Convolutional Neural Networks

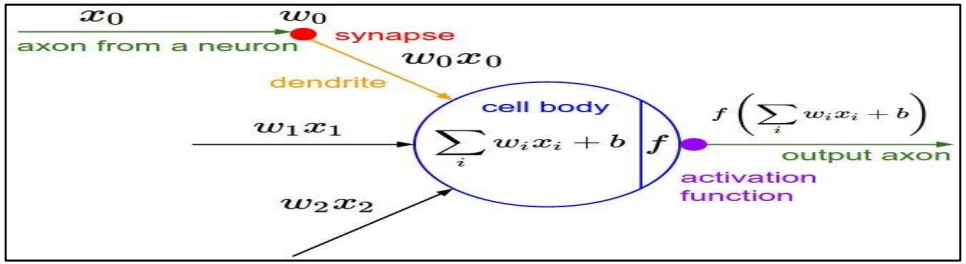
Dr. Dinesh Kumar Vishwakarma
Professor, Department of Information Technology
Delhi Technological University, Delhi
Email: dinesh@dtu.ac.in
Web page: http://www.dtu.ac.in/Web/Departments/InformationTechnology/faculty/dkvishwakarma.php
Biometric Research Laboratory
http://www.dtu.ac.in/Web/Departments/InformationTechnology/lab_and_infra/bml/

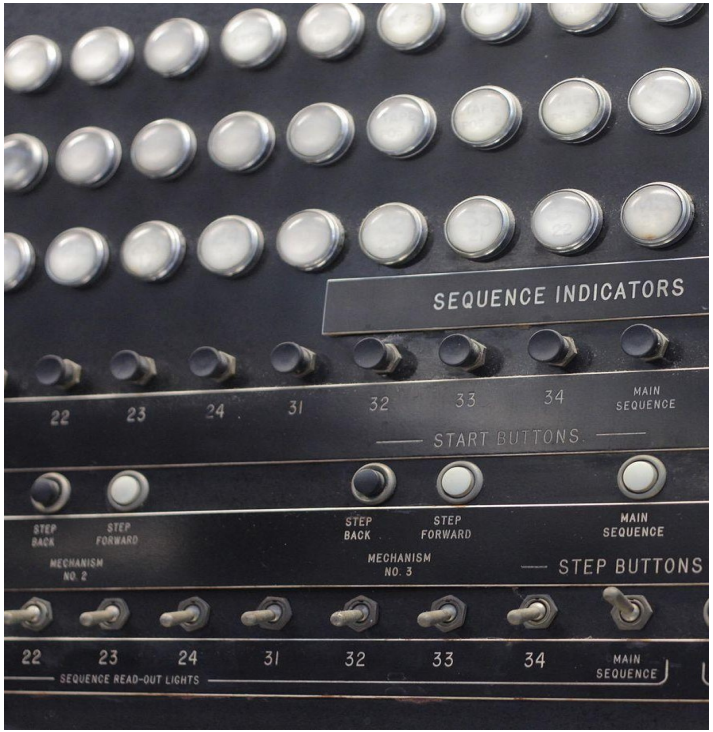
What is CNN?

- A **Convolutional Neural Network** (**ConvNet/CNN**) is a Deep Learning algorithm.
- It can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.
- The **pre-processing** required in a ConvNet is **much lower** as compared to other classification algorithms.
- While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/characteristics.
- The architecture of a **CNN is analogous** to that of the **connectivity pattern of Neurons** in the Human Brain and was inspired by the organization of the **Visual Cortex**.

A bit of history...

- The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.
- The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.
- Recognized letters of the alphabet $f(x) = \begin{cases} 1 \ if \ \omega.x + b > 0 \\ 0 \ otherwise \end{cases}$
- update rule: $\omega_i(t+1) = \omega_i(t) + \eta\left(d_j - y_j(t)\right)x_{ji}$ Frank Rosenblatt, ~1957: Perceptron





A bit of history...

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}$$

recognizable math

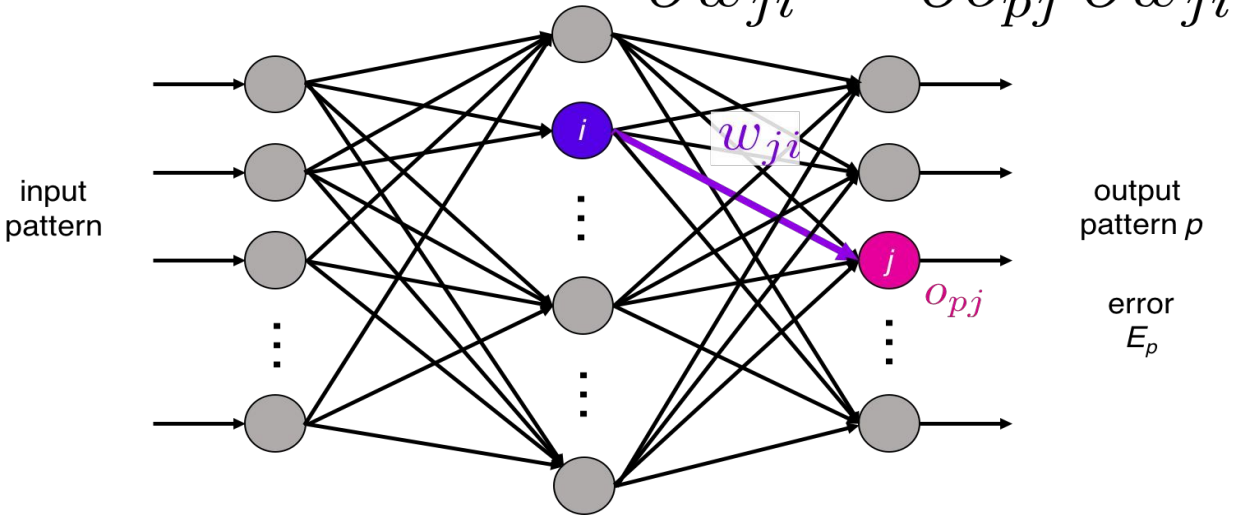


Illustration of Rumelhart et al., 1986 by Lane McIntosh

Rumelhart et al., 1986: First time back-propagation became popular

A bit of history...

[Hinton and Salakhutdinov 2006]

Reinvigorated (a new energy) **research in Deep Learning**

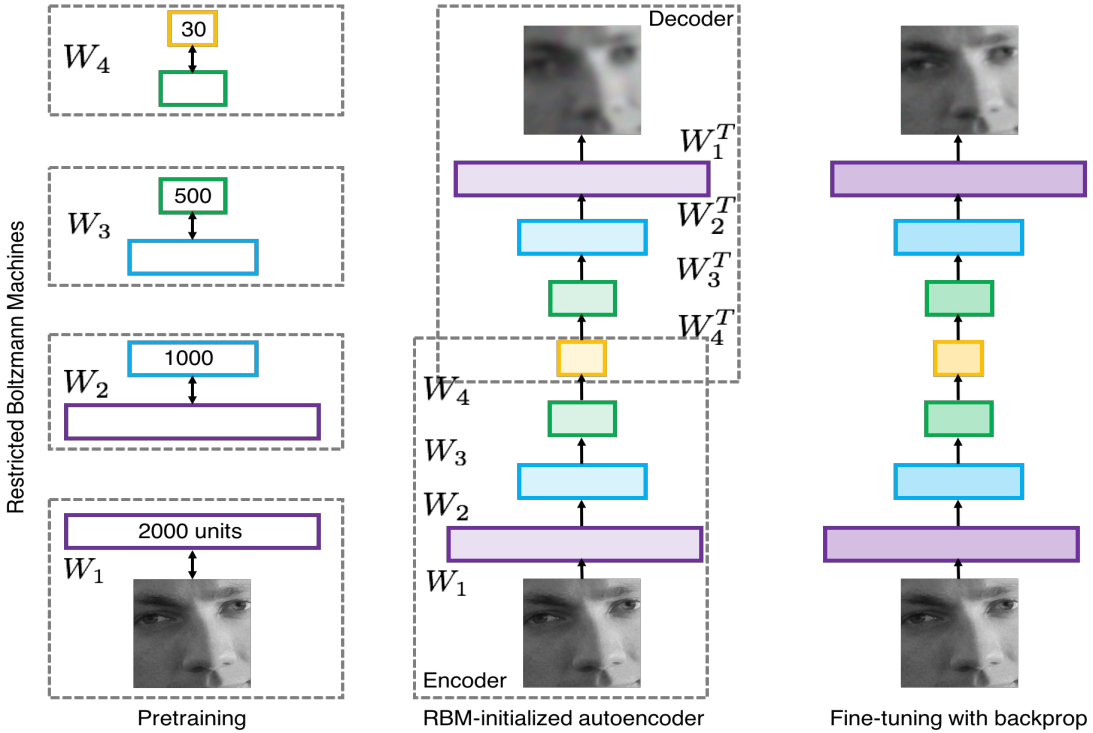


Illustration of Hinton and Salakhutdinov 2006 by Lane McIntosh

First Strong Results

- ***Acoustic Modeling using Deep Belief Networks***, *Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010*
- ***Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition***, George Dahl, Dong Yu, Li Deng, Alex Acero, 2012.
- ***Imagenet classification with deep convolutional neural networks***, Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012

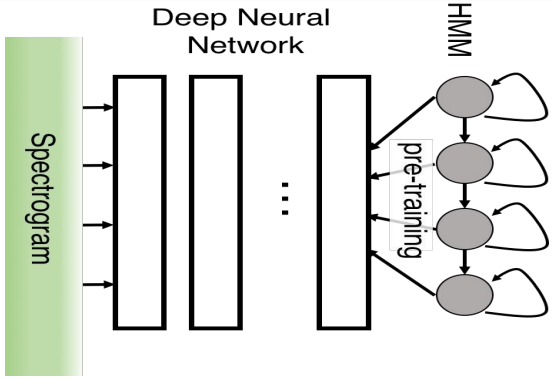
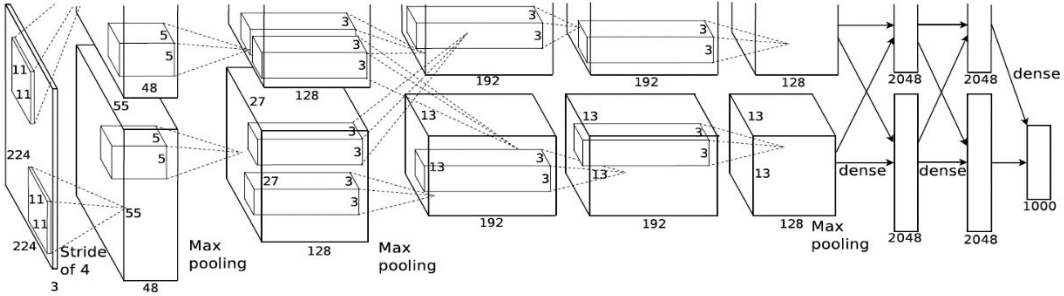
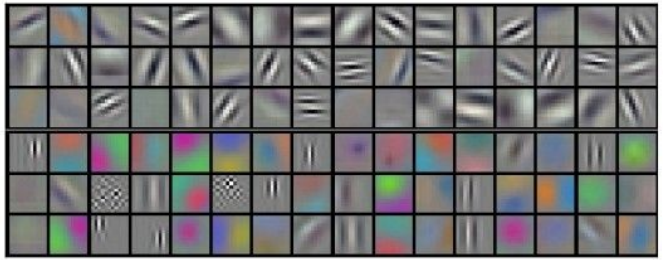


