Deep Learning
MSDS 631

Convolutional Neural Networks

Michael Ruddy

Questions?

- From last lecture?

Overview

- What/Why is a Convolution?
- CNN-specific hyperparameters
- Basic CNN history/set-up

Why are images special?

- Images are deceptively hard
- Images are big
- Geometry matters!
    - Pixels near each other interact in different ways to create features than pixels far away
    - This is free data that we lose if we simply consider an image as a data vector

Different relationships

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ .5 & .75 & 1 & \dots & .25 \\ \vdots & \vdots & \vdots & & \vdots \\ .333 & 0 & 1 & \dots & 0 \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data

The Convolution

- Fancy **linear** operation useful for spatial data

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data
- Element-wise product

$$(1 \times 1) + (.5 \times 0) + (0 \times 0) + (.25 \times 2)$$
$$= 1.5$$

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1.5 & & \\ & & \\ & & \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data
- Element-wise product

$$(.5 \times 1) + (1 \times 0) + (.25 \times 0) + (.5 \times 2) = 1.5$$

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1.5 & \boxed{1.5} & \\ & & \\ & & \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data
- Element-wise product

$$(1 \times 1) + (0 \times 0) + (.5 \times 0) + (1 \times 2) = 3$$

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & \boxed{1} & \boxed{0} \\ 0 & .25 & \boxed{.5} & \boxed{1} \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1.5 & 1.5 & \boxed{3} \\ & & \\ & & \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data
- Element-wise product

$$(0 \times 1) + (.25 \times 0) + (1 \times 0) + (.25 \times 2) = .5$$

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1.5 & 1.5 & 3 \\ \boxed{0.5} & & \\ & & \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data
- Element-wise product

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1.5 & 1.5 & 3 \\ 0.5 & ? & ? \\ ? & ? & ? \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data
- Element-wise product

Grayscale Image

Filter (kernel)

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1.5 & 1.5 & 3 \\ 0.5 & .25 & 2.5 \\ 1 & 2.25 & 2 \end{bmatrix}$$

The Convolution

- Fancy **linear** operation useful for spatial data
- Element-wise product

Grayscale Image

Unknown Parameters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} =$$

“2x2 Filter”

Why Convolution?

- Only four parameters!
  - If input is dimension 16 and output is dimension 9, how many for FC?

Grayscale Image

Unknown Parameters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} =$$

"2x2 Filter"

Why Convolution?

- Only four parameters!
- Translational Equivariance
  - If I shift my image, I shift the output!

Grayscale Image

Unknown Parameters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} =$$

"2x2 Filter"

Why Convolution?

- Only four parameters!
- Translational Equivariance
- Weight Sharing (detect same feature translated to different parts of the image)

Grayscale Image

Unknown Parameters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} =$$

"2x2 Filter"

Why Convolution?

Intuition: [Edge Detection]

- Only four parameters!
- Translational Equivariance
- Weight Sharing (detect same feature translated to different parts of the image)

Grayscale Image

Unknown Parameters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = \boxed{}$$

"2x2 Filter"

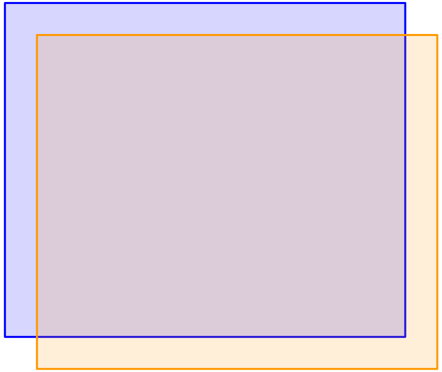
The Convolution

- In a Conv. layer we apply many filter to get many features

Grayscale Image

More Parameters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix} =$$



“2x2 Filter”

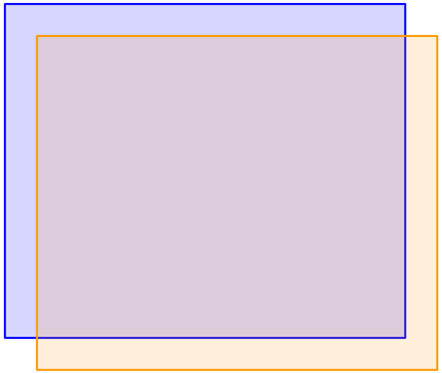
The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”

Grayscale Image

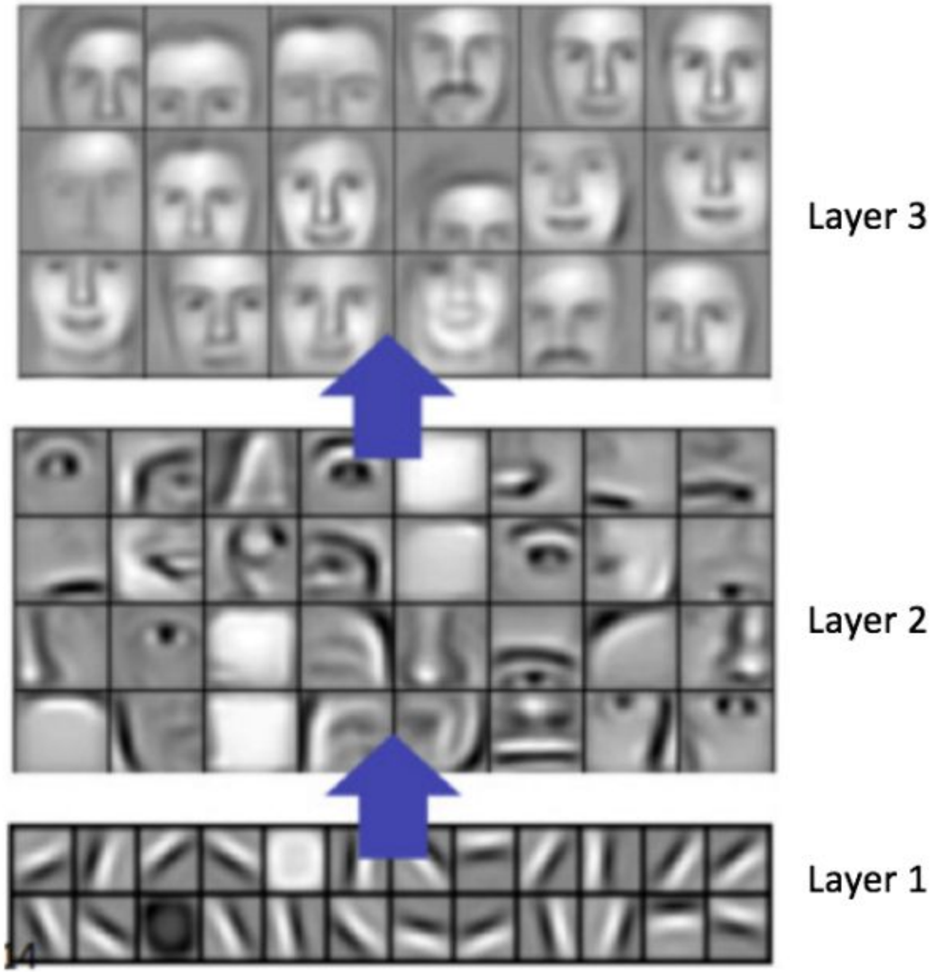
More Parameters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix} =$$



“2x2 Filter”

“2 Channels”

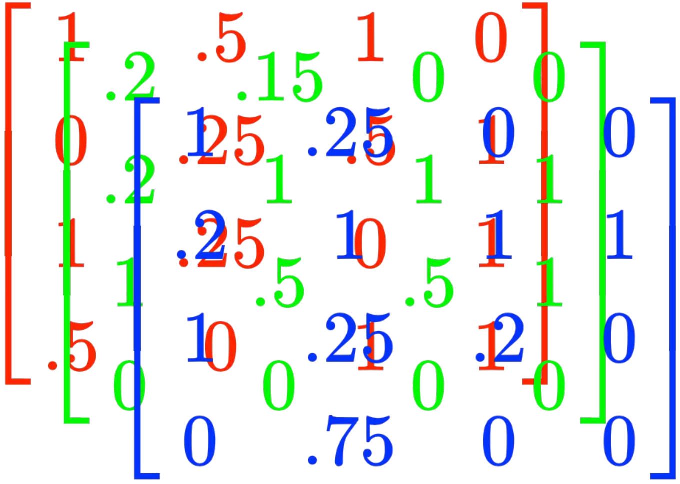


Convolutional Deep Belief Networks for Scalable Unsupervised Laerning of Hierarchical Representations, Lee H., Grosse R., Ranganath R., Ng A.

The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”

RGB Image

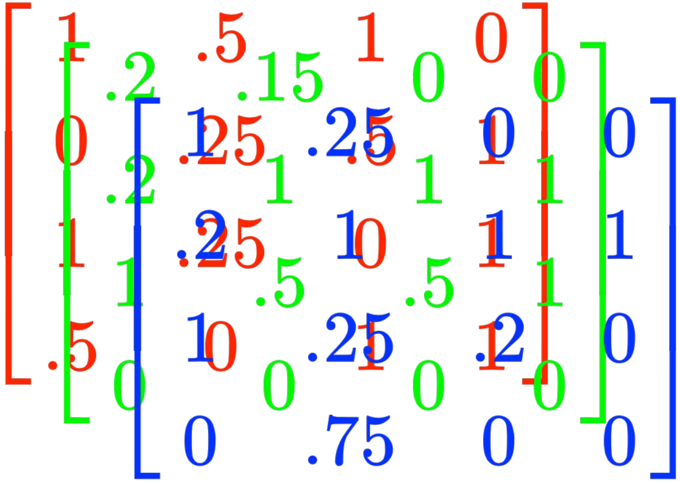


The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N "channels"

RGB Image

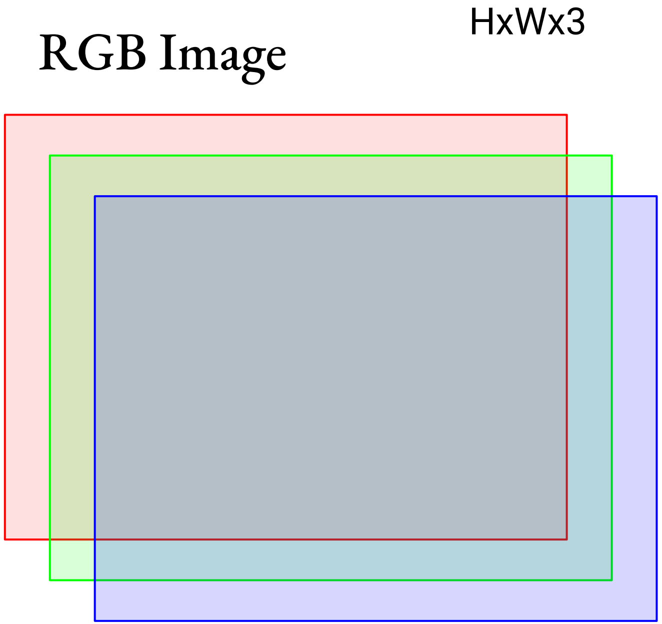
4x4x3



The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”

