



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

$$\mathbf{y} = \mathbf{x} * \mathbf{k}$$

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

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

$\mathbf{y} \in \mathbb{R}^{w \times h \times f}$ is output: f feature maps with dimensions $w \times h$ assuming sufficient padding is applied

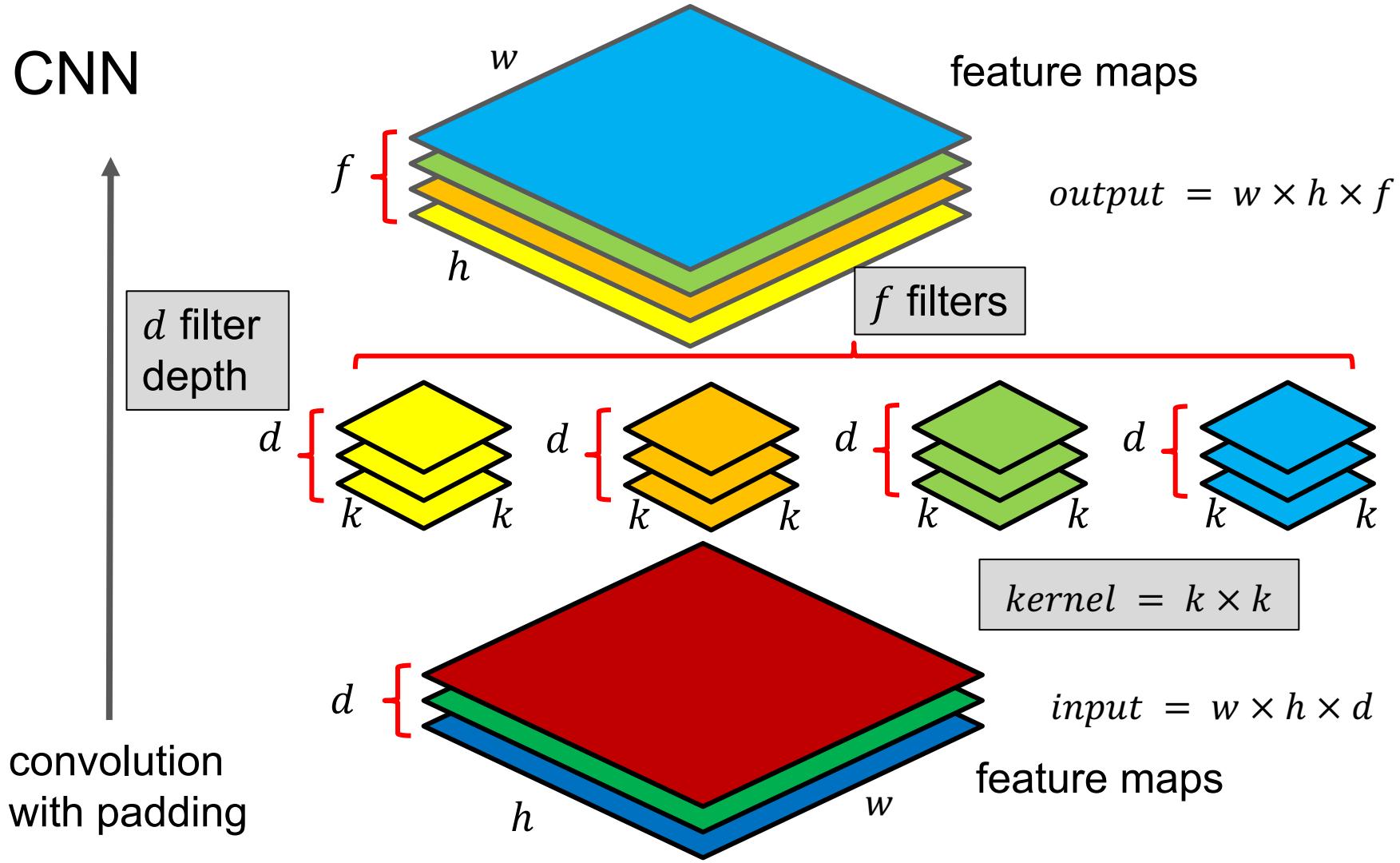
Dimensions are based on 2D inputs and square kernels

