# Convolution Neural Network

Arun Chauhan

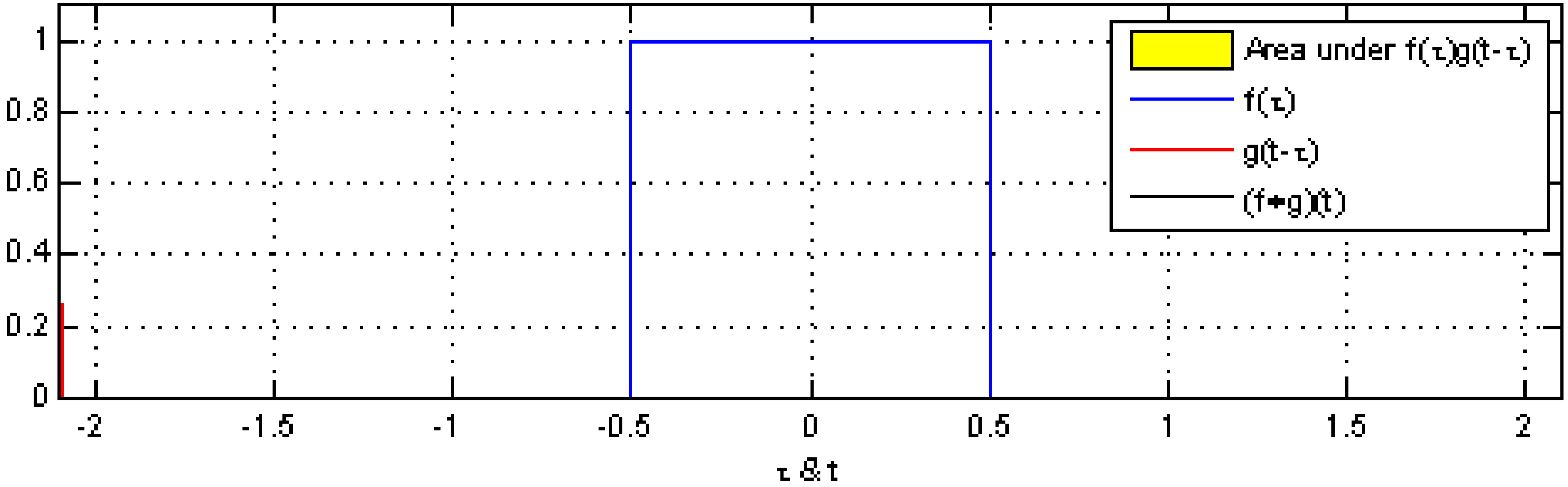
Computer Science and Engineering
Indian Institute of Information Technology Dharwad

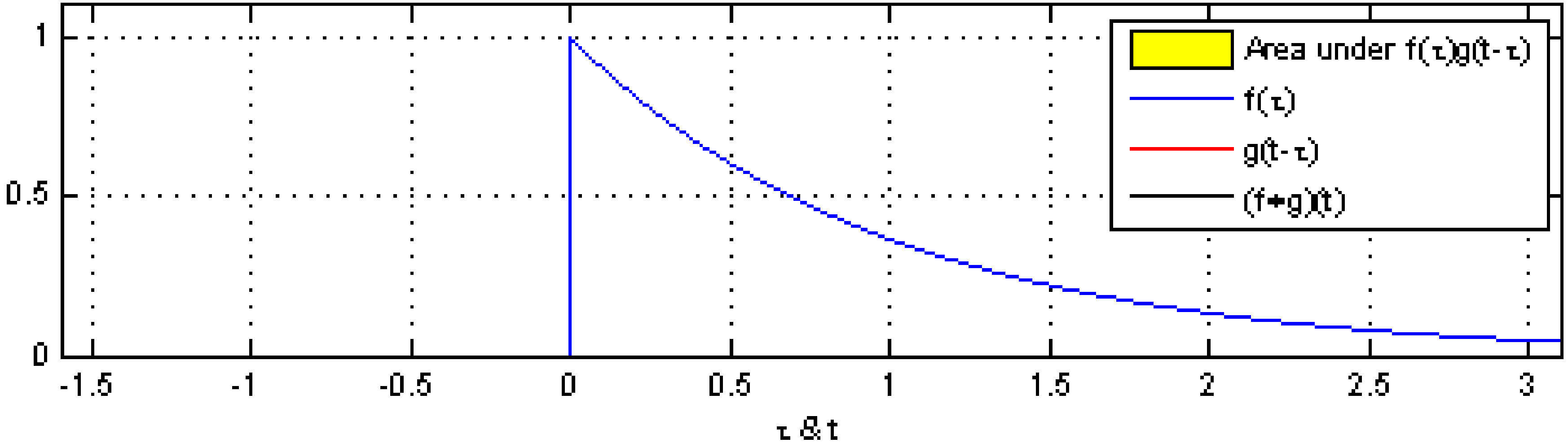
# What is Convolution?

$f$: Input

$g$: Kernel

$(f * g)(t)$: Feature Map



# Convolution on discrete data

## 1 D Convolution:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

## 2 D Convolution:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

## Convolution is commutative:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

## Cross Corelation same as Convolution:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

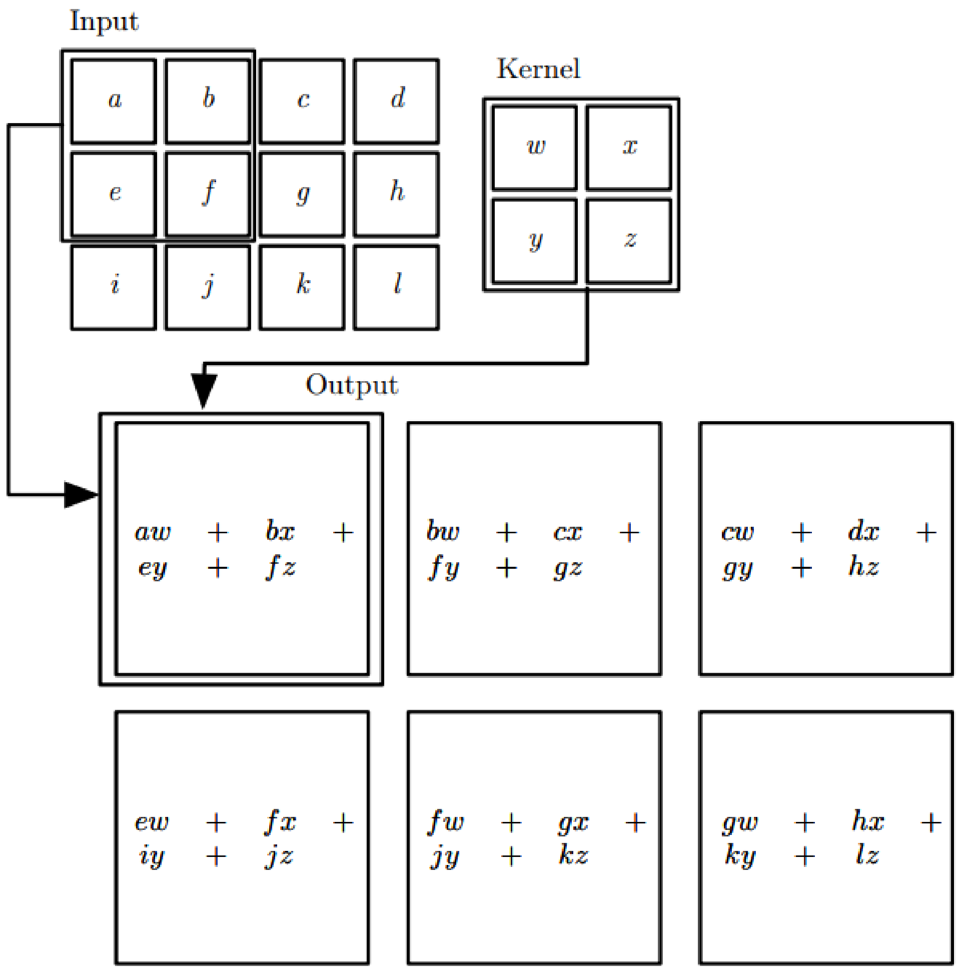
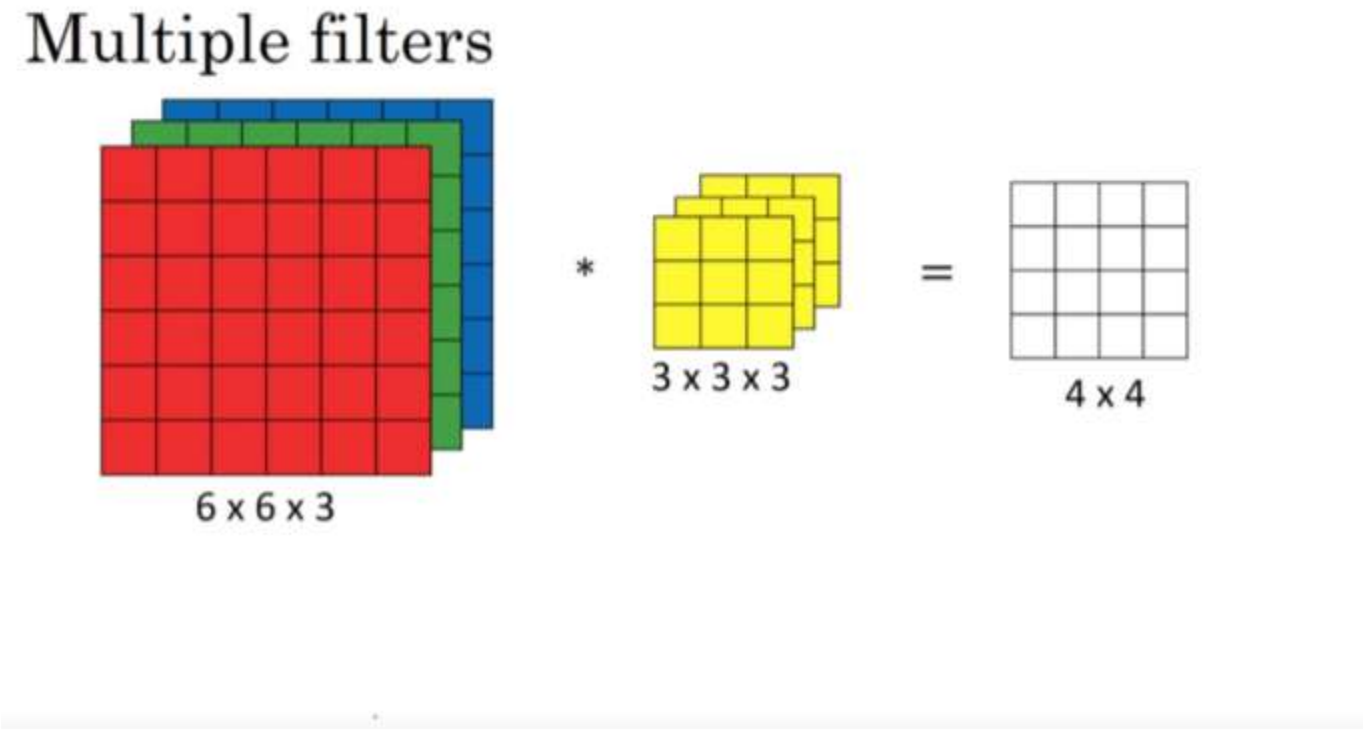
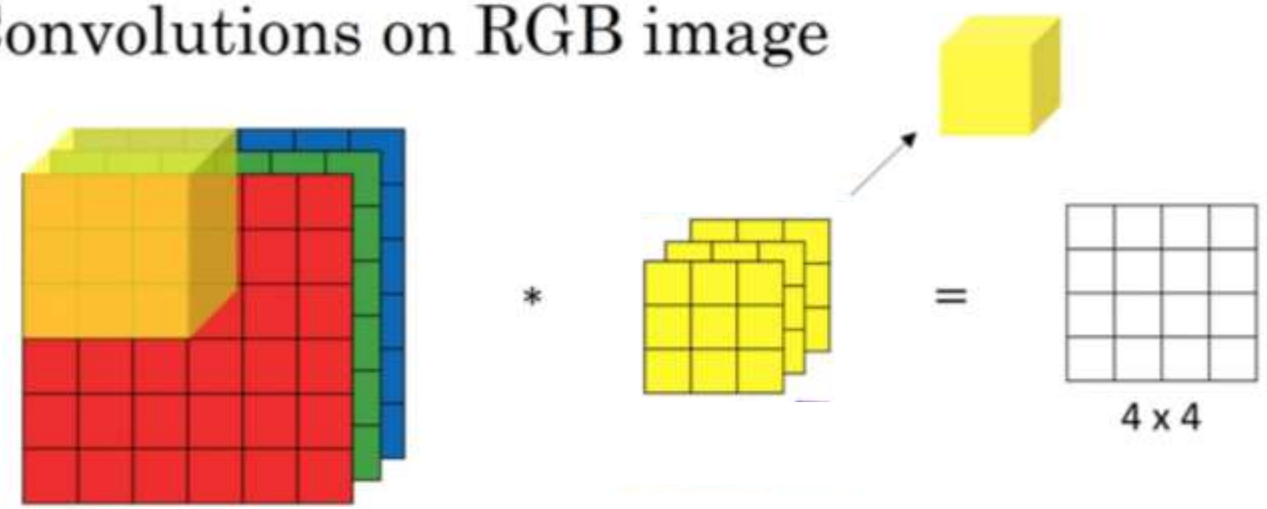