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

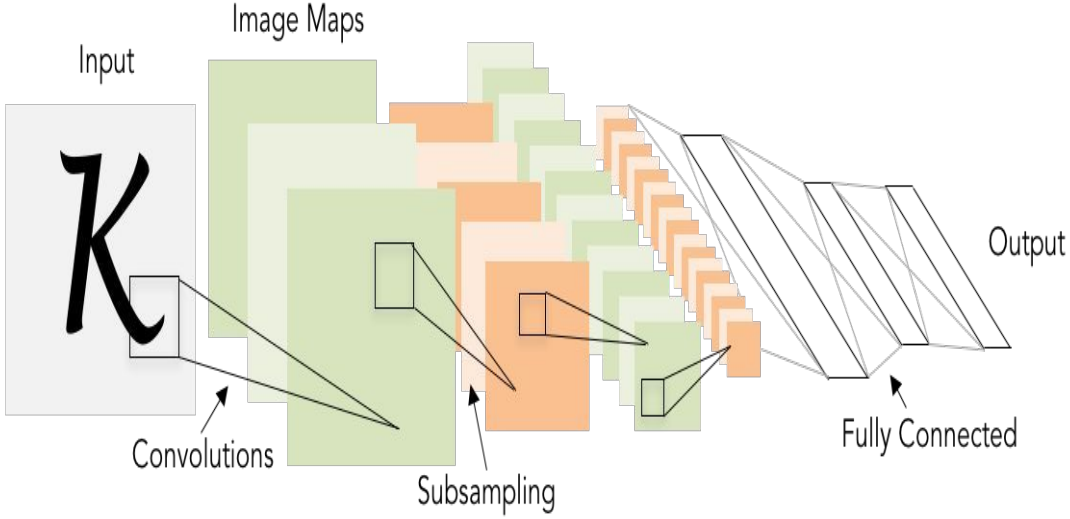


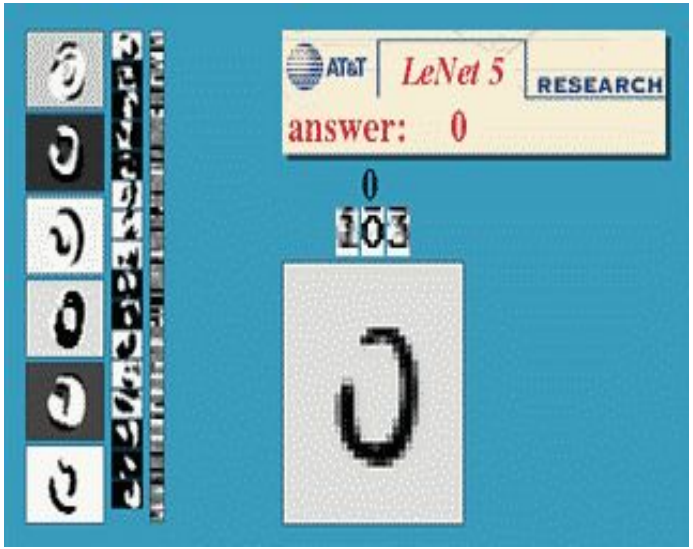


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

A bit of history:

- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.





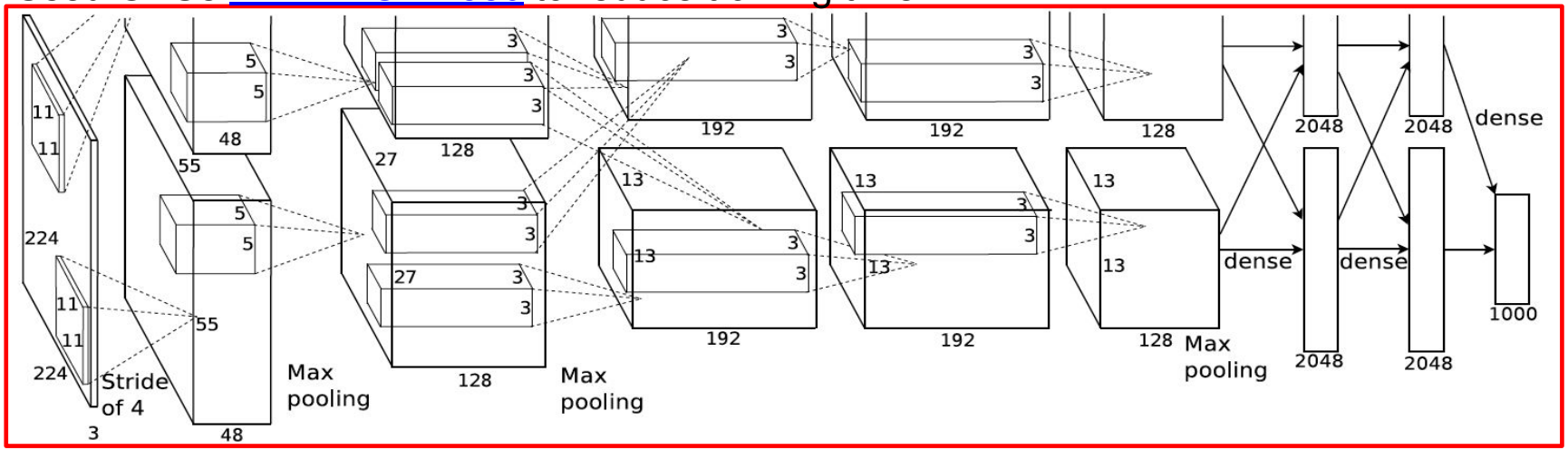
LeNet-5 **Implemented on CPU**

A bit of history…

The success of AlexNet gives small Revolution

- It is deeper and much wider version of the LeNet and won by a large margin the difficult **ImageNet competition**.
- AlexNet is scaled LeNet in to much large NN and it can learn more **complex objects** and **object hierarchies**.
- Used GPUs [NVIDIA GTX 580] to reduce training time.

"AlexNet"



ImageNet Classification with Deep Convolutional Neural Networks *[Krizhevsky, Sutskever, Hinton, 2012]., NIPS citation:* 59454 (02.04.2020)

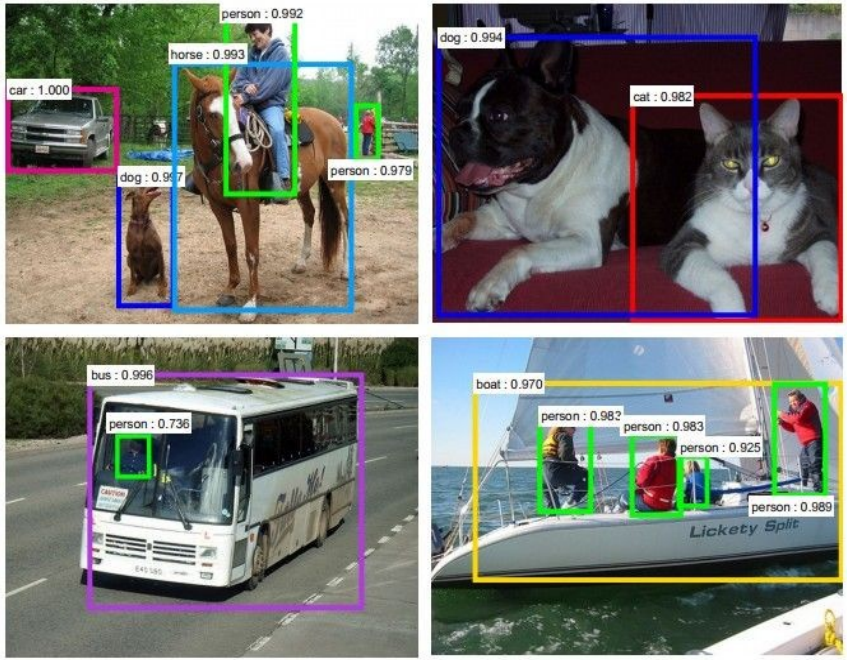
Fast-forward to today: ConvNets are everywhere

Classification Retrieval

Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

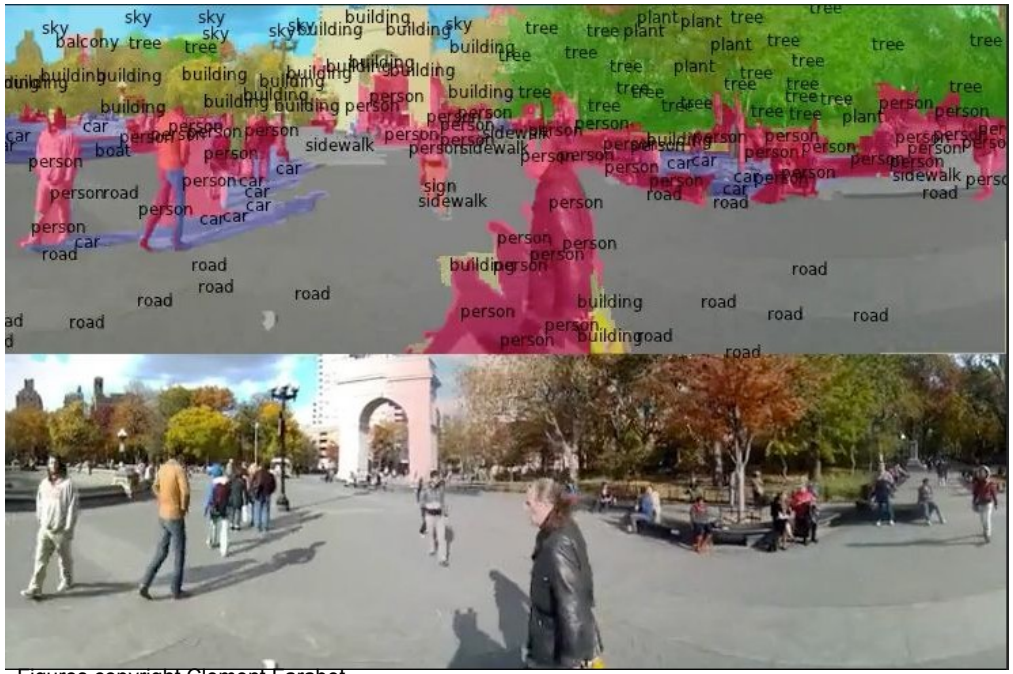
Fast-forward to today: ConvNets are everywhere

