Convolutional Neural Network (CNN)

Rowel Atienza
rowel@eee.upd.edu.ph
University of the Philippines
Updated: 26 Sept 2020

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*

CNN

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

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

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

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

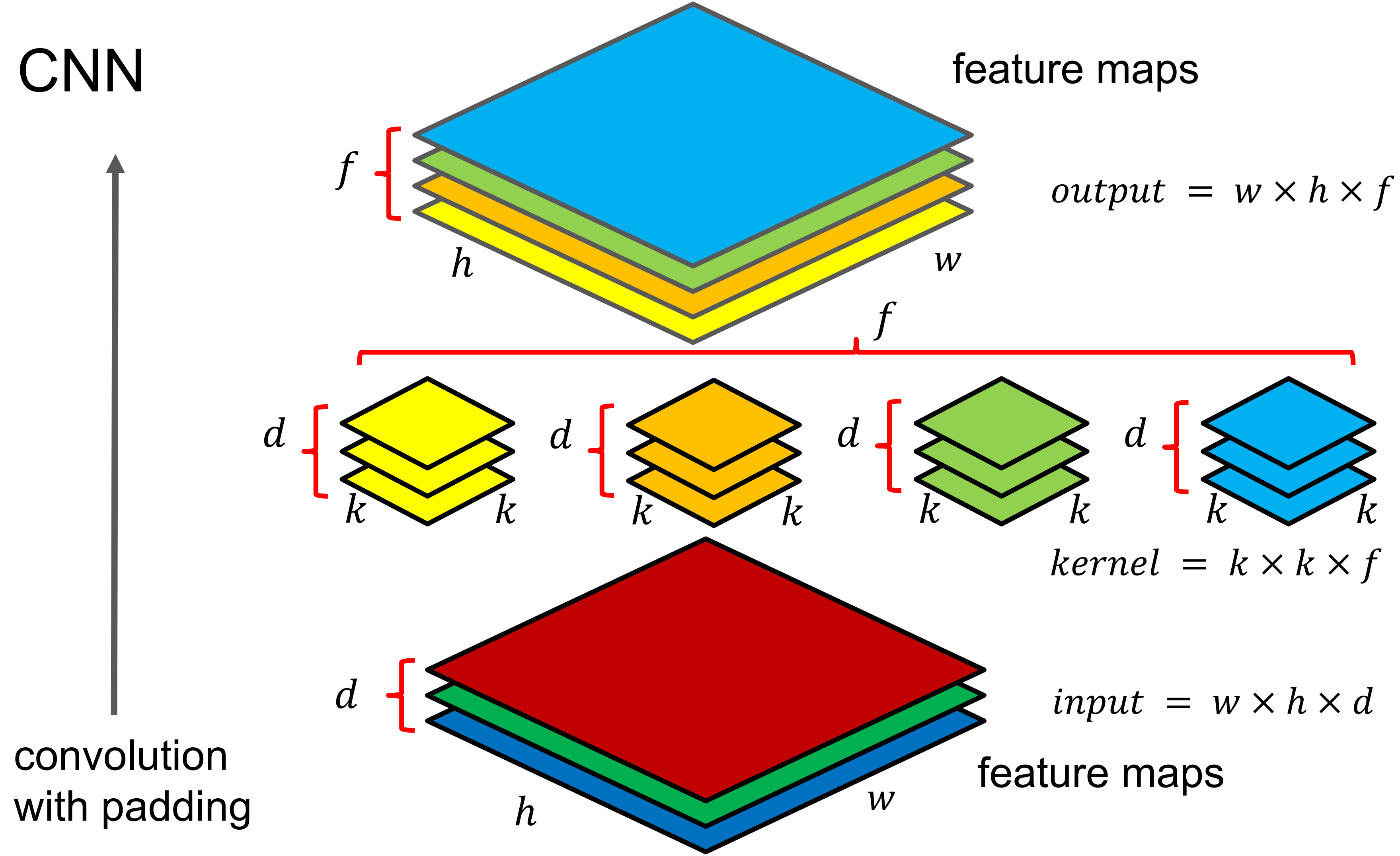
Dimensions are based on 2D inputs and square kernels