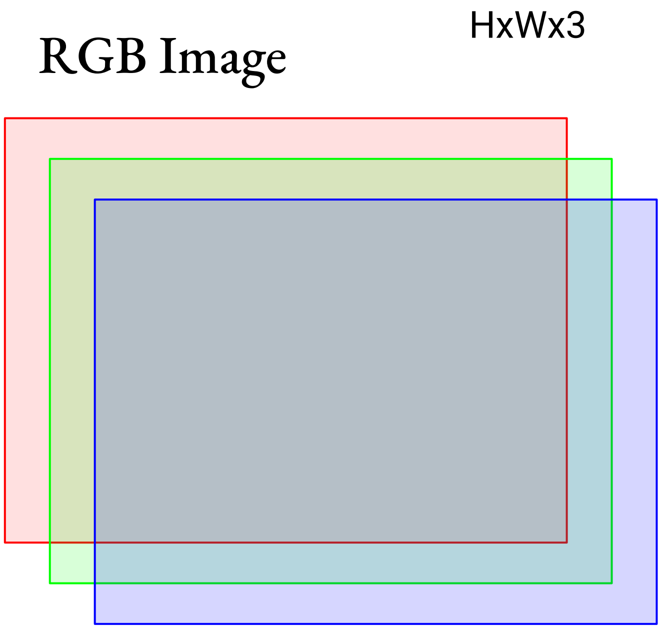
RGB Image



The Convolution

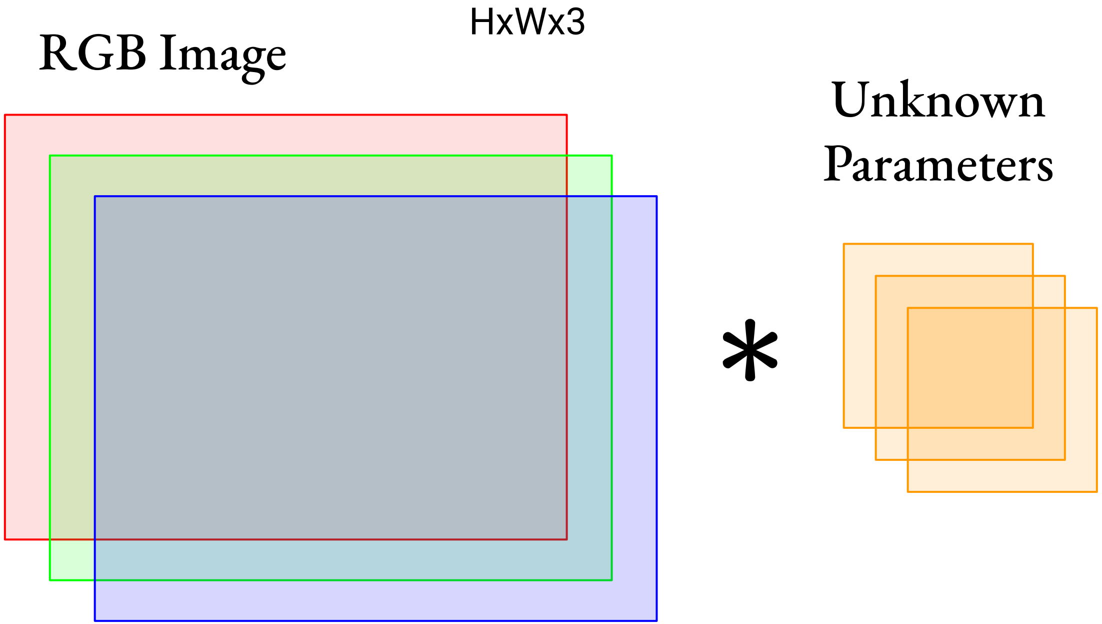
- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”

RGB Image



The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”
- Filter channels must match input channels!!!



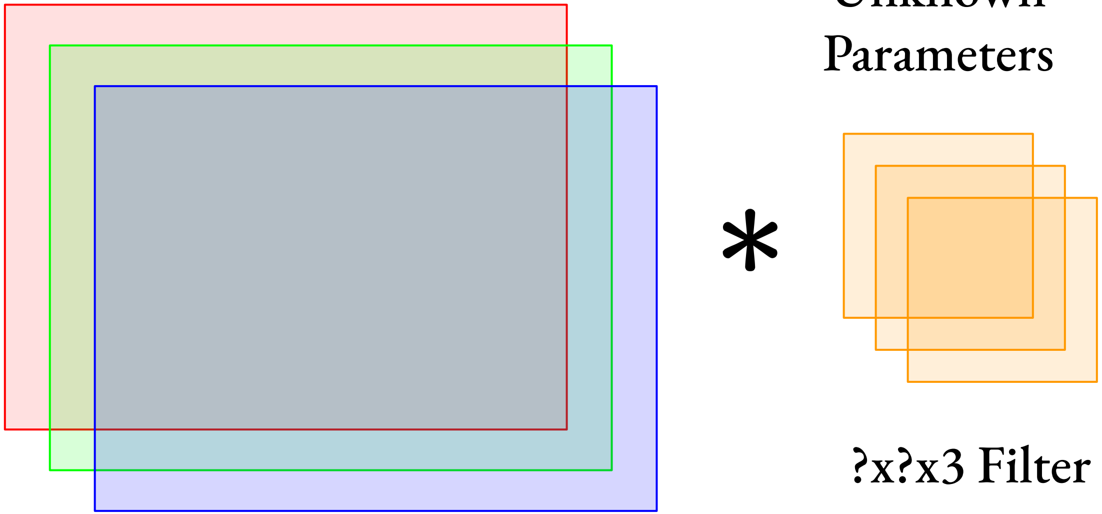
The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”
- Filter channels must match input channels!!!

HxWx3

RGB Image

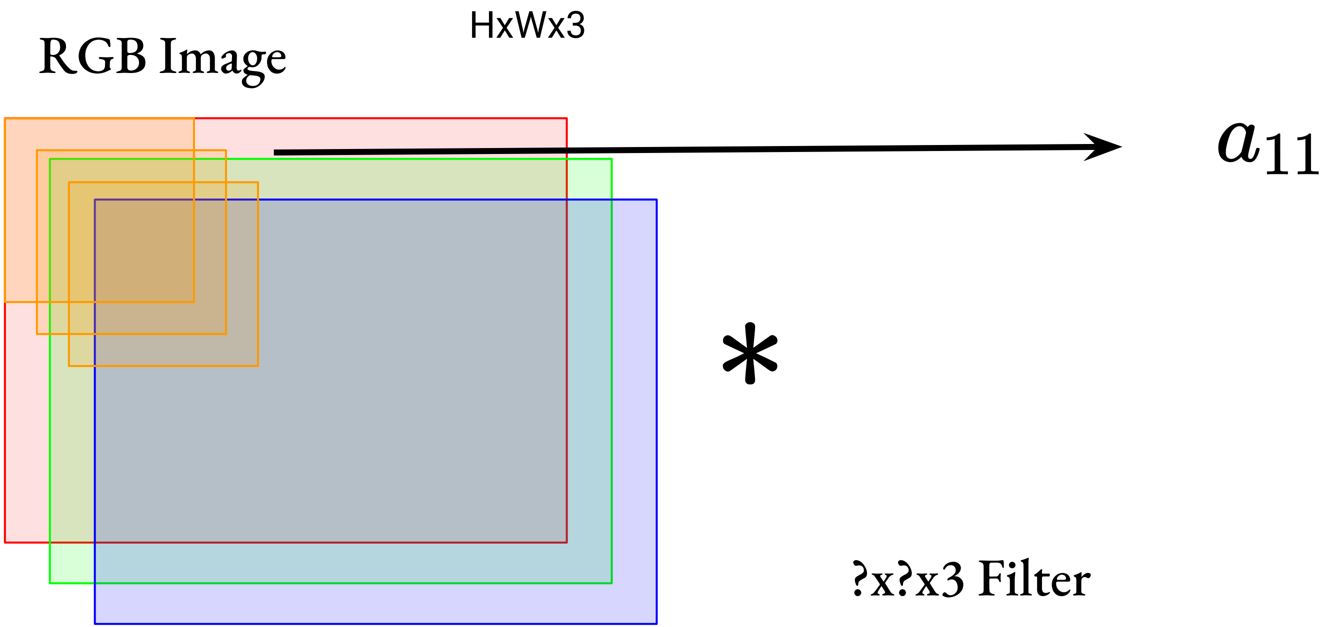
Unknown
Parameters



?x?x3 Filter

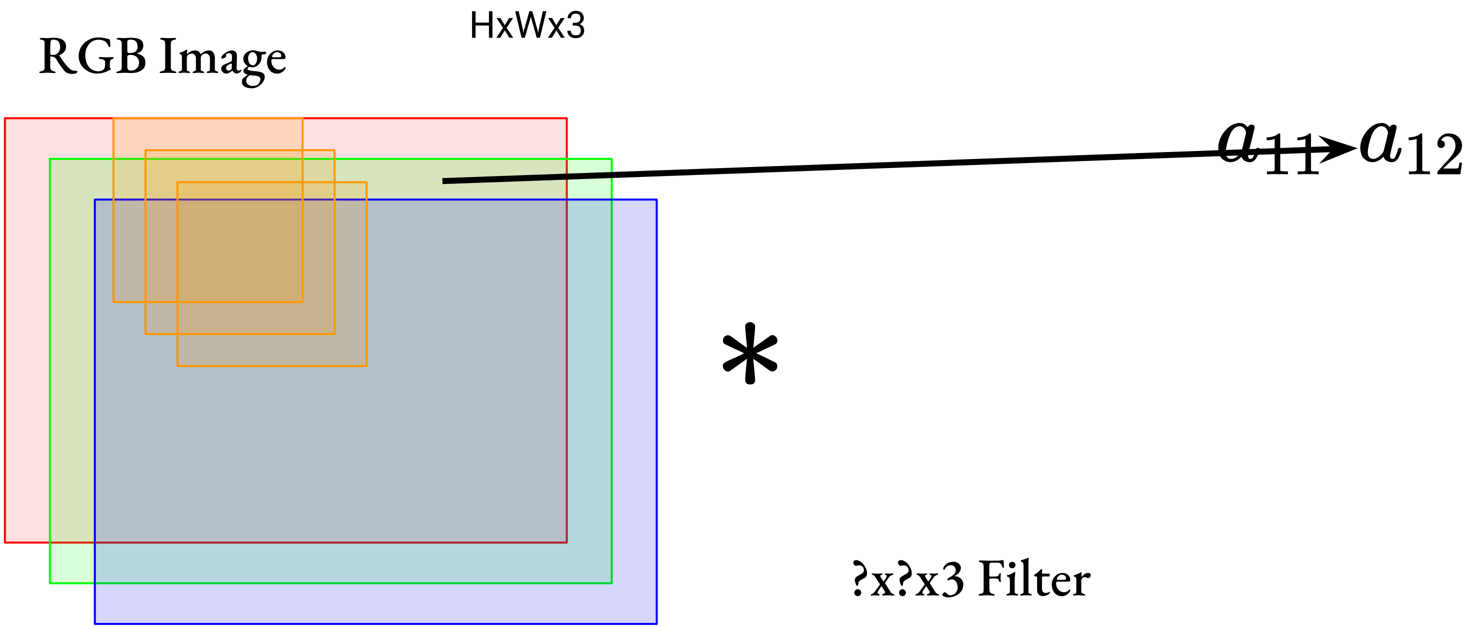
The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”
- Filter channels must match input channels!!!



