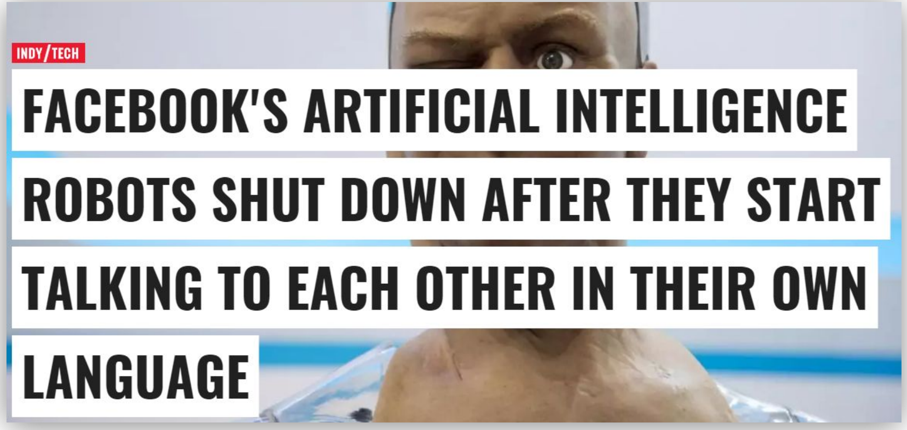
Plant the seed. *Harvest the potential.*

News & Headlines

Computers can now paint like Van Gogh and Picasso

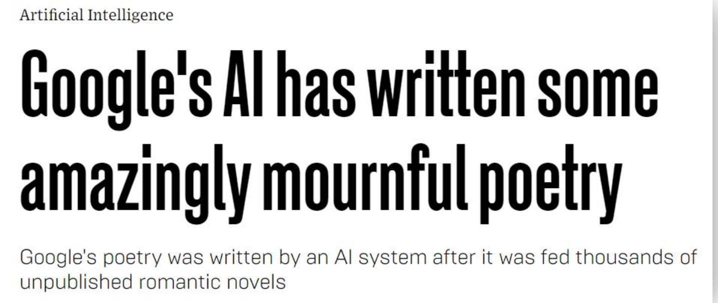


INDY/TECH

FACEBOOK'S ARTIFICIAL INTELLIGENCE ROBOTS SHUT DOWN AFTER THEY START TALKING TO EACH OTHER IN THEIR OWN LANGUAGE

China seeks dominance of global AI industry

Beijing challenges US with plan to create $150bn artificial intelligence sector



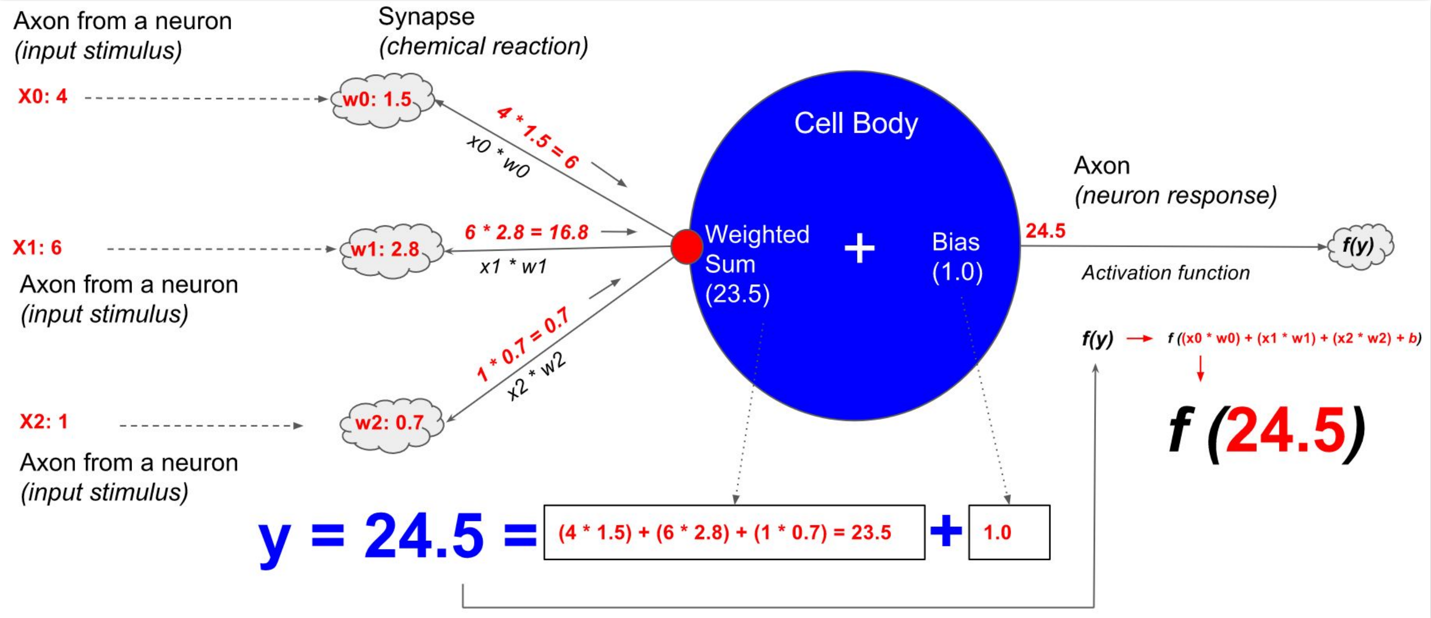
Artificial Intelligence

Google's AI has written some amazingly mournful poetry

Google's poetry was written by an AI system after it was fed thousands of unpublished romantic novels

Where we are...

Last time we examined Multi-Layered Perceptrons...



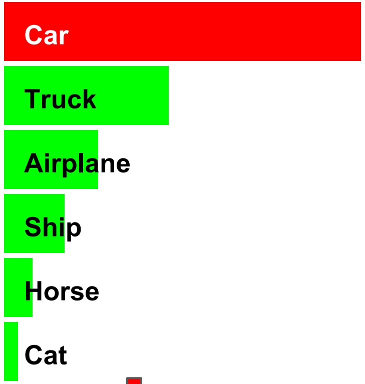
What is this?

“Tell me what you see”

Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 640 x 480
Depth: 3 (Red, Green, Blue)

Neural Network
(black box)



“This is a car”

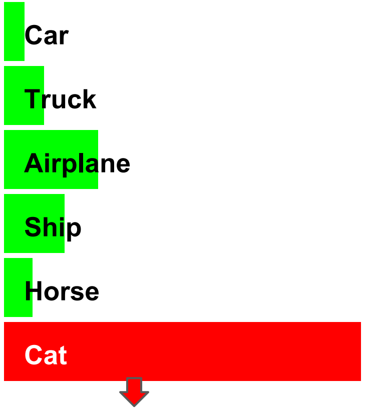
What is this?

“Tell me what you see”

Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 640 x 480
Depth: 3 (Red, Green, Blue)

Neural Network (black box)



“This is a cat”

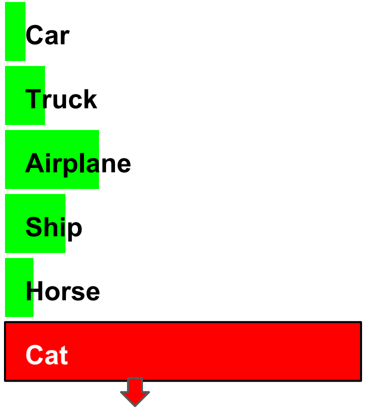
What is this?

"Tell me what you see"

Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 640 x 480
Depth: 3 (Red, Green, Blue)

Neural Network
(black box)



"This is a cat"

What is this?

"Tell me what you see"

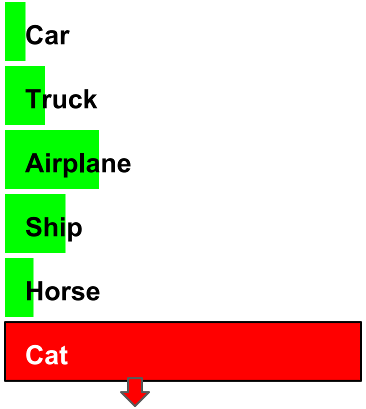
Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 640 x 480
Depth: 3 (Red, Green, Blue)

→

Neural Network
(black box)

→



"This is a cat"

Classification Error

What is this?

My first shot: “I will try with a multi-layered perceptron…”

Assume the hidden layer is **2.8 times** bigger than the input layer.

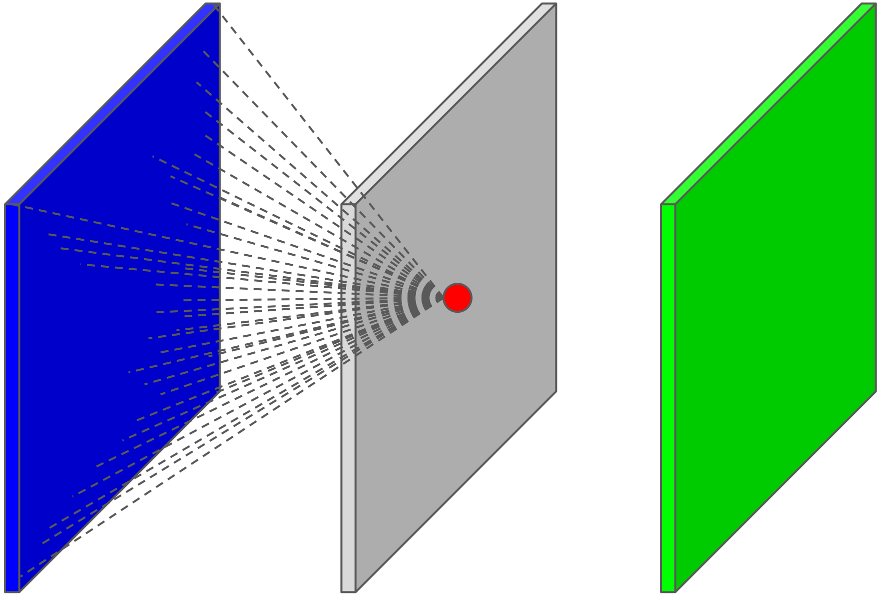
Input Signal

Neural Network

Prediction

Dimensions: 640 x 480
Depth: 3 (Red, Green, Blue)

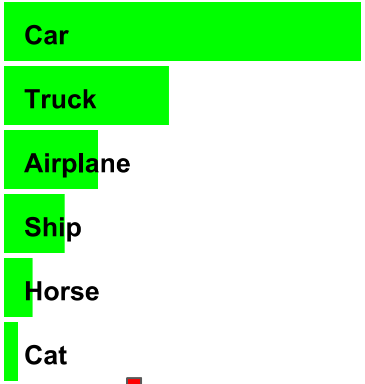
640 x 480 x 3



Input Layer
(921k neurons)

Hidden Layer
(2.5 million neurons)

Output Layer



“This is a car”

What is this?

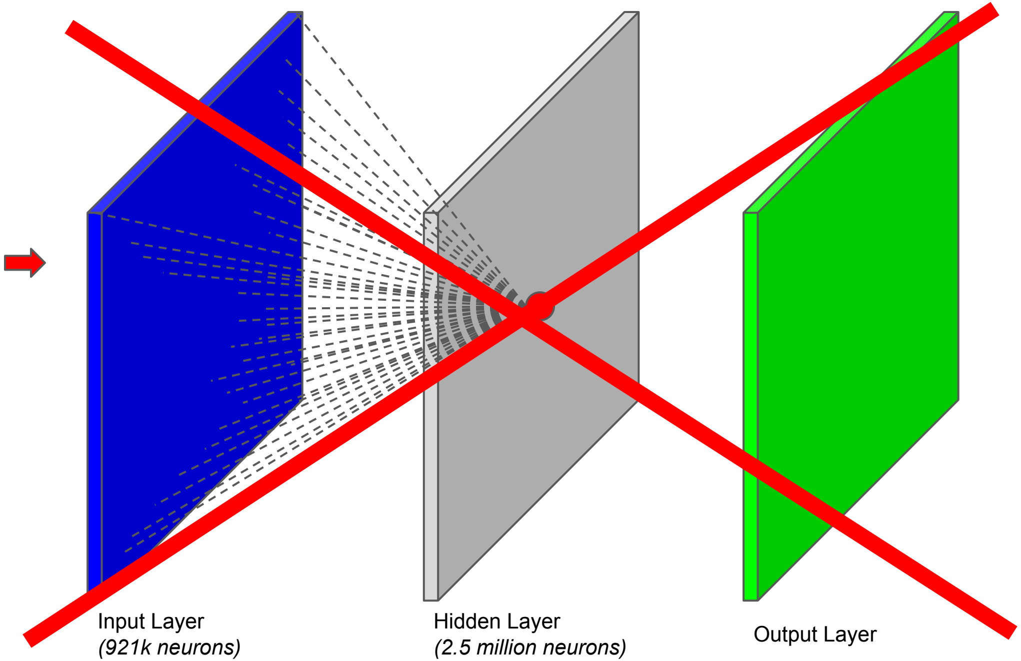
My first shot: "I will try with a multi-layered perceptron…"

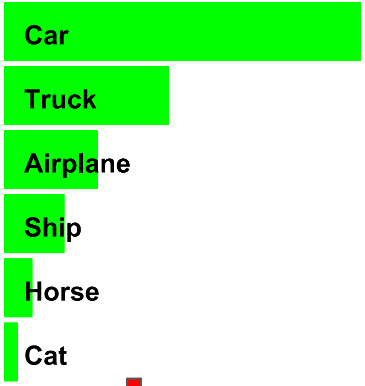
Assume the hidden layer is **2.8 times** bigger than the input layer.

Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 640 x 480
Depth: 3 (Red, Green, Blue)

A minimum of 2.3 Trillion parameters!





"This is a car"

This won't work. Too many parameters!

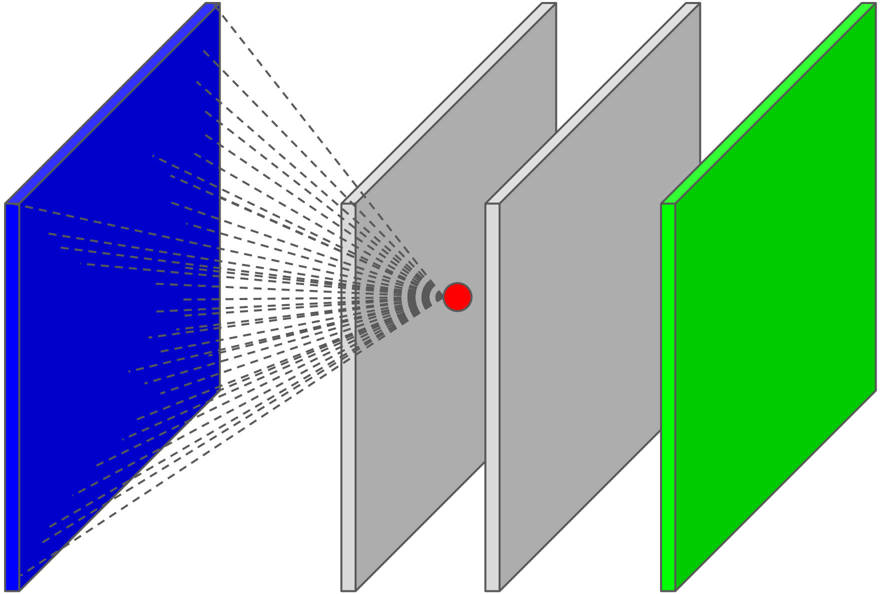
What is this?

Let's try again → Second shot → "Reduce the input size and add more layers"

Assume the hidden layer is **2.8 times** bigger than the input layer.

Input Signal → ***Neural Network*** → ***Prediction***

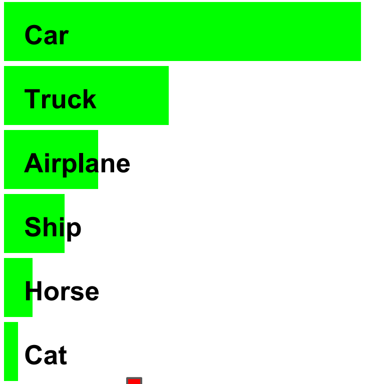
Dimensions: 160 x 120
Depth: 3 (Red, Green, Blue)



Input Layer
(57.6k neurons)

First Hidden Layer
(161k neurons)

Output Layer



"This is a car"

What is this?

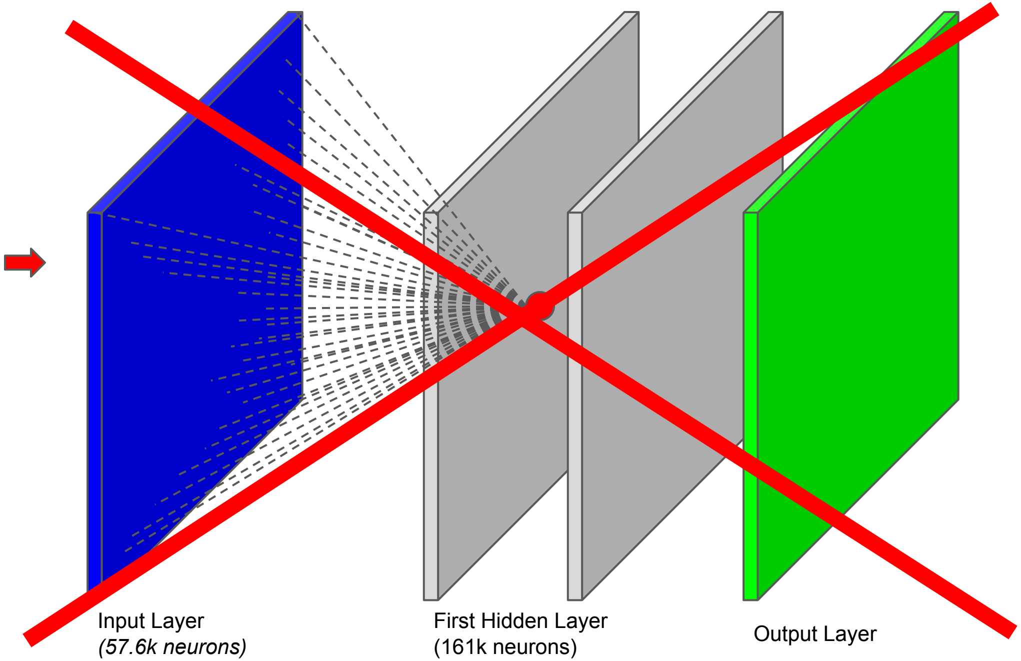
Let's try again → Second shot → "Reduce the input size and add more layers"

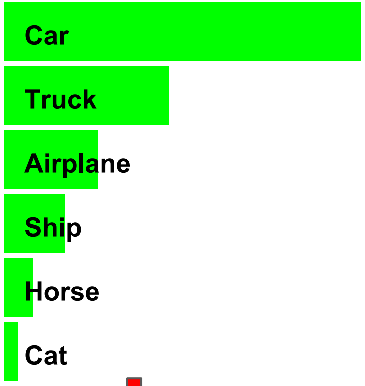
Assume the hidden layer is **2.8 times** bigger than the input layer.

Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 160 x 120
Depth: 3 (Red, Green, Blue)

A minimum of 9.2 Billion parameters!





"This is a car"

This won't work either. Too many parameters!

What is this?

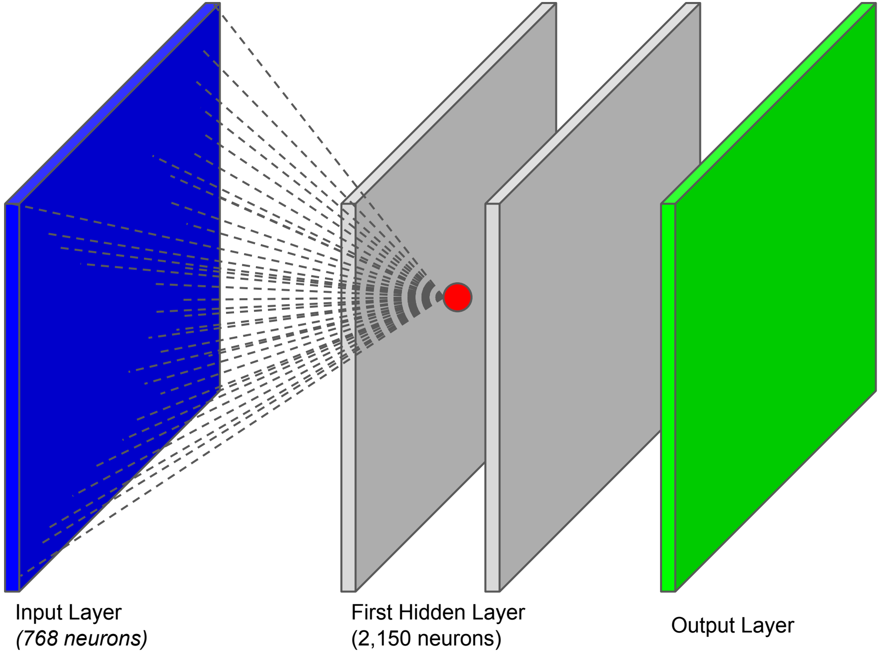
One more time... → Third shot → "Reduce the input size again"

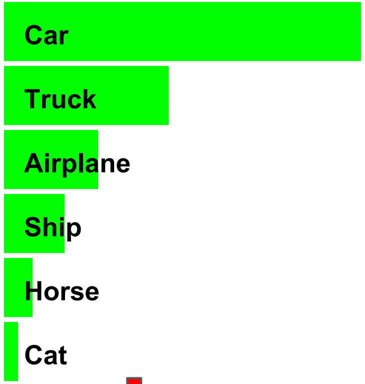
Assume the hidden layer is **2.8 times** bigger than the input layer.

Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 16 x 16
Depth: 3 (Red, Green, Blue)

The size of an icon





"This is a car"

What is this?

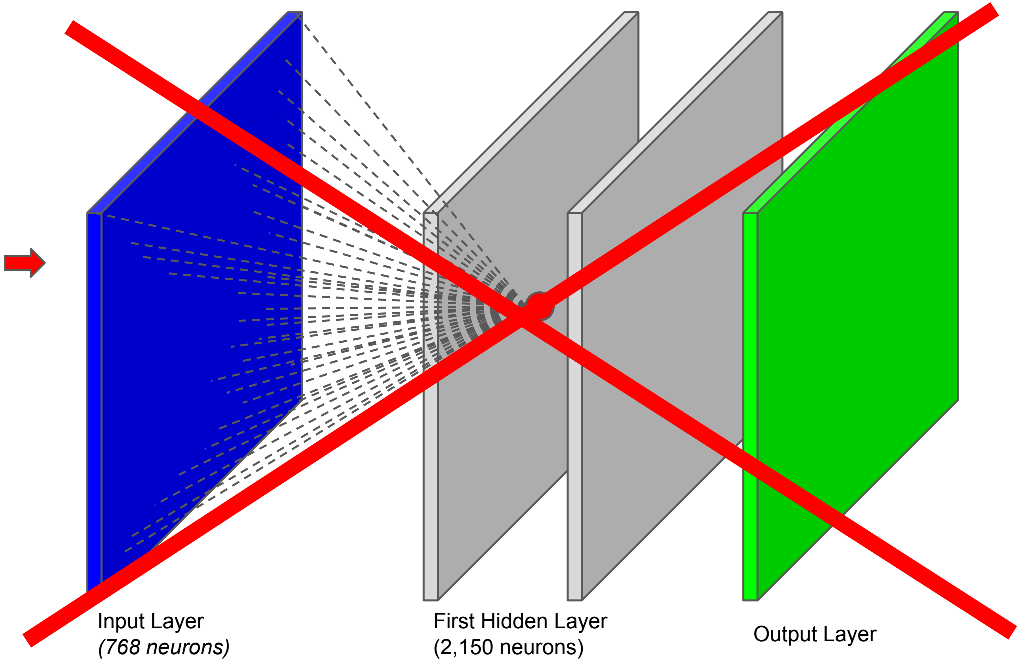
One more time... → Third shot → "Reduce the input size again"

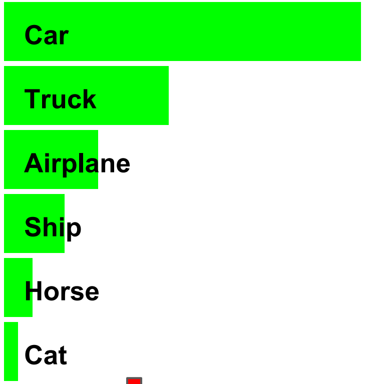
Assume the hidden layer is **2.8 times** bigger than the input layer.

Input Signal → ***Neural Network*** → ***Prediction***

Dimensions: 16 x 16
Depth: 3 (Red, Green, Blue)

A minimum of 1.6 Million parameters!





"This is a car"

Still doesn't work. Too many parameters!

Too many parameters

So I can't solve this problem with an ordinary Multi-Layered Perceptron...

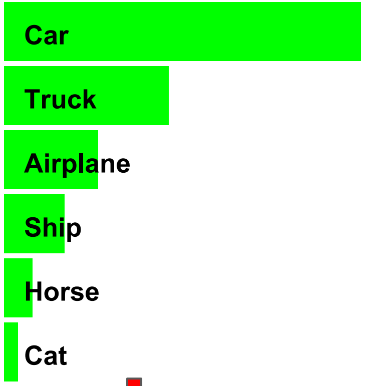
Input Signal → ***Neural Network*** → ***Prediction***

Multi-Layered Perceptron:

1) ***Too many parameters ("weights").***
2) ***Lack of representational power, in this context.***
3) ***Best case scenario their accuracy is extremely poor.***

...and this is just for starters.

It's just not the right tool for the job.

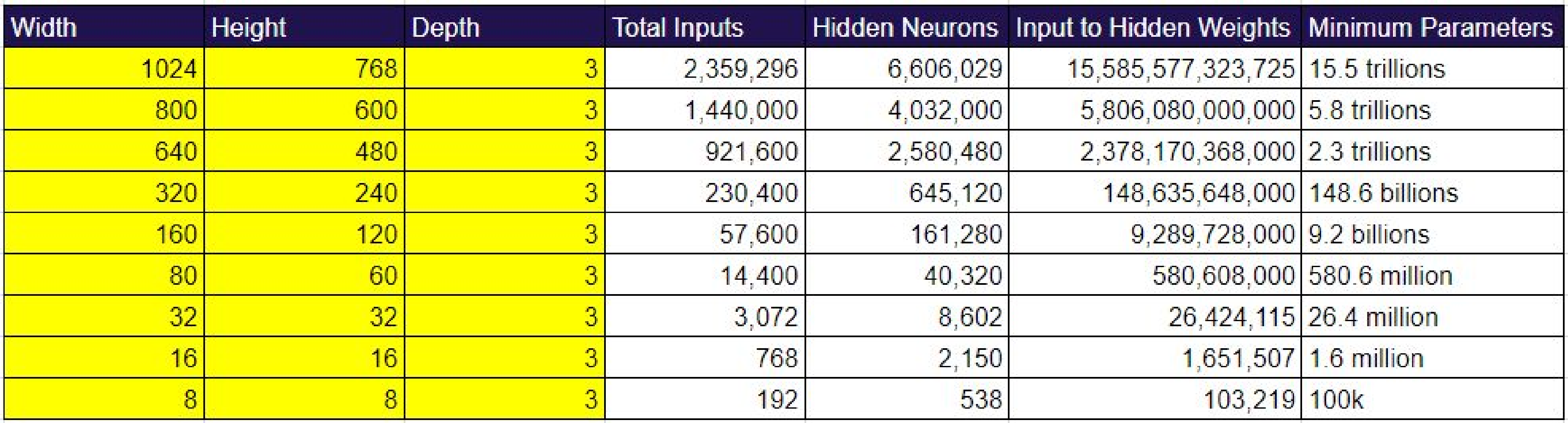


"This is a car"

Too many parameters

Perceptrons are not the right tool for the job. The number of parameters is a killer.

Too many parameters.



Width	Height	Depth	Total Inputs	Hidden Neurons	Input to Hidden Weights	Minimum Parameters
1024	768	3	2,359,296	6,606,029	15,585,577,323,725	15.5 trillions
800	600	3	1,440,000	4,032,000	5,806,080,000,000	5.8 trillions
640	480	3	921,600	2,580,480	2,378,170,368,000	2.3 trillions
320	240	3	230,400	645,120	148,635,648,000	148.6 billions
160	120	3	57,600	161,280	9,289,728,000	9.2 billions
80	60	3	14,400	40,320	580,608,000	580.6 million
32	32	3	3,072	8,602	26,424,115	26.4 million
16	16	3	768	2,150	1,651,507	1.6 million
8	8	3	192	538	103,219	100k

Assume the hidden layer is 2.8 times bigger than the input layer.

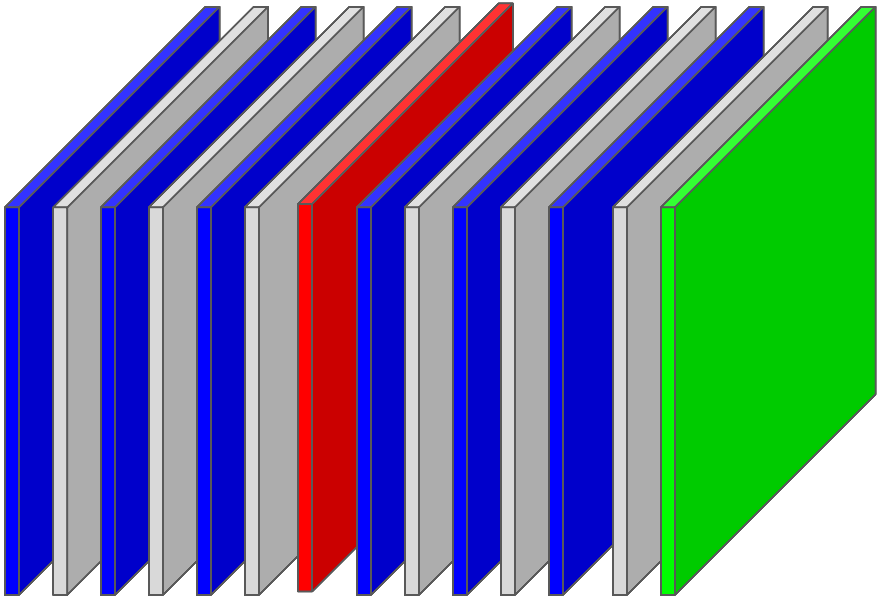
Convolutional Neural Networks

A *Convolutional Neural Network* is a sequence of convolutional layers, interspersed with activation functions.

They use **weight-sharing** in order to overcome the issue with parameters. It doesn't make sense to learn the same thing across all spatial locations.

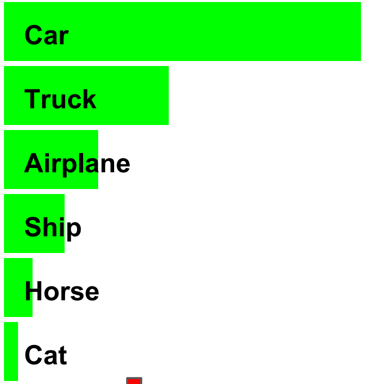
Input Signal

Neural Network



Depth of the Network

Prediction



"This is a car"

"This network is 14 levels deep"

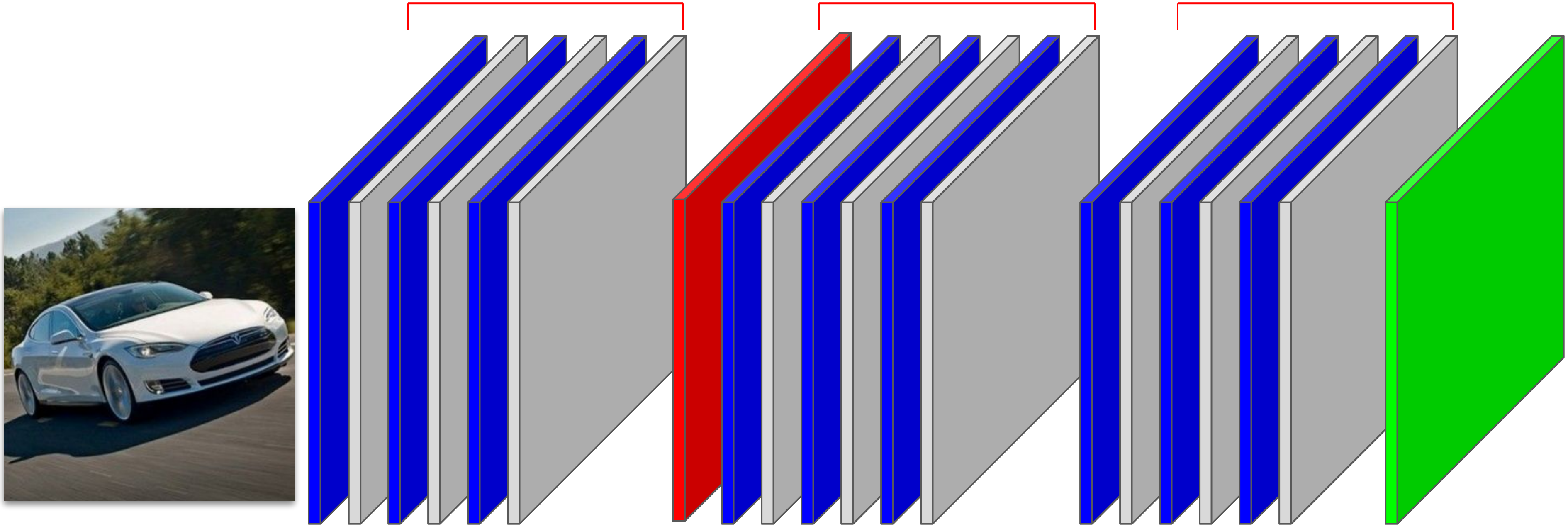
🟦 Conv ⬜ ReLU

🟥 Pooling 🟩 Fully Connected

Convolutional Neural Networks

From low-level details to high-level objects...

Input image → *Low-level features* → *Mid-level features* → *High-level features* → *Classifier*



Basic patterns (lines, edges, corners)

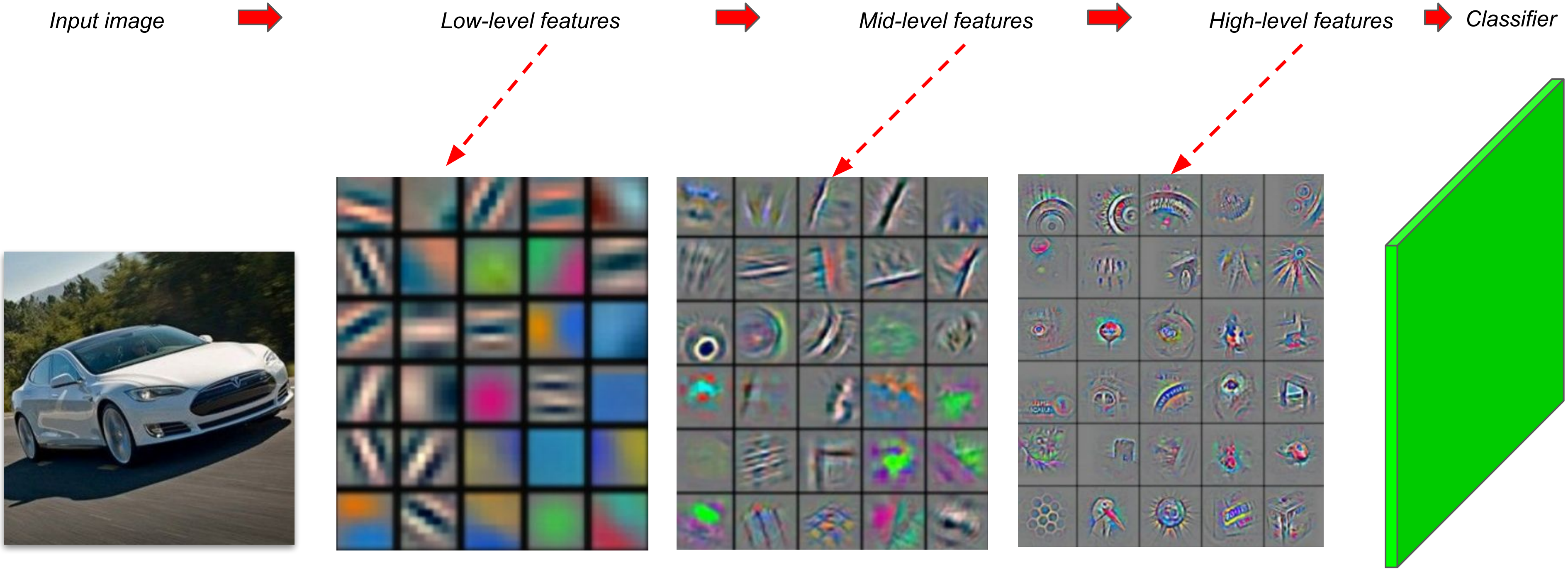
Irregular shapes (object parts)

Object prototypes (eyes, faces, cars, etc)

“This is a car”

Convolutional Neural Networks

From low-level details to high-level objects...