Figure 9.1

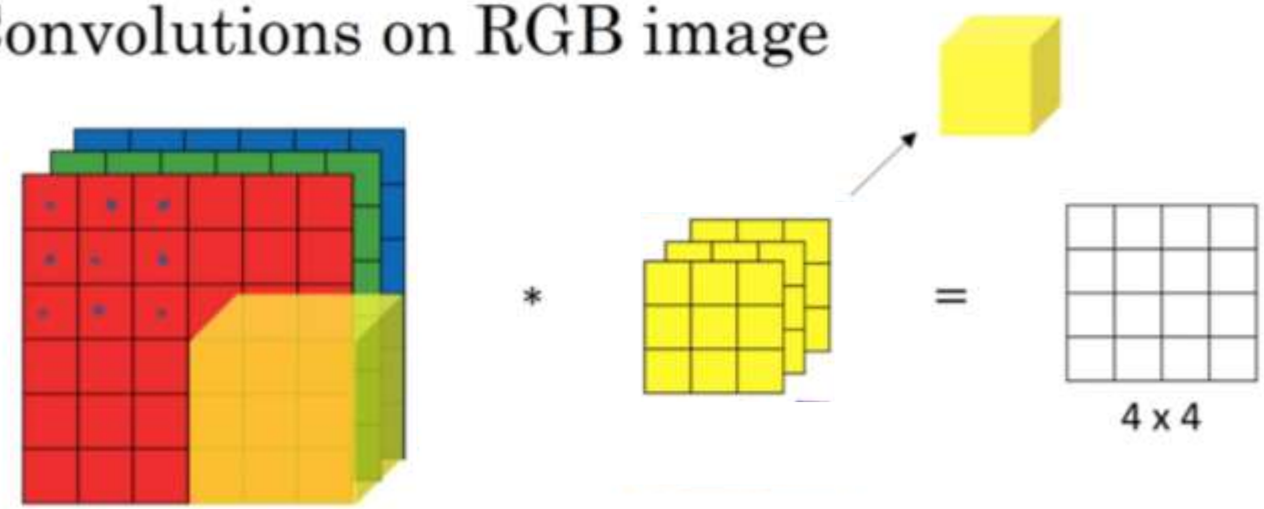
# Convolutions Over Volume



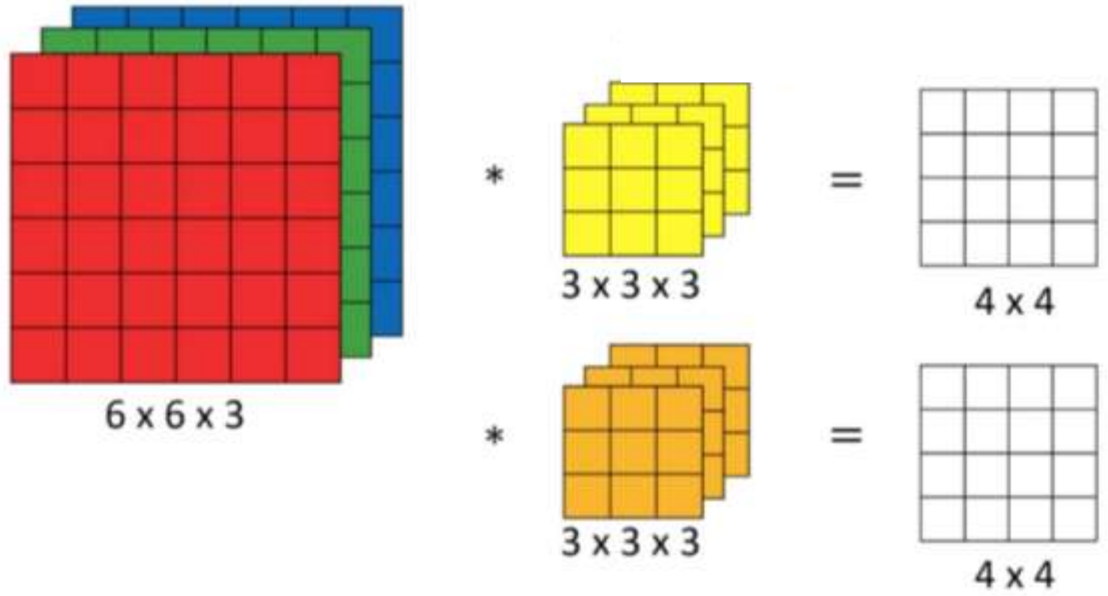
# Convolutions on RGB image



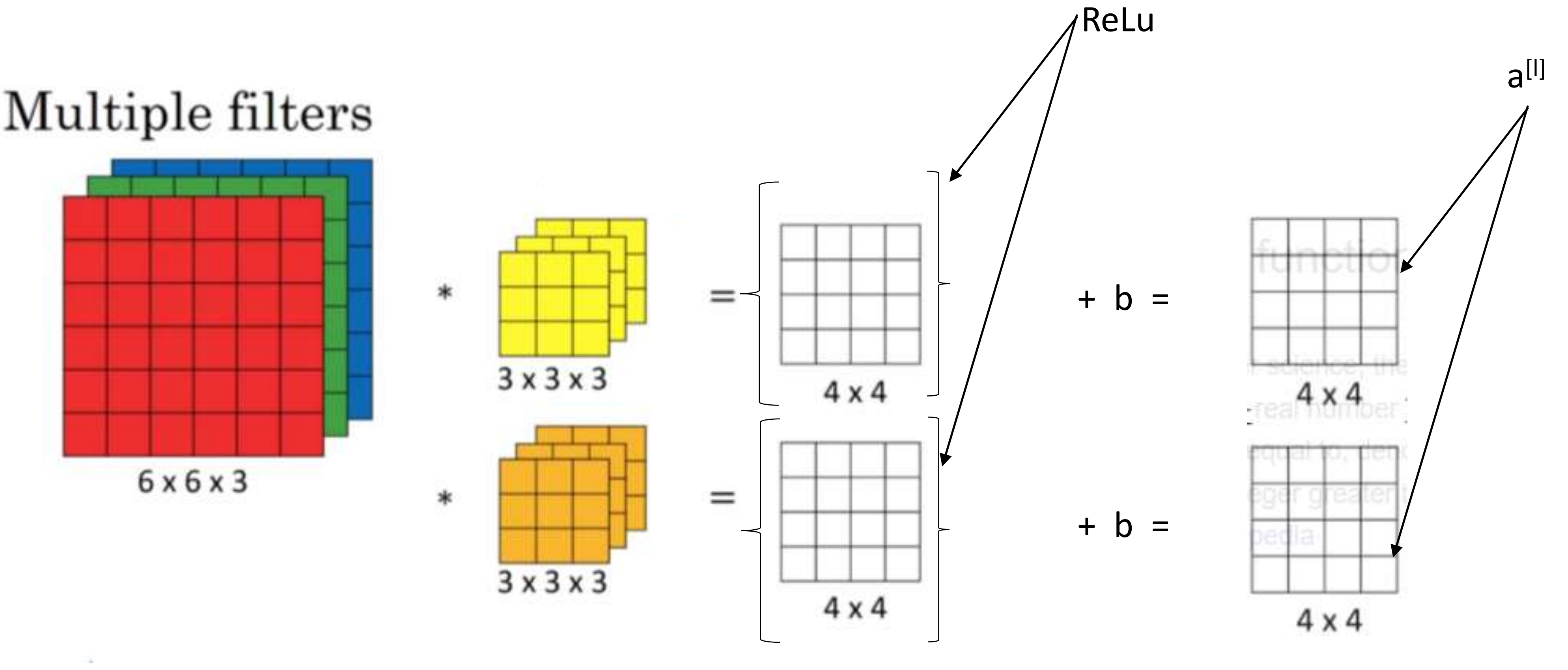
# Convolutions on RGB image



# Multiple filters



# One Layer of a Convolutional Network



# Motivation

- Convolution leverages three important ideas that can help improve a machine learning system:
  1. Sparse interaction.
  2. Parameter sharing.
  3. Equivariant representations.
  4. Also, provides a means for working with inputs of variable size.

# Key Idea

- Replace matrix multiplication in neural nets with convolution
- Everything else stays the same
  - Maximum likelihood
  - Back-propagation
  - etc.

# Three Operations

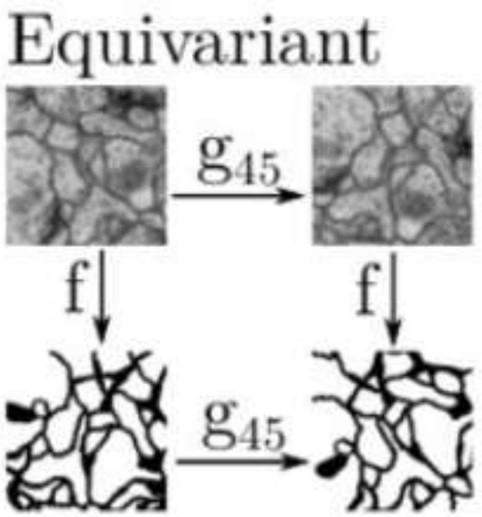
- Convolution: like matrix multiplication
  - Take an input, produce an output (hidden layer)
- "Deconvolution": like multiplication by transpose of a matrix
  - Used to back-propagate error from output to input
  - Reconstruction in autoencoder / RBM
- Weight gradient computation
  - Used to backpropagate error from output to weights
  - Accounts for the parameter sharing

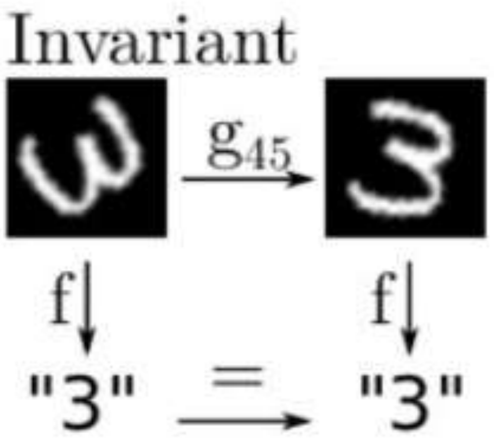
# Equivariance Vs Invariance

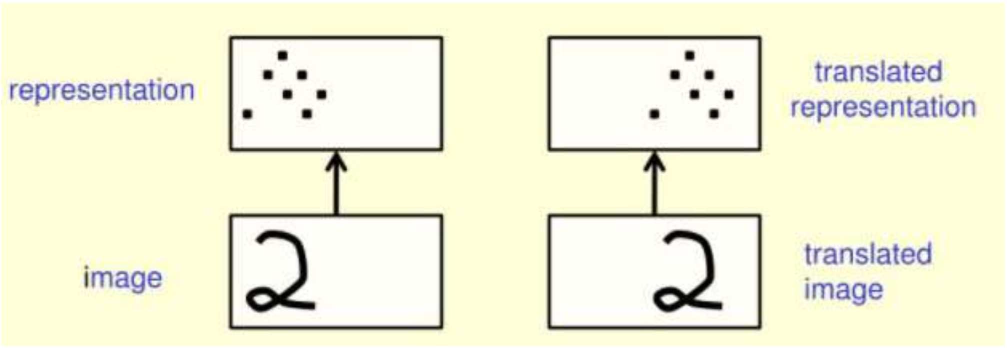
$$T(C(x)) = C(T(x))$$

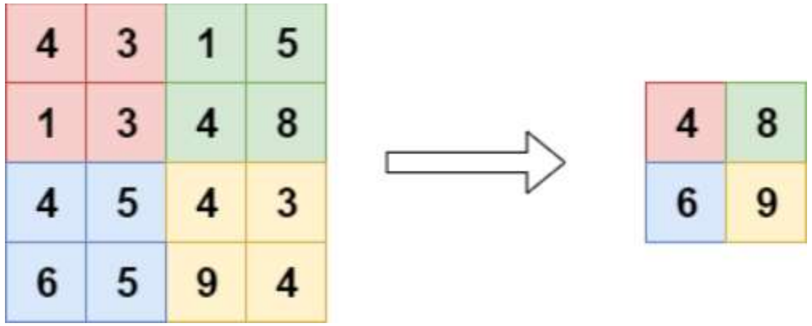
T: Translation

C: Convolution



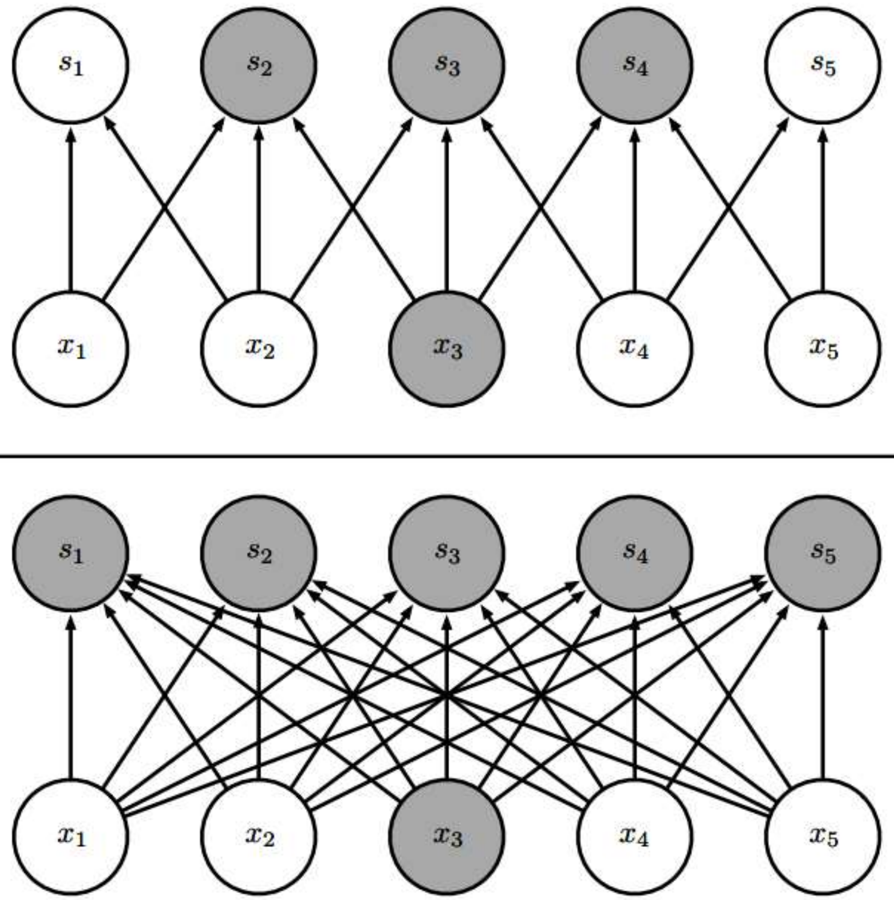



# Sparse Connectivity

Sparse connections due to small convolution kernel




$O(k \times n)$

Dense connections

$O(m \times n)$

Figure 9.2

(Goodfellow 2016)