CNN Parameters: Kernel or Filter + Bias

In MLP, we learn weights and biases

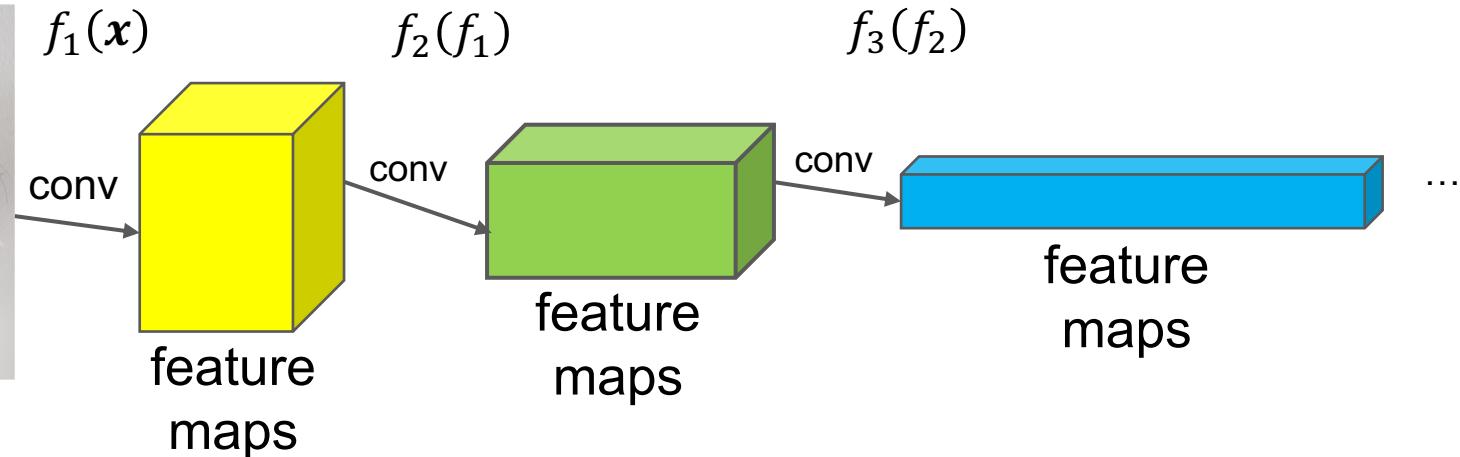
In CNN, we learn the weights and bias of a kernel

CNN



A CNN is made of multiple convolutional layers

The deeper the network, the more representations the network learns



x : input image
or input
feature

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

CNN models can also be used as function approximators

Reasons why CNN models are effective and efficient

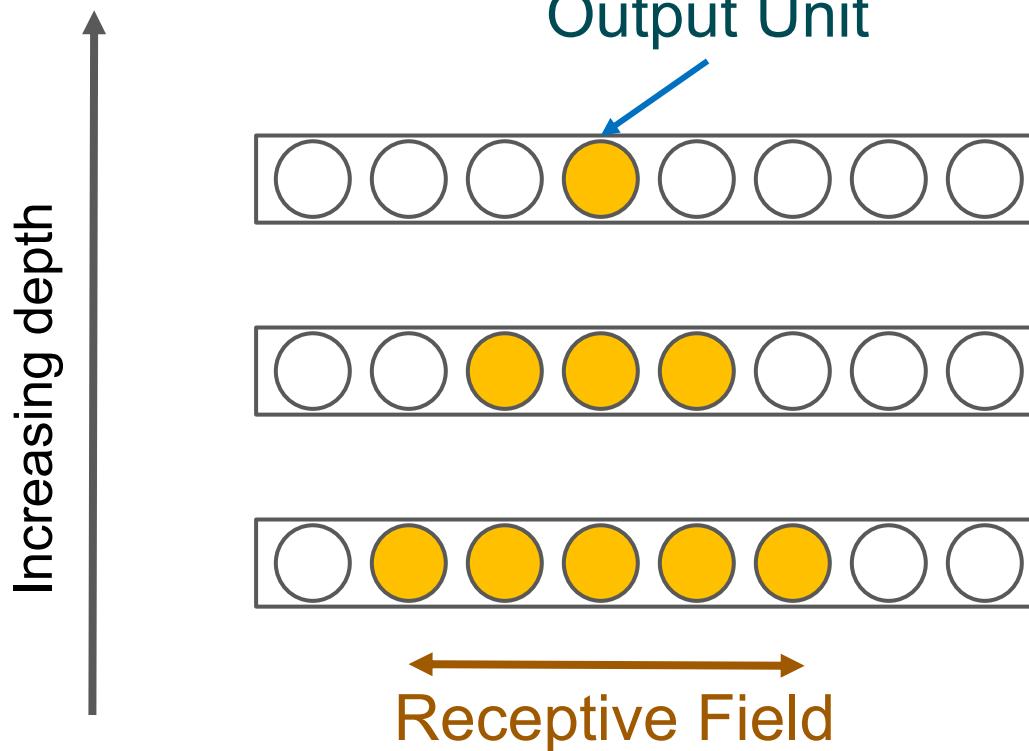
Sparse Interaction - kernel as a feature detector is small compared to the input image thus requires few interaction only