Detection Segmentation



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

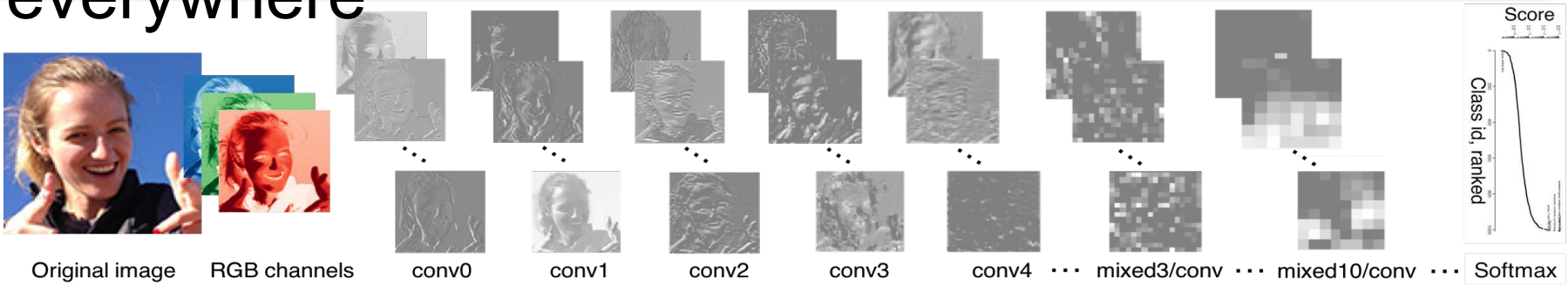
Fast-forward to today: ConvNets are everywhere

self-driving cars

Photo by Lane McIntosh. Copyright CS231n 2017.

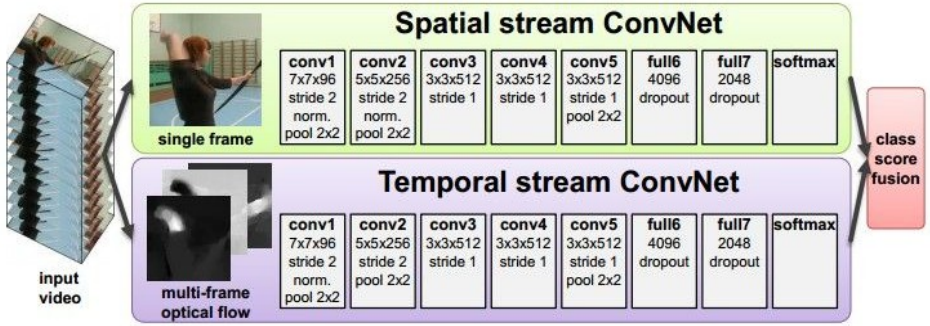
Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

Fast-forward to today: ConvNets are everywhere



[Taigman et al. 2014]

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

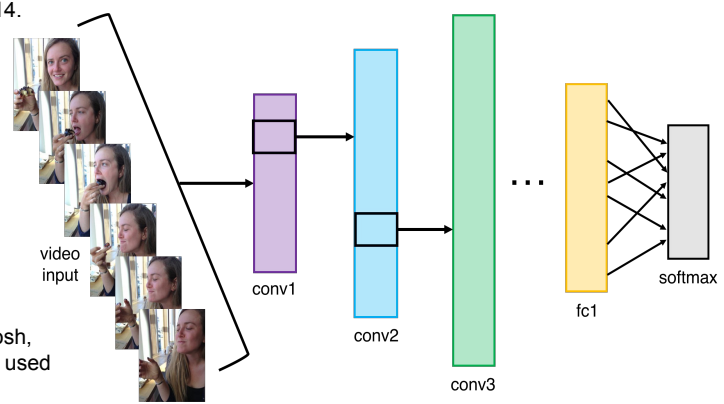
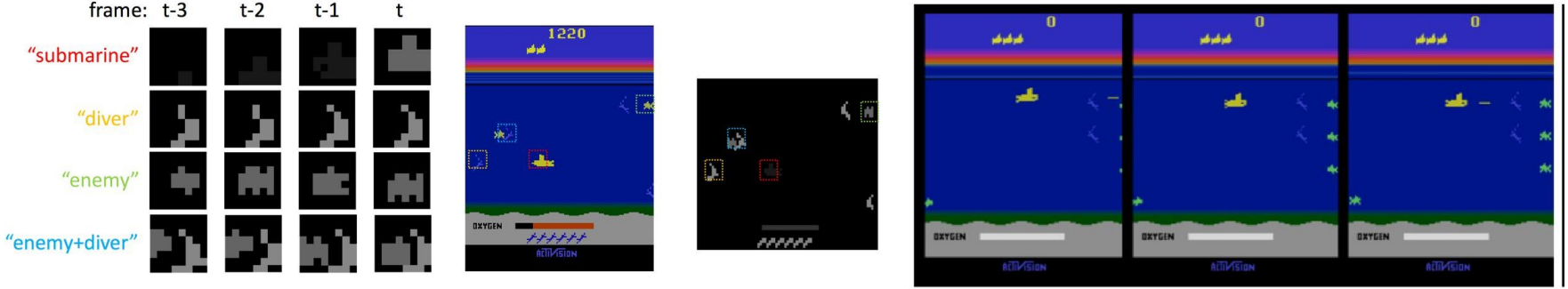


Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

Fast-forward to today: ConvNets are everywhere

[Toshev, Szegedy 2014]

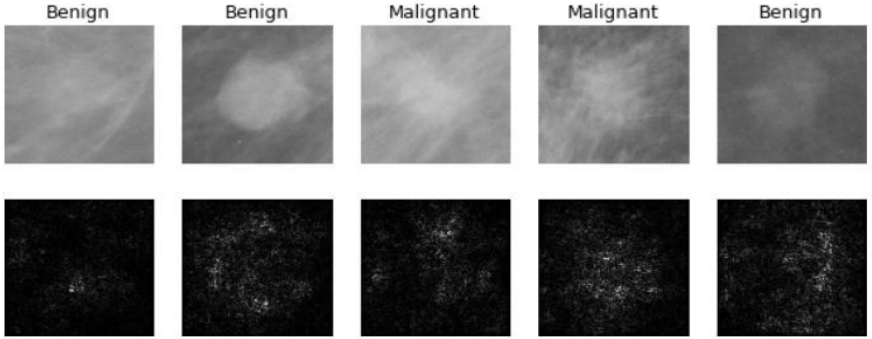
Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.



[Guo et al. 2014]

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

Fast-forward to today: ConvNets are everywhere



[Levy et al. 2016]

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011]
[Ciresan et al.]

Photos by Lane McIntosh.
Copyright CS231n 2017.

Image Captioning

No errors | **Minor errors** | **Somewhat related**

A white teddy bear sitting in the grass

A man in a baseball uniform throwing a ball

A woman is holding a cat in her hand

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

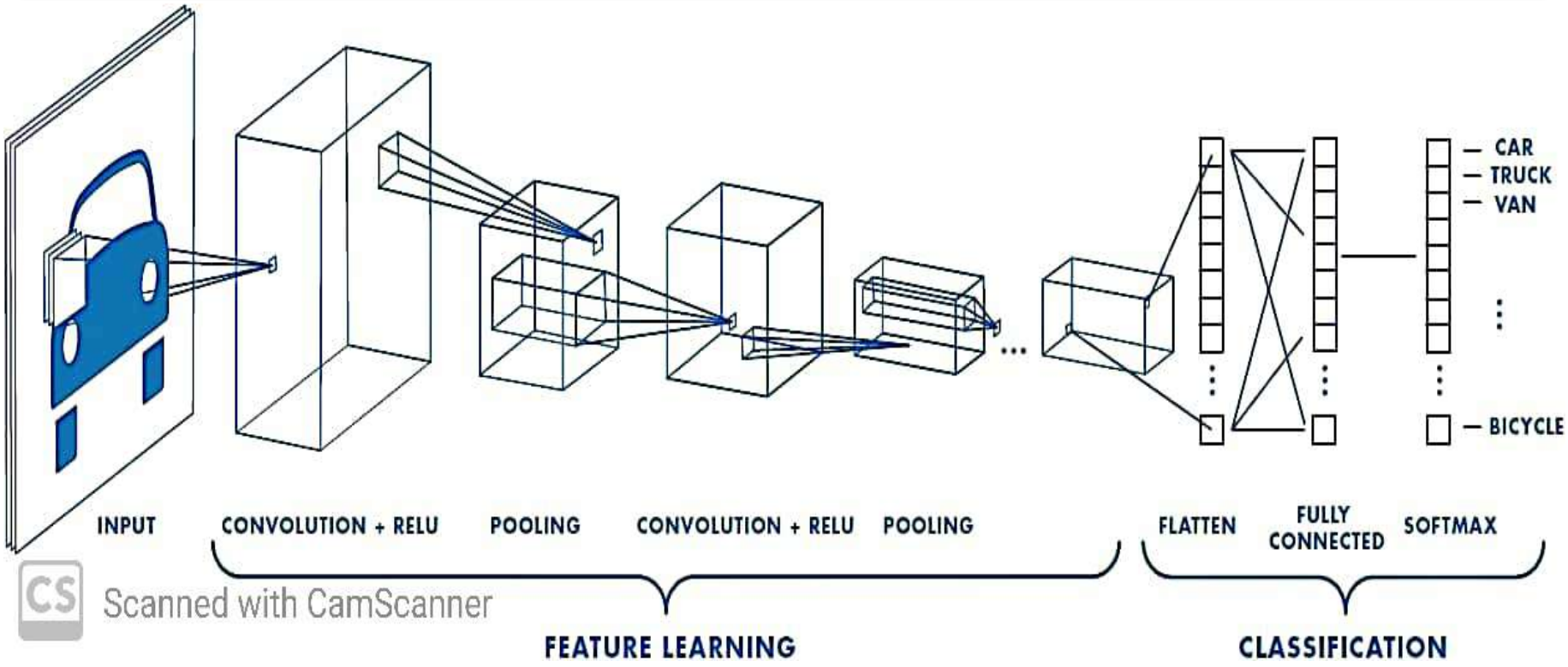
A woman standing on a beach holding a surfboard

