Deep Learning
MSDS 631

Images and
Convolutional Neural Networks

Michael Ruddy

Questions?

- From last lecture?
- From the lab assignment?

Overview

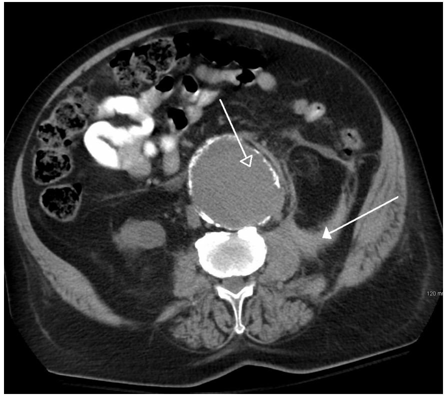
- Why imaging? Why not FF NNs?
- What/Why is a Convolution?
- CNN-specific hyperparameters
- Basic CNN history/set-up

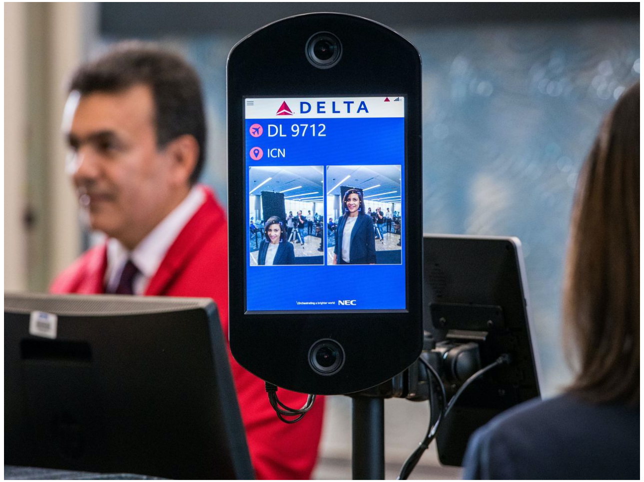
Why Imaging?

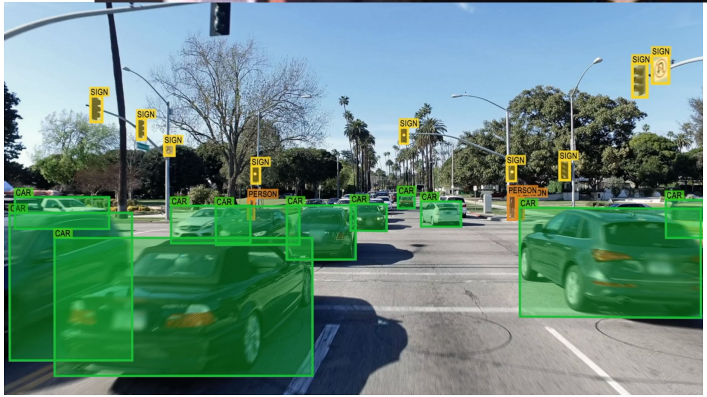
- Humans are really good at looking at things
  - The human eye/brain is an incredibly complicated piece of machinery
- Efforts to recreate vision based on human models of vision have largely been unsuccessful

Why Imaging?

- Humans are really good at looking at things
  - The human eye/brain is an incredibly complicated piece of machinery
- Efforts to recreate vision based on human models of vision have largely been unsuccessful
- Imaging is important!







Why Imaging?

- Humans are really good at looking at things
  - The human eye/brain is an incredibly complicated piece of machinery
- Efforts to recreate vision based on human models of vision have largely been unsuccessful
- Imaging is important!
  - Classification (facial/object recognition, avoid poisonous plants, etc.)
  - Medical Imaging (detecting disease, predicting outcomes of radiation, segmentation of medical images)
  - Autonomous Driving (driver assistance, fully autonomous vehicles)
  - Deepfakes and deepfake detection

Why Imaging?

- Humans are really good at looking at things
  - The human eye/brain is an incredibly complicated piece of machinery
- Efforts to recreate vision based on human models of vision have largely been unsuccessful
- Imaging is important!
  - Classification (facial/object recognition, avoid poisonous plants, etc.)
  - Medical Imaging (detecting disease, predicting outcomes of radiation, segmentation of medical images)
  - Autonomous Driving (driver assistance, fully autonomous vehicles)
  - Deepfakes and deepfake detection
- A lot of these are time-consuming things that human can do really well

Why are images special?

- Images are deceptively hard

Why are images special?

- Images are deceptively hard

This is a cat

Why are images special?

- Images are deceptively hard

This is ??????

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

Why are images special?

- Images are deceptively hard
- Images are big

32x32 image
1024 features

512x512 image
262,144 features

Why are images special?

- Images are deceptively hard
- Images are big

32x32 image
1024 features

512x512 image
262,144 features

Fully Connected Layer

- 1024 -> 1024
- 1024^2 = 1,048,576 parameters

- 262,144 -> 262,144
- 68,719,476,736 parameters

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

Different relationships

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

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)

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

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)

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

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

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

"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”

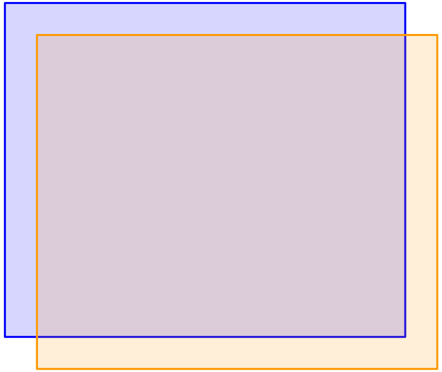
The Convolution

- 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"