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

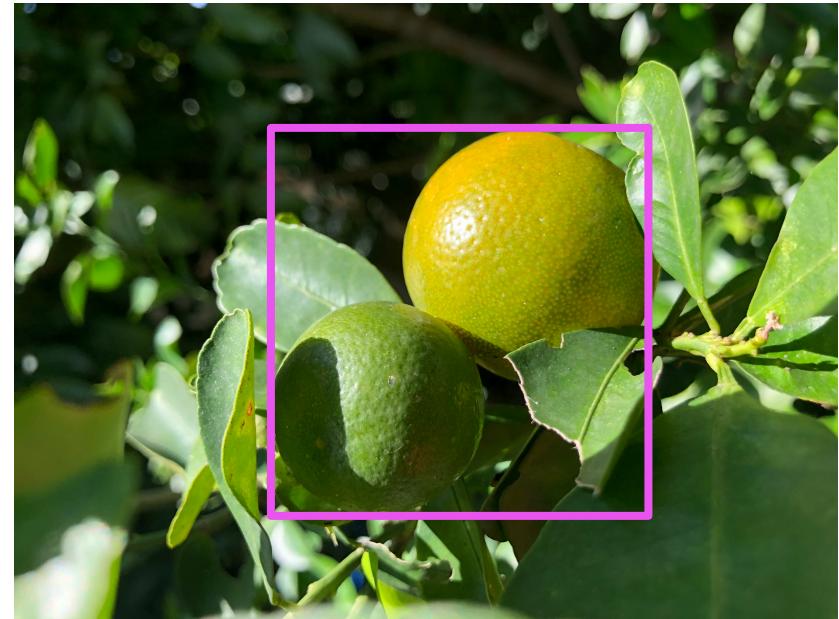
The deeper the network, the larger is the receptive field

Receptive Field



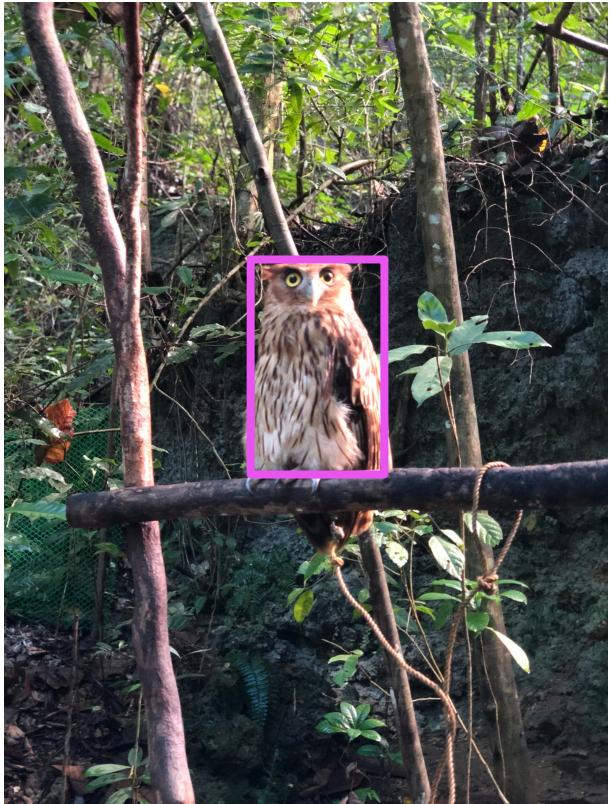
Assume:
Side view
 3×3 kernel

CNN models are scale invariant



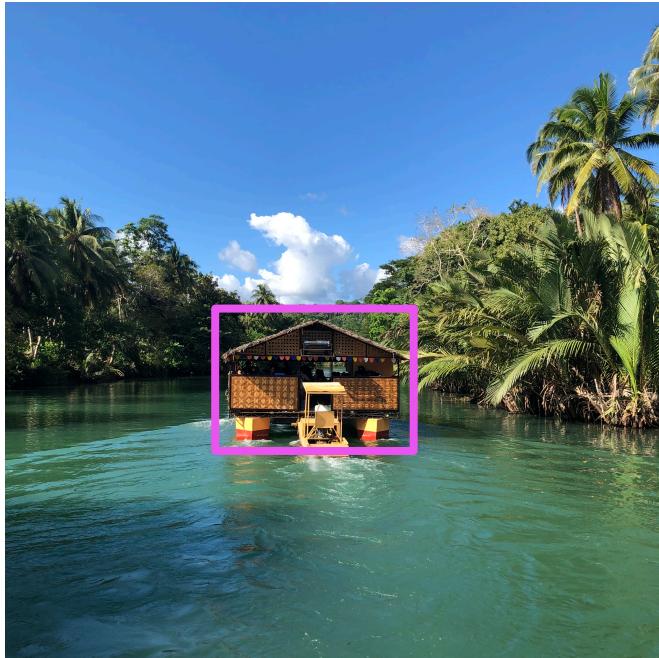
Due to max pooling, CNN models detect the same object regardless of scale