# Sparse Connectivity

Sparse connections due to small convolution kernel

Dense connections

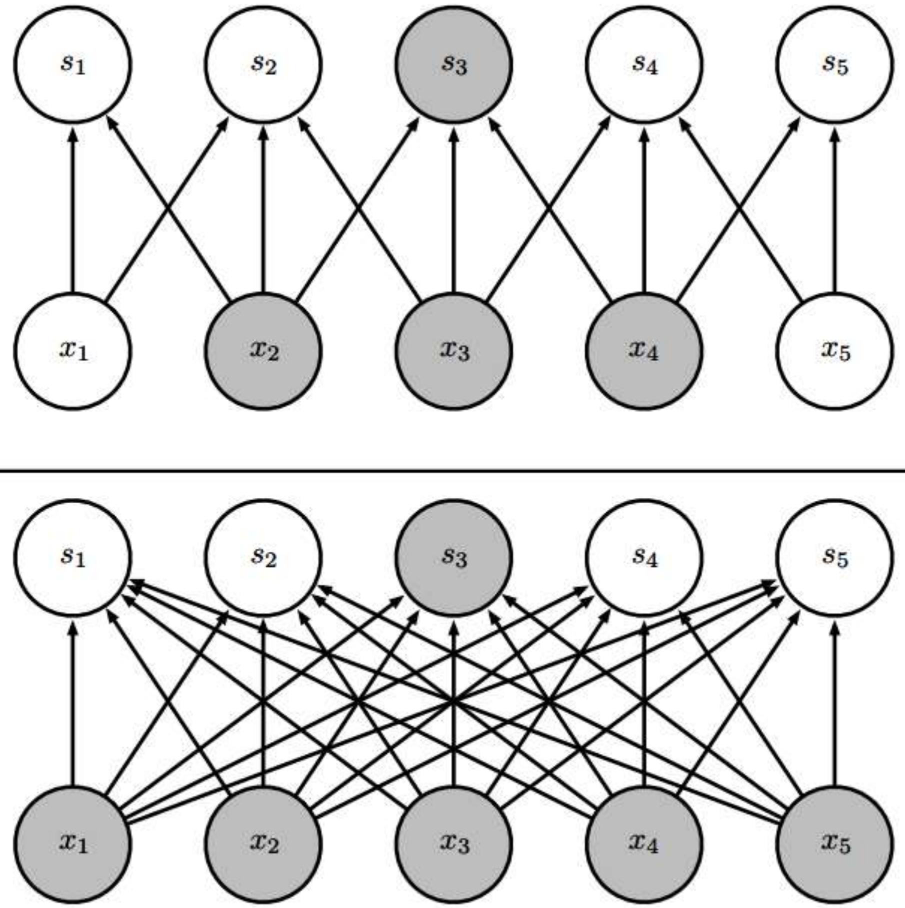



Figure 9.3

(Goodfellow 2016)

# Growing Receptive Fields

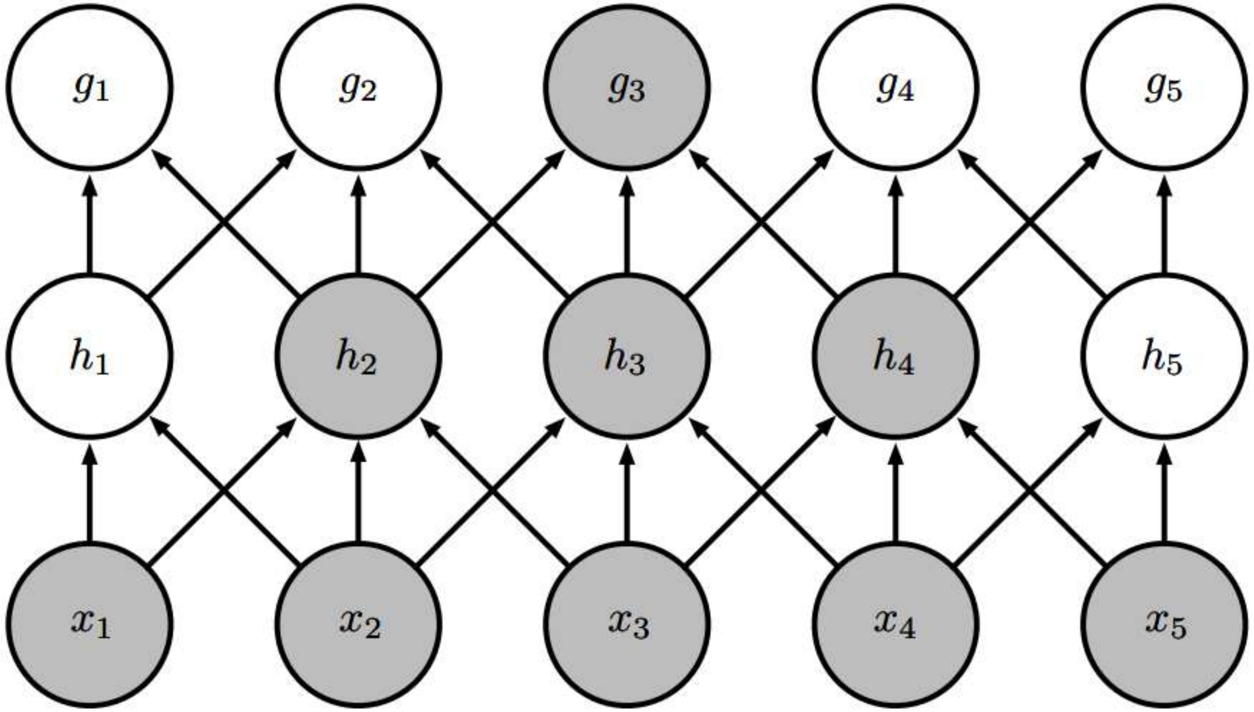
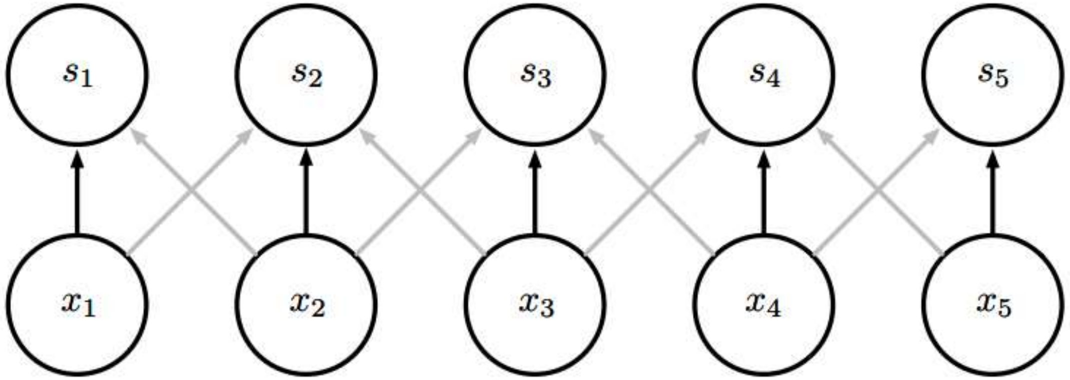



Figure 9.4

(Goodfellow 2016)

# Parameter Sharing

Convolution shares the same parameters across all spatial locations




Traditional matrix multiplication does not share any parameters

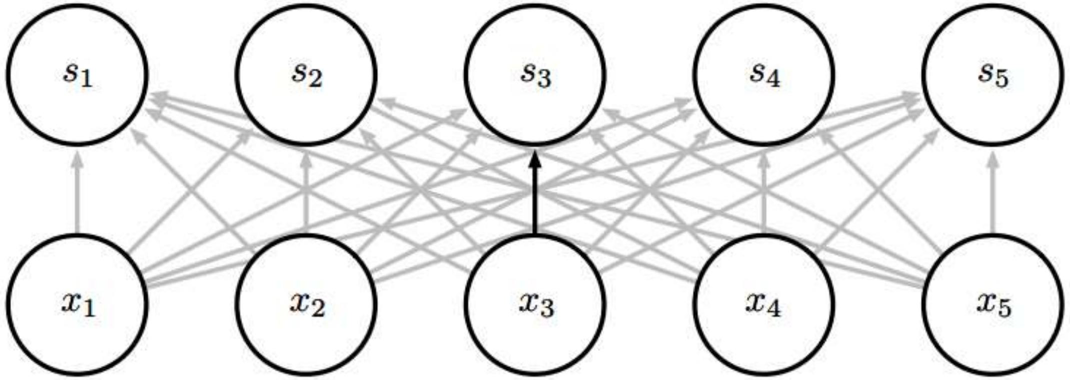



Figure 9.5

(Goodfellow 2016)

# Edge Detection by Convolution

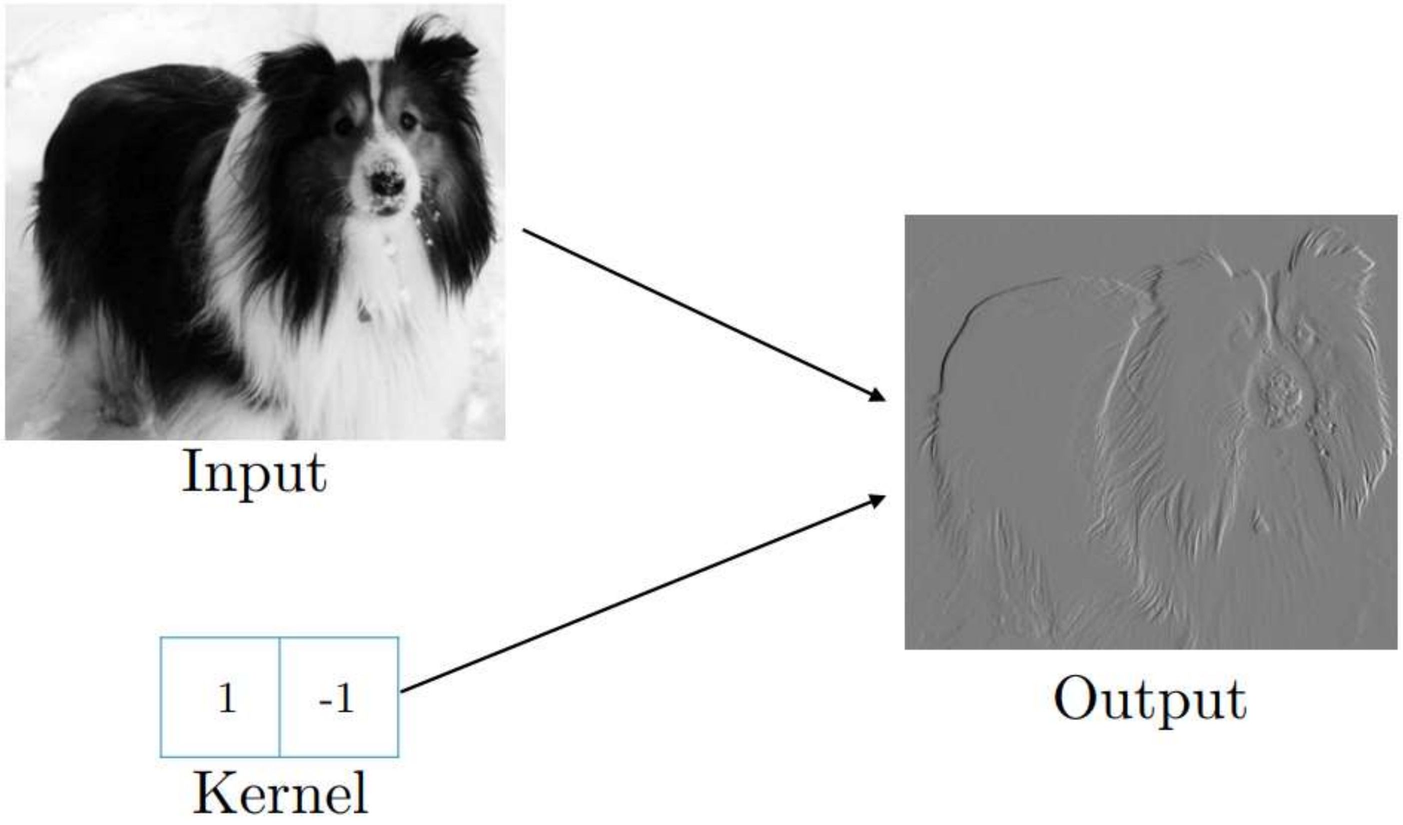



Figure 9.6

(Goodfellow 2016)

# Efficiency of Convolution

Input size: 320 by 280

Kernel size: 2 by 1

Output size: 319 by 280

| | Convolution | Dense matrix | Sparse matrix |
|---|---|---|---|
| **Stored floats** | 2 | 319*280*320*280 $>$ 8e9 | 2*319*280 = 178,640 |
| **Float muls or adds** | 319*280*3 = 267,960 | $>$ 16e9 | Same as convolution (267,960) |

(Goodfellow 2016)