Kernel

In MLP, we learn weights and biases

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

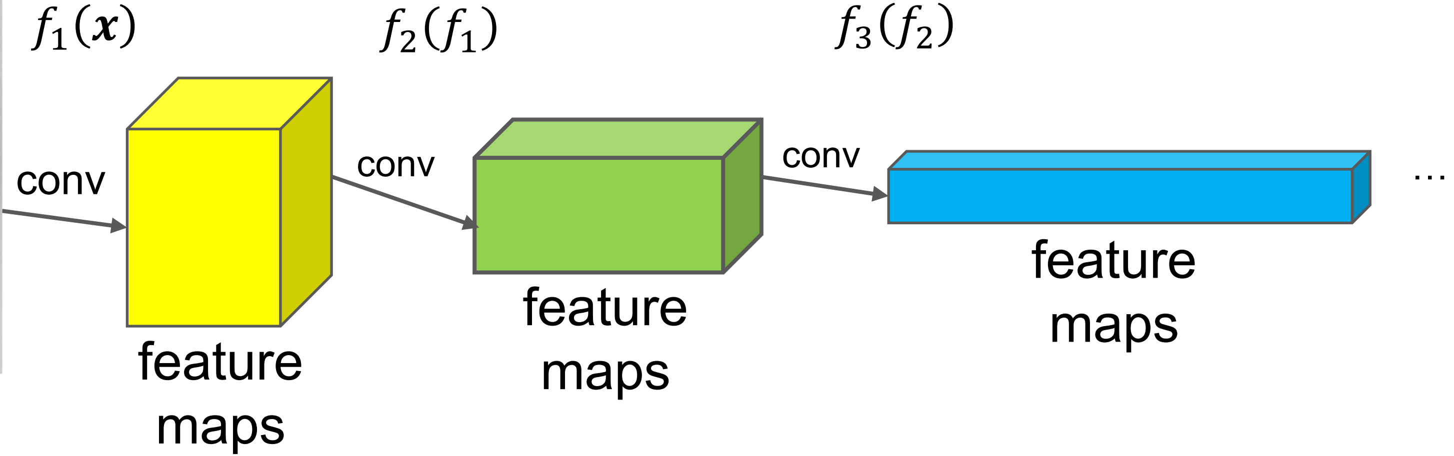
CNN


feature maps
f
$output = w \times h \times f$
h
w
f
d
k
k
$kernel = k \times k \times f$
d
$input = w \times h \times d$
feature maps
h
w
convolution
with padding

CNN

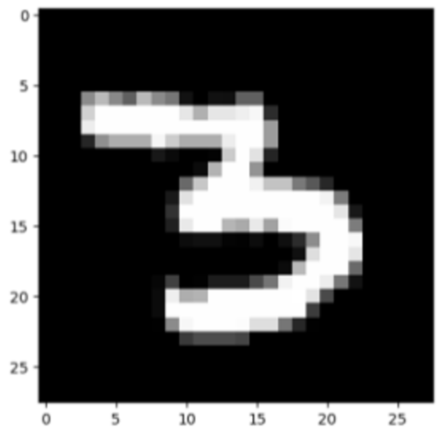
The deeper the network, the more representations the network learns

