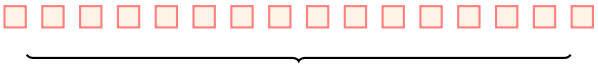
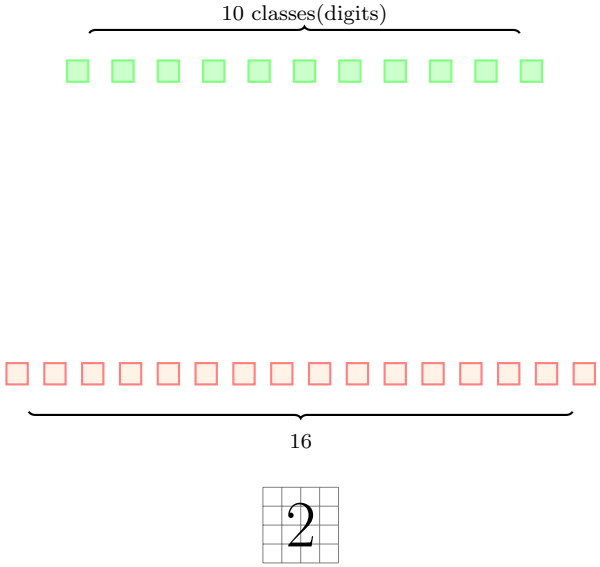
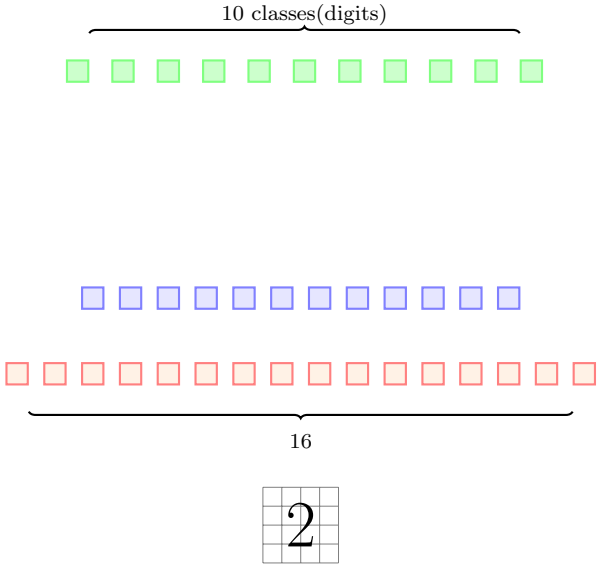
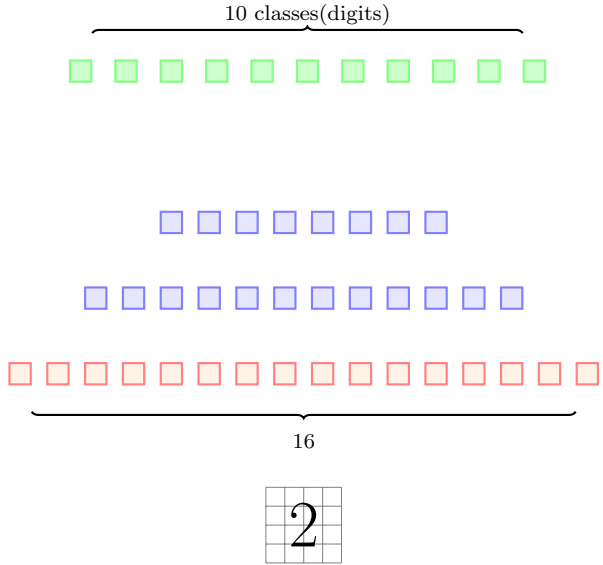
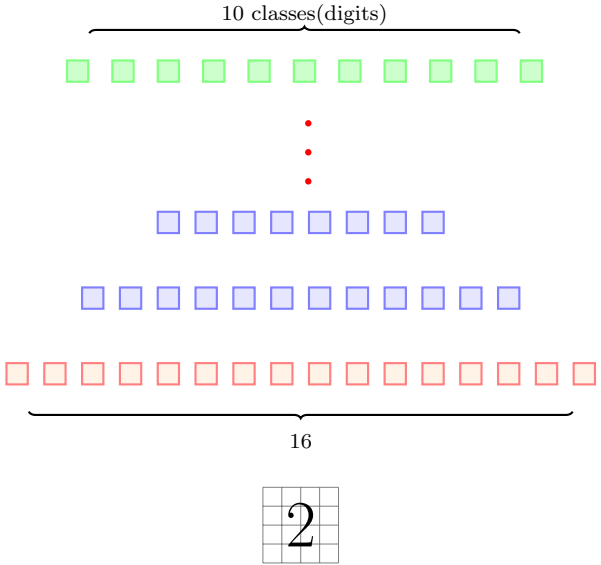

