Deep Neural Networks

Machine Learning and Pattern Recognition

(Largely based on slides from Fei-Fei Li & Justin Johnson & Serena Yeung)

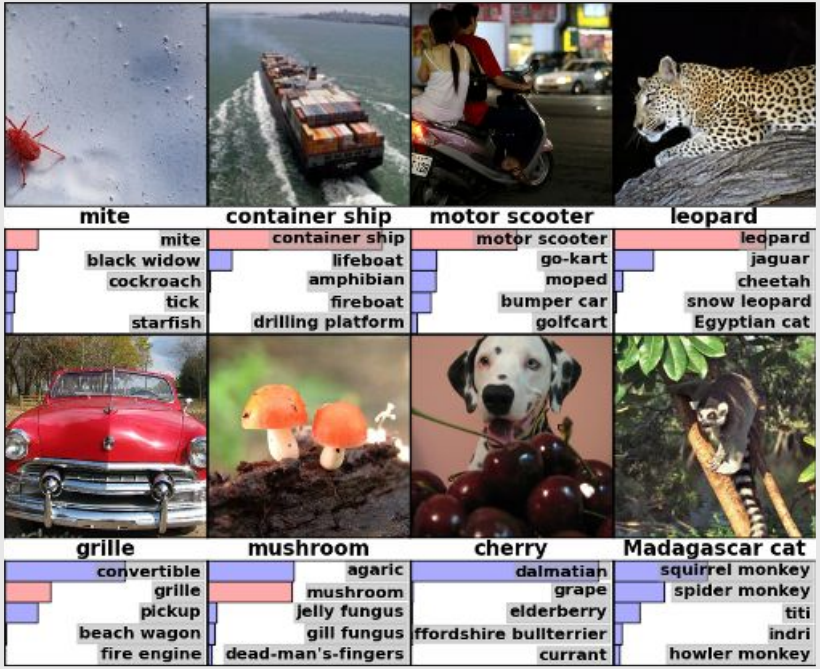
Prof. Sandra Avila

Institute of Computing (IC/Unicamp)

MC886/MO444, October 2, 2018

DNNs are everywhere …

Classification



Credit: Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012

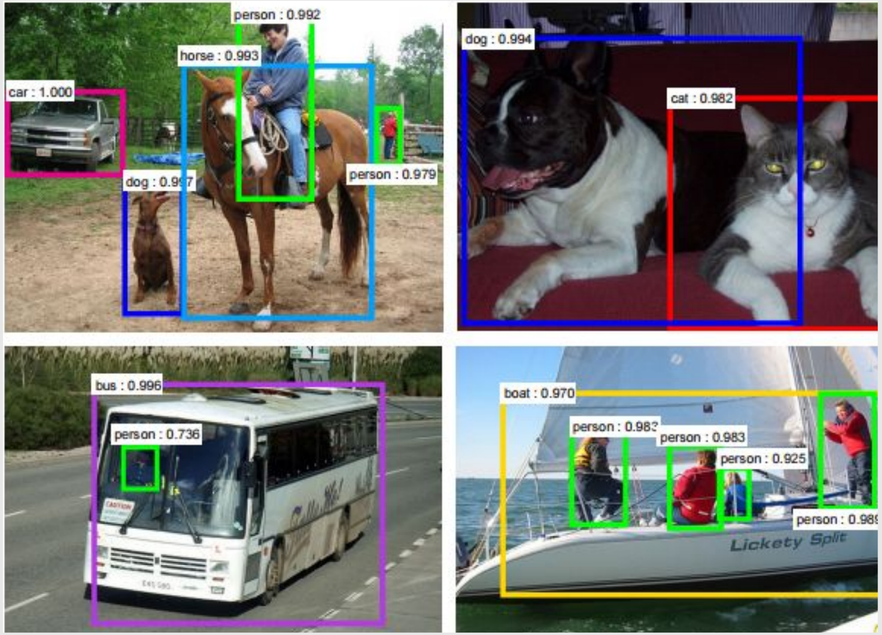
Retrieval

DNNs are everywhere ...

Classification: ElsaGate vs. Safe

DNNs are everywhere ...

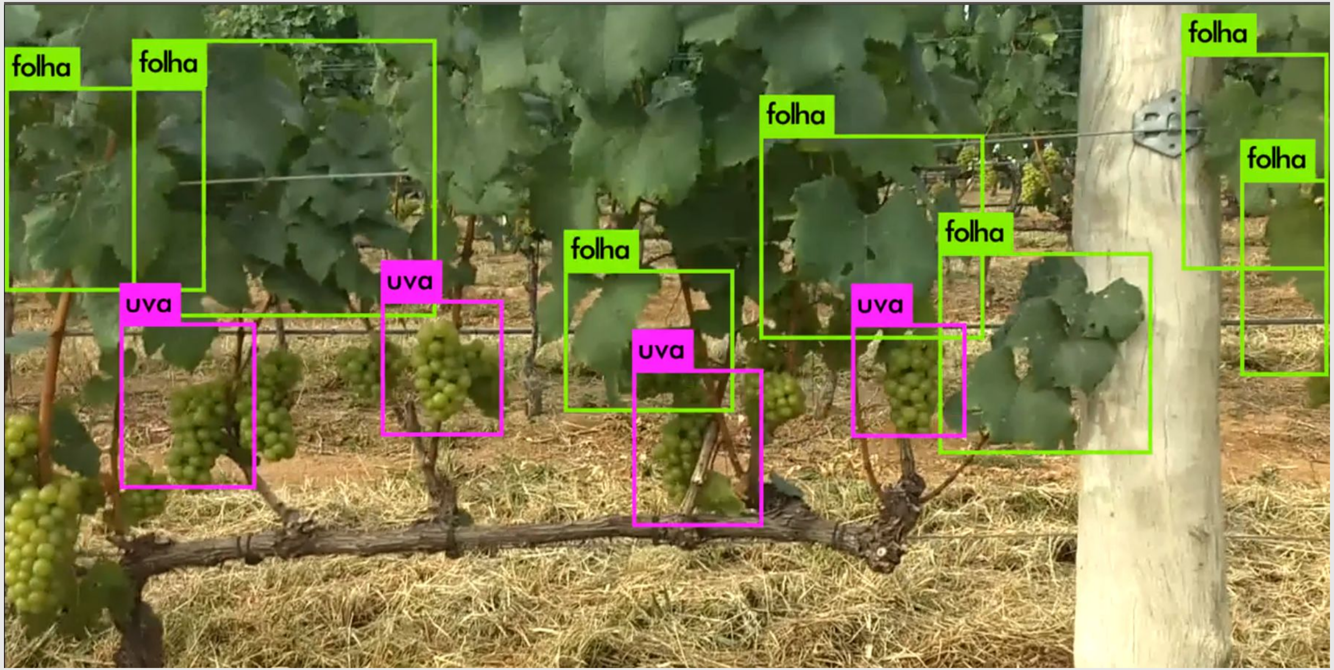
Detection



Segmentation

Credit: Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Clement Farabet, 2012.

DNNs are everywhere ...



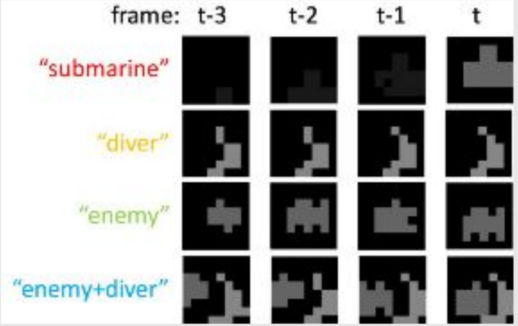
Detection

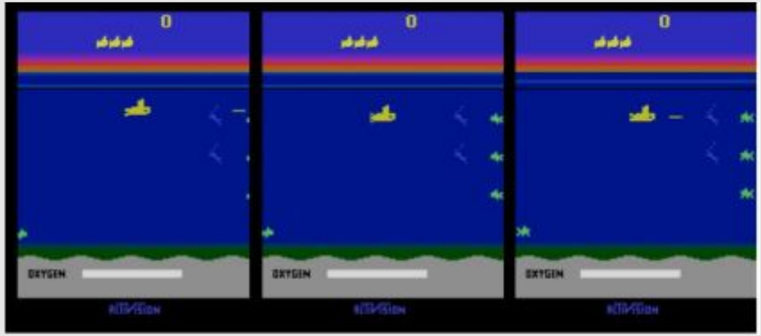
Andreza Santos, Thiago Teixeira Santos, Sandra Avila. 2018.
https://youtu.be/YgZbTca1hl8

DNNs are everywhere ...

Pose Estimation

Playing Games





Credit: Toshev & Szegedy 2014. Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014.

DNNs are everywhere ...

Image Captioning

No errors | Minor errors | Somewhat related

A white teddy bear sitting in the grass

A man in baseball uniform throwing a ball

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

Captions generated by Justin Johnson using Neuraltalk.

DNNs are everywhere ...

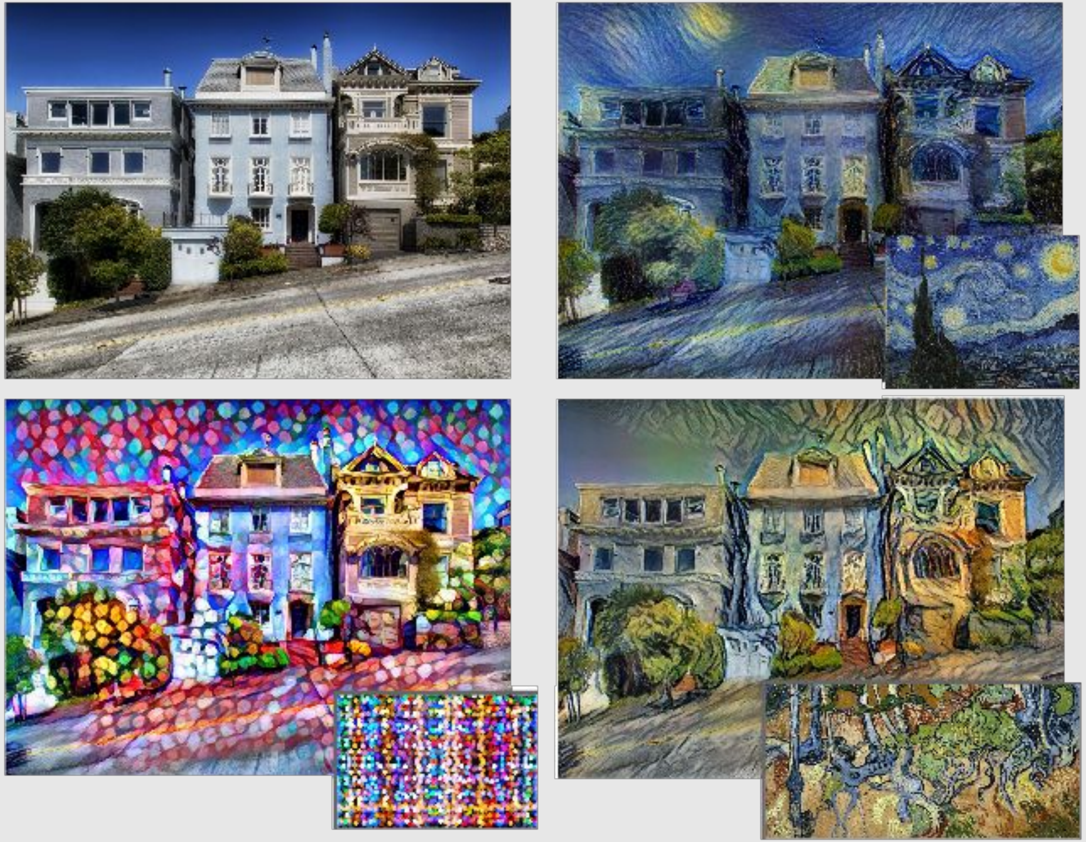


Image Style Transfer

Gatys et al, “Image Style Transfer using Convolutional Neural Networks”, CVPR 2016
Gatys et al, “Controlling Perceptual Factors in Neural Style Transfer”, CVPR 2017

DNNs are everywhere ...

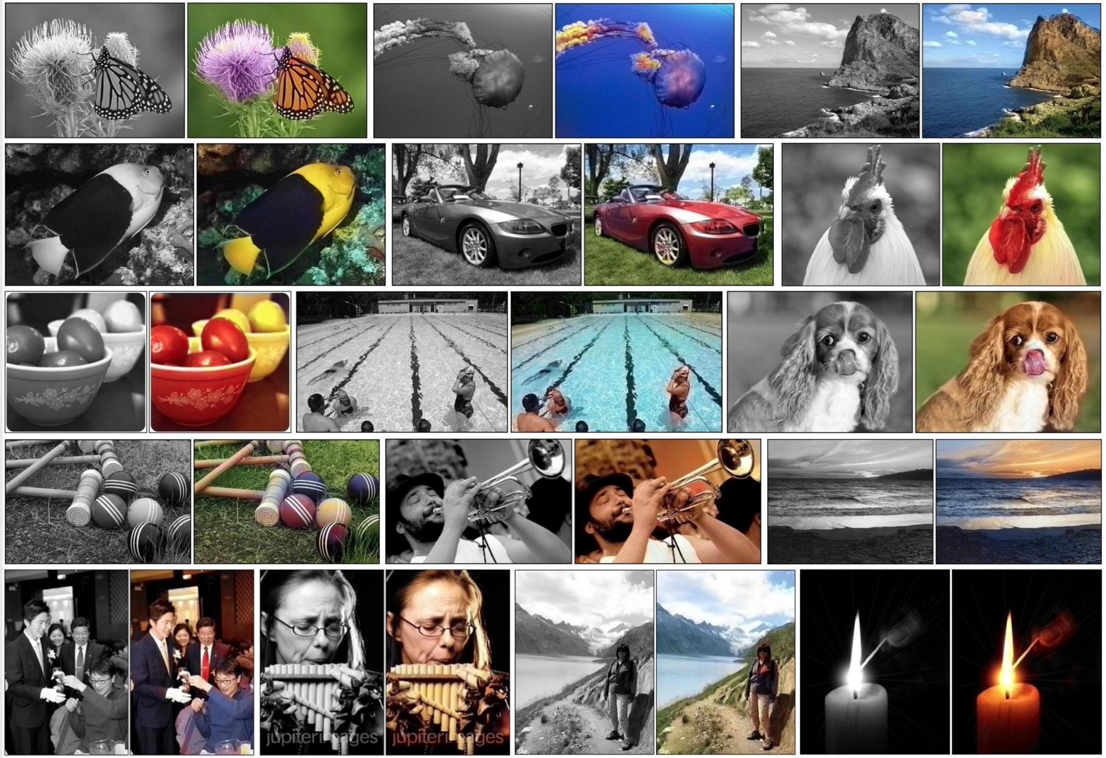
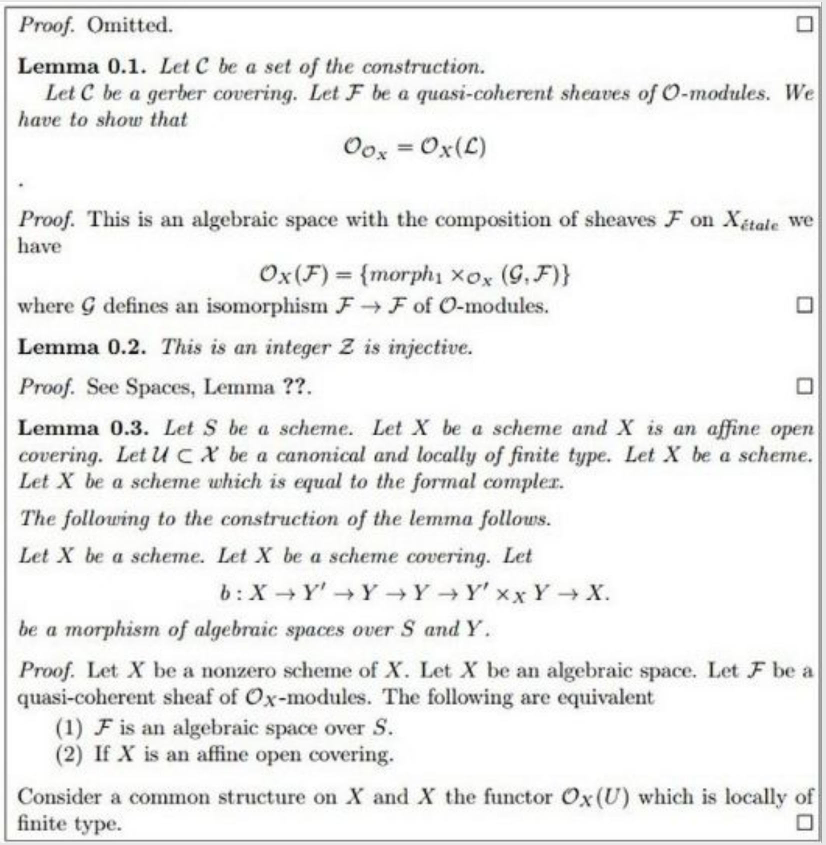


Image Colorization

Zhang et al., “Colorful Image Colorization”, ECCV 2016
https://demos.algorithmia.com/colorize-photos/

DNNs are everywhere ...



Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

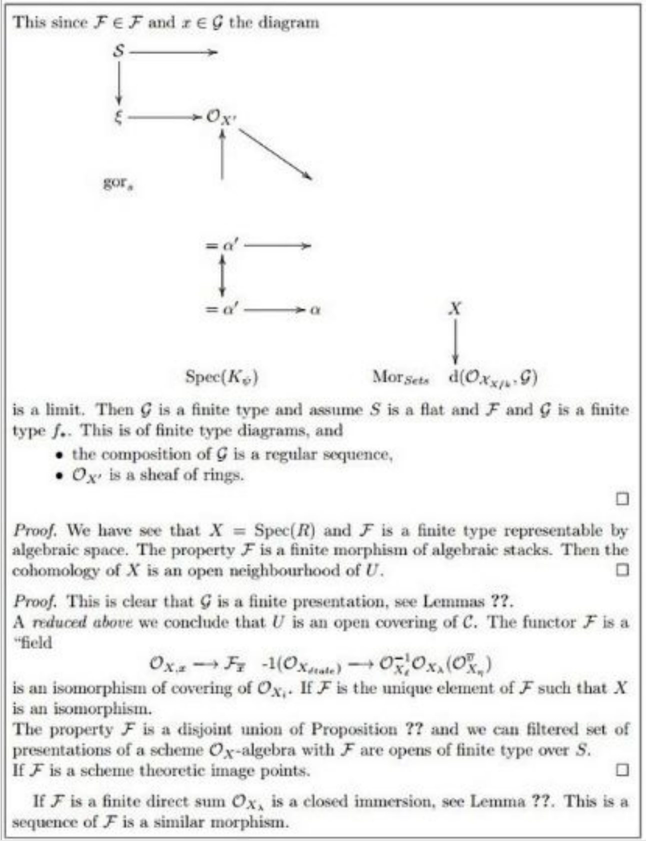
$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □



This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{tikzcd}
\mathcal{S} \arrow[r] \arrow[d] & {} \\
\xi \arrow[r] & \mathcal{O}_{X'} \arrow[dr] \\
\mathrm{gor}_s & {} \arrow[u] & {} \\
{} & = \alpha' \arrow[r] \arrow[d, leftrightarrow] & {} \\
{} & = \alpha' \arrow[r] & \alpha & X \arrow[d] \\
{} & \mathrm{Spec}(K_\psi) & \mathrm{Mor}_{Sets} & \mathrm{d}(\mathcal{O}_{X_{X/k}}, \mathcal{G})
\end{tikzcd}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \;\; -1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Text Generation

DNNs are everywhere ...

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of*

Let $\mathcal{C}$ be a gerber covering.

have to show that

.

Proof. This is an algebraic spa

have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

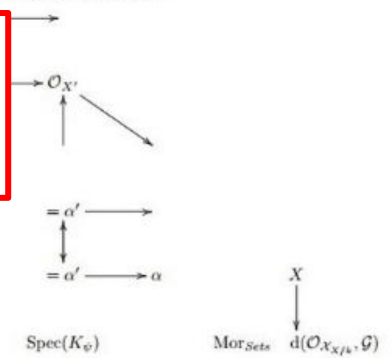
Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

Proof. Omitted.

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad {-1}(\mathcal{O}_{X_{etale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Text Generation

Training: “Maior dúvida da aula” 27/october/2017

https://github.com/llSourcell/recurrent_neural_network

GoogLeNet, Inception Module

Não entendi muito bem sobre as inception layers na GoogLeNet. Entendi a ideia de fazer a mesma coisa de um filtro grande com vários filtros menores. Com vários filtros menores temos menos parâmetros que um filtro grande?

Quando fazemos inception e concatenados os resultados, podemos comparar isso à criação de vetor de características? Porque estamos retirando tipos diferentes de informações de uma mesma camada de input e juntando elas pra formar um output.

Acho que não consegui entender muito bem o inception module da arquitetura GoogLeNet. Para que ele serve exatamente? Obrigada.

no modelo de inception v4, usa a paralelizacao para obter menos parametros, entao esso quer dizer que enquanto menos parametros e mais profundo da melhores resultados?

Não entendi exatamente que fator possibilitou a remoção das camadas fully connected na GoogleLeNet. Pelo que eu entendi, as redes mais modernas voltaram com a camada fully connected. Então quando usá-la ou não usá-la?

Números de parâmetros

Em relação a arquiterua proposta na rede GoogLeNet, não ficou muito claro para mim as camadas internas, principalmente na parte em que aplicar vários filtros menores, equilave a aplicar um filtro maior (embora o resultado não seja o mesmo).

Não ficou claro para mim qual a vantagem de se utilizar, por exemplo, 3 pequenos filtros 3x3 ao invés de um 7x7. Na aula você comentou que é para evitar diminuir drasticamente a imagem, mas qual a desvantagem disso?

Eu nao entendi aquelas contas dos filtros que reduziam o numero de parametros

ResNet Filtro 1x1

Achei um pouco confuso as dimensões do filtro 1x1. Achei confuso a parte da convolução de tal filtro.

Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
----
 'ōqIE:ō:3(é
O Q.L"cÉhíL'uàfMO)êoâz.àãâéláç-)D(iéêdàF(lLFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u
xp5LQ"r24F7é1efL"CabvêúhyLdã 7àã2àObmxv?qnAodí'P)mTg4(u4F7ú13ómrQnmeFNbãoúvâ3i?sxsuRãjáécó.-
záy-
----
```

Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
----
 'ōqIE:ō:3(é
O Q.L"cÉhíL'uàfMO)êoâz.àãâéláç-)D(iéêdàF(lLFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u
xp5LQ"r24F7é1efL"Cabvêúhy Ldã 7àã2àObmxv?qnAodí'P)mTg4(u4F7ú13ómrQnmeFNbãoúvâ3i?sxsuRãjáécó.-
záy
---
```

```
iter 46000, loss: 23.238596
----
 és GoogLeNet. E a rede aprede?

O Daras dúvrvilg. ( ende no pré-tro "rar outlara destidas? Com uttres dessar algo us filtros
parte novados aplicar au mula.

e narepteno Retênne camada entros lemos e m
```

Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
----
 'ōqIE:ō:3(é
O Q.L"cÉhíL'uàfMO)êoâz.àãâéláç-)D(iéêdàF(lLFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u
xp5LQ"r24F7élefL"CabvêúhyLdã 7àã2àObmxv?qnAodí'P)mTg4(u4F7úl3ómrQnmeFNbãoúvâ3i?sxsuRãjáécó.-
záy
---
```

```
iter 46000, loss: 23.238596
----
 és GoogLeNet. E a rede aprede?

O Daras dúvrvilg. ( ende no pré-tro "rar outlara destidas? Com uttres dessar algo us filtros
parte novados aplicar au mula.

e nar
```

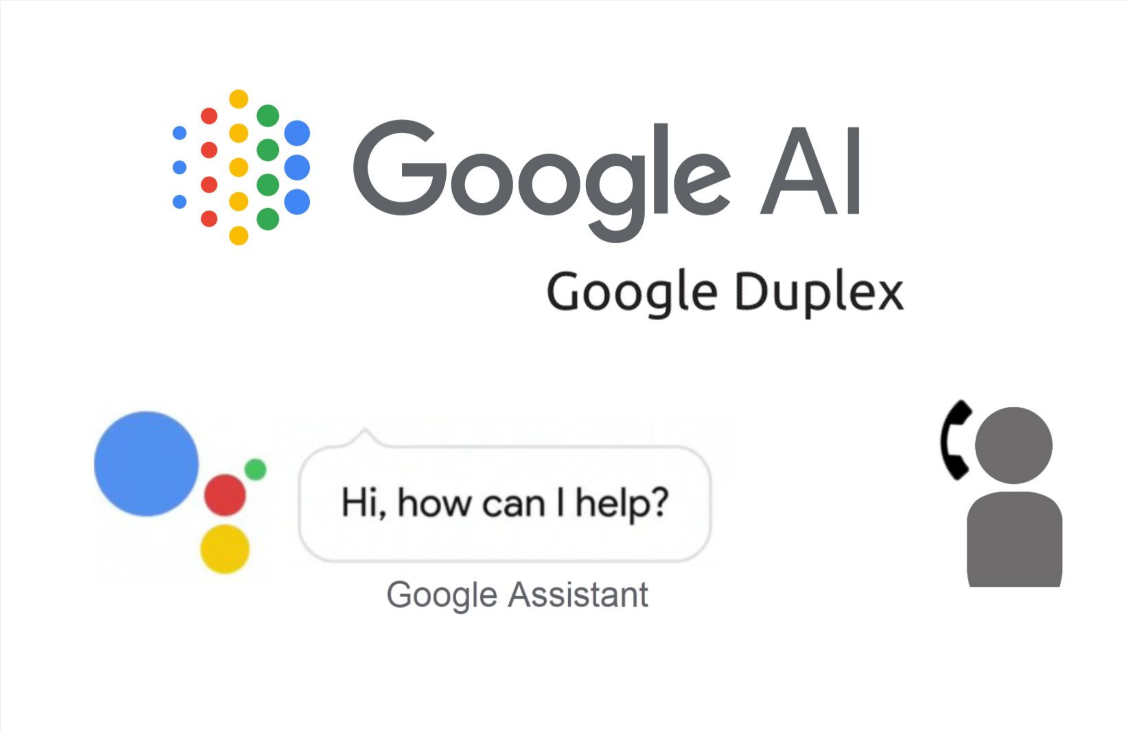
```
iter 204000, loss: 10.733449
----
 to, ina utir alpal asvelum motrio tarada mexexenterna mai reviso de enter meiss grandas

##### ResNet Filtro 1x1? Alheing?

Não entendi exatamente que fia, confenhalo deset desecta..