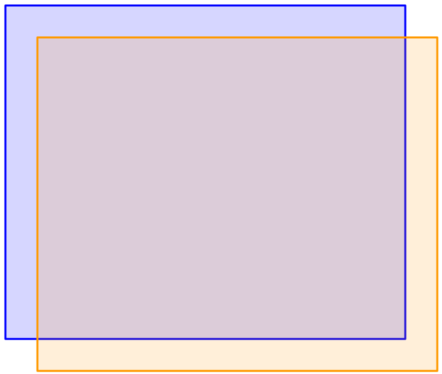
The Convolution

- 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

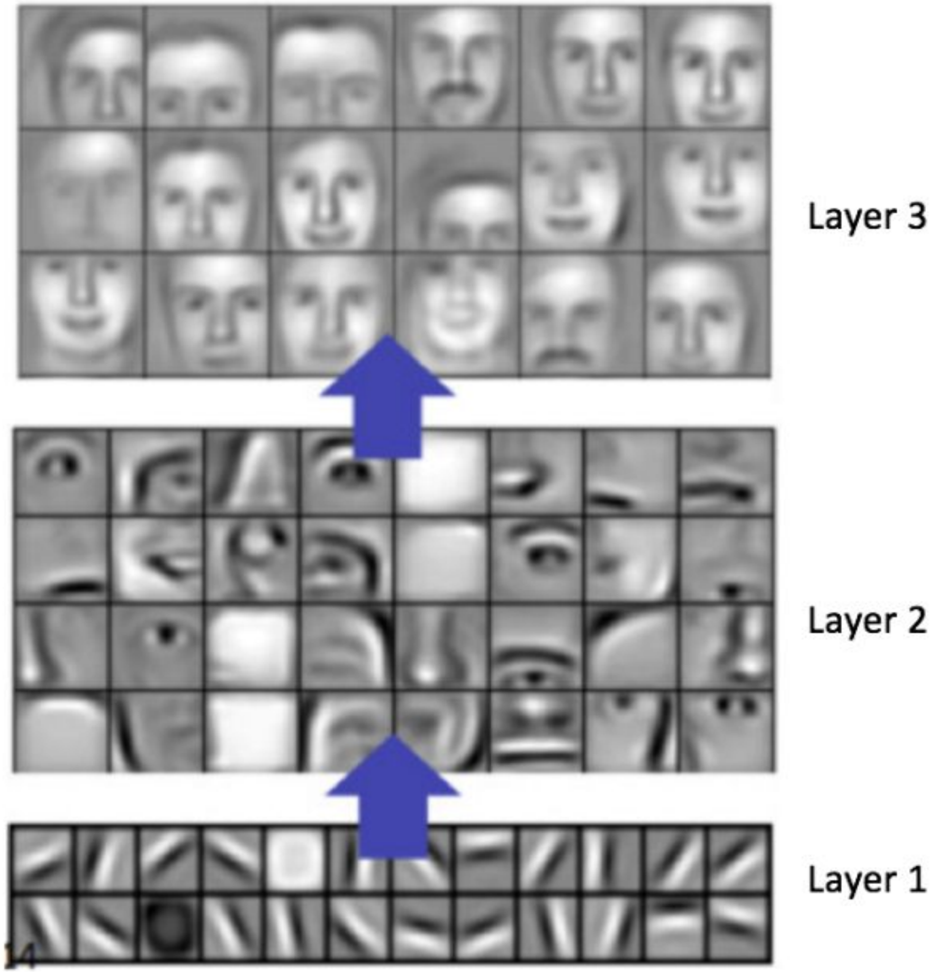
More Parameters

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



“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

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

$$\begin{bmatrix} .2 & .15 & 0 & 0 \\ .2 & 1 & 1 & 1 \\ 1 & .5 & .5 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

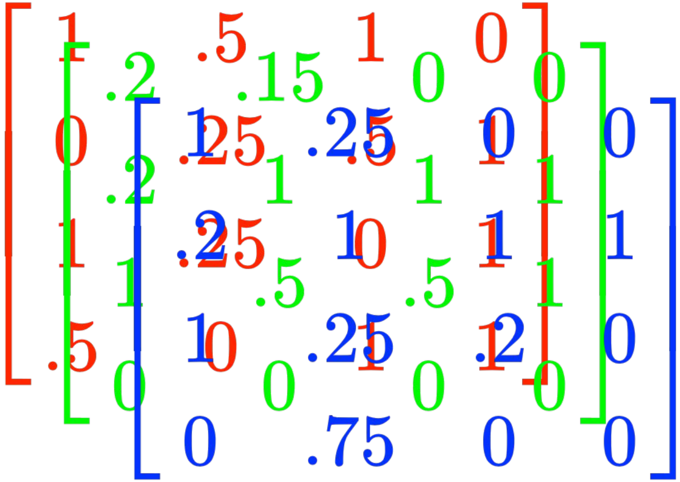
$$\begin{bmatrix} 1 & .25 & 0 & 0 \\ .2 & 1 & 1 & 1 \\ 1 & .25 & .2 & 0 \\ 0 & .75 & 0 & 0 \end{bmatrix}$$

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”

4x4x3

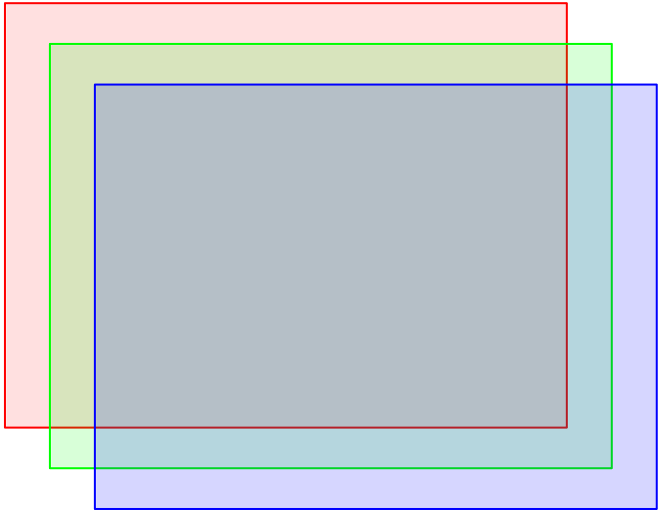
RGB Image



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

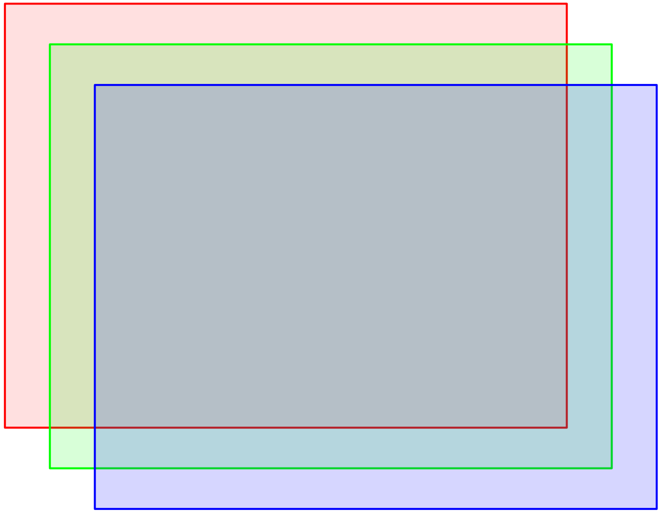


The Convolution

- 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”