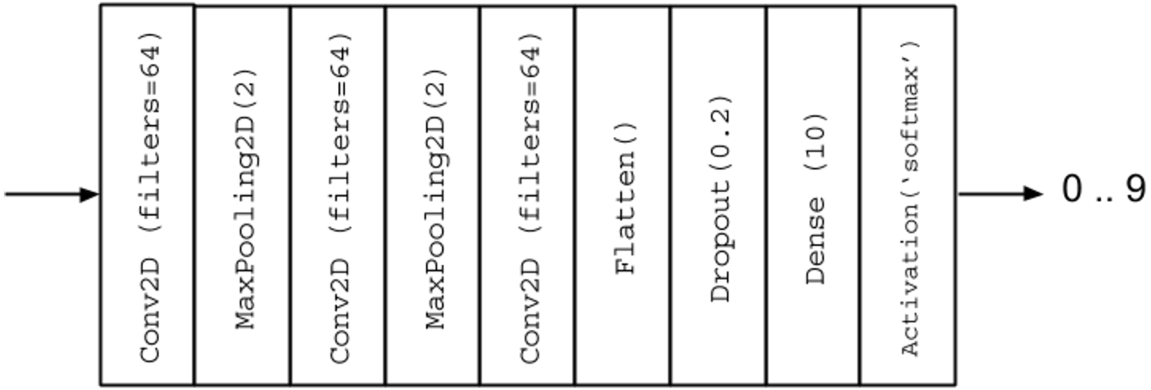
$\boldsymbol{x}$: input image



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

CNN Model on MNIST





Keras Conv2D has built-in ReLU activation

Properties

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

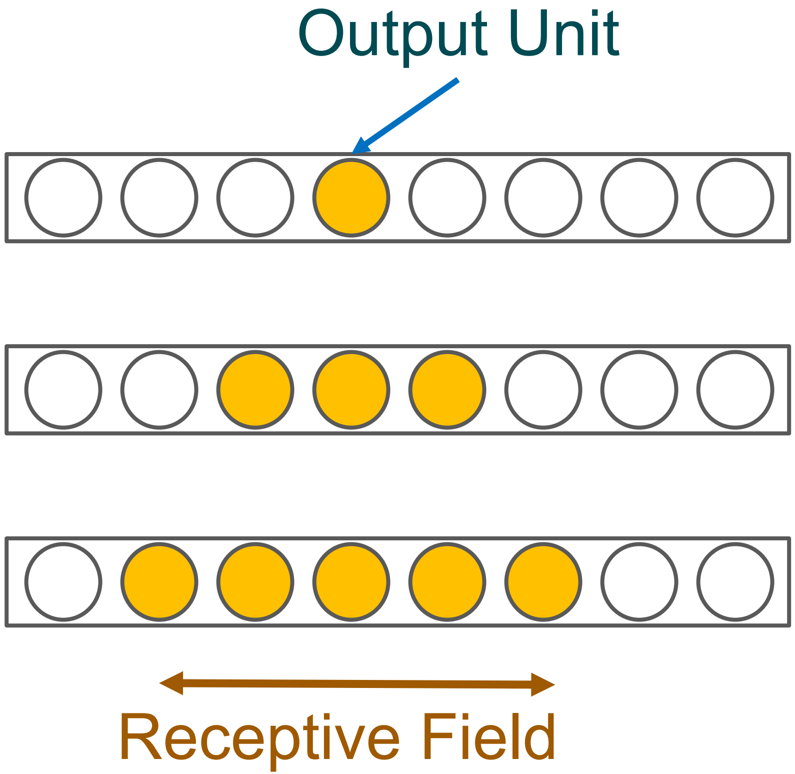
Parameter Sharing - same set of parameters used for more than 1 function in the model

Receptive Field - the input units that affect the output unit

The deeper the network, the larger is the receptive field

Equivariance to Translation (Scale? Rotation?) - if the input changes, the output changes in the same way

Receptive Field



Assume:
Side view
3×3 kernel

Equivariance to Translation

CNN Ops

Convolution

Activation

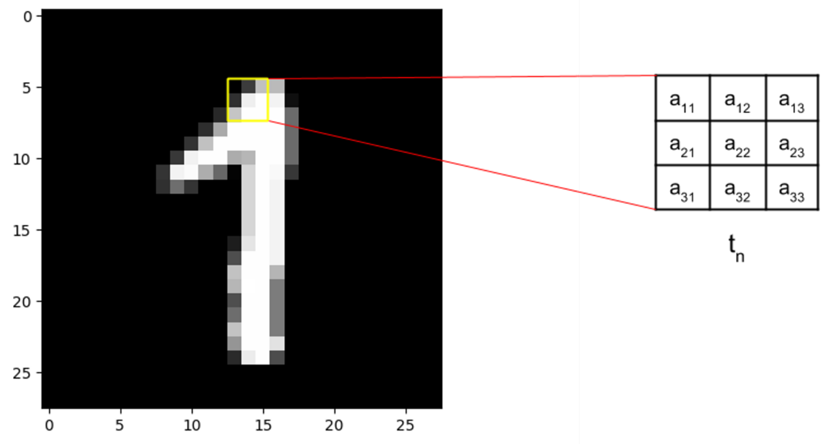
Padding

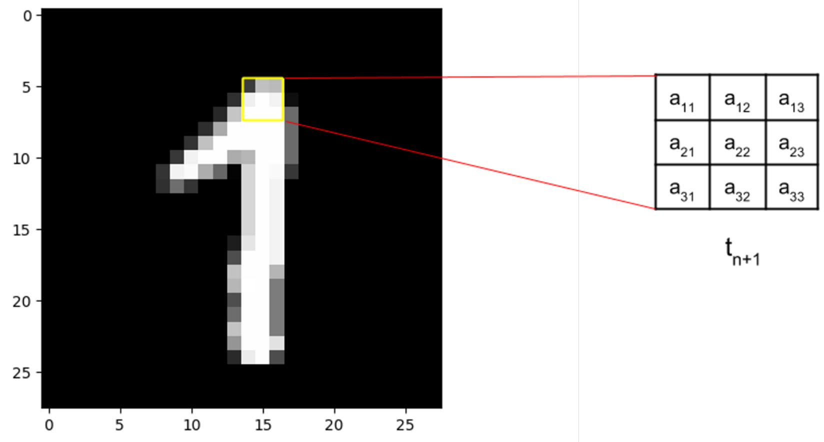
Pooling

Strides

Filters

Convolution





CNN: Convolution No Padding (Valid)

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

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

$i = 1,2,3$
$j = 1,2,3$
$m = 1,2,3$
$n = 1,2,3$

Convolution No Padding (Valid)

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

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

Convolution No Padding (Valid)

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

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

Convolution No Padding (Valid)

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

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

Convolution No Padding (Valid)