Input image

Low-level features

Mid-level features

High-level features

Classifier

Basic patterns (lines, edges, corners)

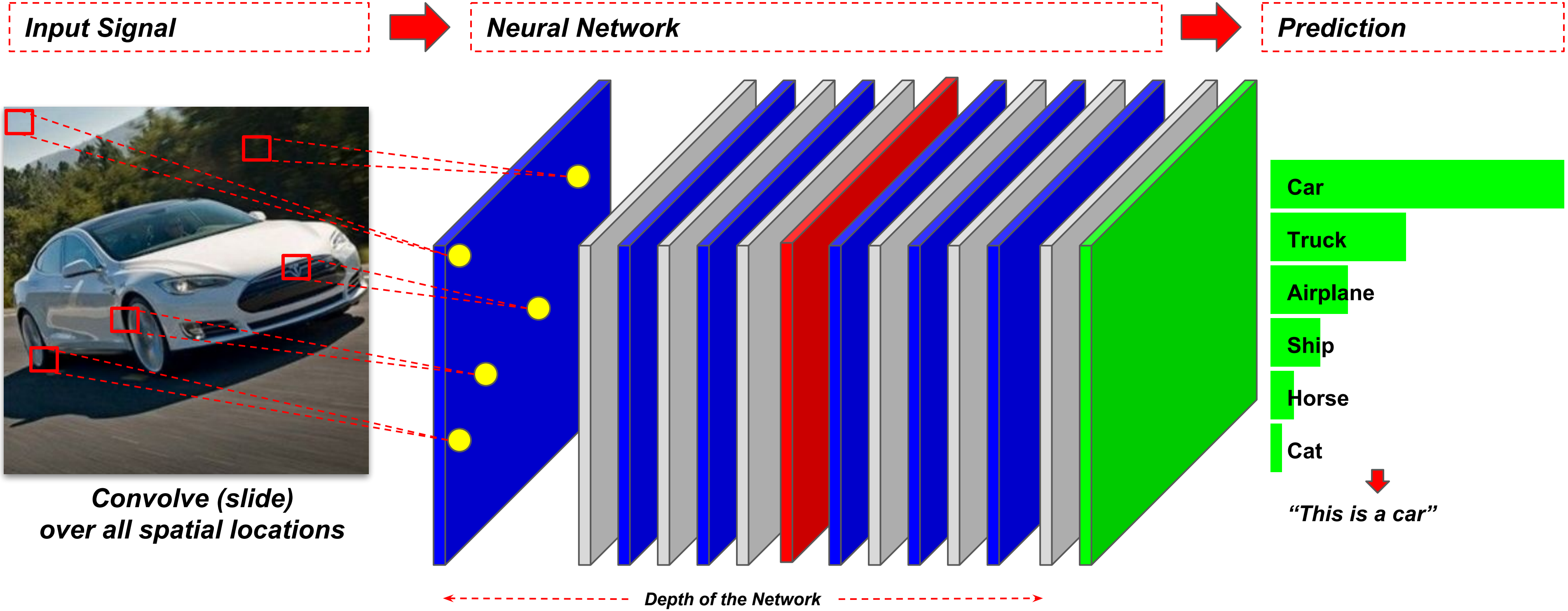
Irregular shapes (object parts)

Object prototypes (eyes, faces, cars, etc)

“This is a car”

Convolutional Neural Networks

Each neuron "*looks*" at a small portion of the input → *(Region Of Interest, or ROI for short)*
The ROI has a size *(could be 1x1, 3x3, 5x5, 7x7, etc.)* → This is the ***receptive field of the neuron***.



Input Signal

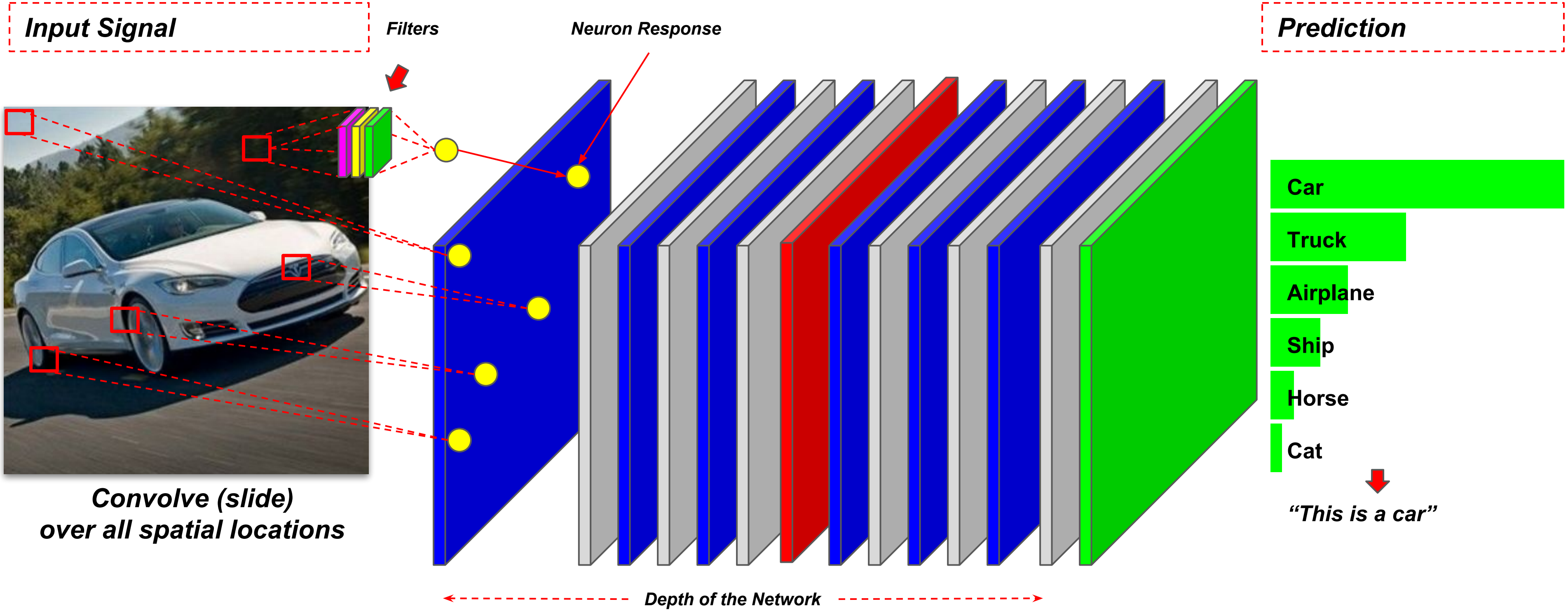
Neural Network

Prediction

Convolve (slide)
over all spatial locations

Convolutional Neural Networks

The network will perform convolutions between an input patch and a set of filters



Input Signal

Filters

Neuron Response

Prediction

Car

Truck

Airplane

Ship

Horse

Cat

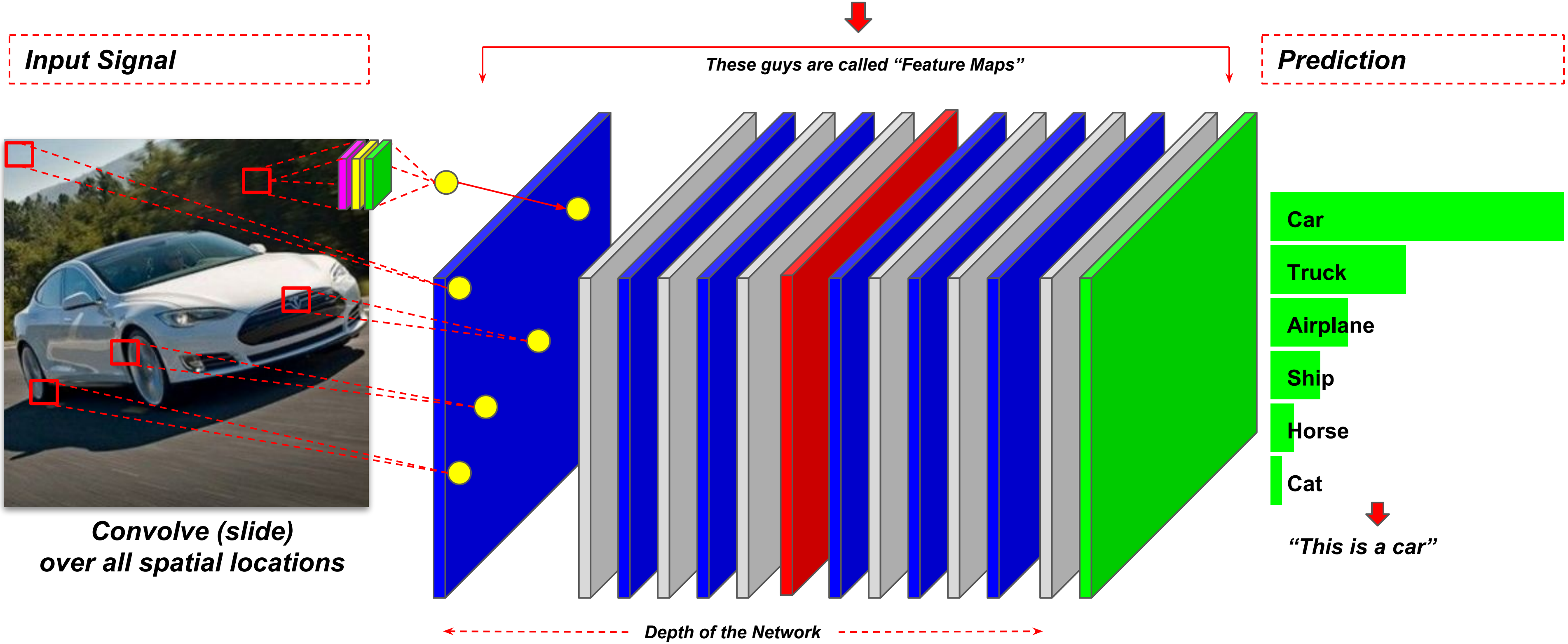
"This is a car"

Convolve (slide)
over all spatial locations

Depth of the Network

Convolutional Neural Networks

So to identify a car, the network will need to learn the *"weights"* required to produce each of the feature maps below.

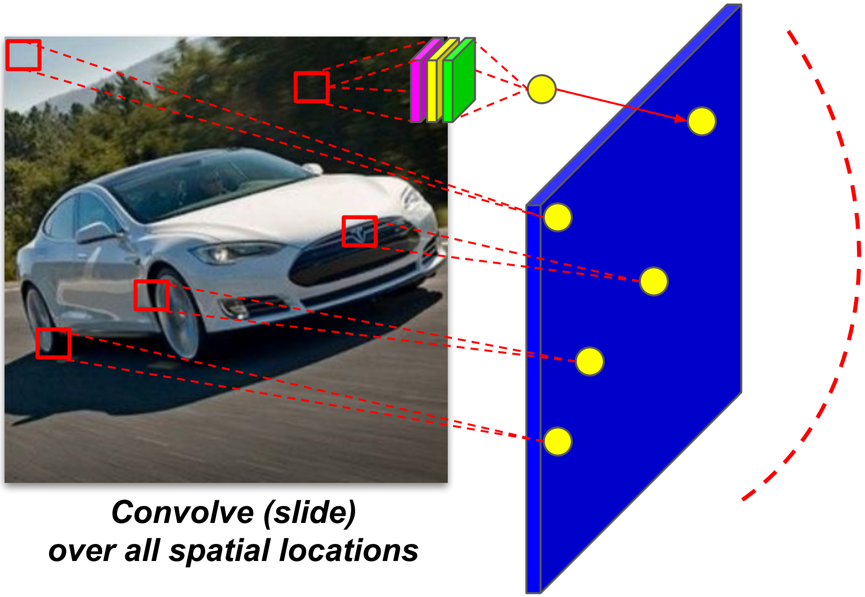


Convolutional Neural Networks

So to identify a car, the network will need to learn the *“weights”* required to produce each of the feature maps below.

Input Signal

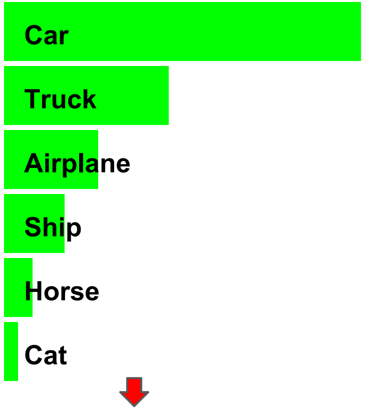
Prediction



Convolve (slide) over all spatial locations

All of the neurons in the same layer share their weights.

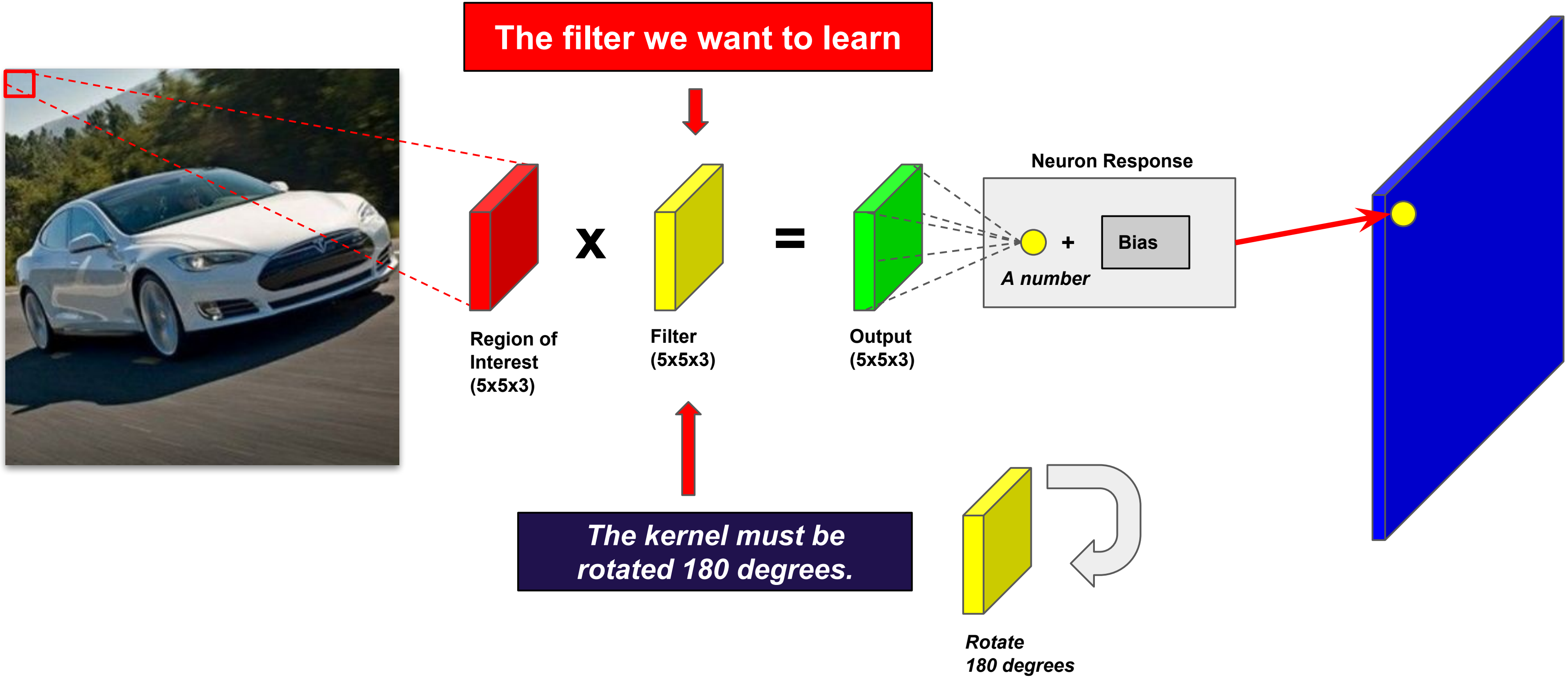
This is called “*Weight sharing*”



“This is a car”

Convolutional Neural Networks

Convolve (slide) over all spatial locations.



The filter we want to learn

Region of Interest (5x5x3)

x

Filter (5x5x3)

=

Output (5x5x3)

Neuron Response

+

Bias

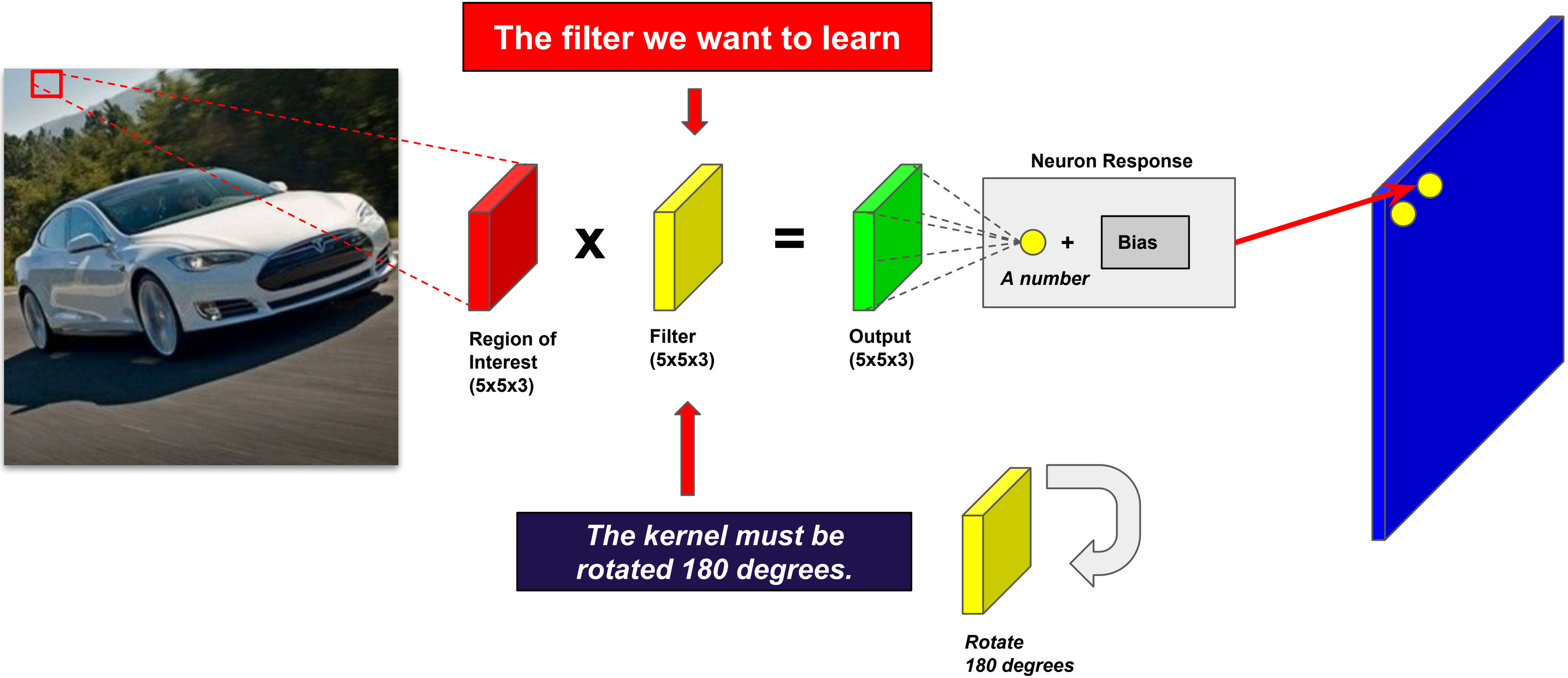
A number

The kernel must be rotated 180 degrees.

Rotate 180 degrees

Convolutional Neural Networks

Convolve (slide) over all spatial locations.



The filter we want to learn

Region of Interest (5x5x3)

x

Filter (5x5x3)

=

Output (5x5x3)

Neuron Response

+ Bias

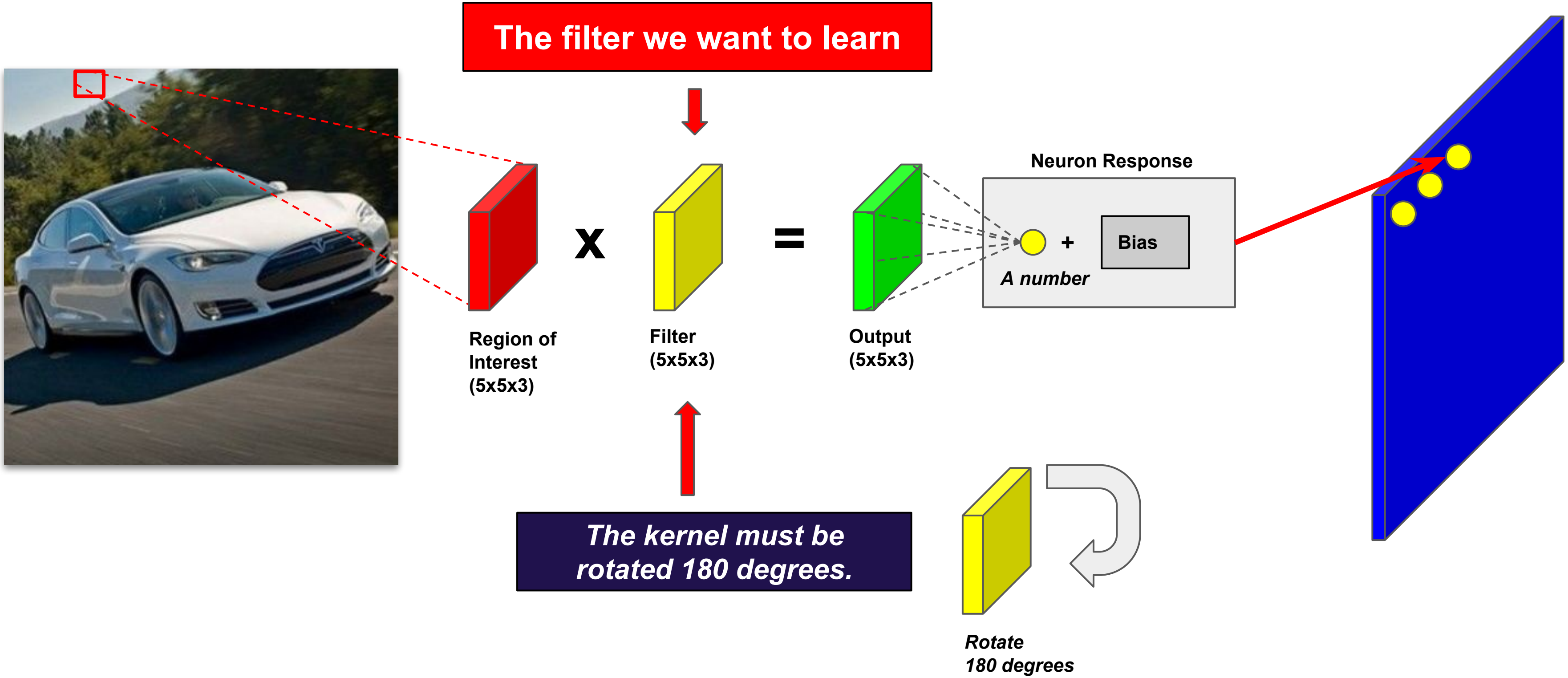
A number

The kernel must be rotated 180 degrees.

Rotate 180 degrees

Convolutional Neural Networks

Convolve (slide) over all spatial locations.



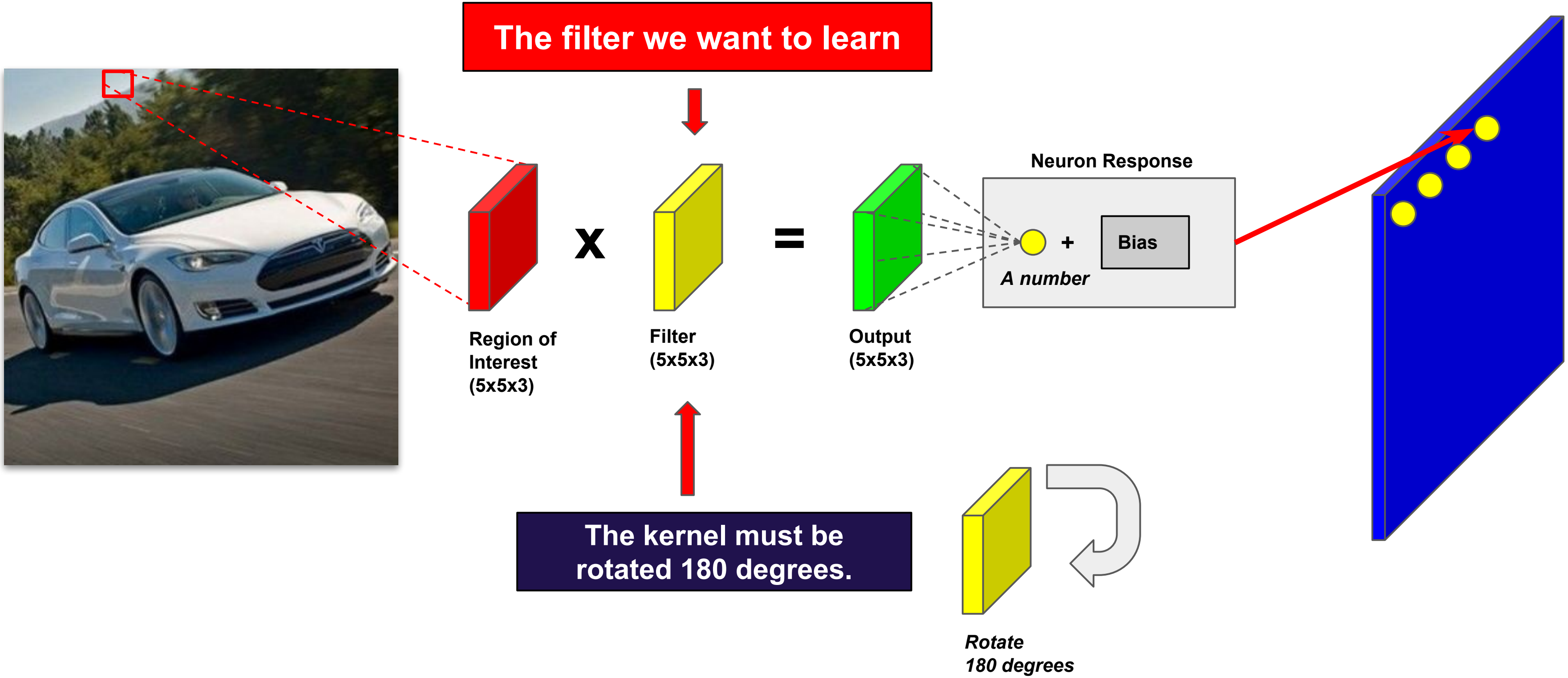
The filter we want to learn

The kernel must be rotated 180 degrees.

Rotate 180 degrees

Convolutional Neural Networks

Convolve (slide) over all spatial locations.



The filter we want to learn

Region of Interest (5x5x3)

x

Filter (5x5x3)

=

Output (5x5x3)

Neuron Response

+ Bias

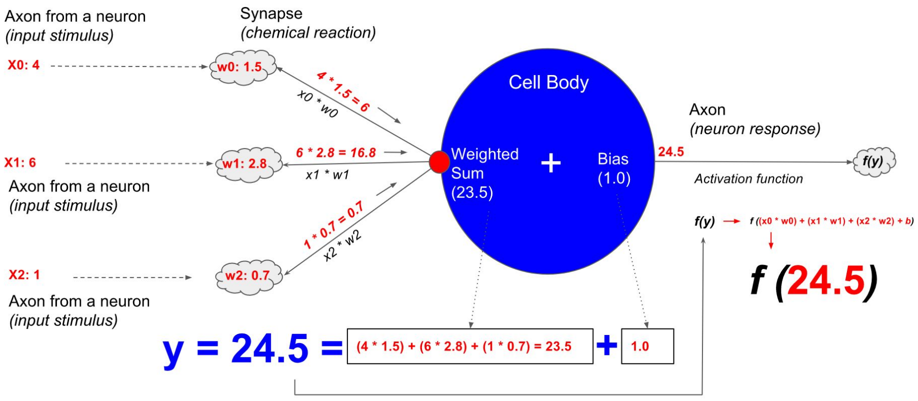
A number

The kernel must be rotated 180 degrees.

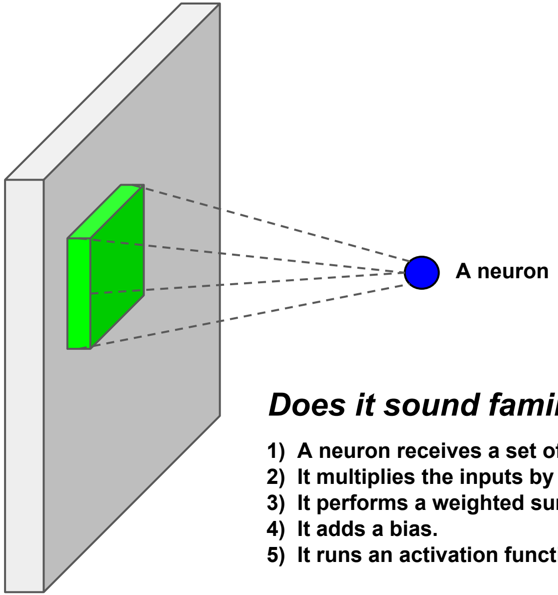
Rotate 180 degrees

Convolutional Neural Networks

Before



Now



Does it sound familiar to you?

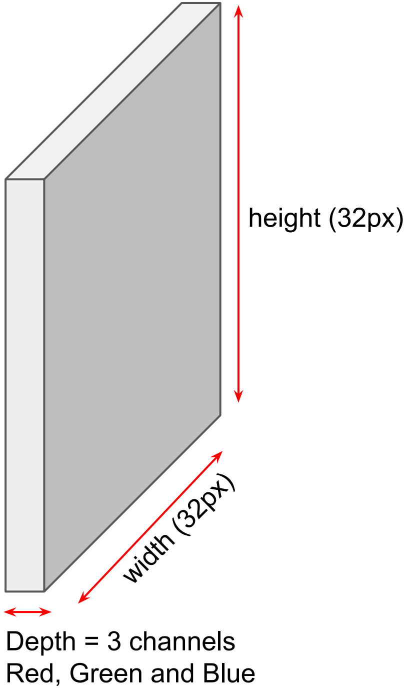
1) A neuron receives a set of inputs.
2) It multiplies the inputs by their respective weights.
3) It performs a weighted sum.
4) It adds a bias.
5) It runs an activation function.

Nothing changes!

Convolutional Layers

Say you've got an image. Let's pretend the image is in RGB format *(Red, Green, Blue)*

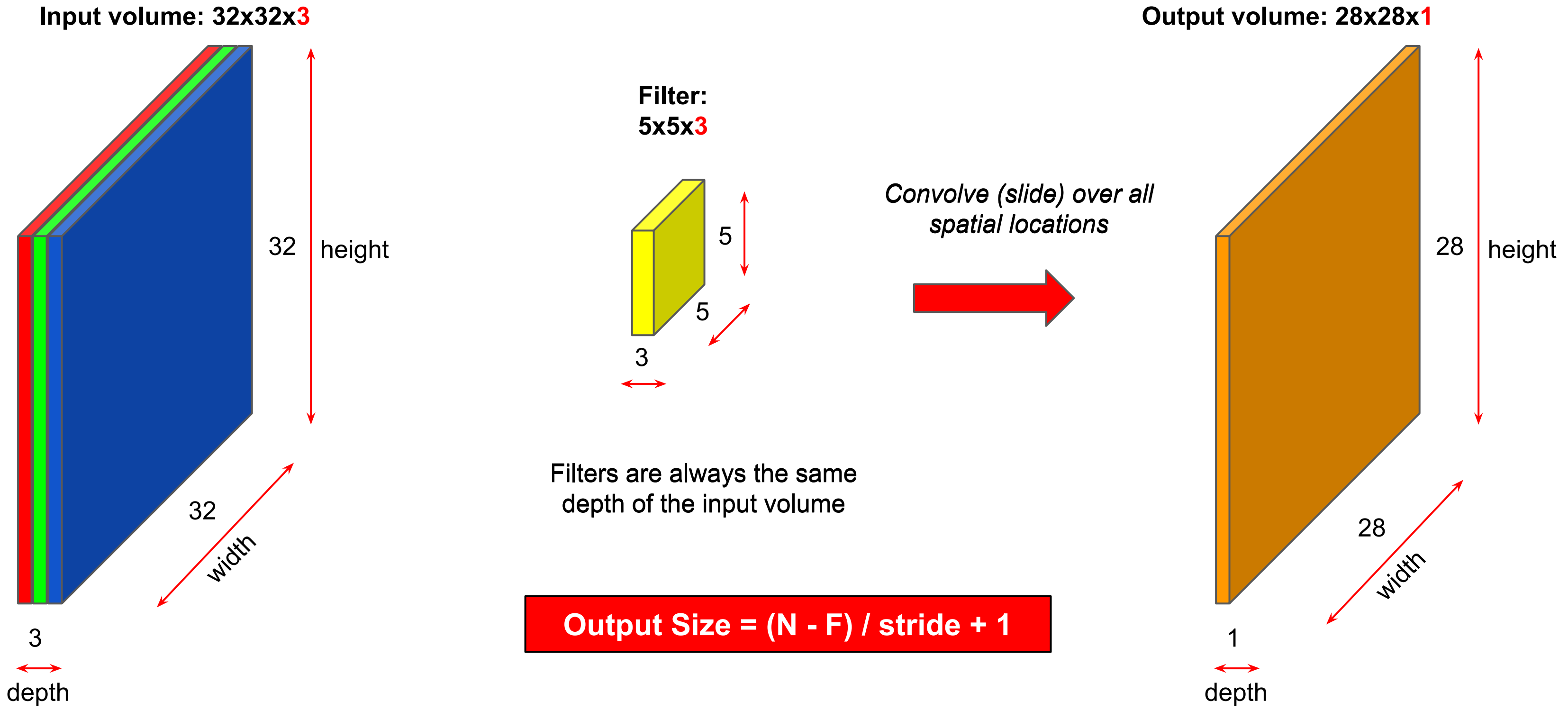
The image is 32 pixels width by 32 pixels height.



Original Image

Convolutional Layers

They produce *feature maps* by convolving an input volume with a bank of filters.