All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

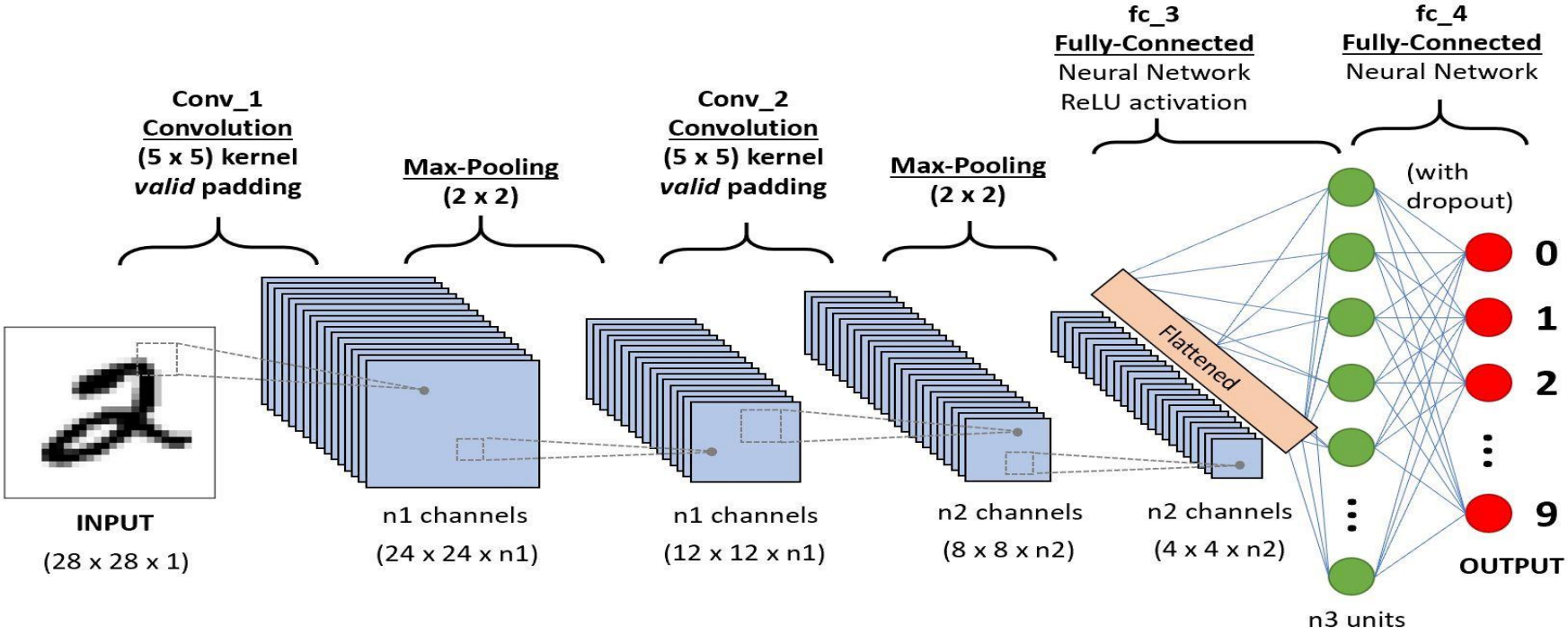
Captions generated by Justin Johnson using Neuraltalk2

CNN Fundamentals

Fundamental Architecture of CNN



Fundamental Architecture of CNN



Why CNN not Feedforward Neural Network?

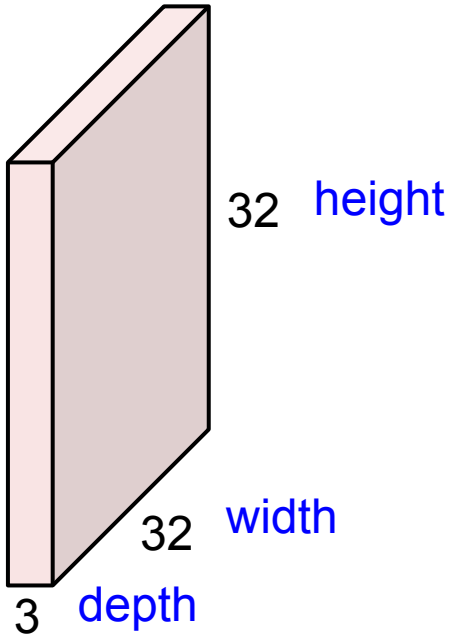
- An image is nothing but a matrix of pixel values, So why not just flatten the image and feed it to a Multi-Layered Perceptron for classification purposes?

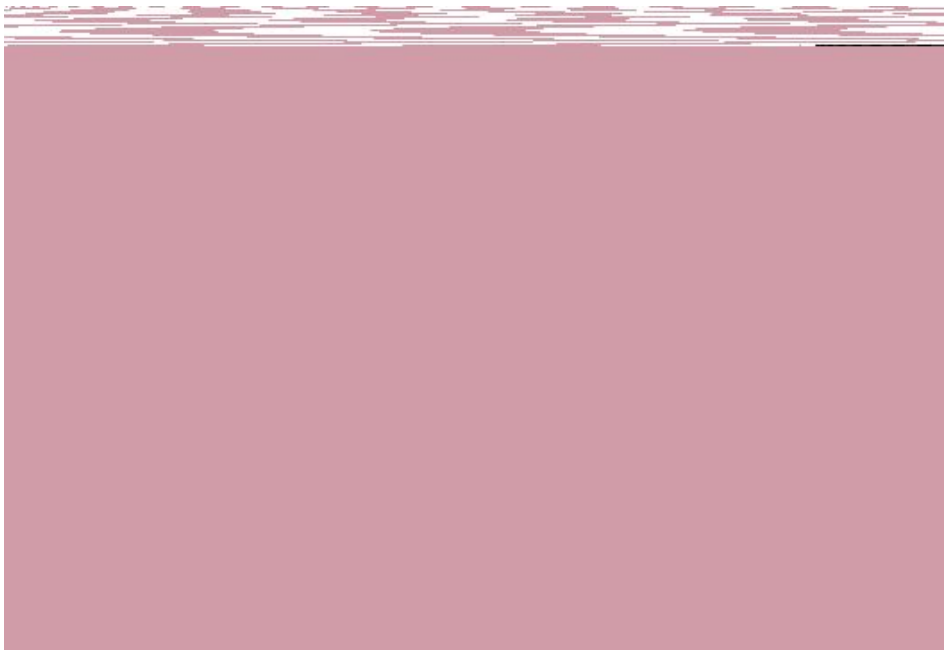
$$\begin{matrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{matrix} \leftrightarrow 1\ \ 2\ 3\ \ 3\ \ 2\ \ 1\ \ 1\ 2\ \ 3$$

- In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.
- A CNN is able to **capture the Spatial and Temporal dependencies** in an image. $y[m,n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h[m,n]x[k-m\,, l-n]$.
- The role of the CNN is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.
- It is very good when we design a system, which can learn features as well as scale the data.