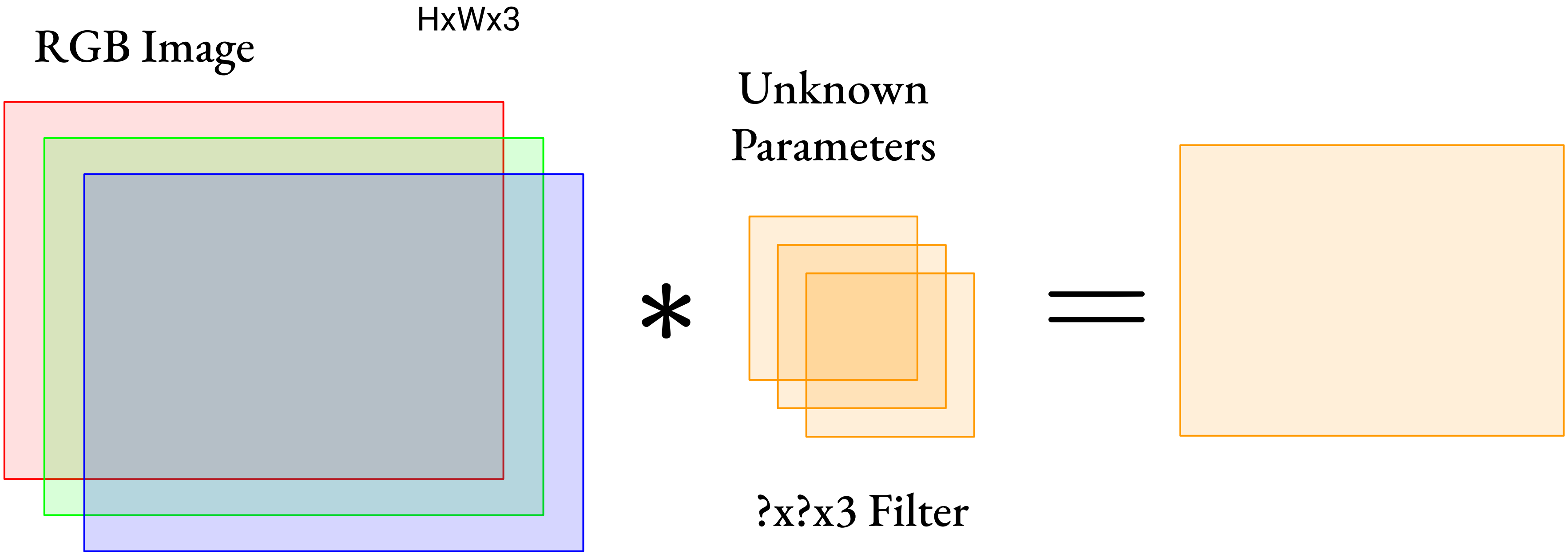
The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”
- Filter channels must match input channels!!!

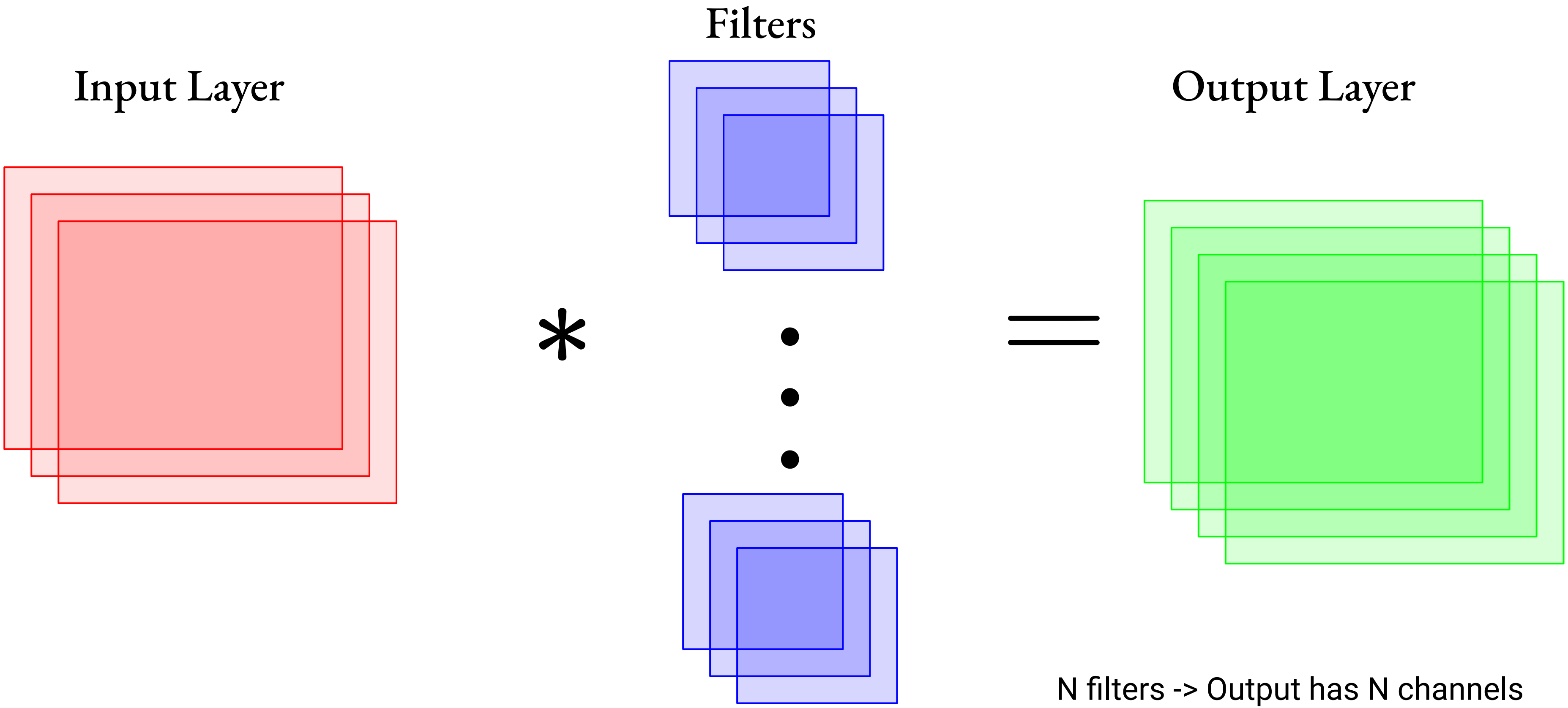


The Convolution

- In a Conv. layer we apply many filter to get many features
- Applying N filters to an image results in an output with N “channels”
- Filter channels must match input channels!!!



The Convolution



N filters -> Output has N channels

Convolution Hyperparameters

- Number of Filters

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
    - "How far it jumps when sliding"

Stride 1

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
    - “How far it jumps when sliding”

Stride 1

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
  - “How far it jumps when sliding”

Stride 1

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
    - “How far it jumps when sliding”

Stride 1

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
    - “How far it jumps when sliding”

Stride 2

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
    - “How far it jumps when sliding”

Stride 2

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
  - “How far it jumps when sliding”

Stride 2

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
  - What is the dimension of the output for Stride 1 vs. Stride 2?

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
- Size of filter

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 5 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 0 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
- Size of filter
  - What is output dimension here if stride = 1?

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 5 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 0 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
- Size of filter
- Problem: size of output keep shrinking!
    - Only a few convolutional layers before the resulting 2D dimensions are very small

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
- Size of filter
- Problem: size of output keep shrinking!
  - Only a few convolutional layers before the resulting 2D dimensions are very small
- Solution: Zero padding

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & .5 & 1 & 0 & 0 \\ 0 & 0 & .25 & .5 & 1 & 0 \\ 0 & 1 & .25 & 0 & 1 & 0 \\ 0 & .5 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
- Size of filter
- Padding

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & .5 & 1 & 0 & 0 \\ 0 & 0 & .25 & .5 & 1 & 0 \\ 0 & 1 & .25 & 0 & 1 & 0 \\ 0 & .5 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Number of Filters
- Stride of the filter
- Size of filter
- Padding

Padding by one

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & .5 & 1 & 0 & 0 \\ 0 & 0 & .25 & .5 & 1 & 0 \\ 0 & 1 & .25 & 0 & 1 & 0 \\ 0 & .5 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Convolution Hyperparameters

- Common choices for a Conv-Layer:
  - Stride = 1
  - Odd Filter Size (3x3, 5x5, etc.)
  - “Same” padding [1,2]

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & .5 & 1 & 0 & 0 \\ 0 & 0 & .25 & .5 & 1 & 0 \\ 0 & 1 & .25 & 0 & 1 & 0 \\ 0 & .5 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & .5 & 2 \end{bmatrix}$$

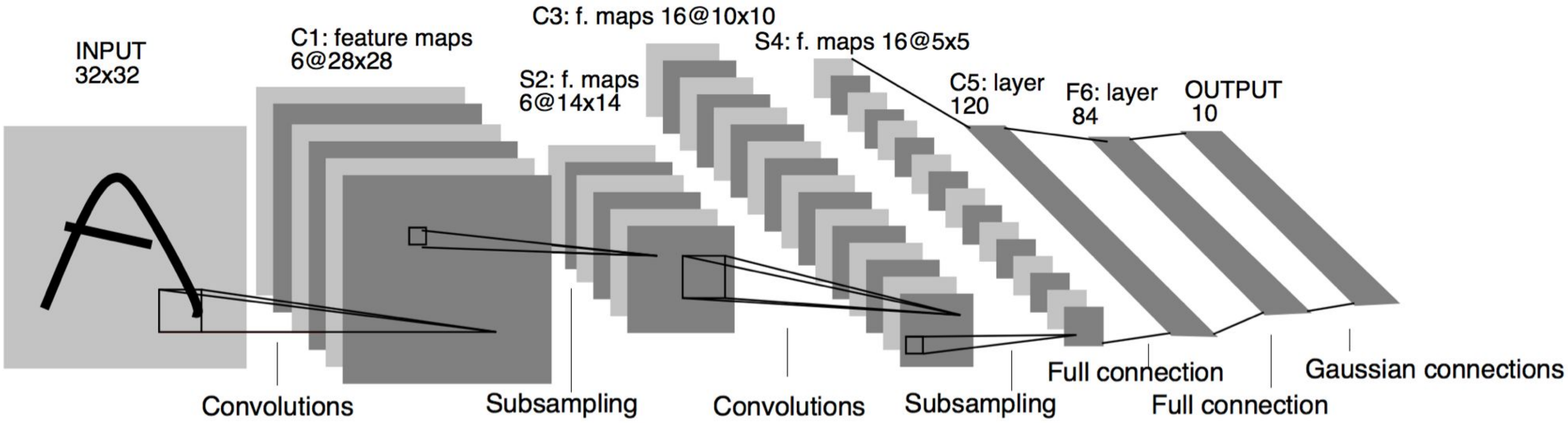
Convolution Hyperparameters

- Common choices for a Conv-Layer:
  - Stride = 1
  - Odd Filter Size (3x3, 5x5, etc.)
  - “Same” padding