##### Como as
```

DNNs are everywhere ...



https://www.youtube.com/watch?v=WPzu6W2rWNs

AI Assistant Calls:
Google Duplex

DNNs are everywhere ...

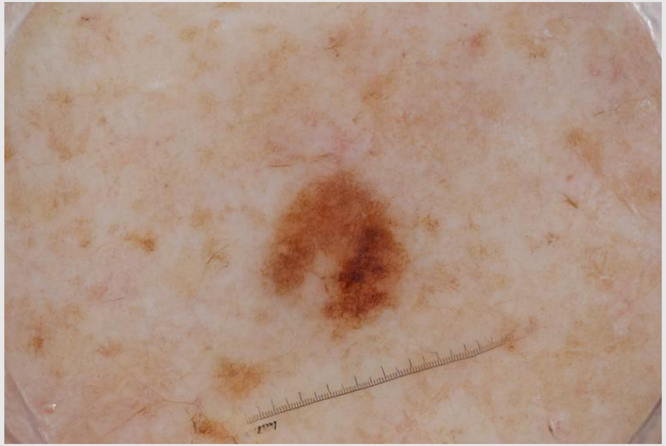
Synthesizing Audio

Suwajanakorn et al., “Synthesizing Obama
Learning Lip Sync from Audio”, SIGGRAPH, 2017

https://youtu.be/mKxgAnuvaZk
https://youtu.be/cQ54GDm1eL0

DNNs are everywhere ...

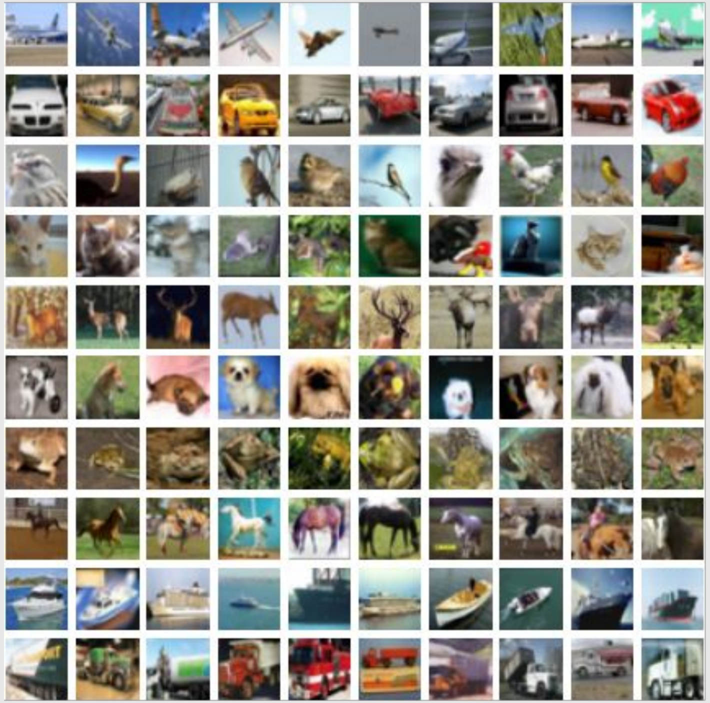
Synthesizing Skin Lesion



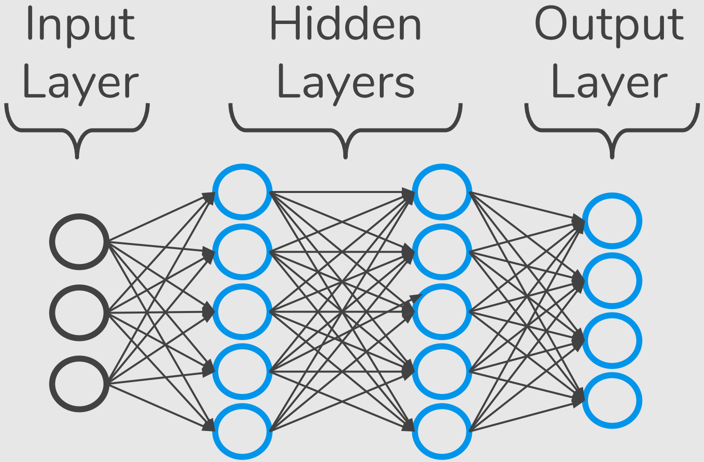
Alceu Bissoto, Fábio Perez, Eduardo Valle, Sandra Avila. “Skin lesion synthesis with generative adversarial networks”, ISIC Skin Image Analysis Workshop at MICCAI, 2018.

Convolutional Neural Networks (CNNs)

Fully Connected Layer

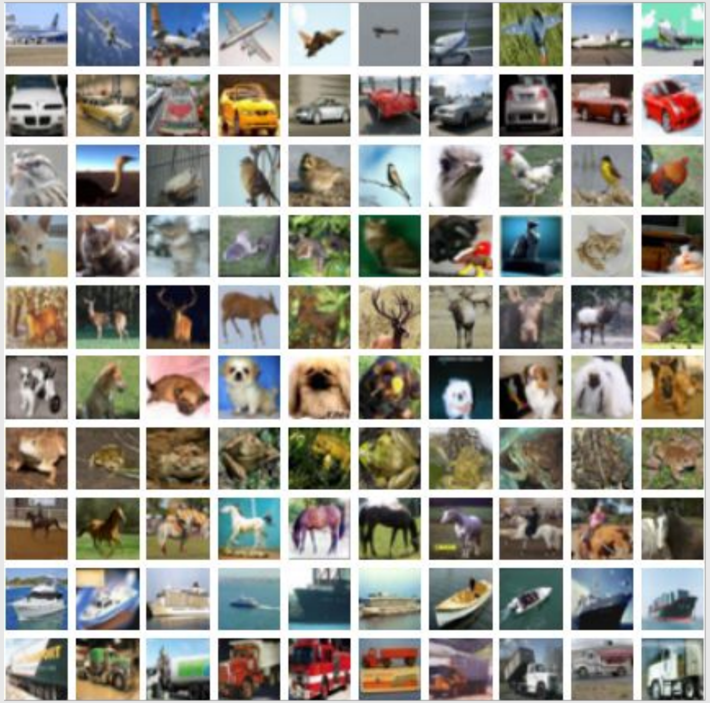


CIFAR-10

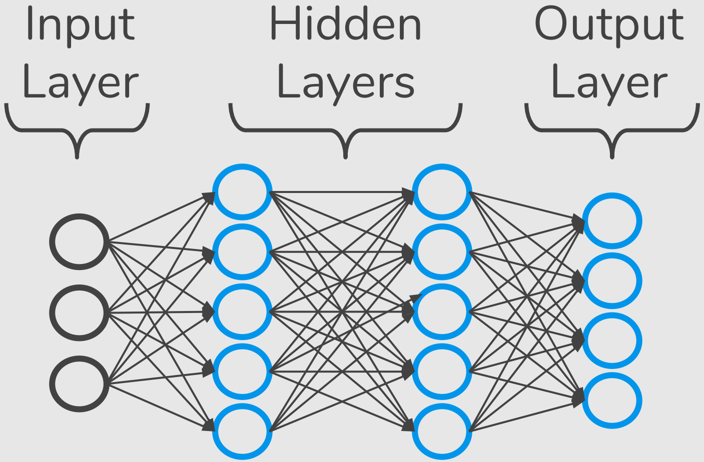


$32 \times 32 \times 3$ image $\Rightarrow$ stretch to 3072×1

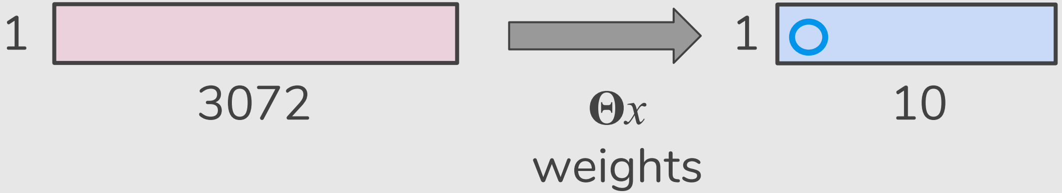
Fully Connected Layer



CIFAR-10



$32 \times 32 \times 3$ image $\Rightarrow$ stretch to 3072×1



What is a Convolution?

What is a Convolution?

0	1	0
1	-4	1
0	1	0

What is a Convolution?

Edge
Detection

0	1	0
1	-4	1
0	1	0

What is a Convolution?

Emboss

-2	-1	0
-1	1	1
0	1	2

→

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

$\rightarrow$

4		

$$1*1 + 1*0 + 1*1 +$$
$$0*0 + 1*1 + 1*0 +$$
$$0*1 + 0*0 + 1*1 = 4$$

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

4	3	

$$1*1 + 1*0 + 0*1 + 1*0 + 1*1 + 1*0 + 0*1 + 1*0 + 1*1 = 3$$

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

4	3	4

$$1*1 + 0*0 + 0*1 +$$
$$1*0 + 1*1 + 0*0 +$$
$$1*1 + 1*0 + 1*1 = 4$$

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

4	3	4
2		

$$0*1 + 1*0 + 1*1 +$$
$$0*0 + 0*1 + 1*0 +$$
$$0*1 + 0*0 + 1*1 = 2$$

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

4	3	4
2	4	

$$1*1 + 1*0 + 1*1 +$$
$$0*0 + 1*1 + 1*0 +$$
$$0*1 + 1*0 + 1*1 = 4$$

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

4	3	4
2	4	3

$$1*1 + 1*0 + 0*1 +$$
$$1*0 + 1*1 + 1*0 +$$
$$1*1 + 1*0 + 0*1 = 3$$

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

3 × 3 filter

