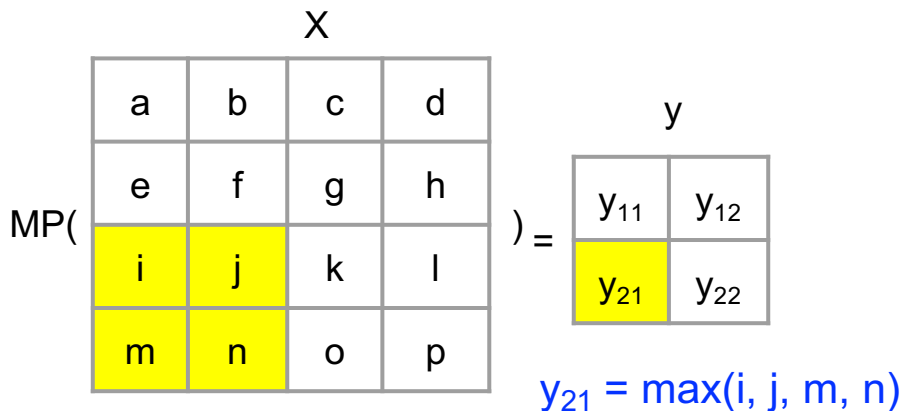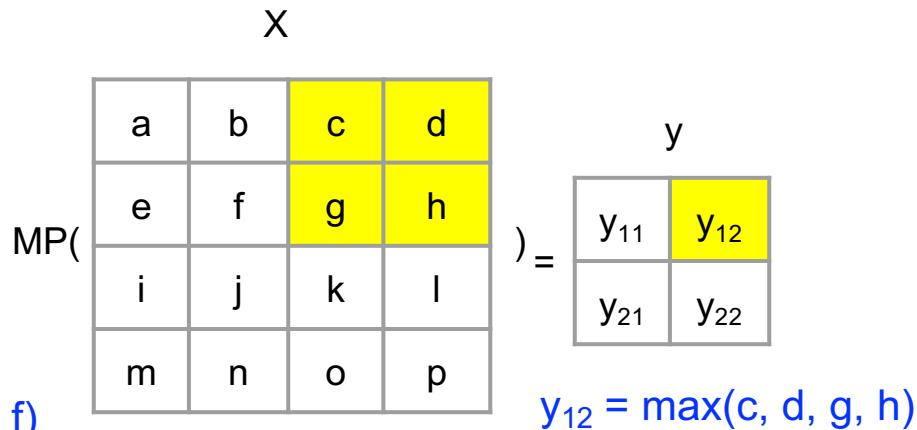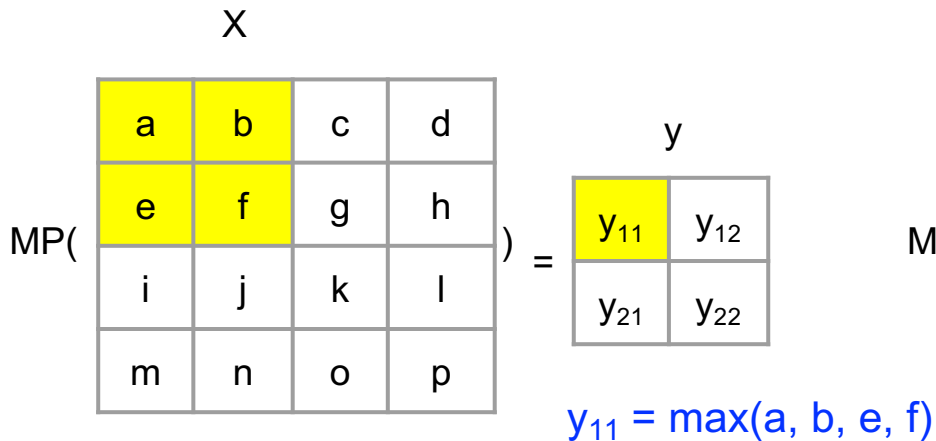
Downsampling using Stride > 1, (e.g. 2)

X
K
y

$y_{11} = aw + bx + ey + fz$

$y_{12} = cw + dx + gy + hz$

$y_{21} = iw + jx + my + nz$

$y_{22} = kw + lx + oy + pz$

Zero Padding

X

0	0	0	0	0	0
0	a	b	c	d	0
0	e	f	g	h	0
0	i	j	k	l	0
0	m	n	o	p	0
0	0	0	0	0	0

*

K

r	s	t
u	v	w
x	y	z

=

y

y_{11}	y_{12}	y_{13}	y_{14}
y_{21}	y_{22}	y_{23}	y_{24}
y_{31}	y_{32}	y_{33}	y_{34}
y_{41}	y_{42}	y_{43}	y_{44}