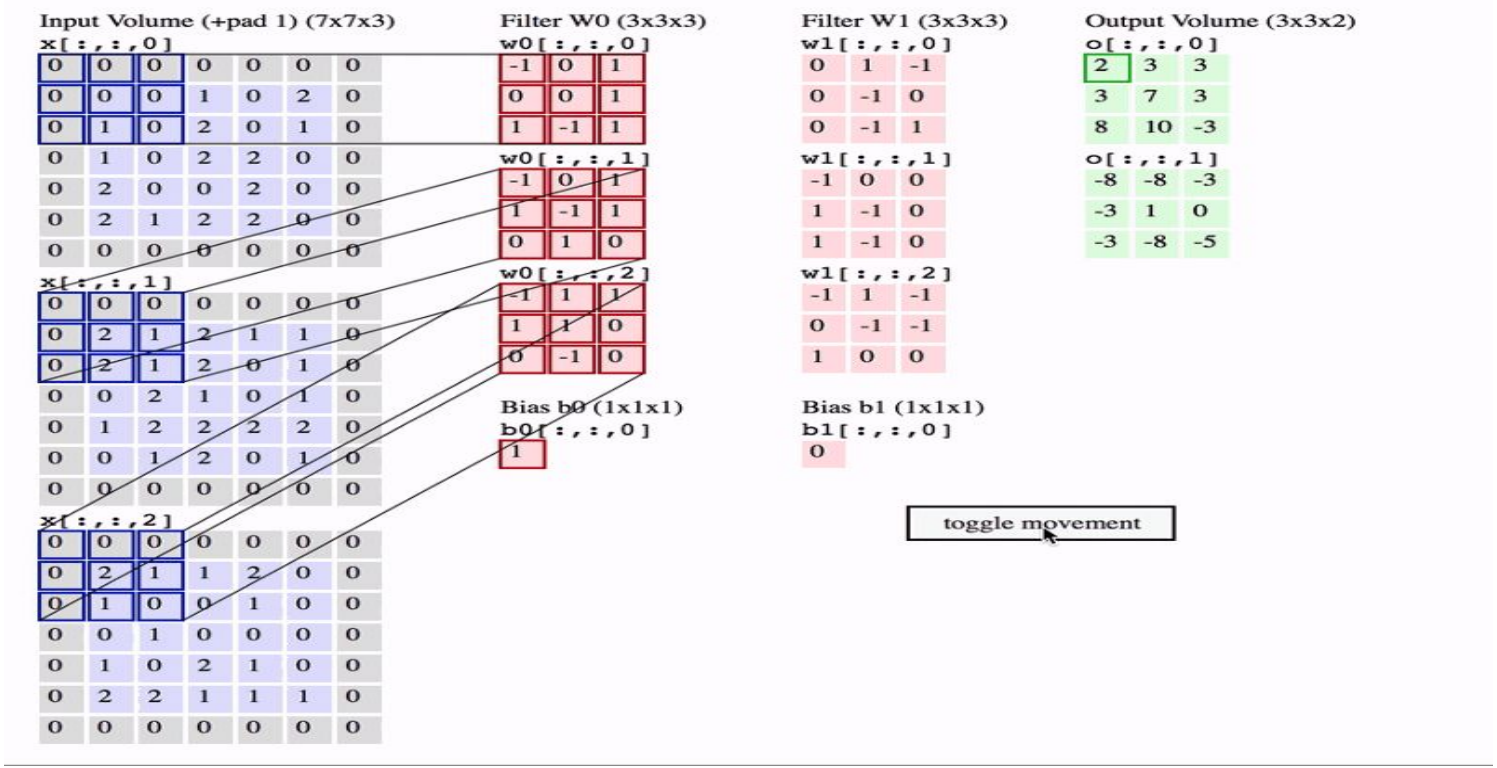
Convolution Layer

32x32x3 image -> preserve spatial structure

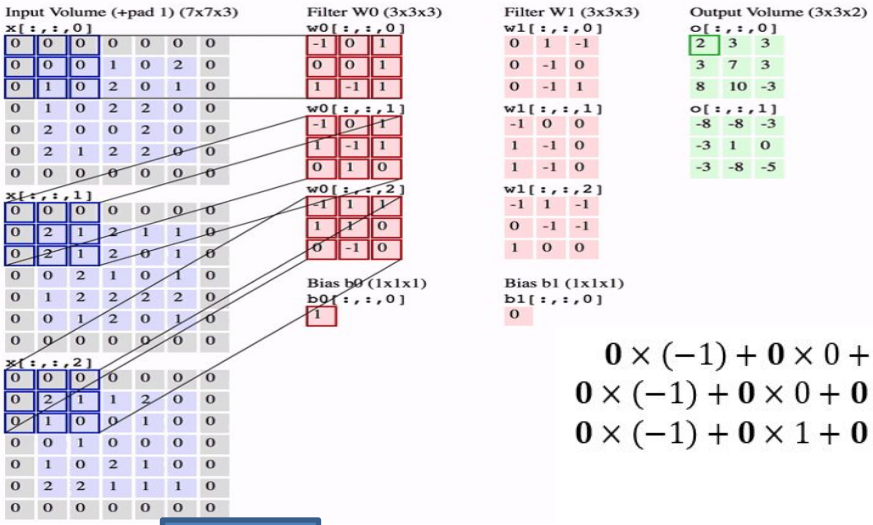




Convolution operation



Convolution operation…

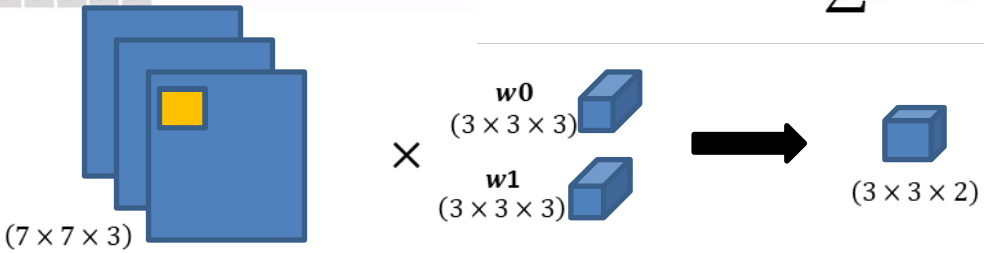


$$\mathbf{0}\times(-1)+\mathbf{0}\times 0+\mathbf{0}\times 1+\mathbf{0}\times 0+\mathbf{0}\times 0+\mathbf{0}\times 1+\mathbf{0}\times 1+\mathbf{1}\times(-1)+\mathbf{0}\times 1=-1$$

$$\mathbf{0}\times(-1)+\mathbf{0}\times 0+\mathbf{0}\times 1+\mathbf{0}\times 1+2\times(-1)+\mathbf{1}\times 1+0\times 0+2\times 1+\mathbf{1}\times 0=1$$

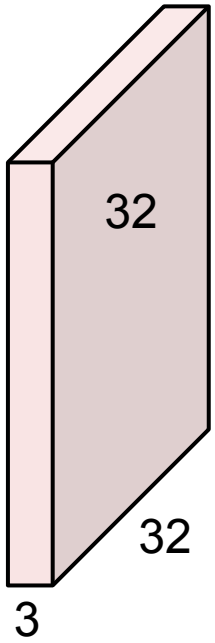
$$\mathbf{0}\times(-1)+\mathbf{0}\times 1+\mathbf{0}\times 1+\mathbf{0}\times 1+2\times 1+\mathbf{1}\times 0+0\times 0+1\times(-1)+0\times 0=1$$

$$\sum(w*x)+\boldsymbol{b0}=(-1+1+1)+\mathbf{1}=2$$

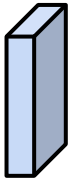


Convolution Layer

32x32x3 image



5x5x3 filter

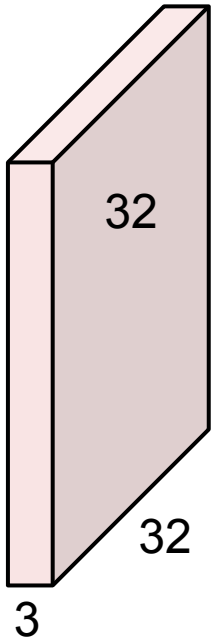


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

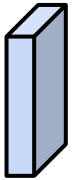
Convolution Layer

32x32x3 image

Filters always extend the full depth of the input volume

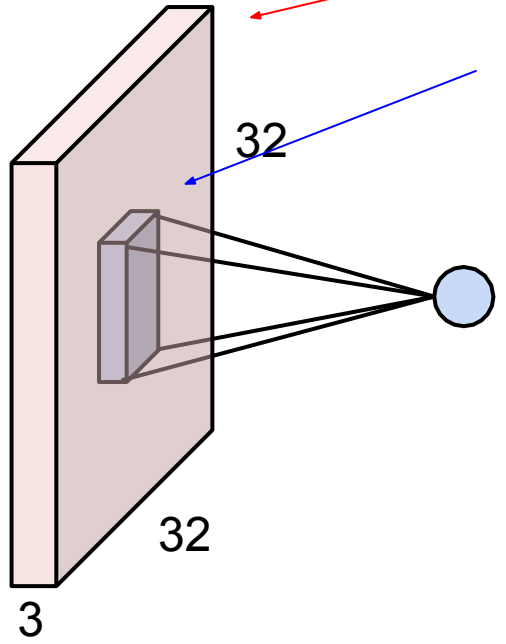


5x5x3 filter



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

Convolution Layer



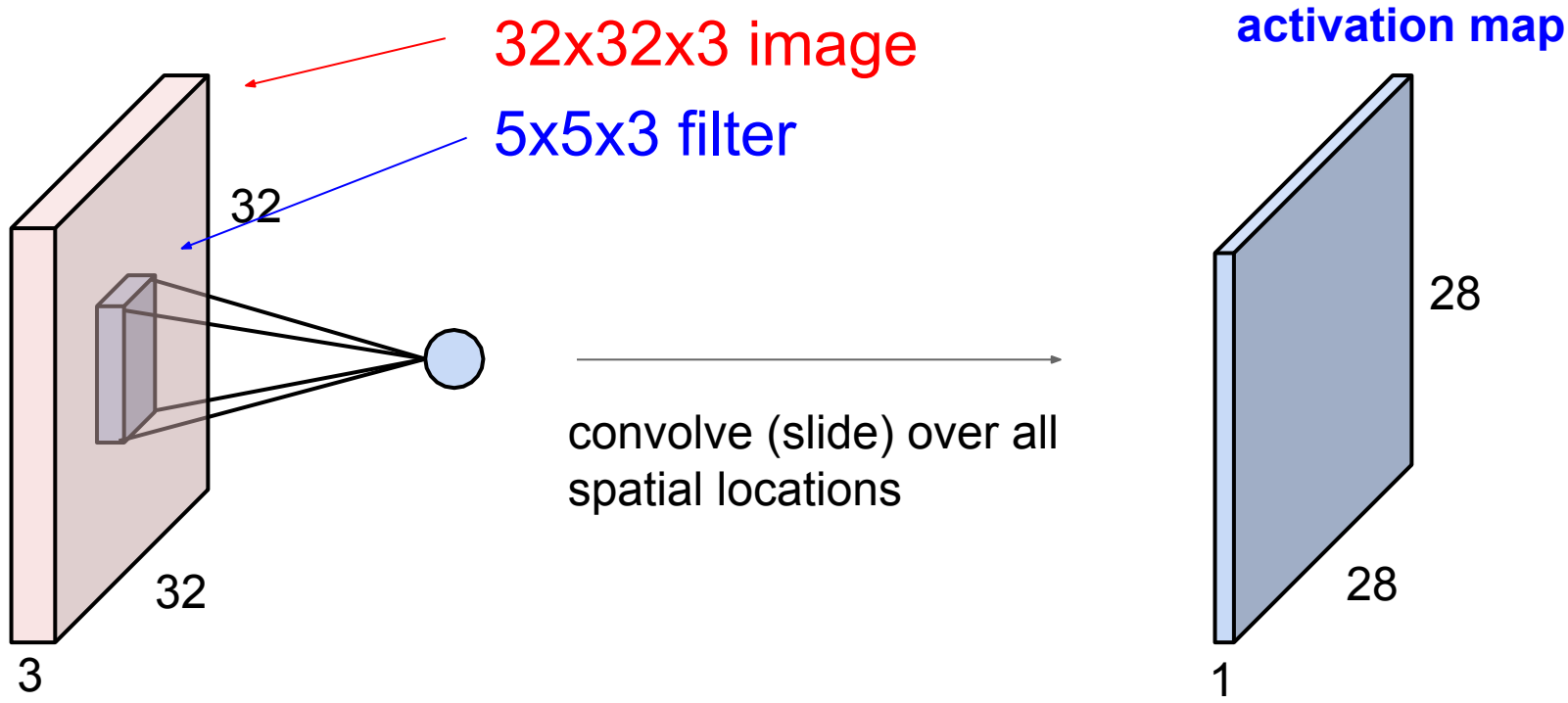
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