4	3	4
2	4	3
2		

0*1 + 0*0 + 1*1 +
0*0 + 0*1 + 1*0 +
0*1 + 1*0 + 1*1 = 2

What is a Convolution?

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

5 × 5 matrix

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

3 × 3 filter

$$\rightarrow \begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & \end{bmatrix}$$

0*1 + 1*0 + 1*1 +
0*0 + 1*1 + 1*0 +
1*1 + 1*0 + 0*1 = 3

What is a Convolution?

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 × 5 matrix

1	0	1
0	1	0
1	0	1

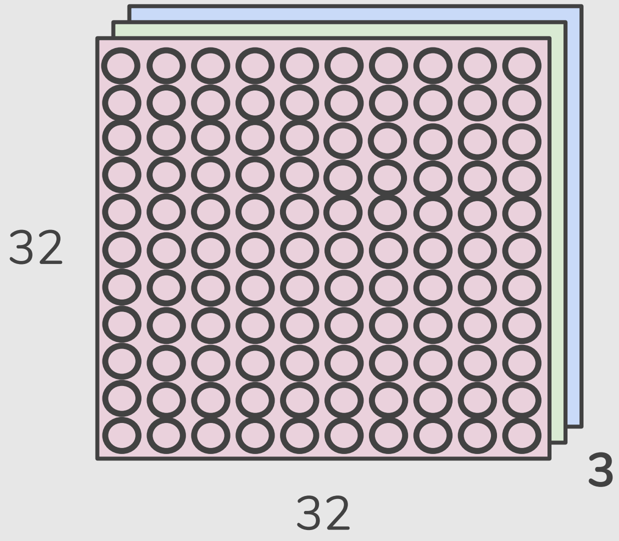
3 × 3 filter

4	3	4
2	4	3
2	3	4

$$1*1 + 1*0 + 1*1 + 1*0 + 1*1 + 0*0 + 1*1 + 0*0 + 0*1 = 4$$

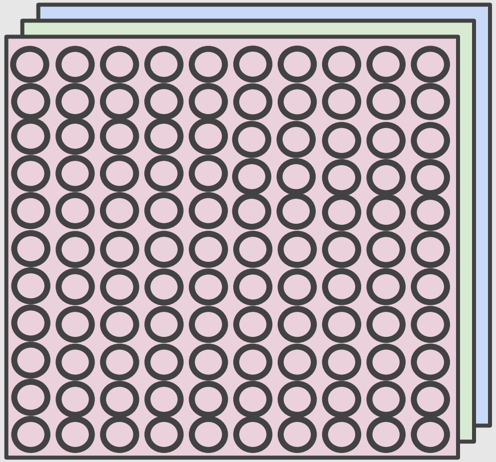
Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



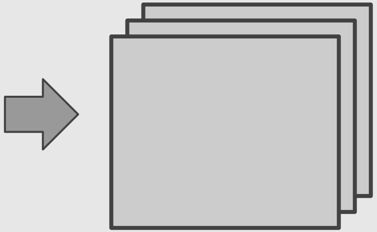
32

32

3

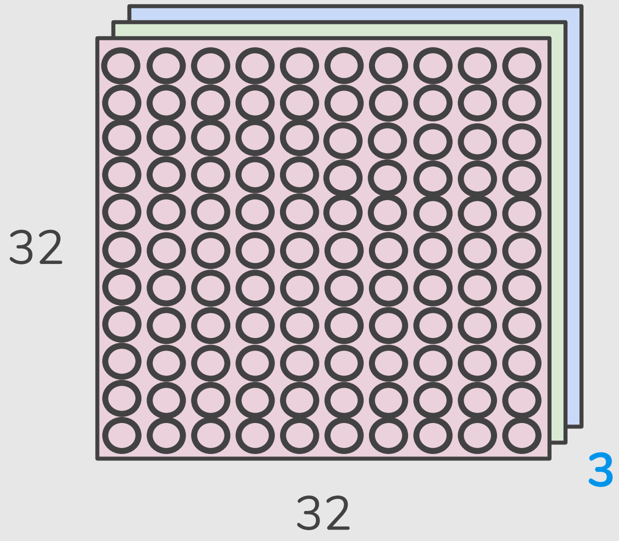
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

$5 \times 5 \times 3$ filter



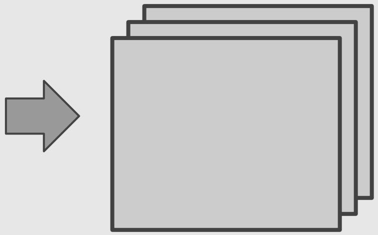
Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

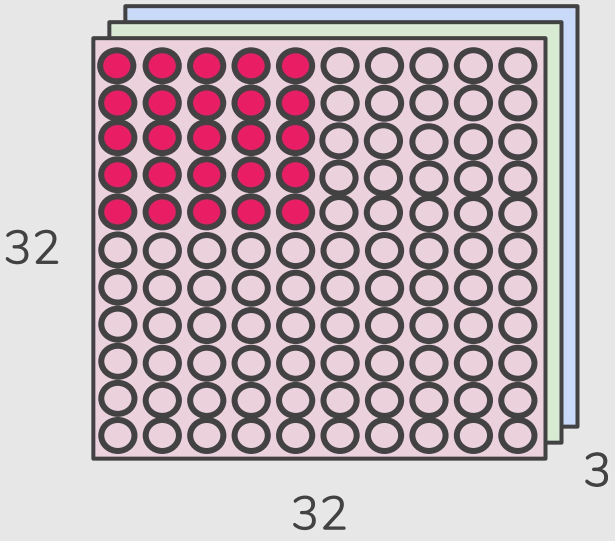
$5 \times 5 \times 3$ filter



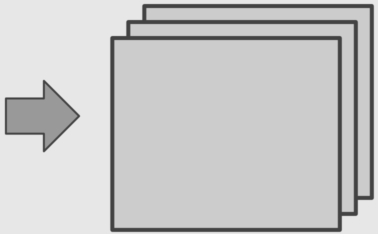
Filters always extend the full depth of the input volume

Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure

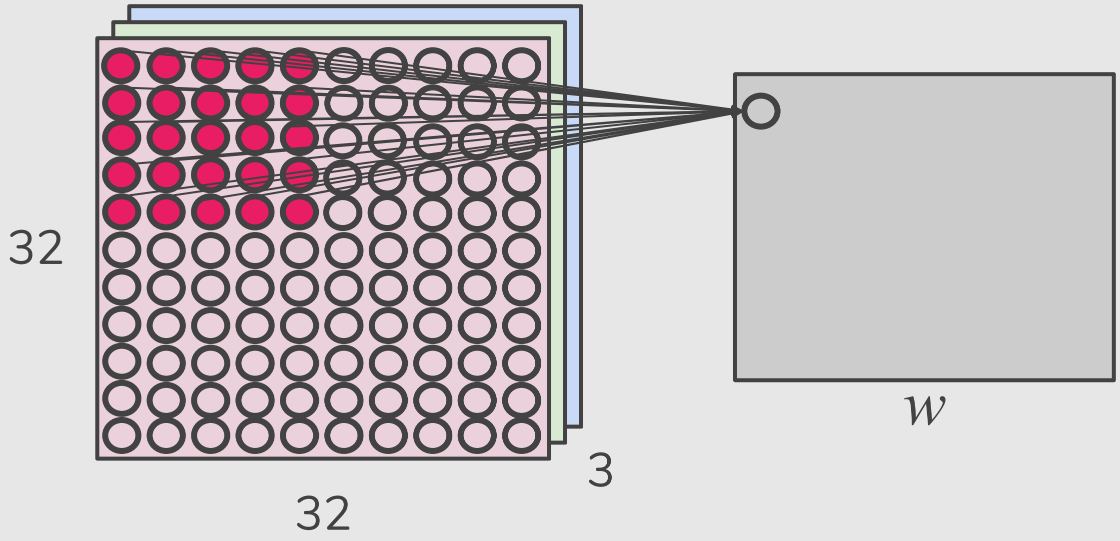


$5 \times 5 \times 3$ filter



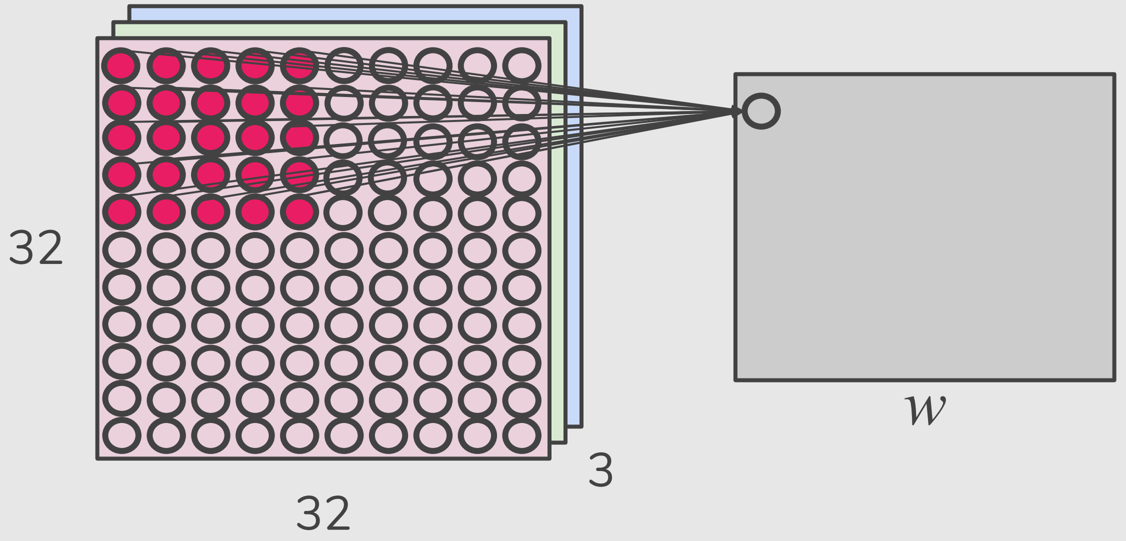
Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



1 number:

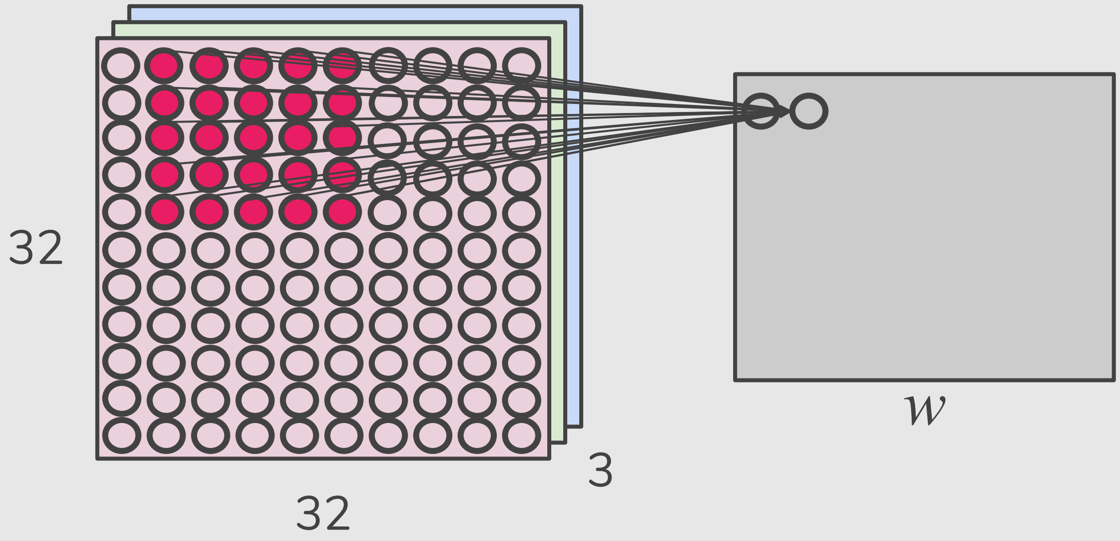
5*5*3 = 75-dimensional

dot product + bias)

$w^{\mathrm{T}}x + b$

Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



1 number:

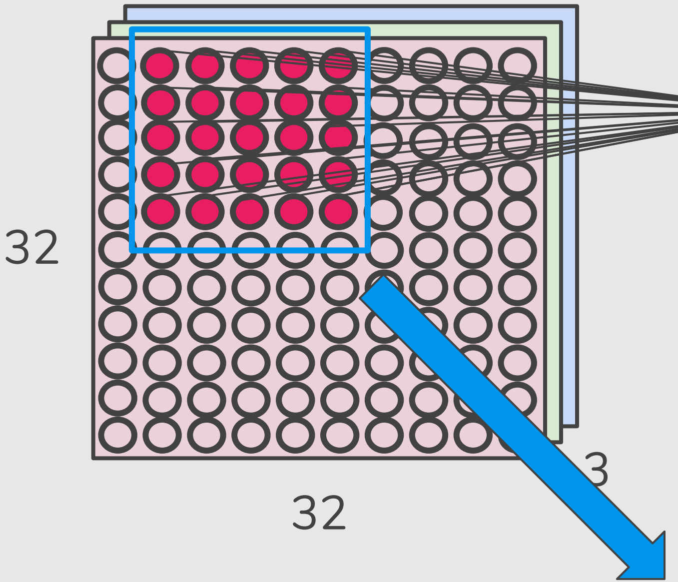
5*5*3 = 75-dimensional

dot product + bias)

$w^{\mathrm{T}}x + b$

Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



1 number:

5*5*3 = 75-dimensional

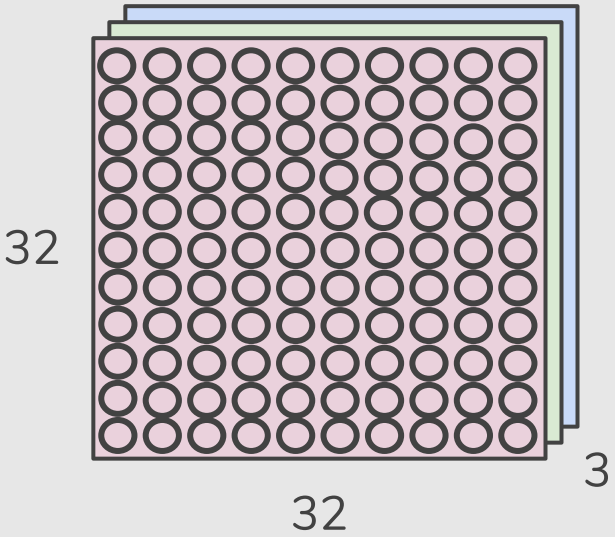
dot product + bias)

$w^{\mathrm{T}}x + b$

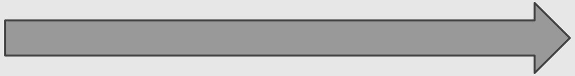
That region in the input image is called the *local receptive field* for the hidden neuron.

Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure