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & .5 & 1 & 0 & 0 \\ 0 & 0 & .25 & .5 & 1 & 0 \\ 0 & 1 & .25 & 0 & 1 & 0 \\ 0 & .5 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & .5 & 2 \end{bmatrix}$$

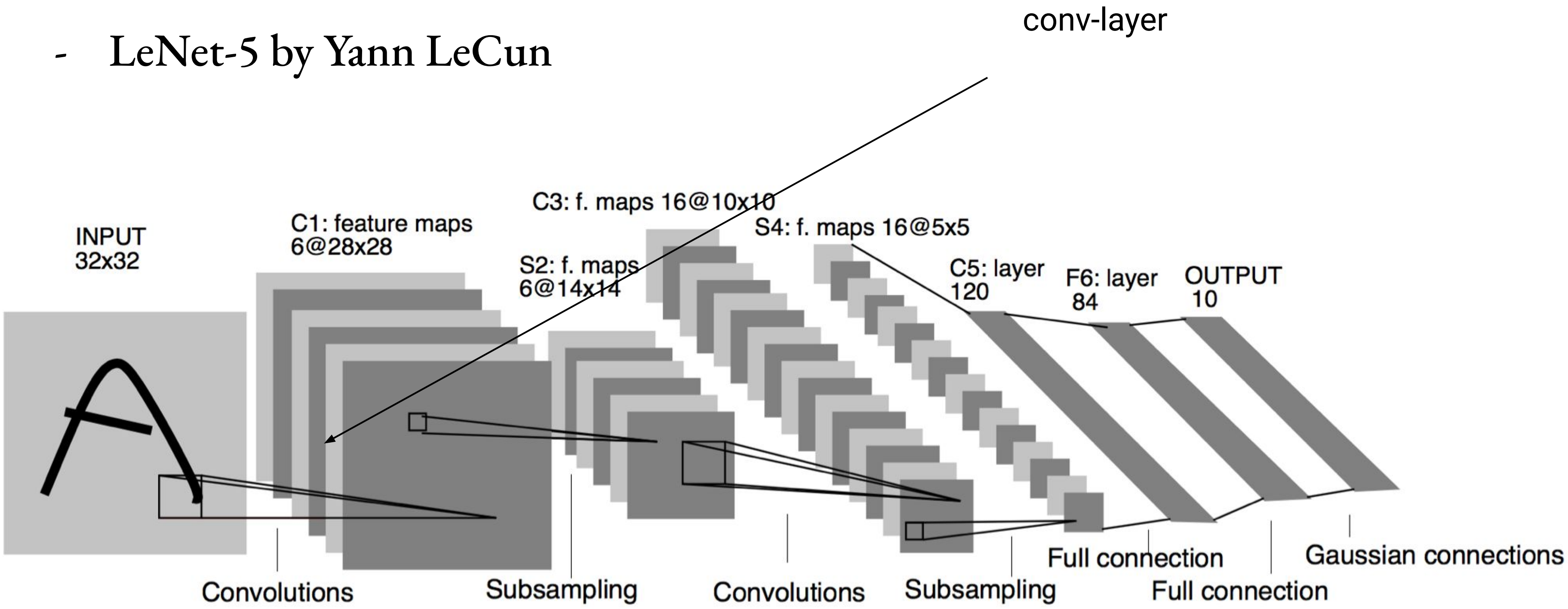
Convolutional Neural Network

- LeNet-5 by Yann LeCun



Convolutional Neural Network

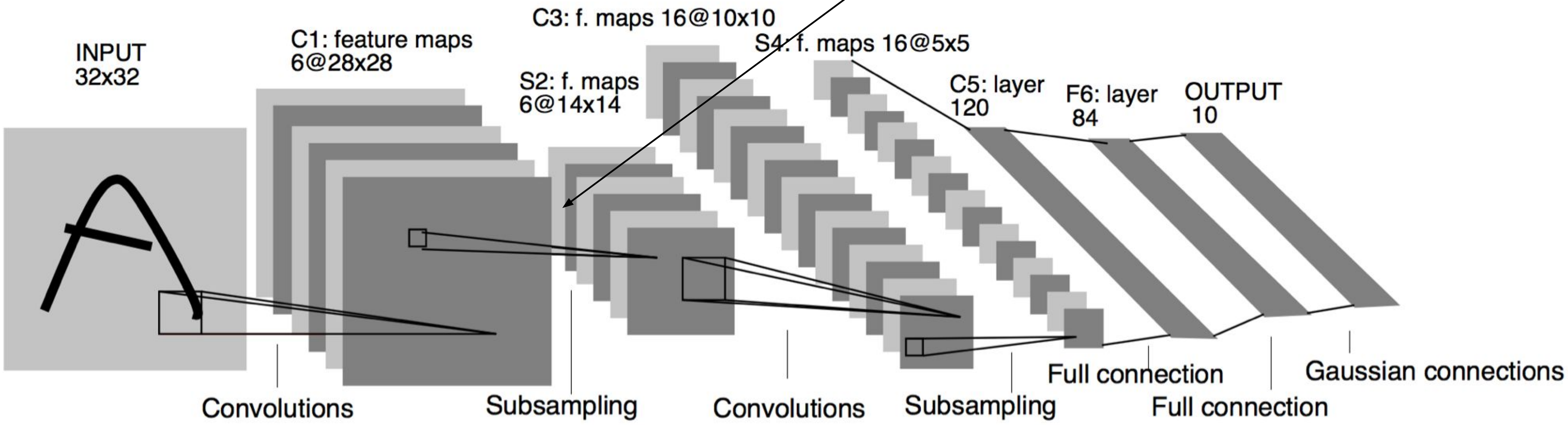
- LeNet-5 by Yann LeCun



Convolutional Neural Network

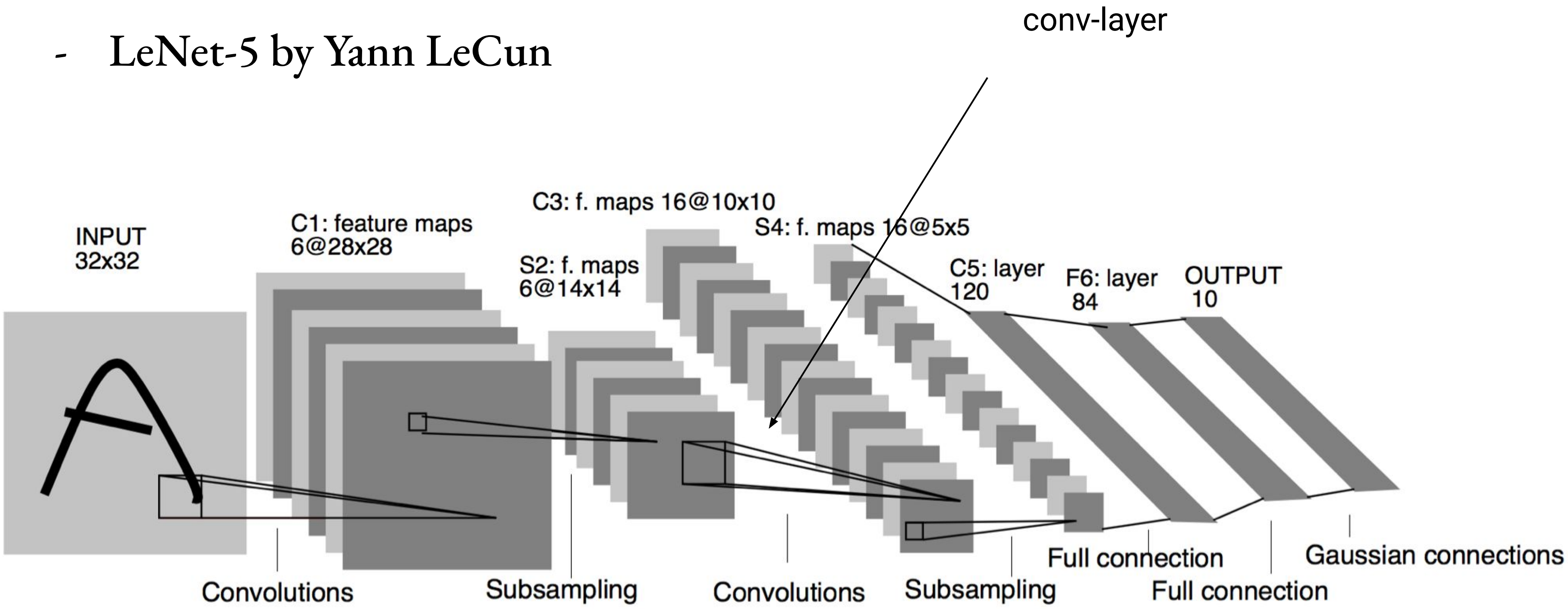
- LeNet-5 by Yann LeCun

down-sampling



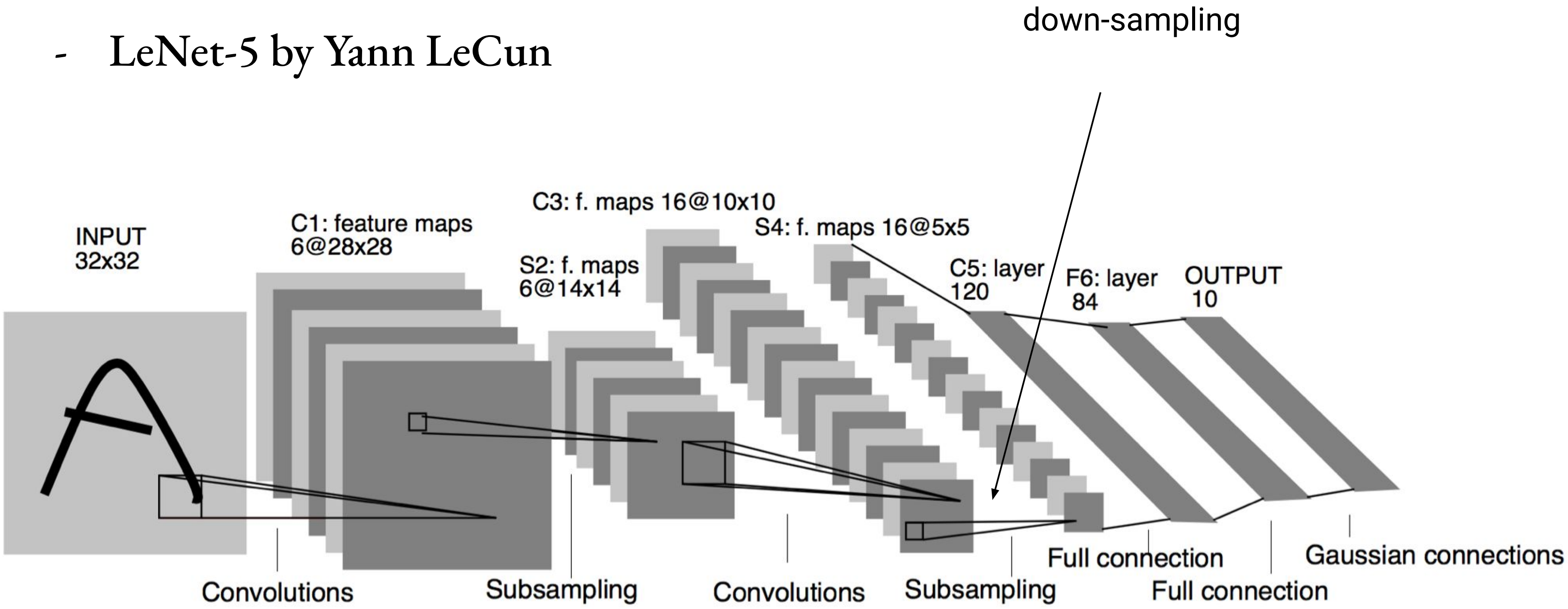
Convolutional Neural Network

- LeNet-5 by Yann LeCun



Convolutional Neural Network