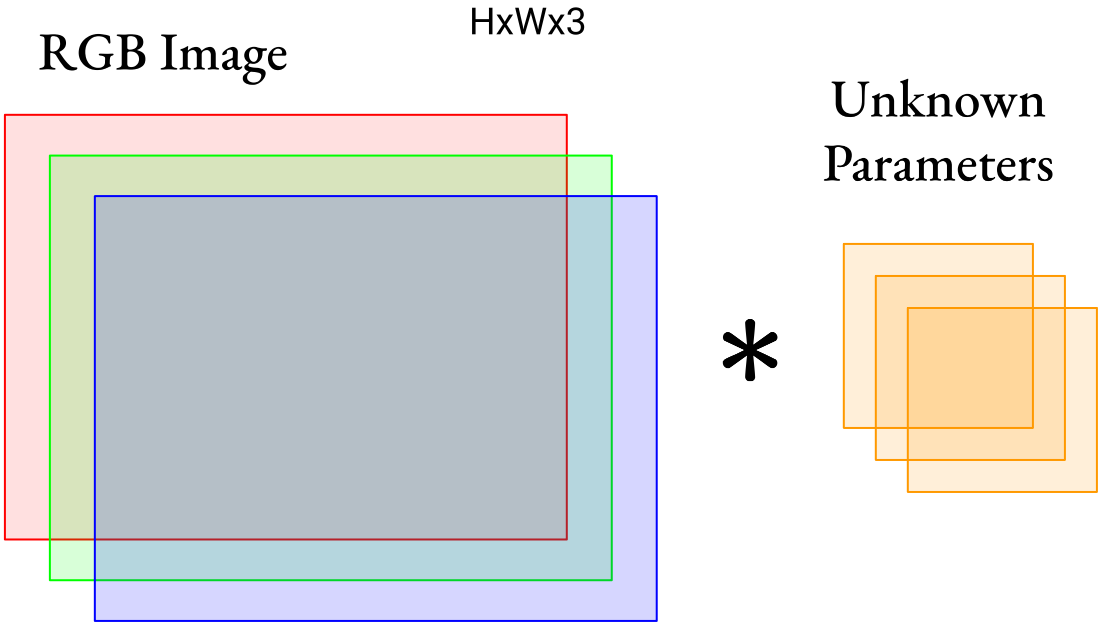
HxWx3

RGB Image



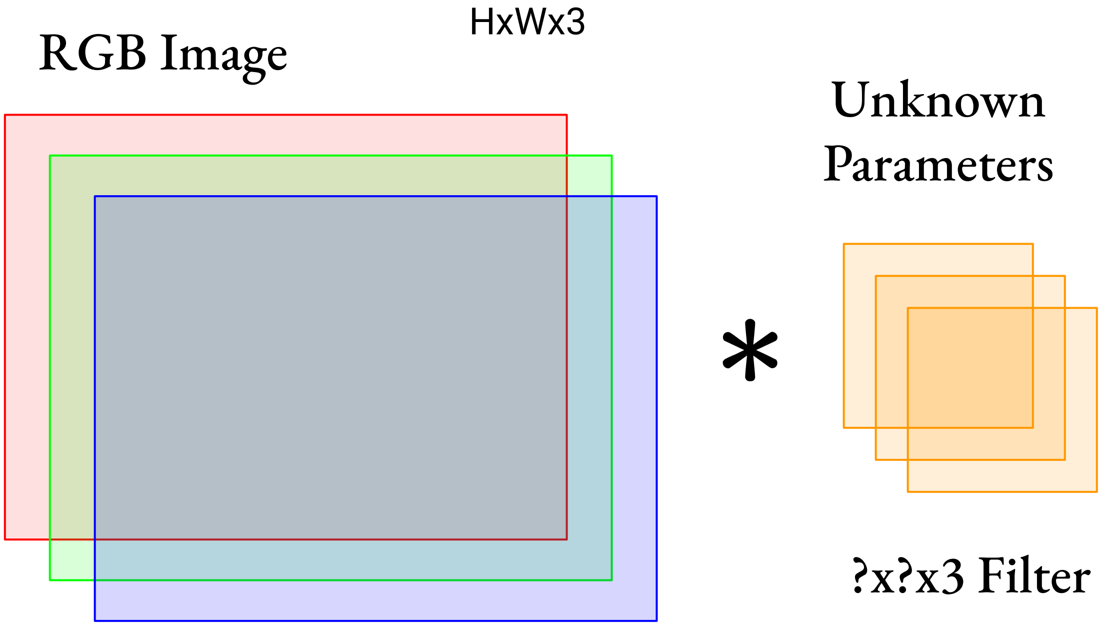
The Convolution

- 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!!!



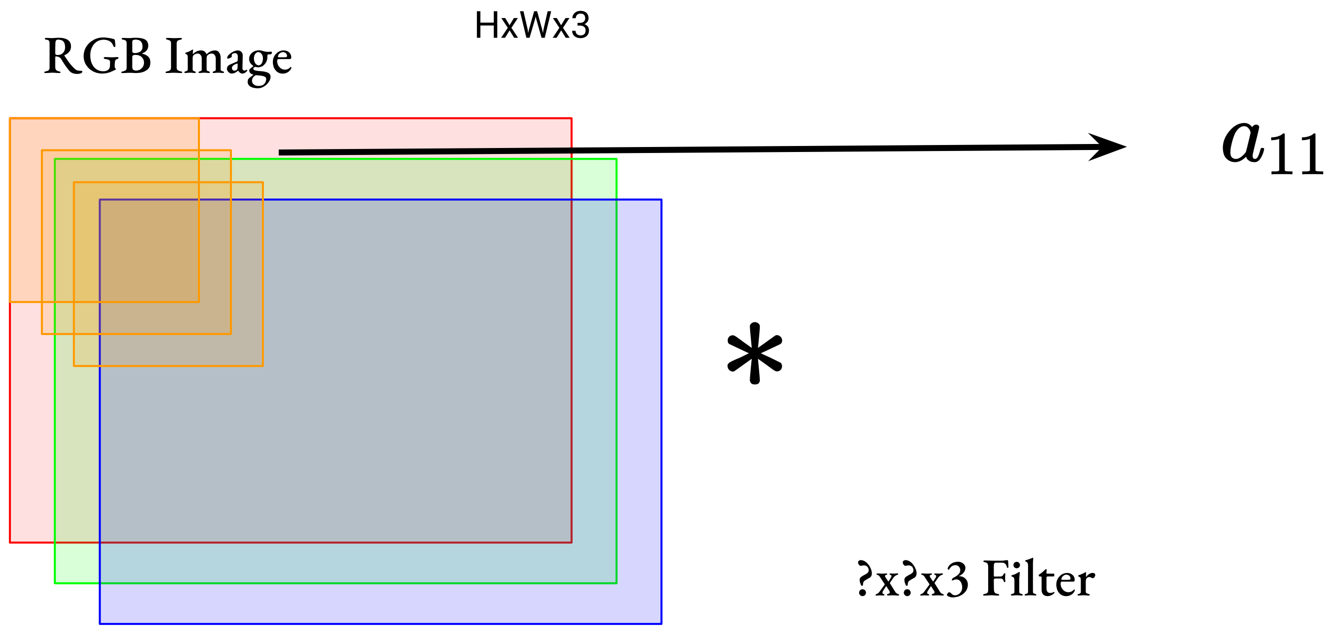
The Convolution

- 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!!!



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!!!