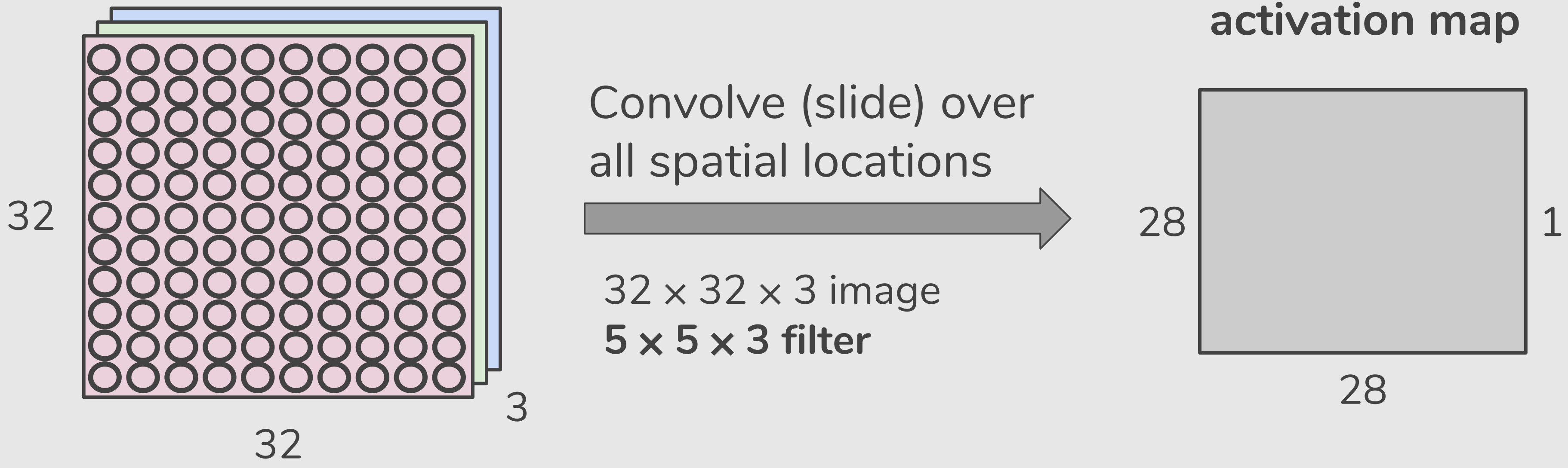


Convolve (slide) over
all spatial locations



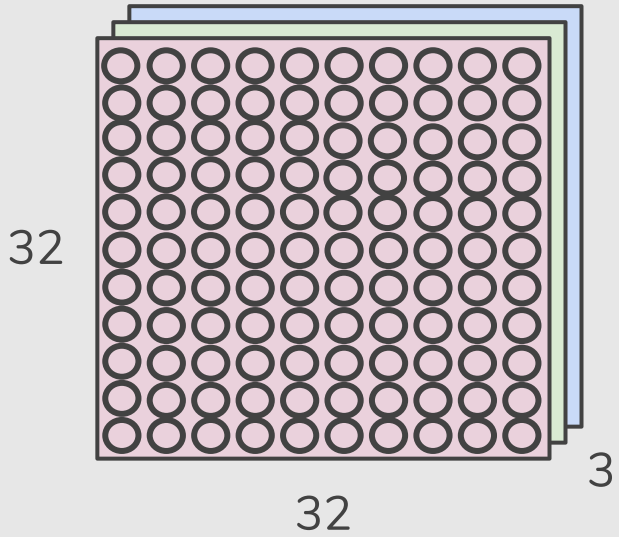
Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure

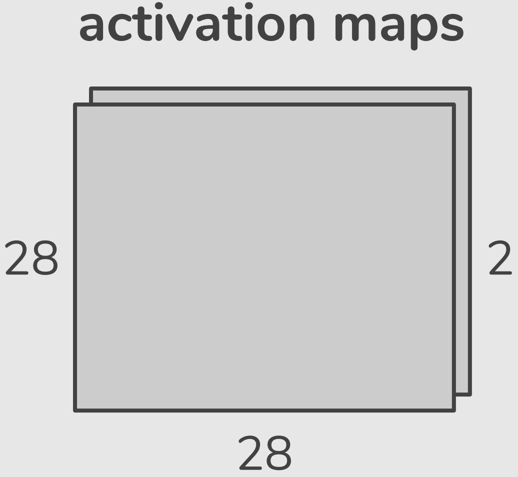


Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



Convolve (slide) over all spatial locations

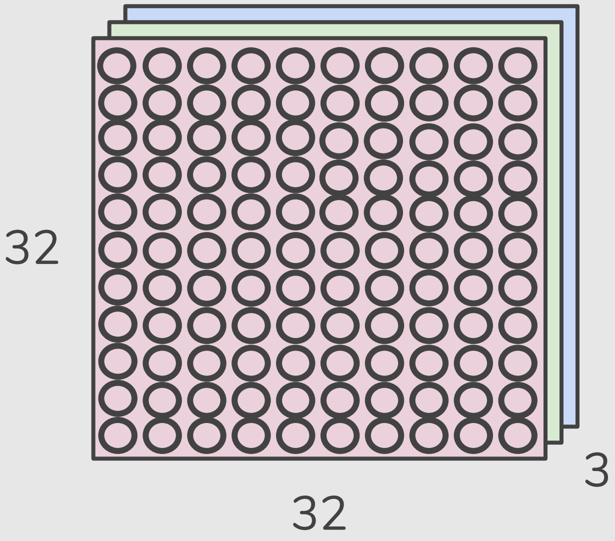


$32 \times 32 \times 3$ image
$5 \times 5 \times 3$ filter

(considering a second filter)

Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure



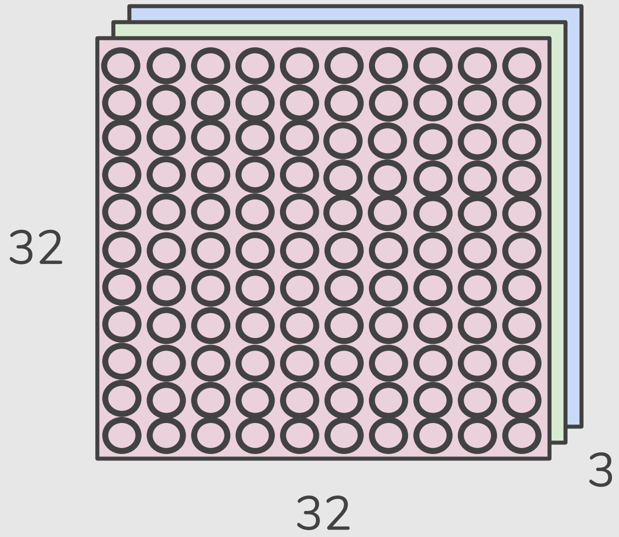
Convolve (slide) over all spatial locations

$32 \times 32 \times 3$ image
$5 \times 5 \times 3$ filter

If we had 6 $5 \times 5 \times 3$ filters ...

Convolution Layer

$32 \times 32 \times 3$ image $\Rightarrow$ preserve spatial structure

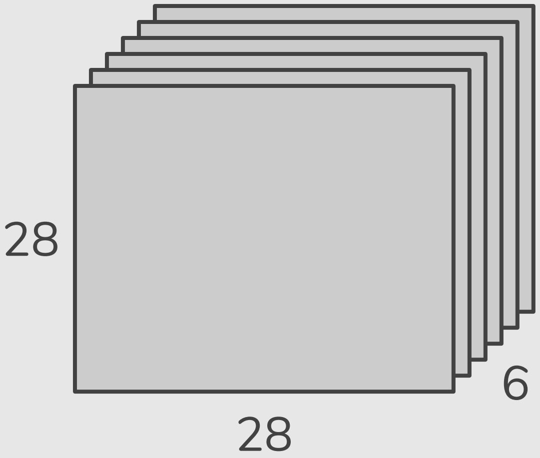


Convolve (slide) over all spatial locations

$32 \times 32 \times 3$ image

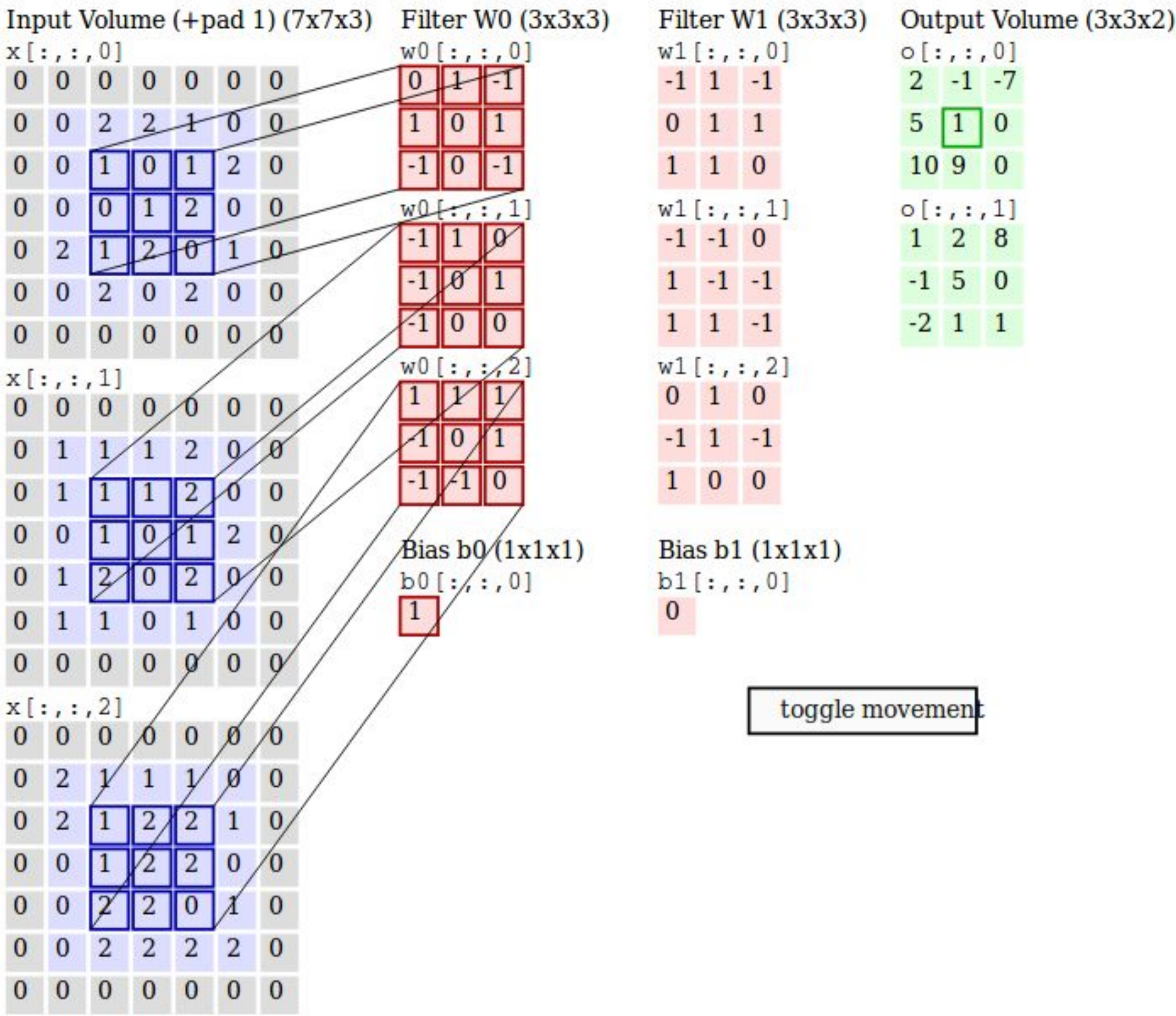
$5 \times 5 \times 3$ filter

6 activation maps



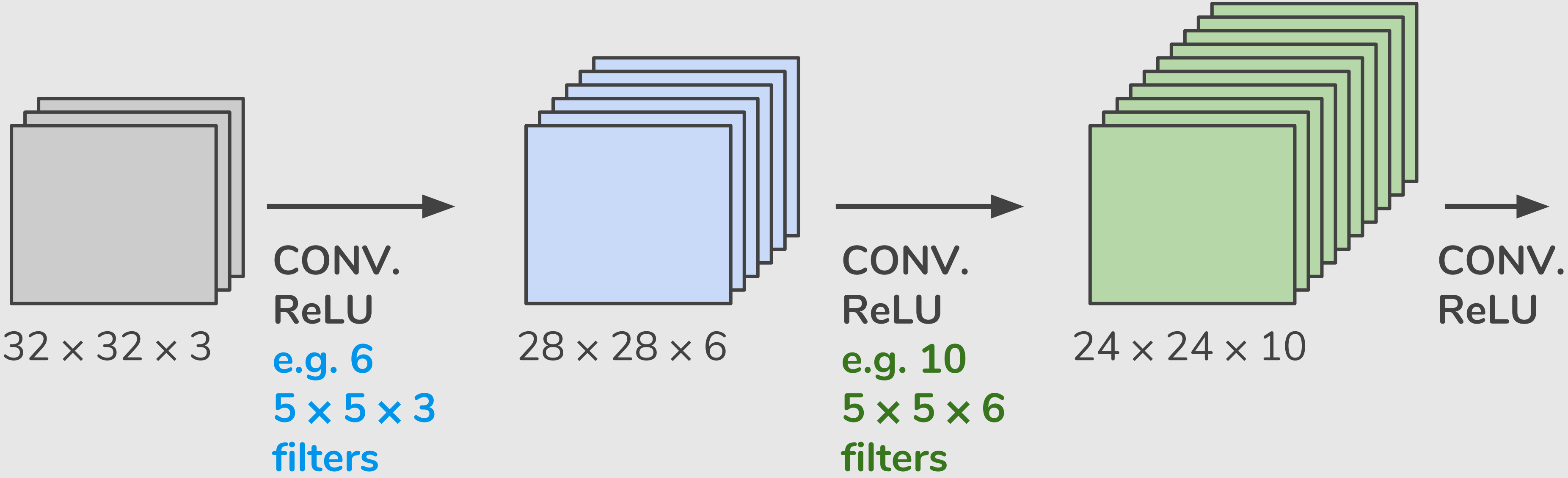
If we had 6 $5 \times 5 \times 3$ filters ...

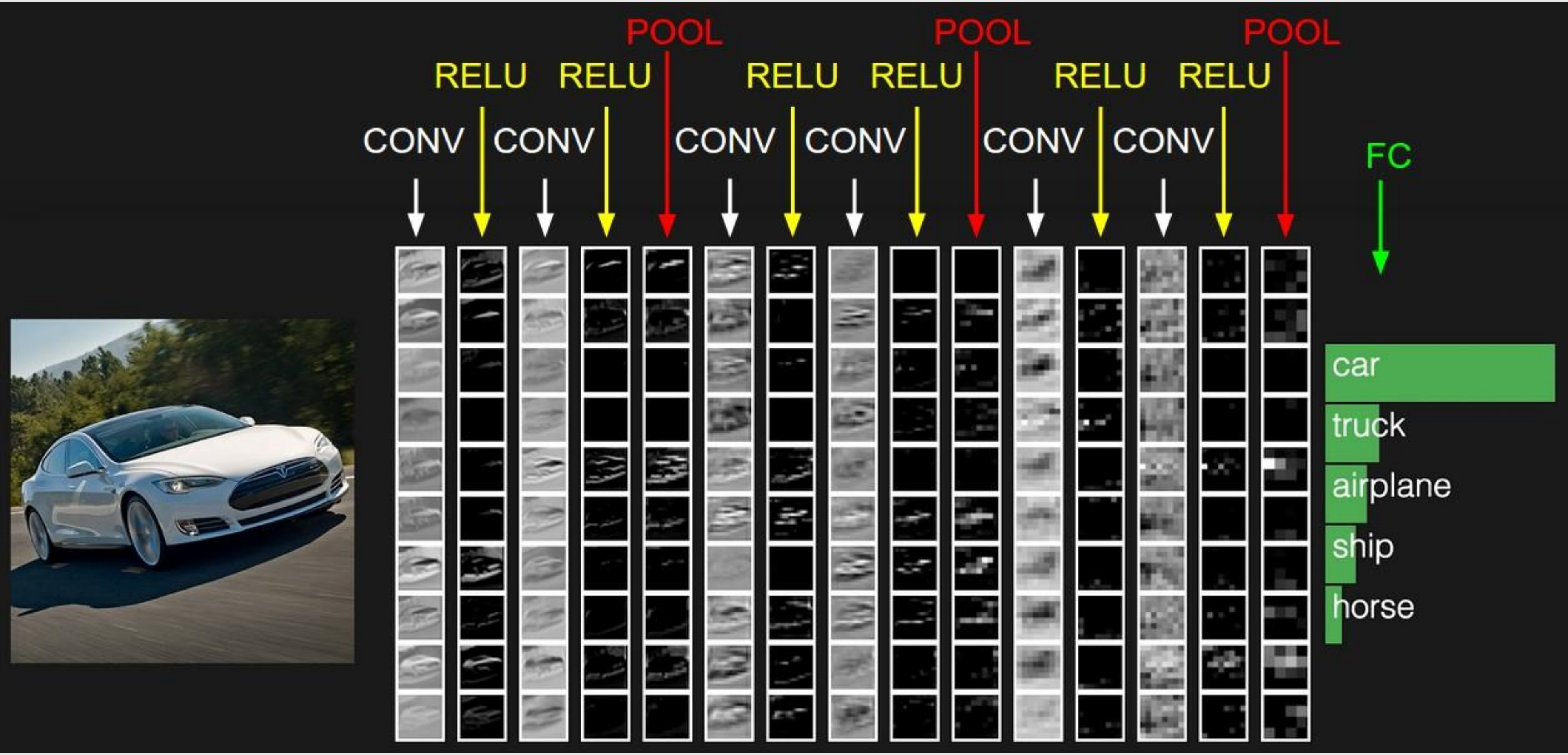
http://cs231n.github.io/convolutional-networks



Convolutional Networks

Sequence of Convolutional Layers, interspersed with activation functions.

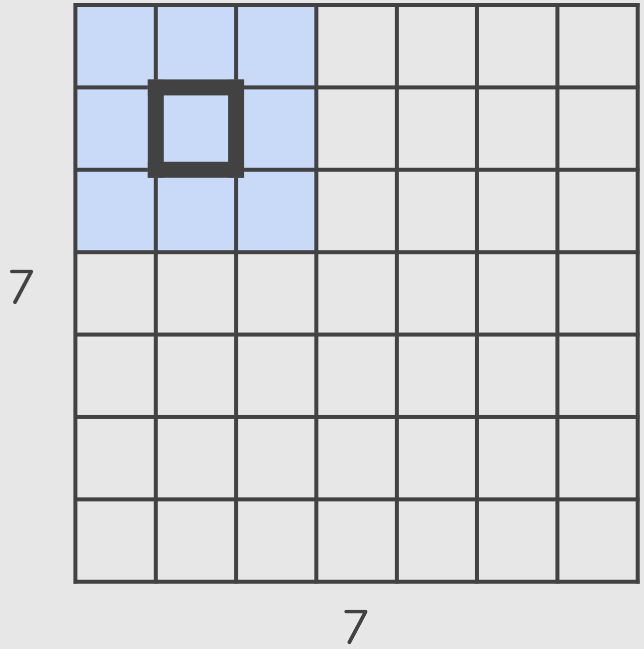




Credit: http://cs231n.github.io/convolutional-networks/

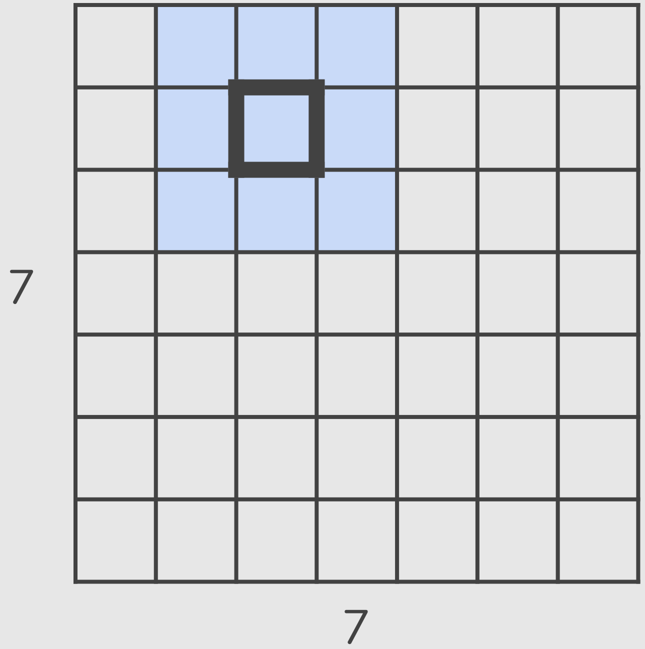
A Closer Look at Spatial Dimensions

A Closer Look at Spatial Dimensions



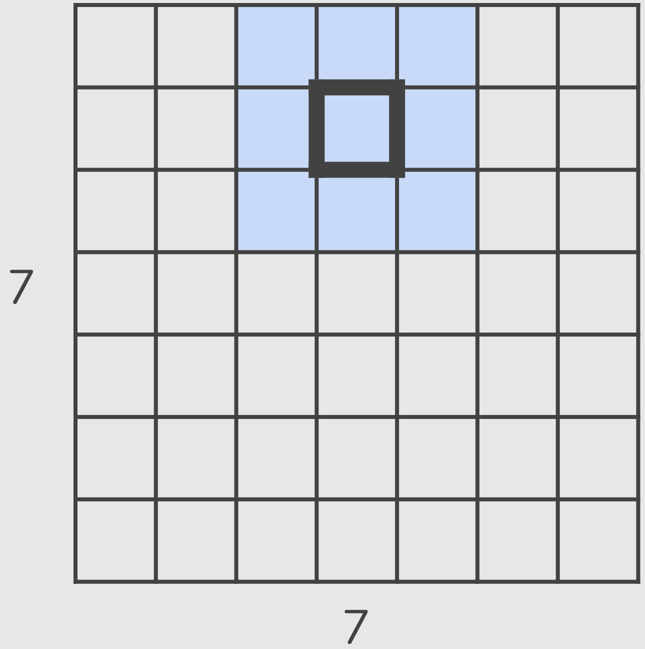
7×7 input (spatially)
assume 3×3 filter

A Closer Look at Spatial Dimensions



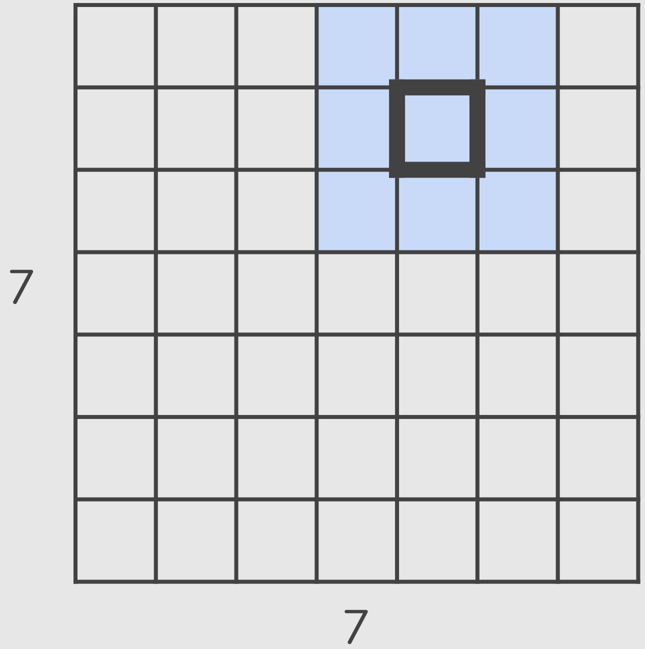
7×7 input (spatially)
assume 3×3 filter

A Closer Look at Spatial Dimensions



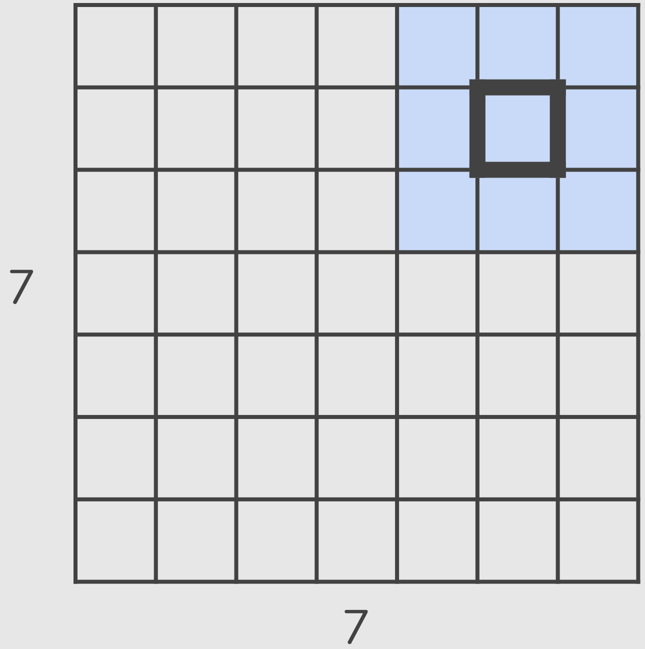
7×7 input (spatially)
assume 3×3 filter

A Closer Look at Spatial Dimensions



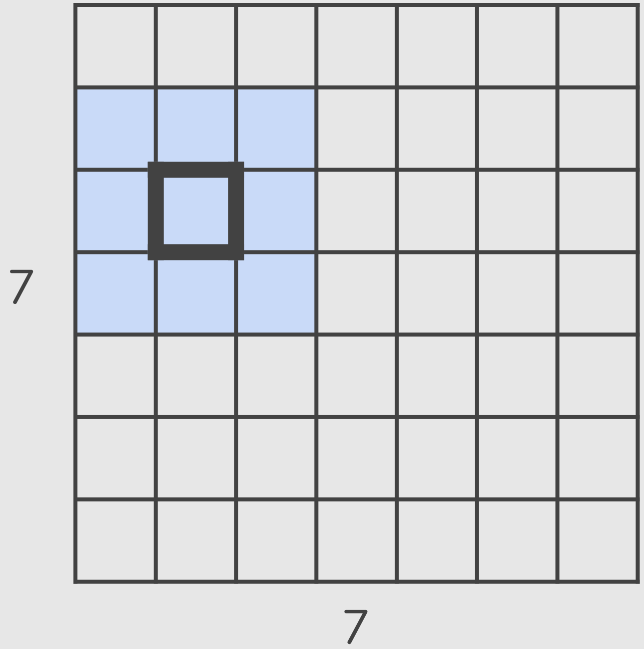
7 × 7 input (spatially)
assume 3 × 3 filter

A Closer Look at Spatial Dimensions



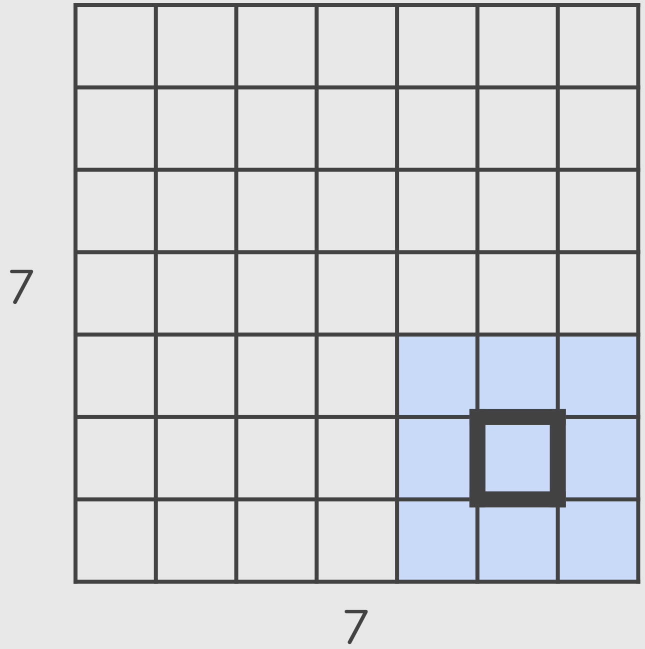
7×7 input (spatially)
assume 3×3 filter