CNN models are translation equivariant

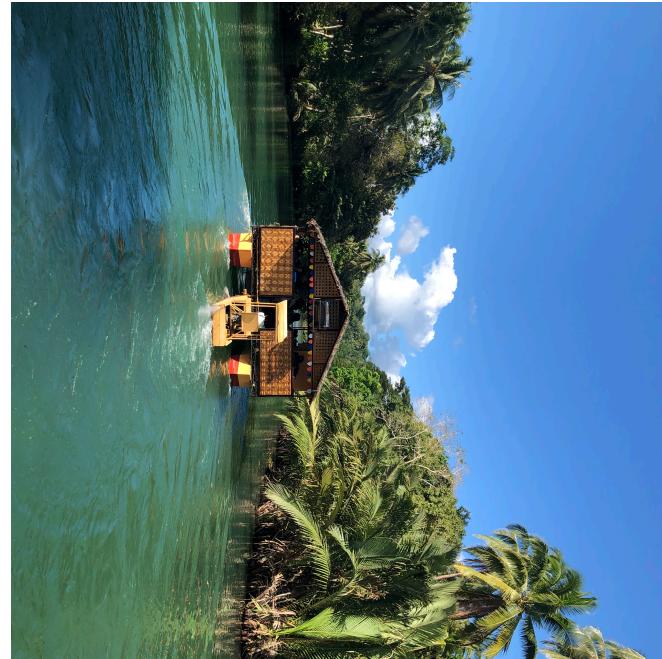


Due to kernel convolution, CNN models detect the same object even if it moves in the image ¹⁰

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

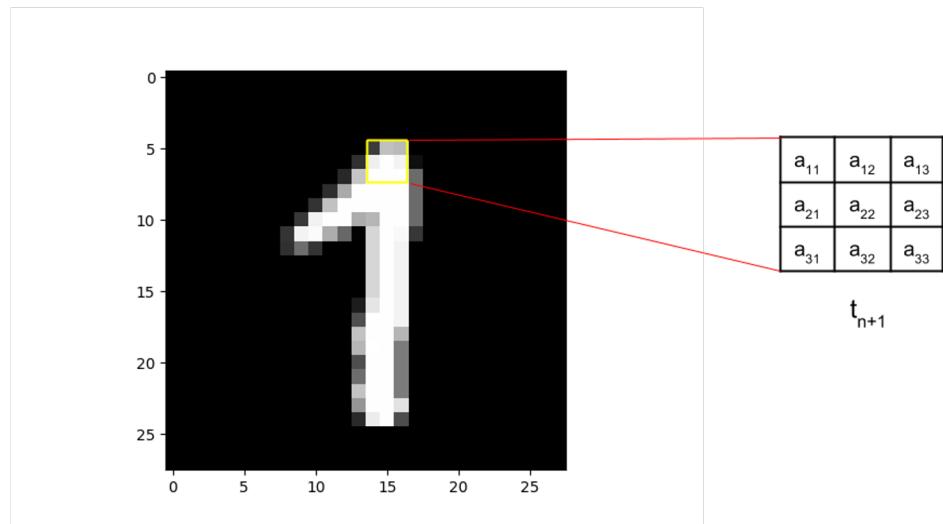
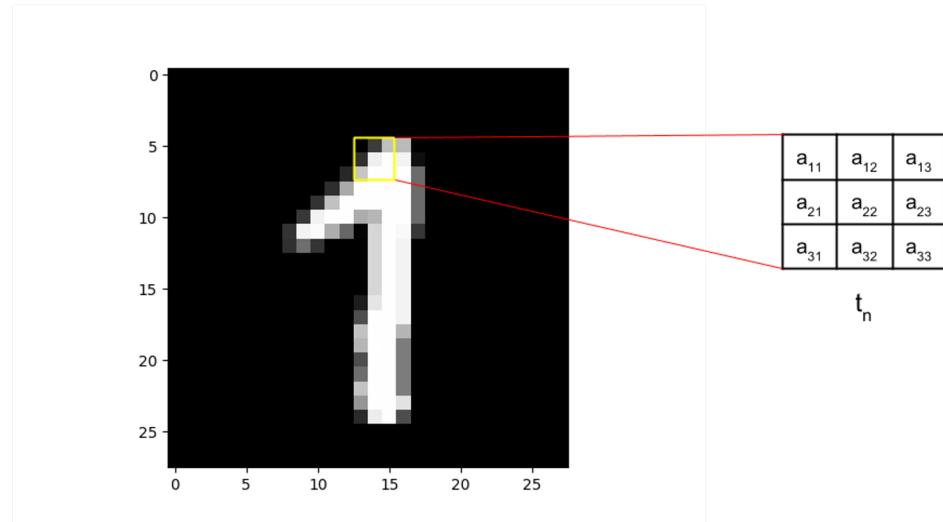
Activation