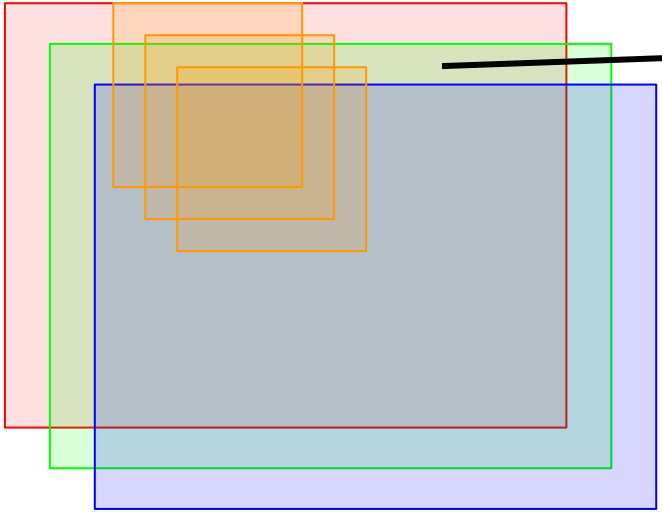


The Convolution

- 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



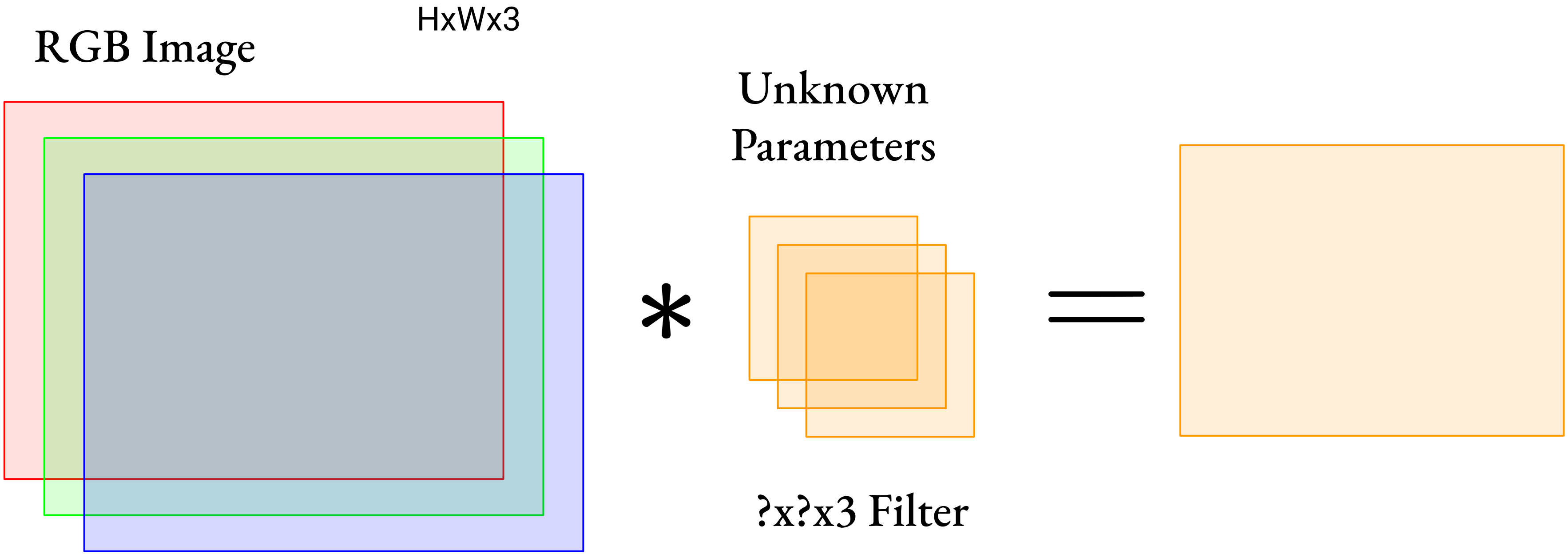
$$a_{11} \rightarrow a_{12}$$

$*$

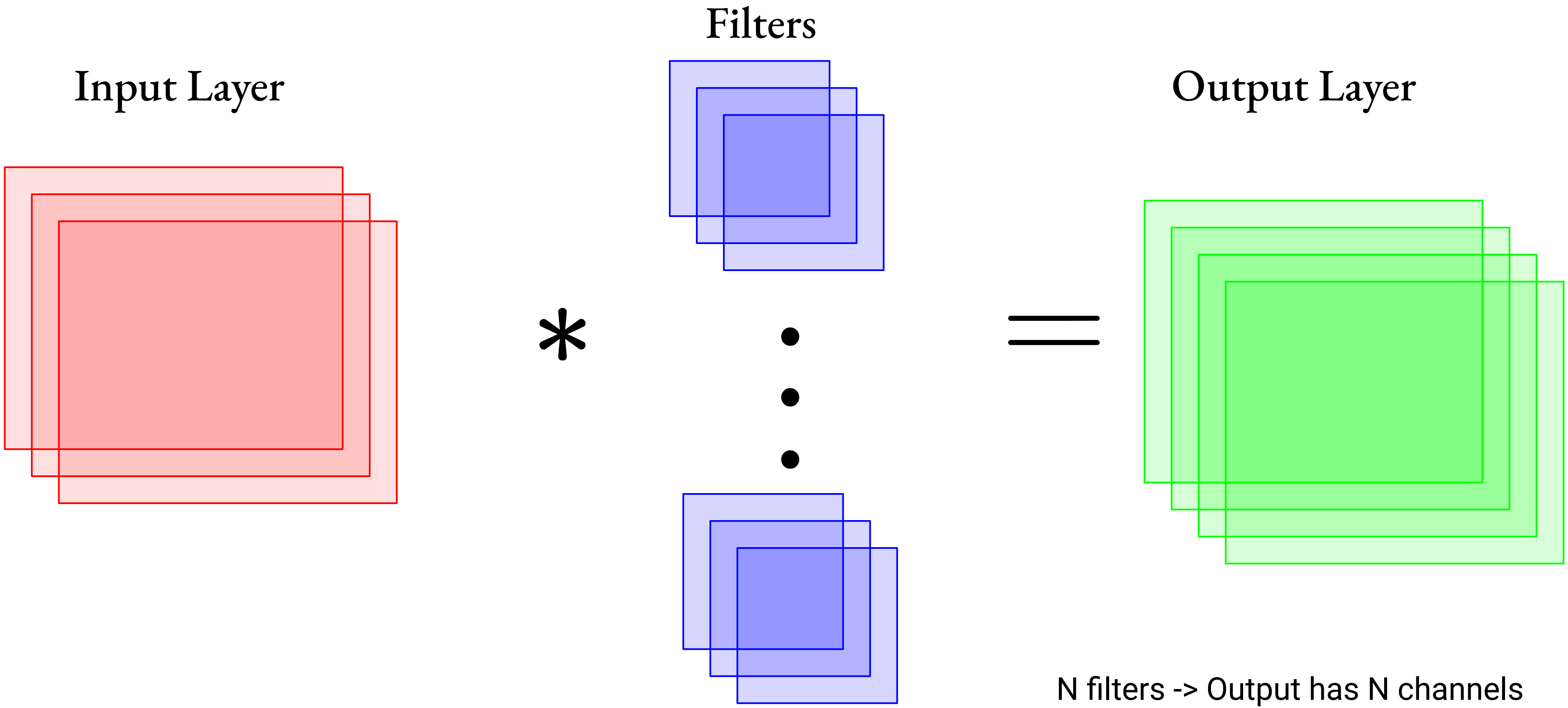
?x?x3 Filter

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



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