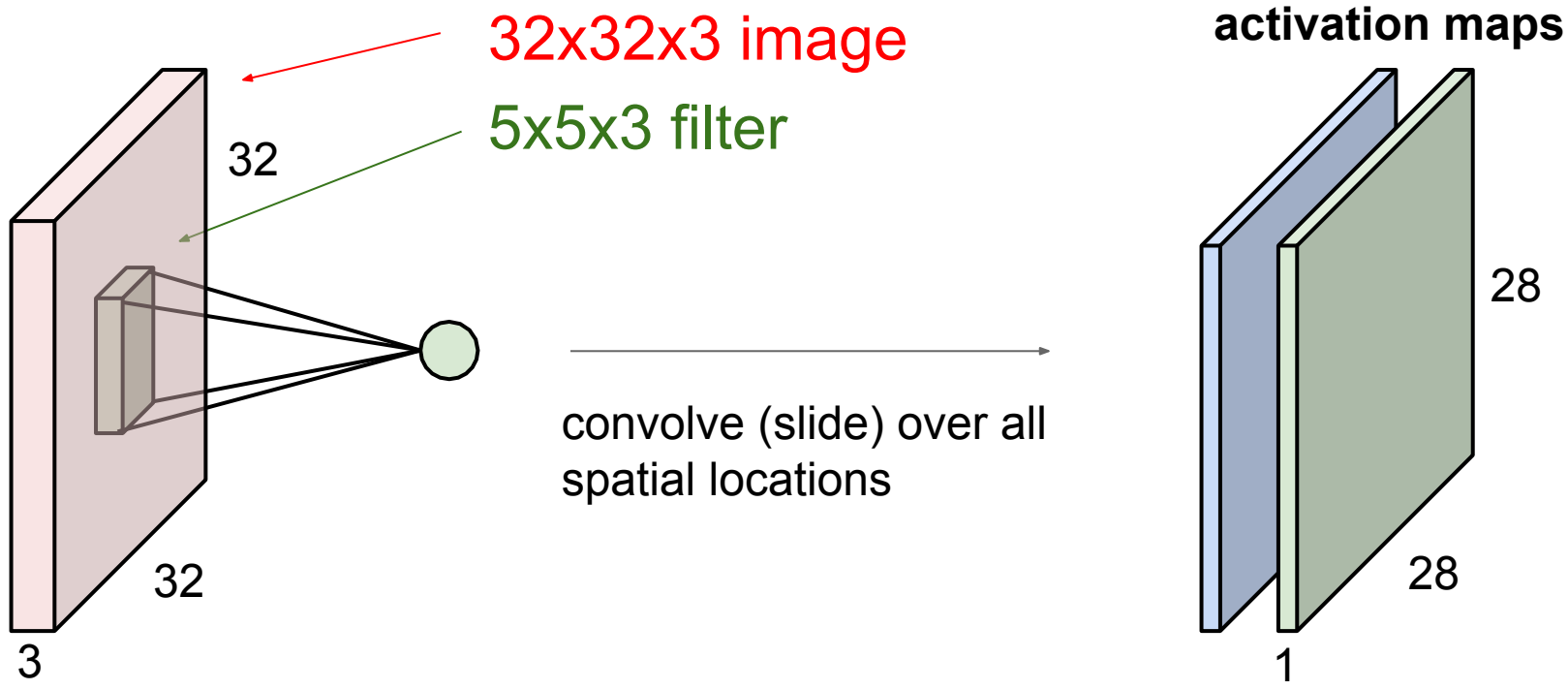
$$w^T x + b$$

Convolution Layer

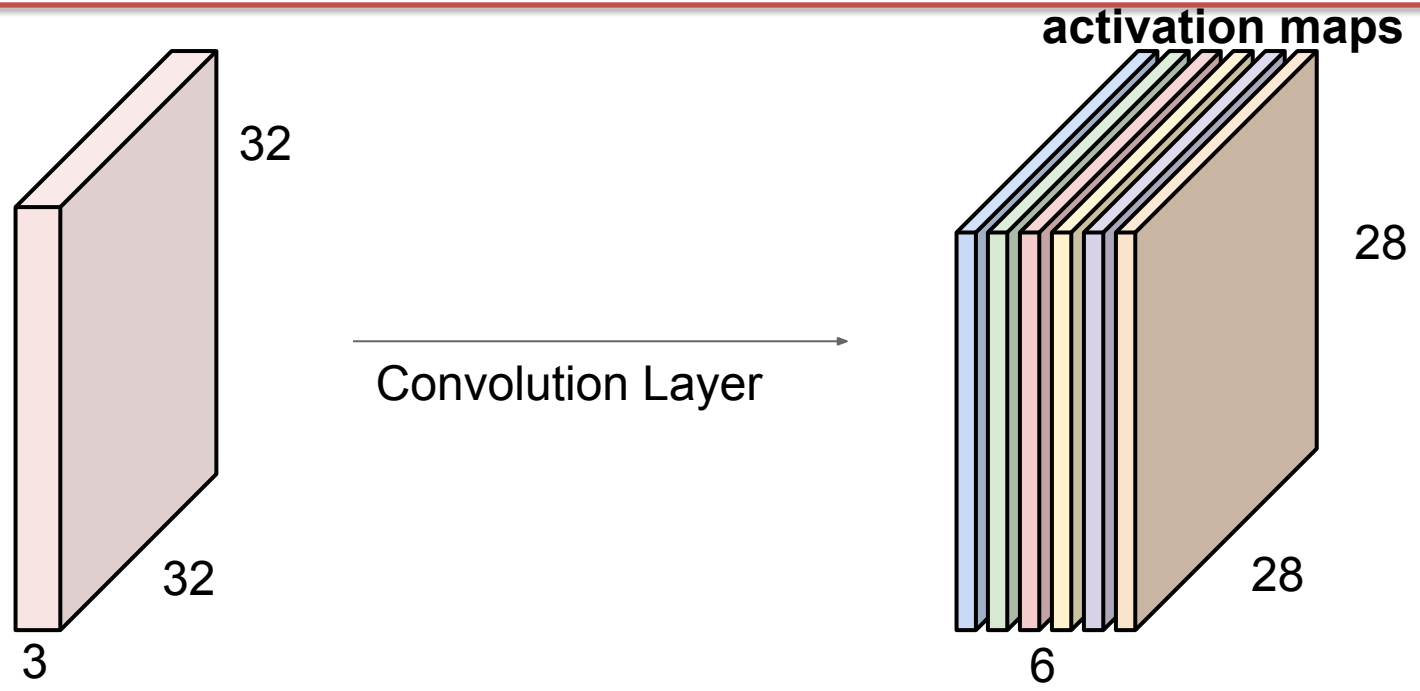


Convolution Layer

consider a second, green filter

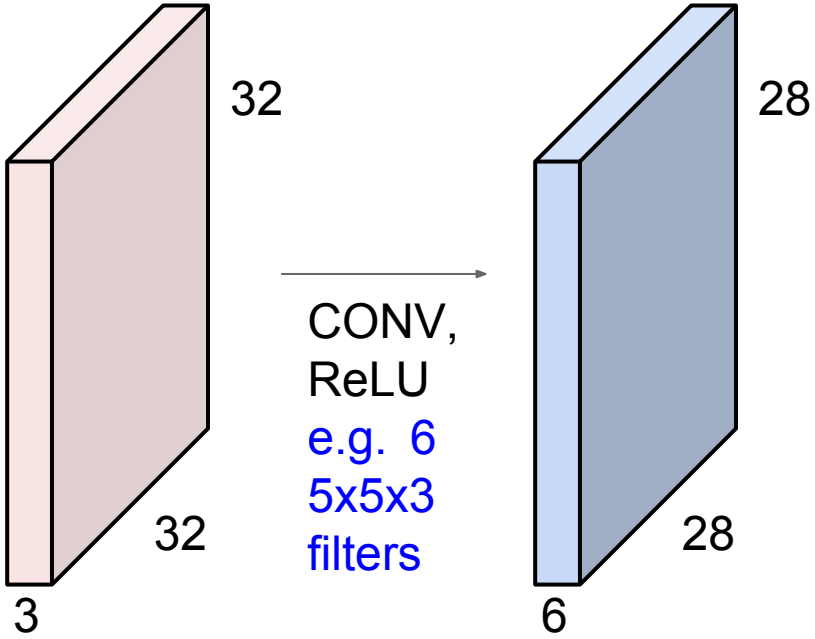


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

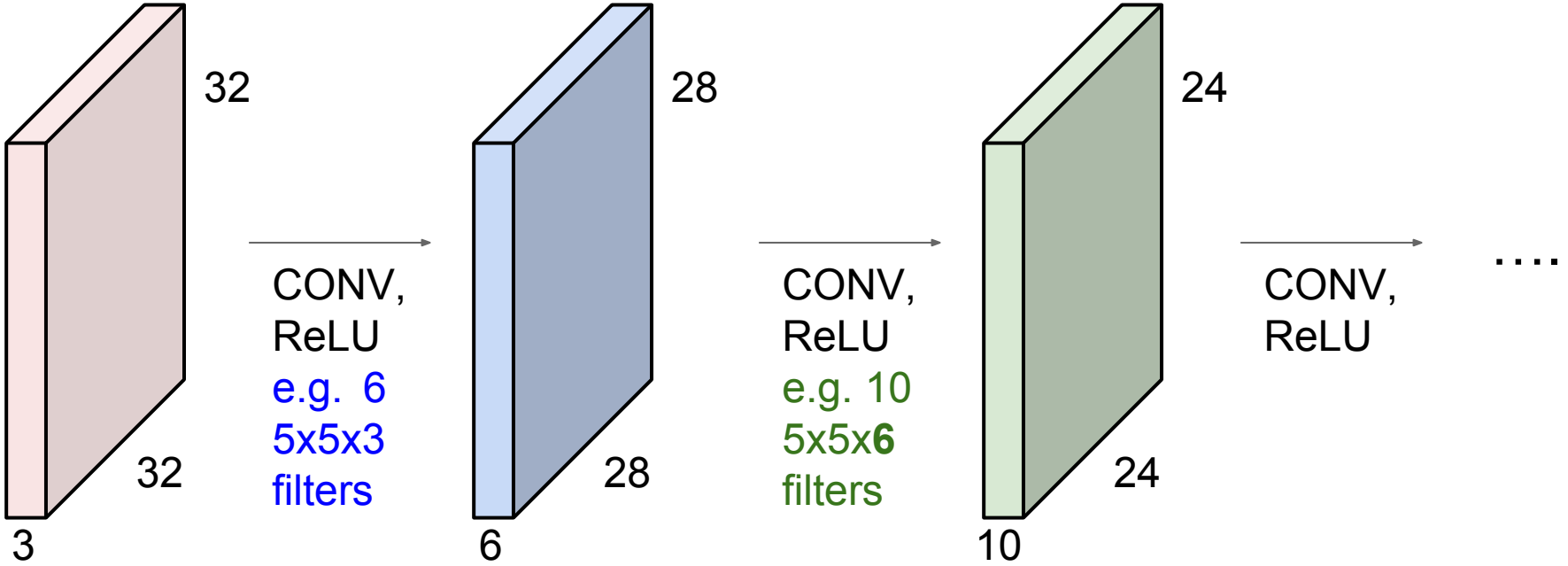


We stack these up to get a "new image" of size 28x28x6!