$y_{11} = av + bw + ey + fz$

$y_{12} = au + bv + cw + ex + fy + gz$

$y_{13} = bu + cv + dw + fx + gy + hz$

$y_{14} = cu + dv + gx + hy$

etc

29

K kernels/filters

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

s	t
u	v

w	x
y	z

K = 2

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

$y_{11} = as + bt + eu + fv$
etc.

t_{11}	t_{12}	t_{13}
t_{21}	t_{22}	t_{23}
t_{31}	t_{32}	t_{33}

$t_{11} = aw + bx + ey + fz$
etc.

Dilated Convolution

Dilation rate > 1 increases kernel coverage w/o increasing computation time

dilation_rate=1

dilation_rate=2

Dilated Convolution No Padding (Valid)

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}	y_{13}
y_{21}	y_{22}	y_{23}
y_{31}	y_{32}	y_{33}

dilation_rate=1

$y_{11} = aw + bx + ey + fz$

X

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

*

K

w	x
y	z

=

y

y_{11}	y_{12}
y_{21}	y_{22}

dilation_rate=2

$y_{11} = aw + cx + iy + kz$

UpSampling

UP(
a	b
c	d
) =

a	a	b	b
a	a	b	b
c	c	d	d
c	c	d	d

Interpolation: same data repeated n times
Other interpolation algorithms: Bilinear

Transposed Convolution for Dense Prediction

Convolution + UpSampling (if strides>2)
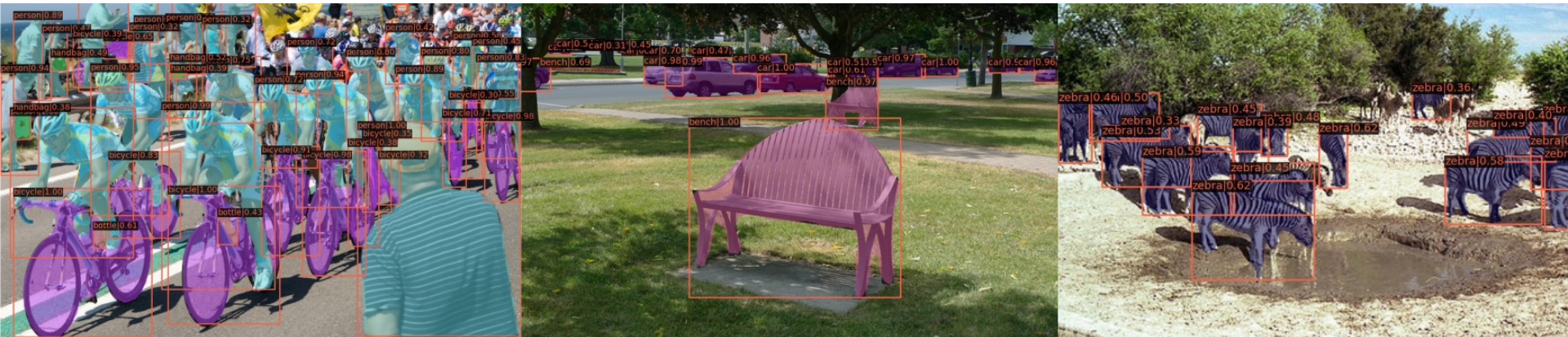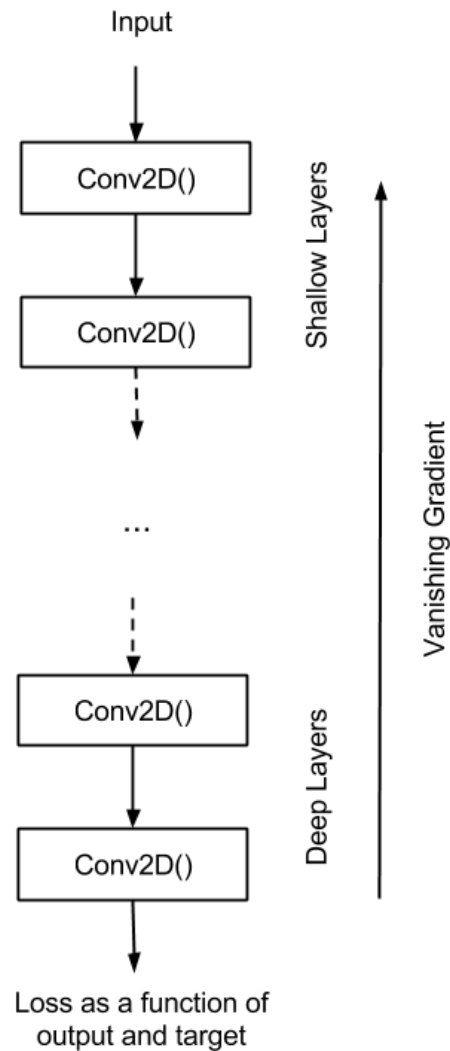