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

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

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

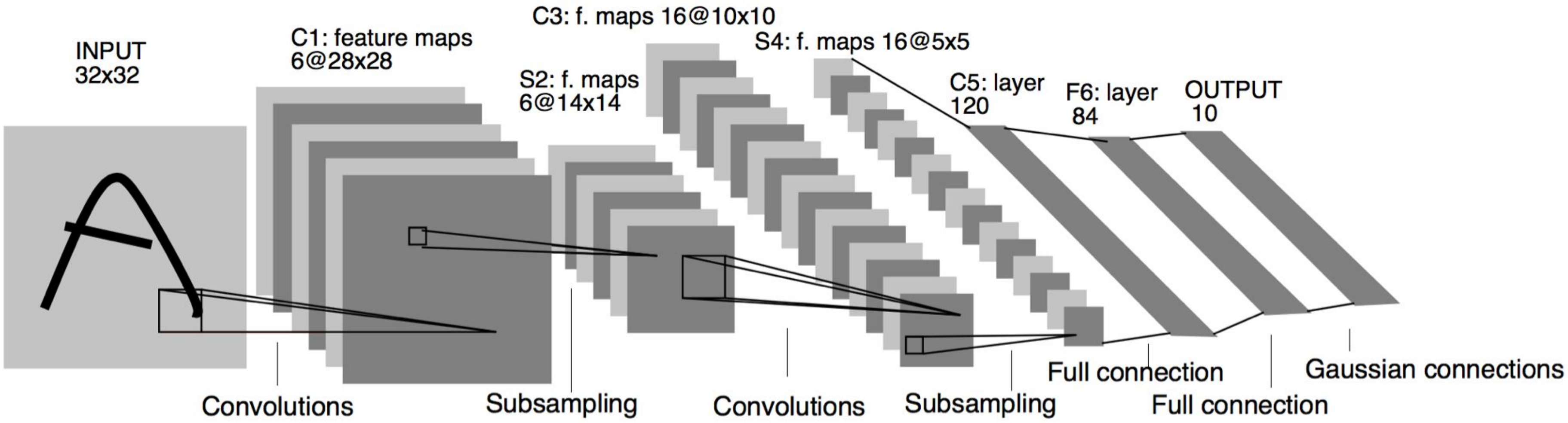
Convolution Hyperparameters

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

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

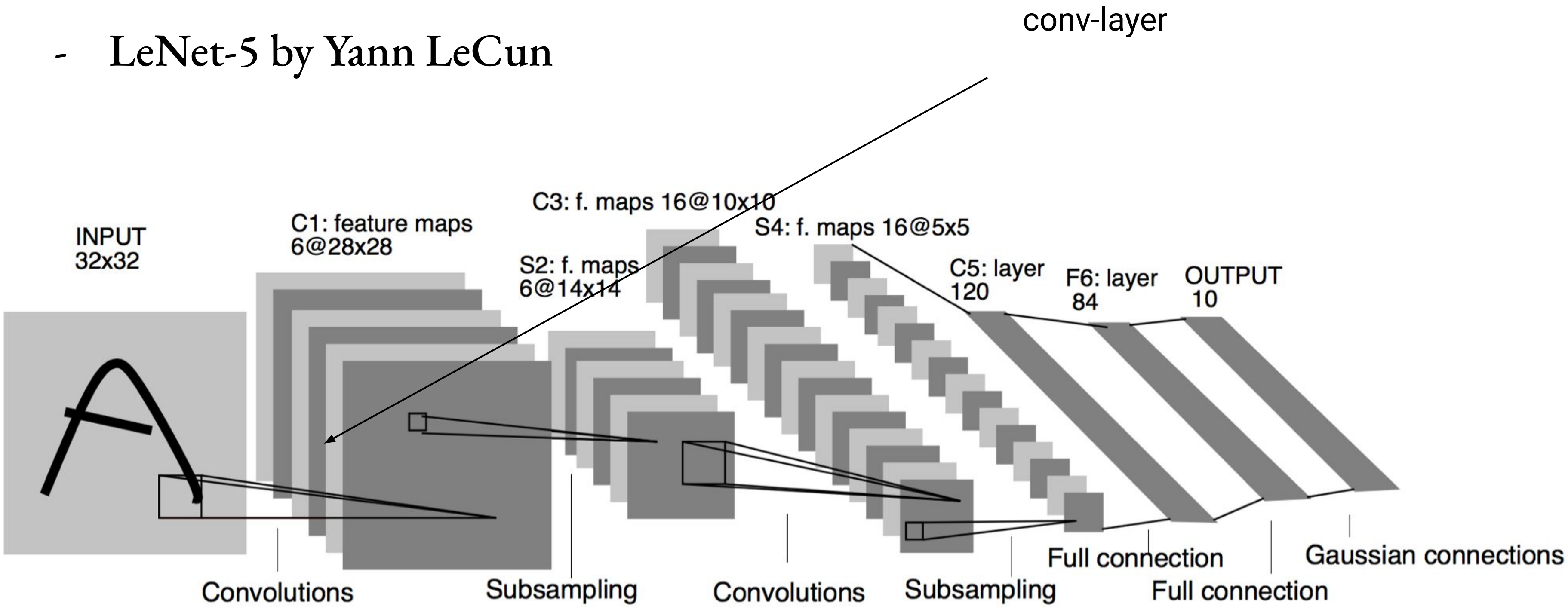