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



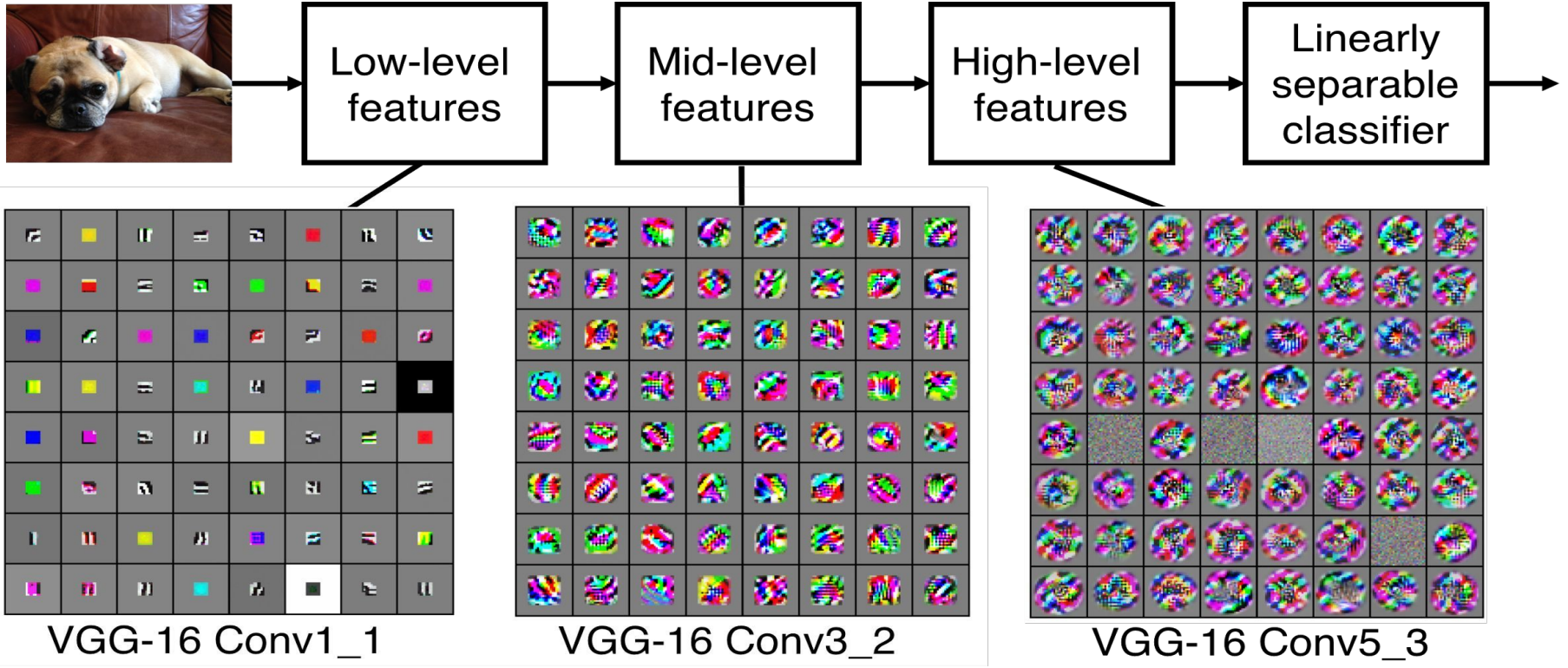
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].

Preview



one filter => one activation map

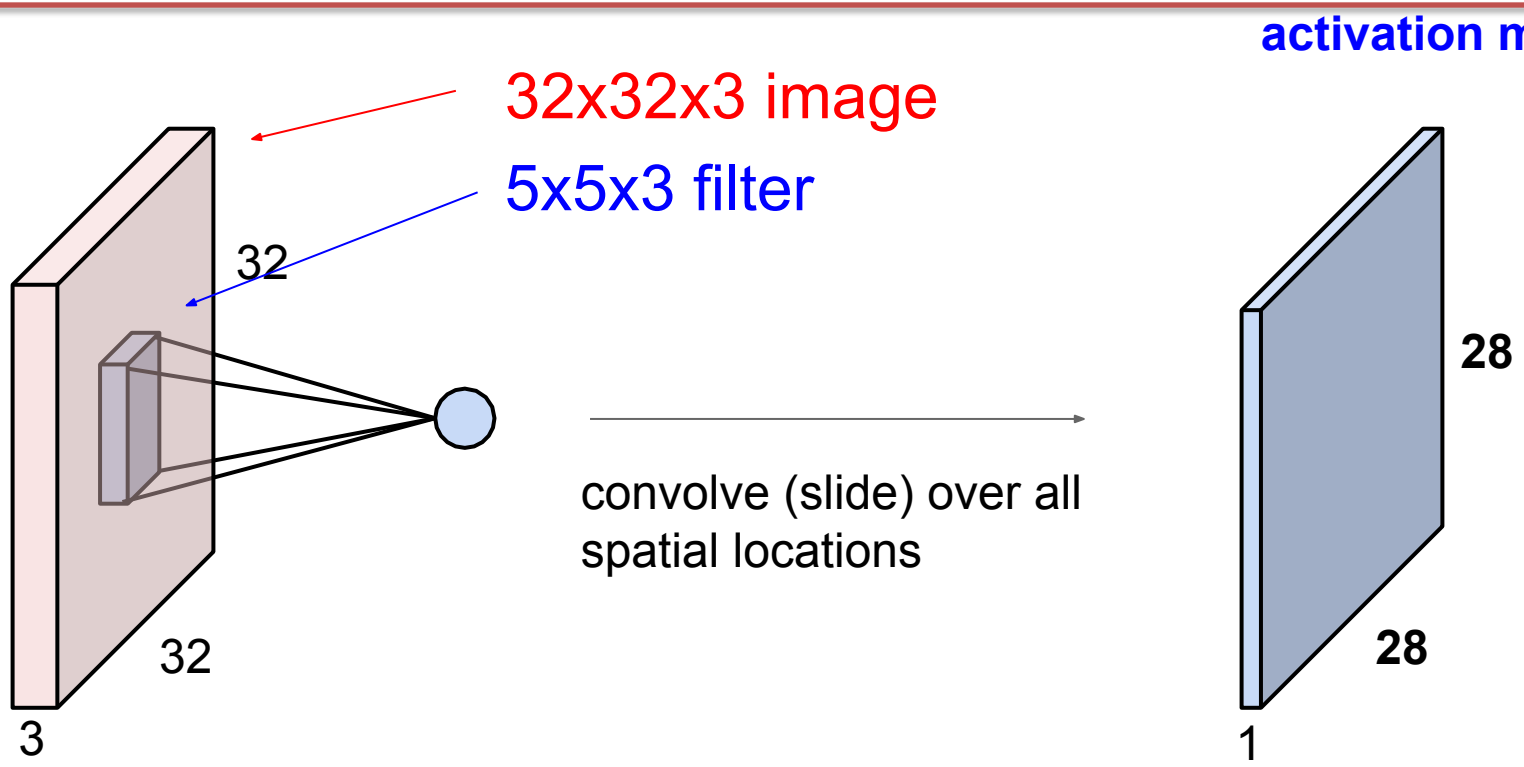
example 5x5 filters (32 total)

We call the layer convolutional because it is related to convolution of two signals:

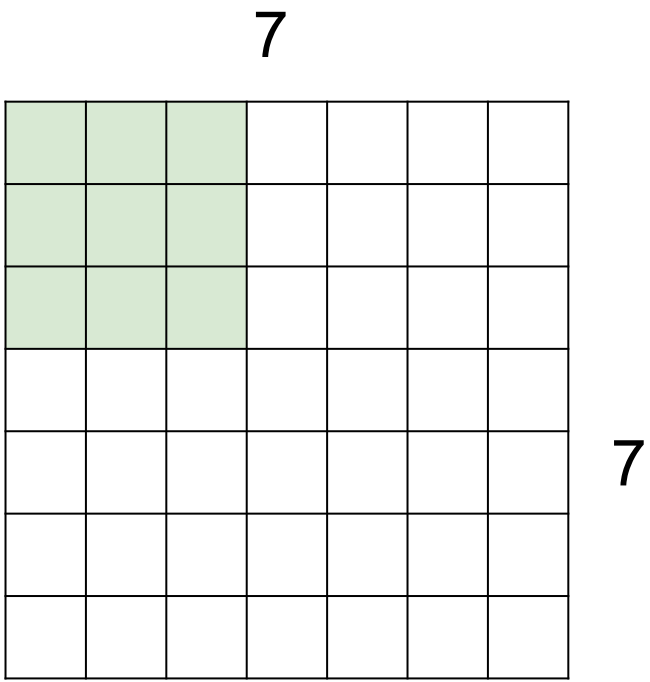
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

A closer look at spatial dimensions:



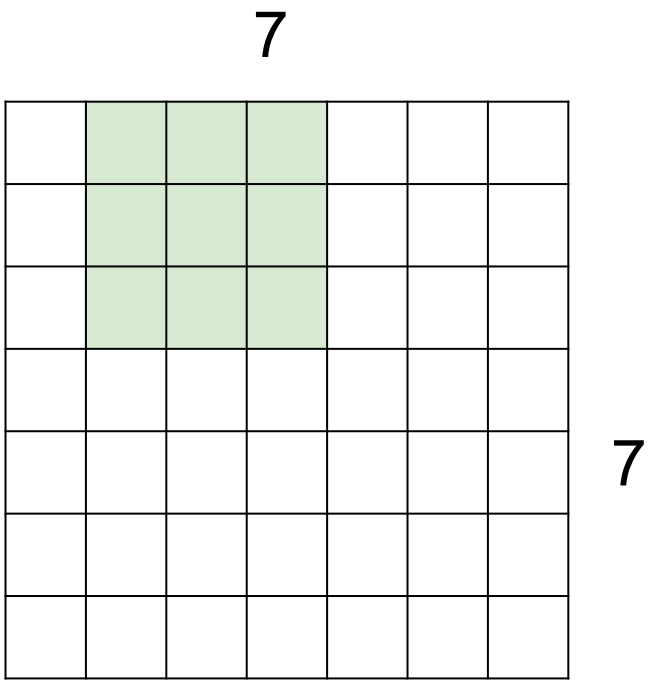
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