Segmentation Masks as Dense Prediction

https://github.com/open-mmlab/mmdetection

Issues with Deep Neural Networks

Vanishing Gradients
Exploding Gradients
Unstable Training

Vanishing Gradients

ResNet

$$x_{l-1} = H(x_{l-2})$$

By introducing a skip connection, ResNet avoids the problem of vanishing gradients

$$x_l = H(x_{l-1})$$

CNN Block

Residual Block
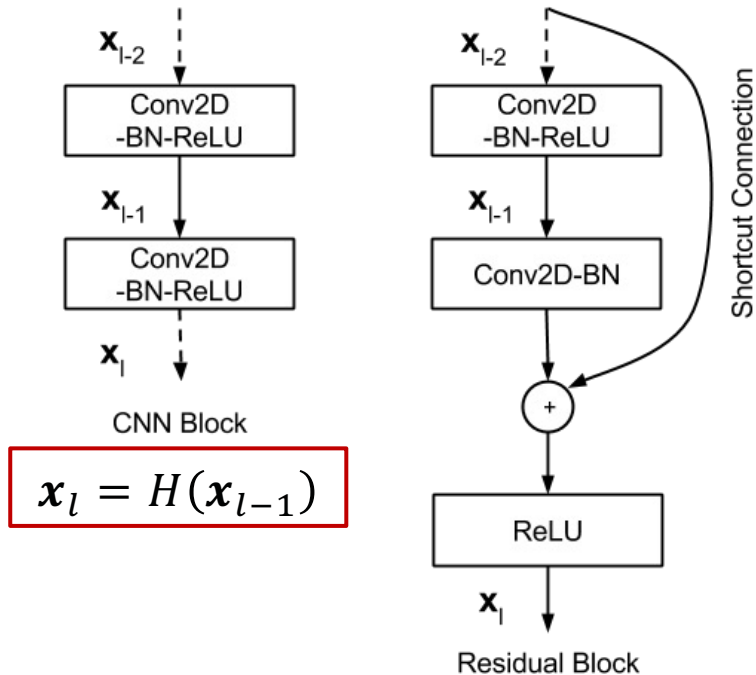
$$x_l = ReLU(F(x_{l-1}) + x_{l-2})$$