16

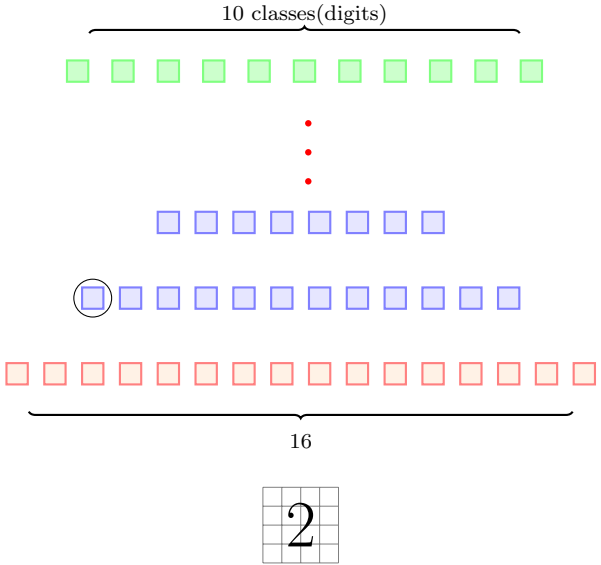

2

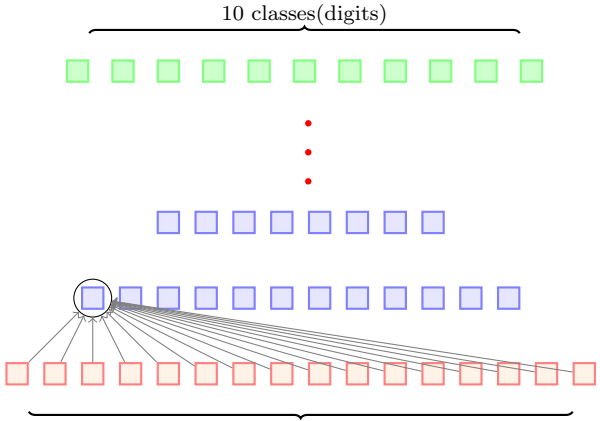

10 classes(digits)
16


10 classes(digits)
16

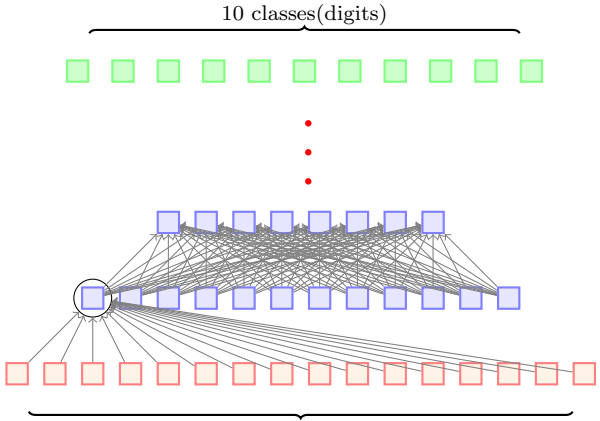

10 classes(digits)
16


10 classes(digits)
16

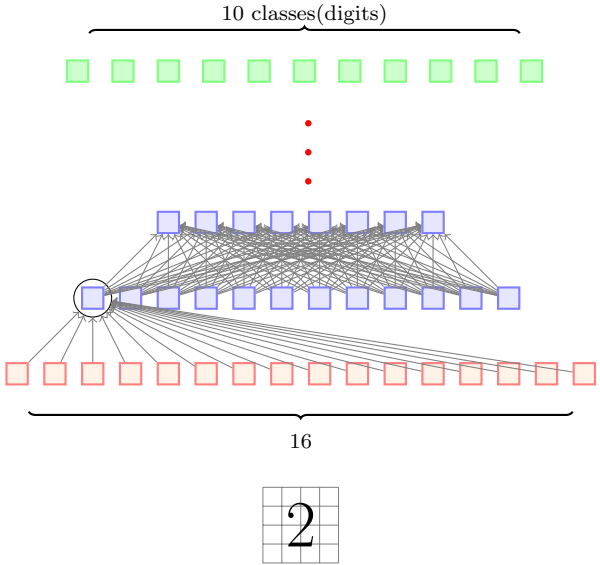

10 classes(digits)
16


10 classes(digits)
16

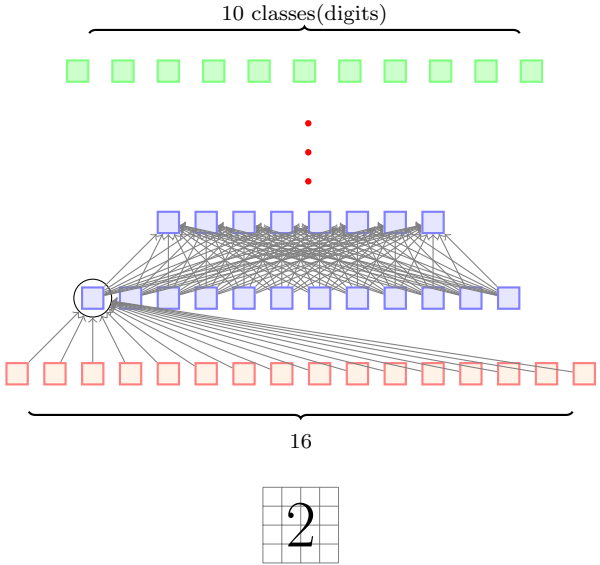

2


10 classes(digits)
16

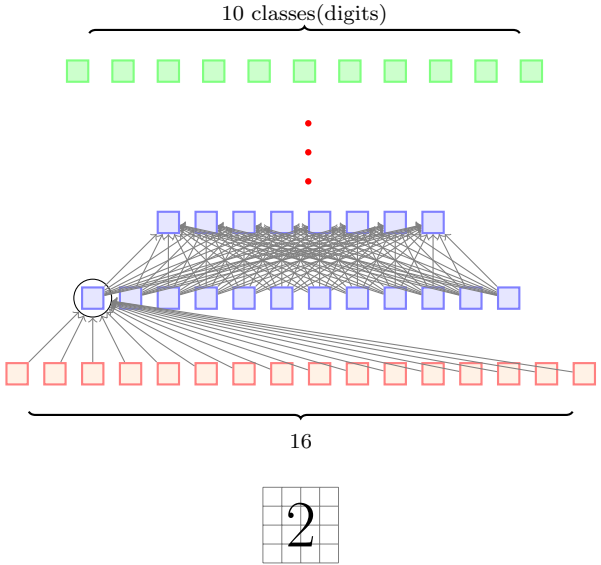

2



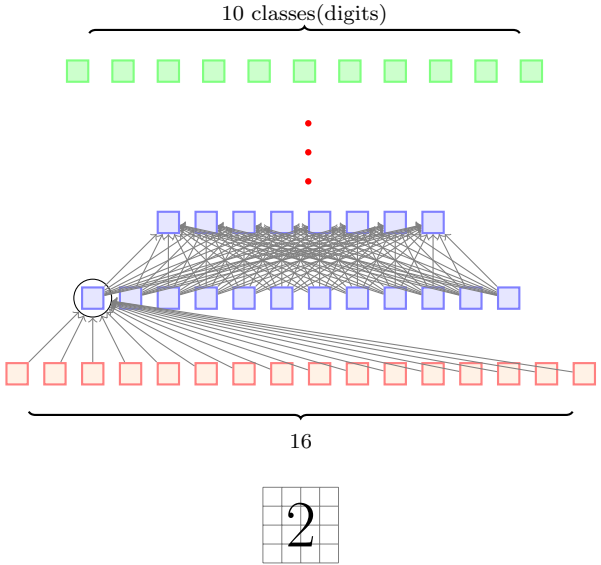
- This is what a regular feed-forward neural network will look like



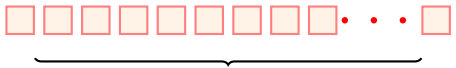
- This is what a regular feed-forward neural network will look like
- There are many dense connections here

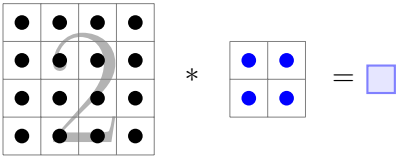


- This is what a regular feed-forward neural network will look like
- There are many dense connections here
- For example all the 16 input neurons are contributing to the computation of h_{11}

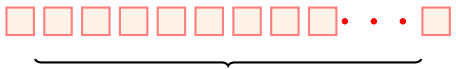


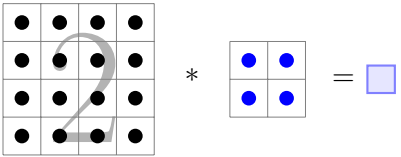
- This is what a regular feed-forward neural network will look like
- There are many dense connections here
- For example all the 16 input neurons are contributing to the computation of h_{11}
- Contrast this to what happens in the case of convolution