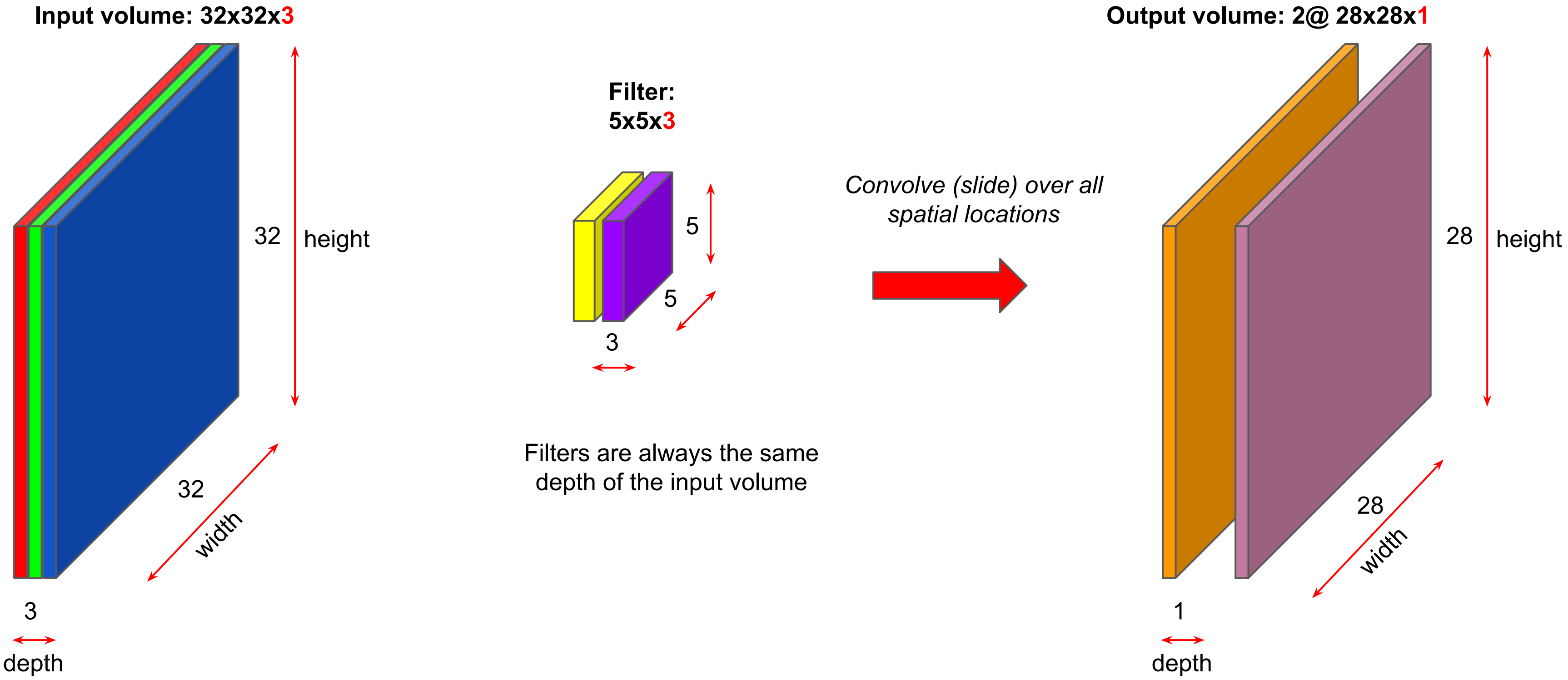


Filters are always the same depth of the input volume

Output Size = (N - F) / stride + 1

Convolutional Layers

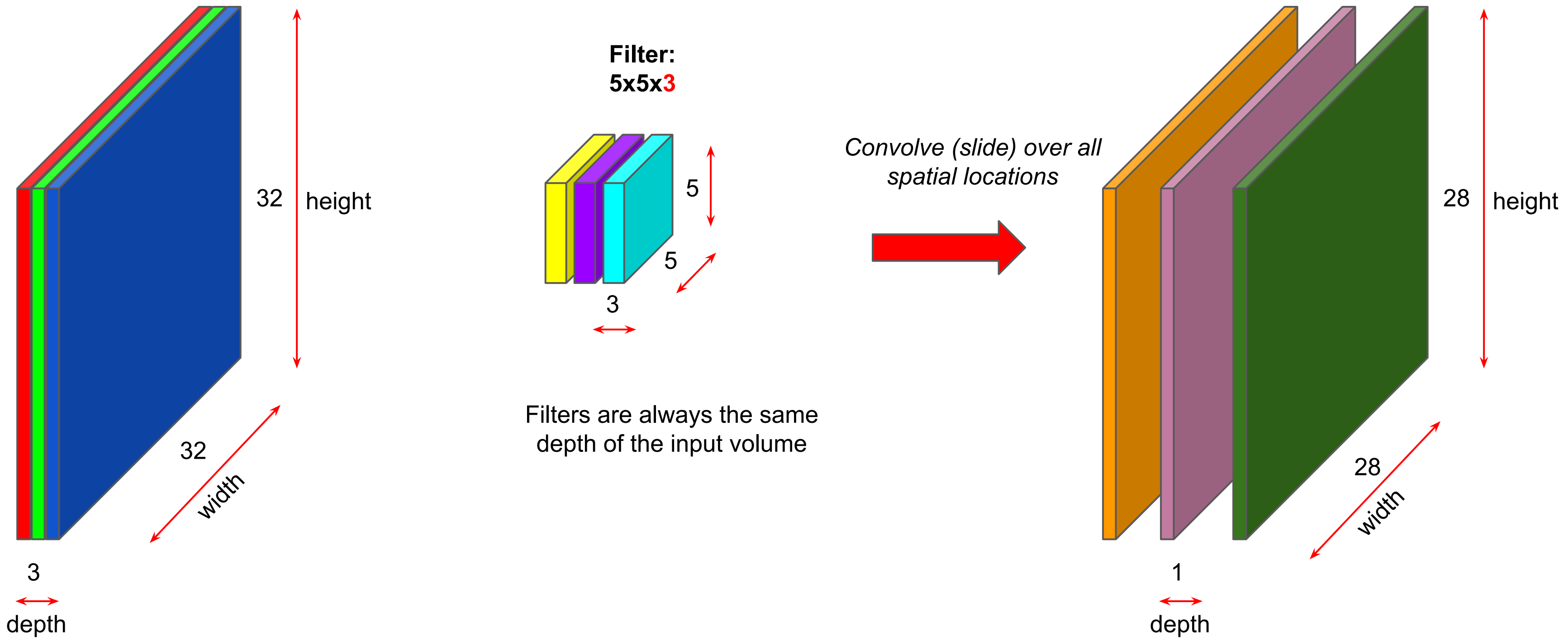
Convolving with a second filter will produce another feature map



Filters are always the same depth of the input volume

Convolutional Layers

If I add a third filter then I will get another feature map



Input volume: 32x32x3

32 height

32 width

3 depth

Filter:
5x5x3

5

5

3

Filters are always the same depth of the input volume

Convolve (slide) over all spatial locations

Output volume: 3@ 28x28x1

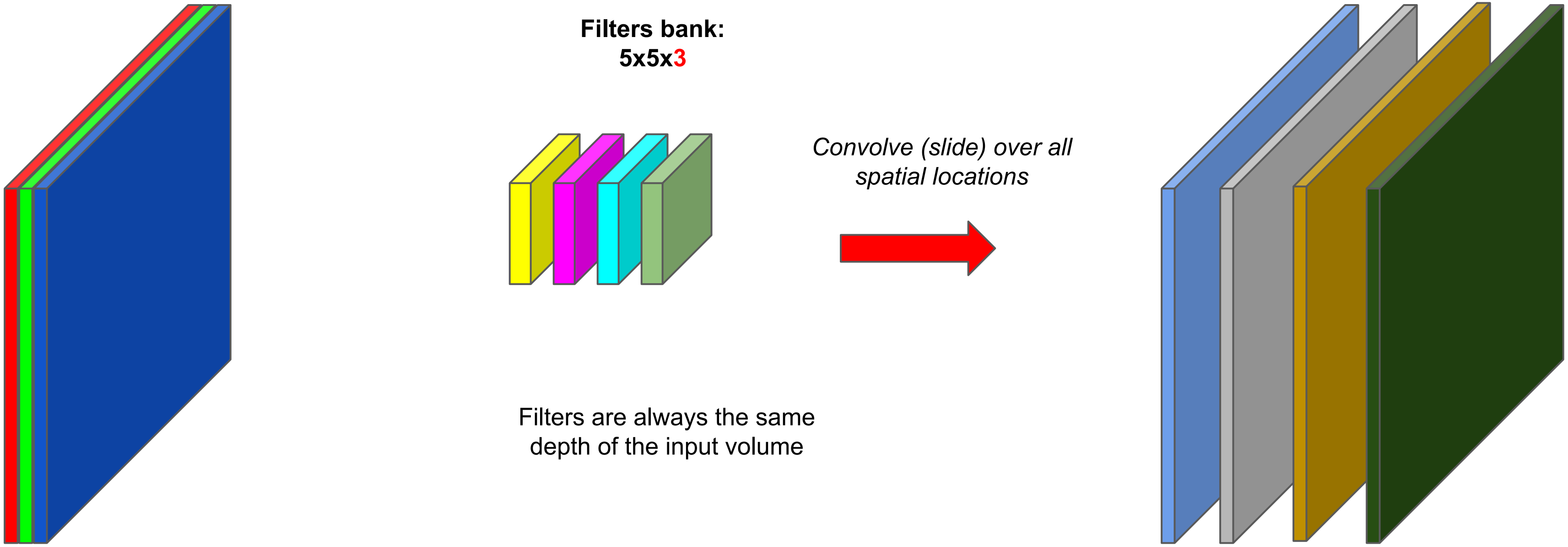
28 height

28 width

1 depth

Convolutional Layers

They produce *feature maps* by convolving an input volume with a bank of filters.



Input volume: 32x32x3

Filters bank:
5x5x3

Convolve (slide) over all spatial locations

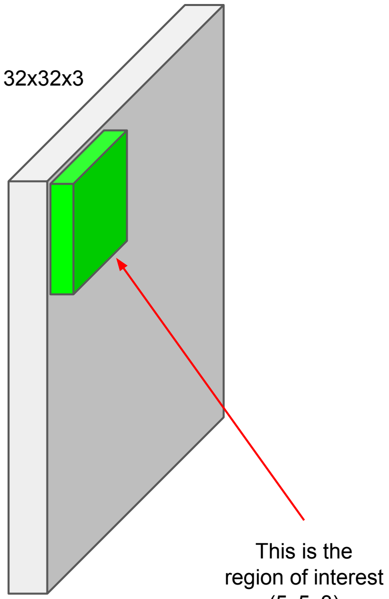
Output volume: 4@ 28x28x1

Filters are always the same depth of the input volume

Feature maps

Convolutional Layers

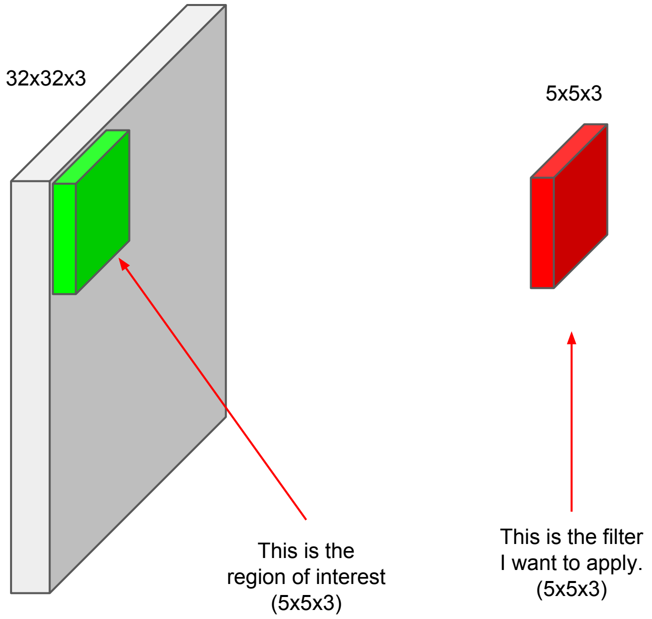
How it works.



This is the region of interest (5x5x3)

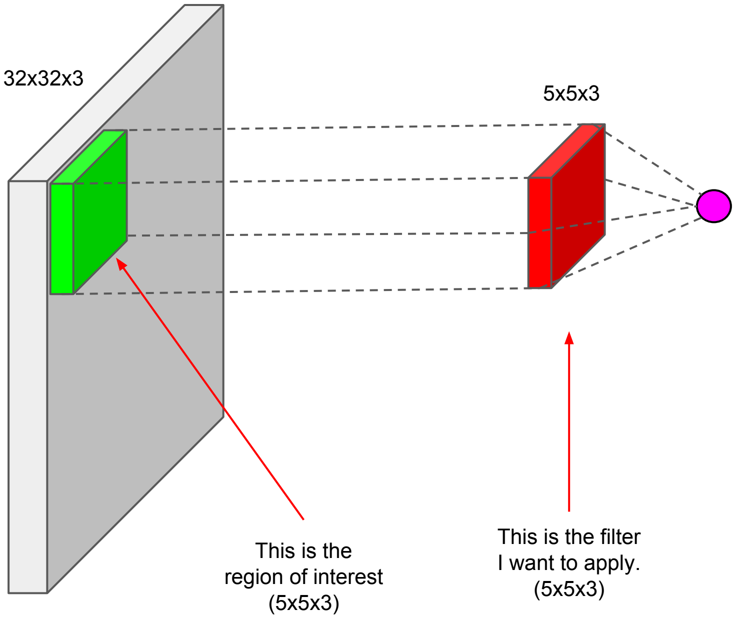
Convolutional Layers

How it works.


32x32x3
5x5x3
This is the region of interest (5x5x3)
This is the filter I want to apply. (5x5x3)

Convolutional Layers

How it works.



This is the region of interest (5x5x3)

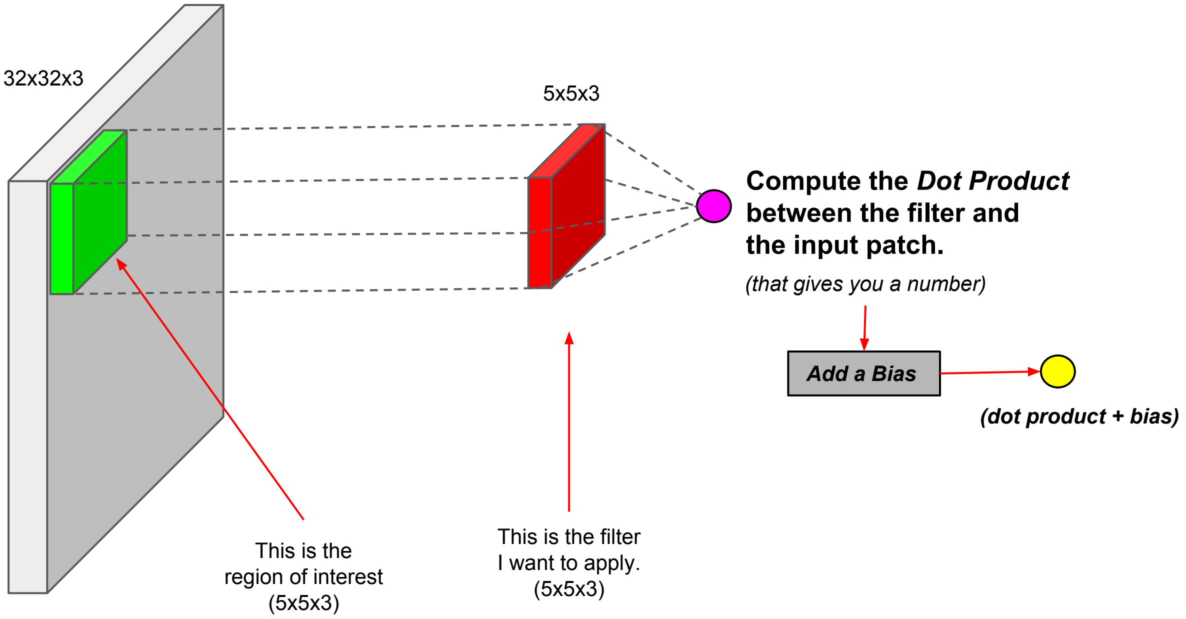
This is the filter I want to apply. (5x5x3)

Compute the *Dot Product* between the filter and the input patch.

(that gives you a number)

Convolutional Layers

How it works.



Compute the *Dot Product* between the filter and the input patch.

(that gives you a number)

Add a Bias

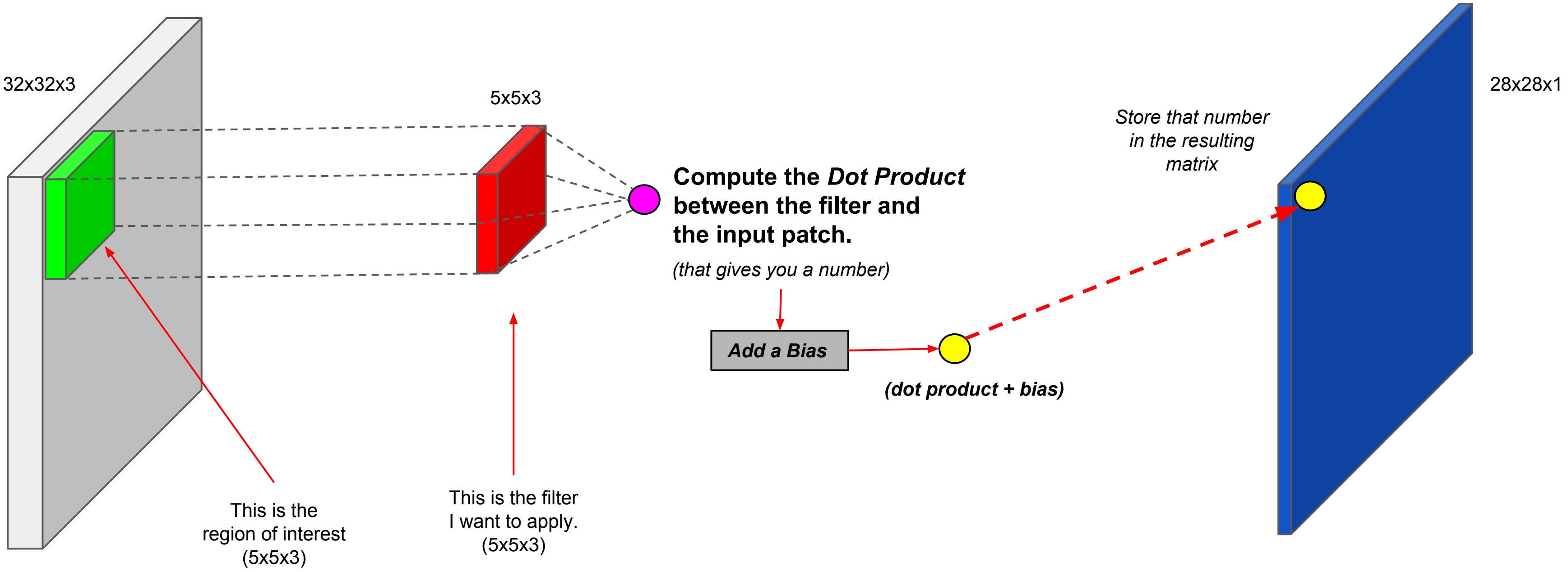
(dot product + bias)

This is the region of interest (5x5x3)

This is the filter I want to apply. (5x5x3)

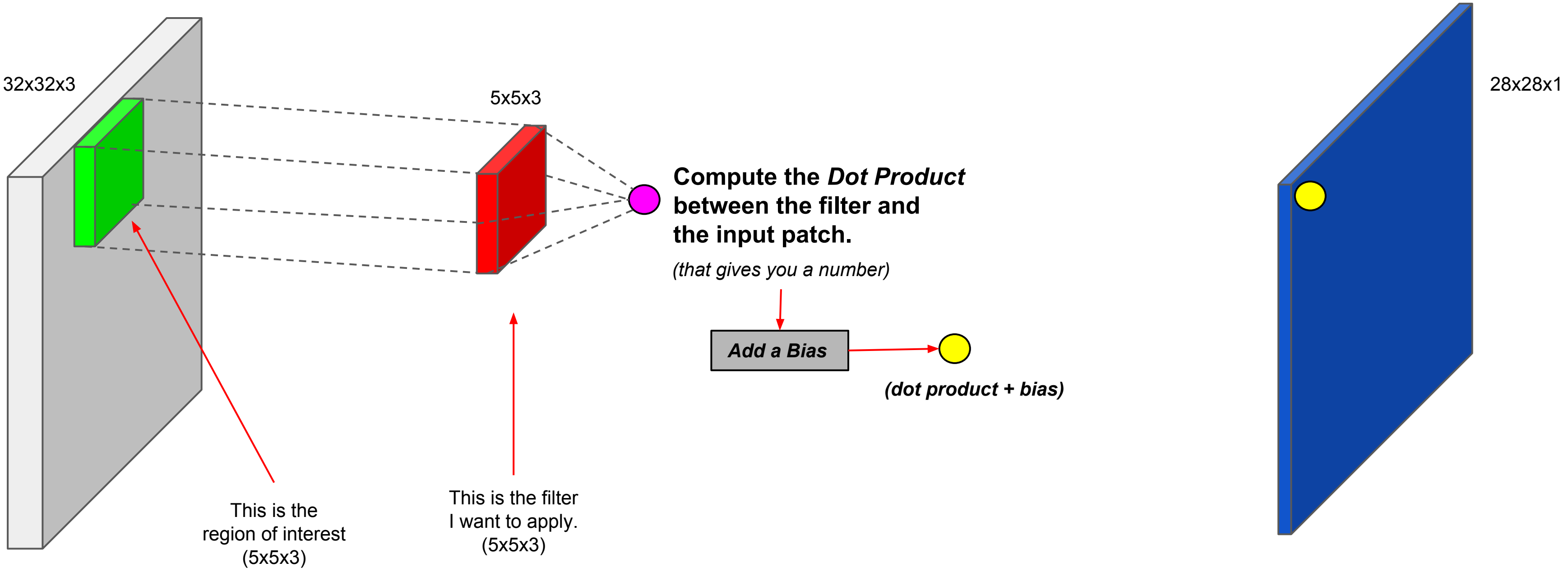
Convolutional Layers

How it works.



Convolutional Layers

How it works.



This is the region of interest (5x5x3)

This is the filter I want to apply. (5x5x3)

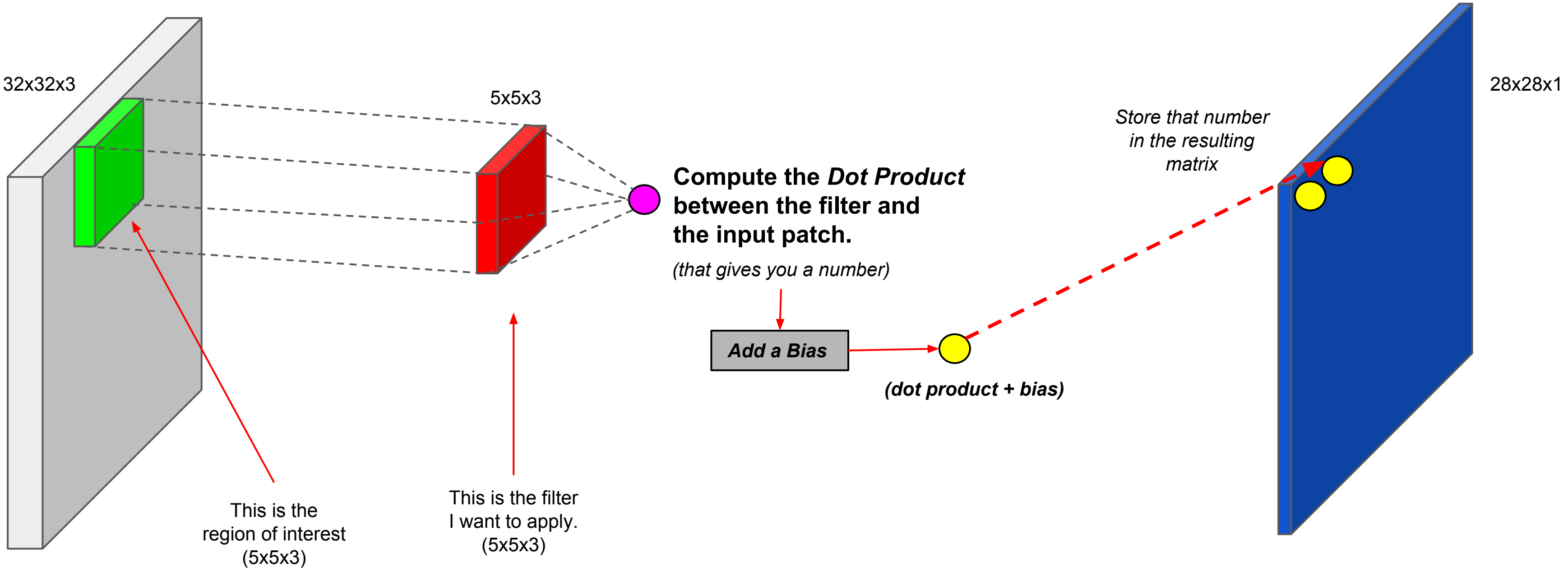
Compute the *Dot Product* between the filter and the input patch.

(that gives you a number)

(dot product + bias)

Convolutional Layers

How it works.



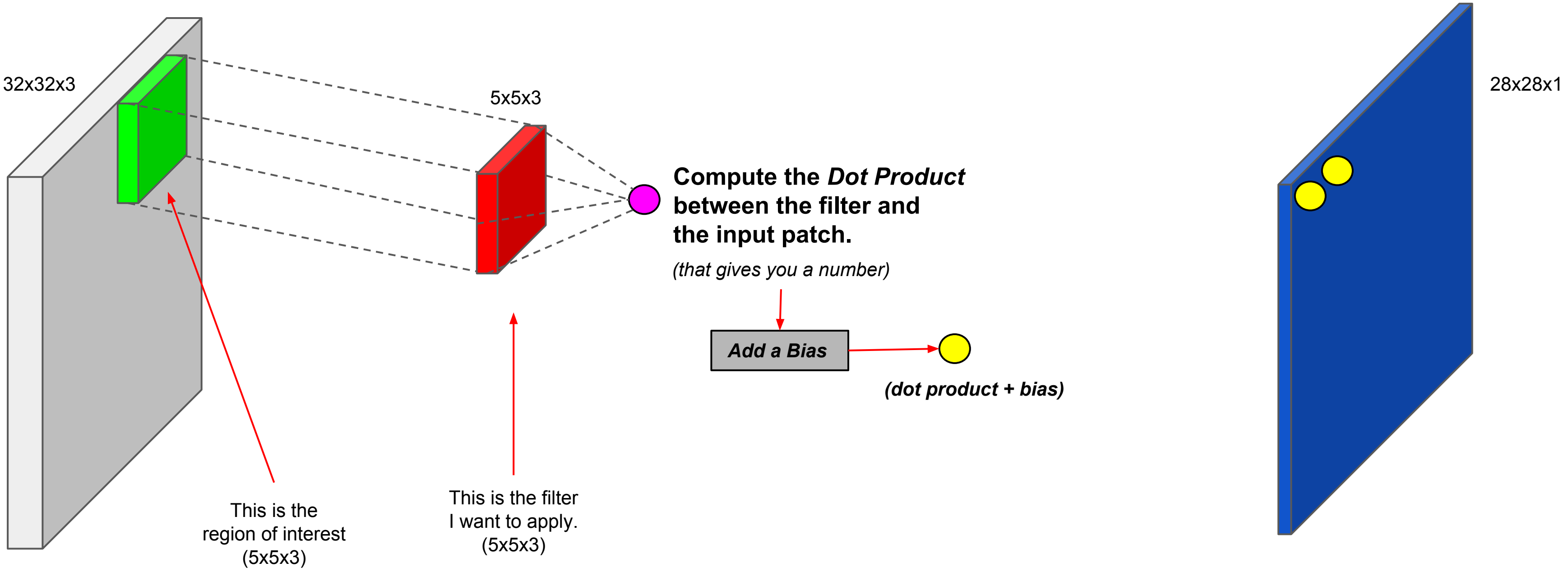
32x32x3
5x5x3
28x28x1
Store that number in the resulting matrix
Add a Bias
(dot product + bias)
This is the region of interest (5x5x3)
This is the filter I want to apply. (5x5x3)

Compute the *Dot Product* between the filter and the input patch.

(that gives you a number)

Convolutional Layers

How it works.



Compute the *Dot Product* between the filter and the input patch.

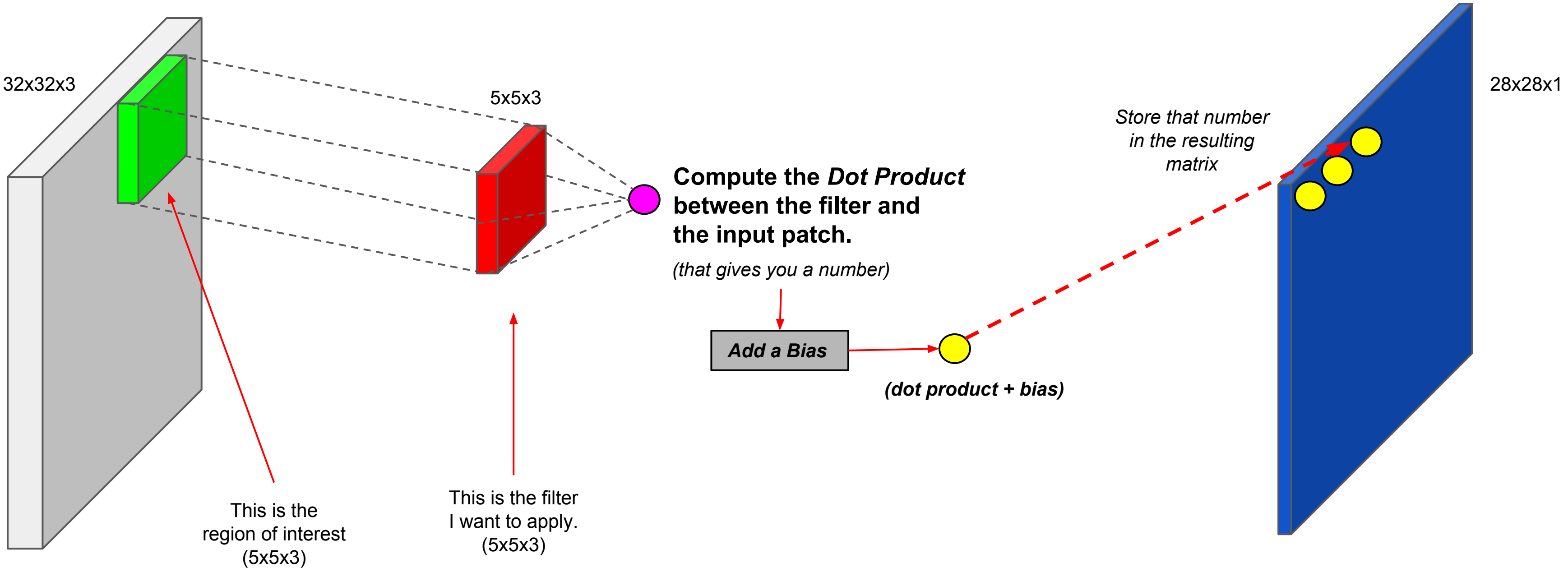
(that gives you a number)

This is the region of interest (5x5x3)

This is the filter I want to apply. (5x5x3)

Convolutional Layers

How it works.

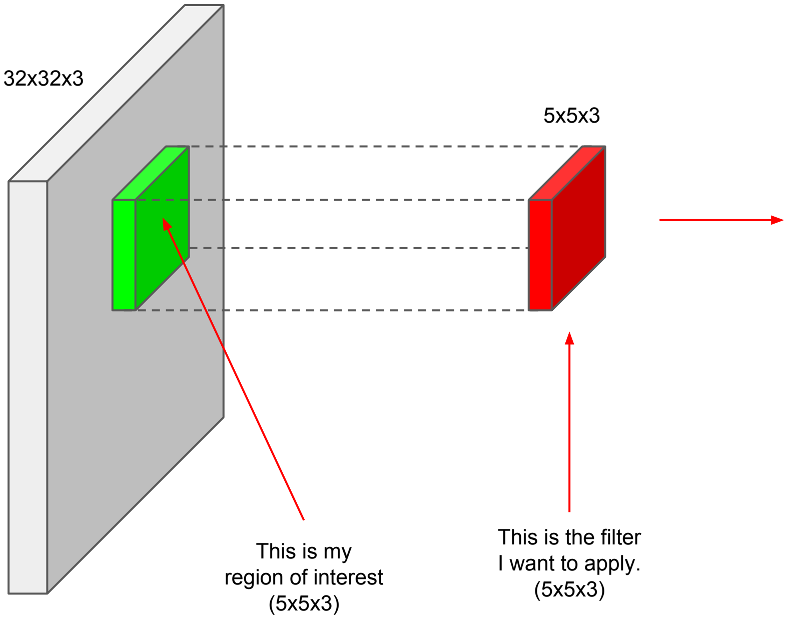

32x32x3
5x5x3
28x28x1
This is the region of interest (5x5x3)
This is the filter I want to apply. (5x5x3)
Add a Bias
(dot product + bias)
Store that number in the resulting matrix

Compute the *Dot Product* between the filter and the input patch.

(that gives you a number)

Convolutional Layers

How it works.



This is my region of interest (5x5x3)

This is the filter I want to apply. (5x5x3)

Dot Product between the patch and the filter

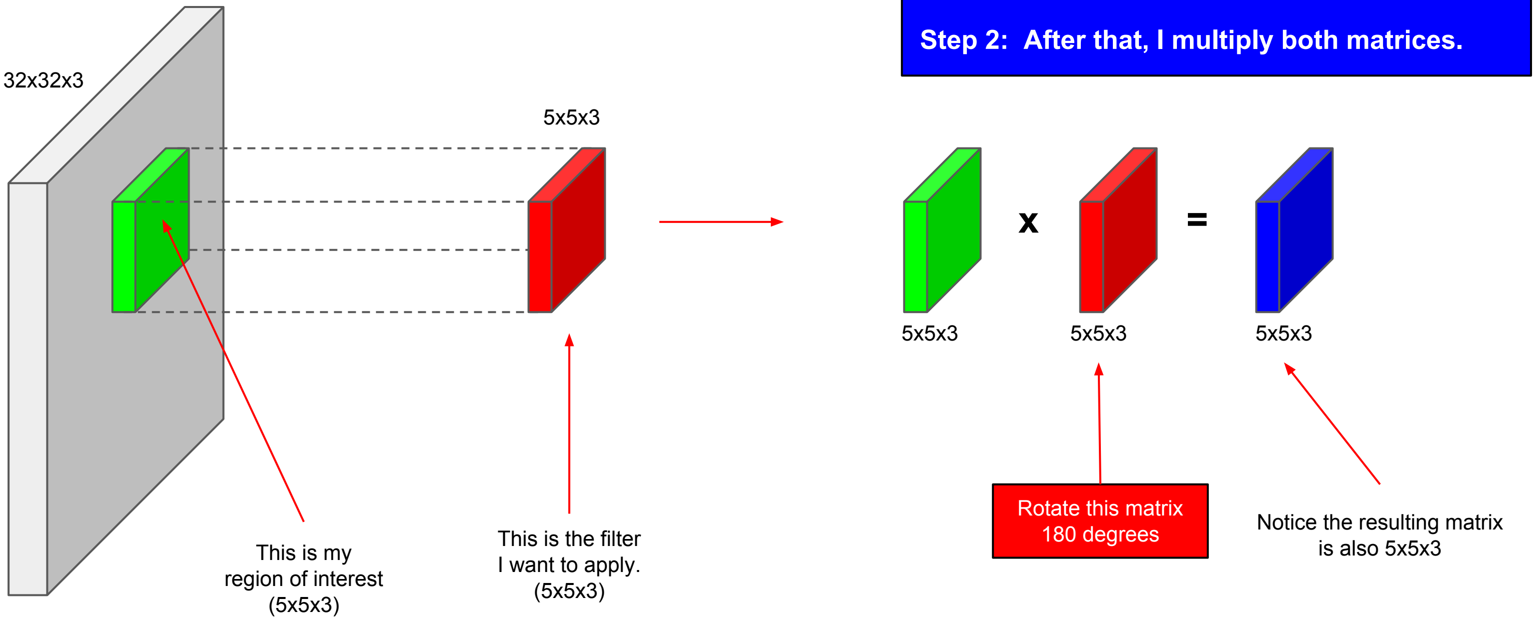
How?

Convolutional Layers

How it works.

Step 1: First, I rotate the kernel 180 degrees.

Step 2: After that, I multiply both matrices.

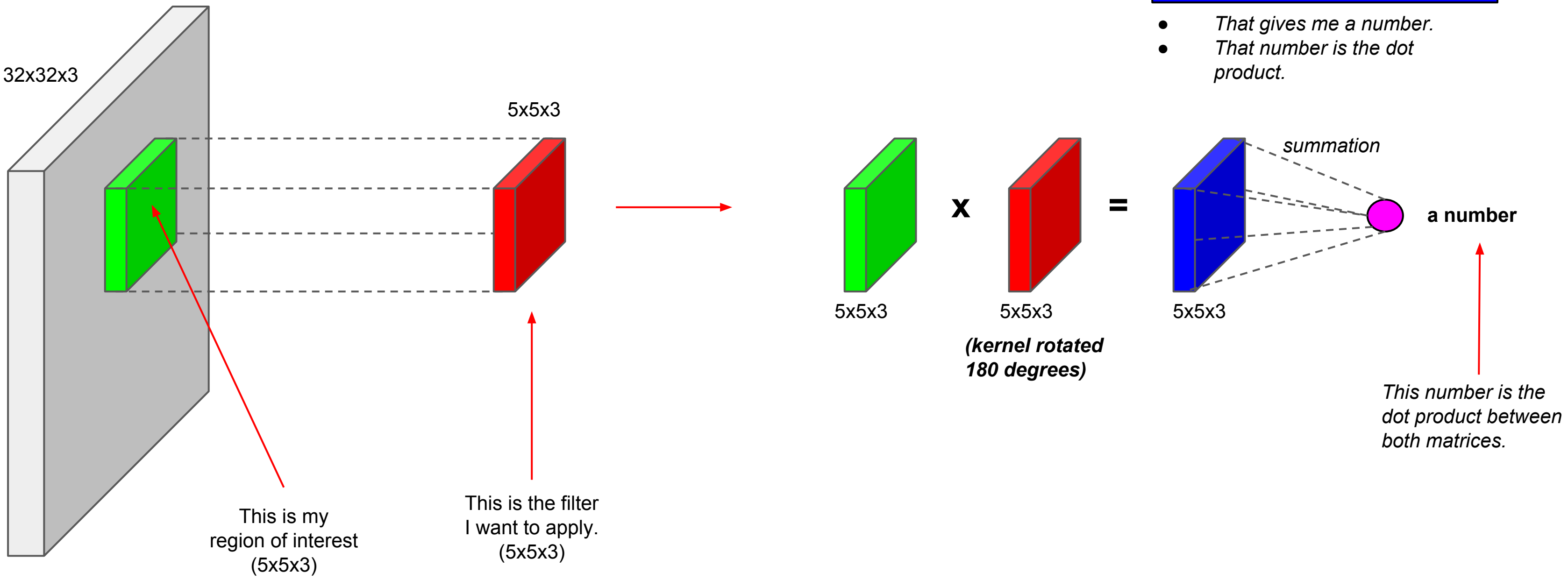


Convolutional Layers

How it works.

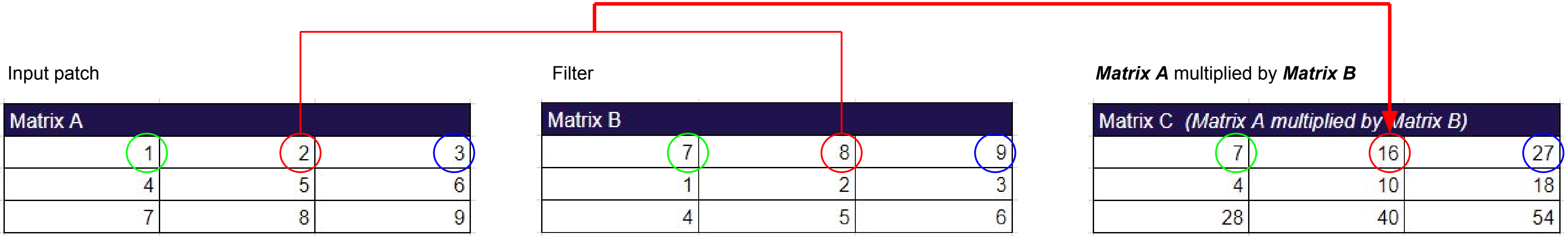
Then I sum up the values of the resulting matrix.