Convolutional Neural Network

- LeNet-5 by Yann LeCun



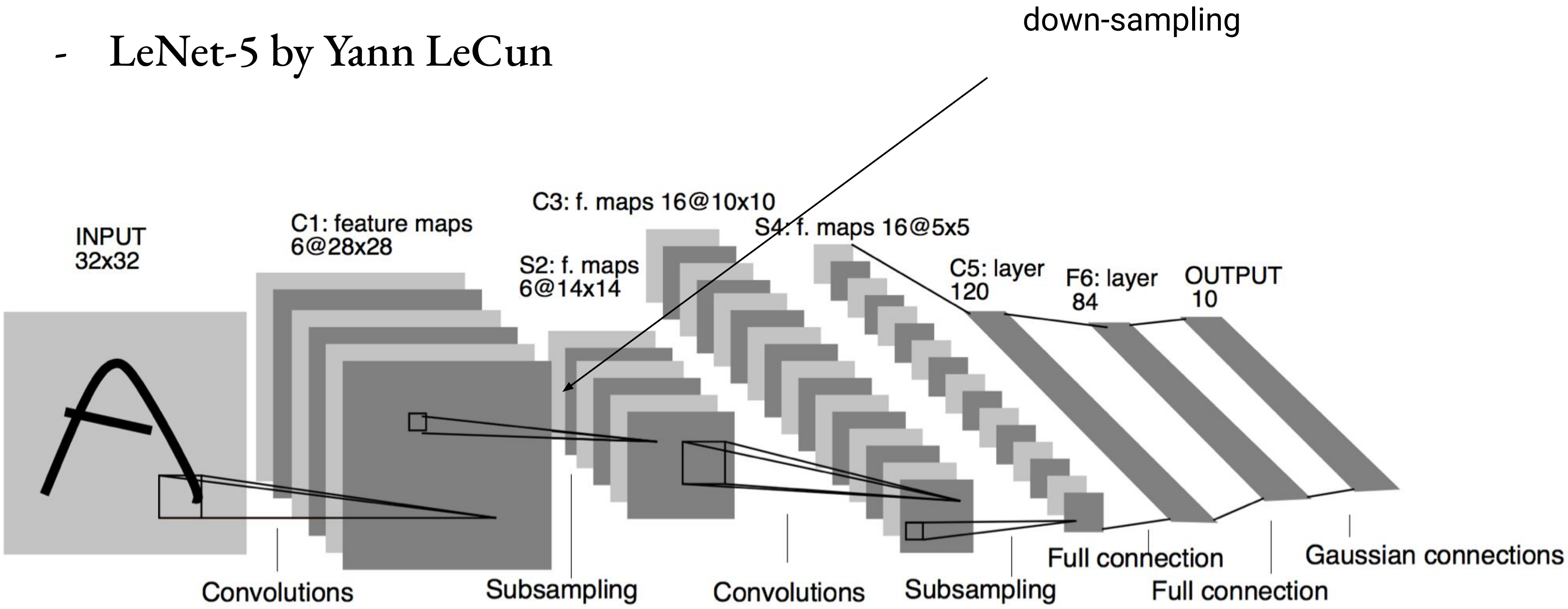
Convolutional Neural Network

- LeNet-5 by Yann LeCun



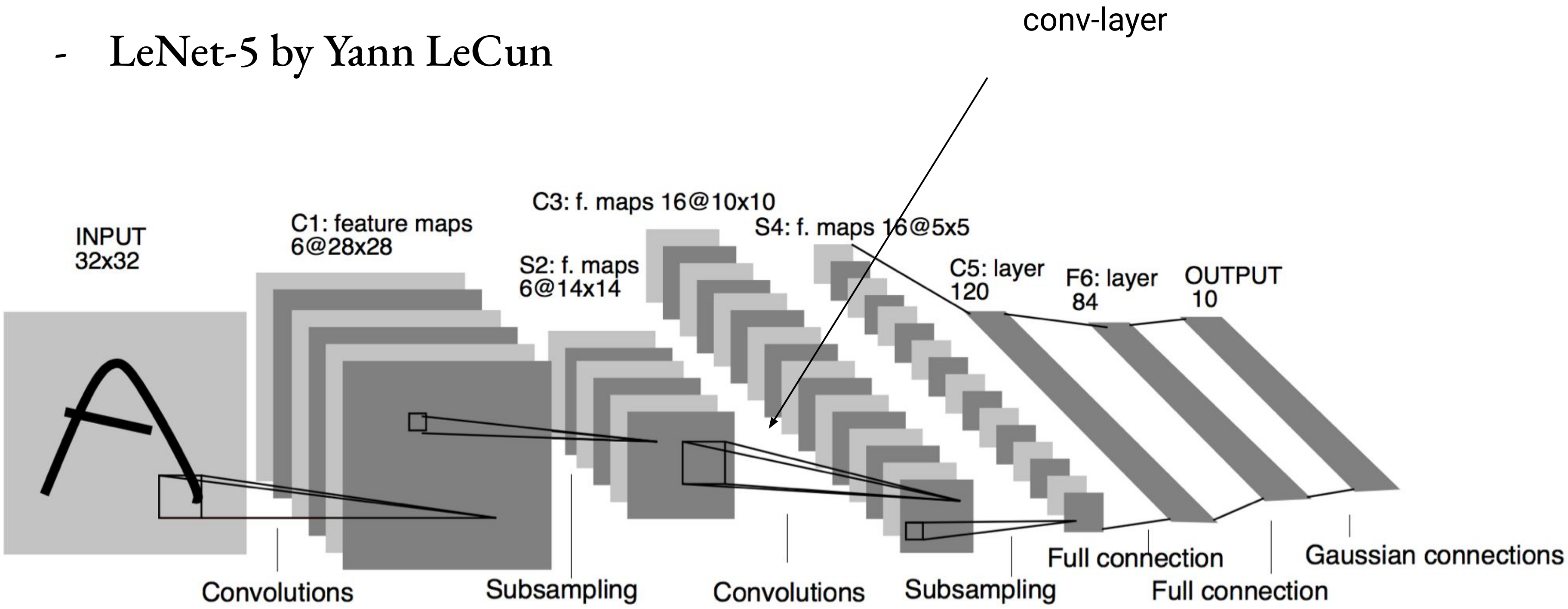
Convolutional Neural Network

- LeNet-5 by Yann LeCun



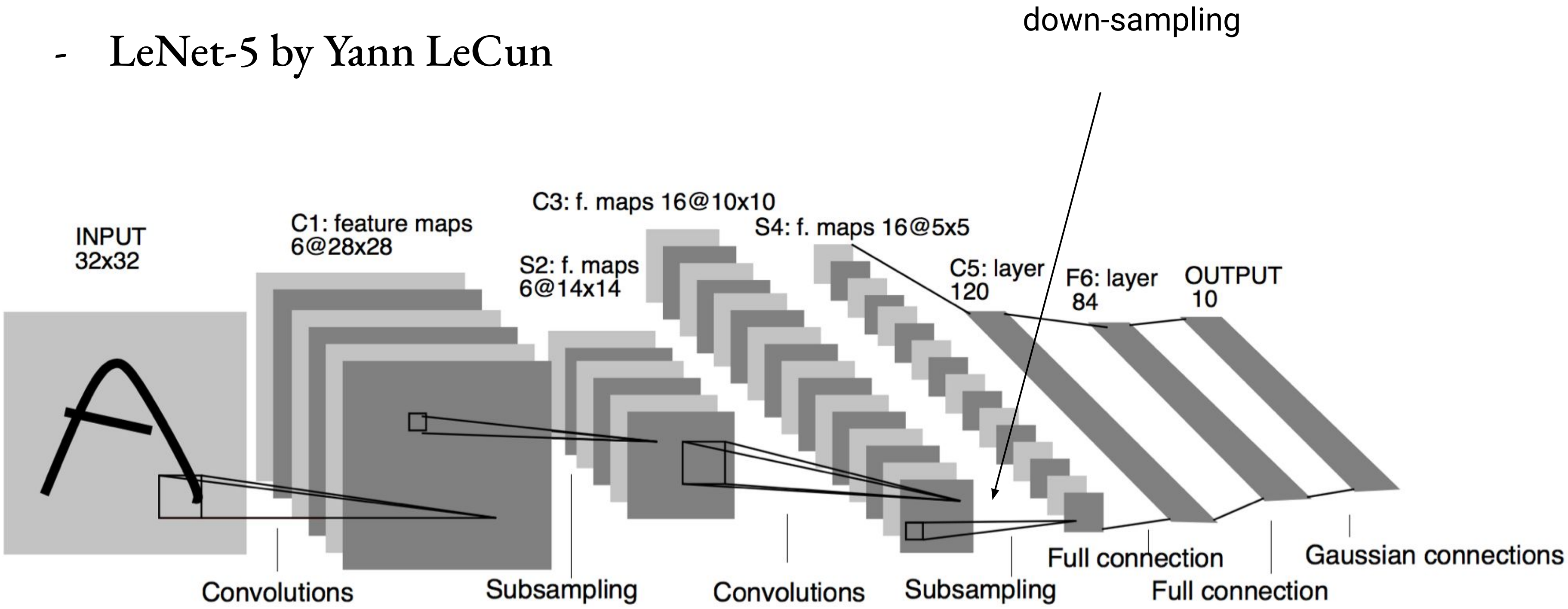
Convolutional Neural Network

- LeNet-5 by Yann LeCun



Convolutional Neural Network