A Closer Look at Spatial Dimensions



7×7 input (spatially)
assume 3×3 filter

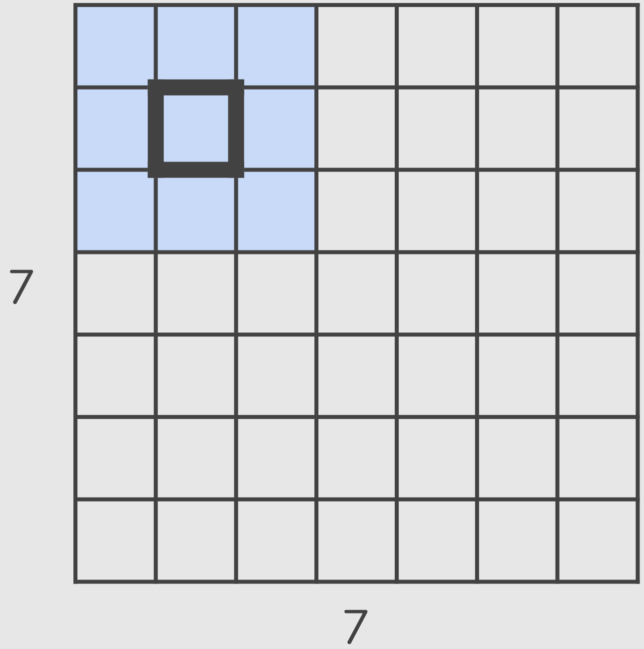
A Closer Look at Spatial Dimensions



7×7 input (spatially)
assume 3×3 filter

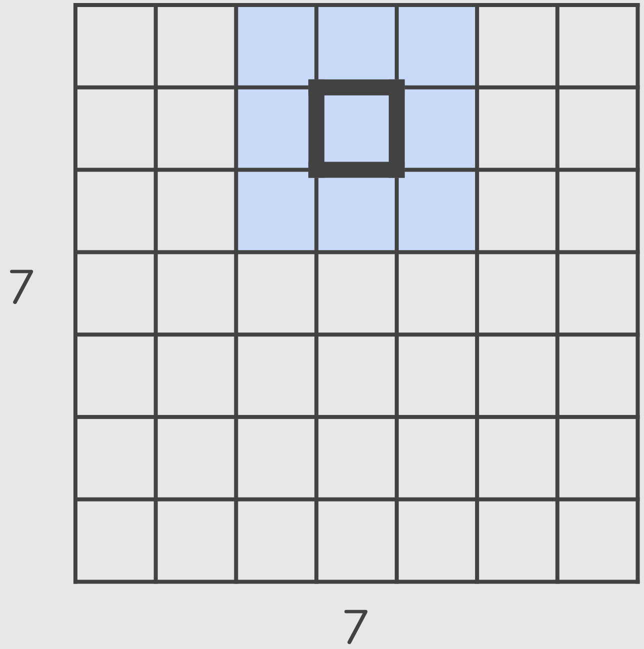
$\Rightarrow$ 5×5 output

A Closer Look at Spatial Dimensions



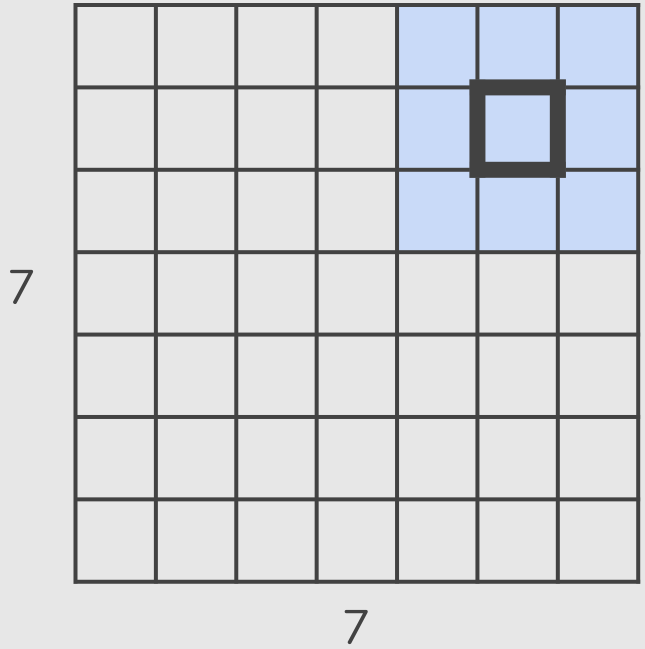
7×7 input (spatially)
assume 3×3 filter
applied with **stride 2**

A Closer Look at Spatial Dimensions



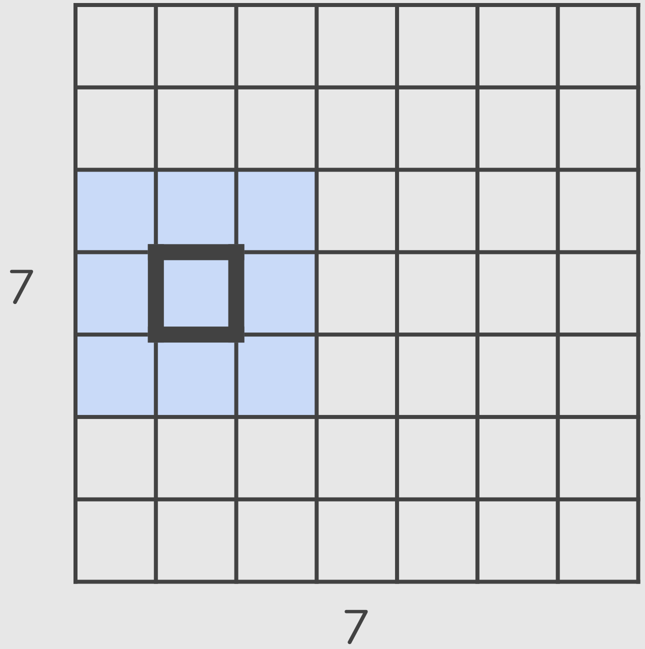
7×7 input (spatially)
assume 3×3 filter
applied with **stride 2**

A Closer Look at Spatial Dimensions



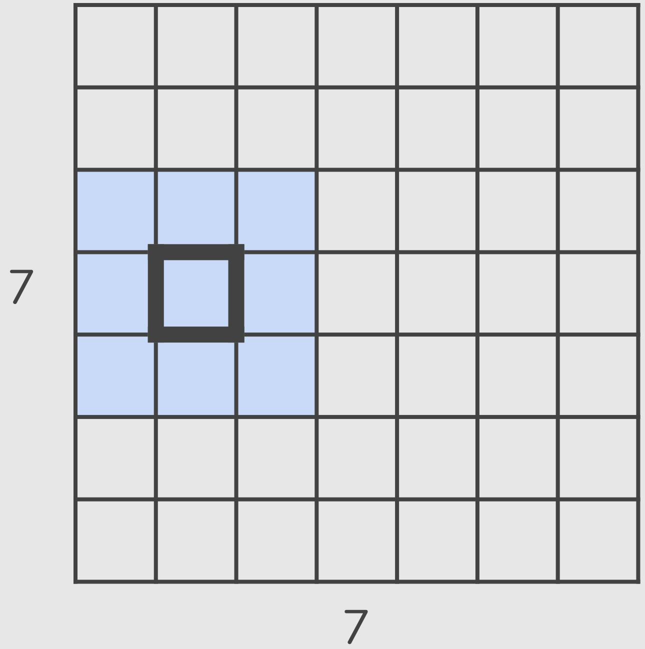
7×7 input (spatially)
assume 3×3 filter
applied with **stride 2**

A Closer Look at Spatial Dimensions



7×7 input (spatially)
assume 3×3 filter
applied with **stride 2**

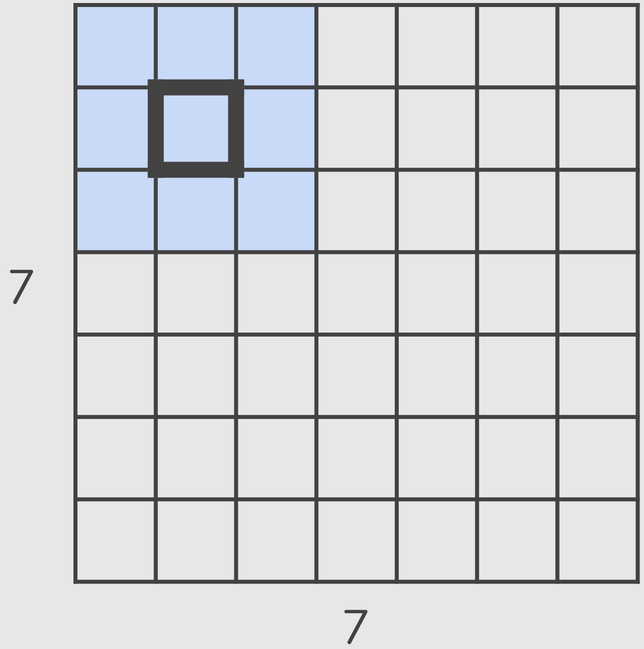
A Closer Look at Spatial Dimensions



7×7 input (spatially)
assume 3×3 filter
applied with **stride 2**

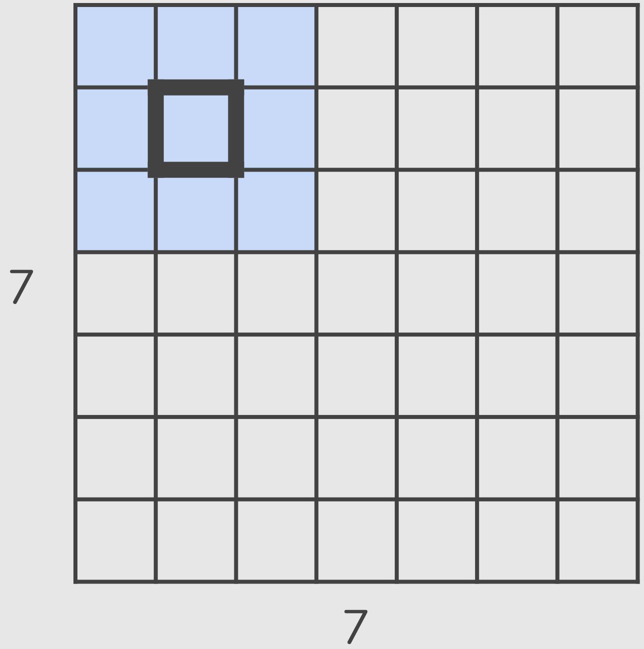
⇒ 3×3 output

A Closer Look at Spatial Dimensions



7×7 input (spatially)
assume 3×3 filter
applied with **stride 3?**

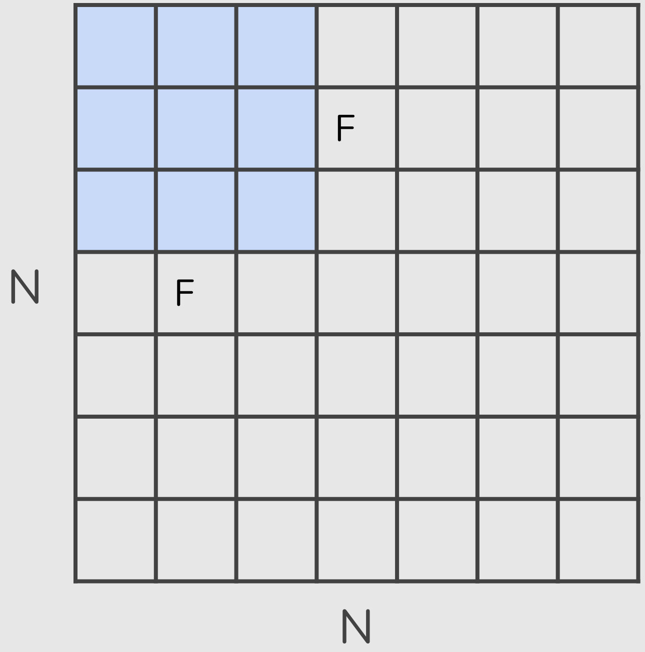
A Closer Look at Spatial Dimensions



7 × 7 input (spatially)
assume 3 × 3 filter
applied with **stride 3?**

Doesn't fit!
cannot apply 3 × 3 filter on
7 × 7 input with stride 3.

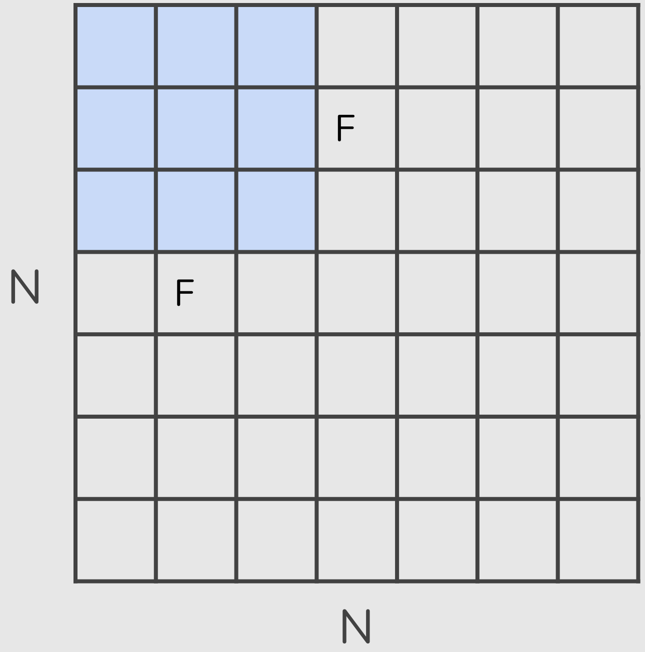
A Closer Look at Spatial Dimensions



Output size:

(N - F) / stride + 1

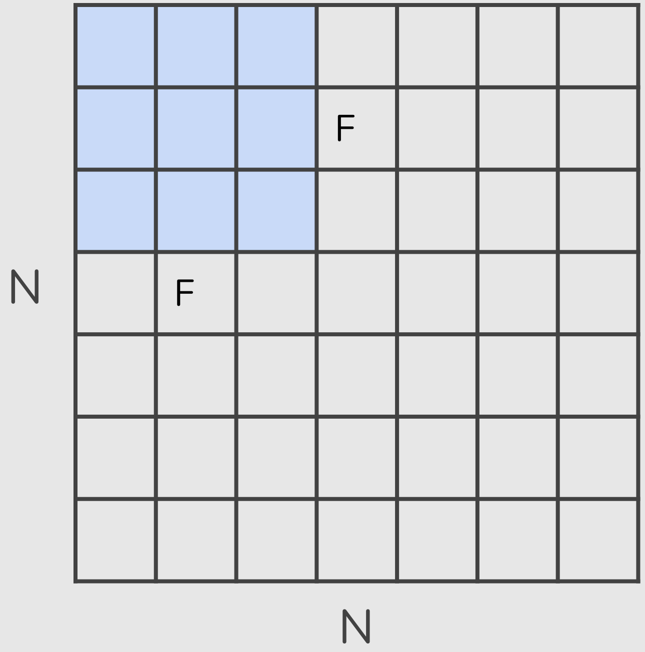
A Closer Look at Spatial Dimensions



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

e.g. N = 7, F = 3:
stride 1 ⇒ (7 - 3)/1 + 1 = 5

A Closer Look at Spatial Dimensions



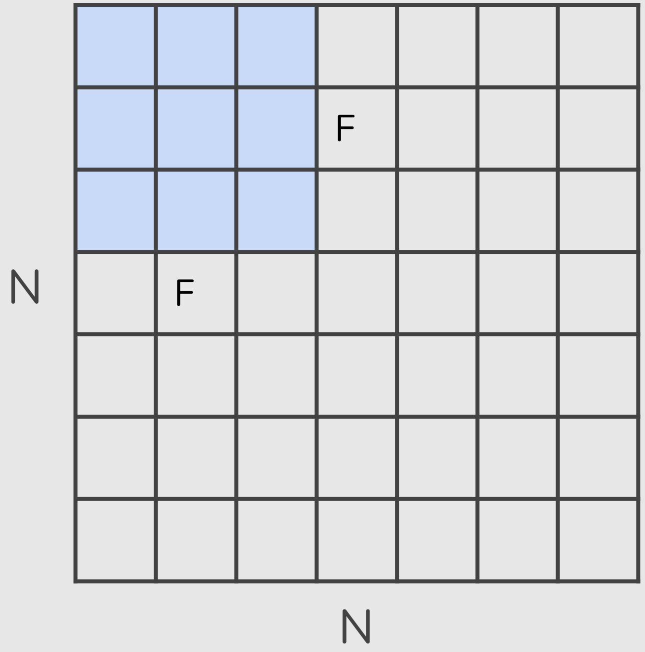
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:

stride 1 ⇒ (7 - 3)/1 + 1 = 5

stride 2 ⇒ (7 - 3)/2 + 1 = 3

A Closer Look at Spatial Dimensions



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

e.g. N = 7, F = 3:

stride 1 ⇒ (7 - 3)/1 + 1 = 5

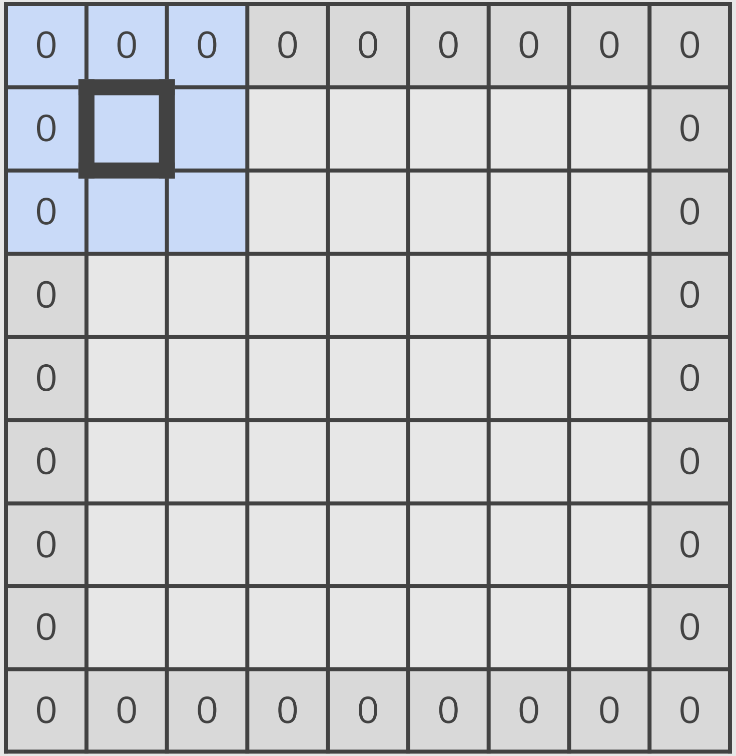
stride 2 ⇒ (7 - 3)/2 + 1 = 3

stride 3 ⇒ (7 - 3)/3 + 1 = 2.33