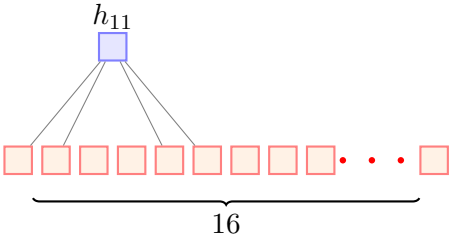

16

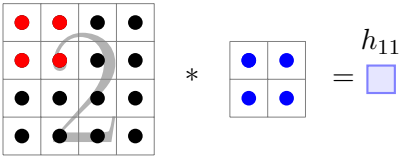

2
*
=

- Only a few local neurons participate in the computation of h_{11}

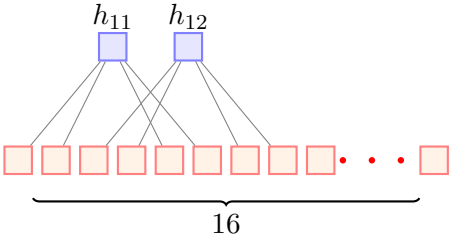


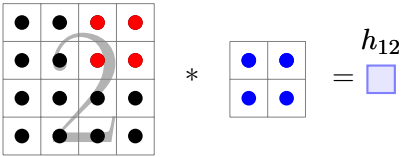




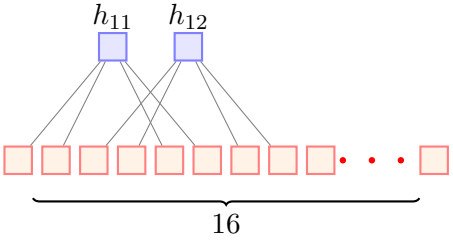


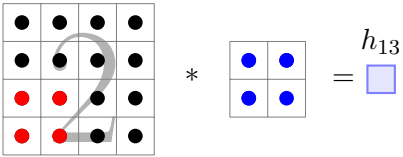
- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}



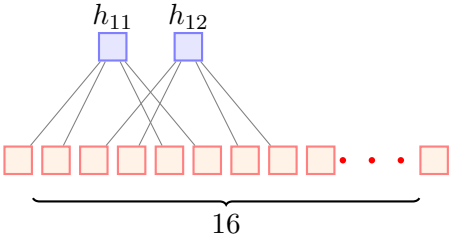


- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}

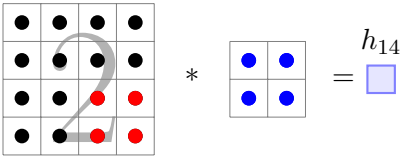


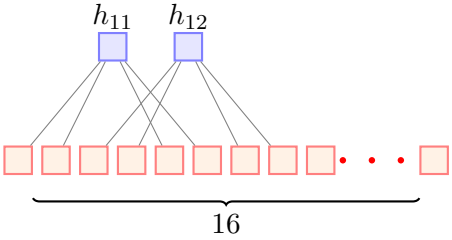


- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}

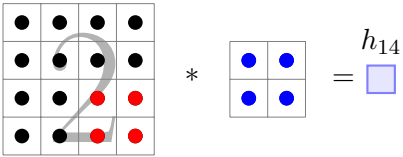


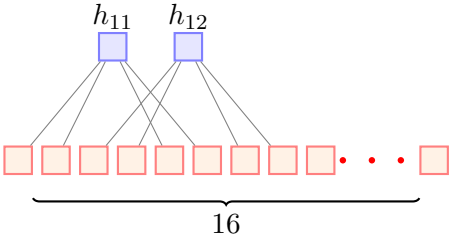
- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}

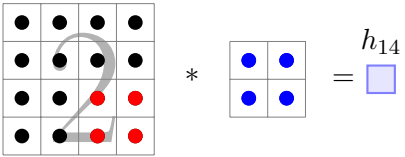




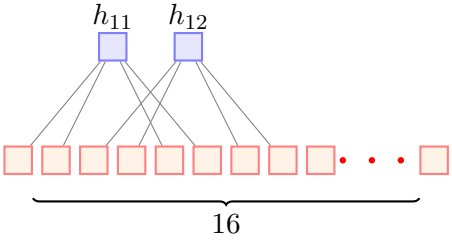
- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}
- The connections are much sparser

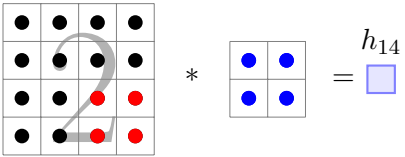






- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}
- The connections are much sparser
- We are taking advantage of the structure of the image(interactions between neighboring pixels are more interesting)





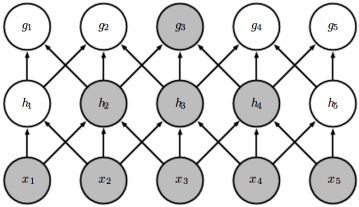
- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}
- The connections are much sparser
- We are taking advantage of the structure of the image(interactions between neighboring pixels are more interesting)
- This **sparse connectivity** reduces the number of parameters in the model

- But is sparse connectivity really good thing ?

* Goodfellow-et-al-2016

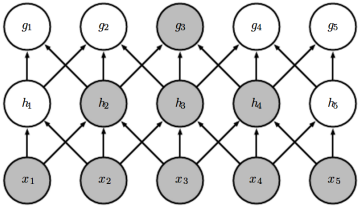
- But is sparse connectivity really good thing ?
- Aren't we losing information (by losing interactions between some input pixels)

* Goodfellow-et-al-2016



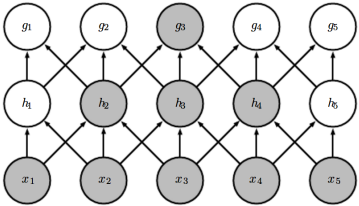
- But is sparse connectivity really good thing ?
- Aren't we losing information (by losing interactions between some input pixels)
- Well, not really

* Goodfellow-et-al-2016



- But is sparse connectivity really good thing ?
- Aren't we losing information (by losing interactions between some input pixels)
- Well, not really
- The two highlighted neurons (x_1 & x_5)* do not interact in *layer* 1

* Goodfellow-et-al-2016



- But is sparse connectivity really good thing ?
- Aren't we losing information (by losing interactions between some input pixels)
- Well, not really
- The two highlighted neurons (x_1 & x_5)* do not interact in *layer* 1
- But they indirectly contribute to the computation of g_3 and hence interact indirectly

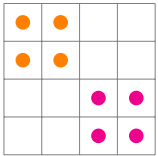
* Goodfellow-et-al-2016

- Another characteristic of CNNs is **weight sharing**

- Another characteristic of CNNs is **weight sharing**
- Consider the following network

- Another characteristic of CNNs is **weight sharing**
- Consider the following network

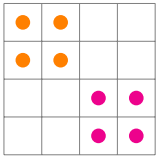
- Kernel 1
- Kernel 2



4x4 Image

- Another characteristic of CNNs is **weight sharing**
- Consider the following network
- Do we want the kernel weights to be different for different portions of the image?