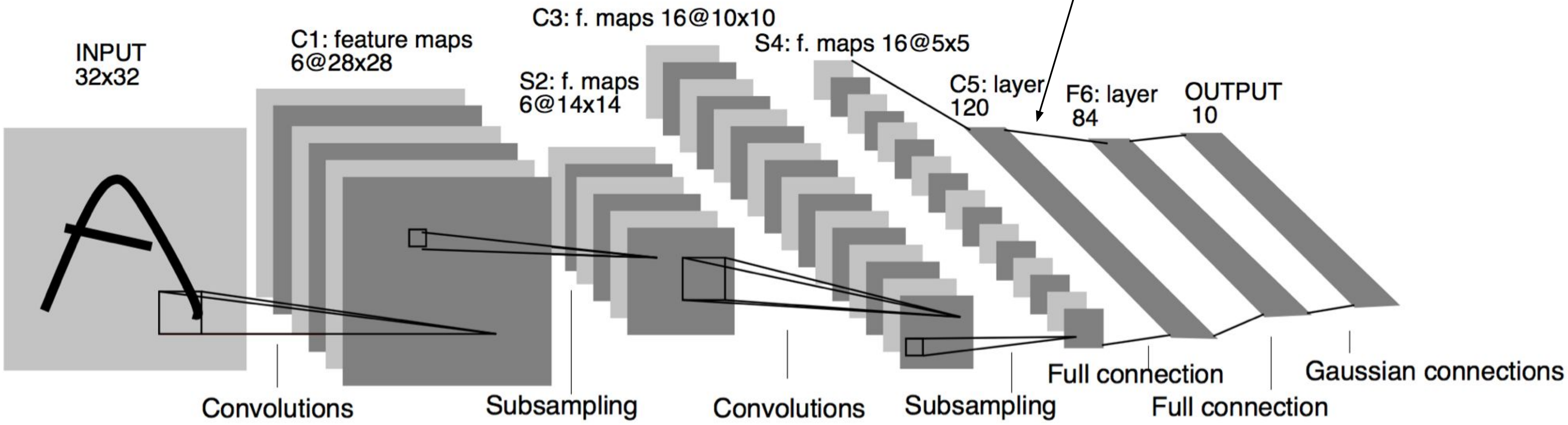
- LeNet-5 by Yann LeCun



Convolutional Neural Network

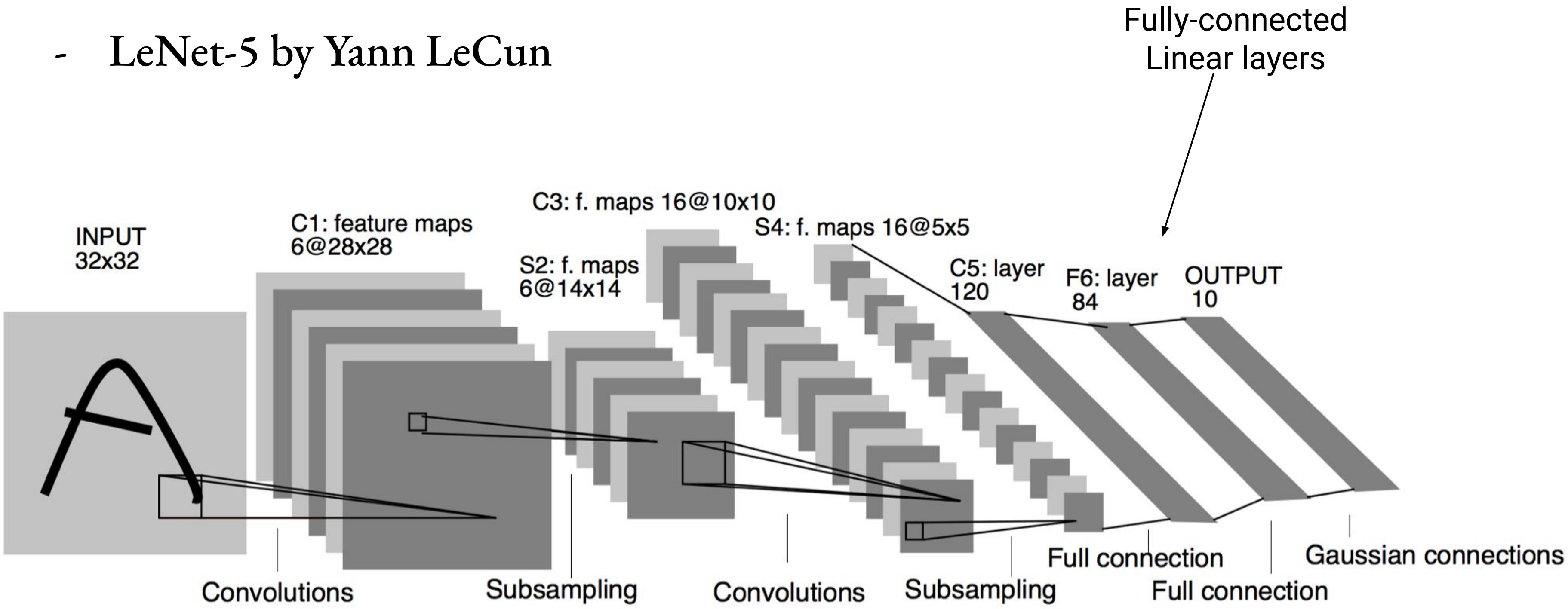
- LeNet-5 by Yann LeCun

“Unroll” into a
one-dimensional vector



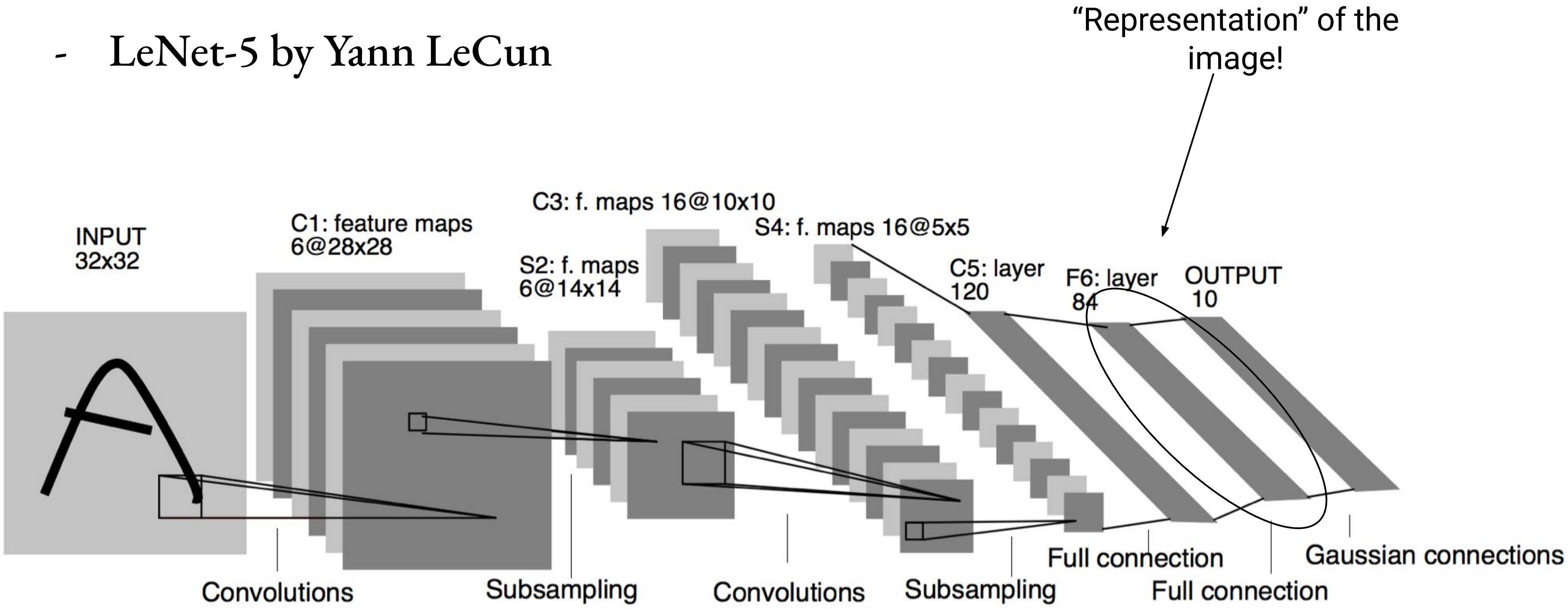
Convolutional Neural Network

- LeNet-5 by Yann LeCun



Convolutional Neural Network

- LeNet-5 by Yann LeCun



Convolutional Neural Network

- AlexNet wins ImageNet Competition in 2012

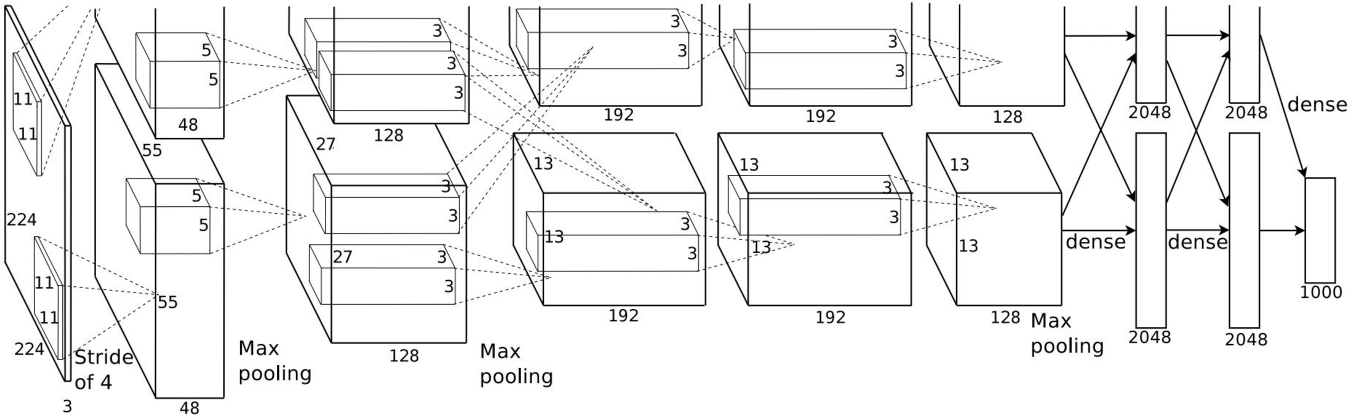


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Convolutional Neural Network

- AlexNet wins ImageNet Competition in 2012
- By 2015 we have CNNs with >100 layers, better than human-level performance