- *That gives me a number.*
- *That number is the dot product.*



Convolutional Layers

Example → Compute the *cross-correlation* between two matrices of depth #1



Input patch

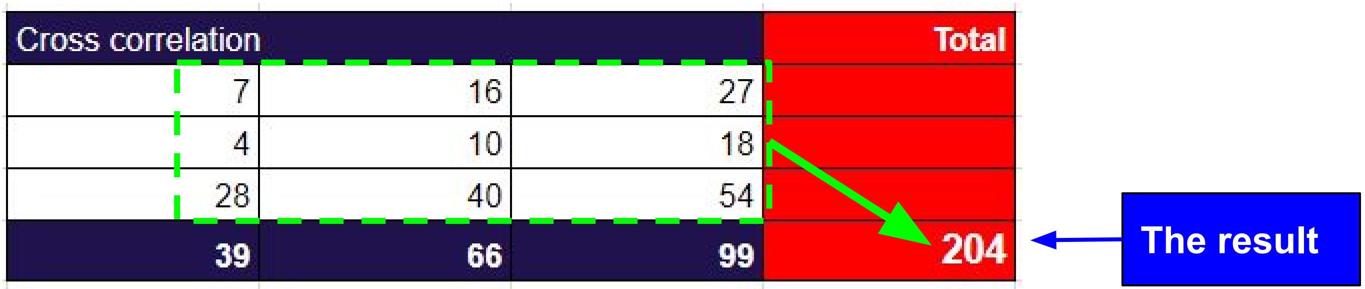
Matrix A		
1	2	3
4	5	6
7	8	9

Filter

Matrix B		
7	8	9
1	2	3
4	5	6

Matrix A **multiplied by** ***Matrix B***

Matrix C *(Matrix A multiplied by Matrix B)*		
7	16	27
4	10	18
28	40	54



Cross correlation			Total
7	16	27	
4	10	18	
28	40	54	
39	**66**	**99**	**204**

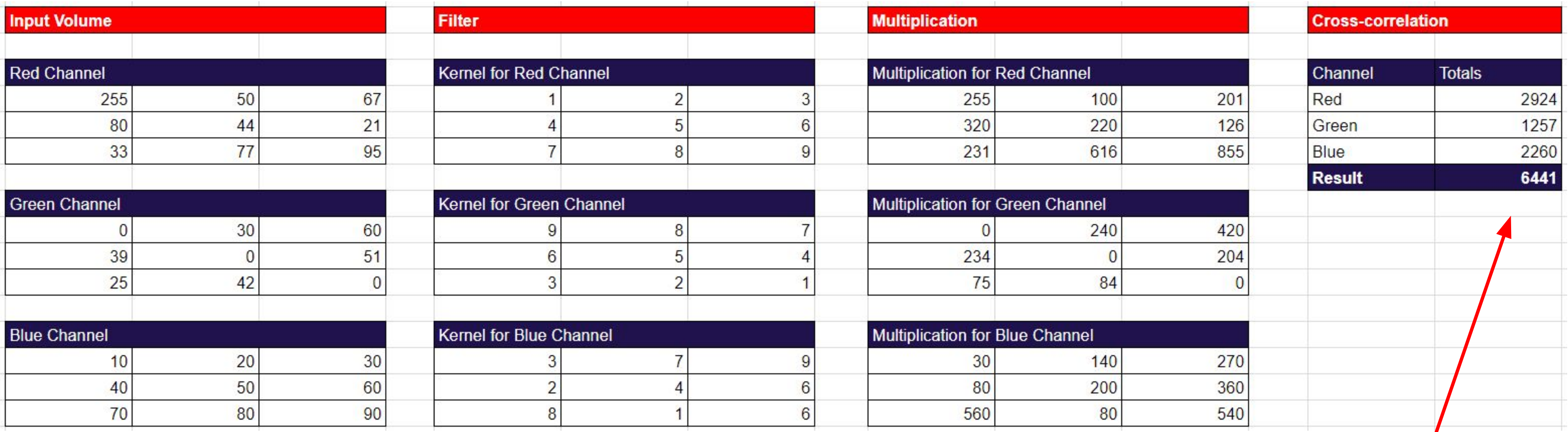
The result

7 + 16 + 27 + 4 + 10 + 18 + 28 + 40 + 54 = 204

Hands on →

Convolutional Layers

Another Example → Compute the cross-correlation between two matrices of size 3x3x3



A

Input Volume

Red Channel		
255	50	67
80	44	21
33	77	95

Green Channel		
0	30	60
39	0	51
25	42	0

Blue Channel		
10	20	30
40	50	60
70	80	90

B

Filter

Kernel for Red Channel		
1	2	3
4	5	6
7	8	9

Kernel for Green Channel		
9	8	7
6	5	4
3	2	1

Kernel for Blue Channel		
3	7	9
2	4	6
8	1	6

C = A x B

Multiplication

Multiplication for Red Channel		
255	100	201
320	220	126
231	616	855

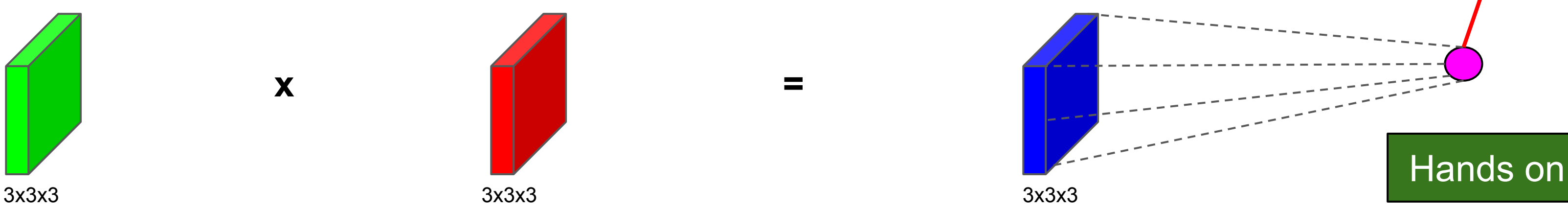
Multiplication for Green Channel		
0	240	420
234	0	204
75	84	0

Multiplication for Blue Channel		
30	140	270
80	200	360
560	80	540

D = SUM(C)

Cross-correlation

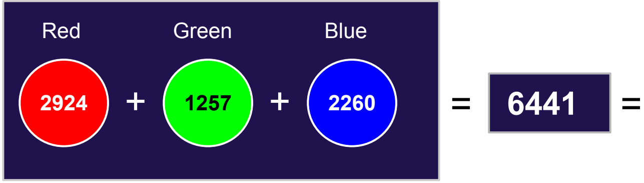
Channel	Totals
Red	2924
Green	1257
Blue	2260
Result	**6441**

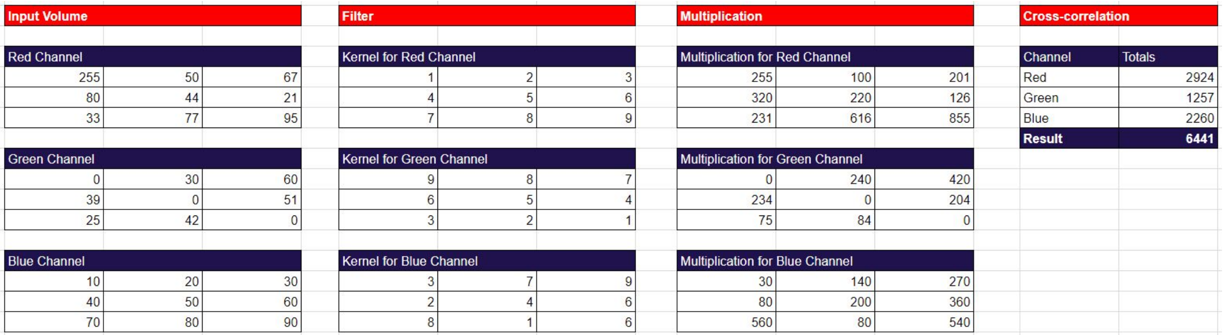


Hands on →

Convolutional Layers

In this case, the output response is the *cross-correlation* *(plus a bias)*





Input Volume

Red Channel		
255	50	67
80	44	21
33	77	95

Green Channel		
0	30	60
39	0	51
25	42	0

Blue Channel		
10	20	30
40	50	60
70	80	90

Filter

Kernel for Red Channel		
1	2	3
4	5	6
7	8	9

Kernel for Green Channel		
9	8	7
6	5	4
3	2	1

Kernel for Blue Channel		
3	7	9
2	4	6
8	1	6

Multiplication

Multiplication for Red Channel		
255	100	201
320	220	126
231	616	855

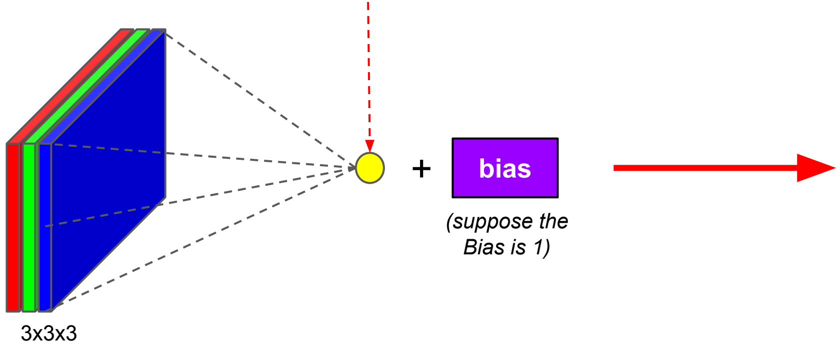
Multiplication for Green Channel		
0	240	420
234	0	204
75	84	0

Multiplication for Blue Channel		
30	140	270
80	200	360
560	80	540

Cross-correlation

Channel	Totals
Red	2924
Green	1257
Blue	2260
Result	**6441**

The result is **6441**

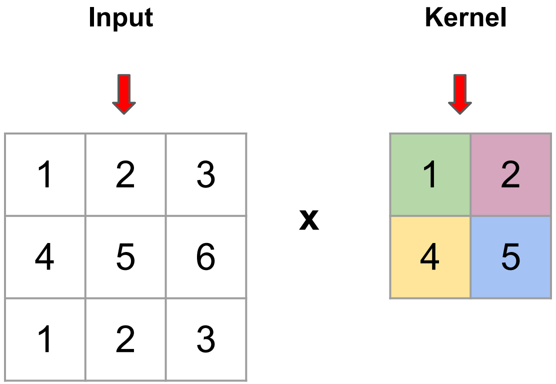


(suppose the Bias is 1)

The value of this neuron is 6441 + 1
(cross correlation + bias)

Computing the *Dot Product*

A worked example, step by step



Input

1	2	3
4	5	6
1	2	3

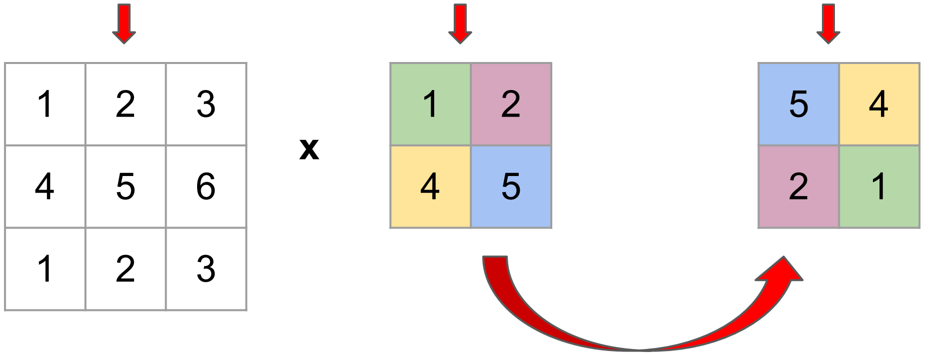
x

Kernel

1	2
4	5

Computing the *Dot Product*

A worked example, step by step



Input

1	2	3
4	5	6
1	2	3

x

Kernel

1	2
4	5

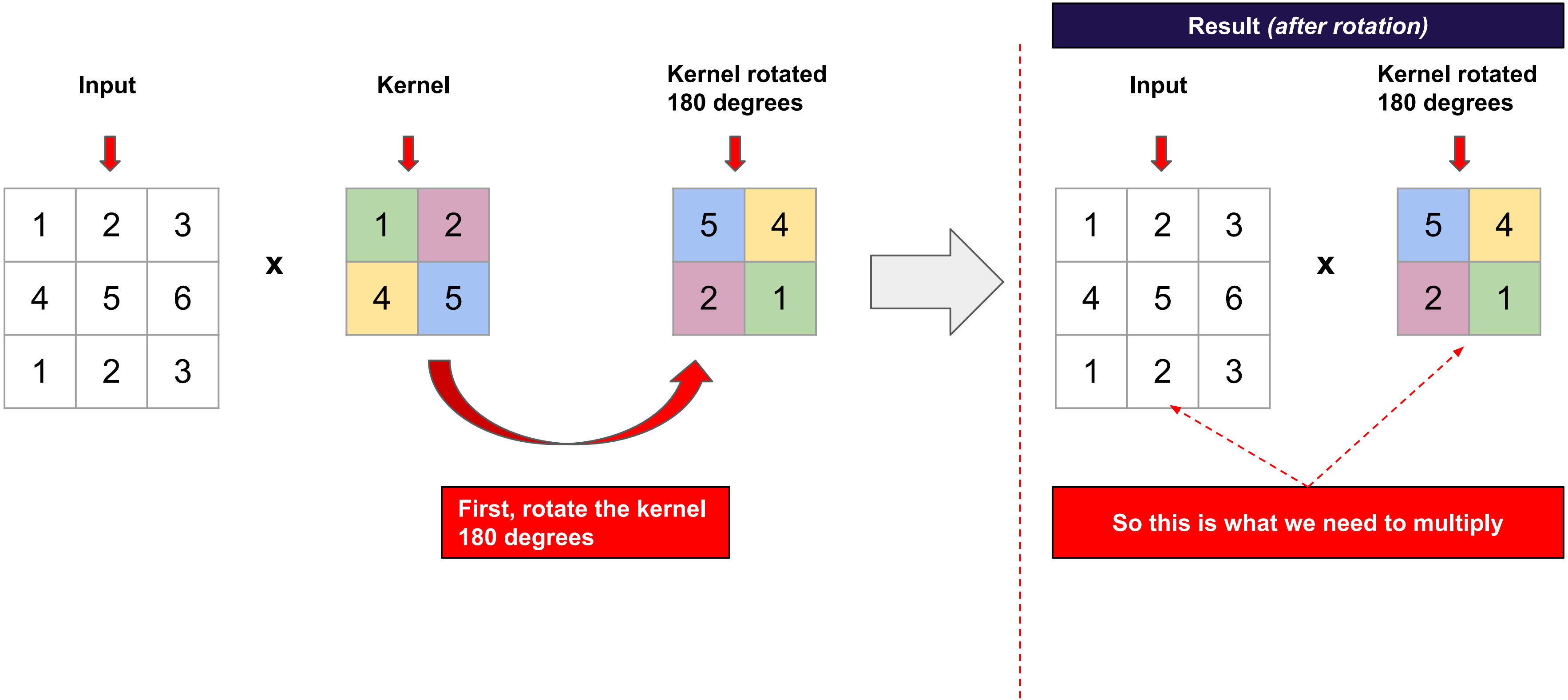
Kernel rotated 180 degrees

5	4
2	1

First, rotate the kernel 180 degrees

Computing the *Dot Product*

A worked example, step by step



Input

1	2	3
4	5	6
1	2	3

x

Kernel

1	2
4	5

Kernel rotated 180 degrees

5	4
2	1

First, rotate the kernel 180 degrees

Result *(after rotation)*

Input

1	2	3
4	5	6
1	2	3

x

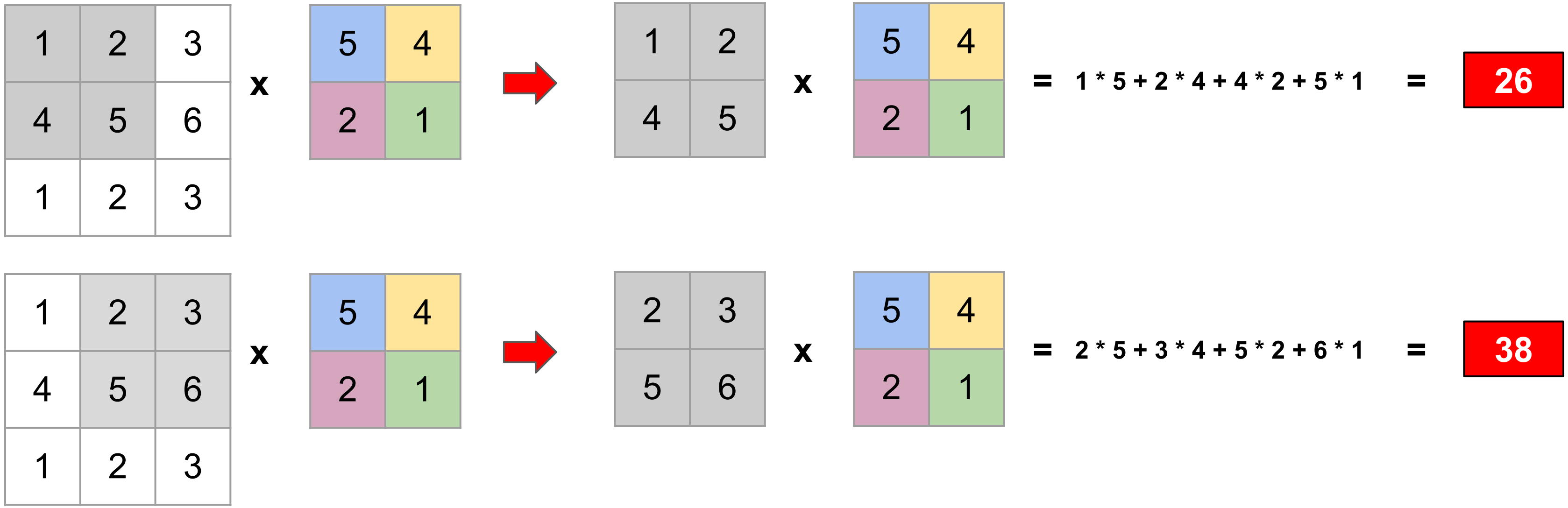
Kernel rotated 180 degrees

5	4
2	1

So this is what we need to multiply

Computing the *Dot Product*

A worked example, step by step

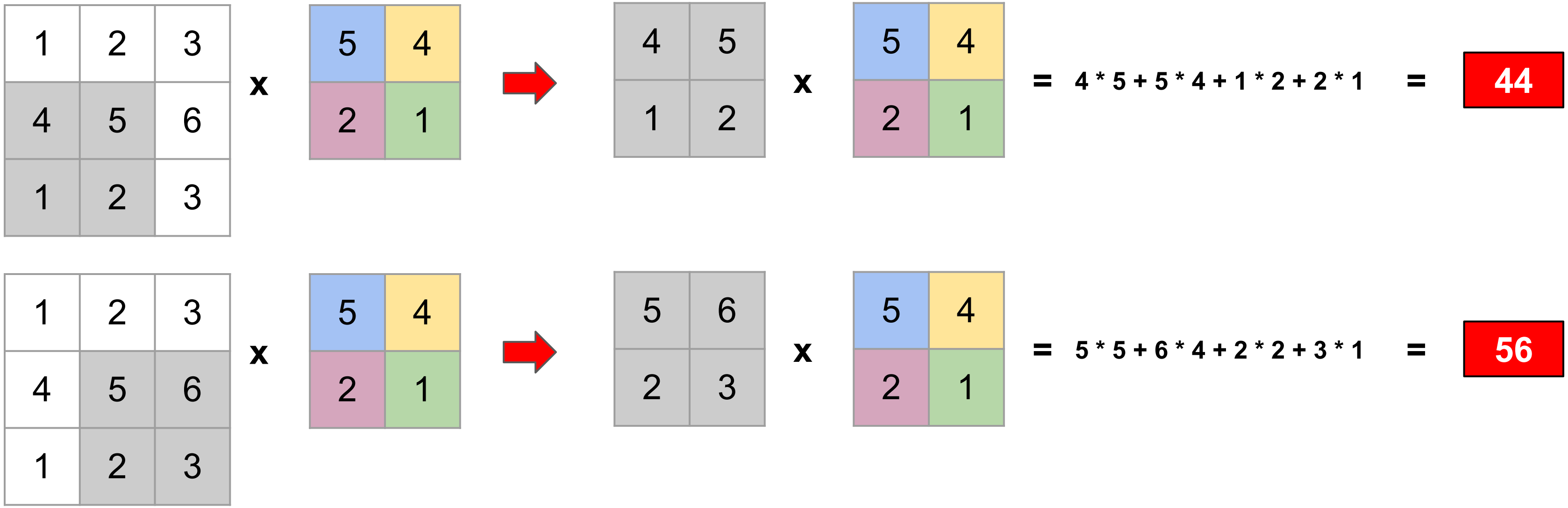


$$\begin{bmatrix}1&2&3\\4&5&6\\1&2&3\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} \rightarrow \begin{bmatrix}1&2\\4&5\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} = 1 * 5 + 2 * 4 + 4 * 2 + 5 * 1 = \boxed{26}$$

$$\begin{bmatrix}1&2&3\\4&5&6\\1&2&3\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} \rightarrow \begin{bmatrix}2&3\\5&6\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} = 2 * 5 + 3 * 4 + 5 * 2 + 6 * 1 = \boxed{38}$$

Computing the *Dot Product*

A worked example, step by step

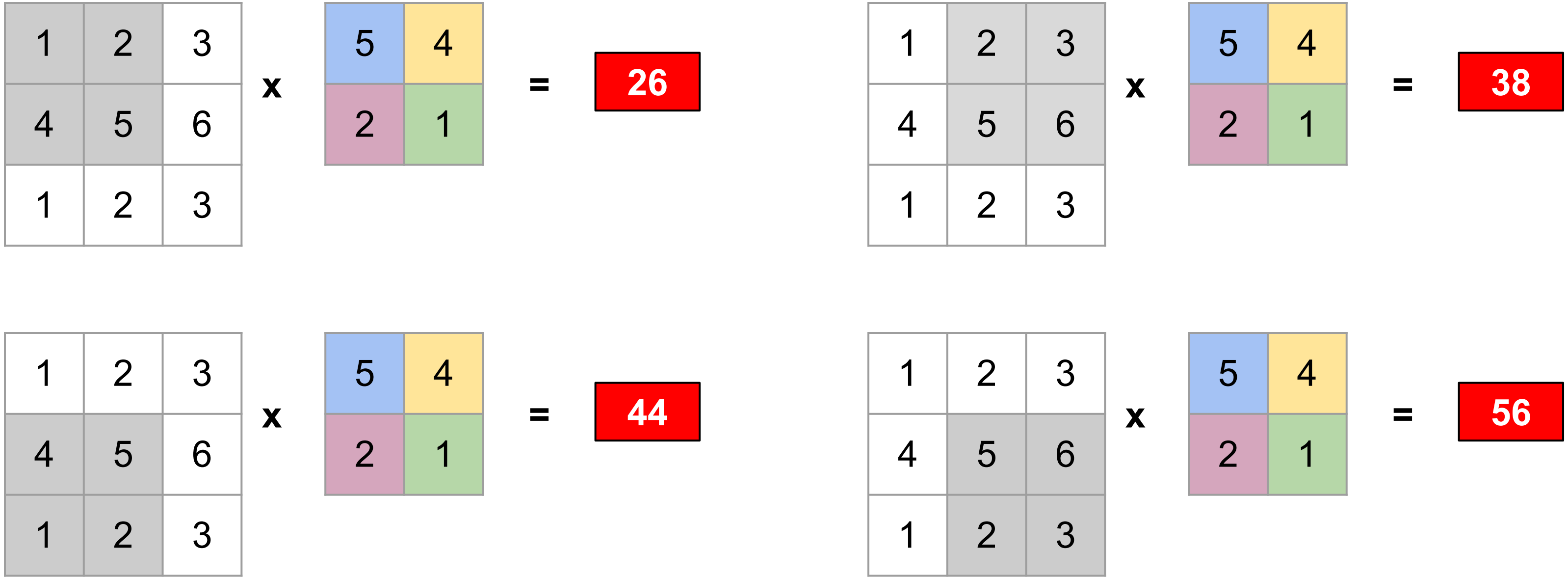


$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \mathbf{x} \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 5 \\ 1 & 2 \end{bmatrix} \mathbf{x} \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} = 4 * 5 + 5 * 4 + 1 * 2 + 2 * 1 = \boxed{44}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \mathbf{x} \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 5 & 6 \\ 2 & 3 \end{bmatrix} \mathbf{x} \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} = 5 * 5 + 6 * 4 + 2 * 2 + 3 * 1 = \boxed{56}$$

Computing the *Dot Product*

A worked example, step by step



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \text{ x } \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} = \boxed{26}$$

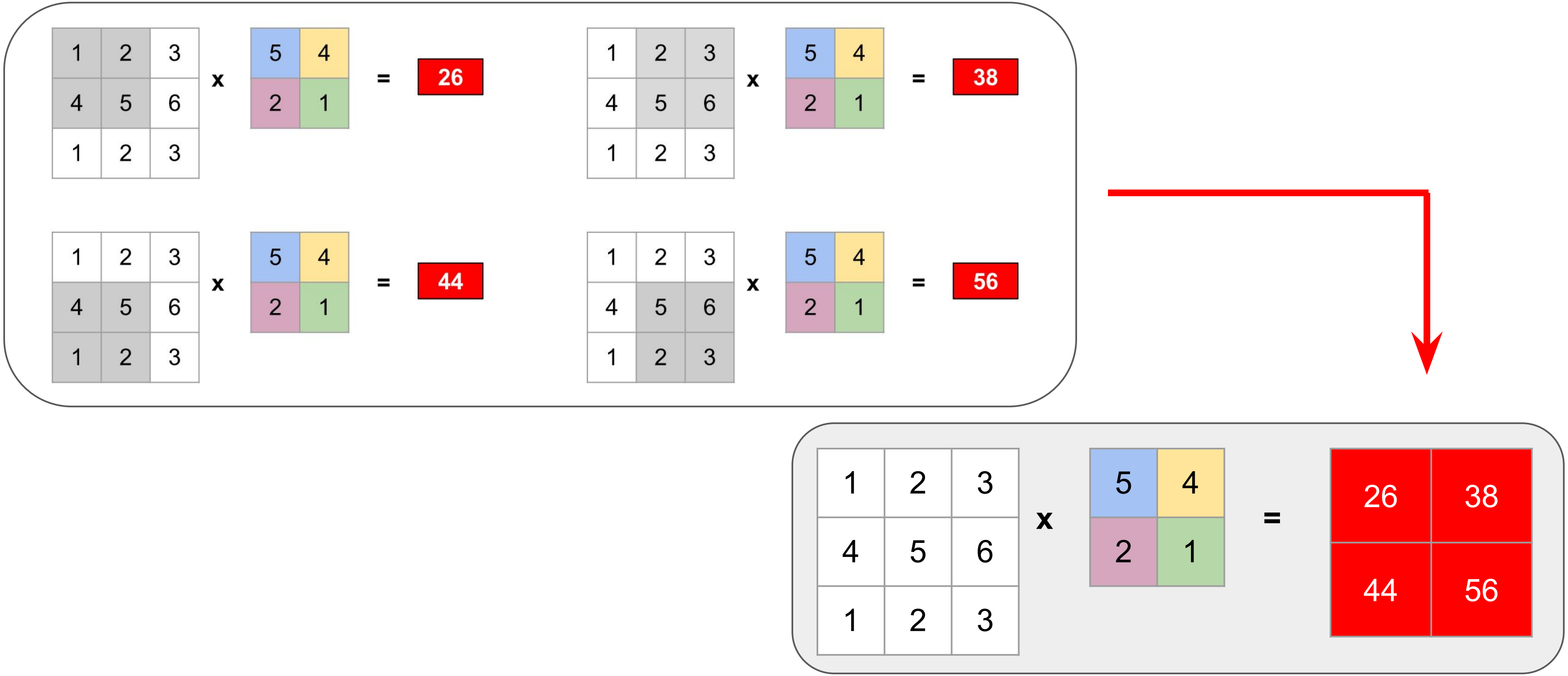
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \text{ x } \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} = \boxed{38}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \text{ x } \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} = \boxed{44}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \text{ x } \begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix} = \boxed{56}$$

Computing the *Dot Product*

A worked example, step by step



$$\begin{bmatrix}1&2&3\\4&5&6\\1&2&3\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} = 26$$

$$\begin{bmatrix}1&2&3\\4&5&6\\1&2&3\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} = 38$$

$$\begin{bmatrix}1&2&3\\4&5&6\\1&2&3\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} = 44$$

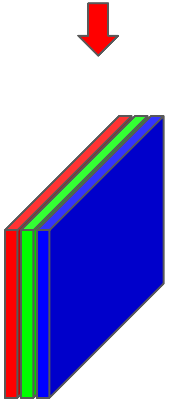
$$\begin{bmatrix}1&2&3\\4&5&6\\1&2&3\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} = 56$$

$$\begin{bmatrix}1&2&3\\4&5&6\\1&2&3\end{bmatrix} \times \begin{bmatrix}5&4\\2&1\end{bmatrix} = \begin{bmatrix}26&38\\44&56\end{bmatrix}$$

Convolutional Layers

Just to recap

I have an input patch
of depth 3
(it has 3 channels)



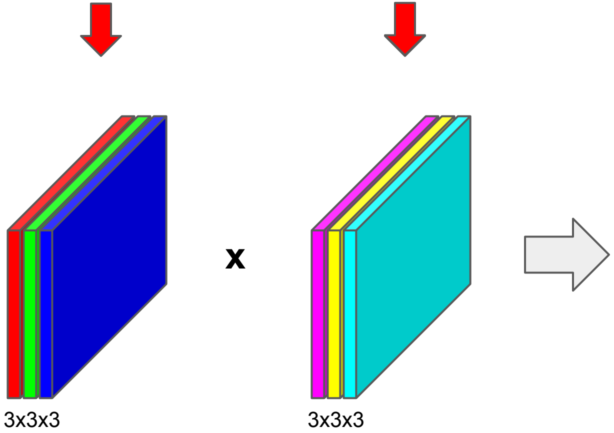
3x3x3

Convolutional Layers

Just to recap

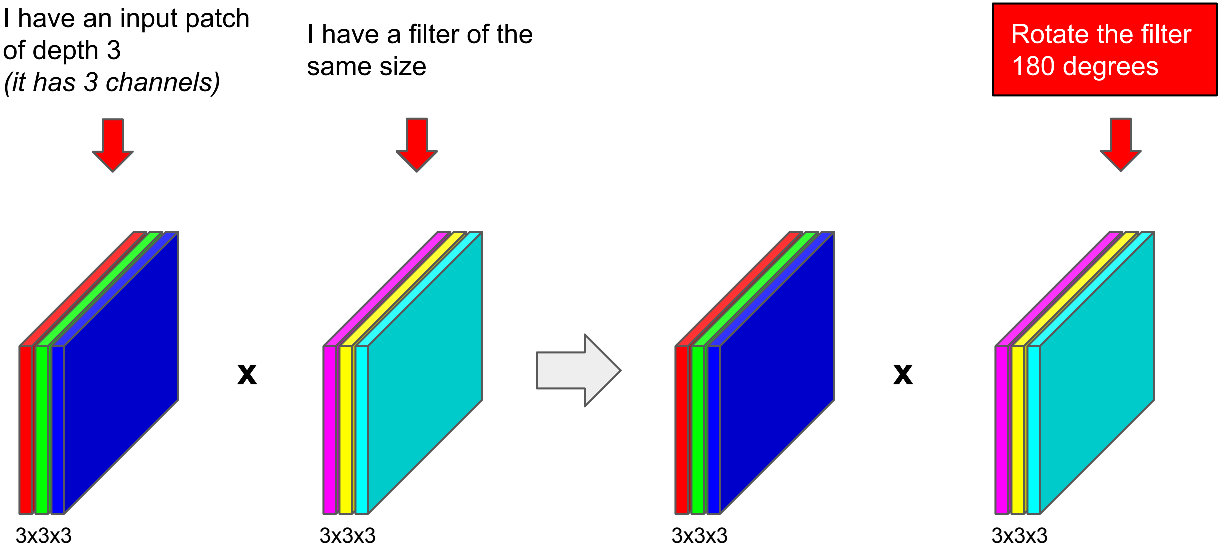
I have an input patch of depth 3
(it has 3 channels)

I have a filter of the same size



Convolutional Layers

Just to recap


I have an input patch of depth 3
(it has 3 channels)
I have a filter of the same size
Rotate the filter 180 degrees
3x3x3
x
3x3x3
3x3x3
x
3x3x3

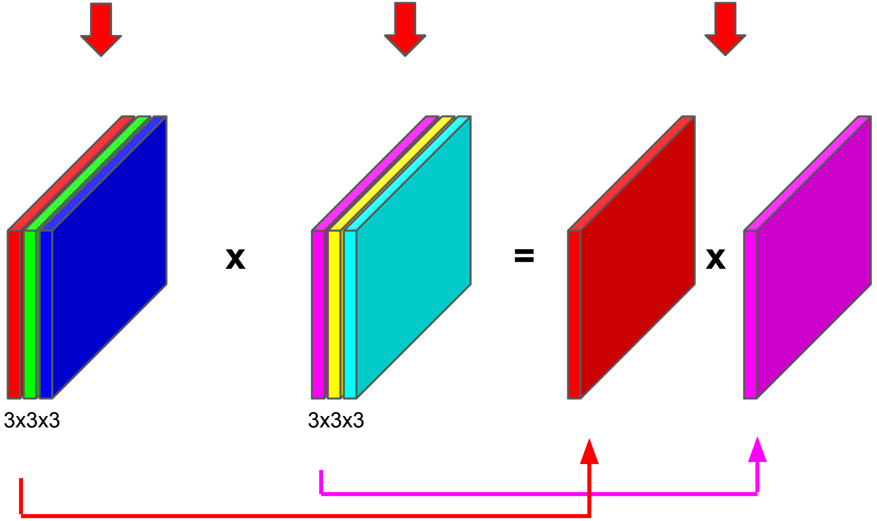
Convolutional Layers

Just to recap

I have an input patch of depth 3
(it has 3 channels)

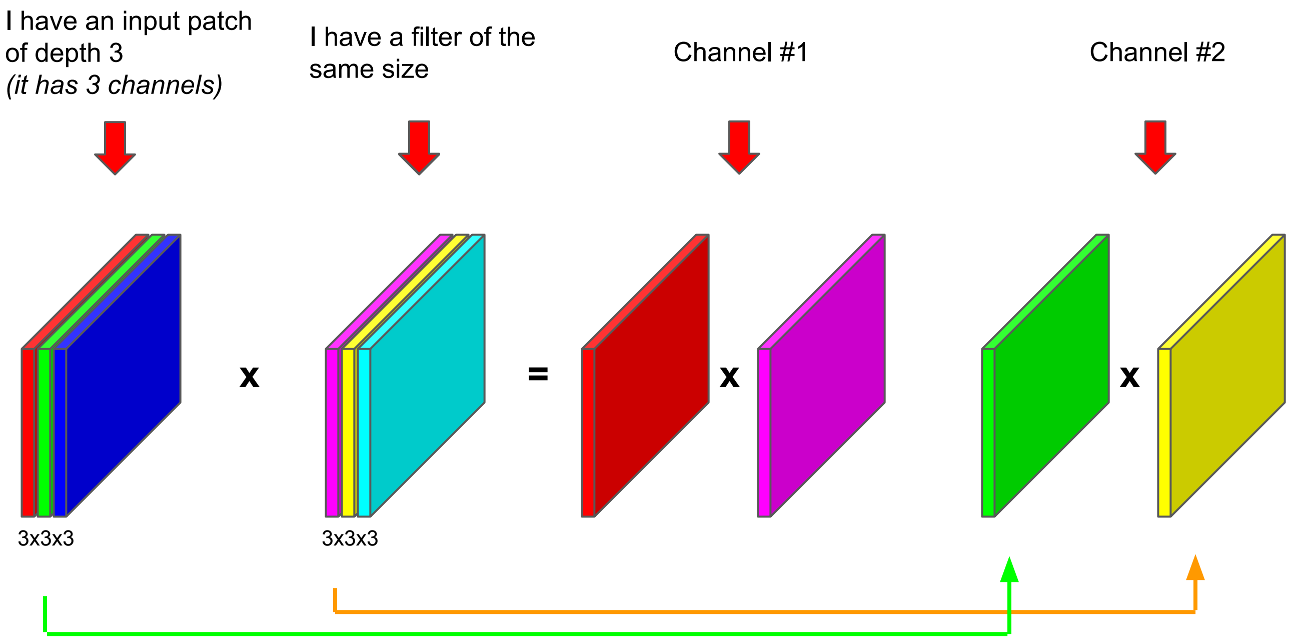
I have a filter of the same size

Channel #1



Convolutional Layers

Just to recap


I have an input patch of depth 3
(it has 3 channels)

I have a filter of the same size

Convolutional Layers

Just to recap

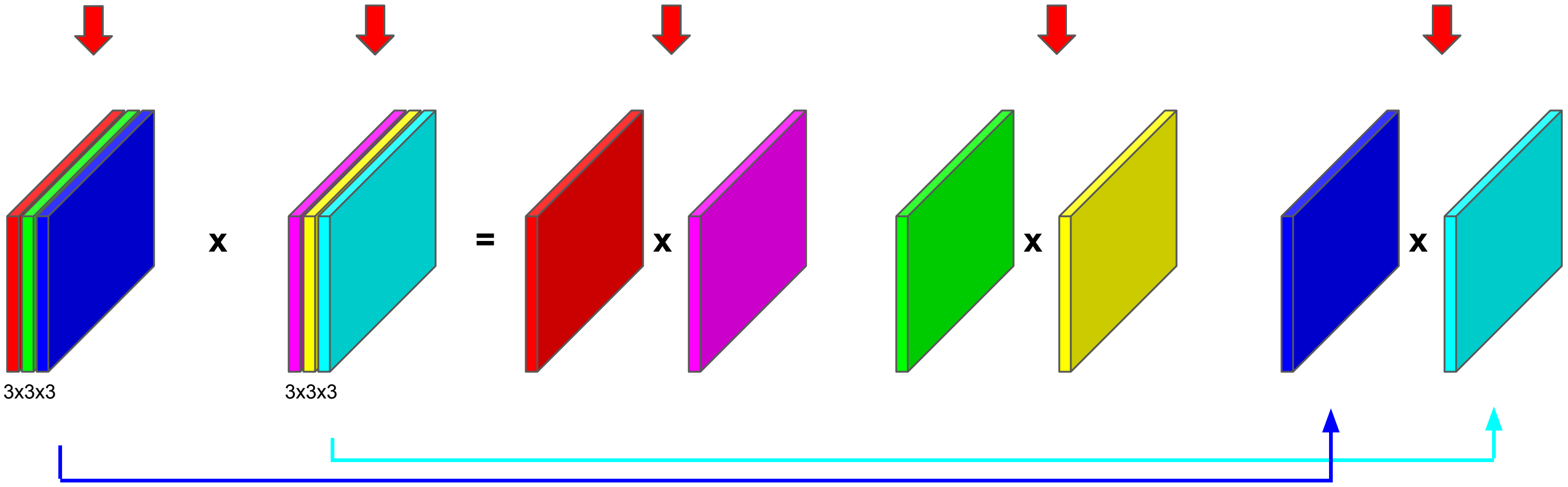
I have an input patch of depth 3
(it has 3 channels)