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

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

Convolution No Padding (Valid)

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

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

Convolution No Padding (Valid)

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

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

Convolution No Padding (Valid)

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

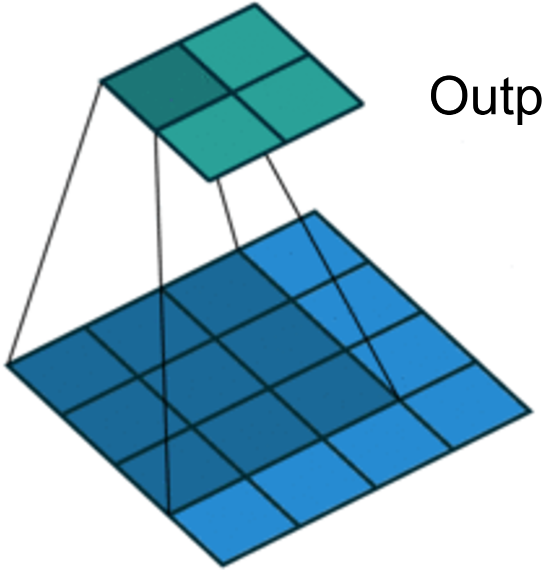
$y_{32} = jw + kx + ny + oz$

Convolution No Padding (Valid)

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

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

CNN in GIF (No padding, stride=1)



Output

3x3 kernel on 4x4 input

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Activation Function - ReLU

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

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

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

Element-wise ReLU

Downsampling - Pooling (eg MaxPooling)

$$R = \begin{bmatrix} \boxed{r_{11}} & \boxed{r_{12}} & r_{13} \\ \boxed{r_{21}} & \boxed{r_{22}} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad = \quad P = \begin{bmatrix} \boxed{p_{11}} \end{bmatrix}$$

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

Downsampling - Pooling

Other Pooling Functions

Average

Median

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

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}

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

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}

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

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}

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

X

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

* K

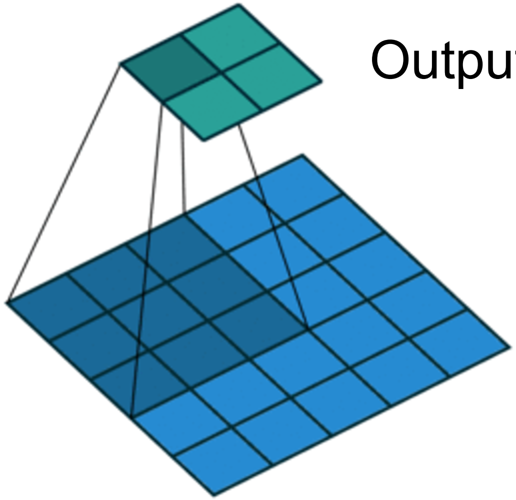
w	x
y	z

= y

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

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

Downsampling using Stride = 2



Output

3x3 kernel on 4x4 input

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Downsampling using MaxPooling (MP)

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

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

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

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

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

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

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

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

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}

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

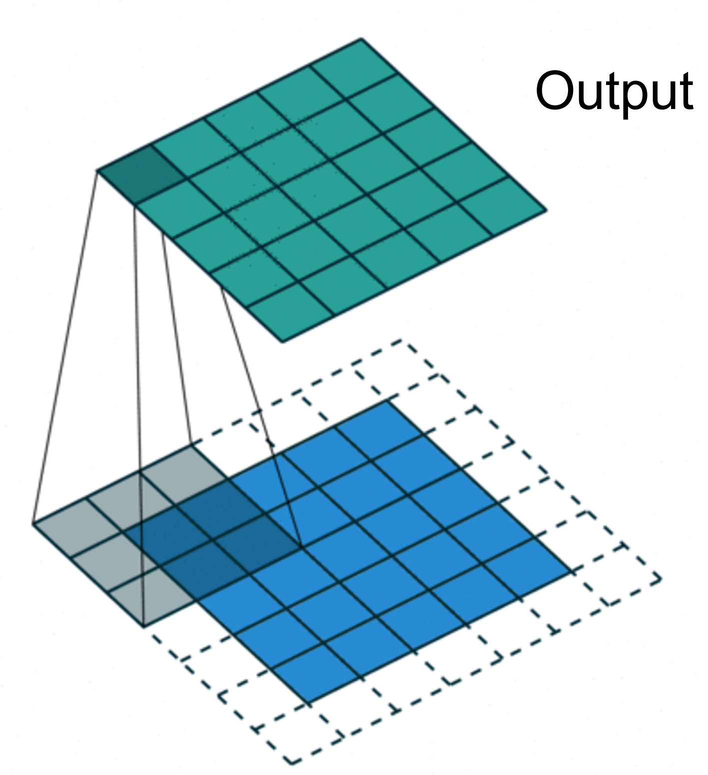
$y_{12} = au + bv + cw + ex + fy + gz$

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

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

etc

Zero Padding, strides=1



https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

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}

$y_{11} = ay + bt + eu + fv$

etc.