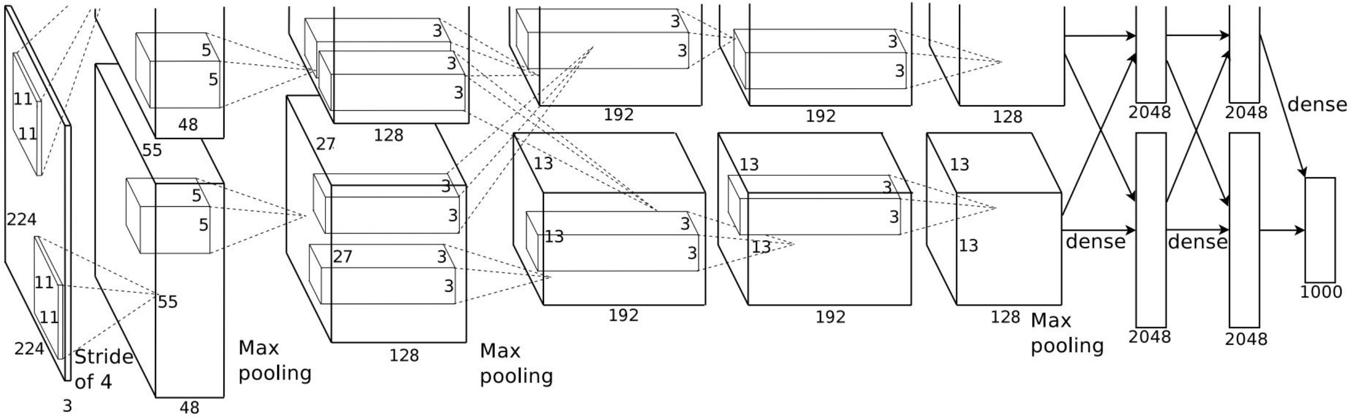


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Downsampling

- Reduce size of output
- Minimal information loss in practice
- Intuition: reduce resolution of the image

Downsampling

- Reduce size of output
- Minimal information loss in practice
- Intuition: reduce resolution of the image
- Max Pooling

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & \\ & \end{bmatrix}$$

Downsampling

- Reduce size of output
- Minimal information loss in practice
- Intuition: reduce resolution of the image
- Max Pooling

- 2x2 filter size
- Stride 2

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ & \end{bmatrix}$$

Downsampling

- Reduce size of output
- Minimal information loss in practice
- Intuition: reduce resolution of the image
- Max Pooling

- 2x2 filter size
- Stride 2

$$\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Downsampling

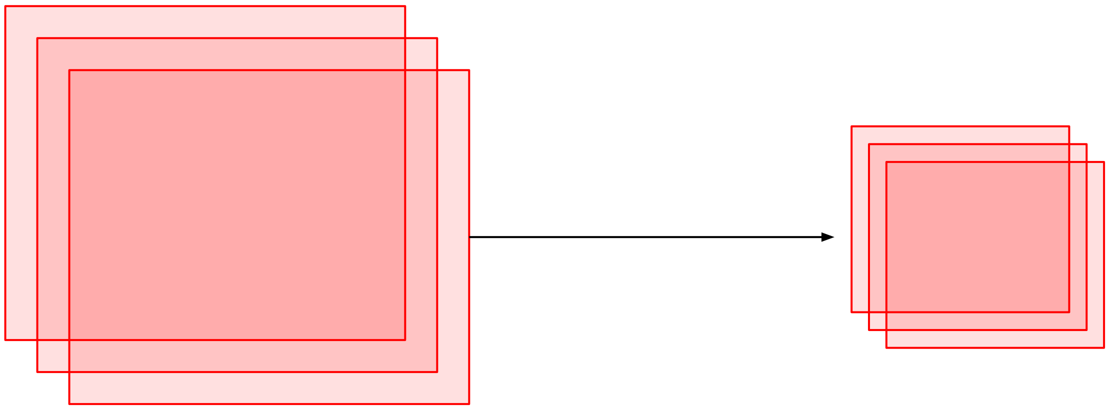
- Reduce size of output
- Minimal information loss in practice
- Intuition: reduce resolution of the image
- Max Pooling
- Average Pooling

- 2x2 filter size
- Stride 2

$$
\begin{bmatrix} 1 & .5 & 1 & 0 \\ 0 & .25 & .5 & 1 \\ 1 & .25 & 0 & 1 \\ .5 & 0 & 1 & 1 \end{bmatrix}
\qquad
\begin{bmatrix} .4375 & .625 \\ .4375 & .75 \end{bmatrix}
$$

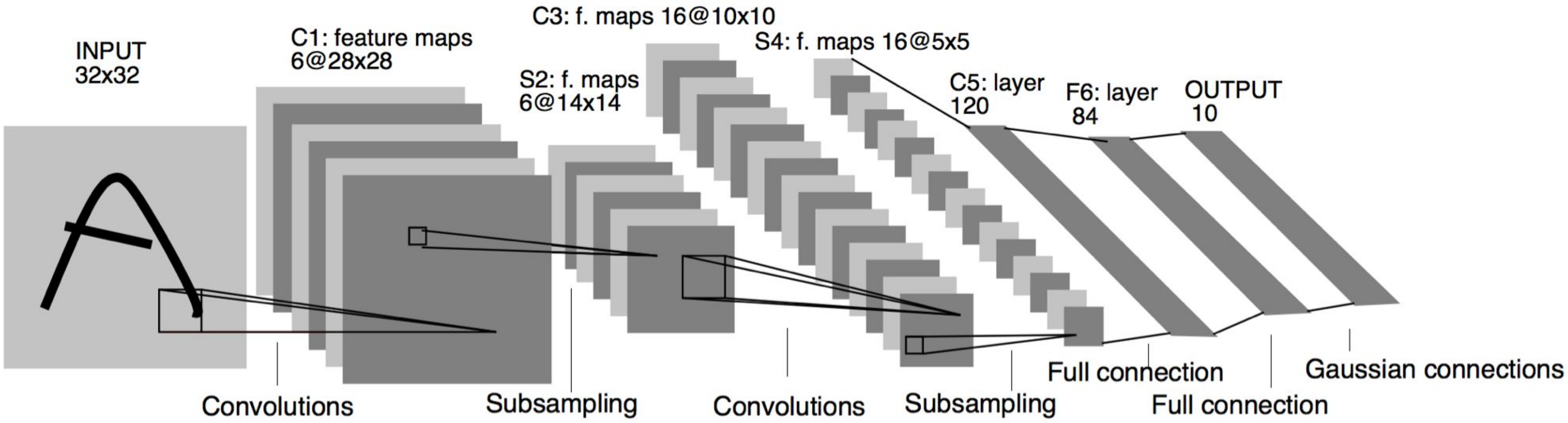
Downsampling

- Done along spatial dimension, preserves channels



Convolutional Neural Network

- LeNet-5 by Yann LeCun

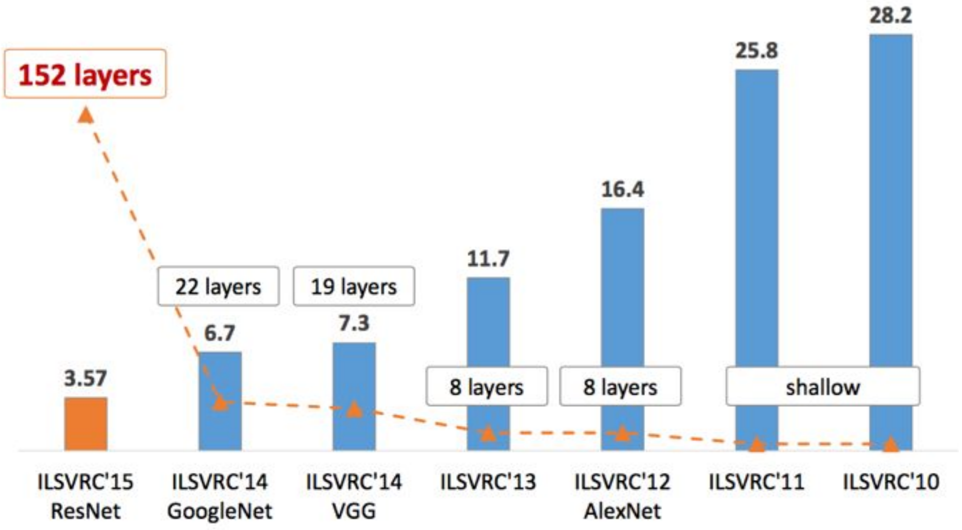


Summary

- Convolution Layers
    - Suited for Spatial Data
    - Less Parameters than FC Layers, Weight sharing
- Common Hyperparameters
    - Number of Filters, Filter Size, Stride, Padding
- Common Sequence
    - Conv -> Activation -> Conv -> Activation -> Downsampling
    - Repeat until unrolled into final FC layers

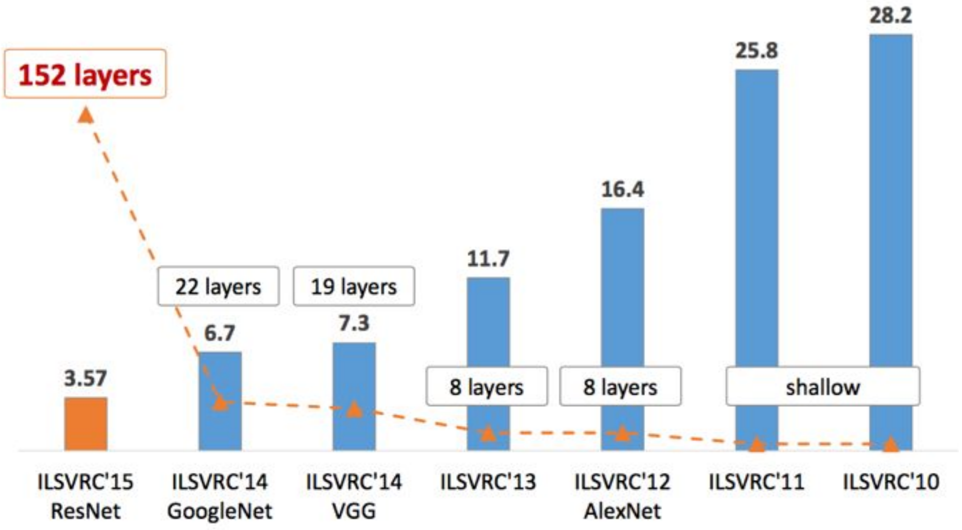
Deeper NNs

- After the success of AlexNet, CNNs got deeper
- Why not just start with as many layers as possible?