I have a filter of the same size

Channel #1

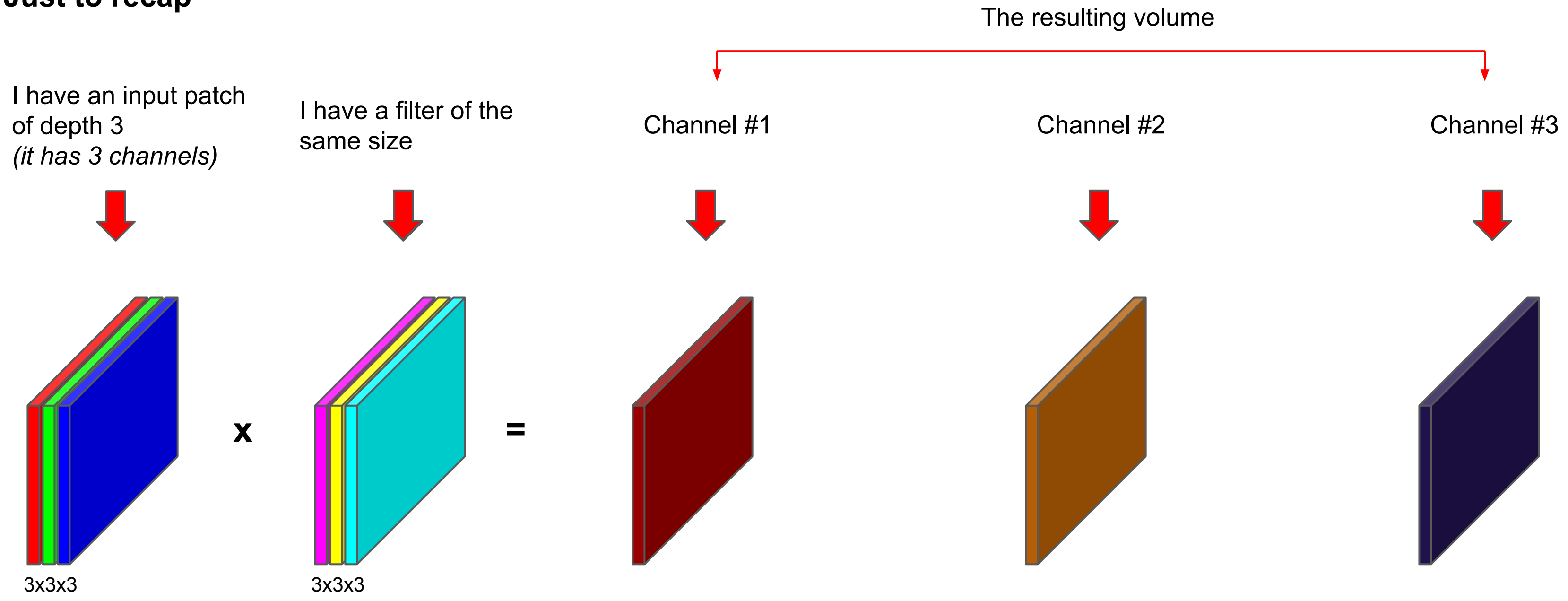
Channel #2

Channel #3



Convolutional Layers

Just to recap



The resulting volume

I have an input patch of depth 3 *(it has 3 channels)*

I have a filter of the same size

Channel #1

Channel #2

Channel #3

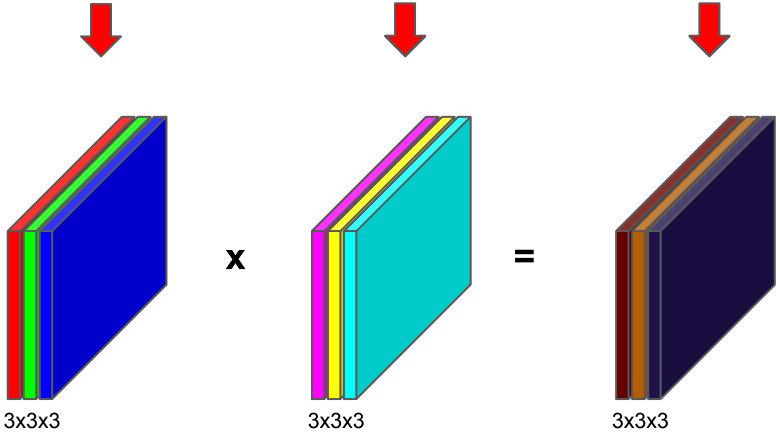
Convolutional Layers

Just to recap

I have an input patch
of depth 3
(it has 3 channels)

I have a filter of the
same size

The resulting volume
C = A x B



Convolutional Layers

Just to recap

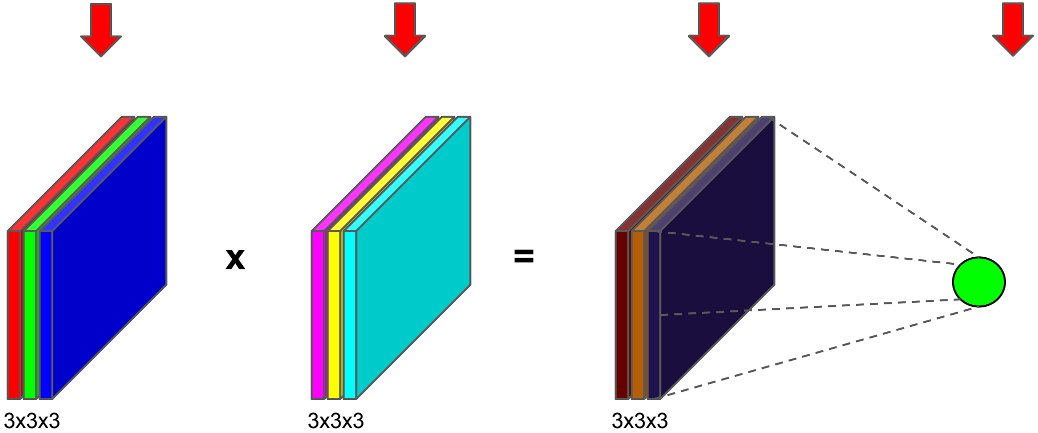
The dot product between A and B

I have an input patch of depth 3
(it has 3 channels)

I have a filter of the same size

The resulting volume
C = A x B

Summation of channels
D = SUM(C)



Convolutional Layers

Just to recap

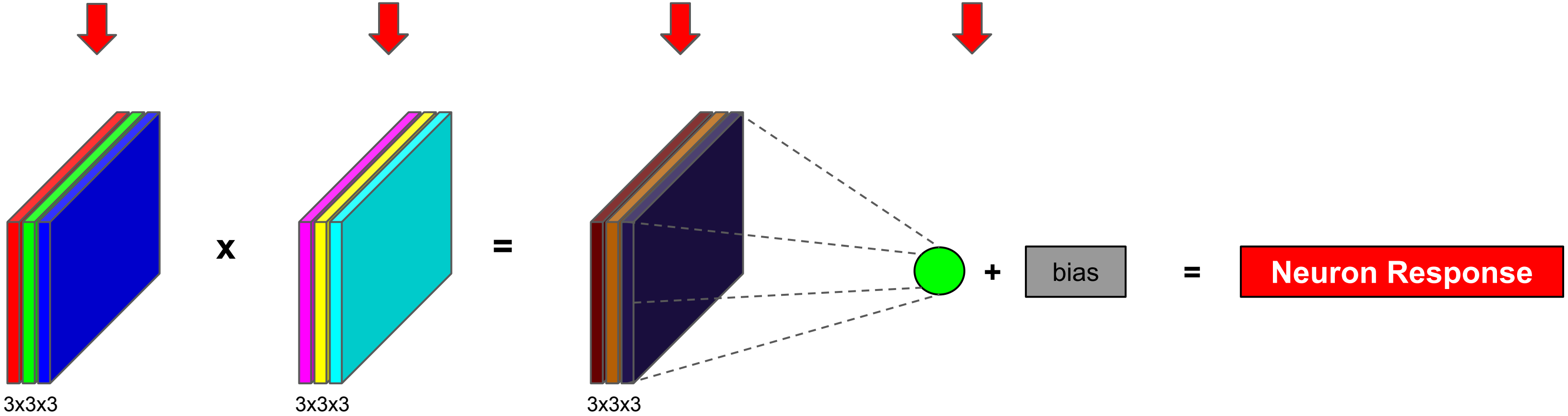
The dot product between A and B

I have an input patch
of depth 3
(it has 3 channels)

I have a filter of the
same size

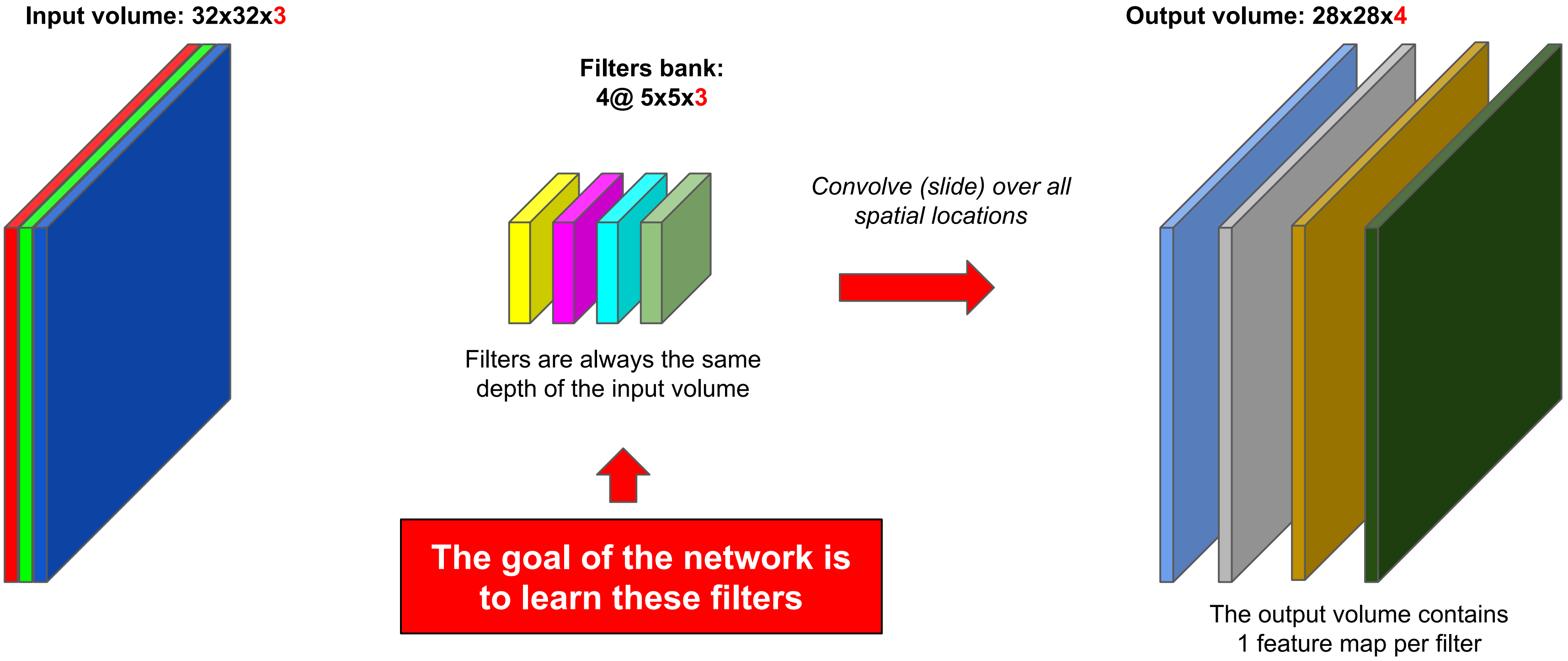
The resulting volume
C = A x B

Summation of channels
D = SUM(C)



Convolutional Layers

They produce *feature maps* by convolving an input volume with a bank of filters.