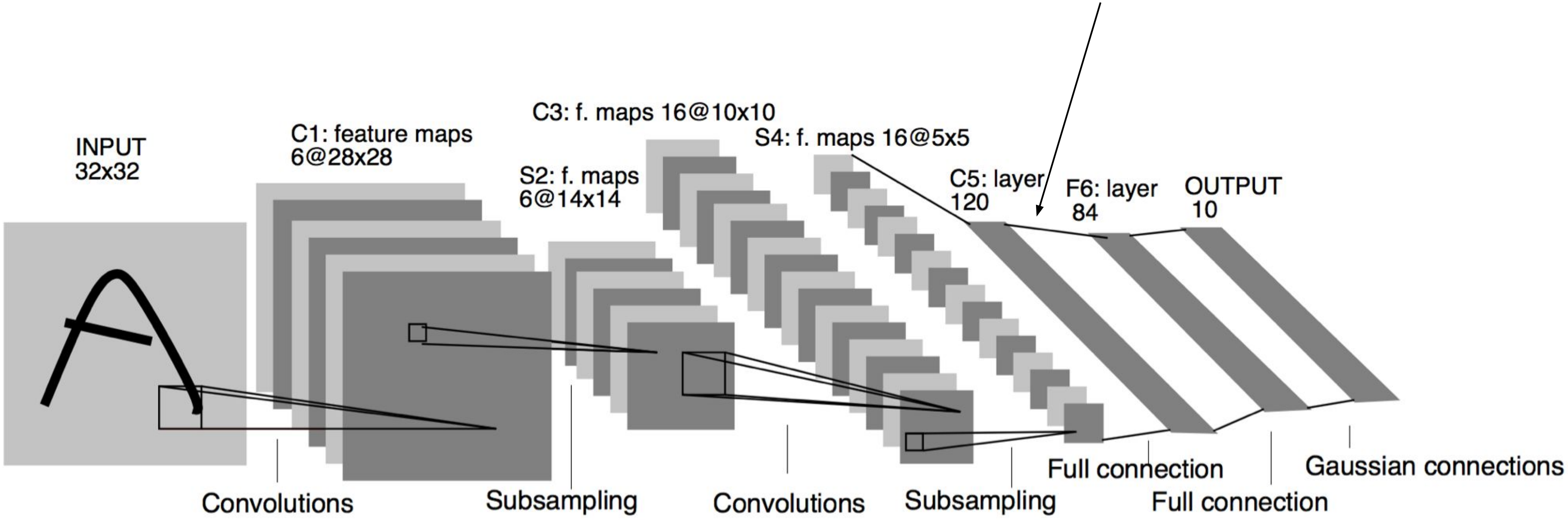
- LeNet-5 by Yann LeCun



Convolutional Neural Network

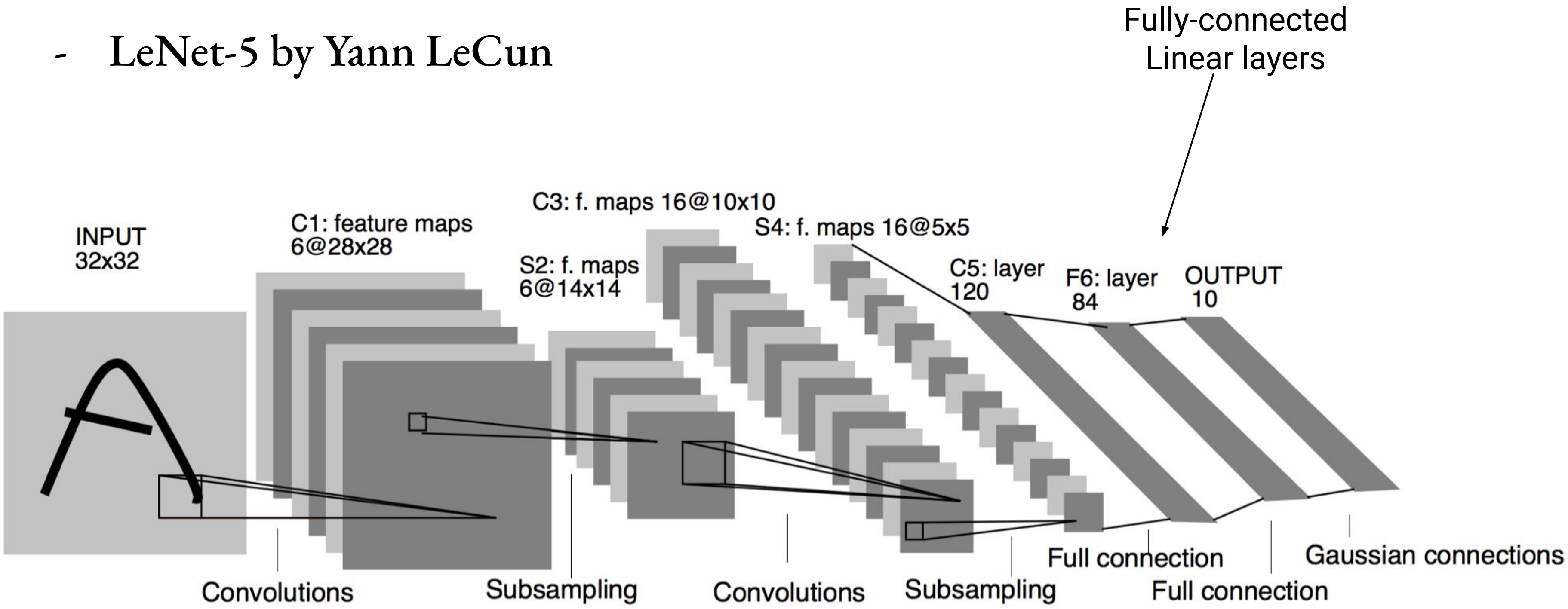
- LeNet-5 by Yann LeCun

“Unroll” into a
one-dimensional vector



Convolutional Neural Network

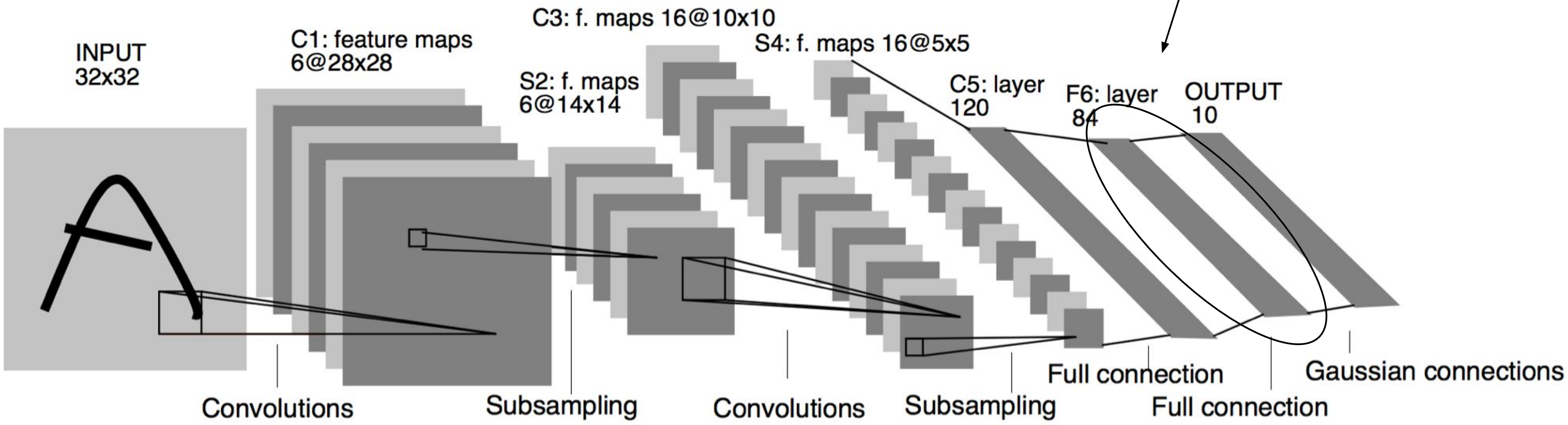
- LeNet-5 by Yann LeCun



Convolutional Neural Network

- LeNet-5 by Yann LeCun

“Representation” of the image!