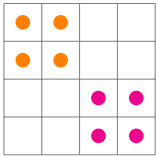
- Kernel 1
- Kernel 2



4x4 Image

- Another characteristic of CNNs is **weight sharing**
- Consider the following network
- Do we want the kernel weights to be different for different portions of the image?
- Imagine that we are trying to learn a kernel that detects edges

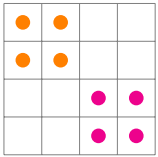
- Kernel 1
- Kernel 2



4x4 Image

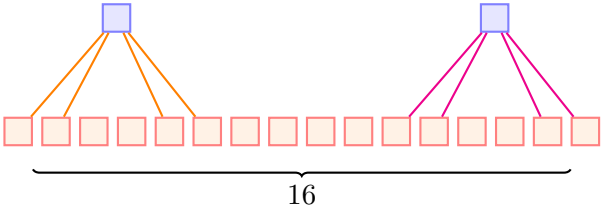
- Another characteristic of CNNs is **weight sharing**
- Consider the following network
- Do we want the kernel weights to be different for different portions of the image?
- Imagine that we are trying to learn a kernel that detects edges
- Shouldn't we be applying the same kernel at all the portions of the image?

- Kernel 1
- Kernel 2

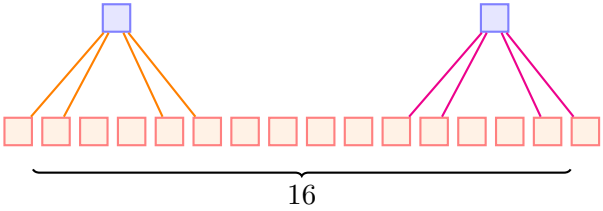


4x4 Image

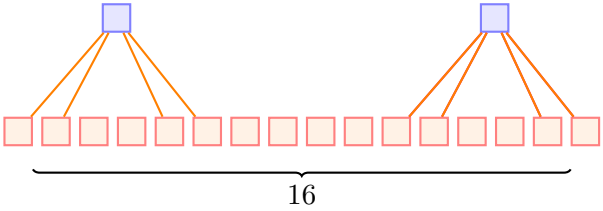
- In other words shouldn't the *orange* and *pink* kernels be the same

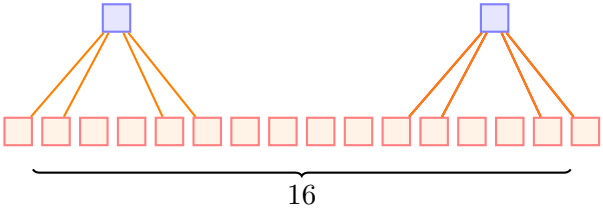


- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed

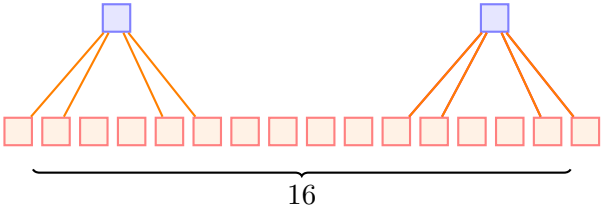


- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed

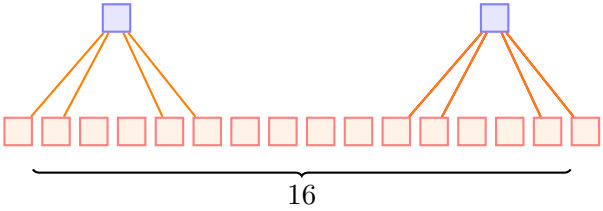




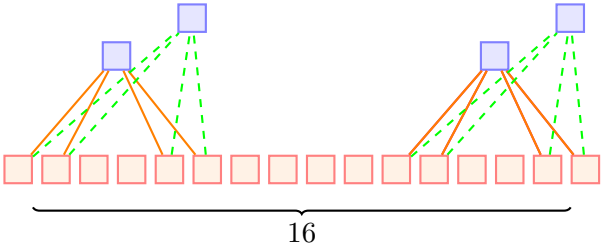
- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed
- This would make the job of learning easier(instead of trying to learn the same weights/kernels at different locations again and again)



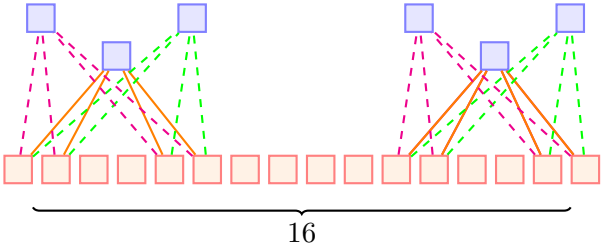
- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed
- This would make the job of learning easier(instead of trying to learn the same weights/kernels at different locations again and again)
- But does that mean we can have only one kernel?



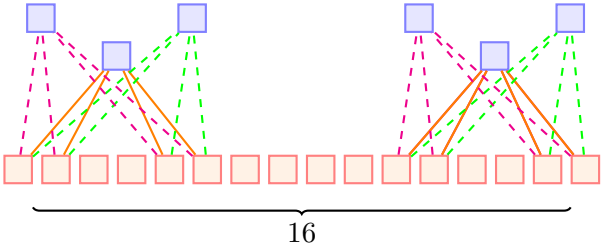
- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed
- This would make the job of learning easier(instead of trying to learn the same weights/kernels at different locations again and again)
- But does that mean we can have only one kernel?
- No, we can have many such kernels but the kernels will be shared by all locations in the image



- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed
- This would make the job of learning easier(instead of trying to learn the same weights/kernels at different locations again and again)
- But does that mean we can have only one kernel?
- No, we can have many such kernels but the kernels will be shared by all locations in the image



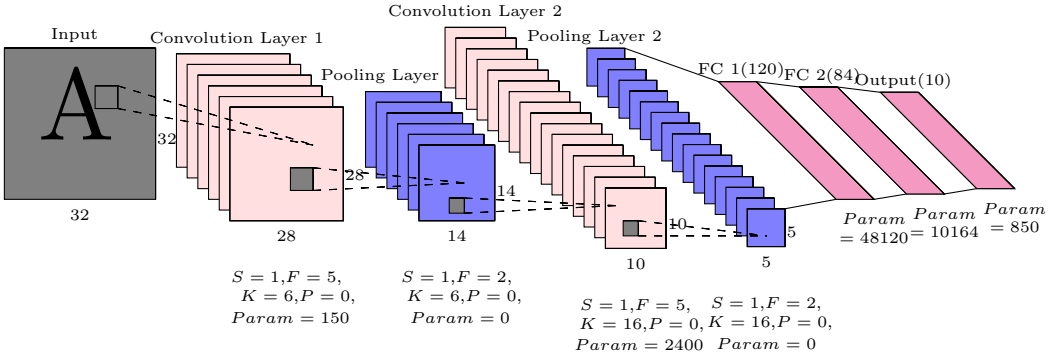
- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed
- This would make the job of learning easier(instead of trying to learn the same weights/kernels at different locations again and again)
- But does that mean we can have only one kernel?
- No, we can have many such kernels but the kernels will be shared by all locations in the image