# Convolutional Network Components

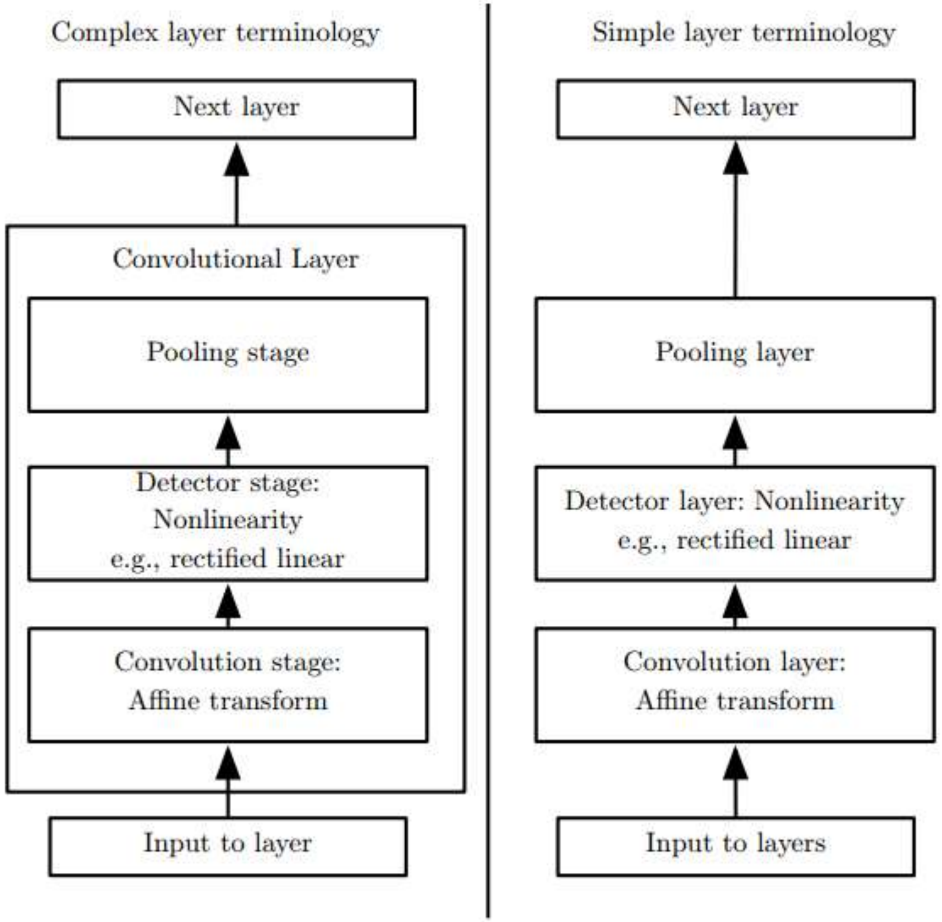



Figure 9.7

(Goodfellow 2016)

# Max Pooling and Invariance to Translation

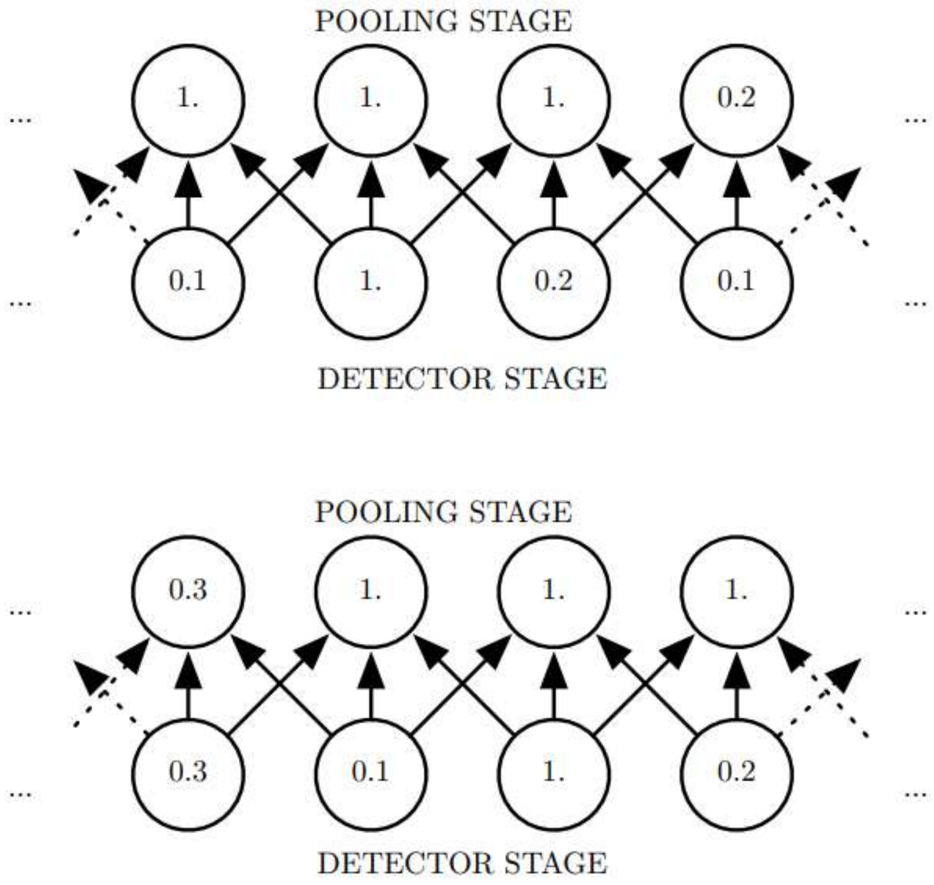



Figure 9.8

(Goodfellow 2016)

# Cross-Channel Pooling and Invariance to Learned Transformations

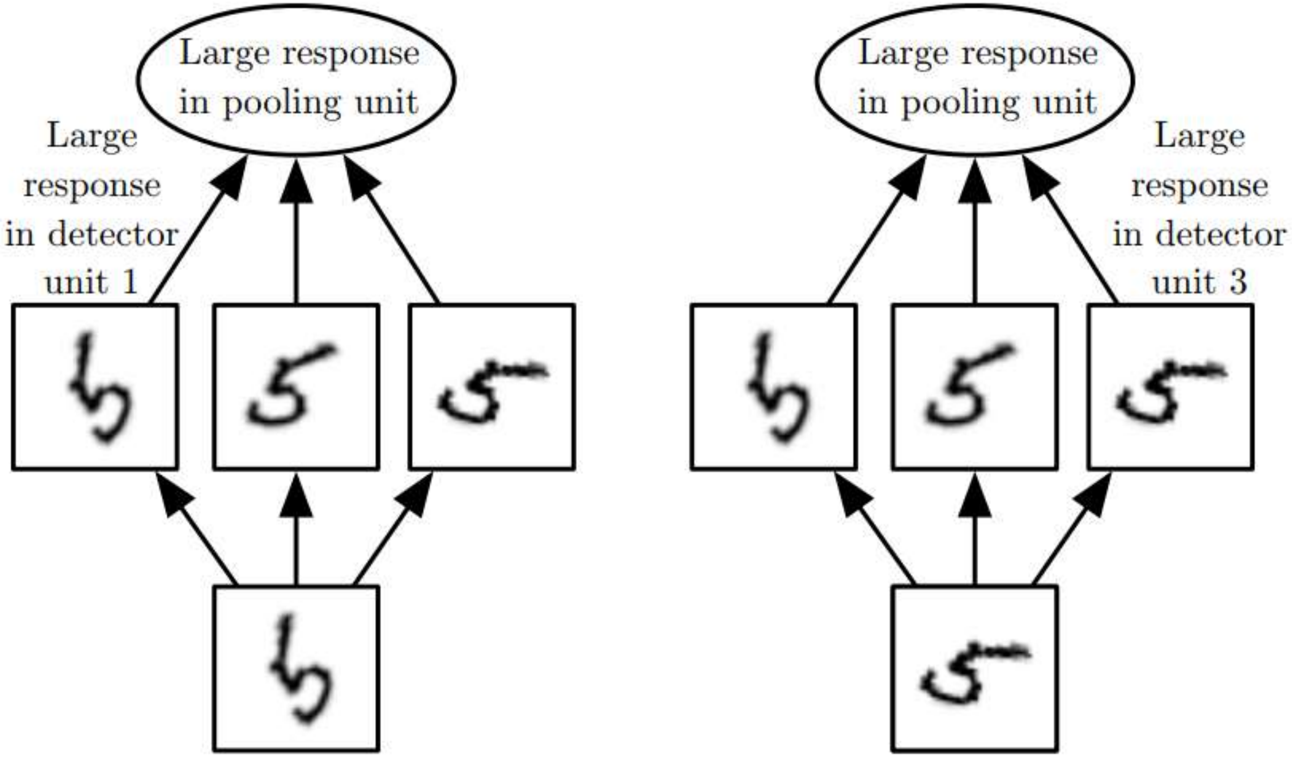



Figure 9.9

(Goodfellow 2016)

# Pooling with Downsampling

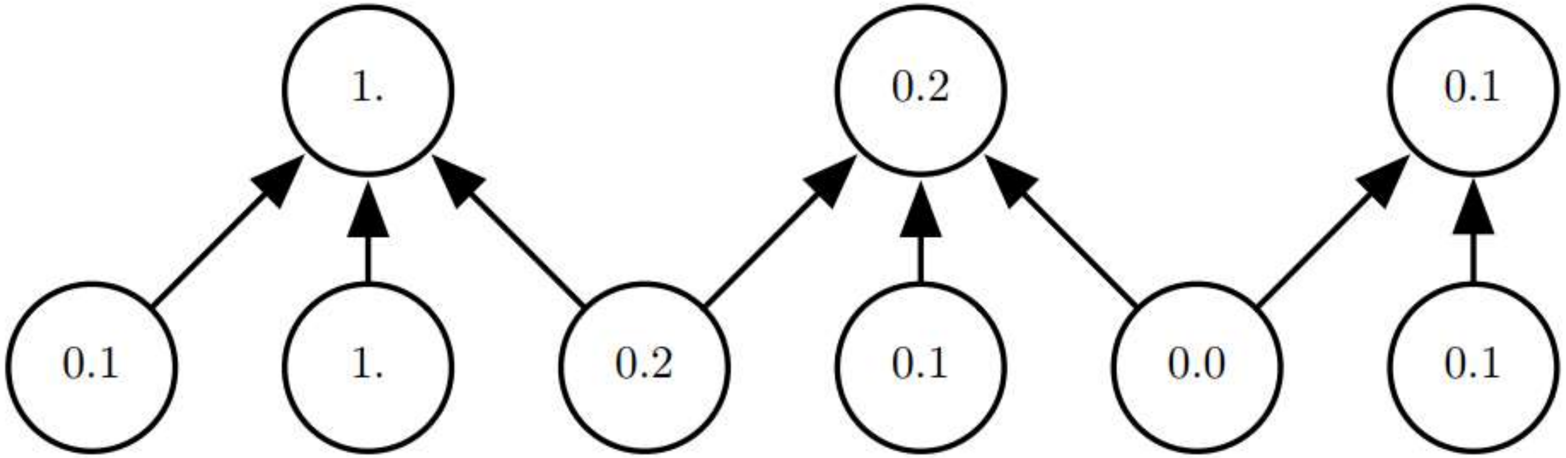
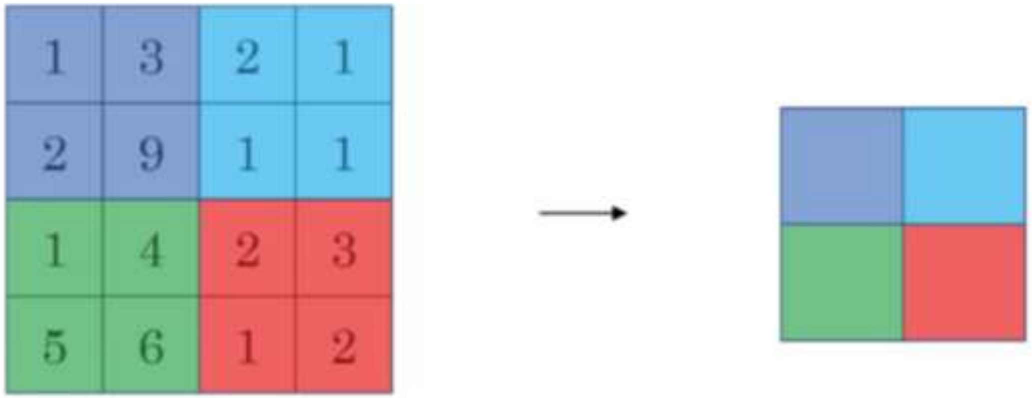



Figure 9.10

(Goodfellow 2016)