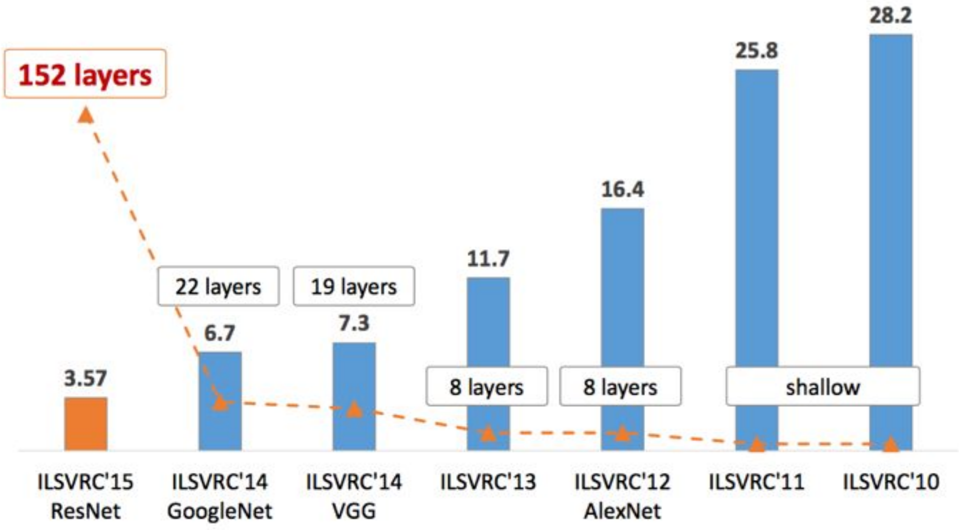
Deeper NNs

- After the success of AlexNet, CNNs got deeper
- Why not just start with as many layers as possible?
  - Computer power, data

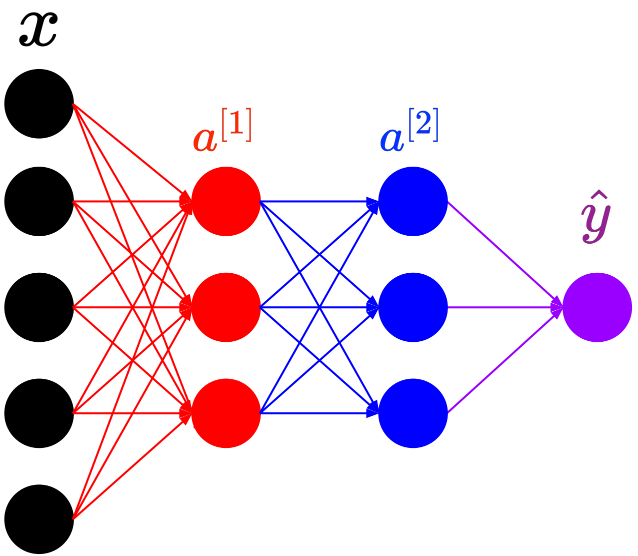


Deeper NNs

- After the success of AlexNet, CNNs got deeper
- Why not just start with as many layers as possible?
  - Computer power, data
  - Problems with training (vanishing/exploding gradients)

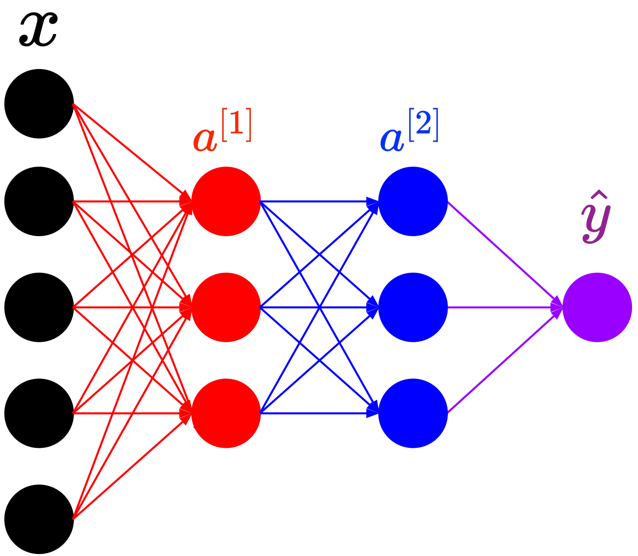


Vanishing/Exploding Gradients



$$a^{[1]} = h(W^{[1]}x + b^{[1]})$$
$$a^{[2]} = h(W^{[2]}a^{[1]} + b^{[2]})$$
$$F(x;\theta) = h(W^{[3]}a^{[2]} + b^{[3]})$$

Vanishing/Exploding Gradients



$$a^{[1]} = h(W^{[1]}x + b^{[1]})$$
$$a^{[2]} = h(W^{[2]}a^{[1]} + b^{[2]})$$
$$F(x;\theta) = h(W^{[3]}a^{[2]} + b^{[3]})$$

$$F = f_1(w_1, f_2(w_2, f_3(w_3)))$$

Vanishing/Exploding Gradients

$$F = f_1(w_1, f_2(w_2, f_3(w_3)))$$

Vanishing/Exploding Gradients

$$F = f_1(w_1, f_2(w_2, f_3(w_3)))$$

$$\frac{\partial F}{\partial w_1} = \frac{\partial f_1}{\partial w_1}$$

Vanishing/Exploding Gradients

$$F = f_1(w_1, f_2(w_2, f_3(w_3)))$$

$$\frac{\partial F}{\partial w_1} = \frac{\partial f_1}{\partial w_1}$$

$$\frac{\partial F}{\partial w_2} = \frac{\partial f_1}{\partial f_2}\frac{\partial f_2}{\partial w_2}$$

Vanishing/Exploding Gradients

$$F = f_1(w_1, f_2(w_2, f_3(w_3)))$$

$$\frac{\partial F}{\partial w_1} = \frac{\partial f_1}{\partial w_1}$$

$$\frac{\partial F}{\partial w_2} = \frac{\partial f_1}{\partial f_2}\frac{\partial f_2}{\partial w_2}$$

$$\frac{\partial F}{\partial w_3} = \frac{\partial f_1}{\partial f_2}\frac{\partial f_2}{\partial f_3}\frac{\partial f_3}{\partial w_3}$$

Vanishing/Exploding Gradients

$$F = f_1(w_1, f_2(w_2, f_3(w_3)))$$

$$(.1)^3 = .001$$

$$\frac{\partial F}{\partial w_1} = \frac{\partial f_1}{\partial w_1}$$

$$\frac{\partial F}{\partial w_2} = \frac{\partial f_1}{\partial f_2} \frac{\partial f_2}{\partial w_2}$$

$$\frac{\partial F}{\partial w_3} = \frac{\partial f_1}{\partial f_2} \frac{\partial f_2}{\partial f_3} \frac{\partial f_3}{\partial w_3}$$

Vanishing/Exploding Gradients

$$F = f_1(w_1, f_2(w_2, f_3(w_3)))$$

$$(2)^3 = 8$$

$$\frac{\partial F}{\partial w_1} = \frac{\partial f_1}{\partial w_1}$$

$$\frac{\partial F}{\partial w_2} = \frac{\partial f_1}{\partial f_2} \frac{\partial f_2}{\partial w_2}$$

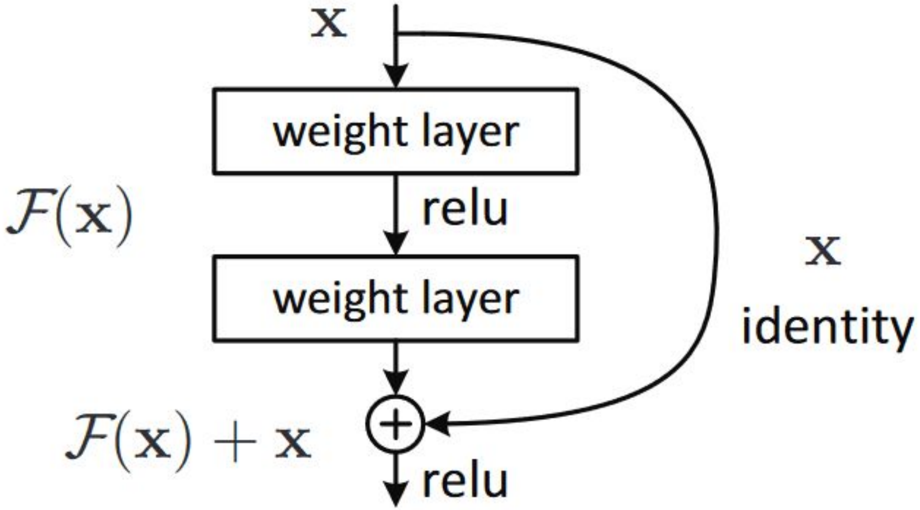
$$\frac{\partial F}{\partial w_3} = \frac{\partial f_1}{\partial f_2} \frac{\partial f_2}{\partial f_3} \frac{\partial f_3}{\partial w_3}$$

Deeper NNs

- Early parameters can either get stuck, or become unstable during training

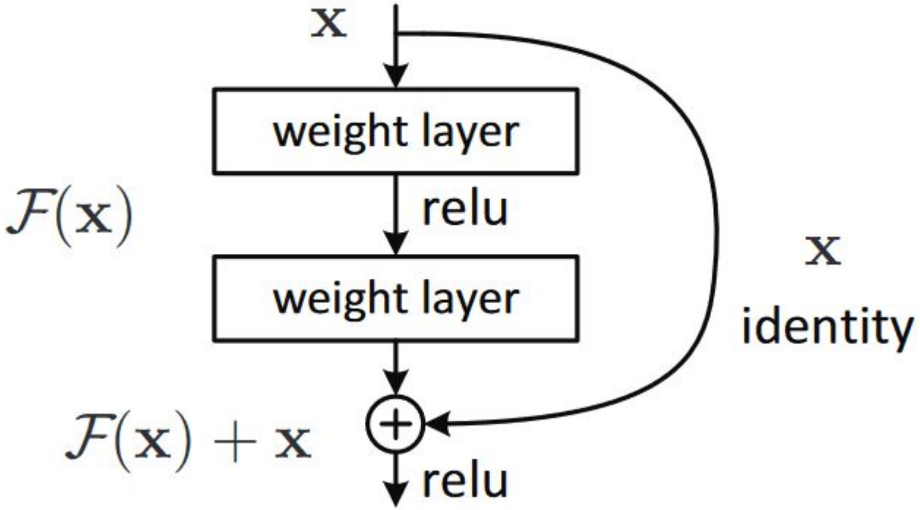
Deeper NNs

- Early parameters can either get stuck, or become unstable during training
- Skip Connection



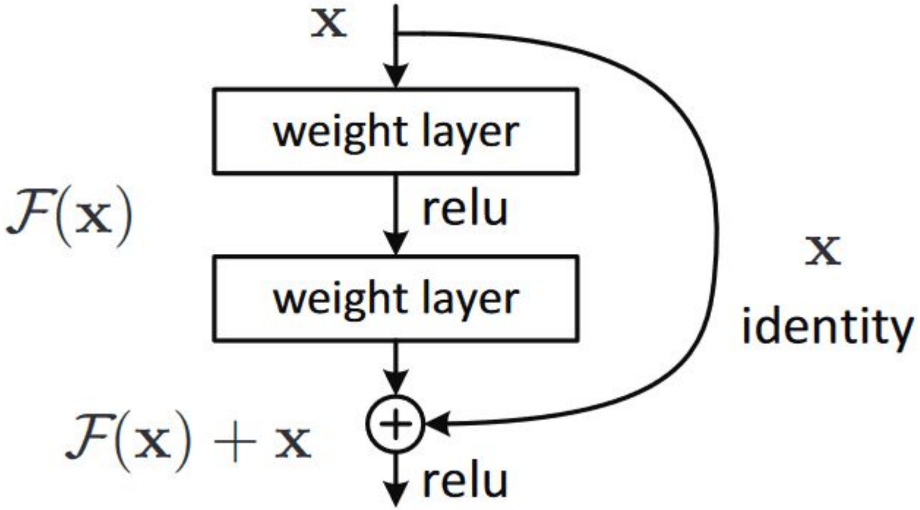
Deeper NNs

- Early parameters can either get stuck, or become unstable during training
- Skip Connection
  - Gradient of earlier parameters depends more directly on output