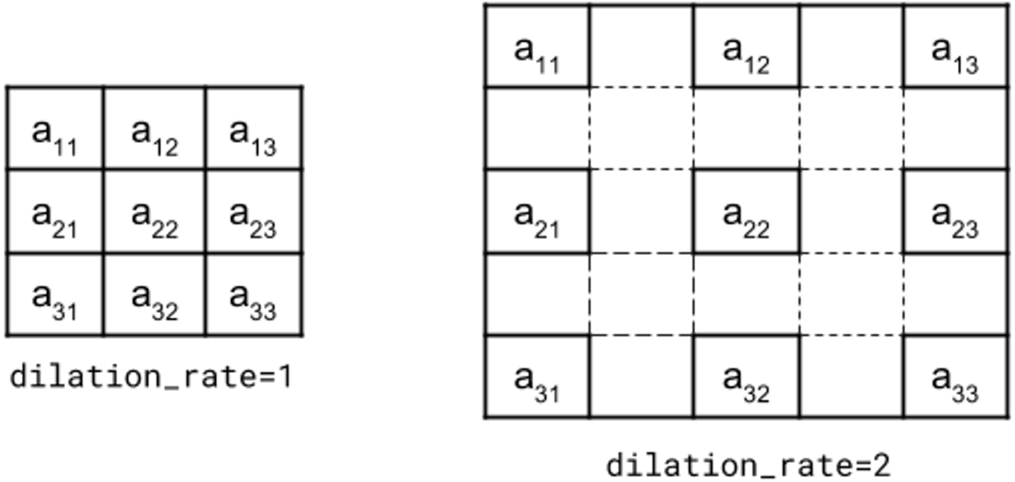
t_{11}	t_{12}	t_{13}
t_{21}	t_{22}	t_{23}
t_{31}	t_{32}	t_{33}

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

etc.

Dilated Convolution

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



Dilated Convolution No Padding (Valid)

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

dilation_rate=1

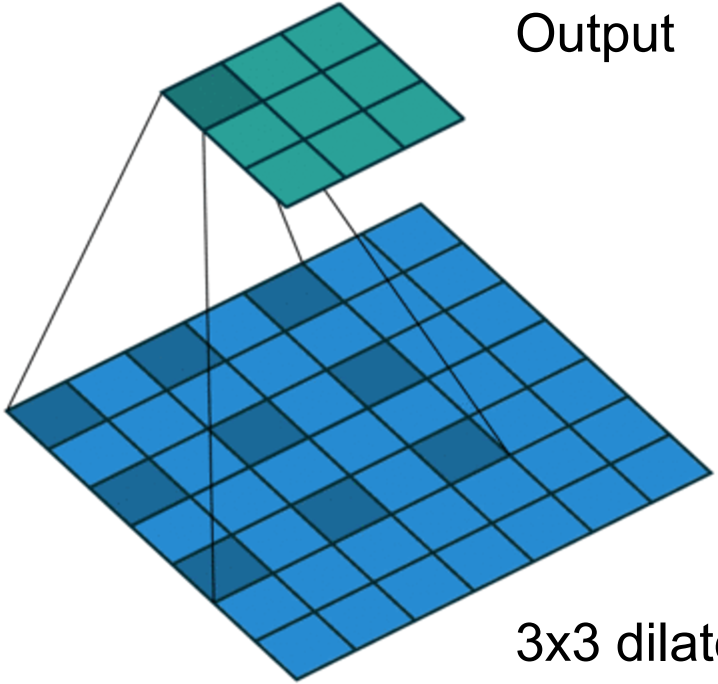
$$y_{11} = aw + bx + ey + fz$$

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

dilation_rate=2

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

Dilated convolution



https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

UpSampling

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

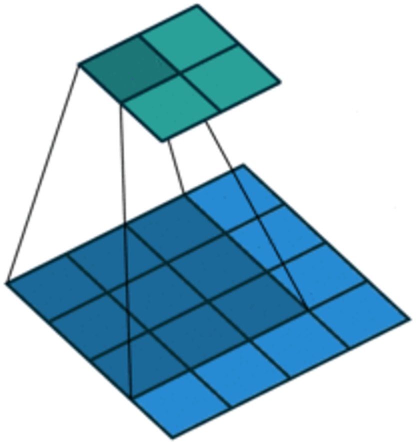
Interpolation: same data repeated n times
Other interpolation algorithms: Bilinear