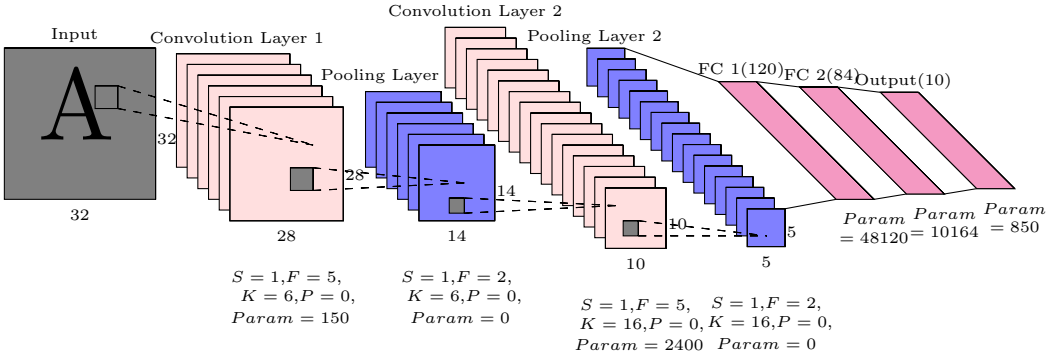


- In other words shouldn't the *orange* and *pink* kernels be the same
- Yes, indeed
- This would make the job of learning easier(instead of trying to learn the same weights/kernels at different locations again and again)
- But does that mean we can have only one kernel?
- No, we can have many such kernels but the kernels will be shared by all locations in the image
- This is called "weight sharing"

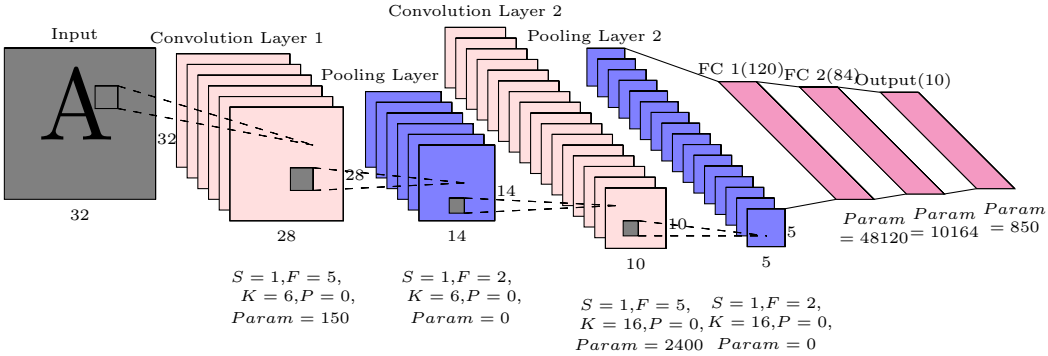
- So far, we have focused only on the convolution operation

- So far, we have focused only on the convolution operation
- Let us see what a full convolutional neural network looks like

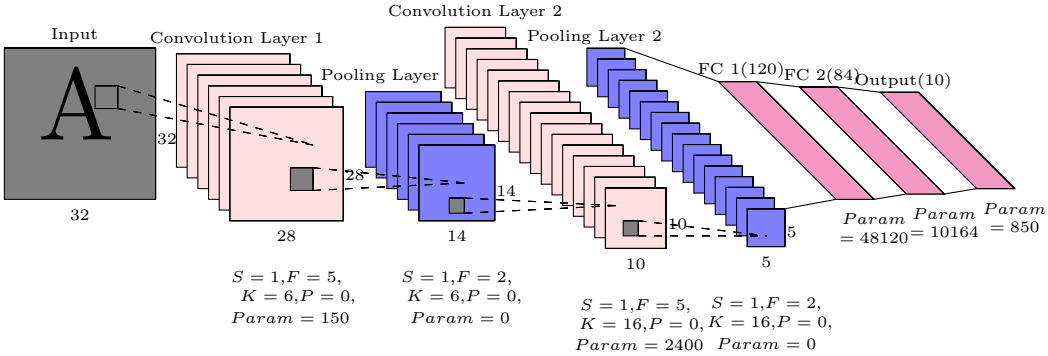

Input
Convolution Layer 1
Pooling Layer
Convolution Layer 2
Pooling Layer 2
FC 1(120)
FC 2(84)
Output(10)
32
32
28
28
14
14
10
10
5
5
$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$
$S = 1, F = 2,$
$K = 6, P = 0,$
$Param = 0$
$S = 1, F = 5,$
$K = 16, P = 0,$
$Param = 2400$
$S = 1, F = 2,$
$K = 16, P = 0,$
$Param = 0$
$Param = 48120$
$Param = 10164$
$Param = 850$



- It has alternate convolution and pooling layers



- It has alternate convolution and pooling layers
- What does a pooling layer do?

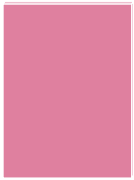


- It has alternate convolution and pooling layers
- What does a pooling layer do?
- Let us see

Input

Input

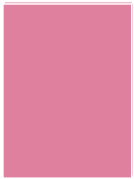
*



1 filter

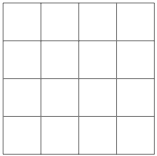
Input

*



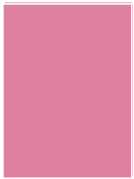
1 filter

=



Input

*



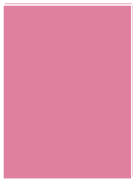
1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

Input

*



1 filter

=

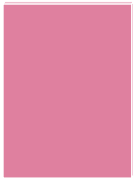
1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

Input

*

1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

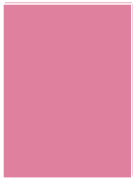
maxpool

2x2 filters (stride 2)

8	

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

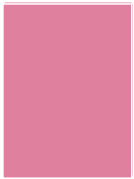
maxpool

2x2 filters (stride 2)

8	4

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

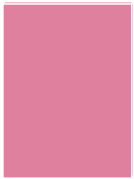
maxpool

2x2 filters (stride 2)

8	4
7	

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

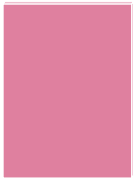
maxpool

2x2 filters (stride 2)

8	4
7	5

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

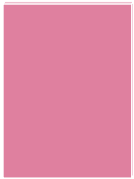
2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

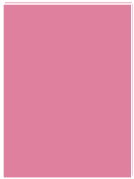
1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 1)

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 1)

Input

*

1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool
2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool
2x2 filters (stride 1)