Padding

Pooling

Strides

Convolution



CNN: Convolution No Padding (Valid)

$$\begin{matrix} & X \\ \begin{matrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{matrix} & * & \begin{matrix} K \\ \begin{matrix} w & x \\ y & z \end{matrix} \end{matrix} & = & \begin{matrix} & y \\ \begin{matrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{matrix} \end{matrix} \end{matrix}$$

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

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)

$$\begin{array}{c} X \\ \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline e & f & g & h \\ \hline i & j & k & l \\ \hline m & n & o & p \\ \hline \end{array} \end{array} * \begin{array}{c} K \\ \begin{array}{|c|c|} \hline w & x \\ \hline y & z \\ \hline \end{array} \end{array} = \begin{array}{c} Y \\ \begin{array}{|c|c|c|} \hline y_{11} & y_{12} & y_{13} \\ \hline y_{21} & y_{22} & y_{23} \\ \hline y_{31} & y_{32} & y_{33} \\ \hline \end{array} \end{array}$$
$$y_{22} = fw + gx + jy + kz$$

Convolution No Padding (Valid)

$$\begin{array}{c} X \\ \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline e & f & g & h \\ \hline i & j & k & l \\ \hline m & n & o & p \\ \hline \end{array} \end{array} * \begin{array}{c} K \\ \begin{array}{|c|c|} \hline w & x \\ \hline y & z \\ \hline \end{array} \end{array} = \begin{array}{c} y \\ \begin{array}{|c|c|c|} \hline y_{11} & y_{12} & y_{13} \\ \hline y_{21} & y_{22} & y_{23} \\ \hline y_{31} & y_{32} & y_{33} \\ \hline \end{array} \end{array}$$

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

Convolution No Padding (Valid)

$$\begin{matrix} & \mathbf{X} \\ \begin{matrix} \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ \mathbf{e} & \mathbf{f} & \mathbf{g} & \mathbf{h} \\ \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ \mathbf{m} & \mathbf{n} & \mathbf{o} & \mathbf{p} \end{matrix} & \ast & \mathbf{K} & = & \mathbf{y} \\ & & \begin{matrix} \mathbf{w} & \mathbf{x} \\ \mathbf{y} & \mathbf{z} \end{matrix} & & \begin{matrix} \mathbf{y}_{11} & \mathbf{y}_{12} & \mathbf{y}_{13} \\ \mathbf{y}_{21} & \mathbf{y}_{22} & \mathbf{y}_{23} \\ \mathbf{y}_{31} & \mathbf{y}_{32} & \mathbf{y}_{33} \end{matrix} \end{matrix}$$

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

Convolution No Padding (Valid)

$$\begin{array}{c} X \\ \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline e & f & g & h \\ \hline i & j & k & l \\ \hline m & n & o & p \\ \hline \end{array} \end{array} * \begin{array}{c} K \\ \begin{array}{|c|c|} \hline w & x \\ \hline y & z \\ \hline \end{array} \end{array} = \begin{array}{c} Y \\ \begin{array}{|c|c|c|} \hline y_{11} & y_{12} & y_{13} \\ \hline y_{21} & y_{22} & y_{23} \\ \hline y_{31} & y_{32} & y_{33} \\ \hline \end{array} \end{array}$$