Convolutional Neural Network

- AlexNet wins ImageNet Competition in 2012

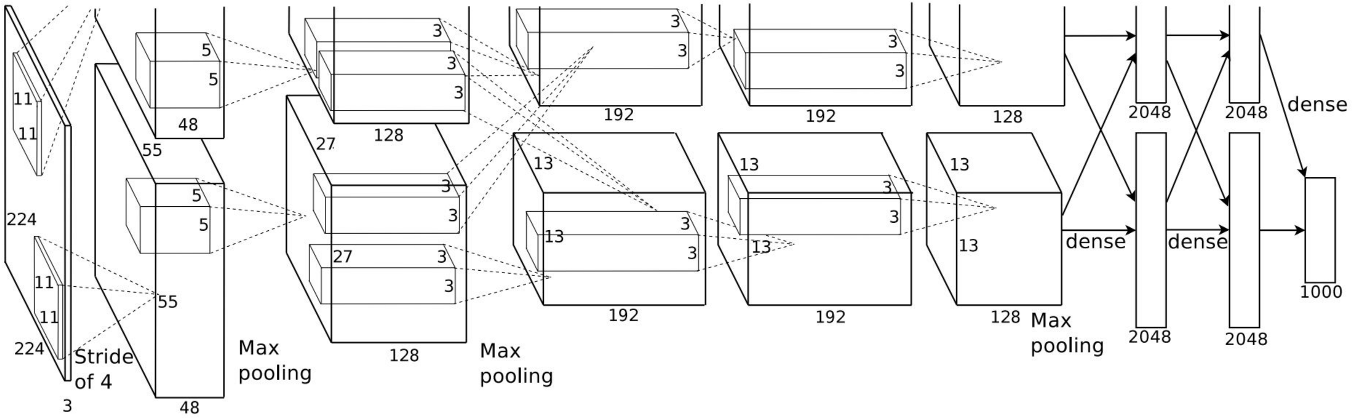


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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

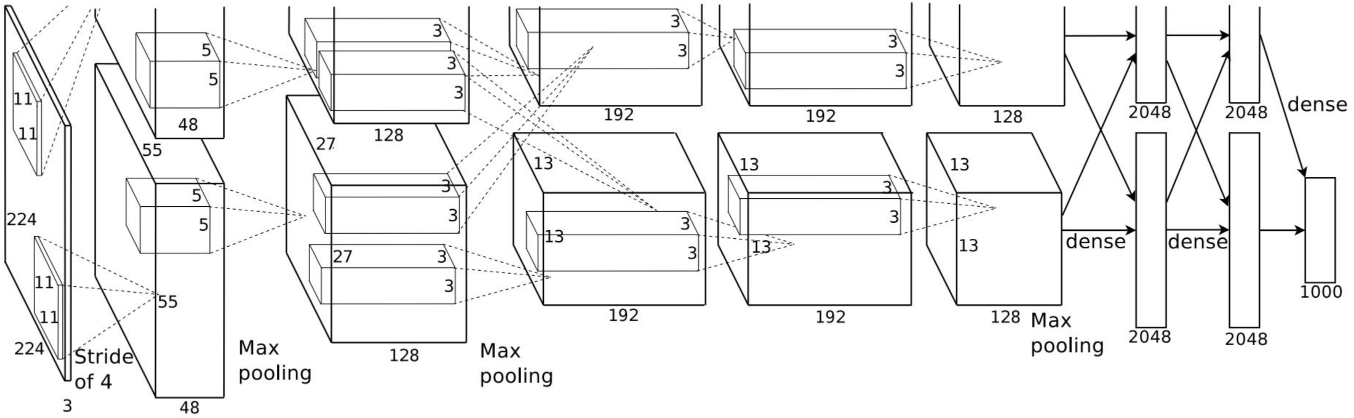


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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

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

Downsampling

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

- 2x2 filter size
- Stride 2

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

Downsampling

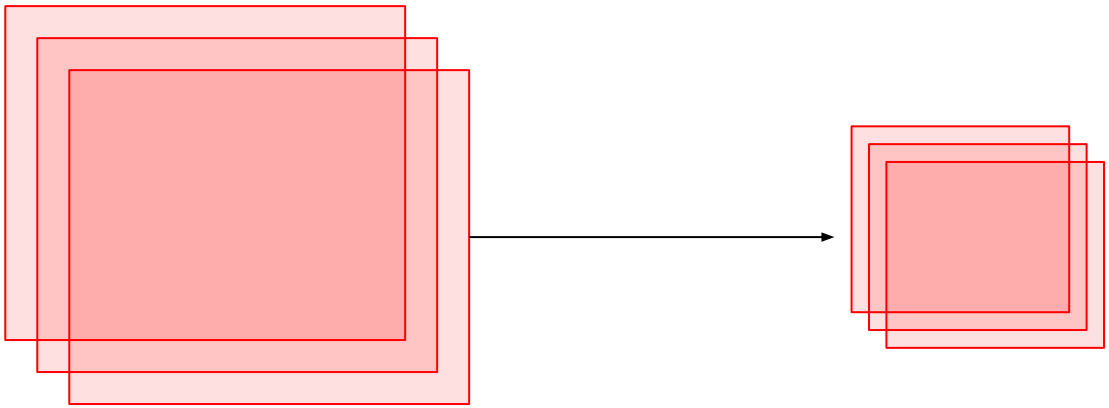
- Reduce size of output
- Minimal information loss in practice
- Intuition: reduce resolution of the image
- Max Pooling
- Average Pooling

- 2x2 filter size
- Stride 2

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

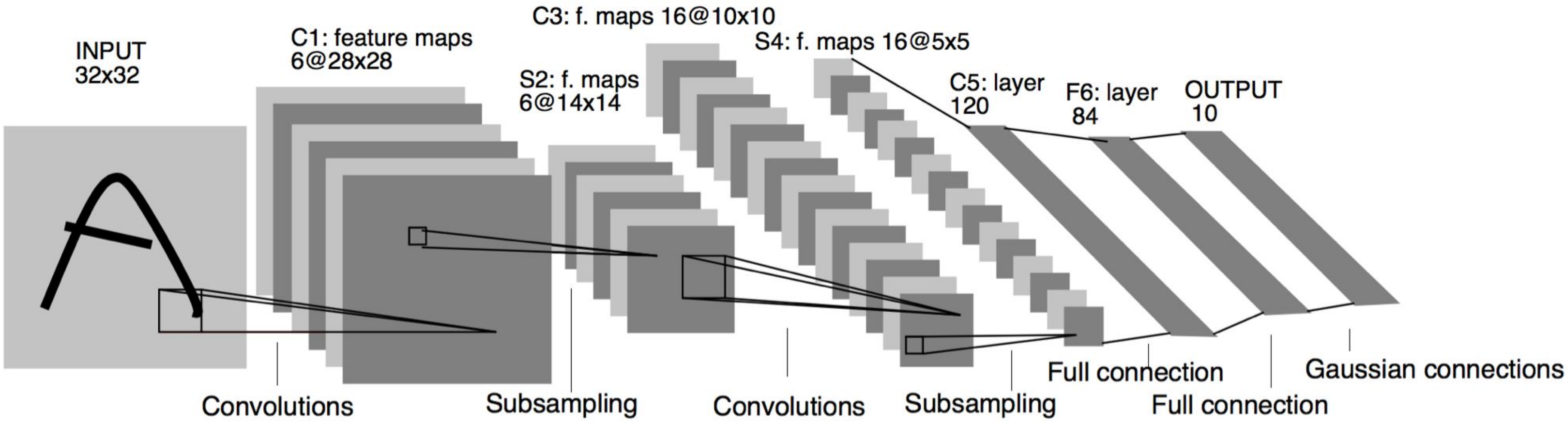
Downsampling

- 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