8		

Input

*

1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

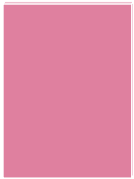
maxpool

2x2 filters (stride 1)

8	8	

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

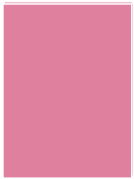
maxpool

2x2 filters (stride 1)

8	8	4

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

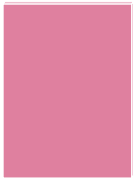
maxpool

2x2 filters (stride 1)

8	8	4
8		

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool
2x2 filters (stride 2)

8	4
7	5

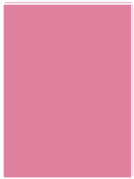
1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool
2x2 filters (stride 1)

8	8	4
8	8	

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

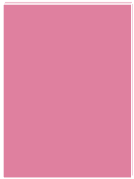
maxpool

2x2 filters (stride 1)

8	8	4
8	8	5

Input

*



1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool
2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool
2x2 filters (stride 1)

8	8	4
8	8	5
7		

Input

*

1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 1)

8	8	4
8	8	5
7	6	

Input

*

1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool

2x2 filters (stride 1)

8	8	4
8	8	5
7	6	5

Input

*

1 filter

=

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

maxpool
2x2 filters (stride 2)

8	4
7	5

1	4	2	1
5	8	3	4
7	6	4	5
1	3	1	2

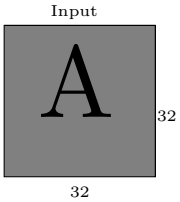
maxpool
2x2 filters (stride 1)

8	8	4
8	8	5
7	6	5

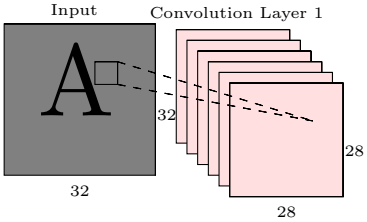
- Instead of max pooling we can also do average pooling

We will now see some case studies where convolution neural networks have been successful

LeNet-5 for handwritten character recognition

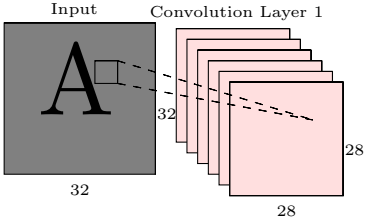


LeNet-5 for handwritten character recognition



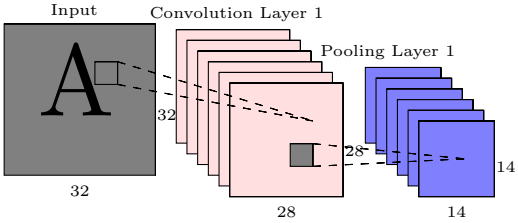
$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = ?$

LeNet-5 for handwritten character recognition

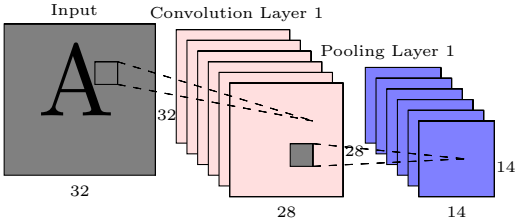


$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$

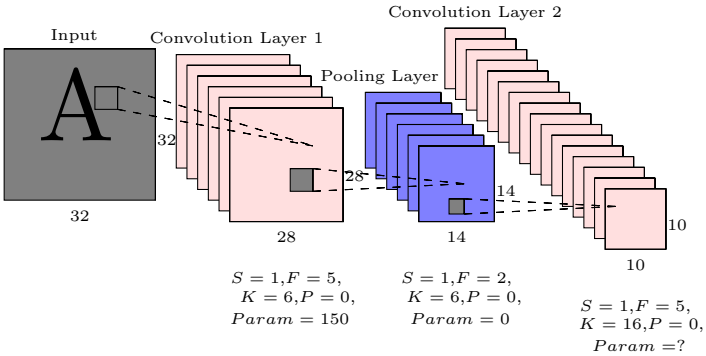
LeNet-5 for handwritten character recognition



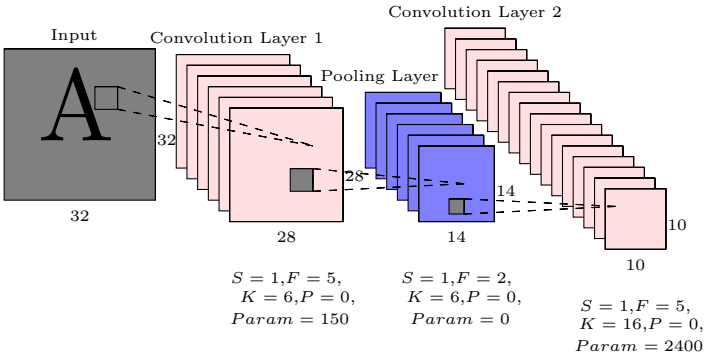
LeNet-5 for handwritten character recognition



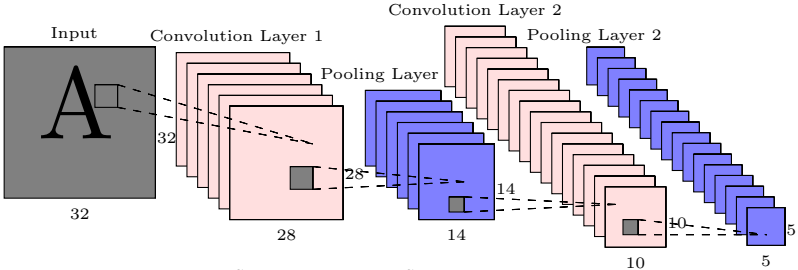
LeNet-5 for handwritten character recognition



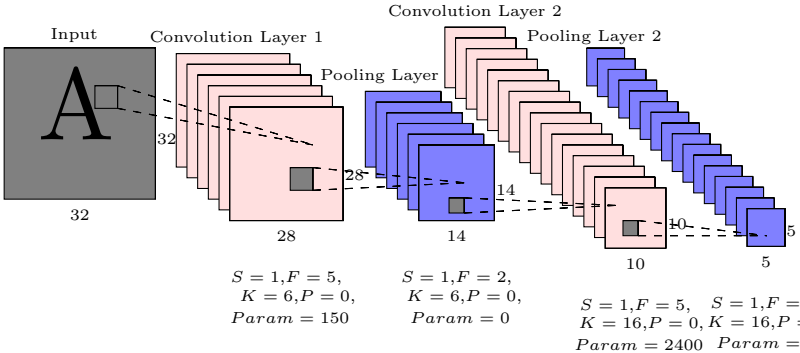
LeNet-5 for handwritten character recognition