Transposed Convolution

Convolution + Upsampling (if strides>2)

See more examples of CNN operations in

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md



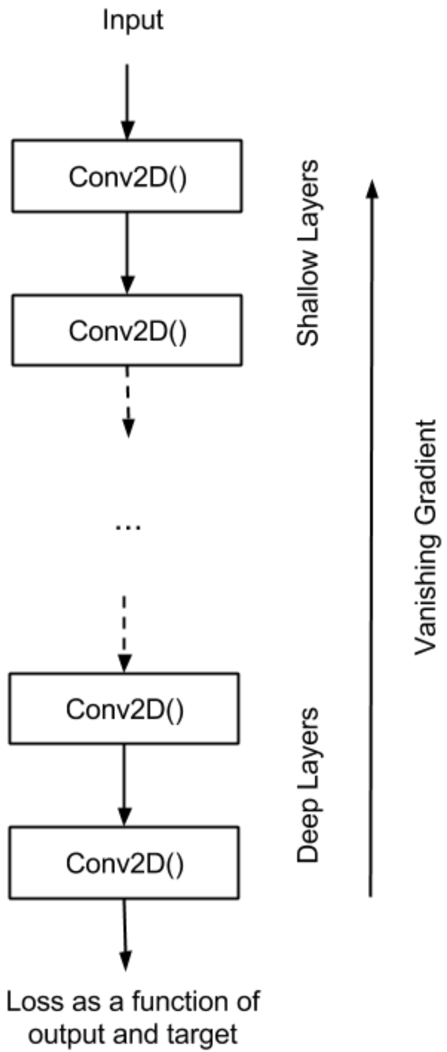
Issues with Deep Neural Networks

Vanishing Gradients

Exploding Gradients

Unstable Training

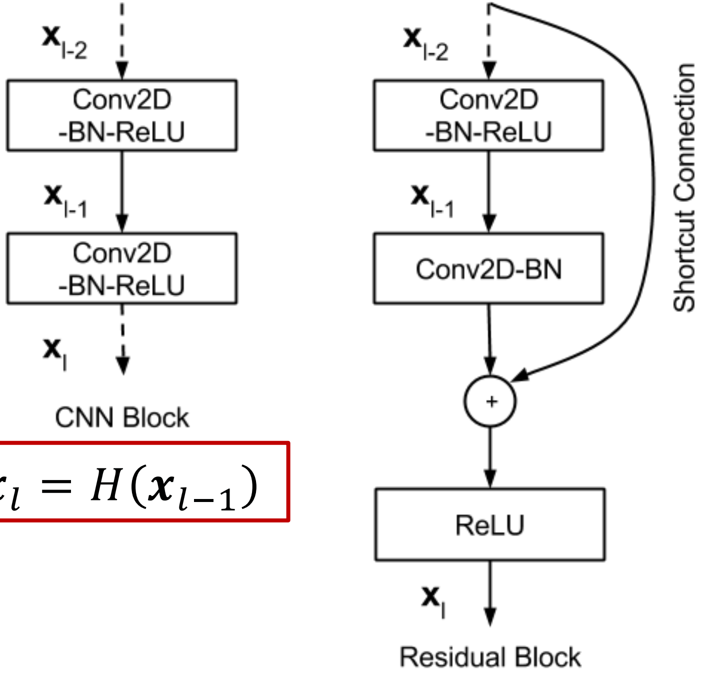
Vanishing Gradients



ResNet