Improved ResNet

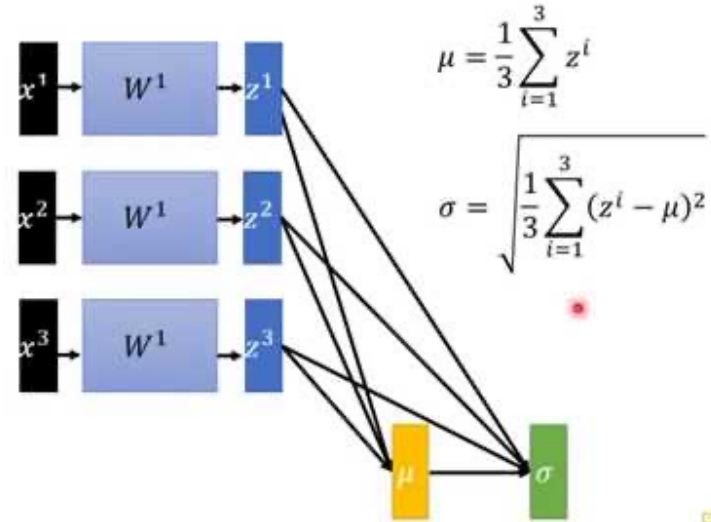Residual Block v1

Residual Block v2
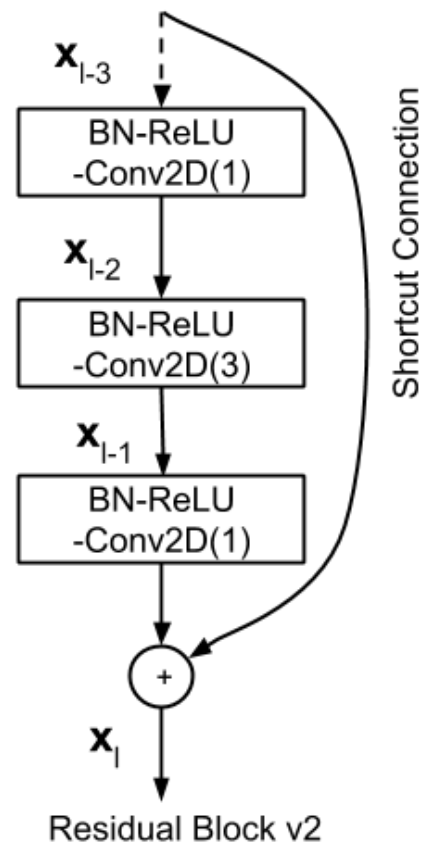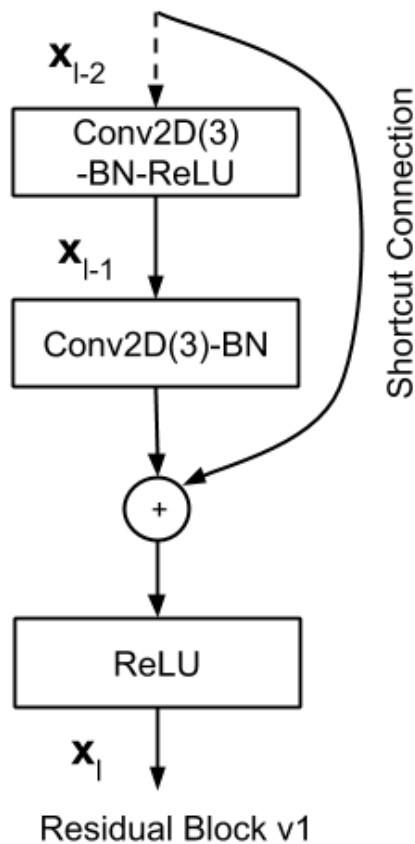
Batch Normalization

Applied layer-wise to maintain zero mean and variance of 1 for activation outputs

Allows use of larger learning rate in deep models w/o causing instability
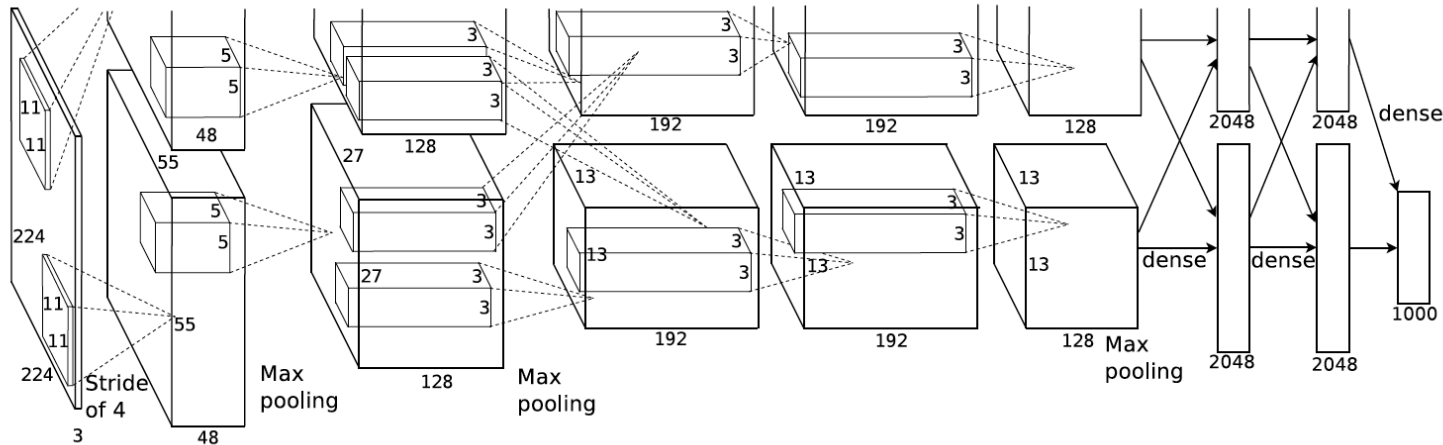
Downside: computed over a batch which is slow

Batch normalization

$$\mu = \frac{1}{3} \sum_{i=1}^{3} z^i$$

$$\sigma = \sqrt{\frac{1}{3} \sum_{i=1}^{3} (z^i - \mu)^2}$$

In Summary

CNN is parameter efficient, parallelizable, translation equivariant, and scale invariant model

Deep CNN exhibits state-of-the-art (SOTA) performances not only in vision tasks

AlexNet [Krizhevsky et al (2012)]

Code demo is next