# Pooling layer: Max pooling



# Pooling layer: Max pooling

| | | | |
|---|---|---|---|
| 1 | 3 | 2 | 1 |
| 2 | 9 | 1 | 1 |
| 1 | 3 | 2 | 3 |
| 5 | 6 | 1 | 2 |

| | |
|---|---|
| 9 | 2 |
| 6 | 3 |

f=2
s=2
No parameters

## Pooling layer: Max pooling

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 2 | 1 | 3 |
| 2 | 9 | 1 | 1 | 5 |
| 1 | 3 | 2 | 3 | 2 |
| 8 | 3 | 5 | 1 | 0 |
| 5 | 6 | 1 | 2 | 9 |

| | | |
|---|---|---|
| 9 | | |
| | | |
| | | |

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

# Pooling layer: Max pooling

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 2 | 1 | 3 |
| 2 | 9 | 1 | 1 | 5 |
| 1 | 3 | 2 | 3 | 2 |
| 8 | 3 | 5 | 1 | 0 |
| 5 | 6 | 1 | 2 | 9 |

| | | |
|---|---|---|
| 9 | 9 | |
| | | |
| | | |

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

# Pooling layer: Max pooling

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 2 | 1 | 3 |
| 2 | 9 | 1 | 1 | 5 |
| 1 | 3 | 2 | 3 | 2 |
| 8 | 3 | 5 | 1 | 0 |
| 5 | 6 | 1 | 2 | 9 |

| | | |
|---|---|---|
| 9 | 9 | 5 |
| | | |
| | | |

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

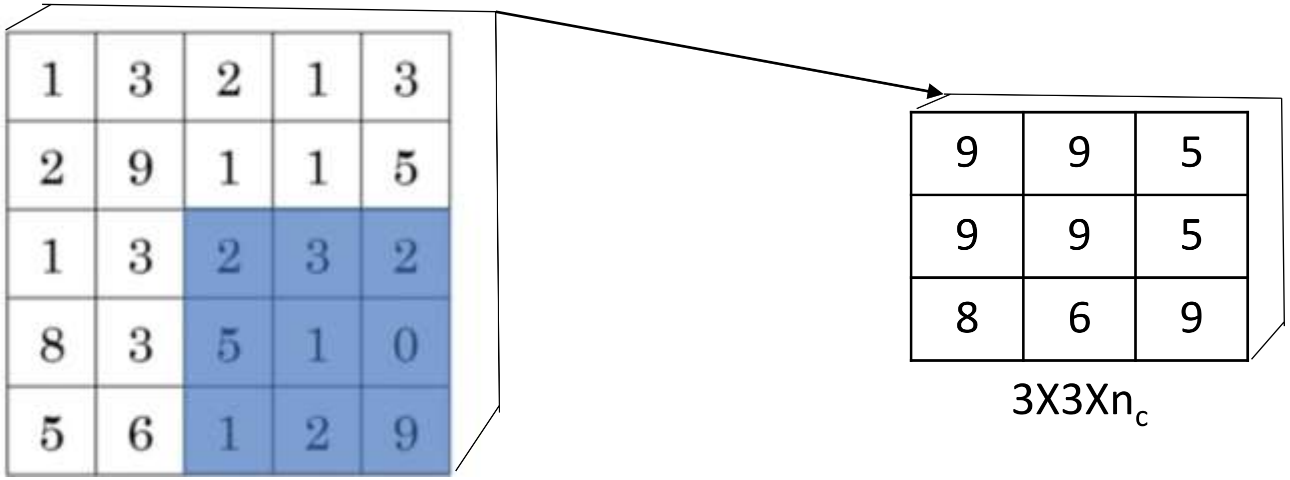
# Pooling layer: Max pooling

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 2 | 1 | 3 |
| 2 | 9 | 1 | 1 | 5 |
| 1 | 3 | 2 | 3 | 2 |
| 8 | 3 | 5 | 1 | 0 |
| 5 | 6 | 1 | 2 | 9 |

| | | |
|---|---|---|
| 9 | 9 | 5 |
| 9 | 9 | 5 |
| 8 | 6 | 9 |

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

# Pooling layer: Max pooling



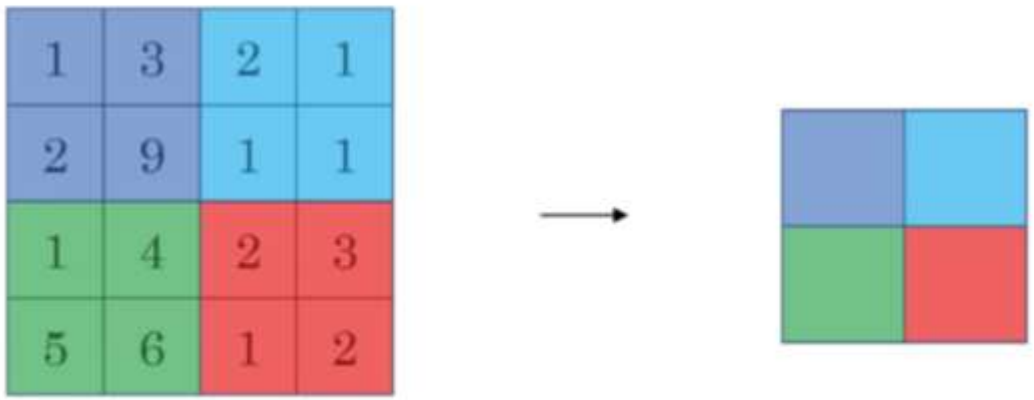

$5X5Xn_c$

$3X3Xn_c$

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

Stride = 1
f=3
p=0

# Pooling layer: Average pooling



# Pooling layer: Average pooling

| | | | |
|---|---|---|---|
| 1 | 3 | 2 | 1 |
| 2 | 9 | 1 | 1 |
| 1 | 3 | 2 | 3 |
| 5 | 6 | 1 | 2 |

| | |
|---|---|
| 3.75 | 1.25 |
| 4 | 2 |

f=2
s=2
No parameters

## Summary of pooling

Hyperparameters:

f : filter size

s : stride

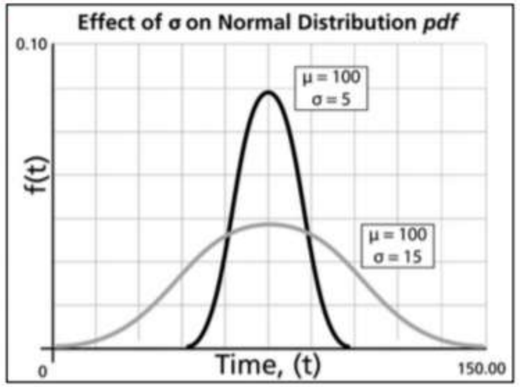
Max or average pooling

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \text{X} \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \text{X}\ n_C$$

# Convolution and Pooling as an Infinitely Strong Prior

# Weak and Strong Priors

- A weak prior
  - A distribution with high entropy
    - e.g., Gaussian with high variance
  - Data can move parameters freely
- A strong prior
  - It has very low entropy
    - E.g., a Gaussian with low variance
  - Such a prior plays a more active role in determining where the parameters end up

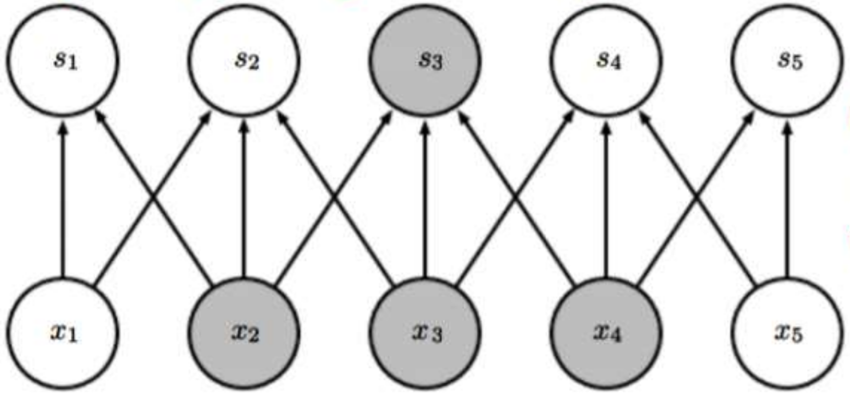


# Infinitely Strong Prior

- An infinitely strong prior places zero probability on some parameters
- It says that some parameter values are forbidden regardless of support from data
  - With an infinitely strong prior, irrespective of the data the prior cannot be changed

# Convolution as infinitely strong prior

- Convolutional net is similar to a fully connected net but with an infinitely strong prior over its weights
  - It says that the weights for one hidden unit must be identical to the weights of its neighbor, but shifted in space
  - Prior also says that the weights must be zero, except for in the small spatially contiguous receptive field assigned to that hidden unit

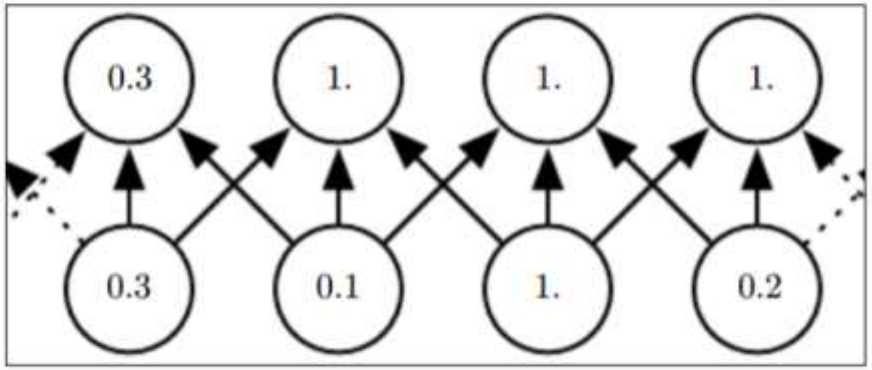



Convolution with a kernel of width $3$
$s_3$ is a hidden unit. It has $3$ weights which are the same as for $s_4$

  - Convolution introduces an infinitely strong prior probability distribution over the parameters of a layer
    - This prior says that the function the layer should learn contains only local interactions and is equivariant to translation

# Pooling as an Infinitely strong prior

- The use of pooling is an infinitely strong prior that each unit should be invariant to small translations
- Maxpooling example:

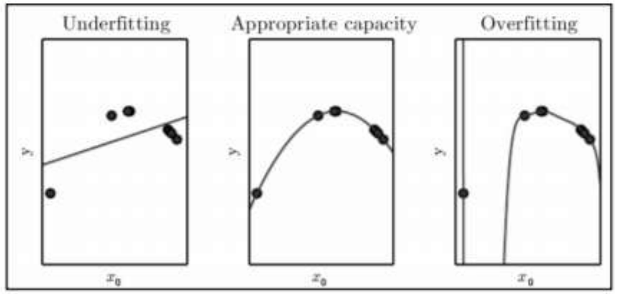


# Implementing as a prior

- Implementing a convolutional net as a fully connected net with an infinitely strong prior would be extremely computationally wasteful
- But thinking of a convolutional net as a fully connected net with an infinitely strong prior can give us insights into how convolutional nets work

# Key Insight: Underfitting

- Convolution and pooling can cause under-fitting
  - Under-fitting happens when model has high bias
- Convolution and pooling are only useful when the assumptions made by the prior are reasonably accurate
- Pooling may be inappropriate in some cases
  - If the task relies on preserving spatial information
    - Using pooling on all features can increase training error




High Bias/Underfit can be countered by:
1. Add hidden layers
2. Increase hidden units/layer
3. Decrease regular. parameter $\lambda$
4. Add features

# When pooling may be inappropriate

- Some convolutional architectures are designed to use pooling on some channels but not on other channels
  - In order to get highly invariant features and features that will not under-fit when the translation invariance prior is incorrect
- When a task involves incorporating information from a distant location
  - In which case,  prior imposed by convolution may be inappropriate

# Comparing models with/without convolution

- Convolutional models have spatial relationships
- In benchmarks of statistical learning performance we should only compare convolutional models to other convolutional models – since they have knowledge of spatial relationships hard-coded
- Models without convolution will be able to learn even if we permuted all pixels in the image
- Permutation invariance: $f\,(x_1,x_2,x_3)=f\,(x_2,x_1,x_3)=f(x_3,x_1,x_2)$
- There are separate benchmarks for models that are permutation invariant

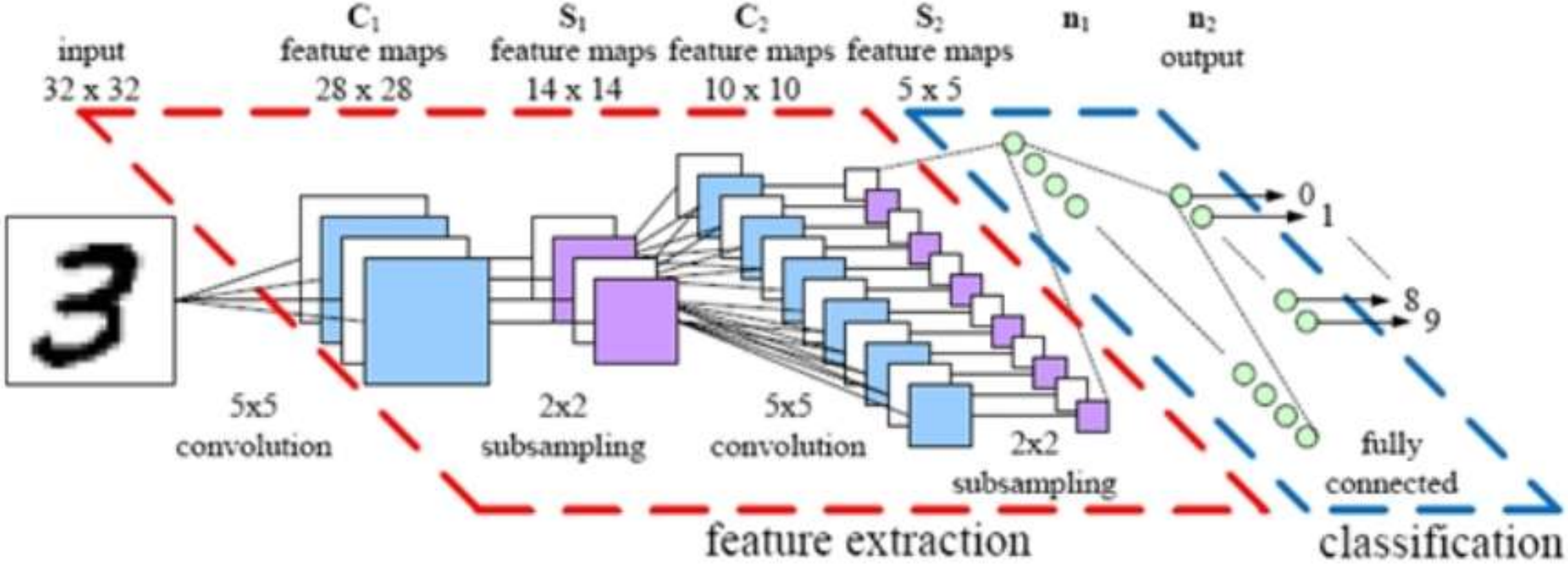
# Variants of the Basic Convolution Function