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

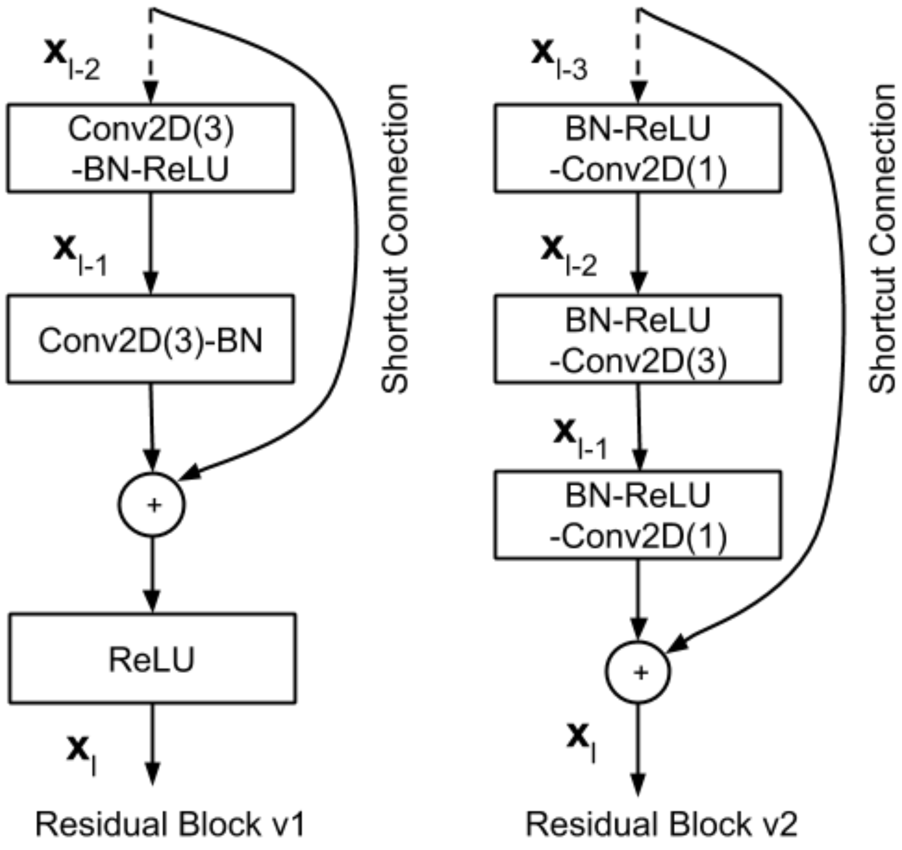
$$\boxed{\boldsymbol{x}_{l-1} = H(\boldsymbol{x}_{l-2})}$$



$$\boxed{\boldsymbol{x}_l = H(\boldsymbol{x}_{l-1})}$$

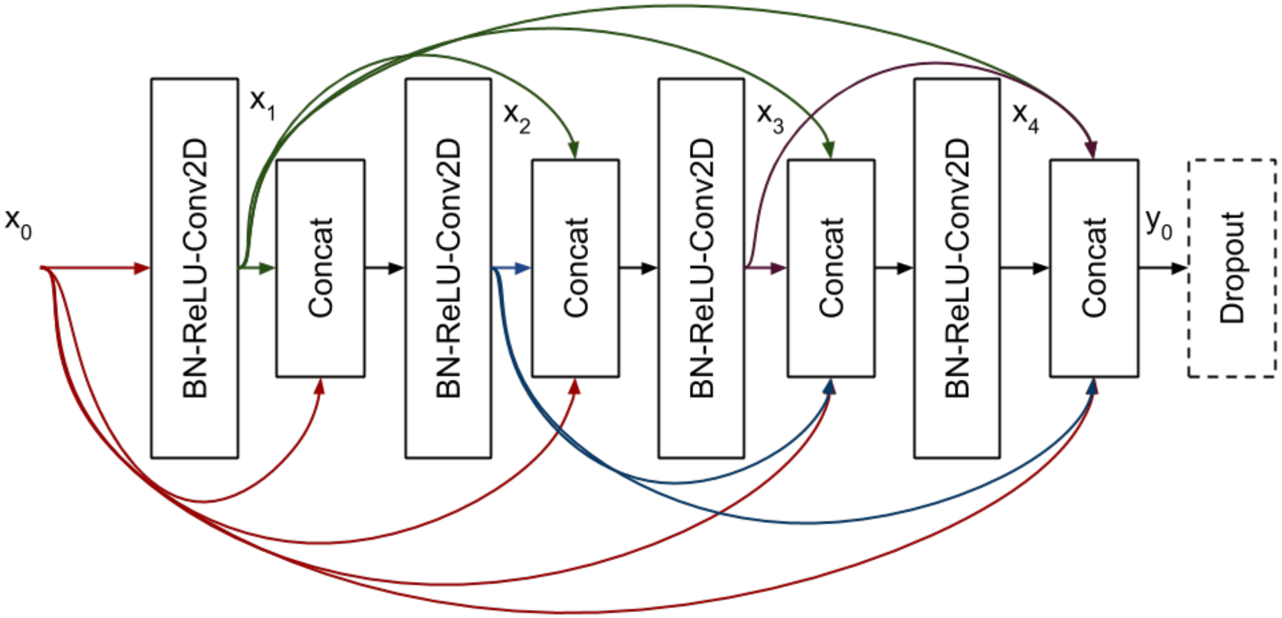
$$\boxed{\boldsymbol{x}_l = ReLU(F(\boldsymbol{x}_{l-1}) + \boldsymbol{x}_{l-2})}$$

Improved ResNet



DenseNet

Why not interconnect all layers?