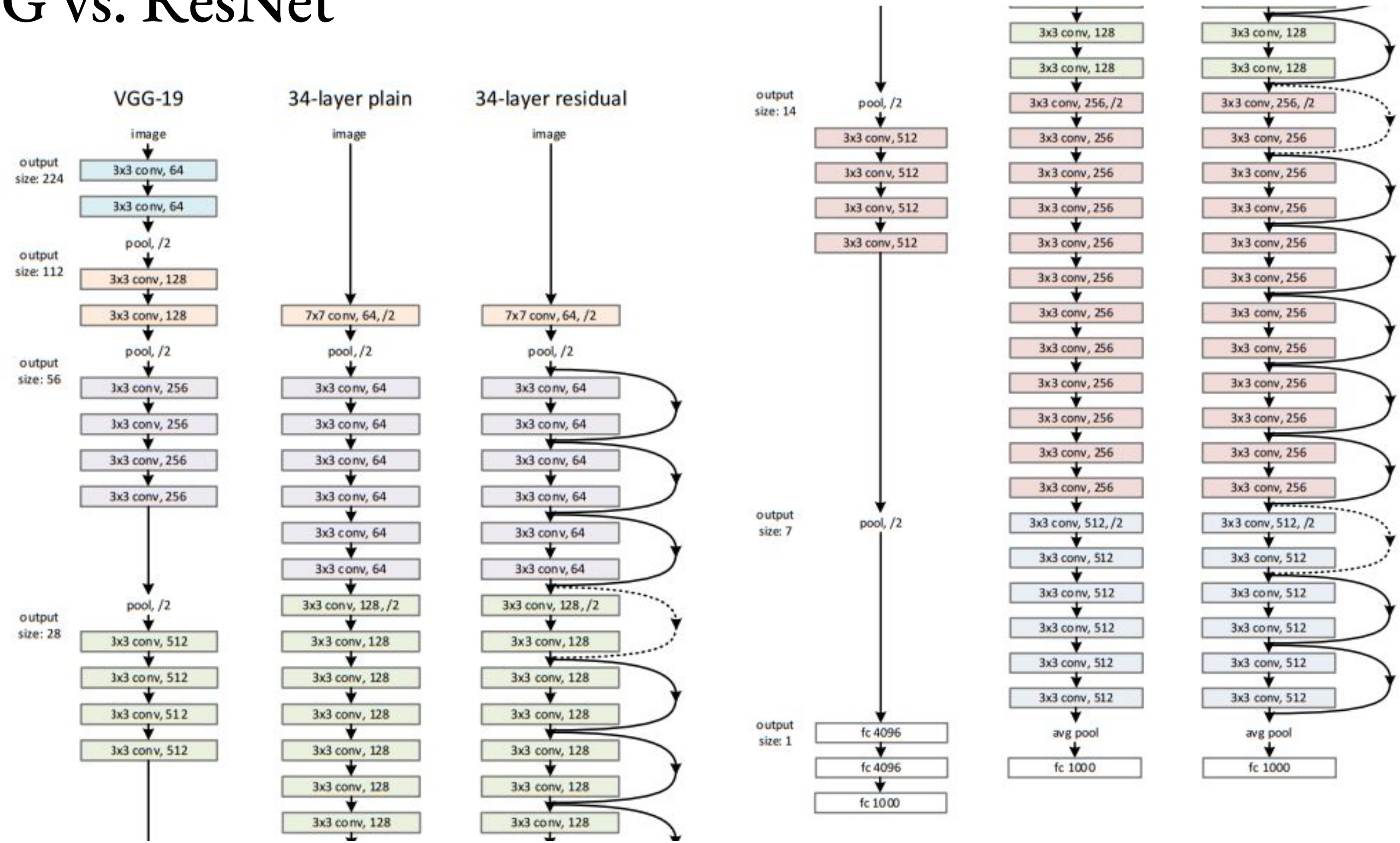


Deeper NNs

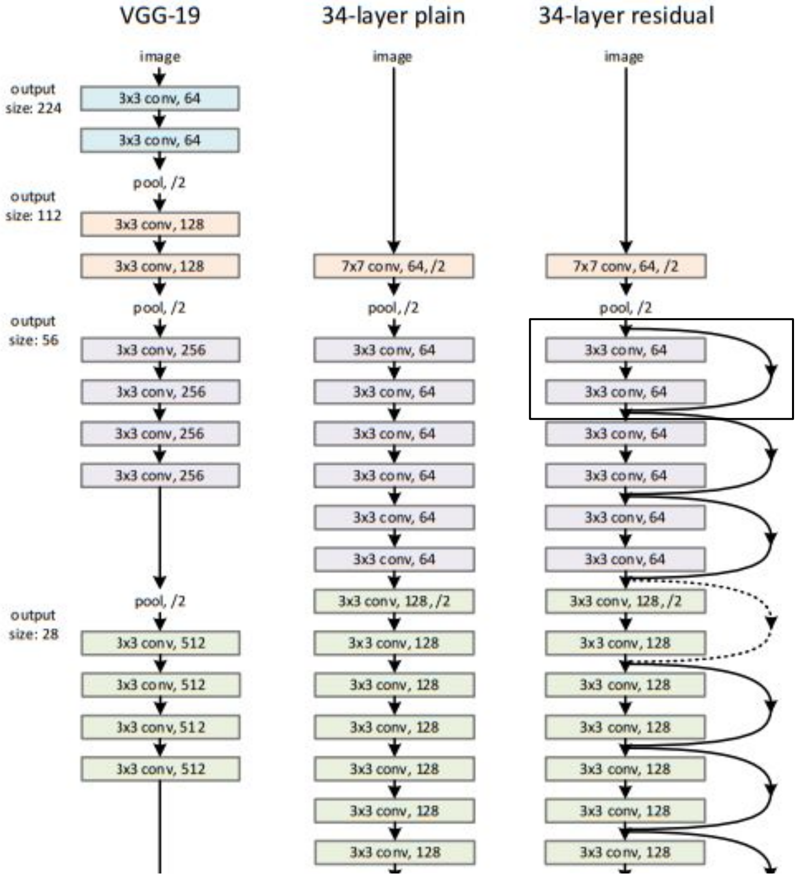
- Early parameters can either get stuck, or become unstable during training
- Skip Connection
  - Gradient of earlier parameters depends more directly on output
  - Identity function easier to learn



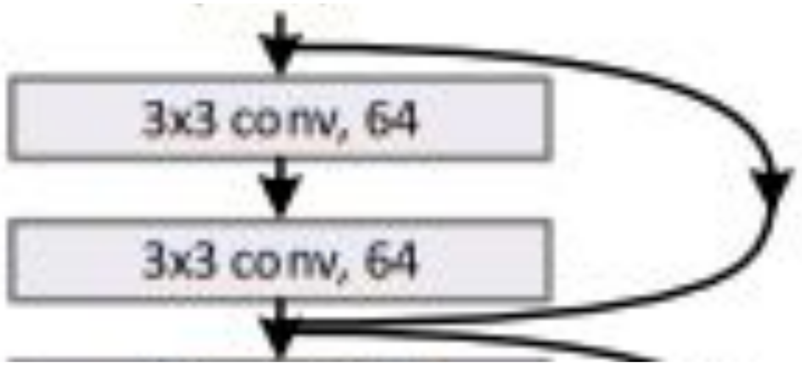
VGG vs. ResNet



VGG vs. ResNet



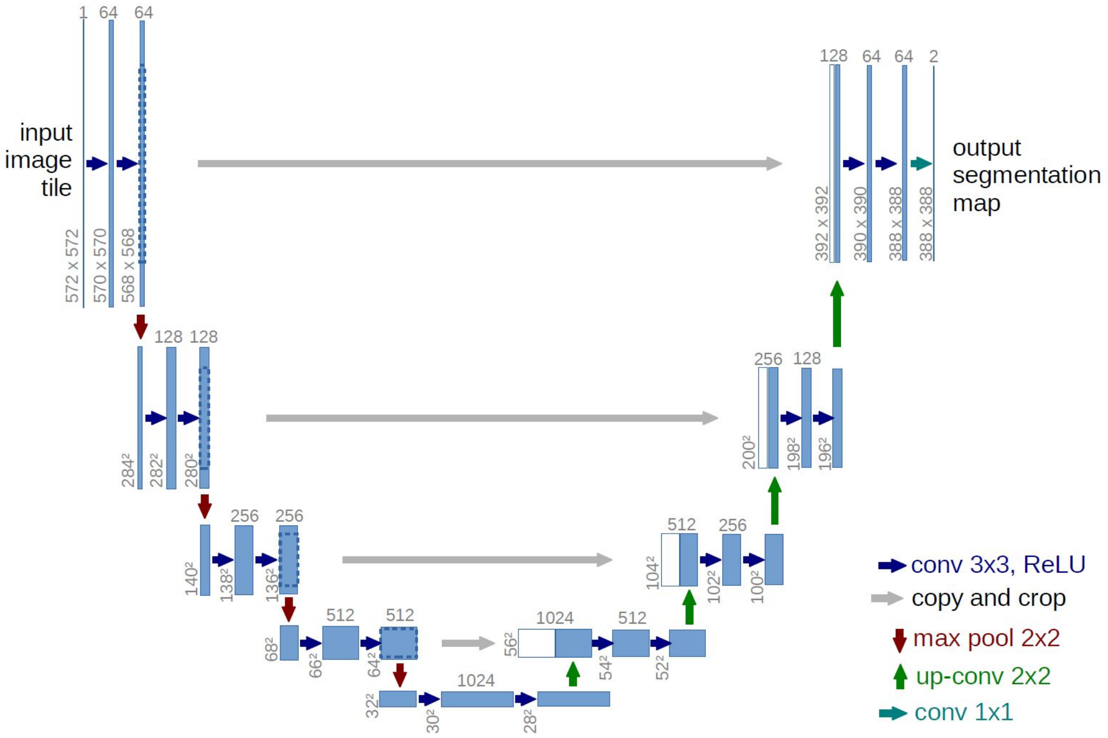
"Residual Block"



Other Techniques

- 1x1 Convolutions
  - With a 1x1 filter size you can condense the channel dimension
- Up-convolution
  - “Up-sample” to increase resolution using parameters
  - UNet
- Adaptive Pooling for Fully Convolutional Networks (FCNs)
  - Pool different shaped images to get same size output
- Normalization
  - Batch Normalization, Layer Normalization, Group Normalization
- 1D/3D Convolutions
  - For 3D: filter size maybe 3x3x3, input is of size (C,H,W,L)

UNet



1
64
64
input
image
tile
572 x 572
570 x 570
568 x 568
128
128
284²
282²
280²
256
256
140²
138²
136²
512
512
68²
66²
64²
1024
32²
30²
28²
1024
512
56²
54²
52²
512
256
104²
102²
100²
256
128
200²
198²
196²
128
64
64
2
392 x 392
390 x 390
388 x 388
388 x 388
output
segmentation
map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1