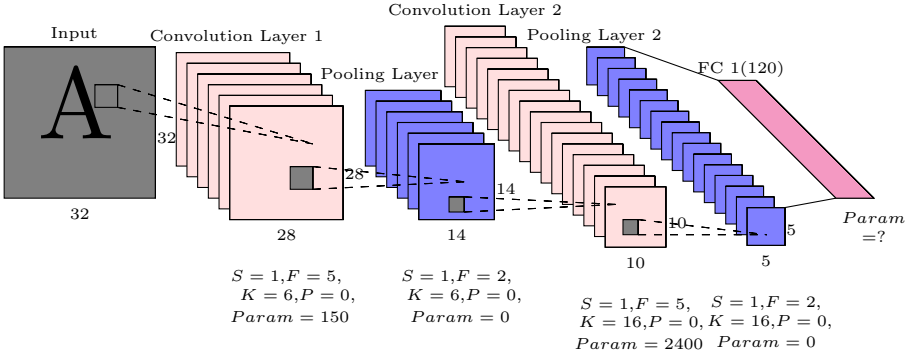
LeNet-5 for handwritten character recognition



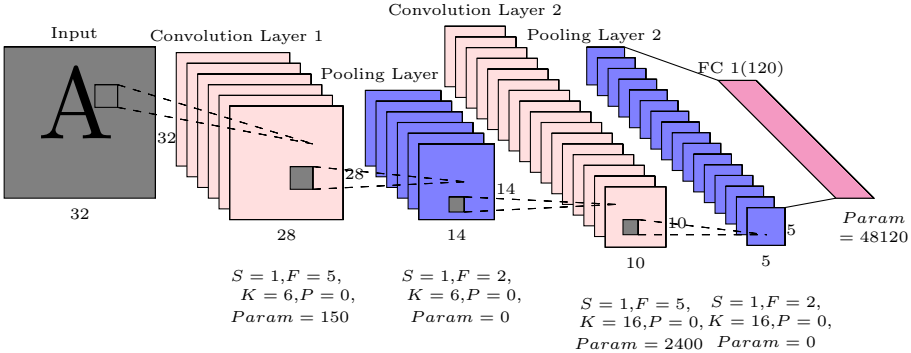
LeNet-5 for handwritten character recognition



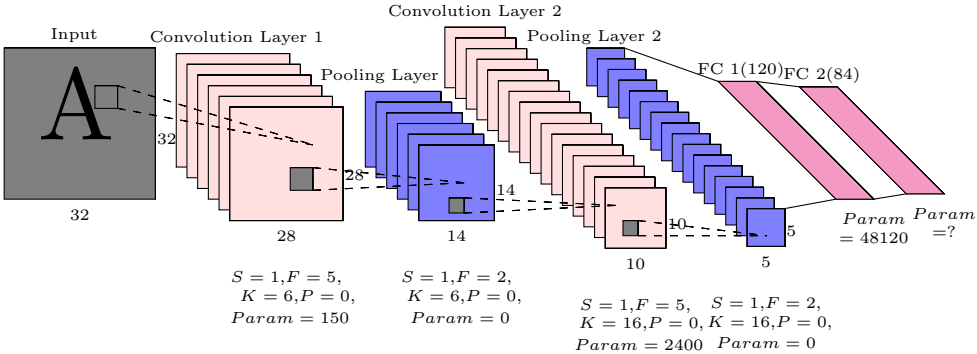
LeNet-5 for handwritten character recognition



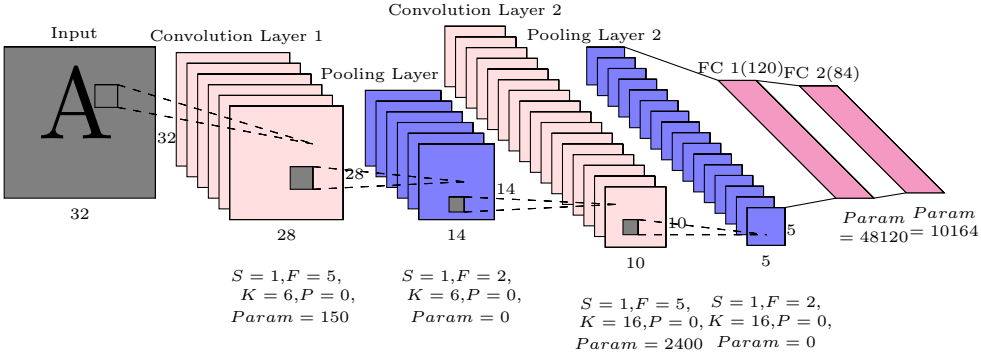
LeNet-5 for handwritten character recognition



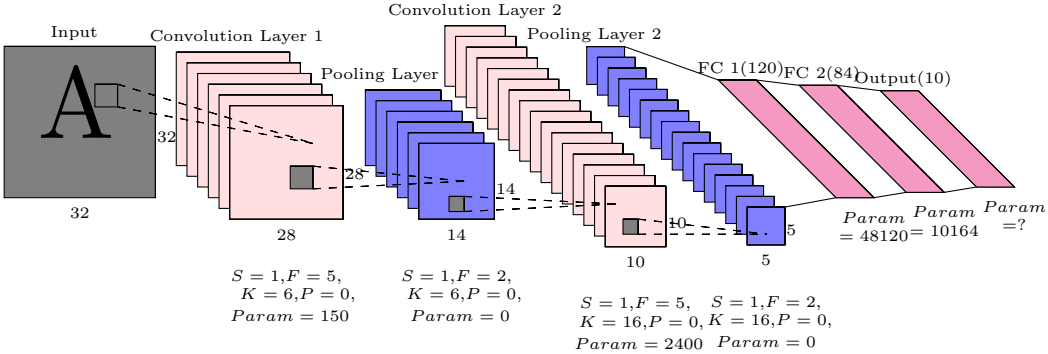
LeNet-5 for handwritten character recognition



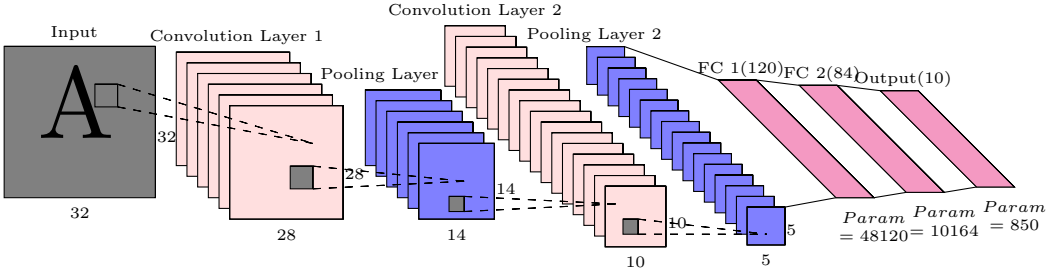
LeNet-5 for handwritten character recognition



LeNet-5 for handwritten character recognition



LeNet-5 for handwritten character recognition



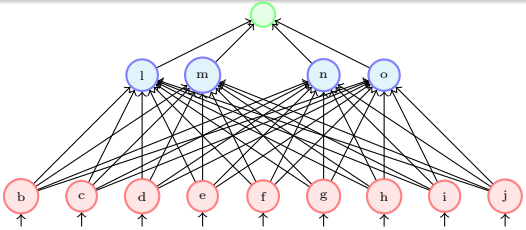
- How do we train a convolutional neural network ?