# Definition of $4\text{-D}$ kernel tensor

- Assume we have a $4\text{-D}$ kernel tensor $\mathbf{K}$ with element $K_{i,j,k,l}$ giving the connection strength between
  - a unit in channel $i$ of the output and
  - a unit in channel $j$ of the input,
  - with an offset of $k$ rows and $l$ columns between output and input units
- Assume our input consists of observed data $\mathbf{V}$ with element $V_{i,j,k}$ giving the value of the input unit
  - within channel $i$ at row $j$ and column $k$.
- Assume our output consists of $\mathbf{Z}$ with the same format as $\mathbf{V}$.
- If $\mathbf{Z}$ is produced by convolving $\mathbf{K}$ across $\mathbf{V}$ without flipping $\mathbf{K}$, then

$$\boxed{Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}}$$

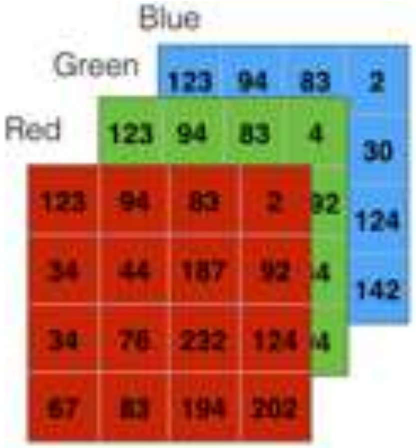
# Simple Convolutional Network Example[1]




$$\boxed{Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}}$$

$l = 3$

$m, n = 3,3$




| | | |
|---|---|---|
| 9 | 9 | 5 |
| 9 | 9 | 5 |
| 8 | 6 | 9 |

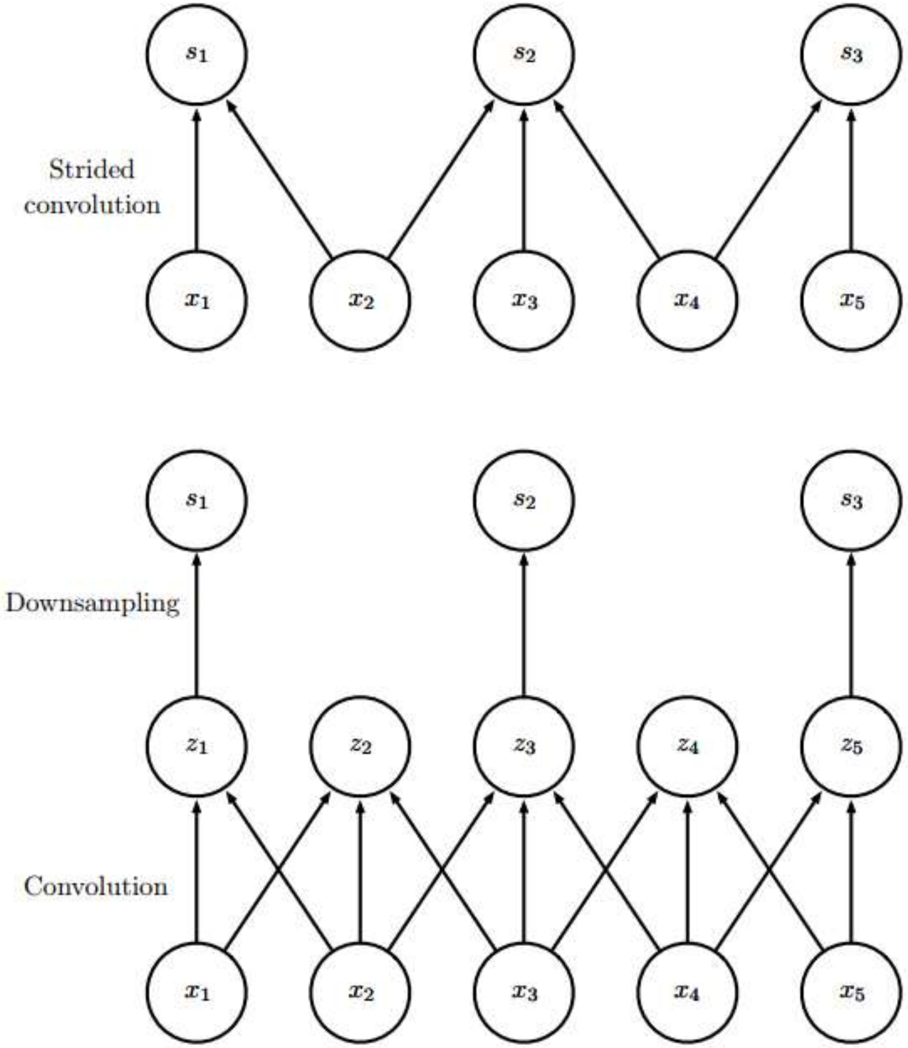
# Convolution with a stride: Definition

- We may want to skip over some positions in the kernel to reduce computational cost
  - At the cost of not extracting fine features
- We can think of this as down-sampling the output of the full convolution function
- If we want to sample only every $s$ pixels in each direction of output, then we can define a down-sampled convolution function $c$ such that

$$Z_{i,j,k} = c(\mathsf{K}, \mathsf{V}, s)_{i,j,k} = \sum_{l,m,n} \left[ V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n} \right]$$

- We refer to $s$ as the stride. It is possible to define a different stride for each direction

# Convolution with Stride




stride of 2

Convolution with a stride greater than one pixel is mathematically equivalent to convolution with a unit stride followed by down-sampling.

Two-step approach is computationally wasteful, because it discard many values that are discarded

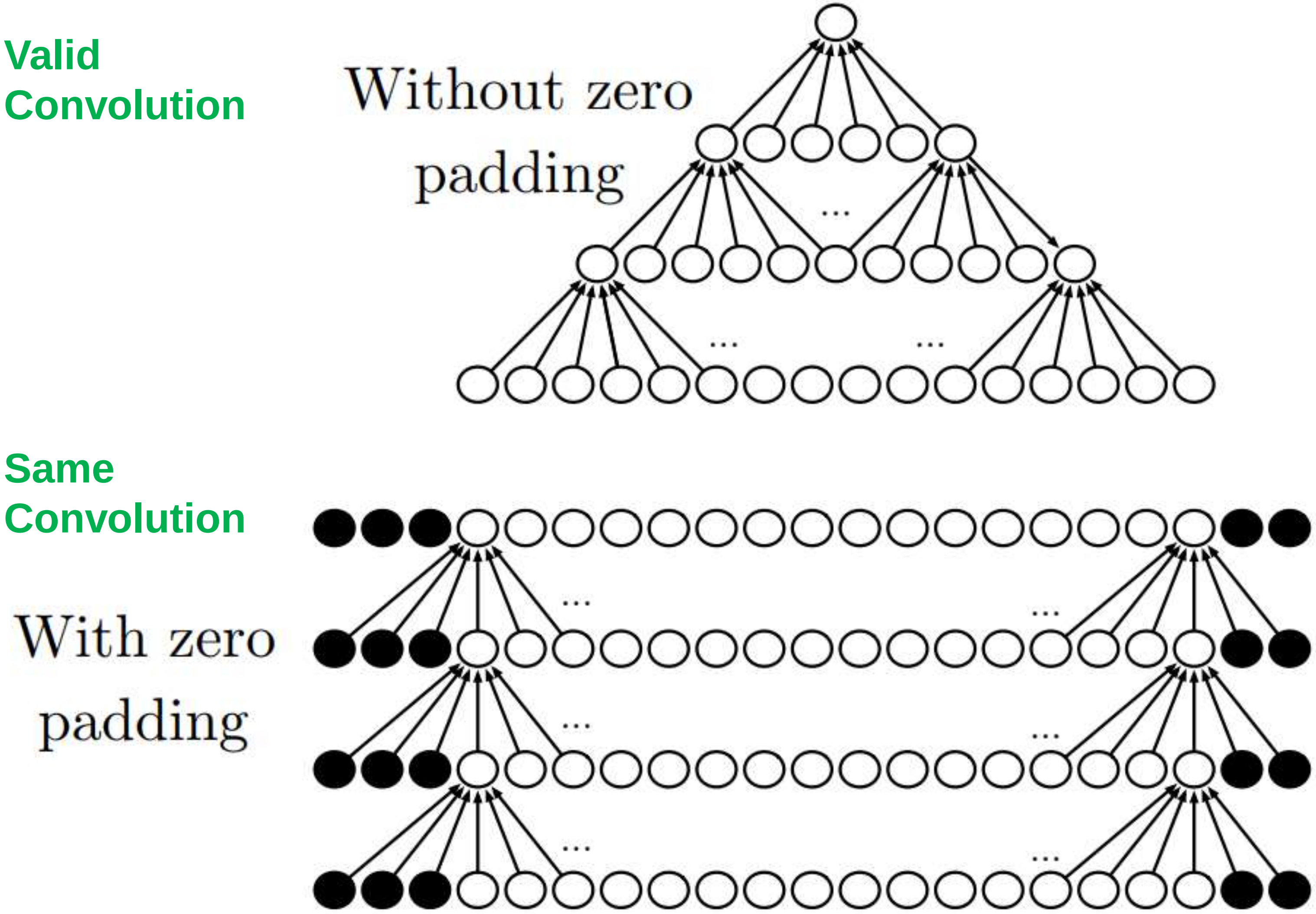
(Goodfellow 2016)

# Zero Padding Controls Size

**Valid Convolution**

- Kernel width, k: 6
- No pooling
- Shrink by (k-1) pixel at every layer

**Same Convolution**

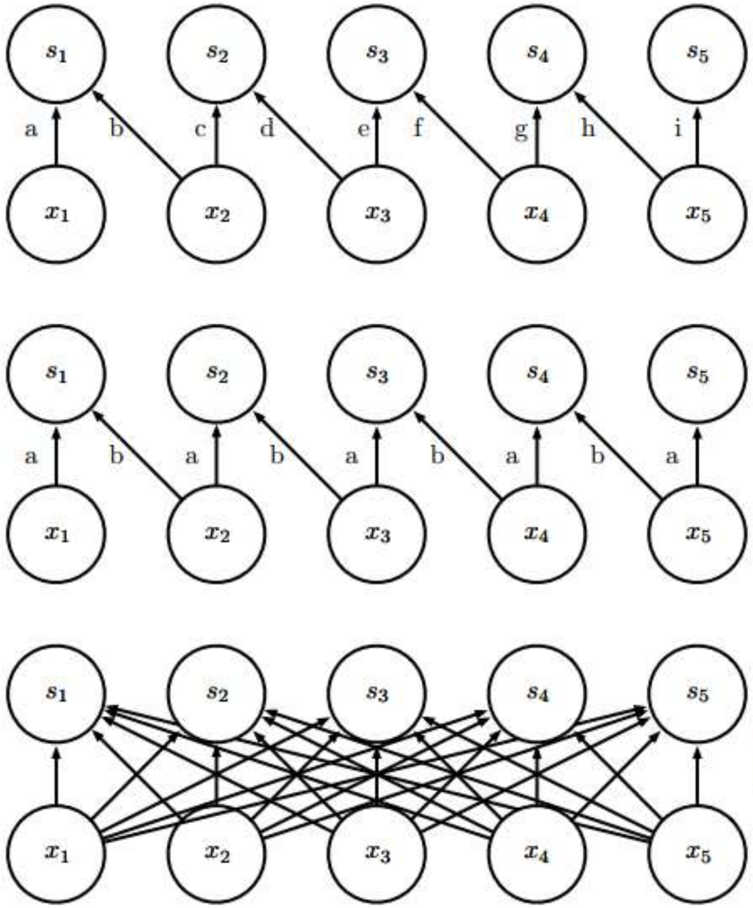



$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor = n$$

Figure 9.13

(Goodfellow 2016)

# Kinds of Connectivity




Local connection: like convolution, but no sharing

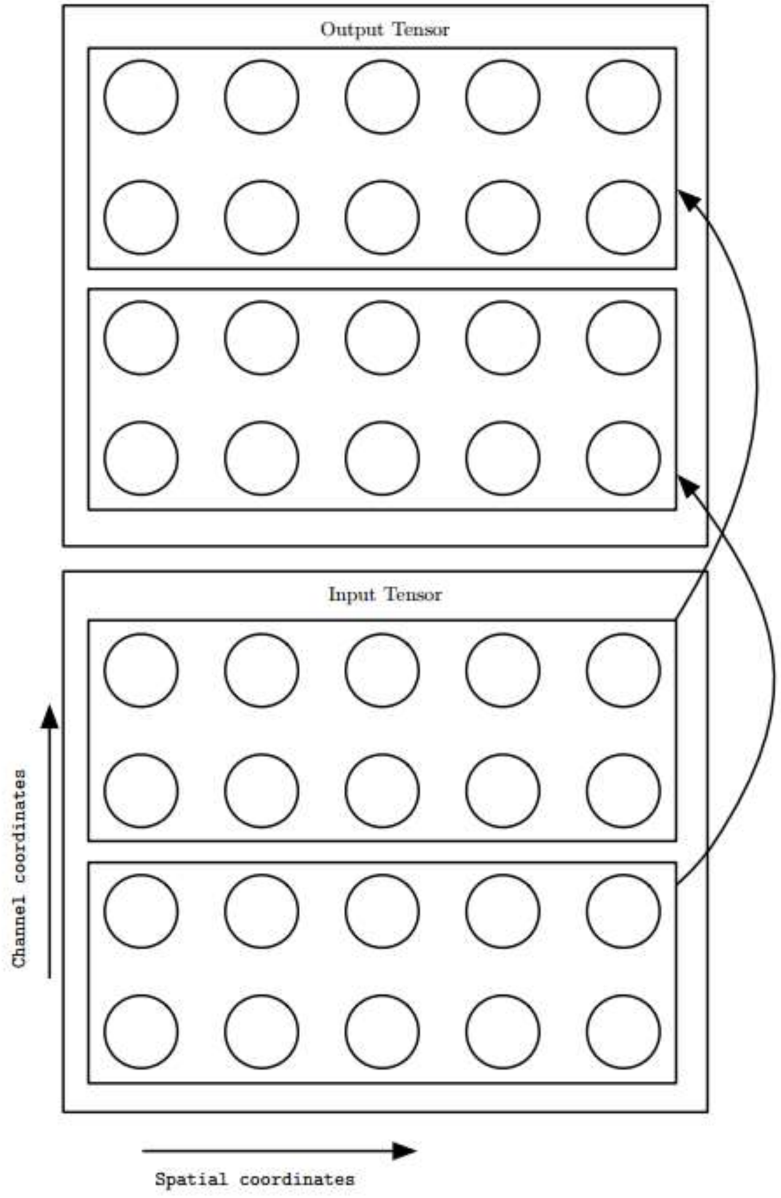
Convolution

Fully connected

Figure 9.14

(Goodfellow 2016)