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

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

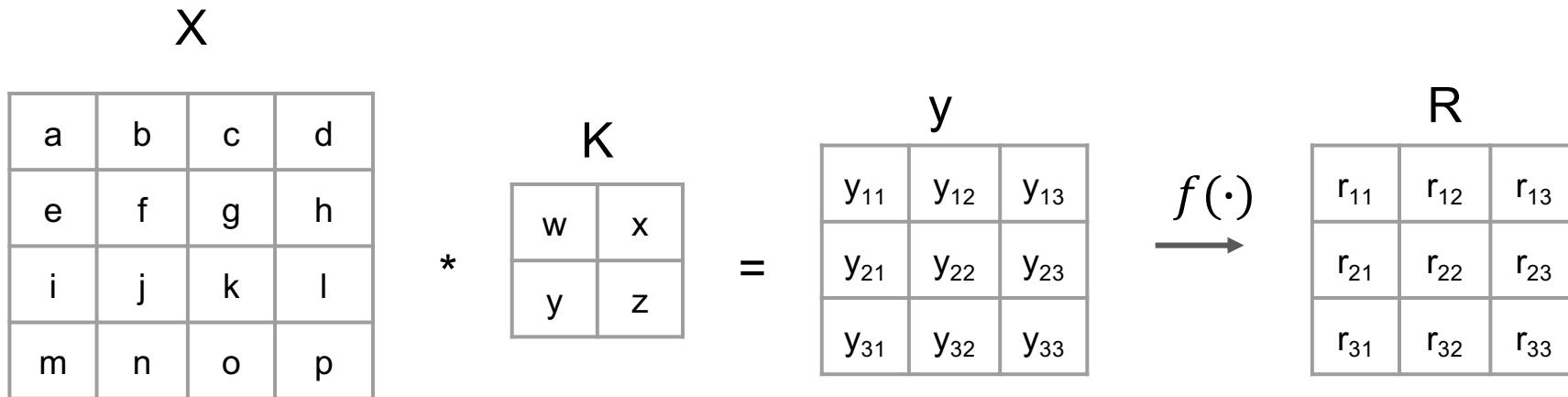
=

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



$$R = \text{ReLU}(y)$$

$$r_{ij} = \text{ReLU}(y_{ij})$$

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

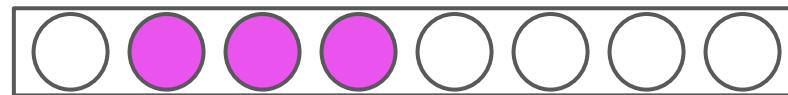
=

p_{11}

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

Why MaxPool? To Increase the Receptive Field

Before MaxPool



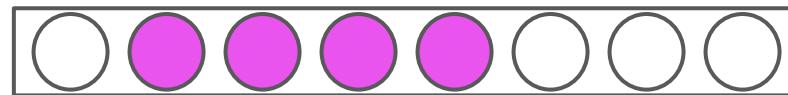
Assume:
Side view
 3×3 kernel



Receptive Field is 3

Why MaxPool? To Increase the Receptive Field

After 2x2 MaxPool



Receptive Field is now 4

Assume:
Side view
 3×3 kernel

Downsampling using MaxPooling (MP)

$$\text{MP}(X) = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$$

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

$$\text{MP}(X) = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$$

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

$$\text{MP}(X) = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$$

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

$$\text{MP}(X) = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$$

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

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

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

$$\begin{array}{ccc}
 X & K & y \\
 \begin{matrix} w & x \\ y & z \end{matrix} & = & \begin{matrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{matrix} \\
 * & & \\
 \end{array}$$

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

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

$$\begin{array}{ccc}
 X & K & y \\
 \begin{matrix} w & x \\ y & z \end{matrix} & = & \begin{matrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{matrix} \\
 * & & \\
 \end{array}$$

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

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

$$\begin{array}{ccc}
 X & K & y \\
 \begin{matrix} w & x \\ y & z \end{matrix} & = & \begin{matrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{matrix} \\
 * & & \\
 \end{array}$$

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

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

$$\begin{array}{ccc}
 X & K & y \\
 \begin{matrix} w & x \\ y & z \end{matrix} & = & \begin{matrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{matrix} \\
 * & & \\
 \end{array}$$

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

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

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

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

dilation_rate=1

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

dilation_rate=2

Dilated Convolution No Padding (Valid)

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

$X \quad * \quad K = y$

dilation_rate=1

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

The diagram illustrates a 4x4 input matrix X and a 2x2 kernel K . The result is a 3x3 output matrix y . The calculation for y_{11} is shown as a weighted sum of input elements a, b, e, f .

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

$X \quad * \quad K = y$

dilation_rate=2

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

The diagram illustrates a 4x4 input matrix X and a 2x2 kernel K . The result is a 2x2 output matrix y . The calculation for y_{11} is shown as a weighted sum of input elements a, c, i, k .