Input volume: 32x32x3

Filters bank:
4@ 5x5x3

Filters are always the same depth of the input volume

Convolve (slide) over all spatial locations

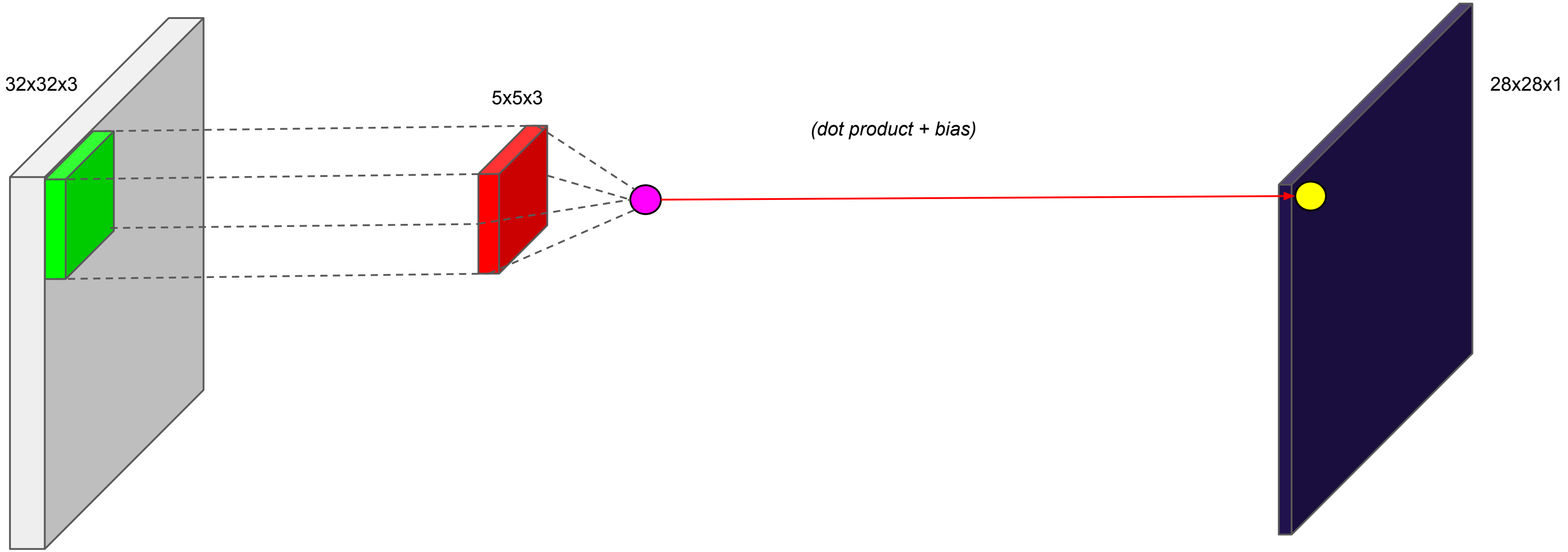
Output volume: 28x28x4

The output volume contains 1 feature map per filter

The goal of the network is to learn these filters

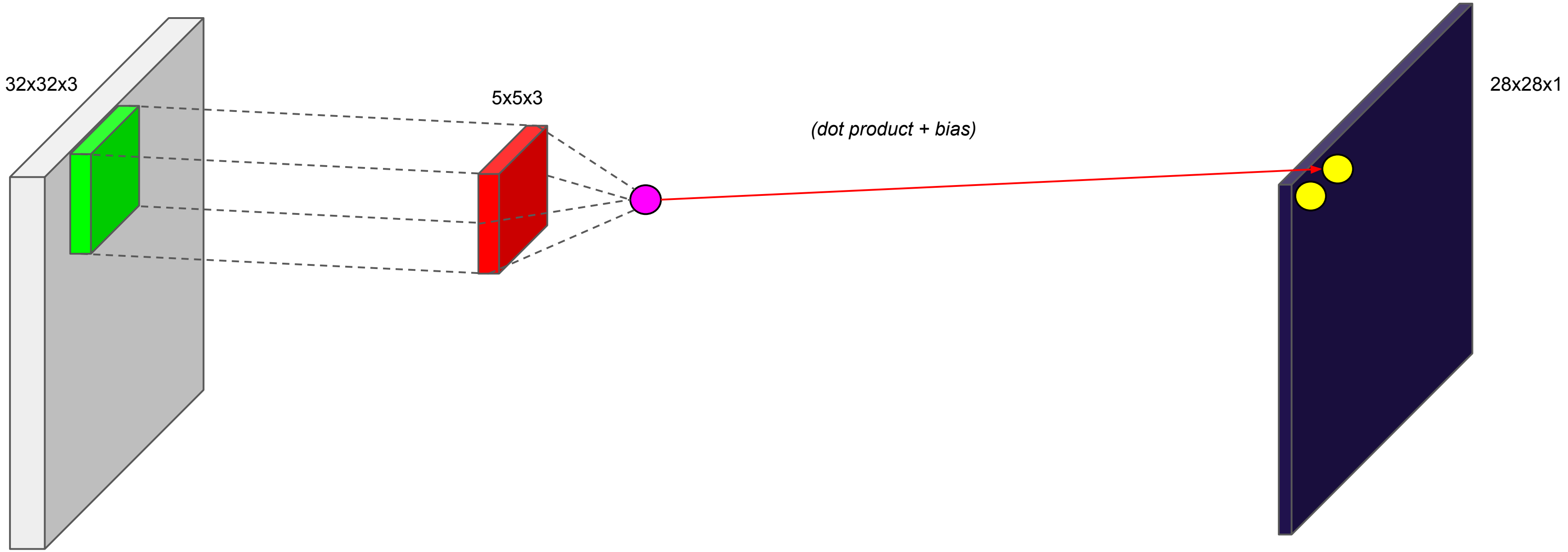
Convolutional Layers

Convolve (slide) over all spatial locations



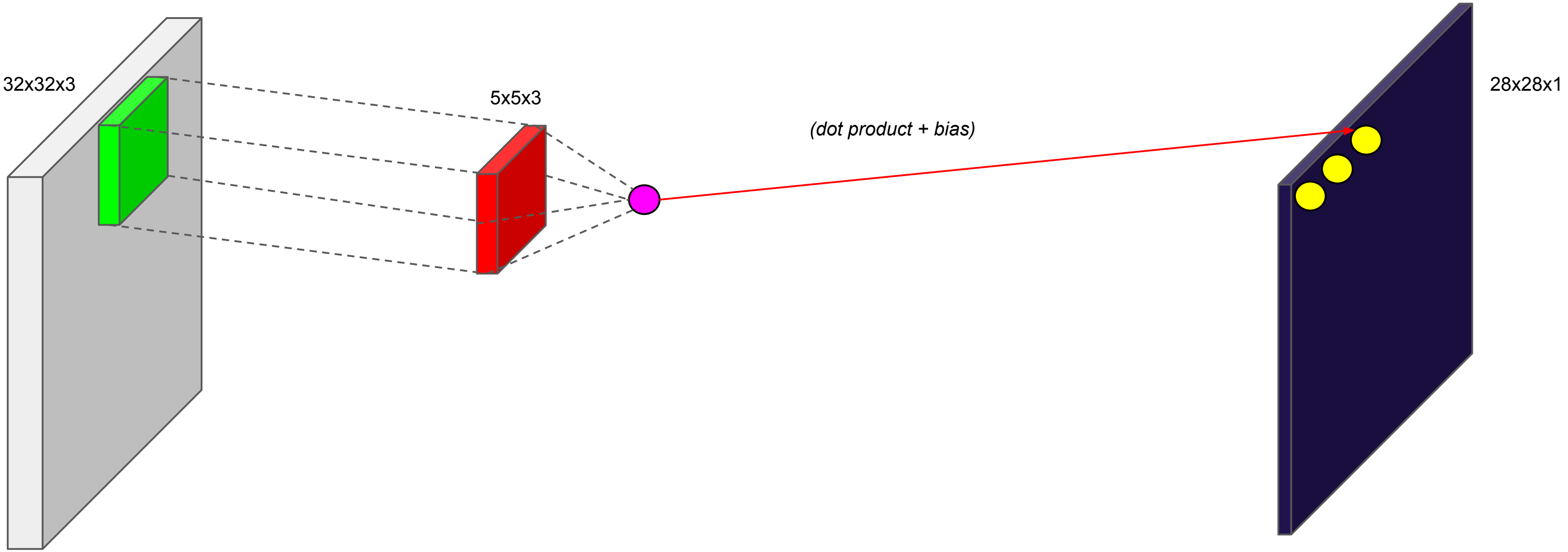
Convolutional Layers

Convolve (slide) over all spatial locations



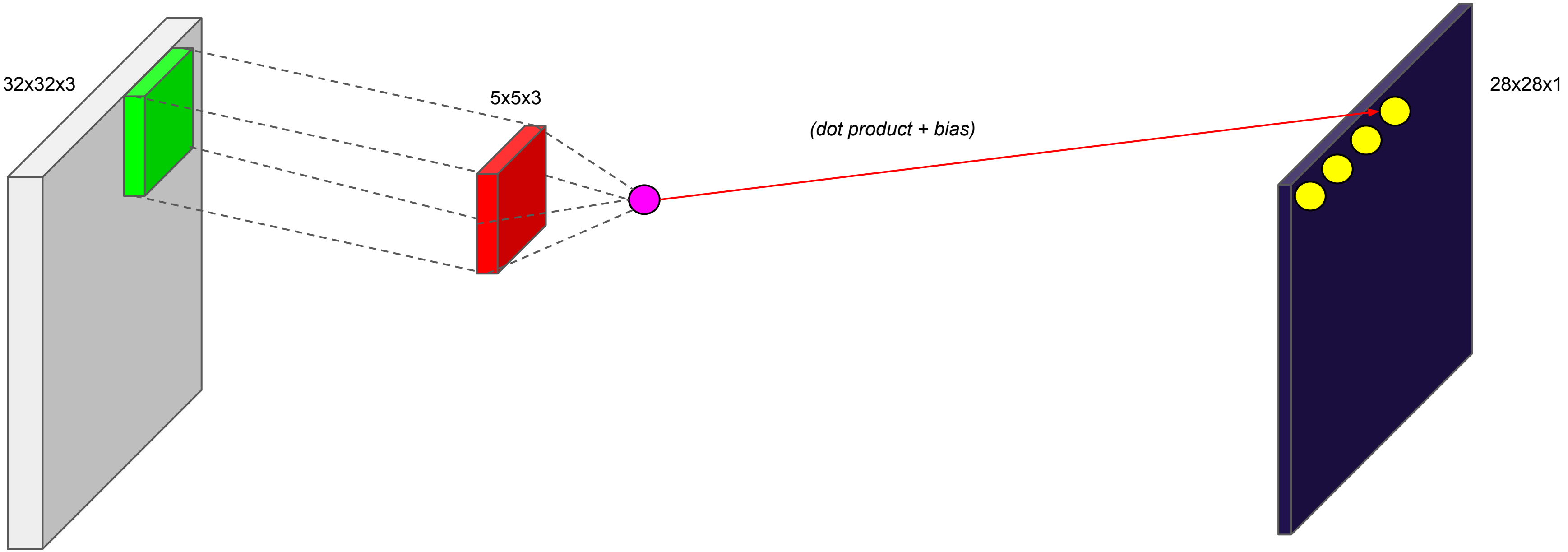
Convolutional Layers

Convolve (slide) over all spatial locations



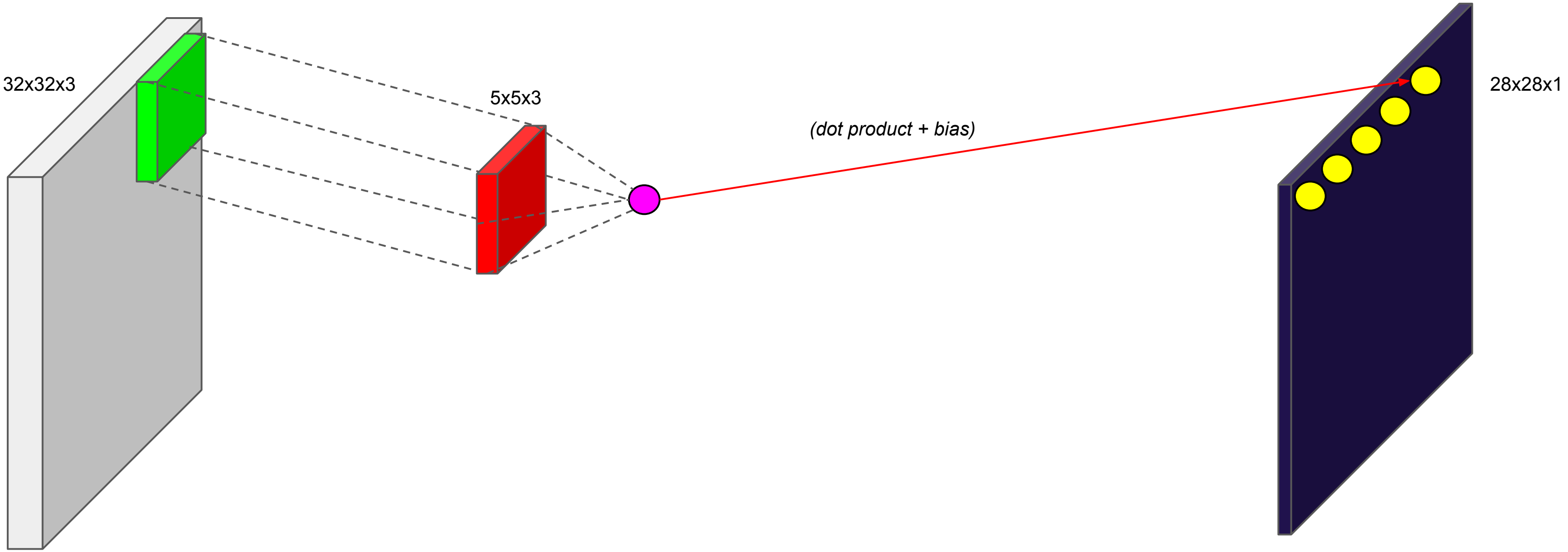
Convolutional Layers

Convolve (slide) over all spatial locations



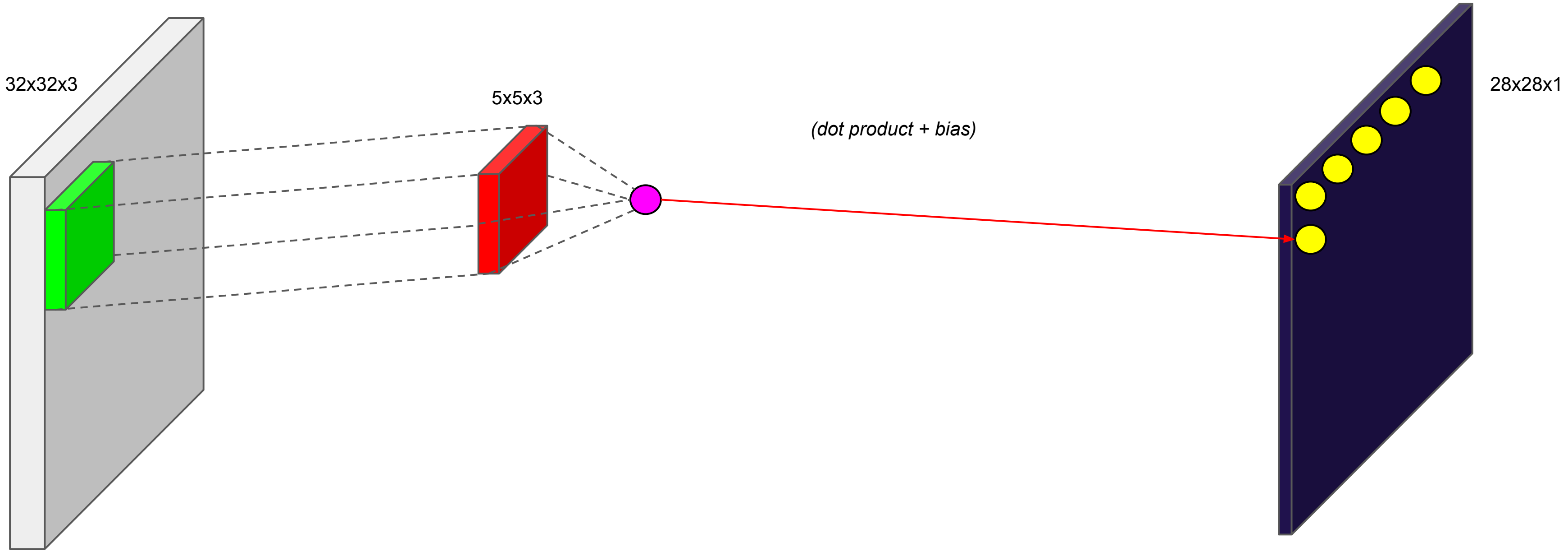
Convolutional Layers

Convolve (slide) over all spatial locations



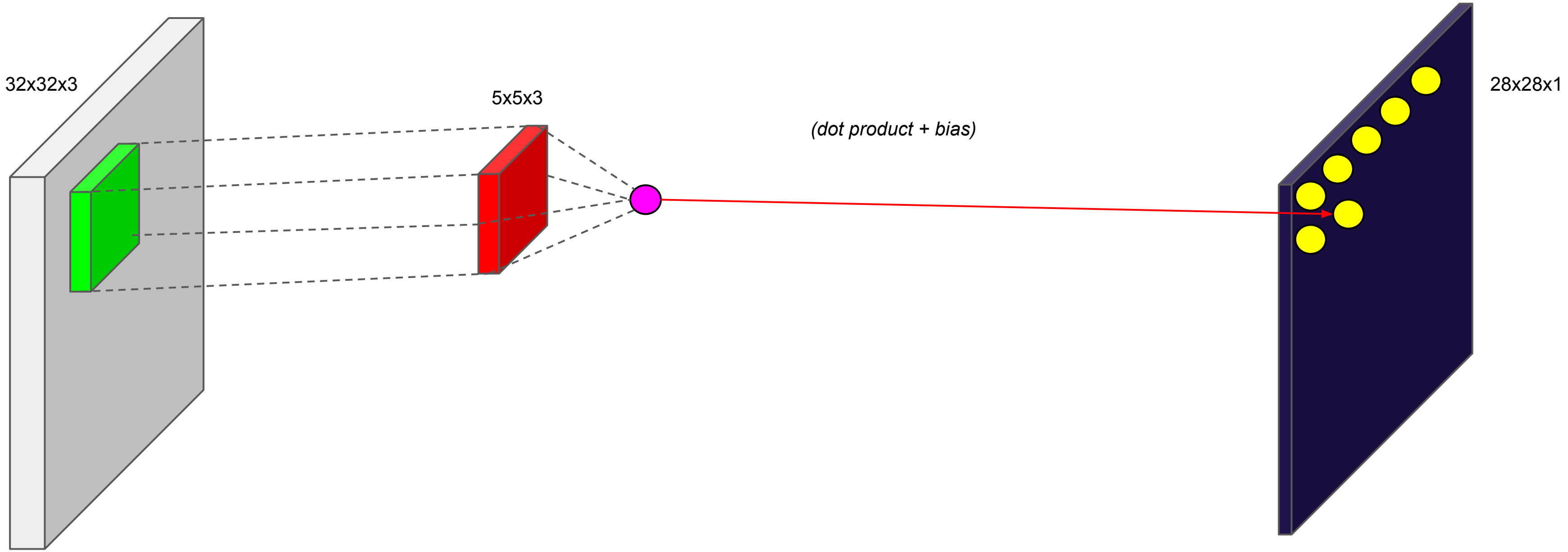
Convolutional Layers

Convolve (slide) over all spatial locations



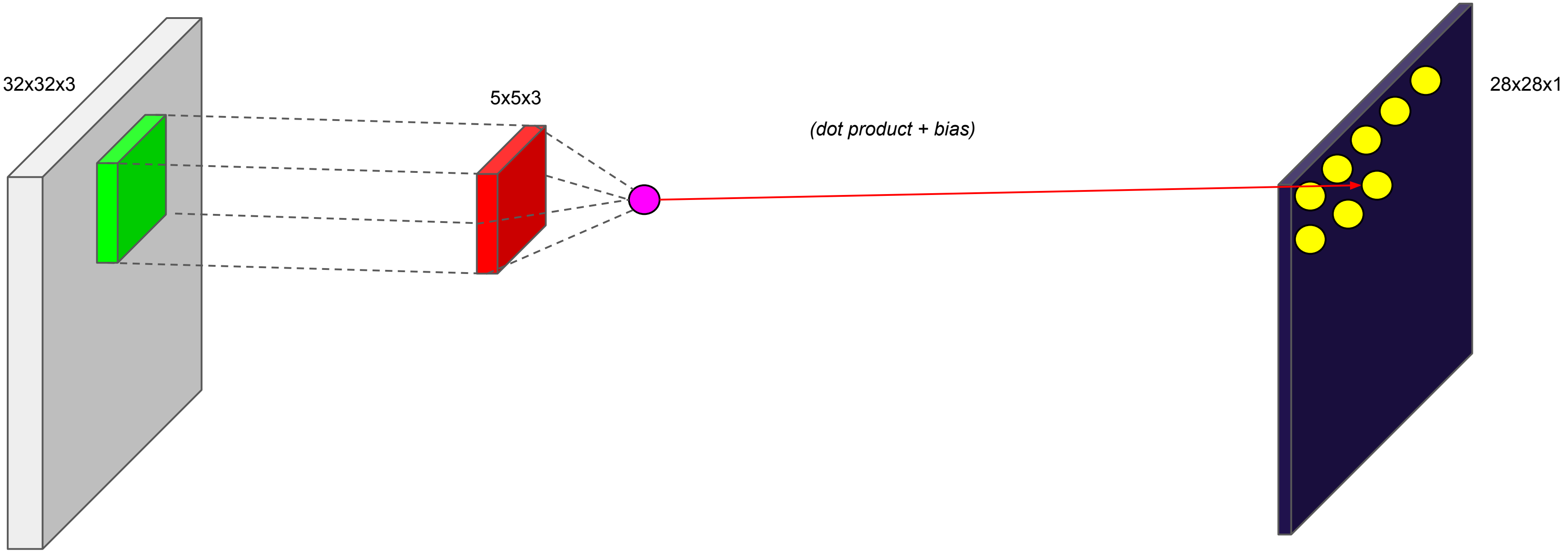
Convolutional Layers

Convolve (slide) over all spatial locations



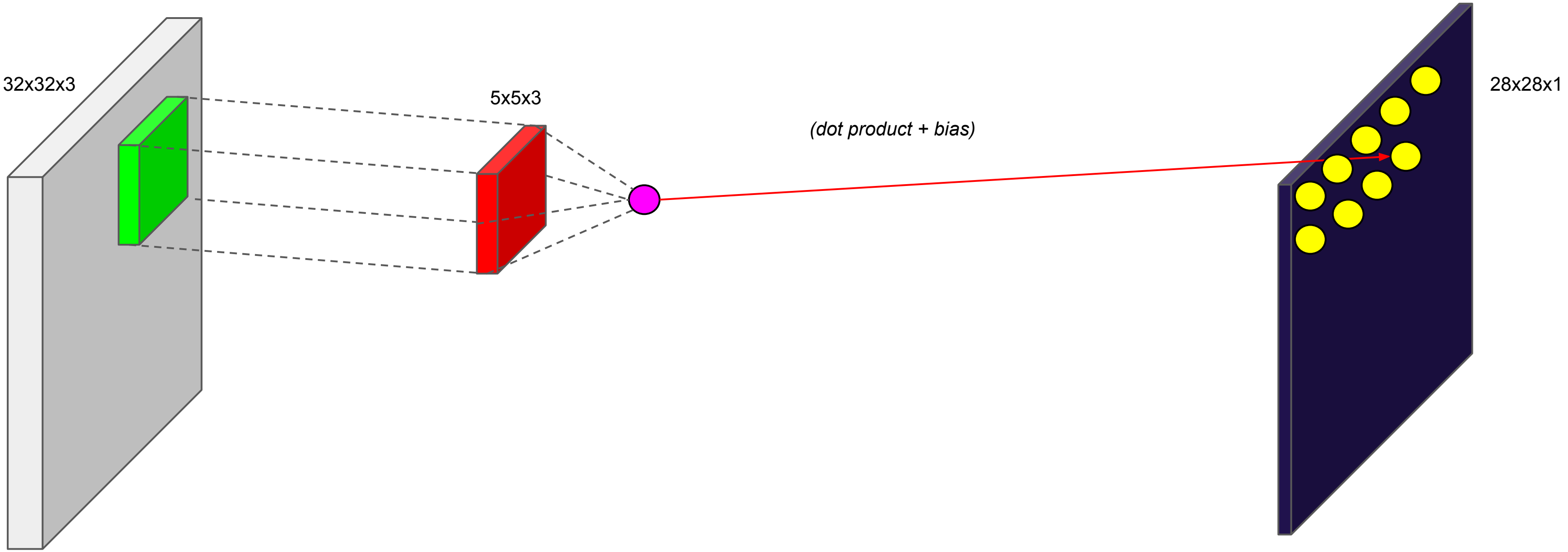
Convolutional Layers

Convolve (slide) over all spatial locations



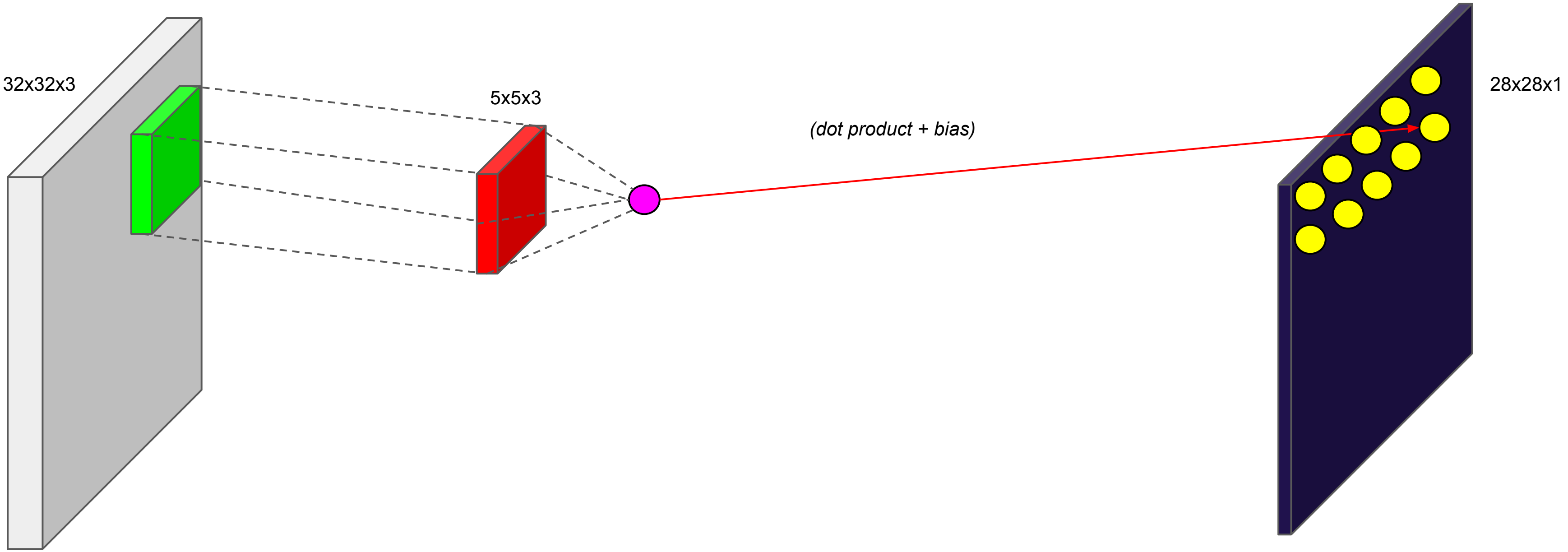
Convolutional Layers

Convolve (slide) over all spatial locations



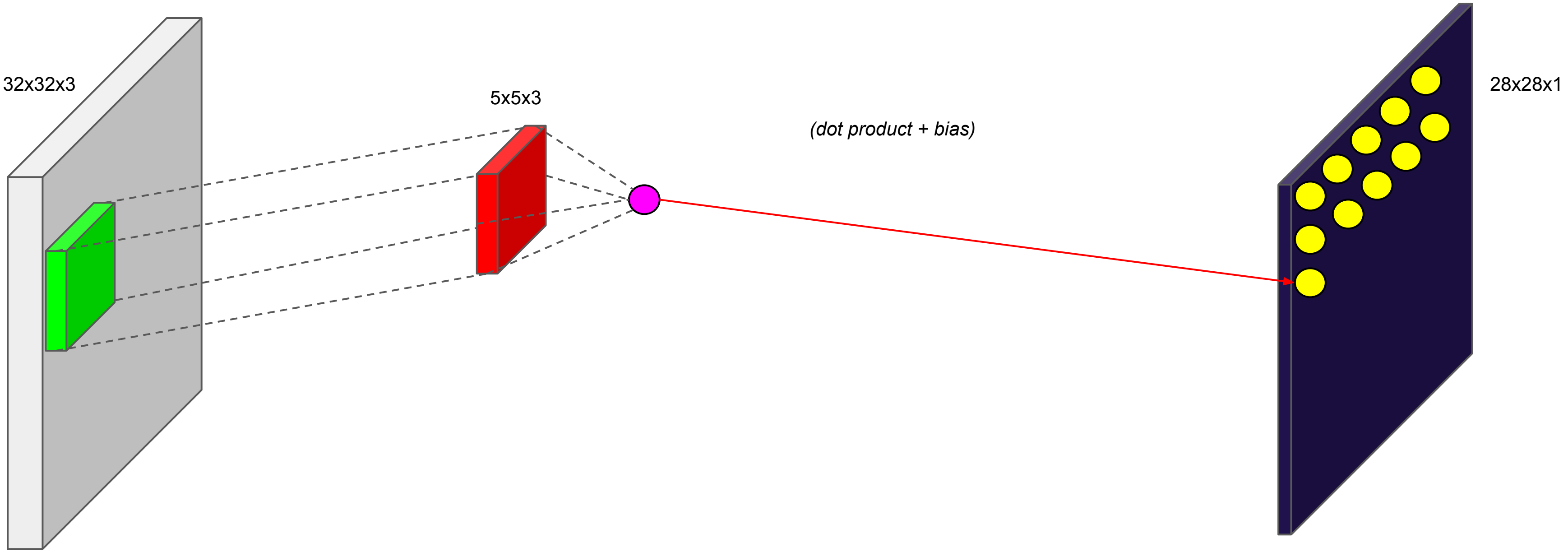
Convolutional Layers

Convolve (slide) over all spatial locations



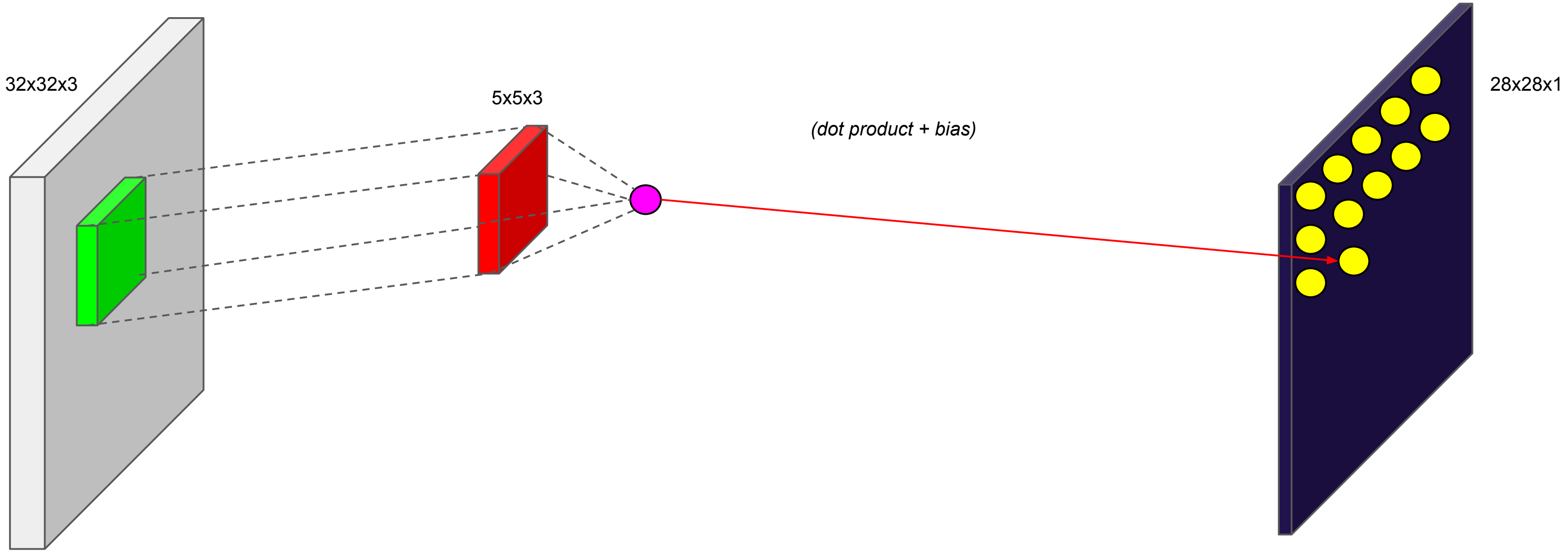
Convolutional Layers

Convolve (slide) over all spatial locations



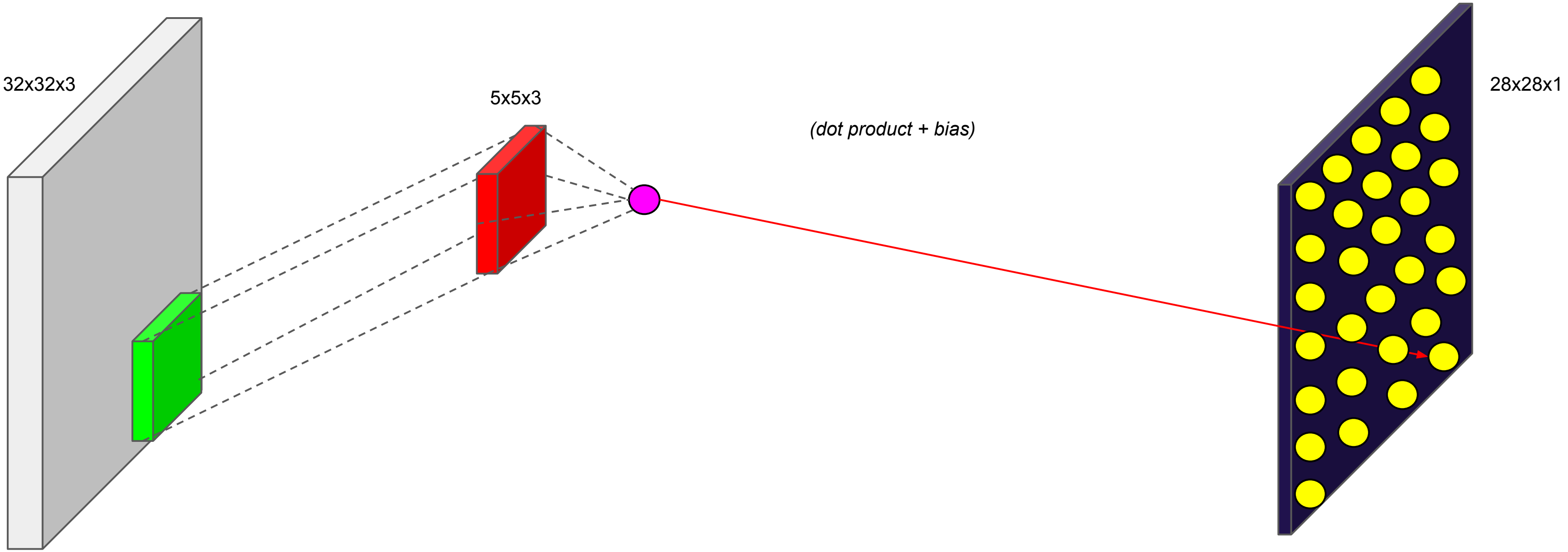
Convolutional Layers

Convolve (slide) over all spatial locations



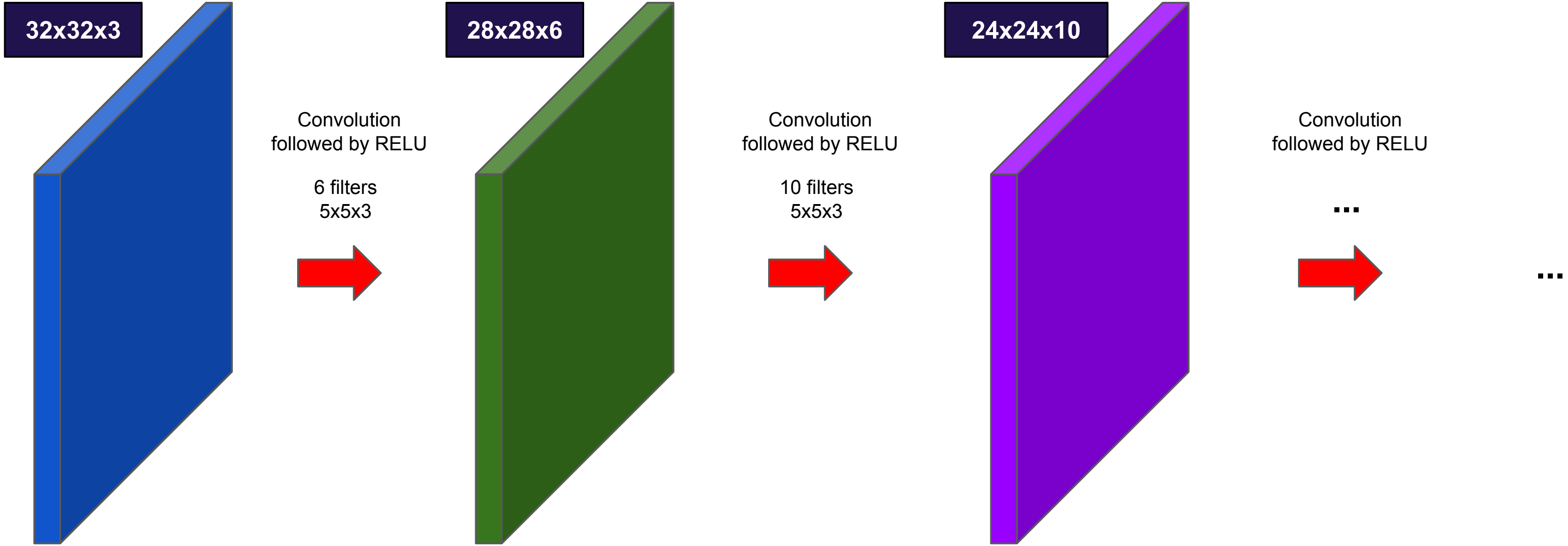
Convolutional Layers

Convolve (slide) over all spatial locations



Convolutional Layers

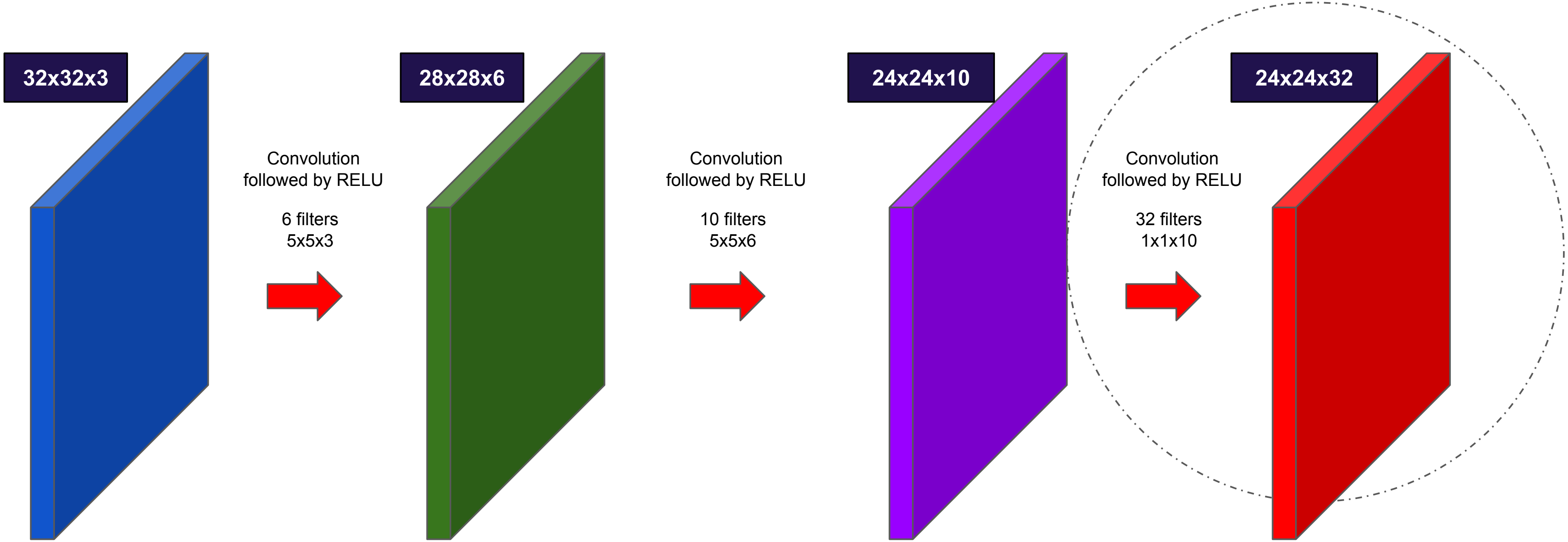
The sample 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially ***(32 → 28 → 24)***



Shrinking too fast is not good. It doesn't work well.

Convolutional Layers

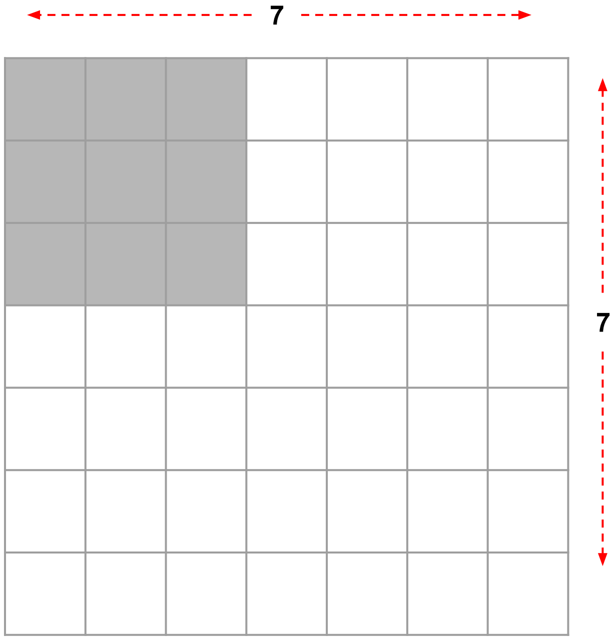
Hey!   1x1 convolutions make perfect sense



Shrinking too fast is not good. It doesn't work well.

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

Stride = the step of the sliding window

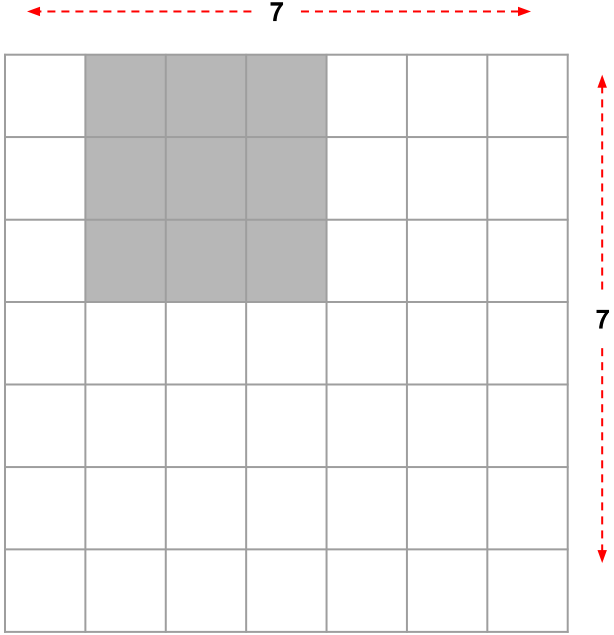
Example: Stride = 1

It means we will move the sliding window one position towards the right.

Movements: 1

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

Stride = the step of the sliding window

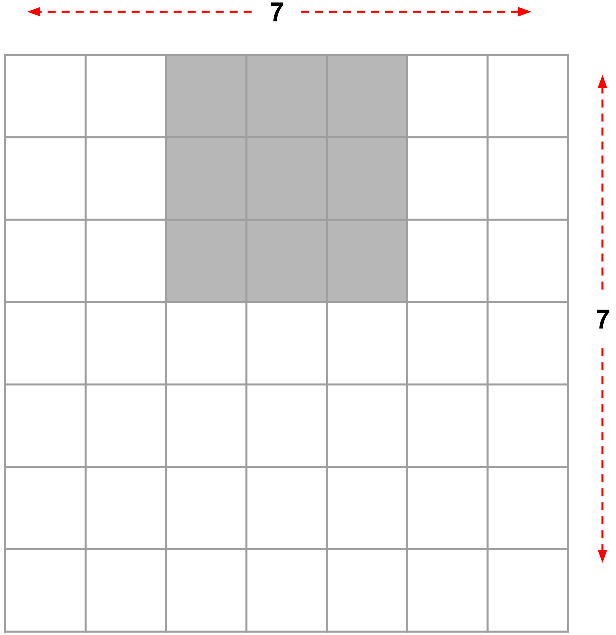
Example: Stride = 1

It means we will move the sliding window one position towards the right.

Movements: 2

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

Stride = the step of the sliding window

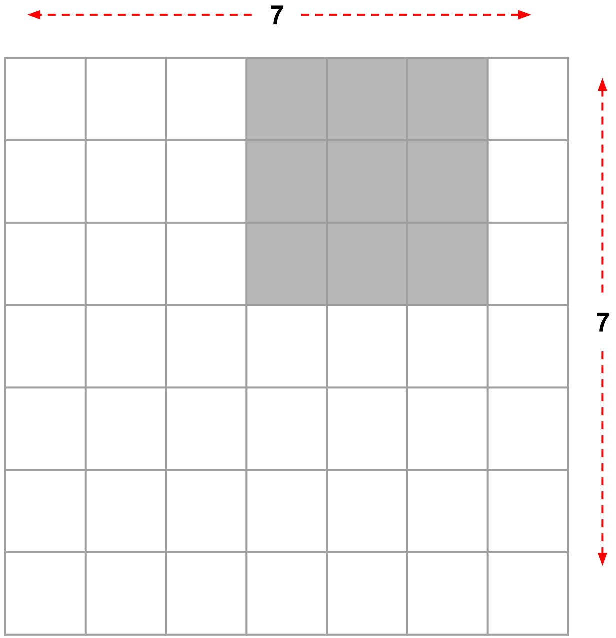
Example: Stride = 1

It means we will move the sliding window one position towards the right.

Movements: 3

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

Stride = the step of the sliding window

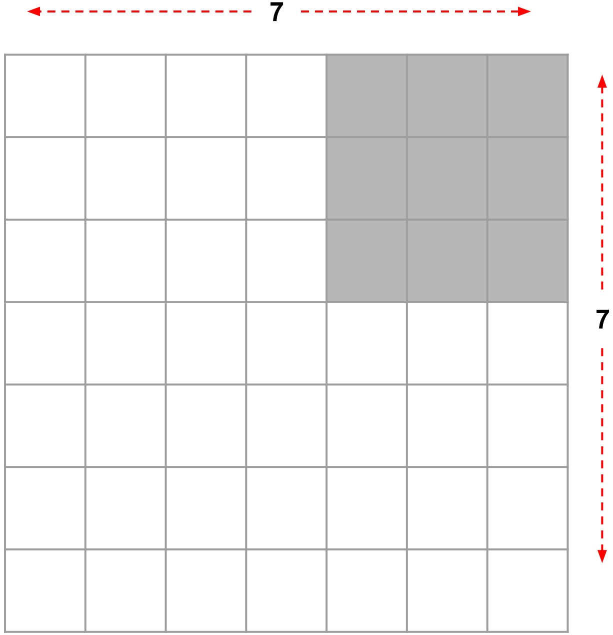
Example: Stride = 1

It means we will move the sliding window one position towards the right.

Movements: 4

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

Stride = the step of the sliding window

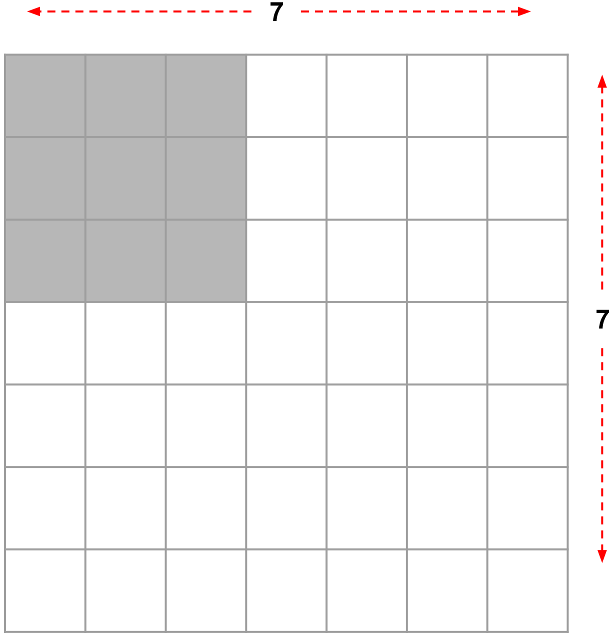
Example: Stride = 1

It means we will move the sliding window one position towards the right.

Output Size: 5x5

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

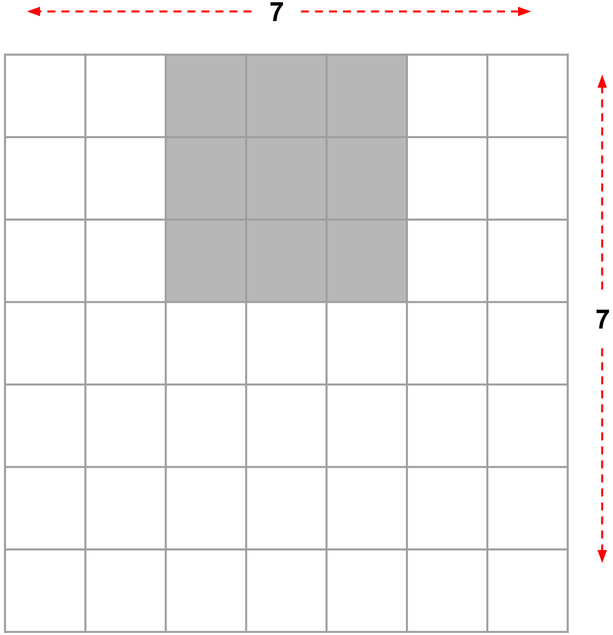
Example: Stride = 2

It means we will move the sliding window two positions towards the right.

Movements: 1

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

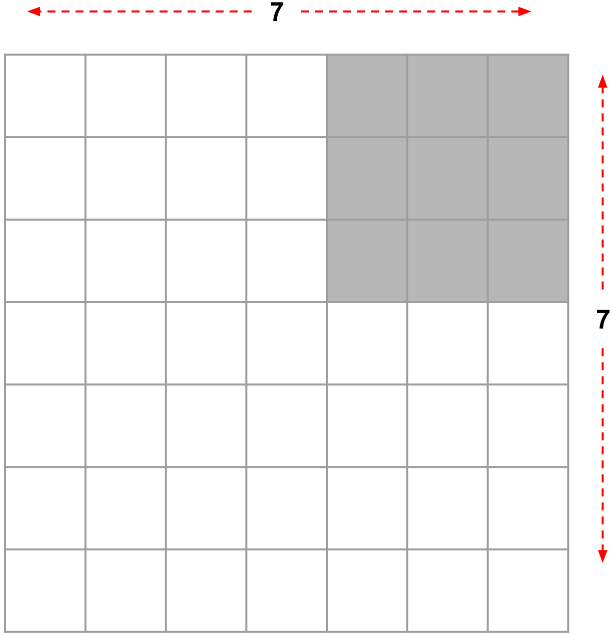
Example: Stride = 2

It means we will move the sliding window two positions towards the right.

Movements: 2

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

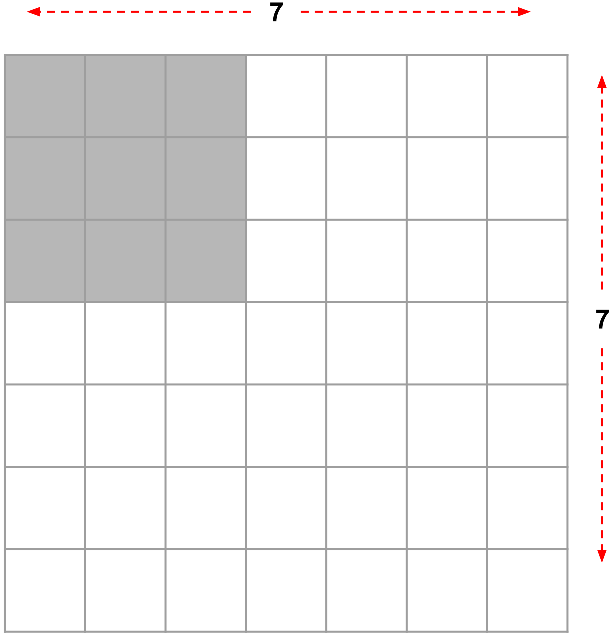
Example: Stride = 2

It means we will move the sliding window two positions towards the right.

Output Size: 3x3

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

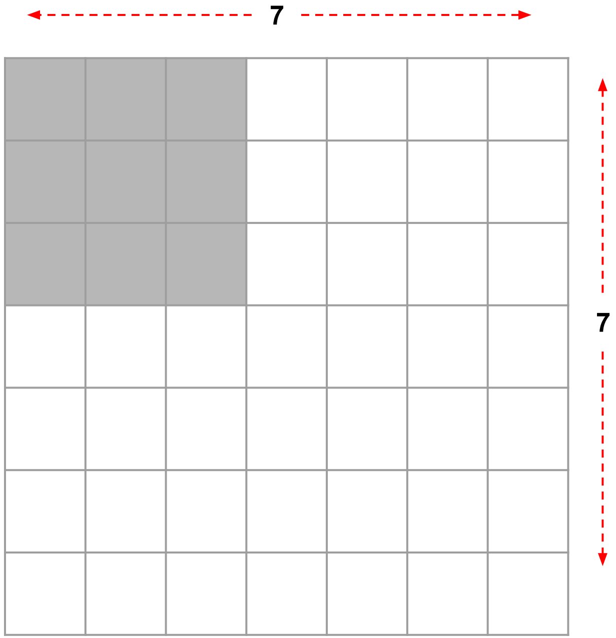
Example: Stride = 3

It means we will move the sliding window three positions towards the right.

Output Size: ?

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

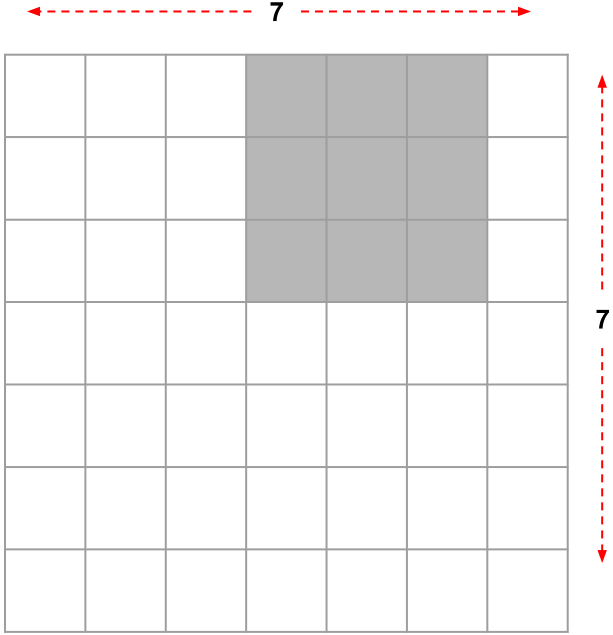
Example: Stride = 3

It means we will move the sliding window three positions towards the right.

Movements: 1

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

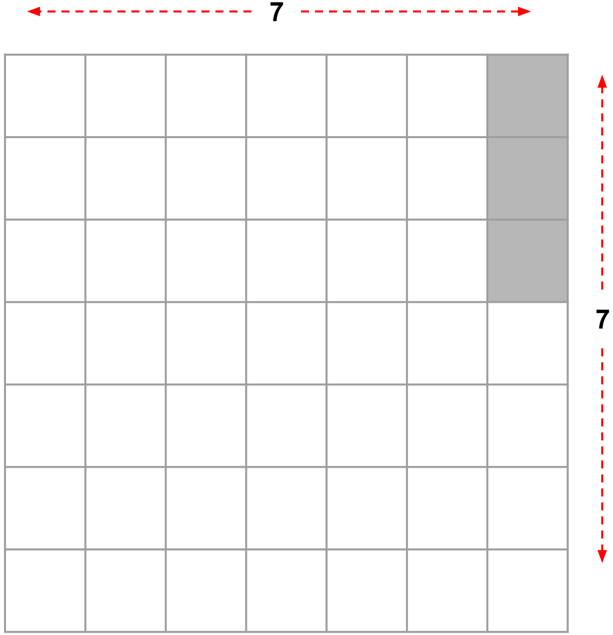
Example: Stride = 3

It means we will move the sliding window three positions towards the right.

Movements: 2

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

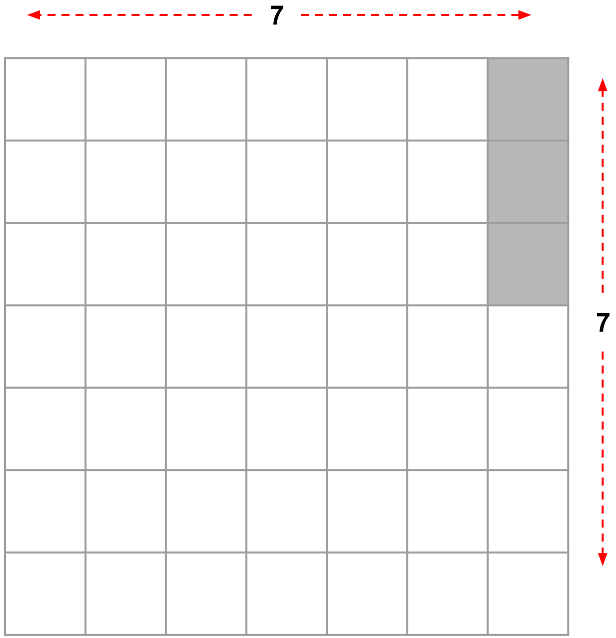
Example: Stride = 3

It means we will move the sliding window three positions towards the right.

Movements: 0.3333

How to calculate the output size

Calculating the output size



- Suppose our input patch is: 7x7
- Suppose the filter size is: 3x3

Example: Stride = 3

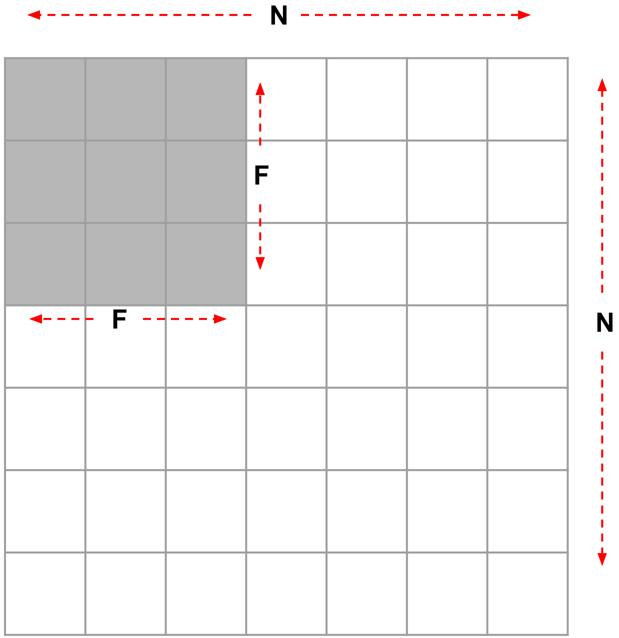
It means we will move the sliding window three positions towards the right.

It doesn’t fit !!!