# Partial Connectivity Between Channels

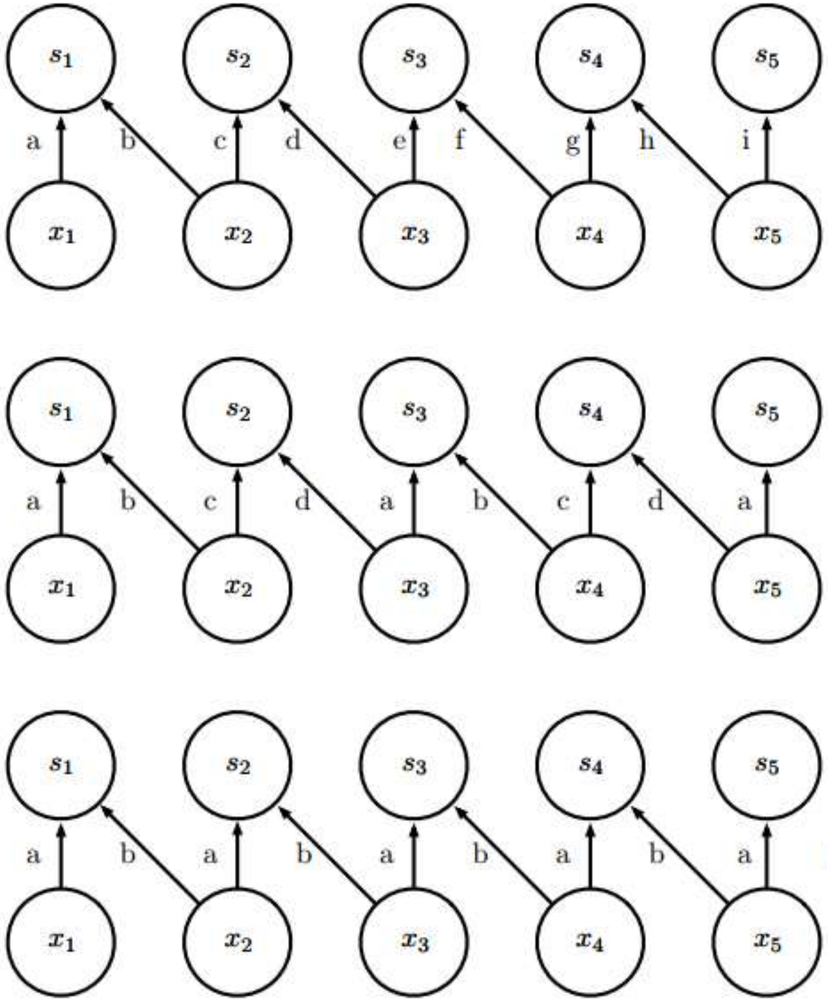



- Modeling interactions between few channels allows fewer parameters to:
  - Reduce memory, increase statistical efficiency, reduce computation for forward/back-propagation.
  - It accomplishes these goals without reducing no.of hidden units.

Figure 9.15

(Goodfellow 2016)

# Tiled convolution




Local connection (no sharing)

Tiled convolution (cycle between groups of shared parameters)

Convolution (one group shared everywhere)

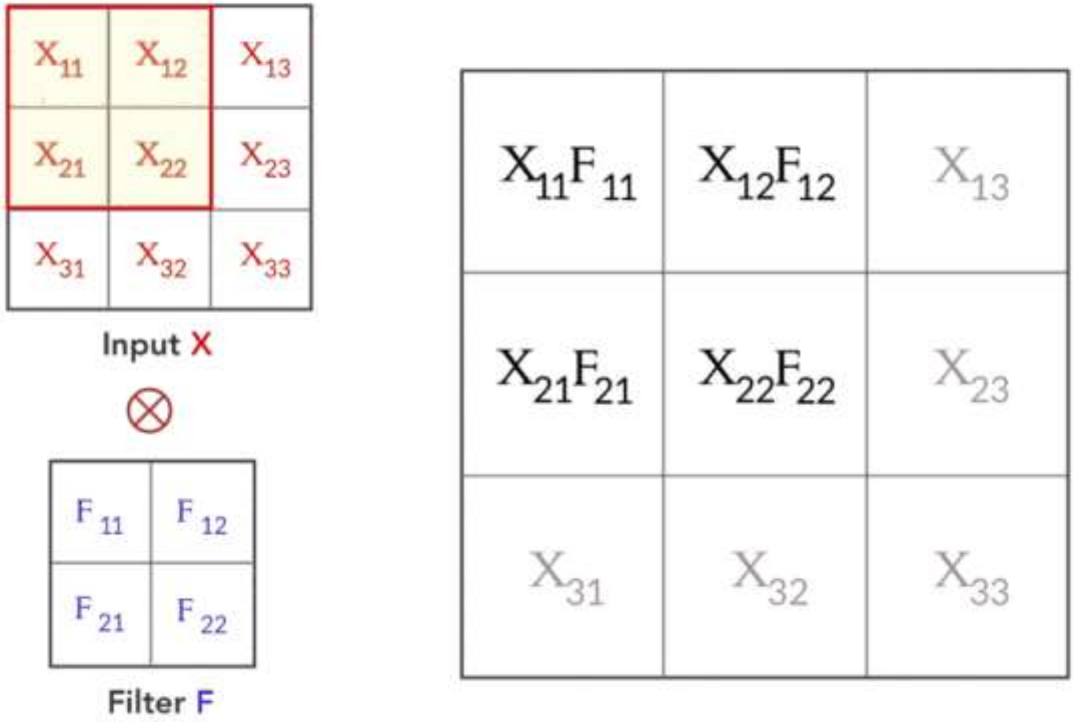
Figure 9.16

(Goodfellow 2016)

# Backpropagation in CNN

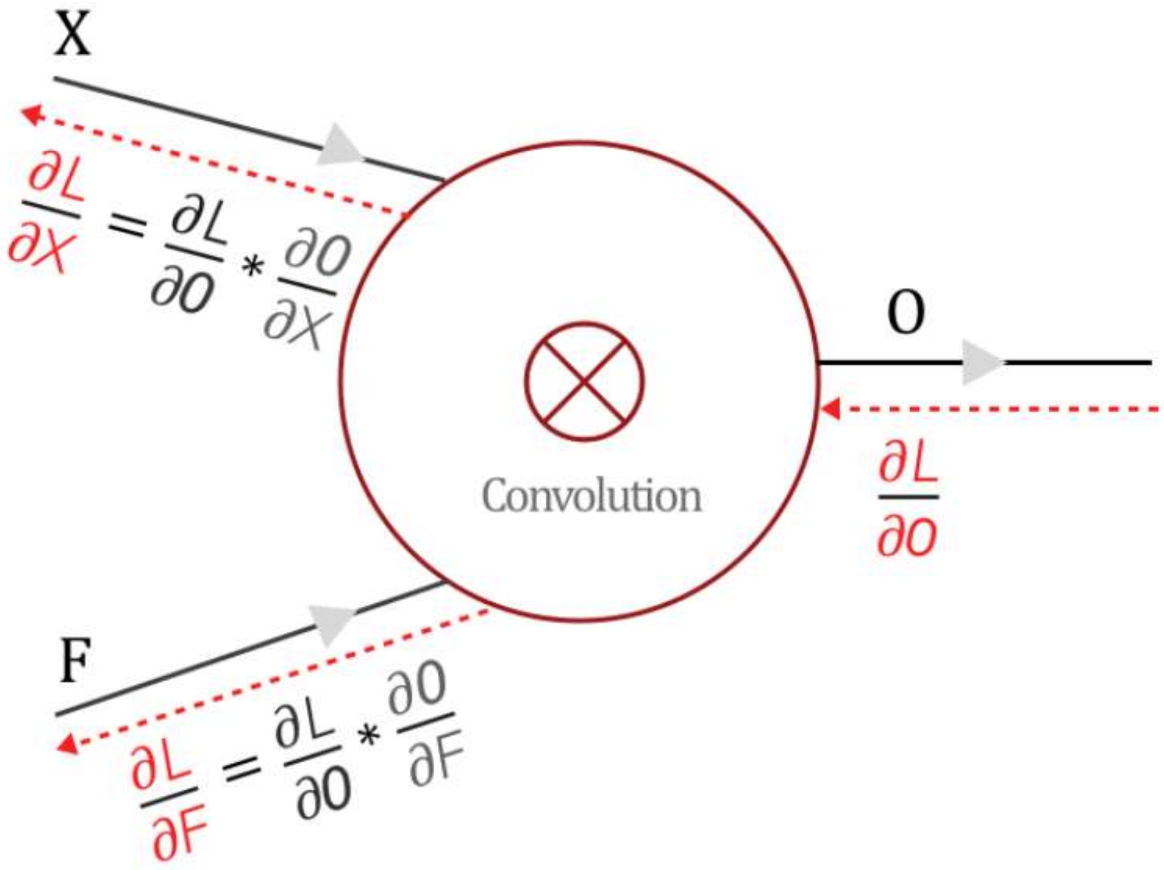
https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

# Forward Prop CNN:




$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

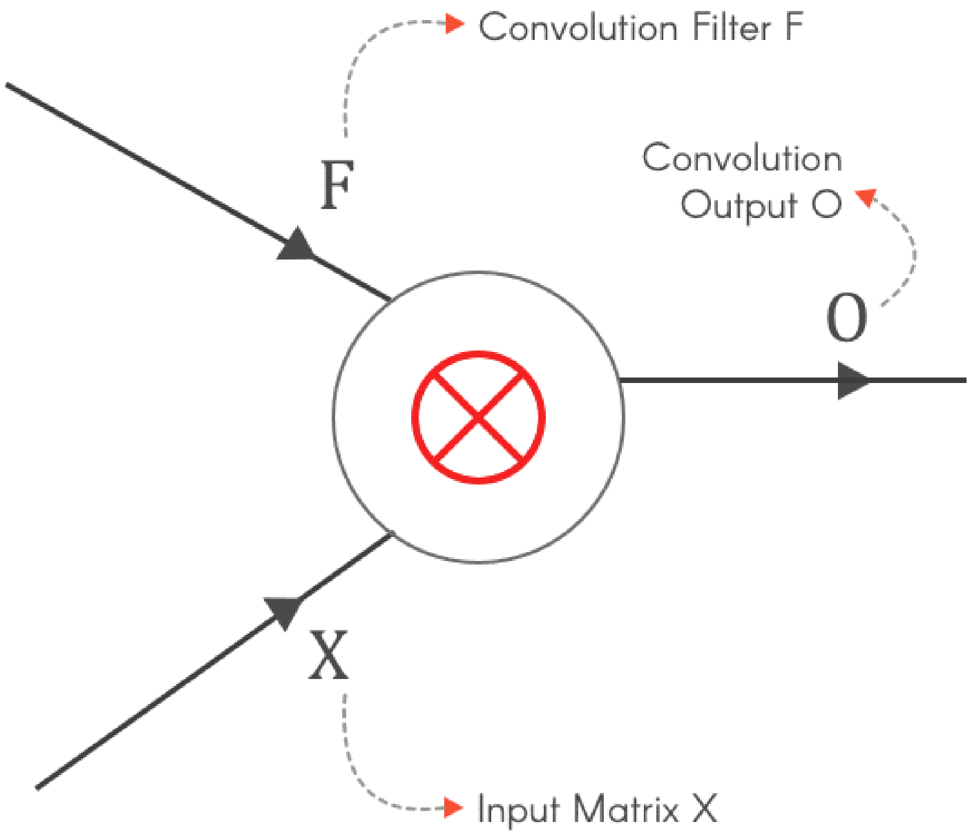
# Backpropagation in CNN




$\frac{\partial O}{\partial X}$ & $\frac{\partial O}{\partial F}$ *are local gradients*

$\frac{\partial L}{\partial z}$ *is the loss from the previous layer which has to be backpropagated to other layers*

# Backpropagation in CNN




So let's find the gradients for X and F — **∂L/∂X** and **∂L/∂F**

# Finding ∂L/∂F:

This has two steps as we have done earlier.