UpSampling

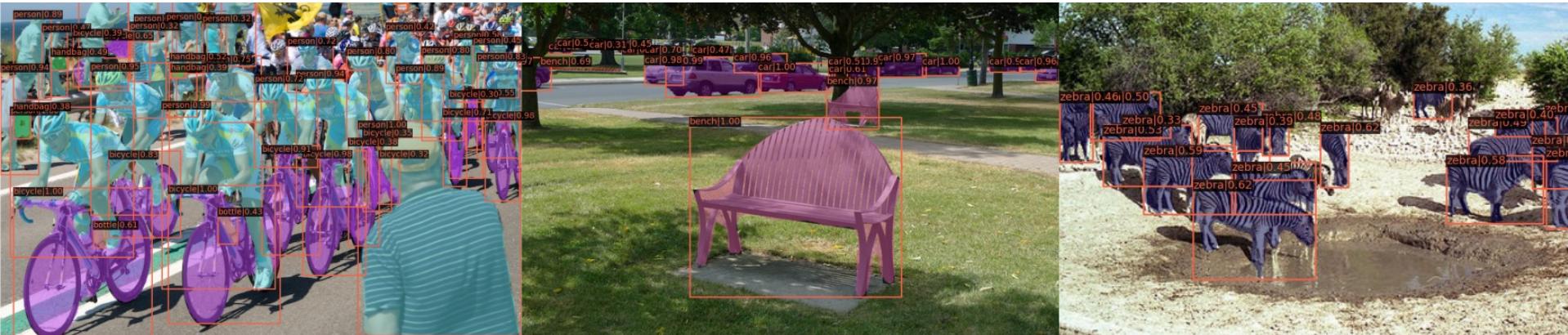
$$\text{UP}(\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline\end{array}) =$$