I cannot apply a 3x3 filter on a 7x7 input with stride 3

How to calculate the output size

Calculating the output size



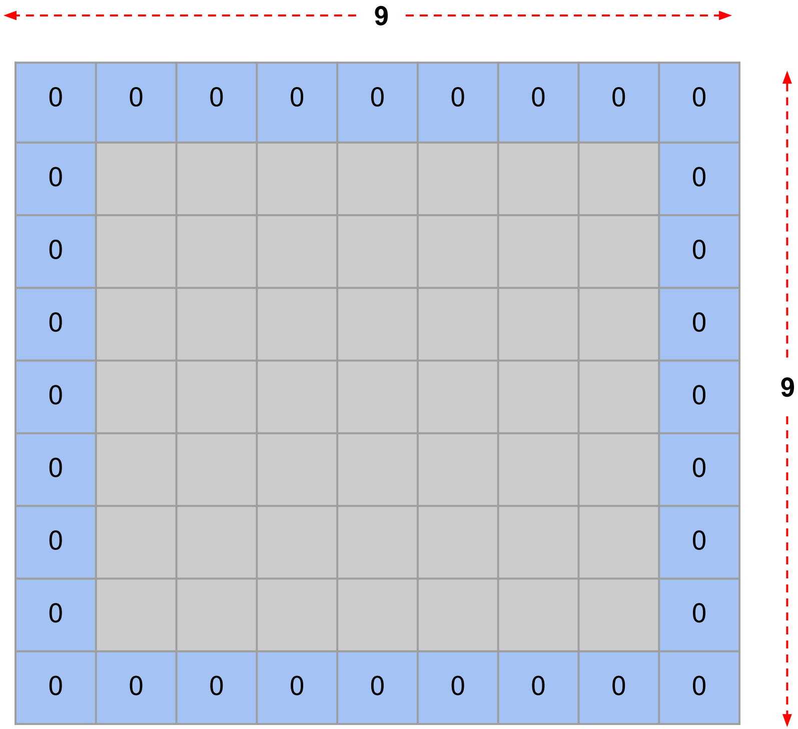
Output size = (N - F) / stride + 1

Example: N = 7 F = 3

- ***Stride 1 → (7 - 3) / 1 + 1 = 5 → Output: 5x5***
- ***Stride 2 → (7 - 3) / 2 + 1 = 3 → Output: 3x3***
- ***Stride 3 → (7 - 3) / 3 + 1 = 2.33 → Doesn't fit !***

Padding

Zero Padding



0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Example

- *Input: 7x7*
- *Filter: 3x3*
- *Stride: 1*
- *Padding: 1*

Output size = (N - F) / stride + 1

⇩

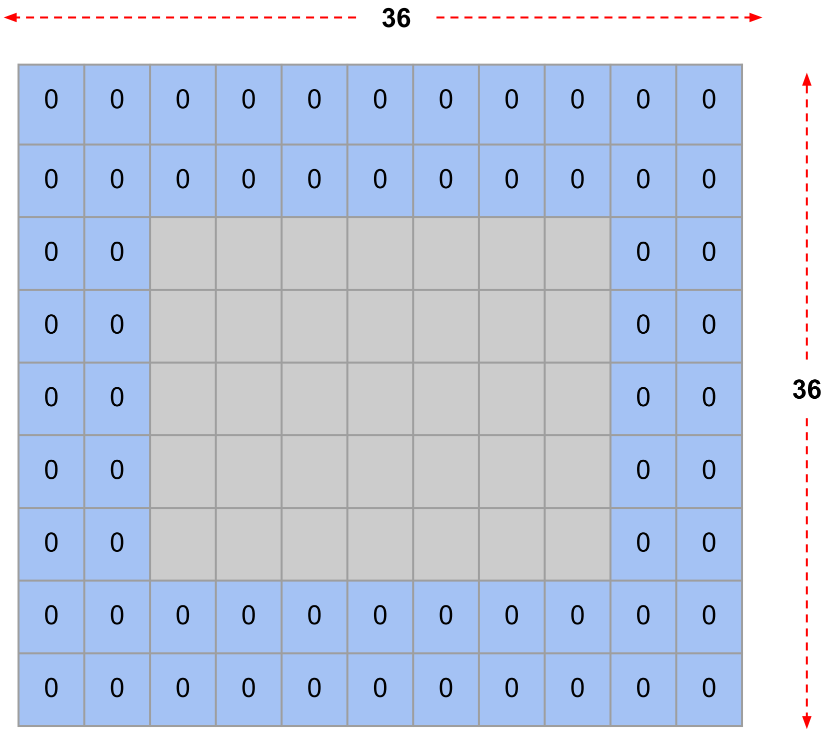
Output size = (7 + 2 - 3) / 1 + 1 *(padding applied)*

⇩

Output Size: 7x7

Padding

Zero Padding *(imagine the matrix below is 36x36)*



0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0								0	0
0	0								0	0
0	0								0	0
0	0								0	0
0	0								0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Example

- *Input: 32x32*
- *Filter: 5x5*
- *Stride: 1*
- *Padding: 2*

Output size = (N - F) / stride + 1

⇩

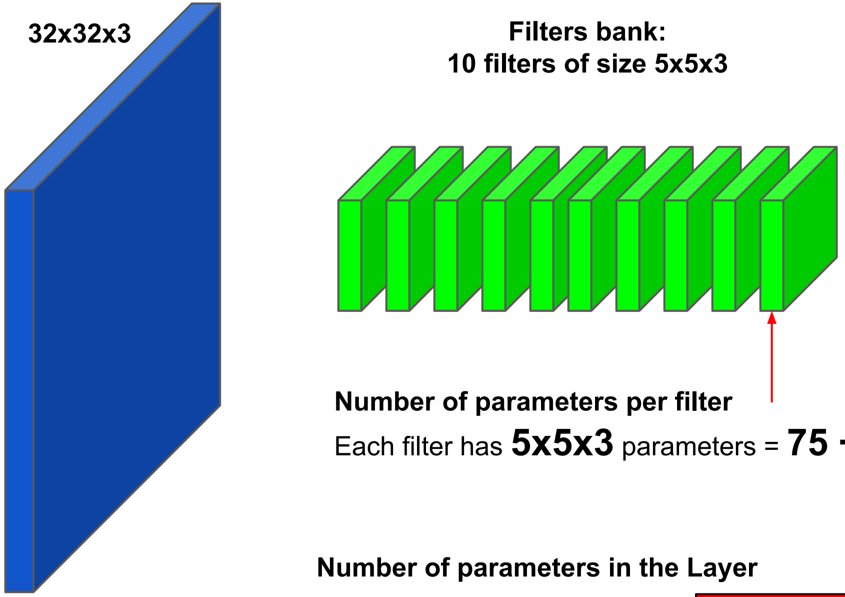
Output size = (32 + 4 - 5) / 1 + 1 *(padding applied)*

⇩

Output Size: 32x32

How to calculate the *Number of Parameters*

Calculating the number of parameters



- Input Volume: **32x32x3**
- **10** filters of size **5x5x3**
- Stride 1, padding 2

Number of parameters per filter

Each filter has **5x5x3** parameters = **75 + 1** *(plus one for the bias)* = **76** parameters

Number of parameters in the Layer

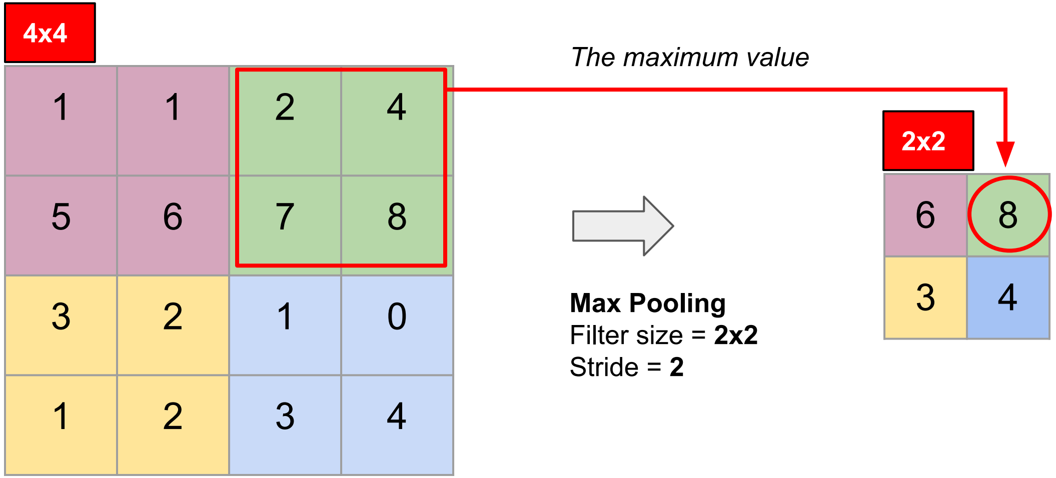
76 parameters x **10** filters = **760 parameters**

Pooling Layers

Makes the representations smaller and more manageable without losing too much information

Max Pooling *(most common)*

Get the max value in the neighborhood, and pass it forward.



4x4

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

The maximum value

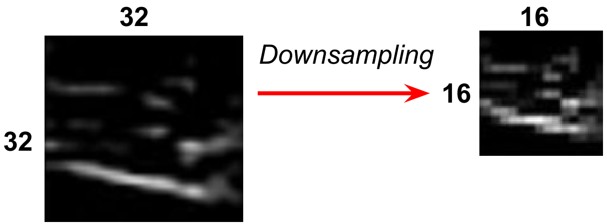
2x2

6	8
3	4

Max Pooling
Filter size = **2x2**
Stride = **2**

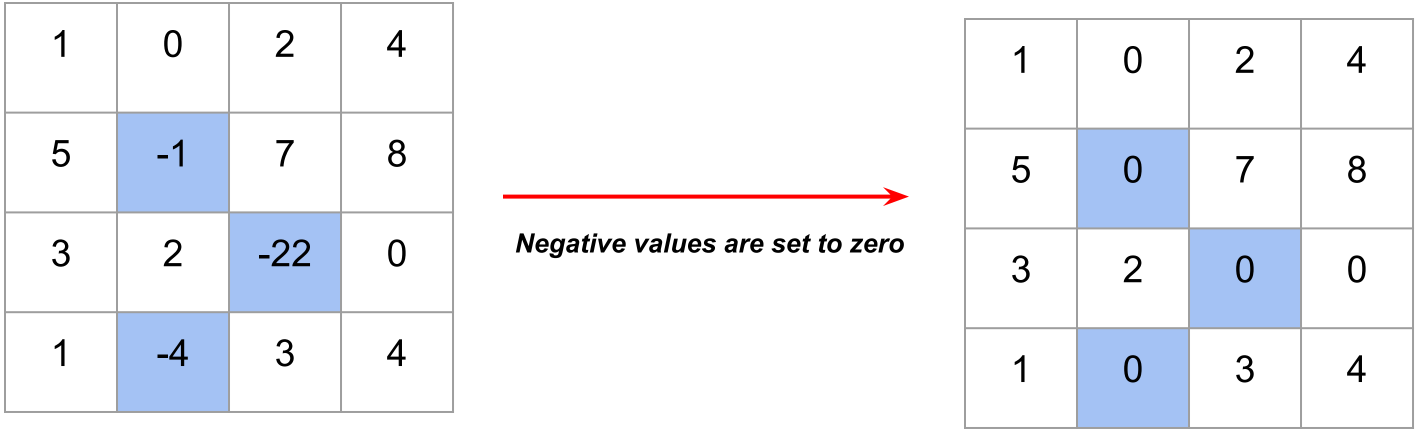
Types

- **Max Pooling**
- **Average Pooling**
- *etc*



ReLU Layers

Think like a Proton → Stay positive → If negative, then set to zero.



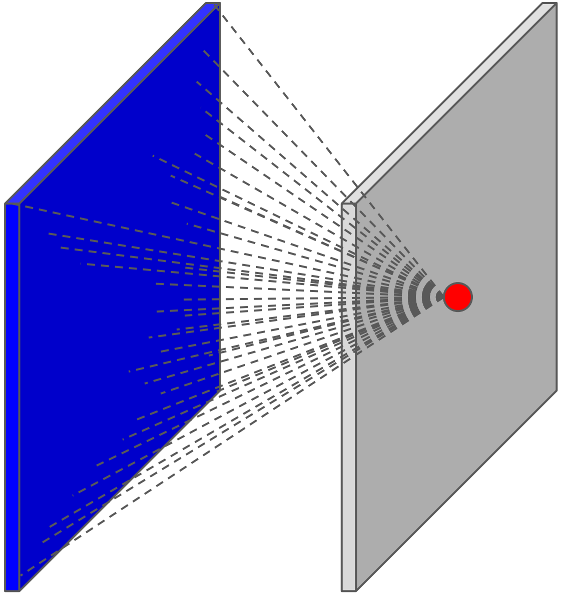
1	0	2	4
5	-1	7	8
3	2	-22	0
1	-4	3	4

Negative values are set to zero

1	0	2	4
5	0	7	8
3	2	0	0
1	0	3	4

Fully Connected Layers

Contain neurons that connect to the entire input volume, as in ordinary *Multi-Layered Perceptrons*

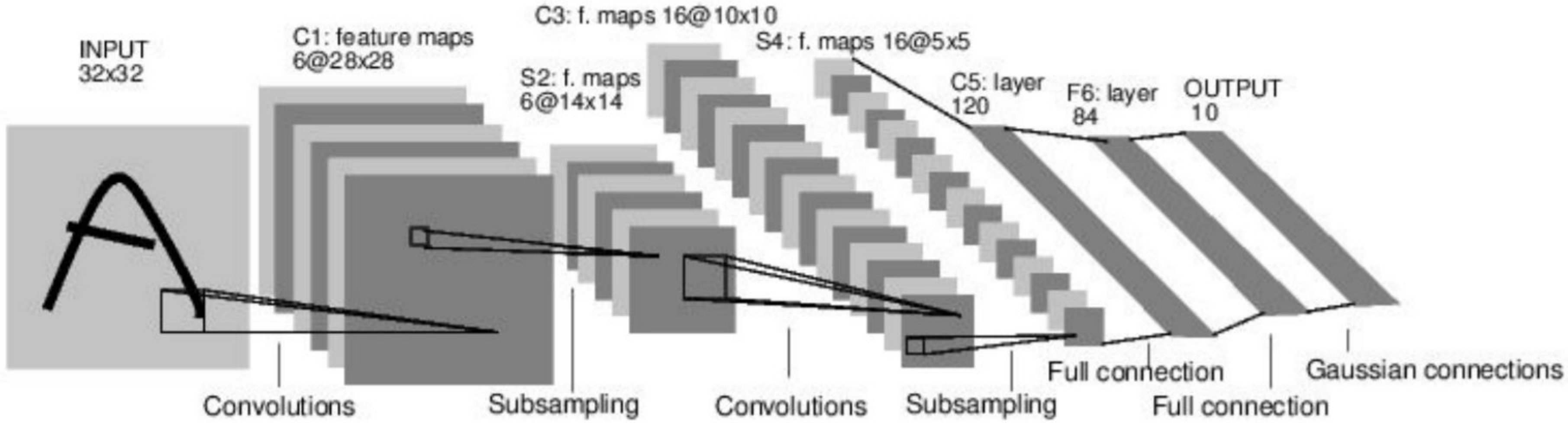


This neuron is connected to all of the neurons available in the previous layer

Architectures

LeNet-5

[LeCun et al., 1998]

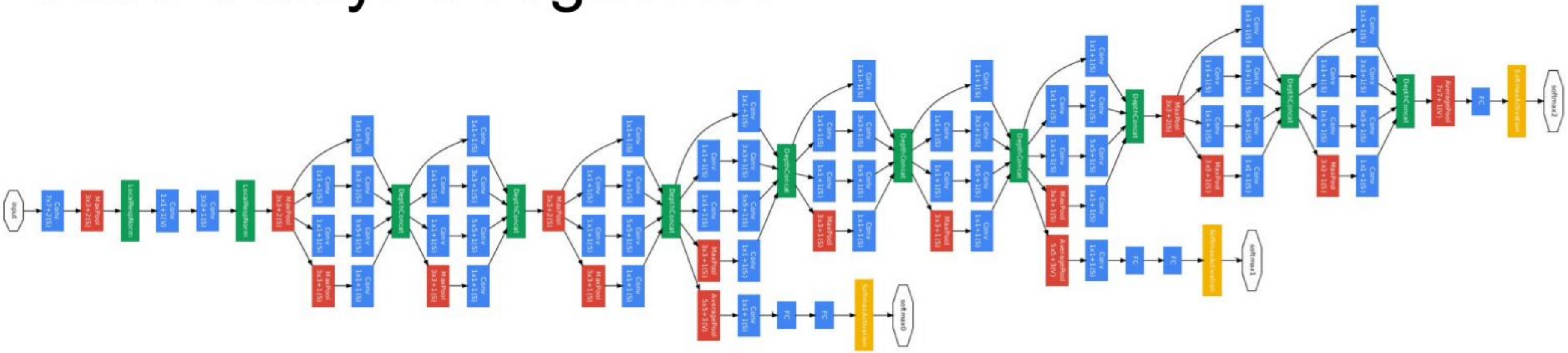


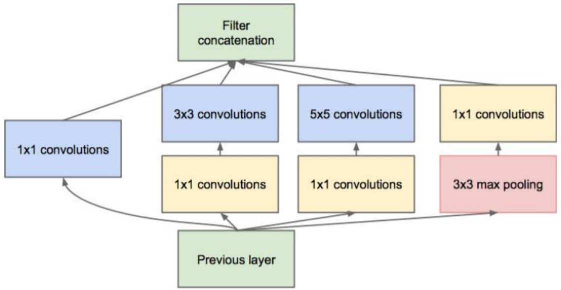
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2

Architectures

Case Study: GoogLeNet

[Szegedy et al., 2014]





Filter concatenation

1x1 convolutions

3x3 convolutions

5x5 convolutions

1x1 convolutions

1x1 convolutions

1x1 convolutions

3x3 max pooling

Previous layer

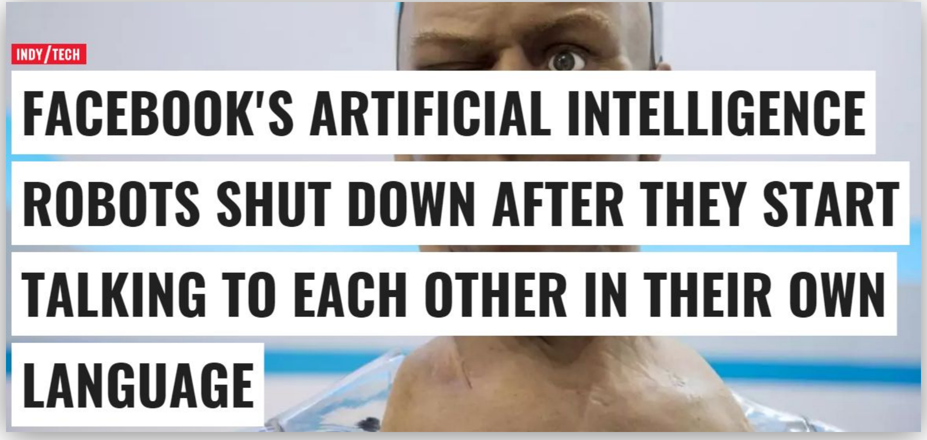
Inception module

ILSVRC 2014 winner (6.7% top 5 error)

Can networks become creative?

As seen on the Press / TV

Computers can now paint like Van Gogh and Picasso



INDY/TECH

FACEBOOK'S ARTIFICIAL INTELLIGENCE ROBOTS SHUT DOWN AFTER THEY START TALKING TO EACH OTHER IN THEIR OWN LANGUAGE

Artificial Intelligence

Google's AI has written some amazingly mournful poetry

Google's poetry was written by an AI system after it was fed thousands of unpublished romantic novels

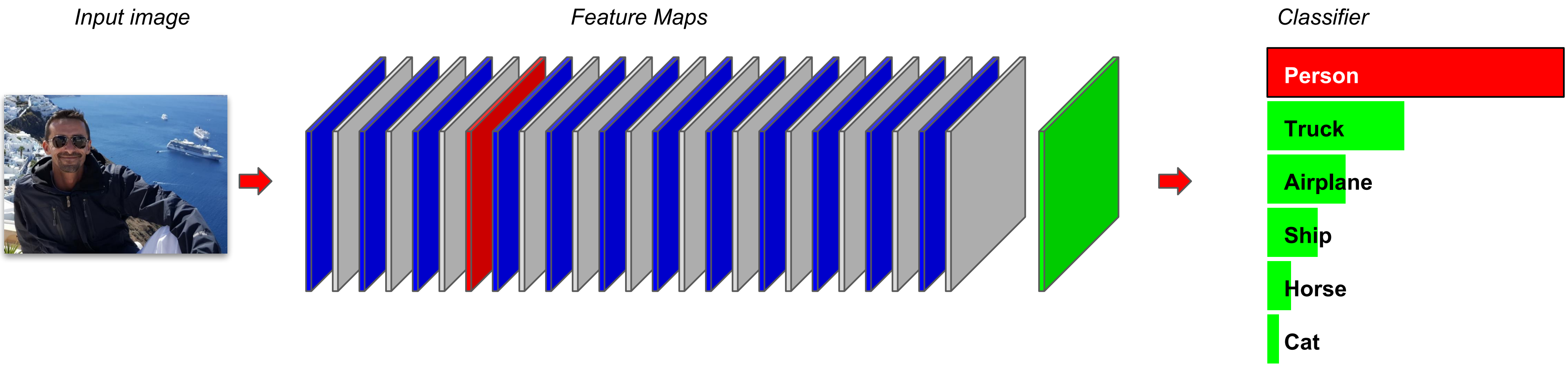
Can networks become creative?

Yes.

More science than fiction.

Generating images

Our network learned to identify people, cars, cats and other classes.



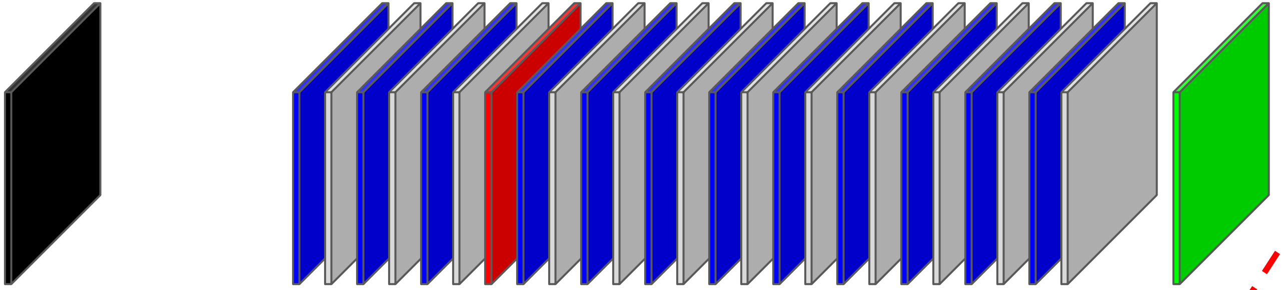
Can we generate a new image from scratch?

Generating images

The goal is to maximize the score of a given class (e.g. “Person”)

Input image

Gradients of the Score vector



*(This is a matrix.
All values were
set to zero)*

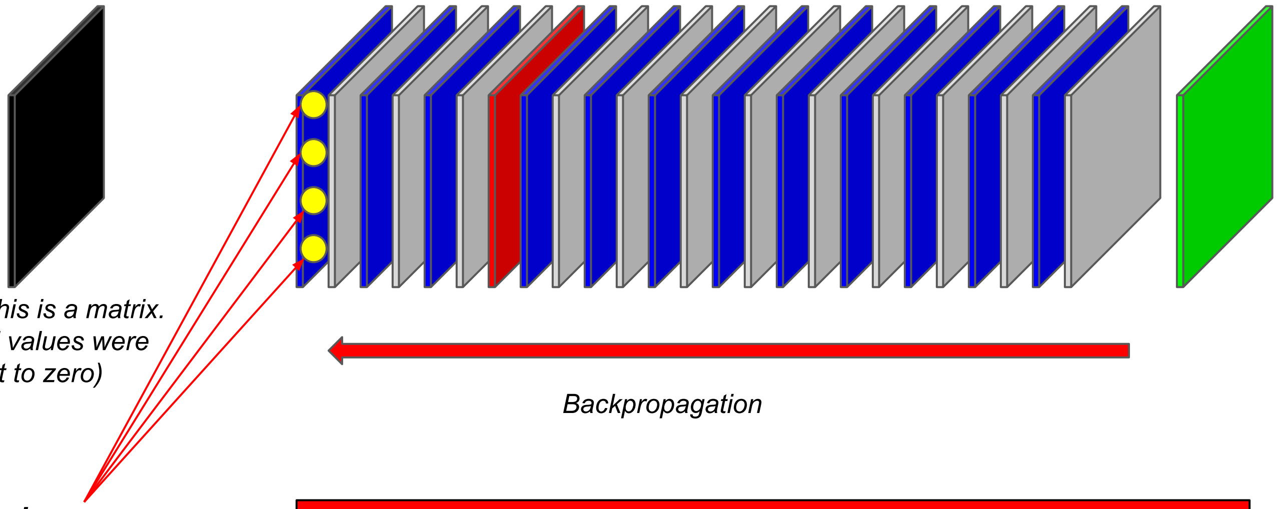
1	**Person**
0	**Truck**
0	**Airplane**
0	**Ship**
0	**Horse**
0	**Cat**

Step #1:
Set the gradient of the scores vector to be [0, 0,....1,....,0]

Generating images

The goal is to maximize the score of a given class (e.g. “Person”)

Input image



(This is a matrix.
All values were
set to zero)

Backpropagation

Hey!
I’ve got some
gradients here!

Gradients of the Score vector

1	**Person**
0	**Truck**
0	**Airplane**
0	**Ship**
0	**Horse**
0	**Cat**

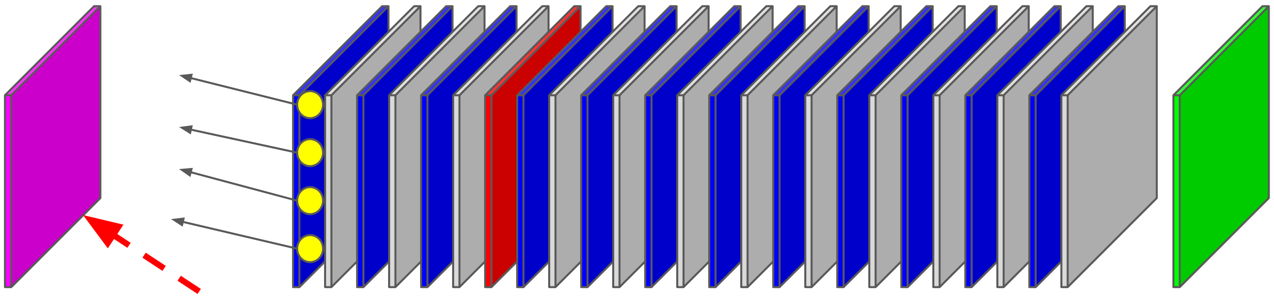
Step #2:
Propagate the error backwards (compute the gradients)

Generating images

The goal is to maximize the score of a given class (e.g. “Person”)

Input image

Gradients of the Score vector



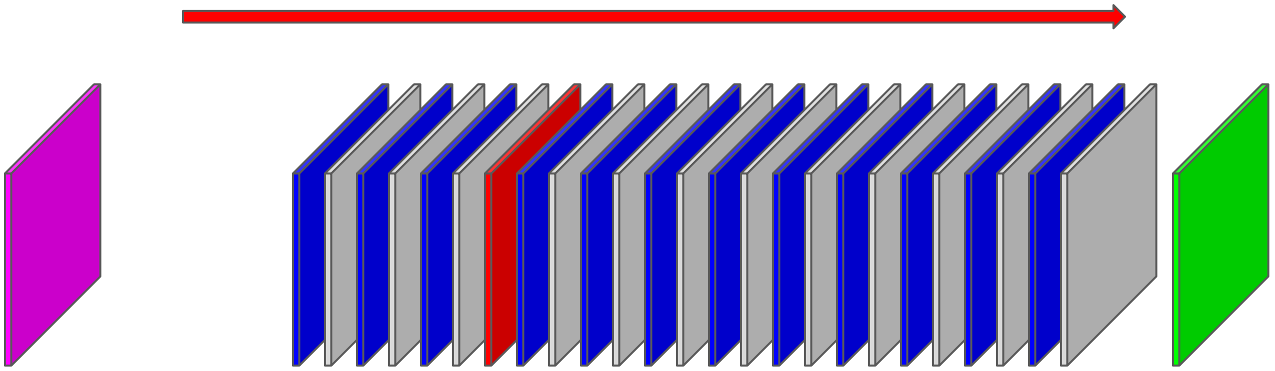
The image is no longer zero

1	Person
0	Truck
0	Airplane
0	Ship
0	Horse
0	Cat

Step #3:
Update the input image

Generating images

The goal is to maximize the score of a given class (e.g. “Person”)



Input image

Forward pass

The image is no longer zero

Gradients of the Score vector

1	**Person**
0	**Truck**
0	**Airplane**
0	**Ship**
0	**Horse**
0	**Cat**

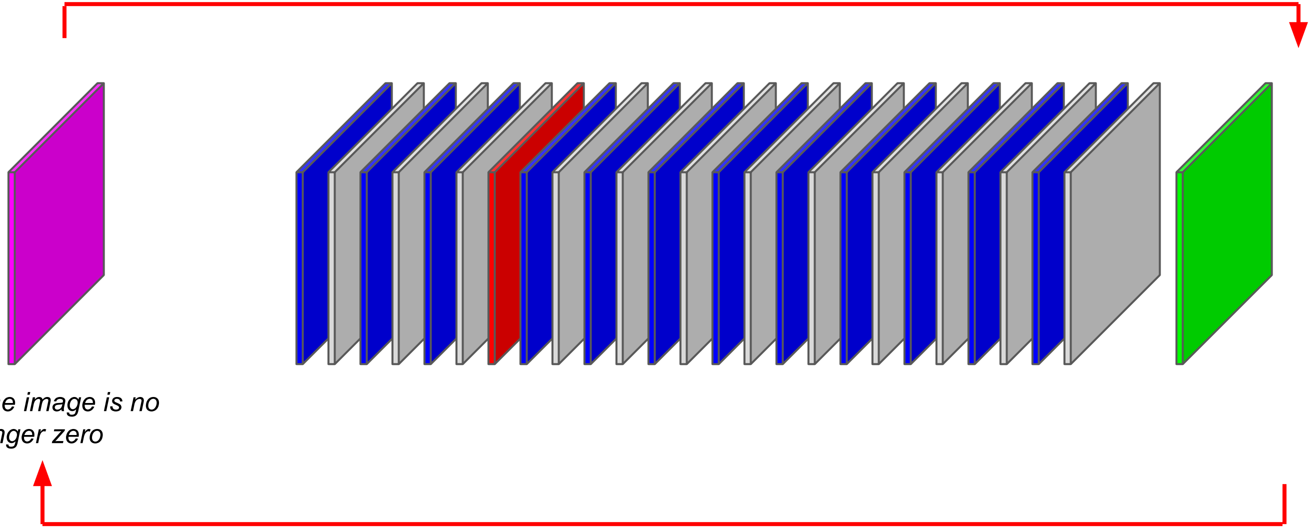
Step #4:
Forward the image through the network.

Generating images

The goal is to maximize the score of a given class (e.g. “Person”)

Input image

Gradients of the Score vector



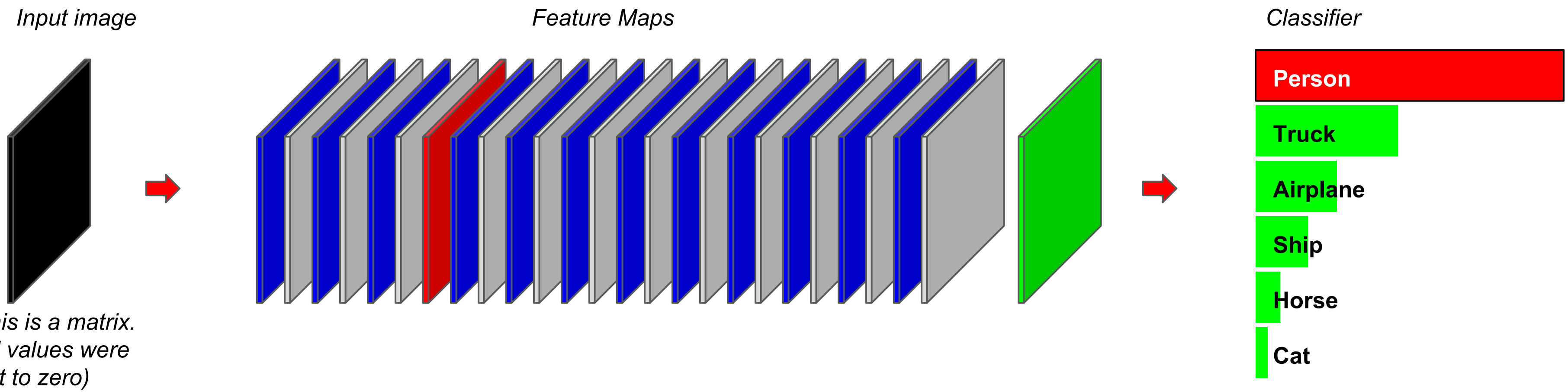
The image is no longer zero

1	**Person**
0	**Truck**
0	**Airplane**
0	**Ship**
0	**Horse**
0	**Cat**

Go back to step #1 (repeat the process again)
Set the gradient of the scores vector to be [0, 0,....1,....,0]

Generating images *(summary)*

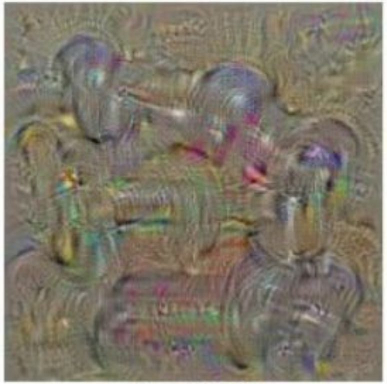
The goal is to maximize the score of a given class (e.g. “Person”)



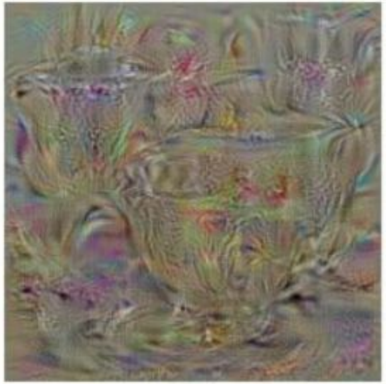
(this is a matrix. All values were set to zero)

Step #1: Set the gradient of the scores vector to be [0, 0,....1,....,0]
Step #2: Backpropagate the error
Step #3: do a small “image update”
Step #4: forward the image through the network.
Step #5: go back to step #1

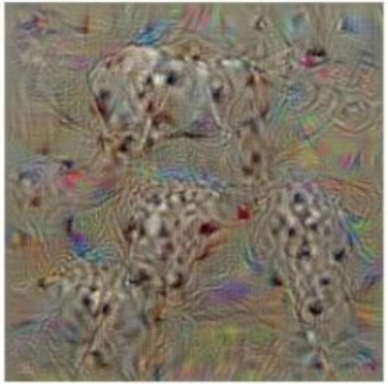
Generating images *(examples)*



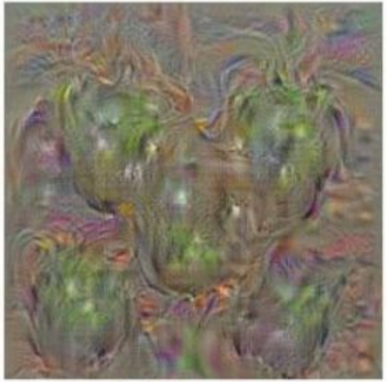
dumbbell



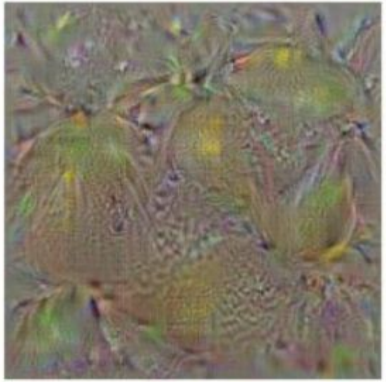
cup



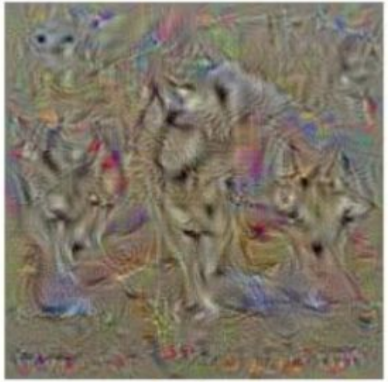
dalmatian



bell pepper



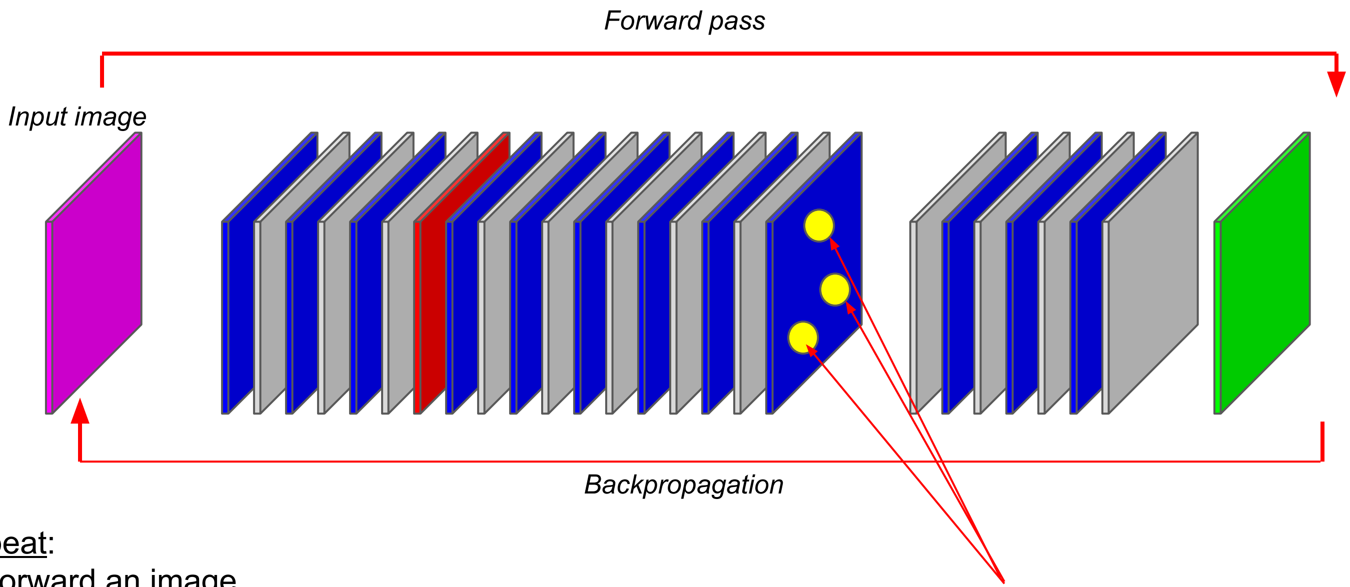
lemon



husky

Generating images

Hey! We could also boost specific activations and neurons



Gradients of the Score vector

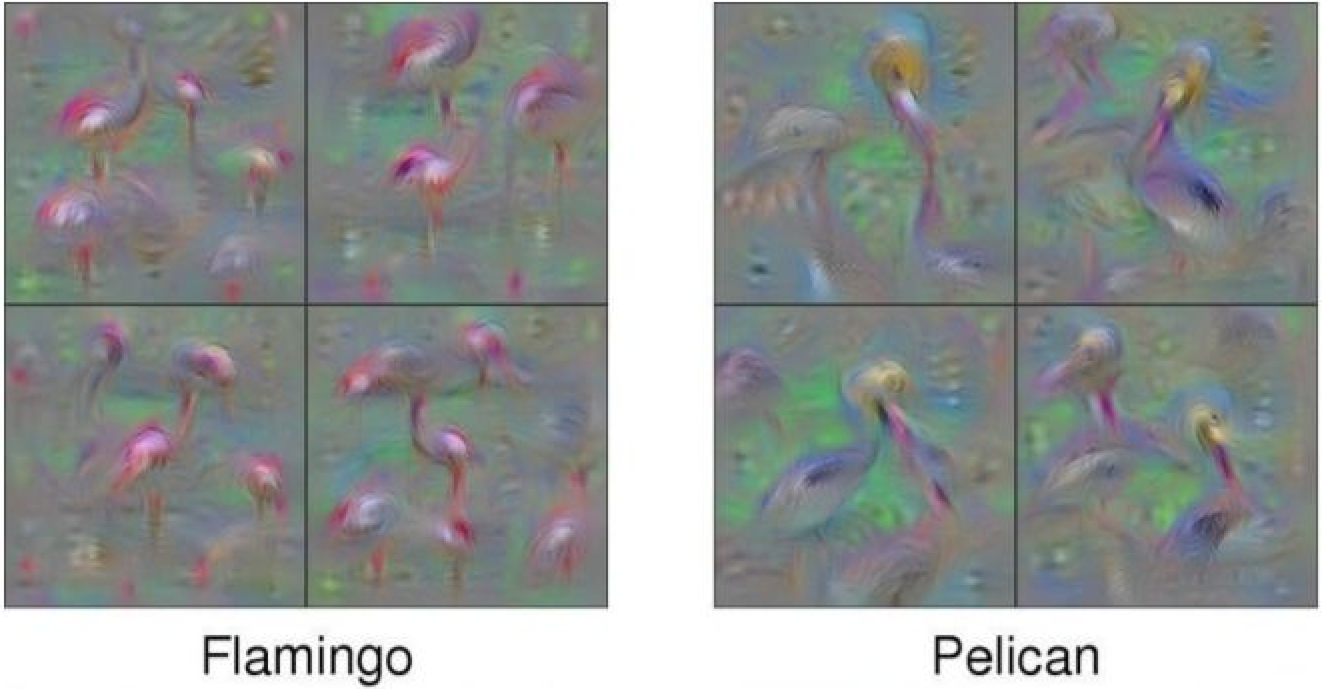
1	**Person**
0	**Truck**
0	**Airplane**
0	**Ship**
0	**Horse**
0	**Cat**