Batch Normalization

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

Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift)

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

Batch Normalization

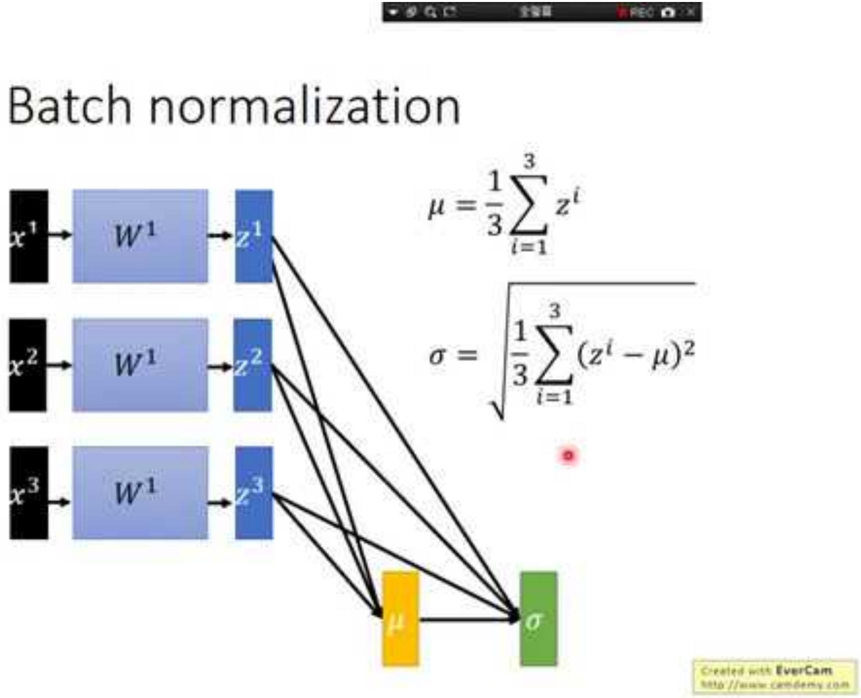
Applied to any input or hidden layer

$$\mathbf{H}' = (\mathbf{H} - \boldsymbol{\mu})/\boldsymbol{\sigma}$$

batch-wise operation

where **H** matrix is a minibatch of activations of the layer to normalize

μ vector of mean of each unit, **σ** vector of std of each unit



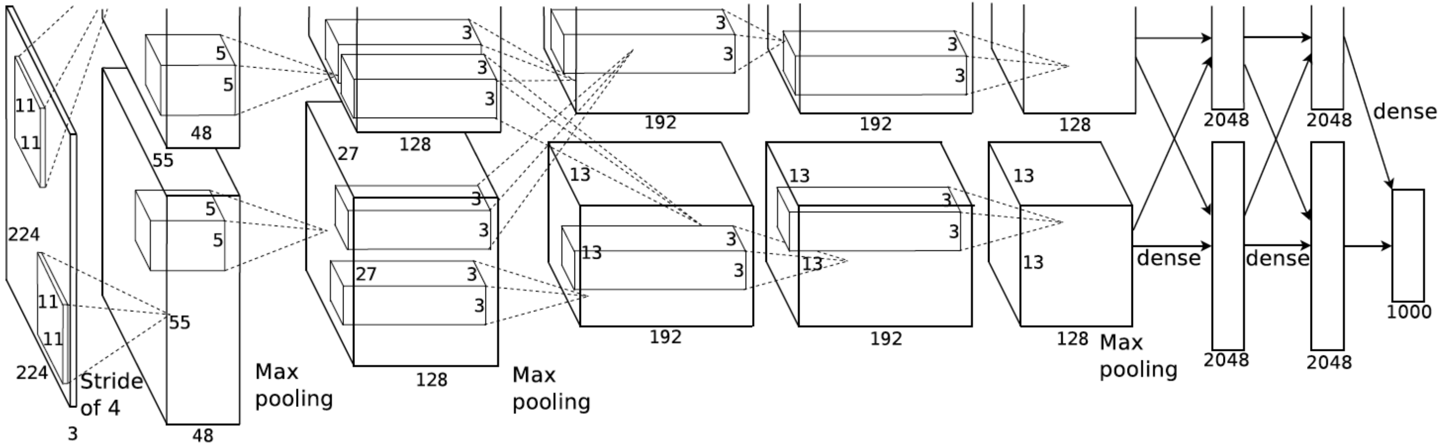
Assignment (Due: Oct 10)

1. Build a classifier on CIFAR10 dataset using
   a. MLP
   b. CNN
2. The last layer is Dense
3. Up to you to determine the hyperparameters (ie kernel size, # hidden units, # of filters, learning rate, etc) but the optimizer must be SGD
4. Show your solution using Jupyter notebook shared via your github
5. Implement using Keras or Pytorch
6. Compare performance of both networks (best MLP vs best CNN)
7. Best performing network (for MLP and CNN) has additional points

In Summary

CNN is parameter efficient, parallelizable and translation equivariant (invariant with depth) layer

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



AlexNet [Krizhevsky et al (2012)]