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>

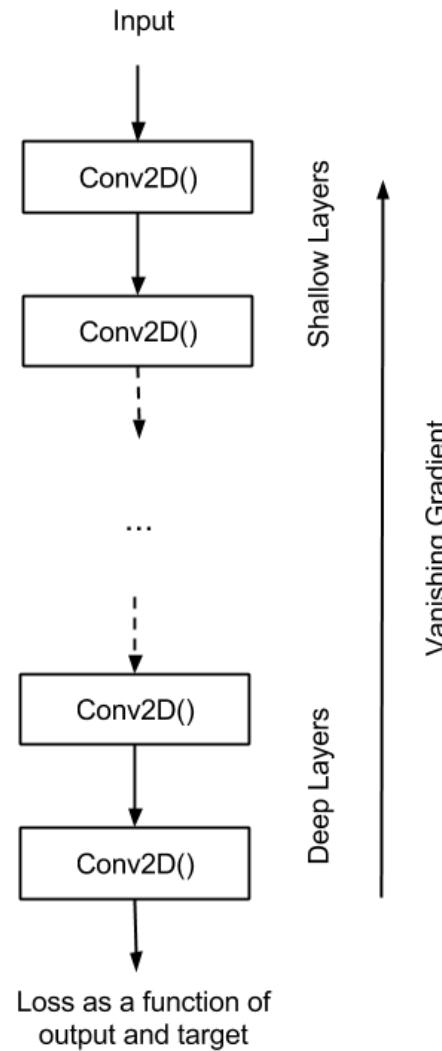
Issues with Deep Neural Networks

Vanishing Gradients

Exploding Gradients

Unstable Training

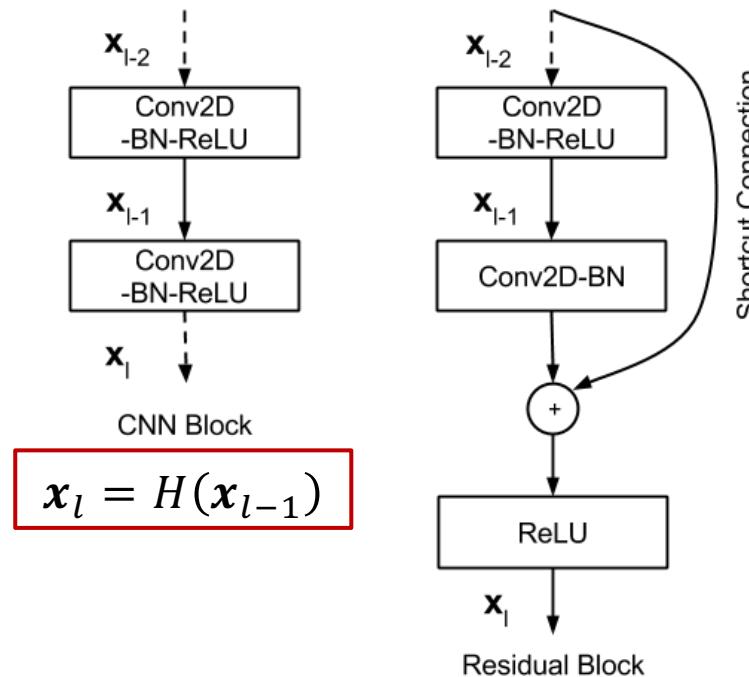
Vanishing Gradients



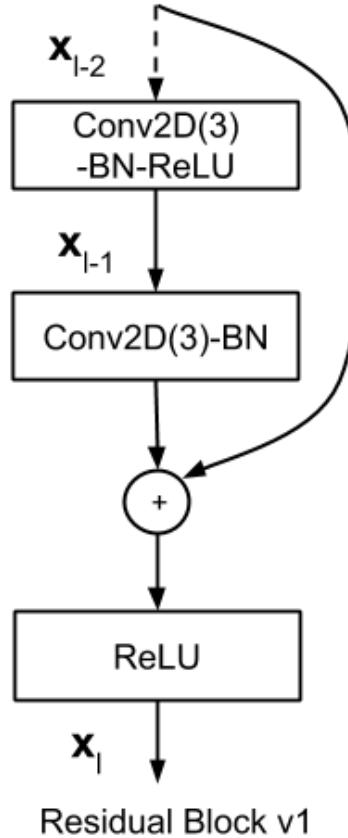
ResNet

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

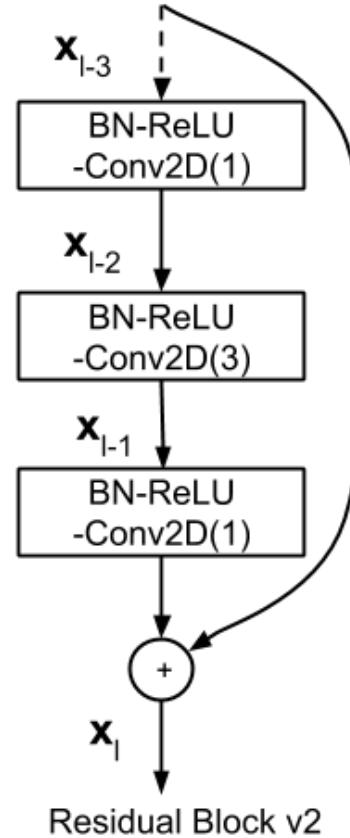
$$\mathbf{x}_{l-1} = H(\mathbf{x}_{l-2})$$



Improved ResNet



Shortcut Connection



Shortcut Connection

Batch Normalization

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

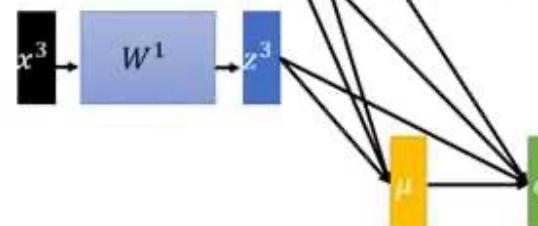
Allows use of larger learning rate in deep models w/o causing instability



Batch normalization

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

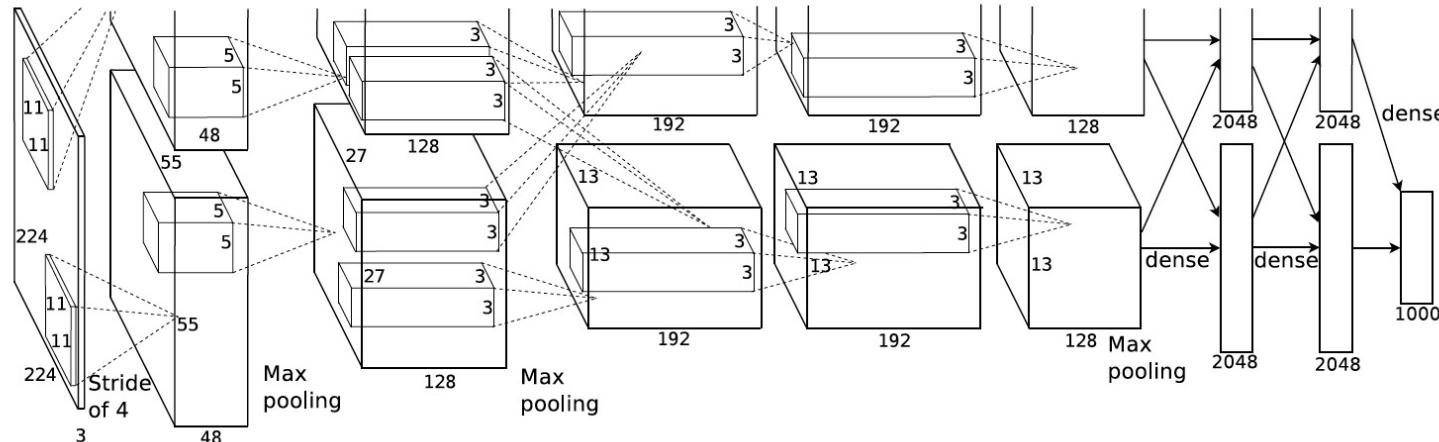
$$\sigma = \sqrt{\frac{1}{3} \sum_{i=1}^3 (z^i - \mu)^2}$$



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