Repeat:

1. Forward an image
2. Set activations in layer of interest to all zero, except for a 1.0 for a neuron of interest
3. Backprop to image
4. Do an “image update”

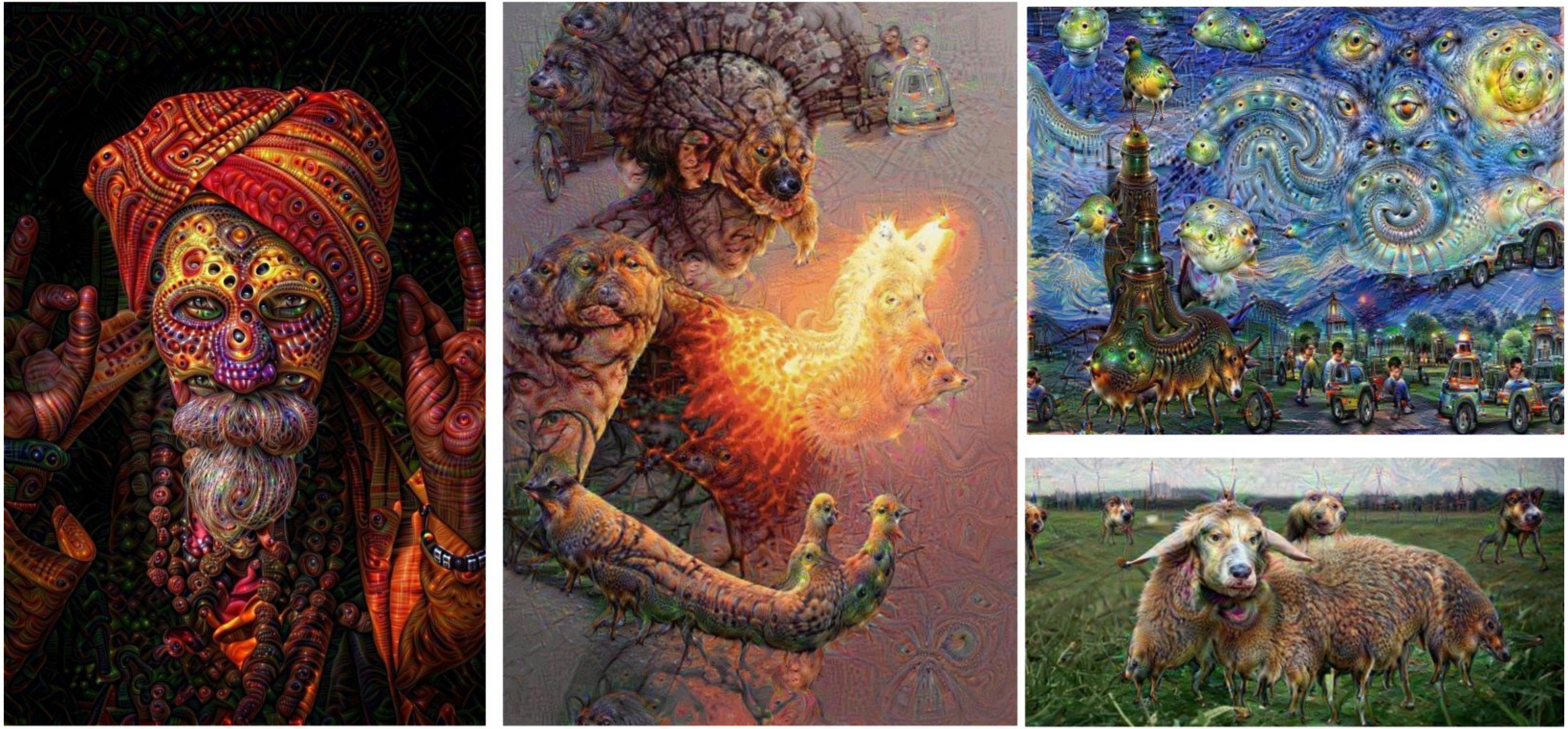
Generating images

Hey!   We could also boost specific activations and neurons



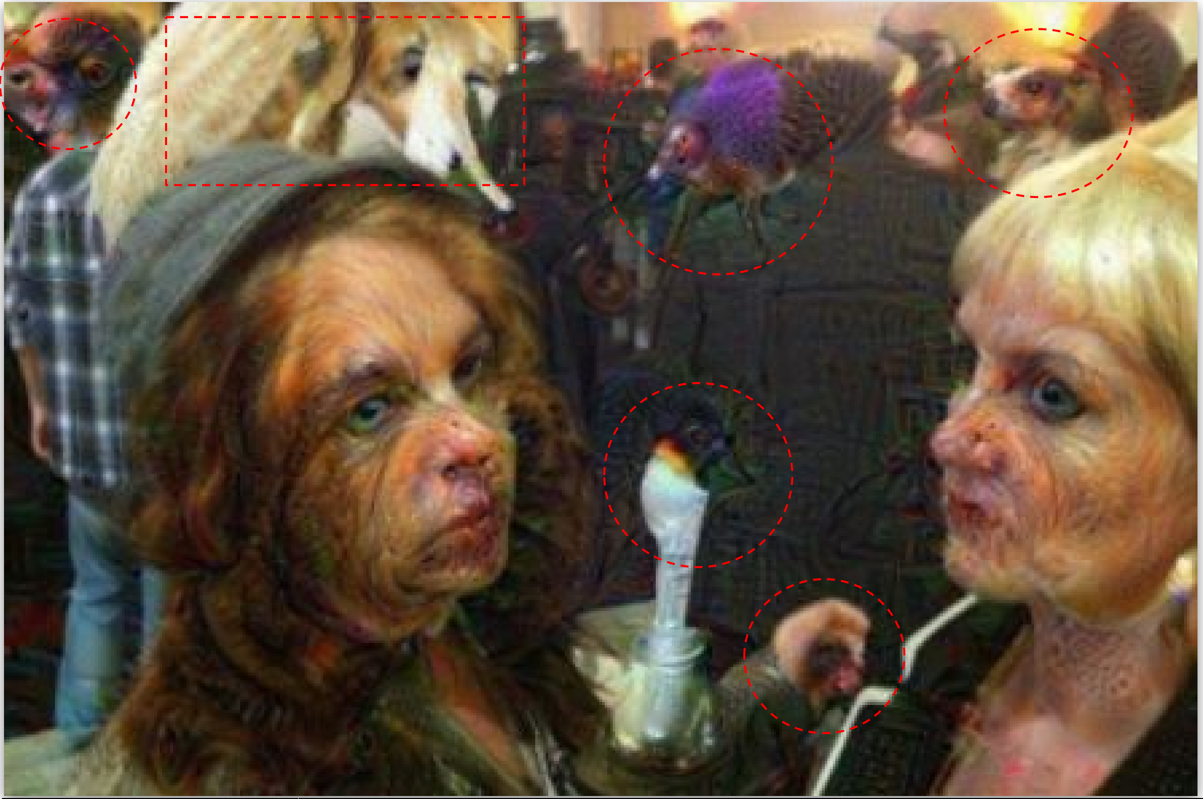
Dreams *(and nightmares)*

It happens when you **boost** *(maximize/minimize)* all activations, at any layer



Dreams *(and nightmares)*

It happens when you boost *(maximize/minimize)* all activations, at any layer



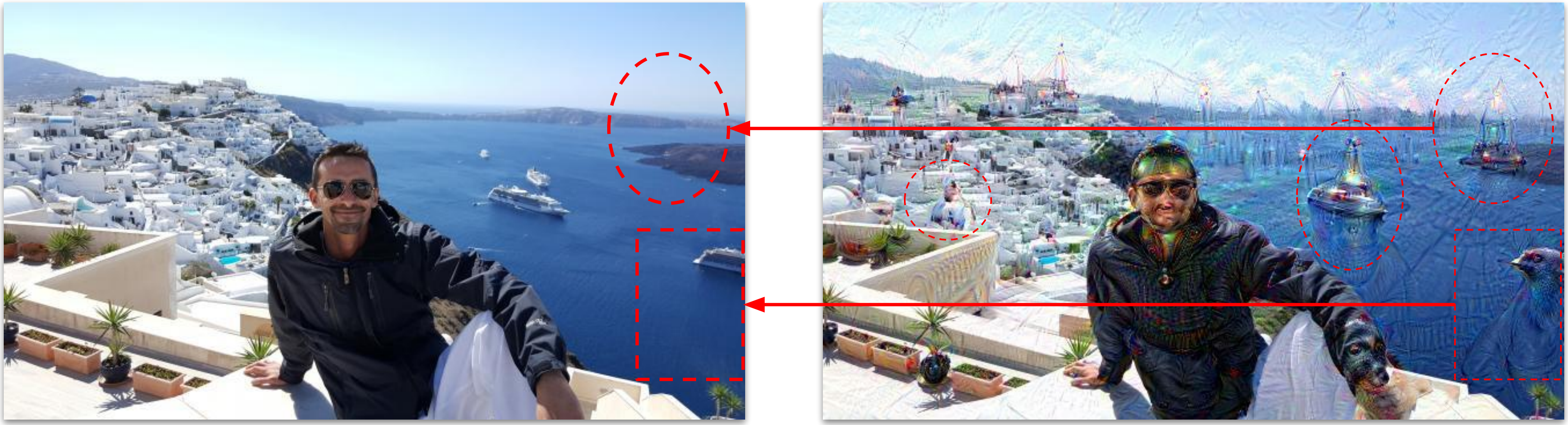
This creates a feedback loop.

For example: any slightly detected dog face will be made more and more dog like over time.

Dreams *(and nightmares)*

It happens when you boost *(maximize/minimize)* all activations, at any layer

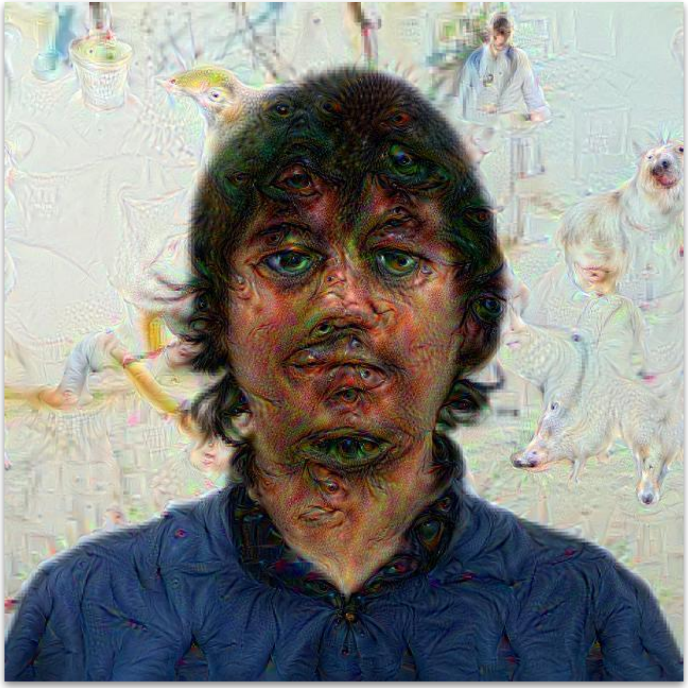
Image learned by the network ➡ *The dream produced by the Neural Network*



Dreams *(and nightmares)*

It happens when you boost *(maximize/minimize)* all activations, at any layer

Image learned by the network → *The dream produced by the Neural Network*

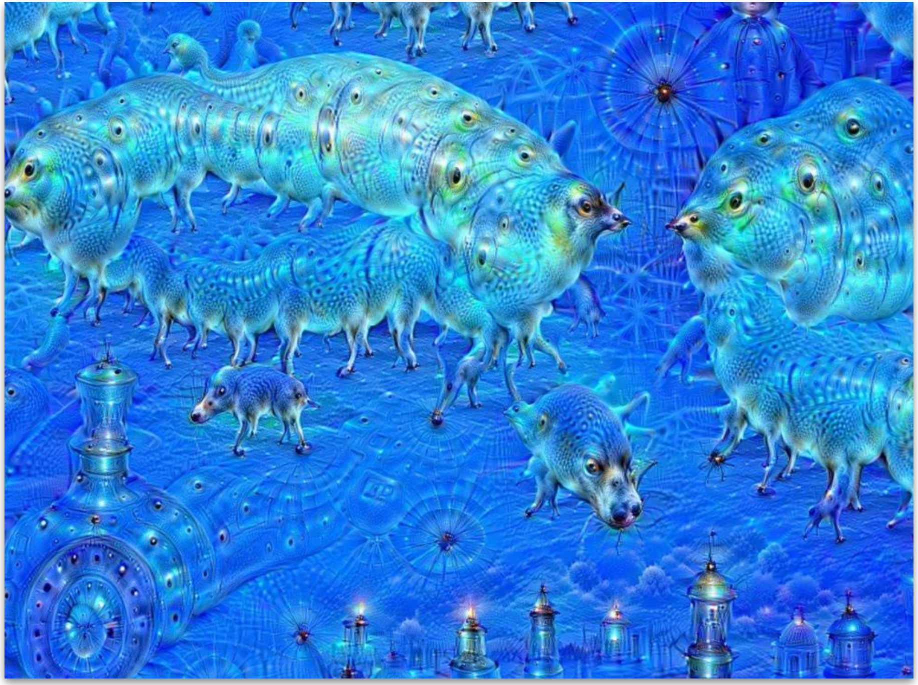


Dreams *(and nightmares)*

It happens when you boost *(maximize/minimize)* all activations, at any layer

Image learned by the network

The dream produced by the Neural Network



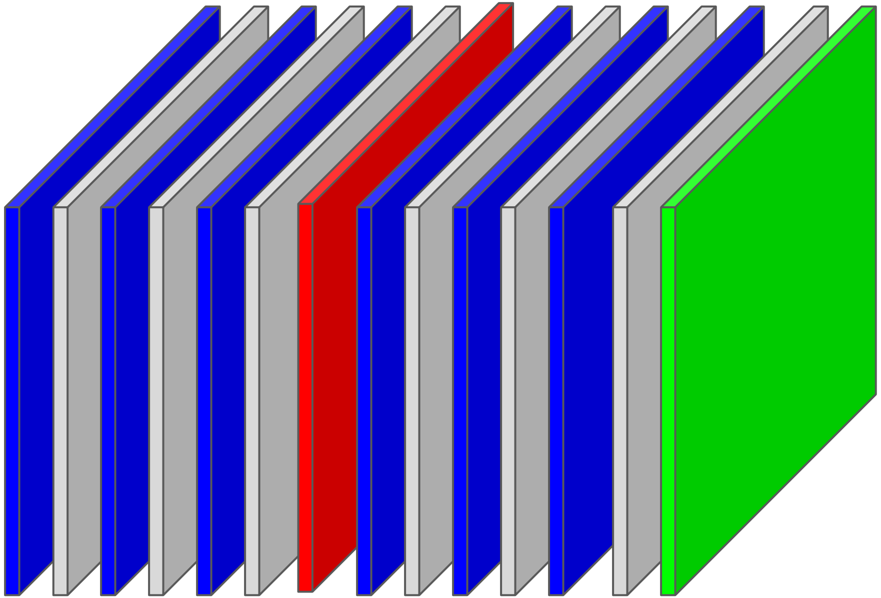
Dreams *(and nightmares)*

How it works

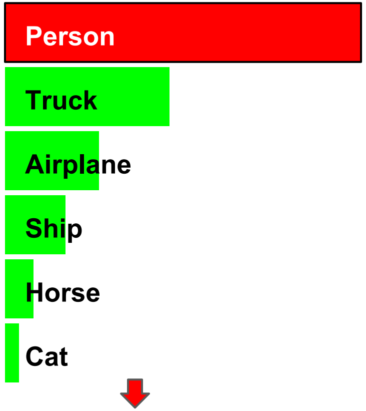
The network has been trained to identify things like *cars, cats, people, airplanes and other objects.*

Input image

Feature Maps



Classifier



"This is a person"

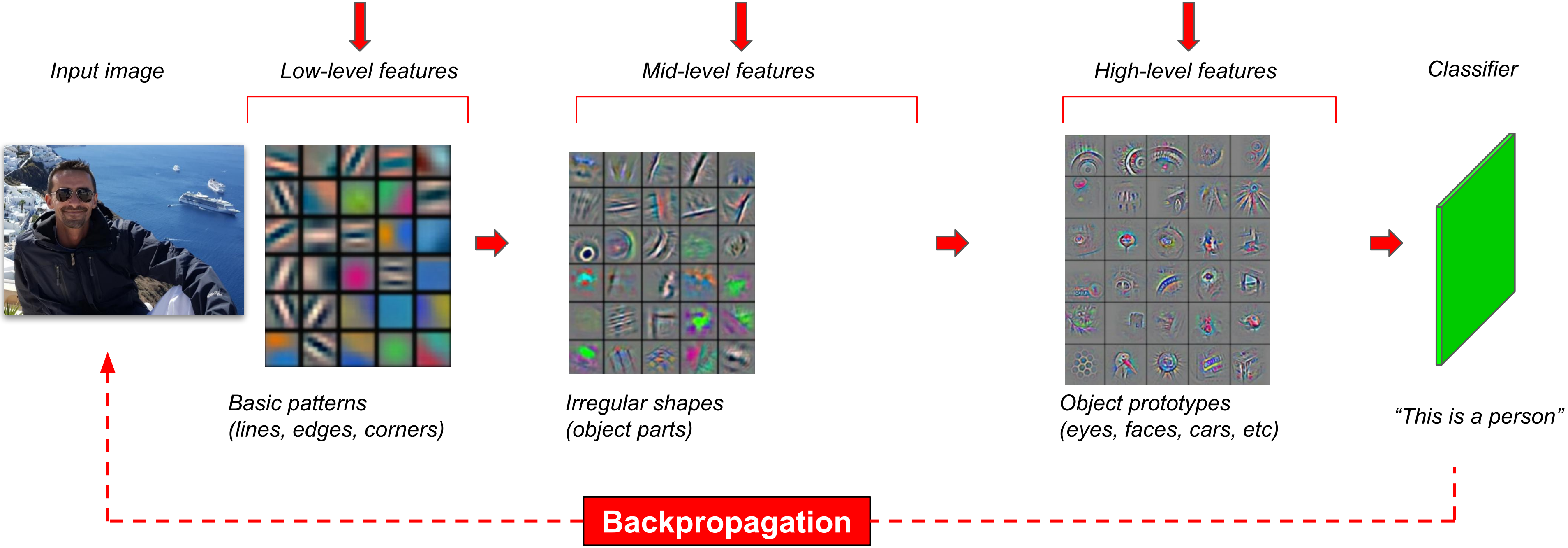
🟦 **Conv** ⬜ **ReLU**

🟥 **Pooling** 🟩 **Fully Connected**

Dreams *(and nightmares)*

How it works

It knows *<u>what filters need to be applied</u>* in order to produce each of the feature maps below.
It learned that through backpropagation.



Input image

Low-level features

Mid-level features

High-level features

Classifier

Basic patterns
(lines, edges, corners)

Irregular shapes
(object parts)

Object prototypes
(eyes, faces, cars, etc)

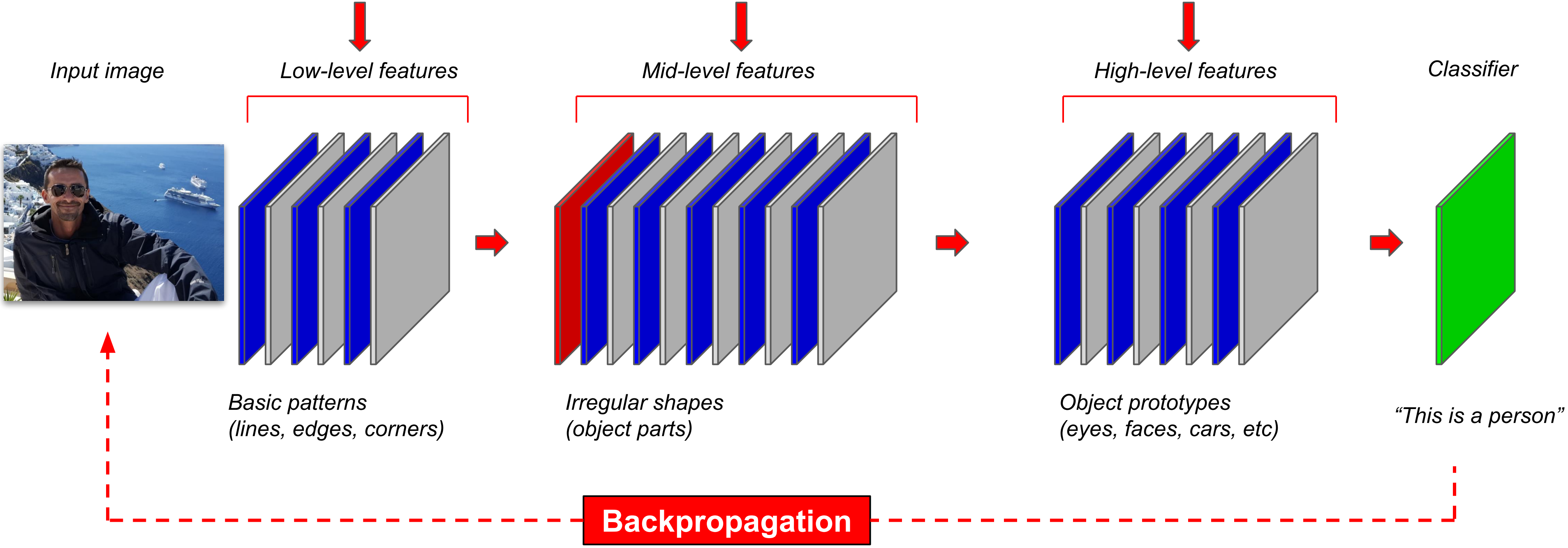
“This is a person”

Backpropagation

Dreams *(and nightmares)*

How it works

It knows *<u>what filters need to be applied</u>* in order to produce each of the feature maps below.
It learned that through backpropagation.



Input image

Low-level features

Mid-level features

High-level features

Classifier

Basic patterns
(lines, edges, corners)

Irregular shapes
(object parts)

Object prototypes
(eyes, faces, cars, etc)

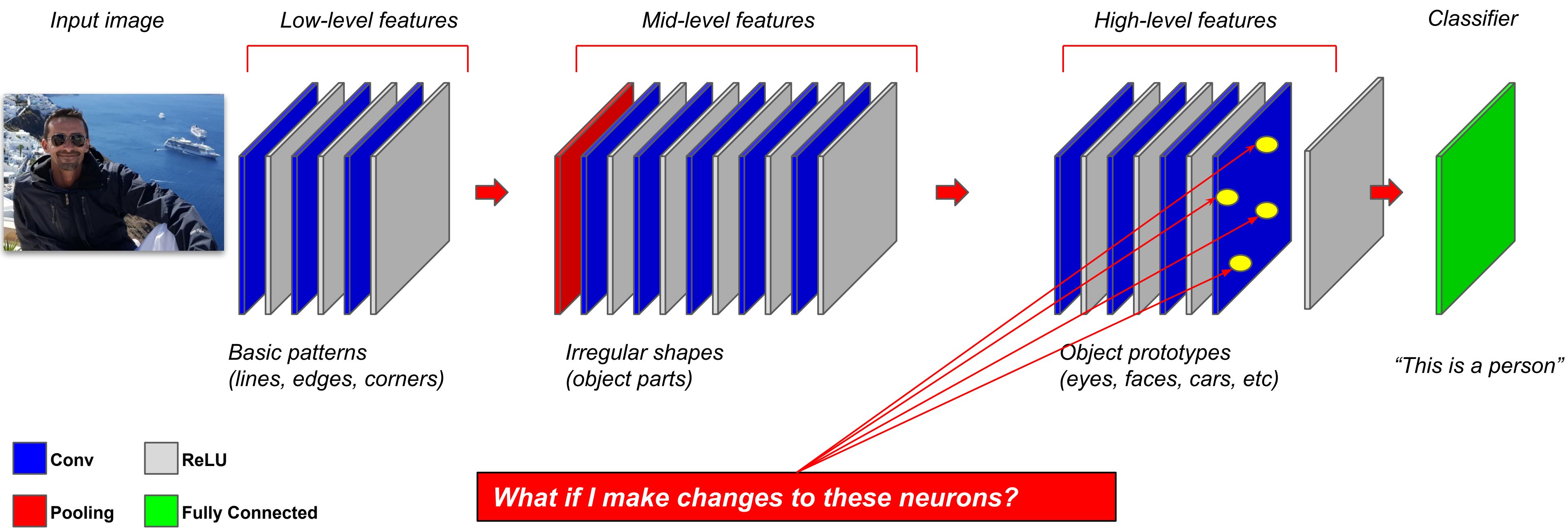
“This is a person”

Backpropagation

Dreams *(and nightmares)*

So what happens if I boost the values of the high-level features?

For example → "Do it more cat-like", "smaller eyes", "bigger ears"



Input image

Low-level features

Basic patterns (lines, edges, corners)

Mid-level features

Irregular shapes (object parts)

High-level features

Object prototypes (eyes, faces, cars, etc)

Classifier

"This is a person"

What if I make changes to these neurons?

🟦 **Conv**

⬜ **ReLU**

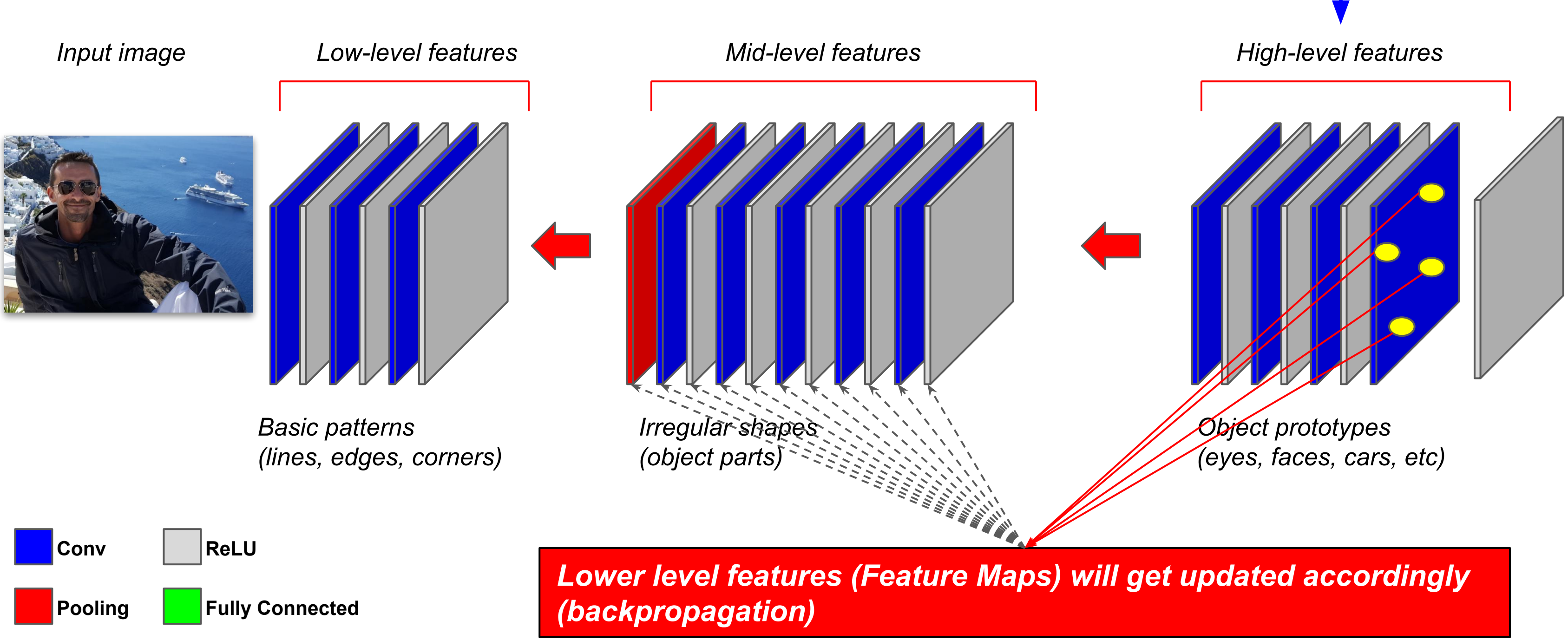
🟥 **Pooling**

🟩 **Fully Connected**

Dreams *(and nightmares)*

So what happens if I boost the values of the high-level features?

For example → "Do it more cat-like", "smaller eyes", "bigger ears"

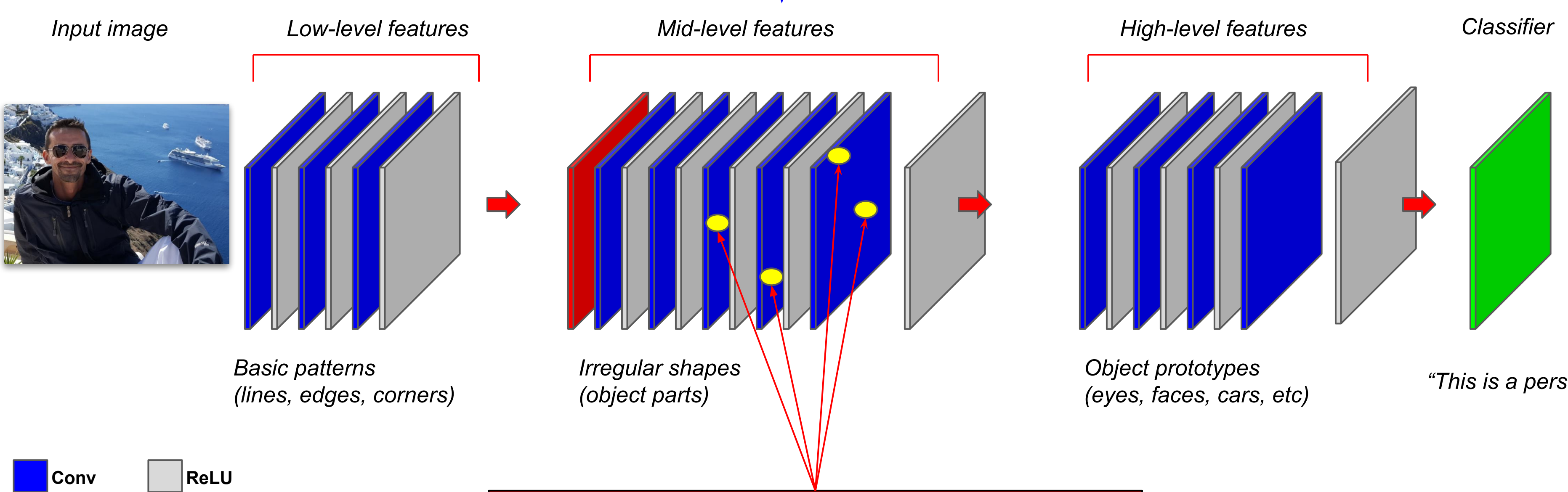


Lower level features (Feature Maps) will get updated accordingly (backpropagation)

Dreams *(and nightmares)*

How about this? What if I boost mid-level features?

"inhibit rectangles", "maximize circular shapes"



Input image

Low-level features

Mid-level features

High-level features

Classifier

Basic patterns
(lines, edges, corners)

Irregular shapes
(object parts)

Object prototypes
(eyes, faces, cars, etc)

"This is a person"

🟦 **Conv** ⬜ **ReLU**

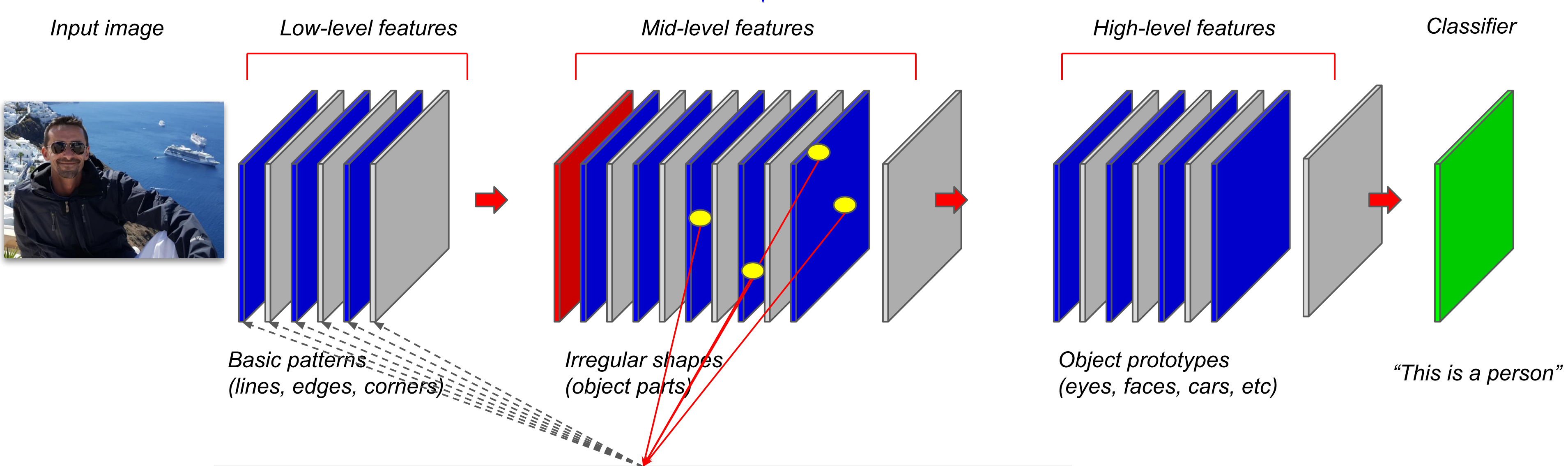
🟥 **Pooling** 🟩 **Fully Connected**

What if I make changes to these neurons?

Dreams *(and nightmares)*

How about this?   What if I boost mid-level features?

“inhibit rectangles”, “maximize circular shapes”



Input image

Low-level features

Mid-level features

High-level features

Classifier

Basic patterns
(lines, edges, corners)

Irregular shapes
(object parts)

Object prototypes
(eyes, faces, cars, etc)

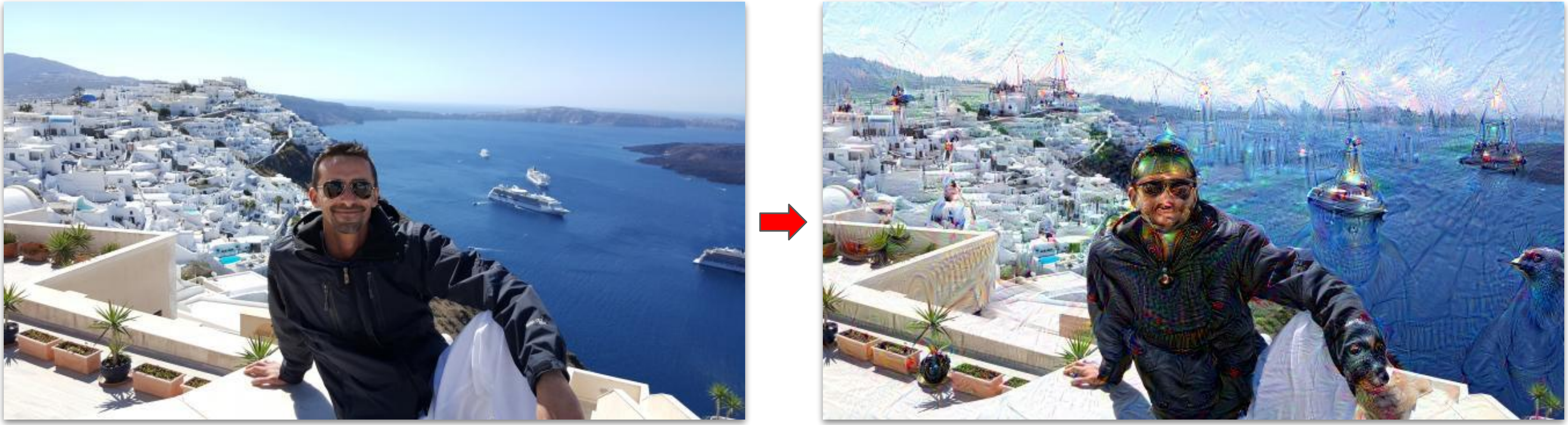
“This is a person”

Lower level features (Feature Maps) will get updated accordingly (backpropagation)

Dreams - *Summary*

Dreaming consists in a boost of the values produced by inner feature maps.

Example → *“Any slightly detected bird face will be made more and more bird-like over time”*



Next Steps

What you get is the one you see	**How to do it**	**What you get**
Lazy mode	Download someone else's code. It takes 10 seconds. Try to learn about it.	**Just click on the *"Download"* button.** Quick and easy. Enjoy your copy/paste. *Don't forget to brag about it !!!*
A better way	Use a mature technology or framework: ● *TensorFlow (my suggestion)* ● *Caffe, Lasagne, Keras, ConvNetJs, etc.* *Google them!*	**Enjoy what others have created for you** ● Click on the *"Download"* button. ● Learn how to use the tool at hand. ● Create your Neural Networks. **It's a great start. You should take the offer!**
The hard way, ***(for crazy people like me)*** *"Eat well, Spartan. Because this night we're gonna dine in **hell**"*	Create your own CNN from scratch, in any language of your preference. ● C++, Java, C#, etc… **Earn your wings, Spartan!**	**This is Sparta!** Learning and fun is 110% guaranteed. You can trust me on this. ***For road warriors ONLY!***

Happy coding!

Thank you !