Input

b	c	d
e	f	g
h	i	j

Kernel

w	x
y	z



- A CNN can be implemented as a feedforward neural network

Input

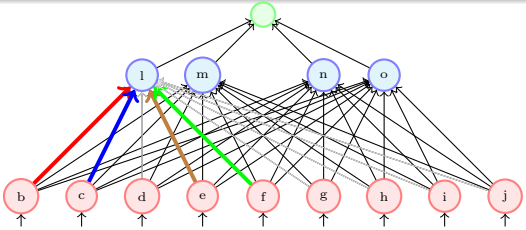
b	c	d
e	f	g
h	i	j

Kernel

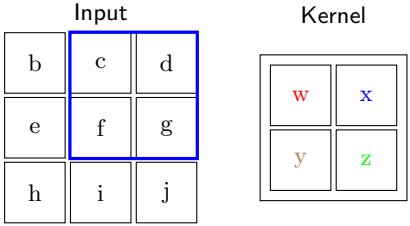
w	x
y	z

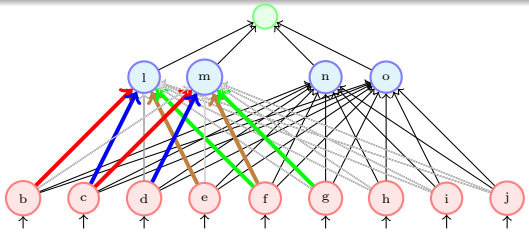
Output

ℓ	

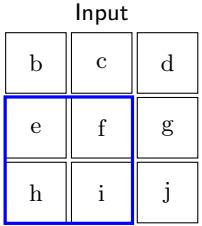


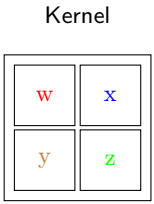
- A CNN can be implemented as a feedforward neural network
- wherein only a few weights(in color) are active

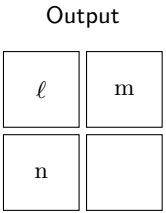


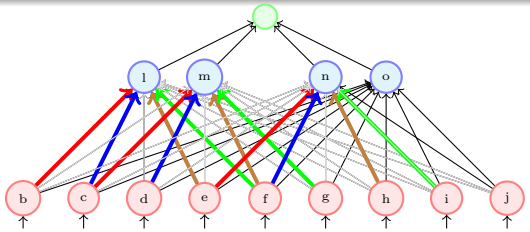


- A CNN can be implemented as a feedforward neural network
- wherein only a few weights(in color) are active
- the rest of the weights (in gray) are zero









- A CNN can be implemented as a feedforward neural network
- wherein only a few weights(in color) are active
- the rest of the weights (in gray) are zero

Input

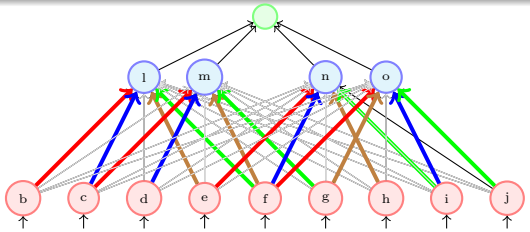
b	c	d
e	f	g
h	i	j

Kernel

w	x
y	z

Output

ℓ	m
n	o



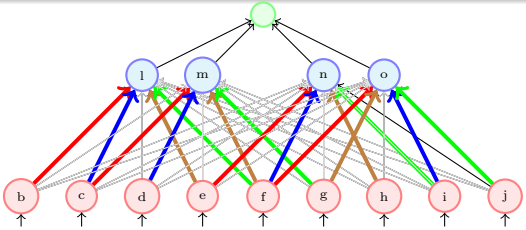
- A CNN can be implemented as a feedforward neural network
- wherein only a few weights(in color) are active
- the rest of the weights (in gray) are zero

Input

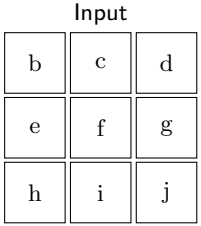
b	c	d
e	f	g
h	i	j

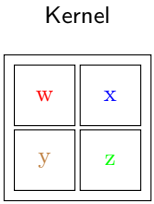
Kernel

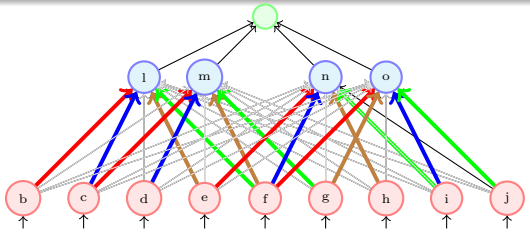
w	x
y	z



- A CNN can be implemented as a feedforward neural network
- wherein only a few weights(in color) are active
- the rest of the weights (in gray) are zero







- A CNN can be implemented as a feedforward neural network
- wherein only a few weights(in color) are active
- the rest of the weights (in gray) are zero

- We can thus train a convolution neural network using backpropagation by thinking of it as a feedforward neural network with sparse connections

Module 11.4 : CNNs (success stories on ImageNet)

ImageNet Success Stories(roadmap for rest of the talk)

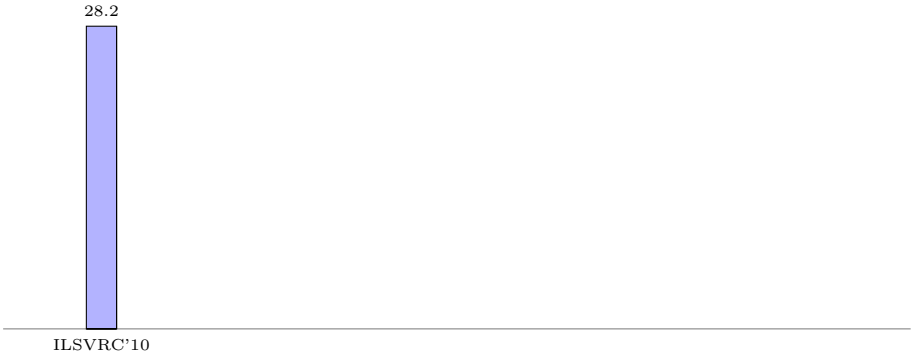
- AlexNet

ImageNet Success Stories(roadmap for rest of the talk)

- AlexNet
- ZFNet

ImageNet Success Stories(roadmap for rest of the talk)

- AlexNet
- ZFNet
- VGGNet


28.2
ILSVRC'10


28.2
25.8
ILSVRC'10
ILSVRC'11