In Practice: Common to zero pad the border

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

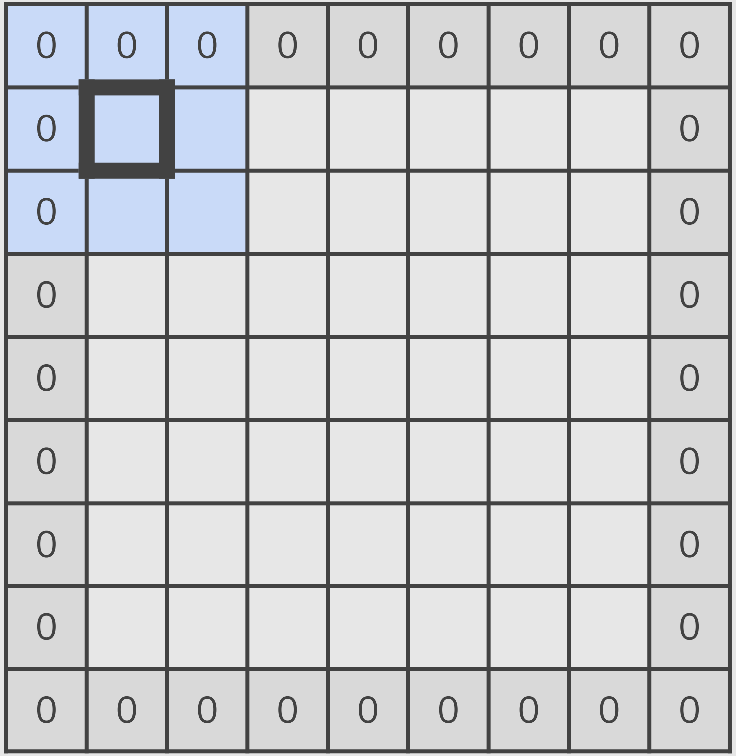
In Practice: Common to zero pad the border



7 × 7 input,
3 × 3 filter applied
with **stride 1 with pad 1**

What is the output?

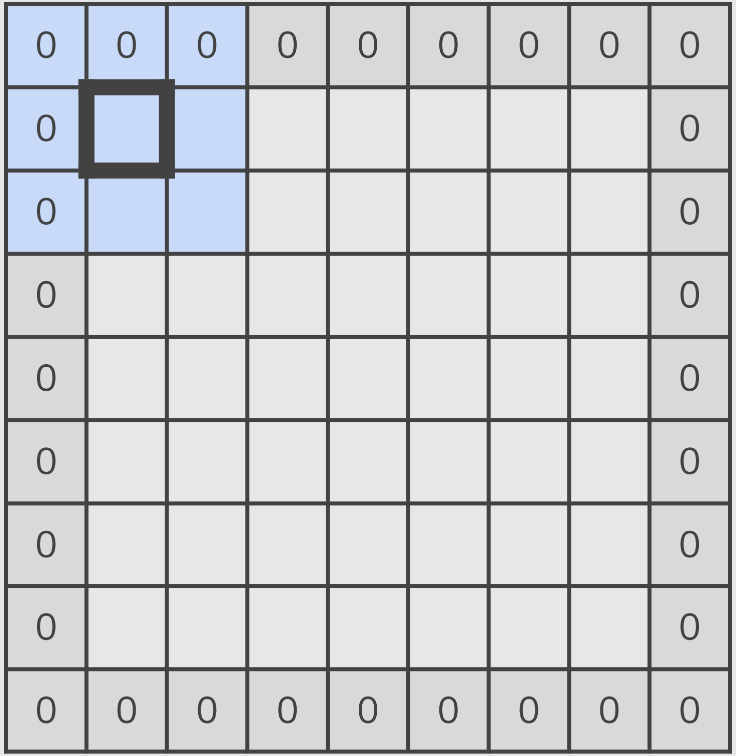
In Practice: Common to zero pad the border



7×7 input,
3×3 filter applied
with **stride 1 with pad 1**

What is the output?
7×7 output

In Practice: Common to zero pad the border



In general, common to see CONV layers with stride 1, filters of size F × F, and zero-padding with (F-1)/2 **(will preserve size spatially)**.

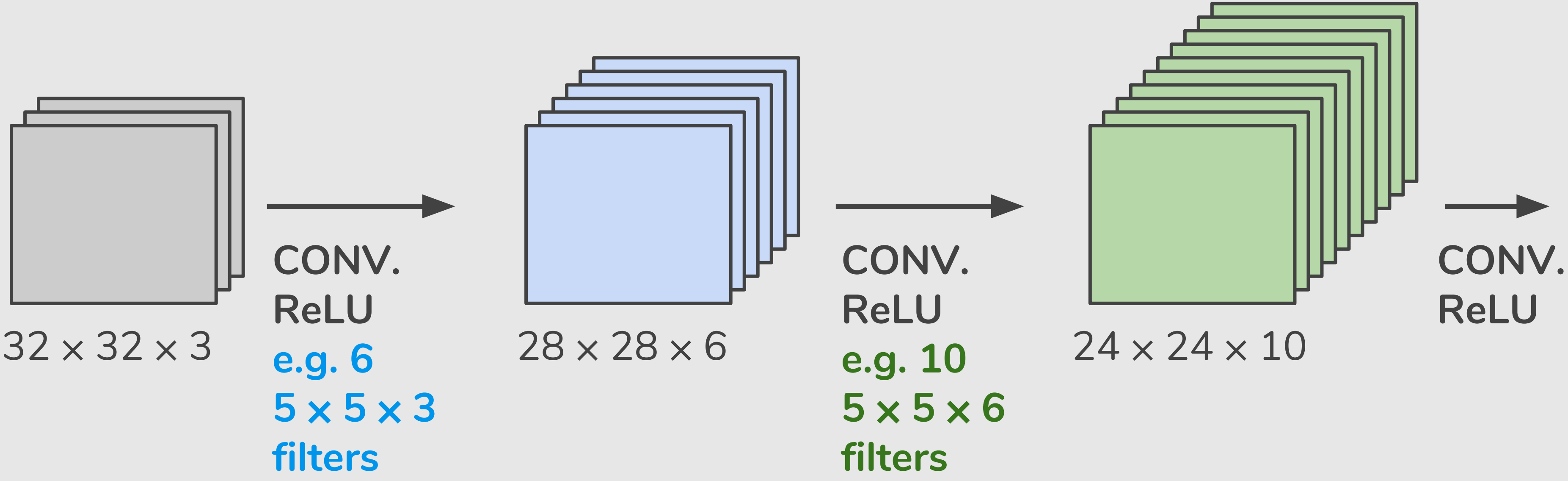
e.g. F = 3 ⇒ zero pad with 1

F = 5 ⇒ zero pad with 2

F = 7 ⇒ zero pad with 3

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

$32 \rightarrow 28 \rightarrow 24 \rightarrow \ldots$



Number of Parameters

Input volume: $\mathbf{32 \times 32 \times 3}$

10 5×5 filters with stride 1, pad 2

Number of parameters in this layer?

Number of Parameters

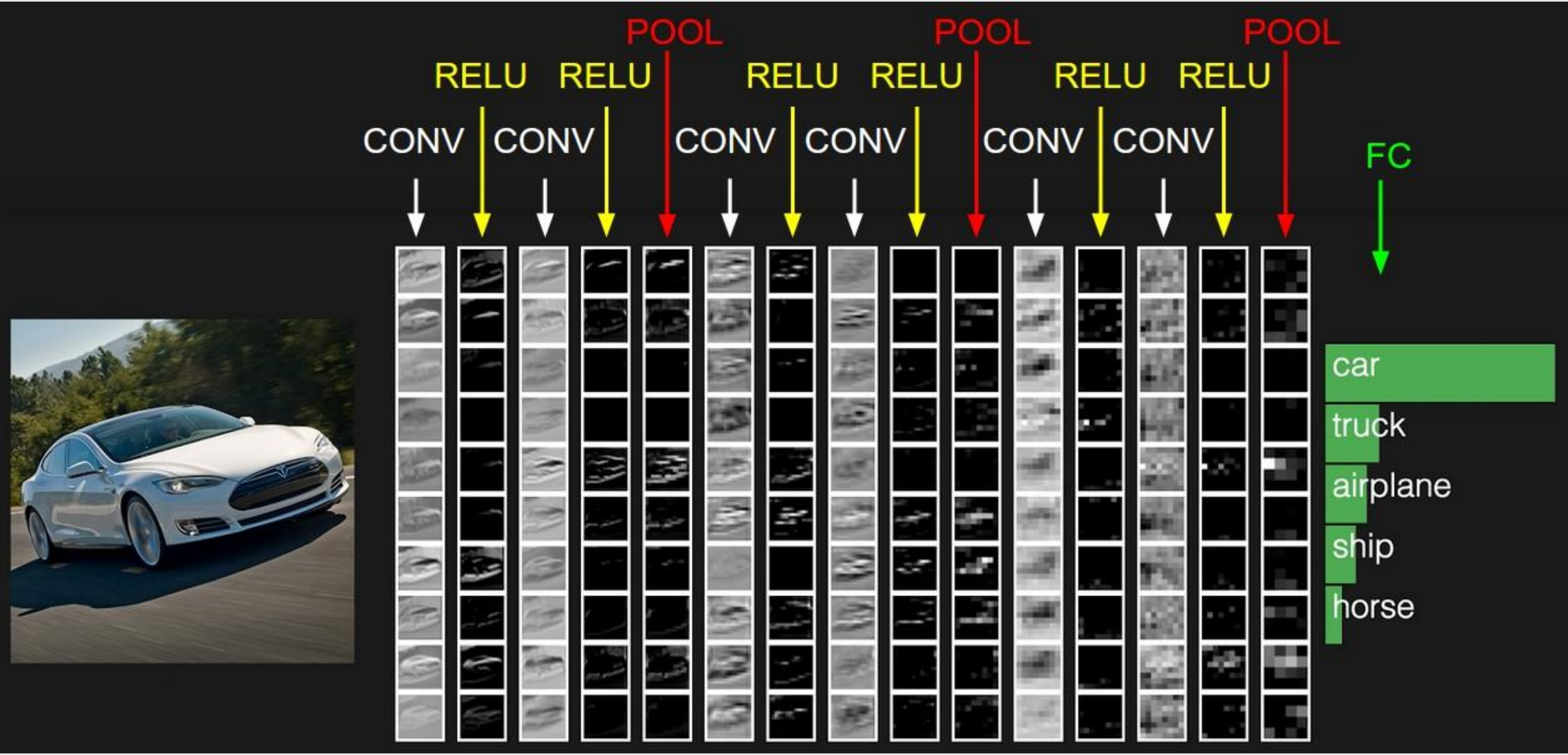
Input volume: **32** $\times$ **32** $\times$ **3**

10 **5** $\times$ **5** filters with stride 1, pad 2

Number of parameters in this layer?

Each filter has 5*5*3 + 1 = **76** parameters (+1 for bias)

=> **76***10** = 760



Credit: http://cs231n.github.io/convolutional-networks/

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently

Pooling Layer

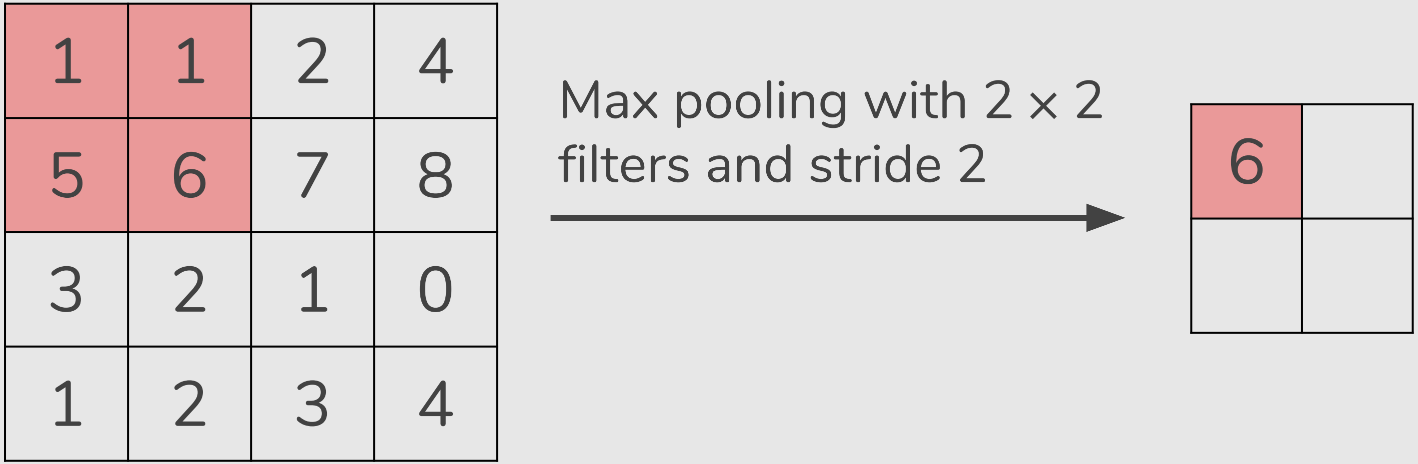
- Makes the representations smaller and more manageable
- Operates over each activation map independently

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

Max pooling with 2×2 filters and stride 2 $\longrightarrow$

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently

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

Max pooling with 2×2 filters and stride 2 →

6	8

Pooling Layer

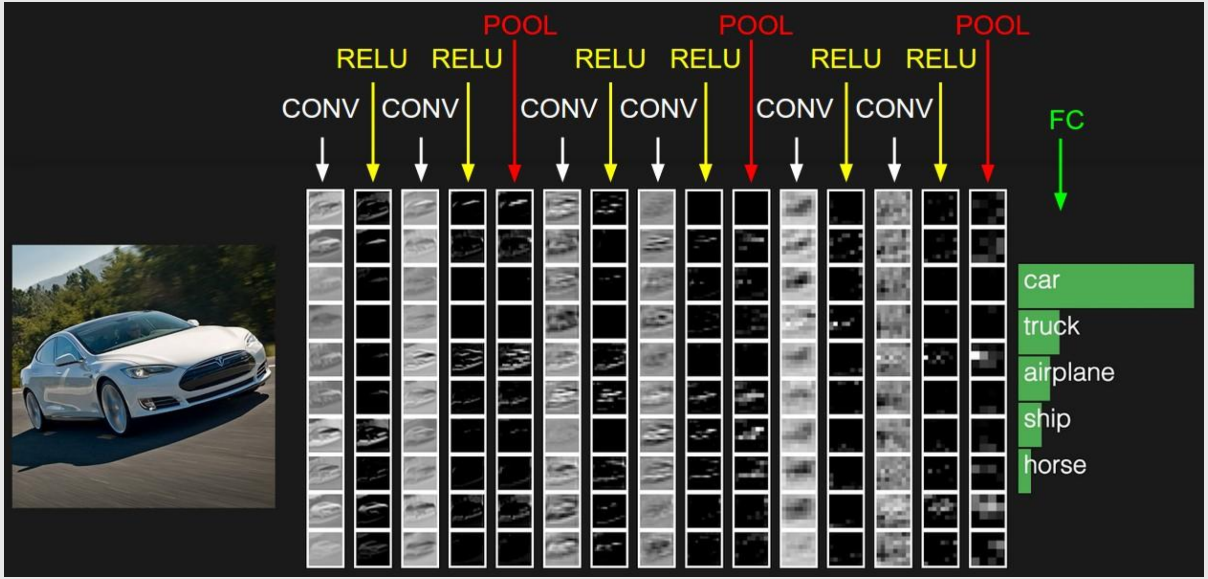
- Makes the representations smaller and more manageable
- Operates over each activation map independently

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently

Fully Connected Layer

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Credit: http://cs231n.github.io/convolutional-networks/

http://neuralnetworksanddeeplearning.com/chap6.html#final_conv

neuralnetworksanddeeplearning.com/chap6.html#final_conv Search

Convolutional neural networks in practice

We've now seen the core ideas behind convolutional neural networks. Let's look at how they work in practice, by implementing some convolutional networks, and applying them to the MNIST digit classification problem. The program we'll use to do this is called `network3.py`, and it's an improved version of the programs `network.py` and `network2.py` developed in earlier chapters*. If you wish to follow along, the code is available on GitHub. Note that we'll work through the code for `network3.py` itself in the next section. In this section, we'll use `network3.py` as a library to build convolutional networks.

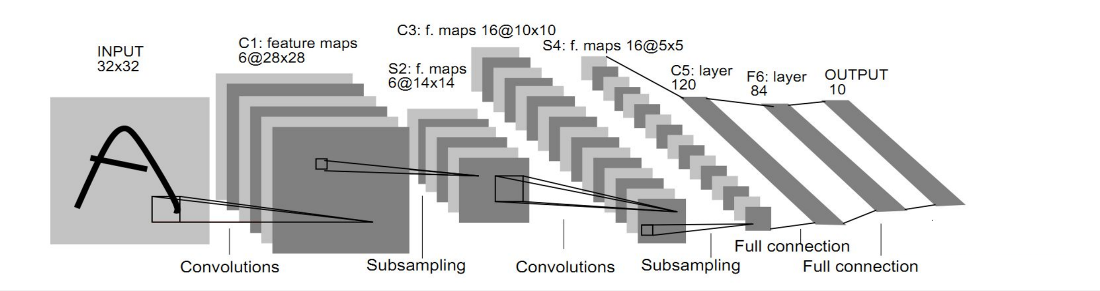
*Note also that `network3.py` incorporates ideas from the Theano library's documentation on convolutional neural nets (notably the implementation of LeNet-5), from Misha Denil's implementation of dropout, and from Chris Olah.

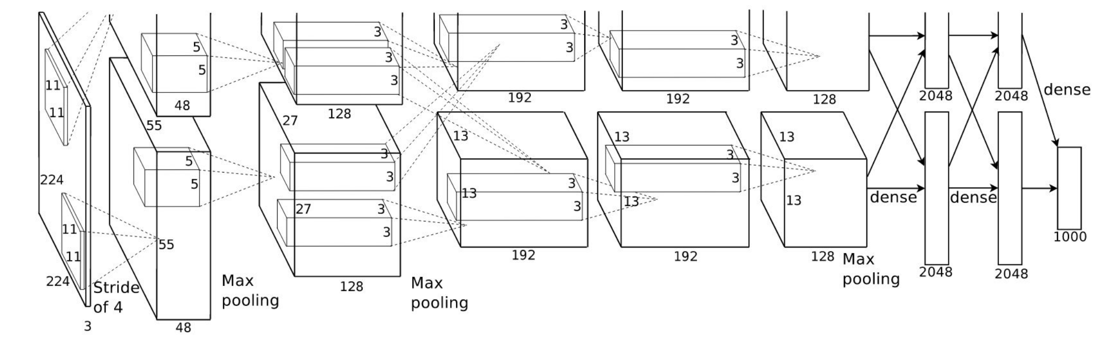
CNNs Architectures

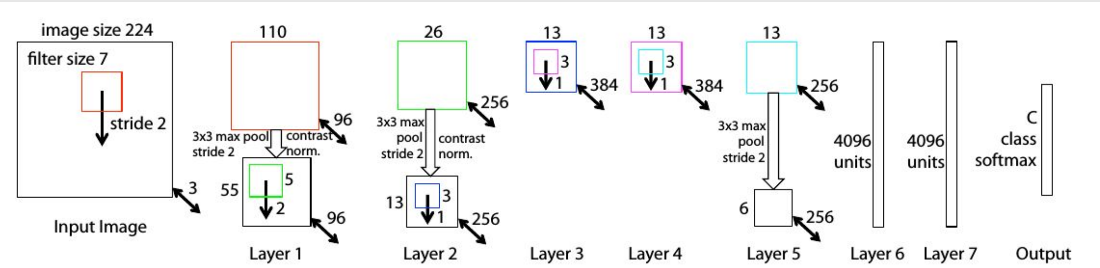
CNNs Architectures

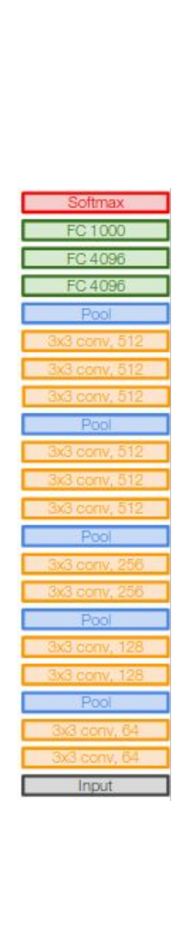
- **LeNet** by Yann LeCun, Léon Bottou & Yoshua Bengio (1998)
- **AlexNet** by Alex Krizhevsky, Ilya Sutskever & Geoff Hinton (2012)
- **ZF Net** by Matthew Zeiler & Rob Fergus (2013)
- **GoogLeNet** by Szegedy et al. (2014)
- **VGGNet** by Karen Simonyan & Andrew Zisserman (2014)
- **ResNet** by Kaiming He et al. (2015)

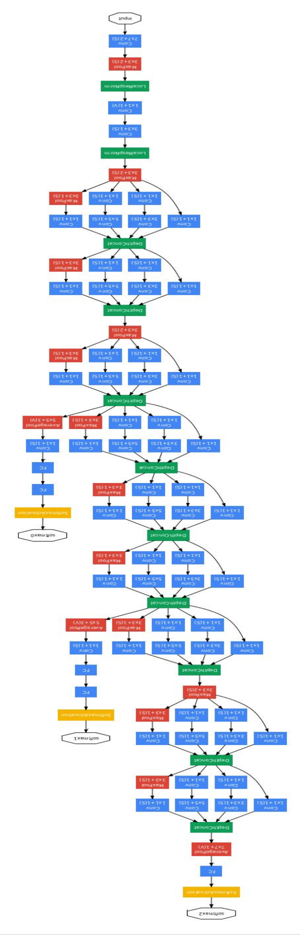
CNN-based Architectures

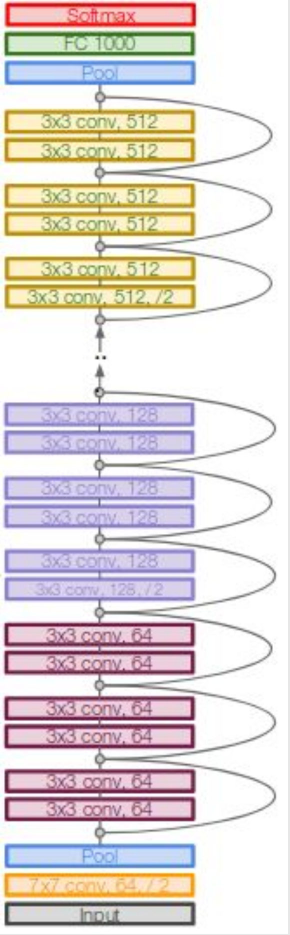












References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 11 & 13

Machine Learning Courses

- https://www.coursera.org/learn/neural-networks
- “The 3 popular courses on Deep Learning”: https://medium.com/towards-data-science/the-3-popular-courses-for-deeplearning-ai-ac37d4433bd