- Find the local gradient ∂O/∂F
- Find ∂L/∂F using chain rule

*Local Gradients* → (A)

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

*Finding derivatives with respect to $F_{11}$, $F_{12}$, $F_{21}$ and $F_{22}$*

$$\frac{\partial O_{11}}{\partial F_{11}} = X_{11} \quad \frac{\partial O_{11}}{\partial F_{12}} = X_{12} \quad \frac{\partial O_{11}}{\partial F_{21}} = X_{21} \quad \frac{\partial O_{11}}{\partial F_{22}} = X_{22}$$

*Similarly, we can find the local gradients for $O_{12}$, $O_{21}$ and $O_{22}$*

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial O} * \frac{\partial O}{\partial F}$$

Gradient to update Filter F

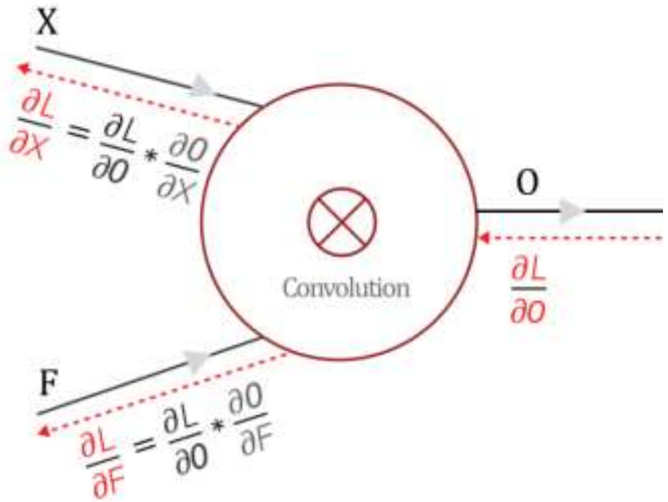
Loss Gradient from previous layer

Local Gradients

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \quad \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \quad \begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix}$$

# Finding ∂L/∂F:

$$\frac{\partial L}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial F_i}$$




$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \quad \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \quad \begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix}$$

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{11}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{11}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{11}}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{12}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{12}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{12}}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{21}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{21}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{21}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{21}}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{22}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{22}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{22}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{22}}$$

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

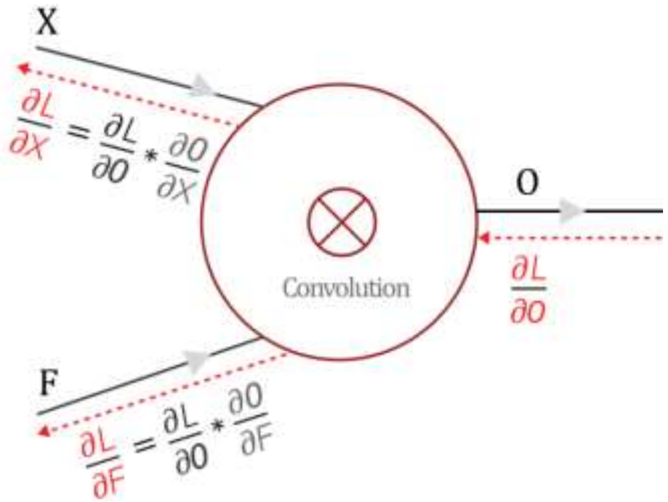
$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

# Finding ∂L/∂F:

$$
\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}
\qquad
\begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}
\qquad
\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix}
$$




$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

$$
\begin{bmatrix} \frac{\partial L}{\partial F_{11}} & \frac{\partial L}{\partial F_{12}} \\ \frac{\partial L}{\partial F_{21}} & \frac{\partial L}{\partial F_{22}} \end{bmatrix}
= \text{Convolution}\left(
\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix},
\underbrace{\begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix}}_{\frac{\partial L}{\partial O}}
\right)
$$

# Finding ∂L/∂X:

| | |
|---|---|
| $F_{11}$ | $F_{12}$ |
| $F_{21}$ | $F_{22}$ |

$$\frac{\partial L}{\partial X_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial X_i}$$

| | | |
|---|---|---|
| $X_{11}$ | $X_{12}$ | $X_{13}$ |
| $X_{21}$ | $X_{22}$ | $X_{23}$ |
| $X_{31}$ | $X_{32}$ | $X_{33}$ |

| | |
|---|---|
| $O_{11}$ | $O_{12}$ |
| $O_{21}$ | $O_{22}$ |

*Local Gradients:* → B

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

*Differentiating with respect to* $X_{11}, X_{12}, X_{21}$ *and* $X_{22}$

$$\frac{\partial O_{11}}{\partial X_{11}} = F_{11} \quad \frac{\partial O_{11}}{\partial X_{12}} = F_{12} \quad \frac{\partial O_{11}}{\partial X_{21}} = F_{21} \quad \frac{\partial O_{11}}{\partial X_{22}} = F_{22}$$

*Similarly, we can find local gradients for* $O_{12}, O_{21}$ *and* $O_{22}$

$$\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial O_{11}} * F_{11}$$

$$\frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial O_{11}} * F_{12} + \frac{\partial L}{\partial O_{12}} * F_{11}$$

$$\frac{\partial L}{\partial X_{13}} = \frac{\partial L}{\partial O_{12}} * F_{12}$$

$$\frac{\partial L}{\partial X_{21}} = \frac{\partial L}{\partial O_{11}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{11}$$

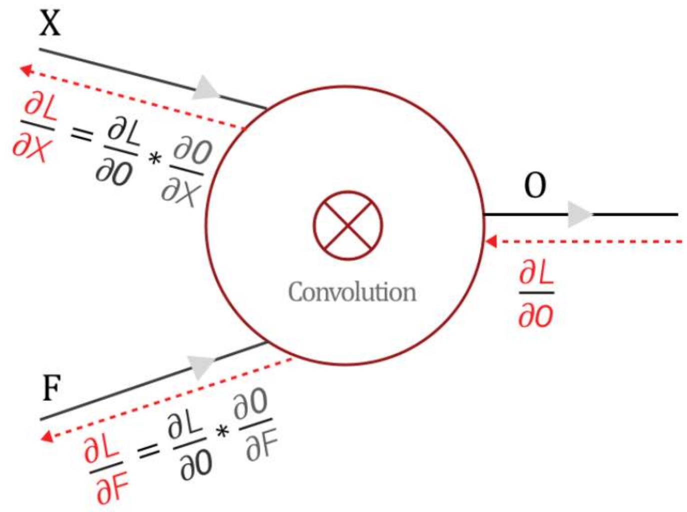
$$\frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial O_{11}} * F_{22} + \frac{\partial L}{\partial O_{12}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{12} + \frac{\partial L}{\partial O_{22}} * F_{11}$$

$$\frac{\partial L}{\partial X_{23}} = \frac{\partial L}{\partial O_{12}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{12}$$

$$\frac{\partial L}{\partial X_{31}} = \frac{\partial L}{\partial O_{21}} * F_{21}$$

$$\frac{\partial L}{\partial X_{32}} = \frac{\partial L}{\partial O_{21}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{21}$$

$$\frac{\partial L}{\partial X_{33}} = \frac{\partial L}{\partial O_{22}} * F_{22}$$



# Finding ∂L/∂X:

$$\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial O_{11}} * F_{11}$$

$$\frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial O_{11}} * F_{12} + \frac{\partial L}{\partial O_{12}} * F_{11}$$

$$\frac{\partial L}{\partial X_{13}} = \frac{\partial L}{\partial O_{12}} * F_{12}$$

$$\frac{\partial L}{\partial X_{21}} = \frac{\partial L}{\partial O_{11}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{11}$$

$$\frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial O_{11}} * F_{22} + \frac{\partial L}{\partial O_{12}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{12} + \frac{\partial L}{\partial O_{22}} * F_{11}$$

$$\frac{\partial L}{\partial X_{23}} = \frac{\partial L}{\partial O_{12}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{12}$$

$$\frac{\partial L}{\partial X_{31}} = \frac{\partial L}{\partial O_{21}} * F_{21}$$

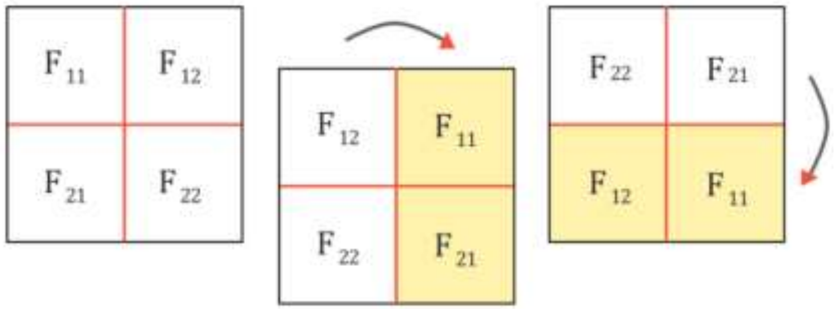
$$\frac{\partial L}{\partial X_{32}} = \frac{\partial L}{\partial O_{21}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{21}$$

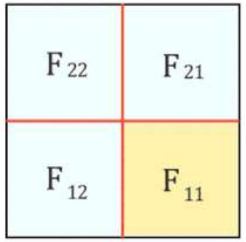
$$\frac{\partial L}{\partial X_{33}} = \frac{\partial L}{\partial O_{22}} * F_{22}$$

| | |
|---|---|
| $F_{11}$ | $F_{12}$ |
| $F_{21}$ | $F_{22}$ |

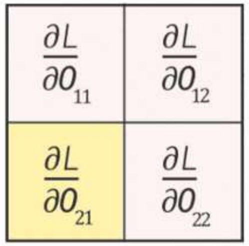
| | | |
|---|---|---|
| $X_{11}$ | $X_{12}$ | $X_{13}$ |
| $X_{21}$ | $X_{22}$ | $X_{23}$ |
| $X_{31}$ | $X_{32}$ | $X_{33}$ |

| | |
|---|---|
| $O_{11}$ | $O_{12}$ |
| $O_{21}$ | $O_{22}$ |

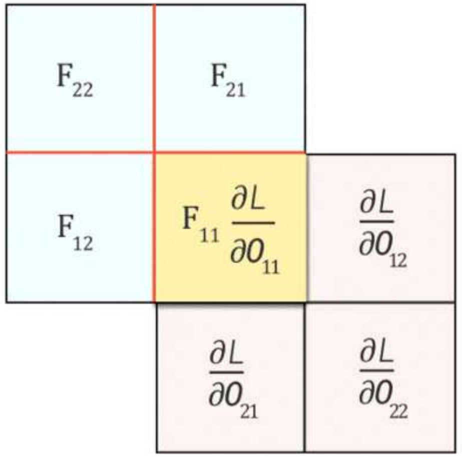


Filter F

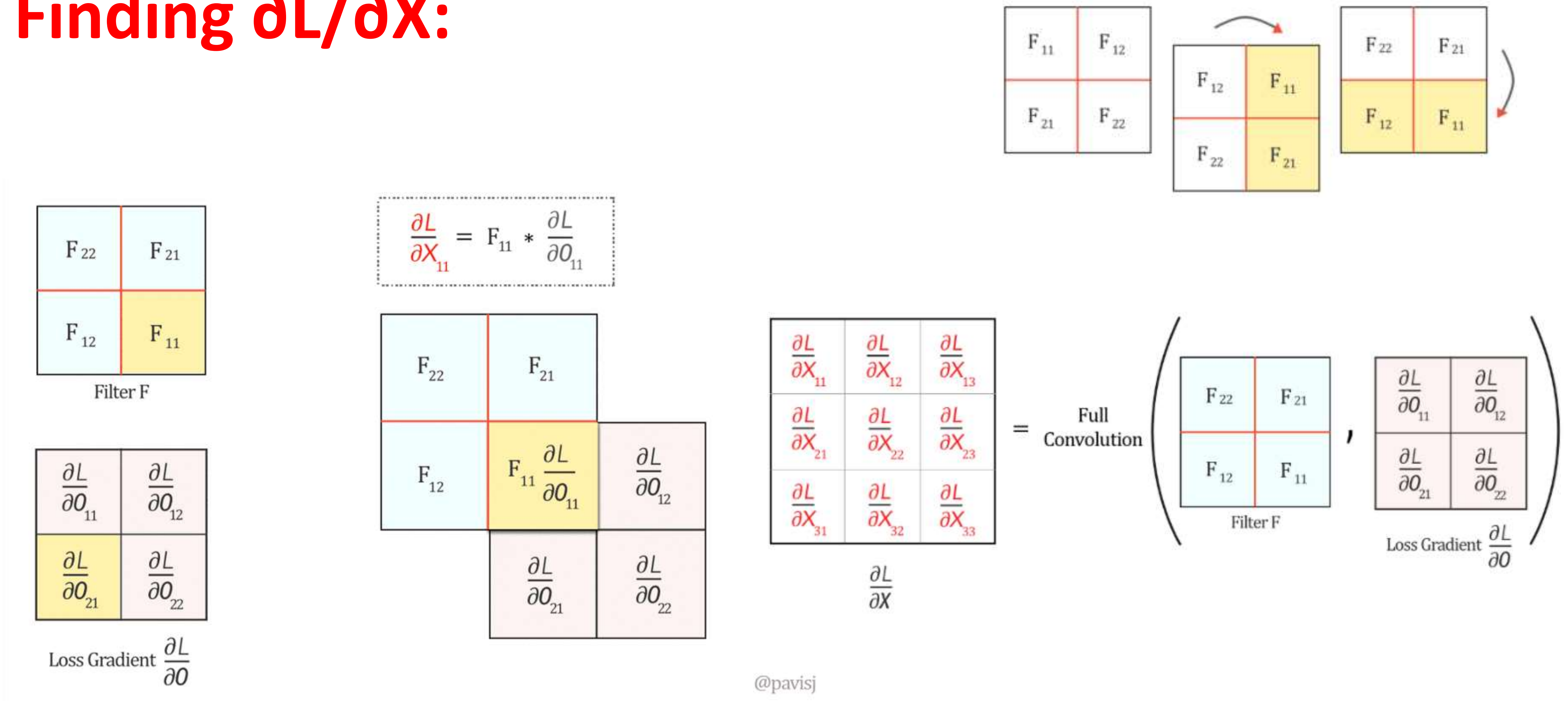

Loss Gradient $\frac{\partial L}{\partial O}$

$$\boxed{\frac{\partial L}{\partial X_{11}} = F_{11} * \frac{\partial L}{\partial O_{11}}}$$




@pavisj

# Finding ∂L/∂X:


Filter F
Loss Gradient $\frac{\partial L}{\partial O}$
$\frac{\partial L}{\partial X_{11}} = F_{11} * \frac{\partial L}{\partial O_{11}}$
$\frac{\partial L}{\partial X}$
= Full Convolution
Filter F
Loss Gradient $\frac{\partial L}{\partial O}$
@pavisj

# Summary: Backpropagation in CNN

**Backpropagation in a Convolutional Layer of a CNN**

**Finding the gradients:**

$$\frac{\partial L}{\partial F} = \text{Convolution}\left(\text{Input } X,\ \text{Loss gradient } \frac{\partial L}{\partial O}\right)$$

$$\frac{\partial L}{\partial X} = \begin{matrix}\text{Full}\\ \text{Convolution}\end{matrix}\left(\begin{matrix}180^{\circ}\text{rotated}\\ \text{Filter } F\end{matrix},\ \begin{matrix}\text{Loss}\\ \text{Gradient}\end{matrix}\ \frac{\partial L}{\partial O}\right)$$

# References


https://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/

https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

Chapter 8 Deep Learning, Goodfellow et. al.