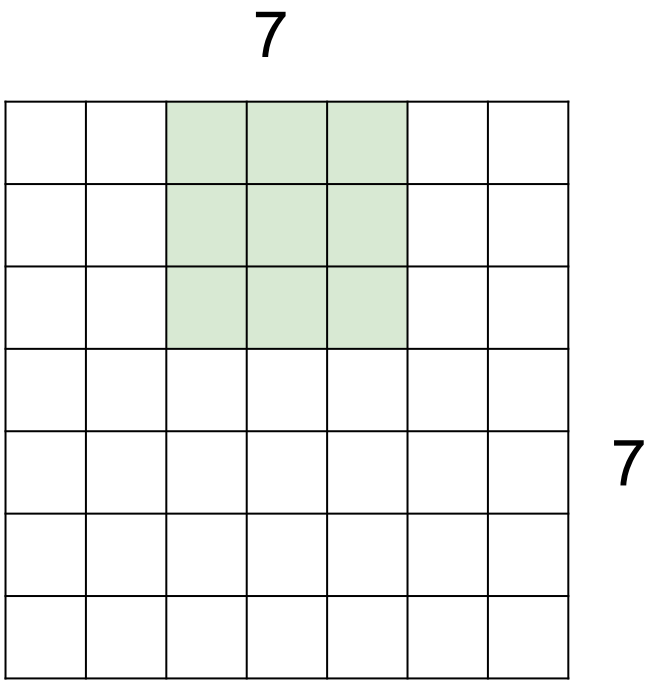
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

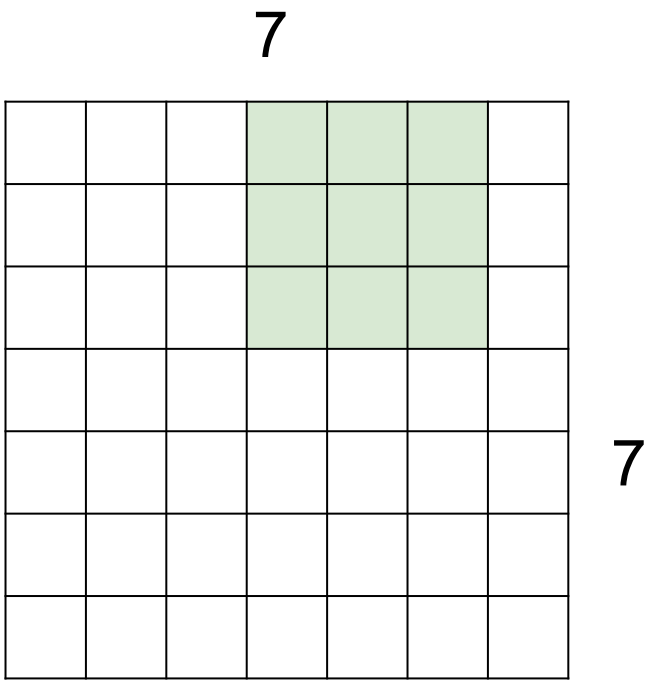
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

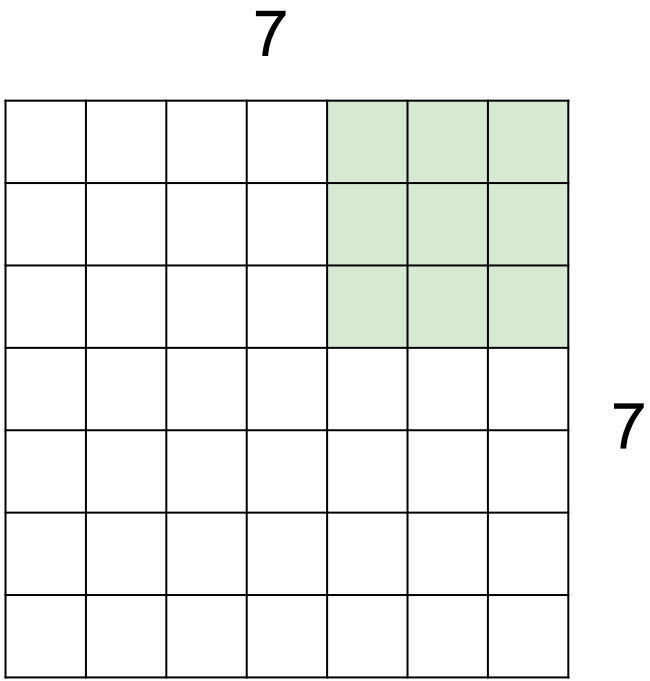
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

A closer look at spatial dimensions:

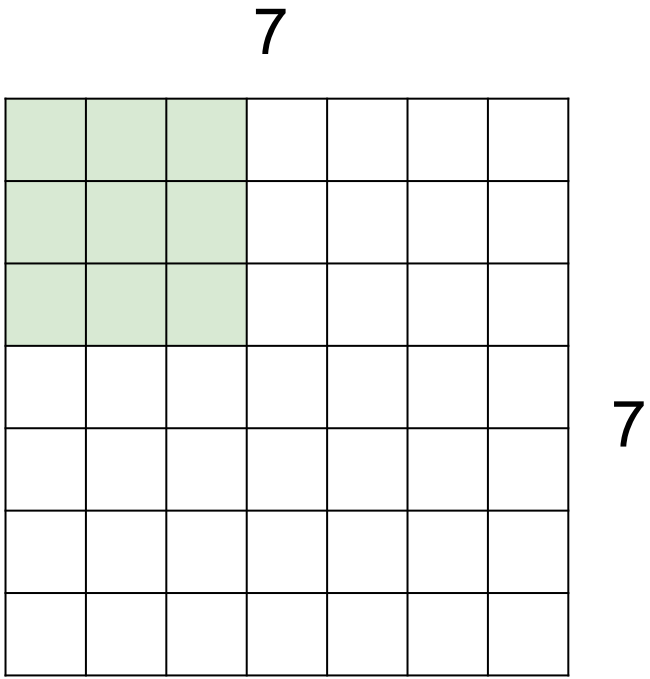


7x7 input (spatially)

assume 3x3 filter

=> 5x5 output

A closer look at spatial dimensions:

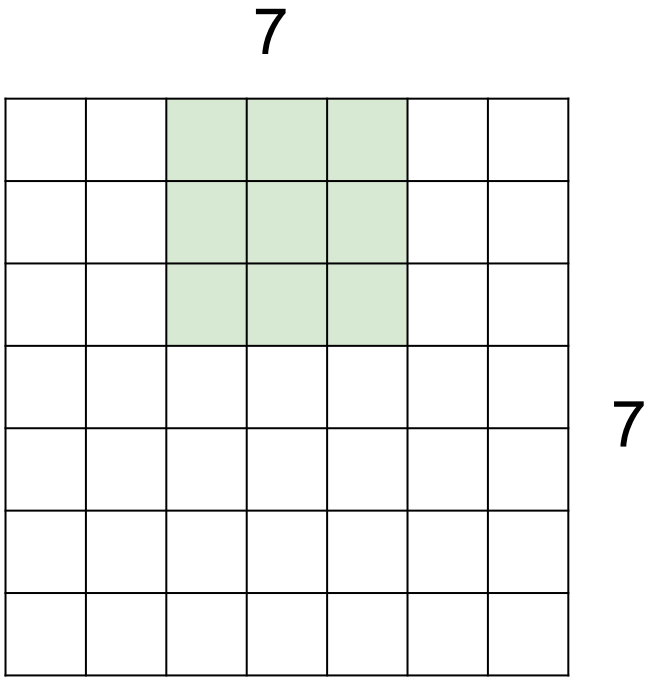


7x7 input (spatially)

assume 3x3 filter

applied **with stride 2**

A closer look at spatial dimensions:

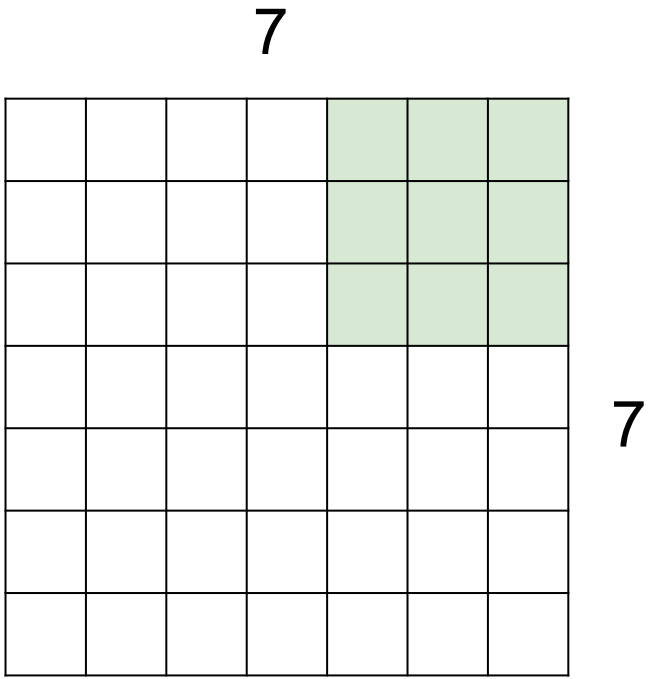


7x7 input (spatially)

assume 3x3 filter

applied **with stride 2**

A closer look at spatial dimensions:



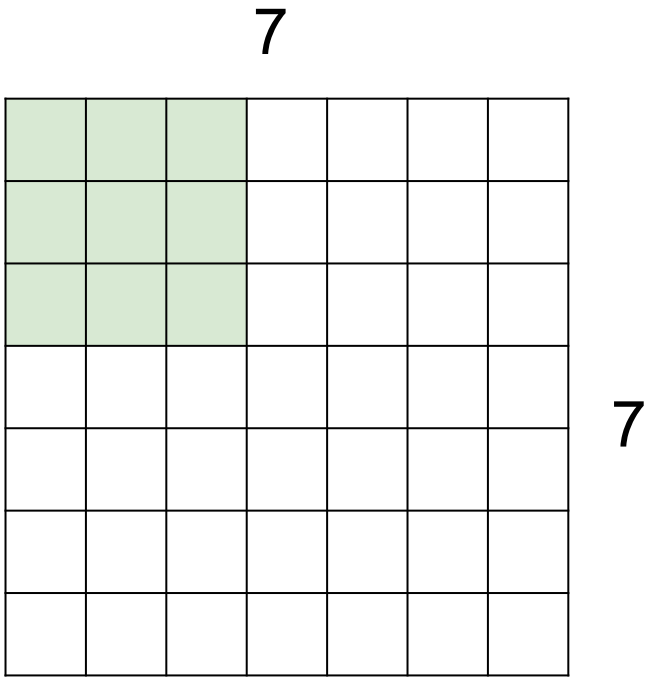
7x7 input (spatially)

assume 3x3 filter

applied **with stride 2**

=> 3x3 output!

A closer look at spatial dimensions:

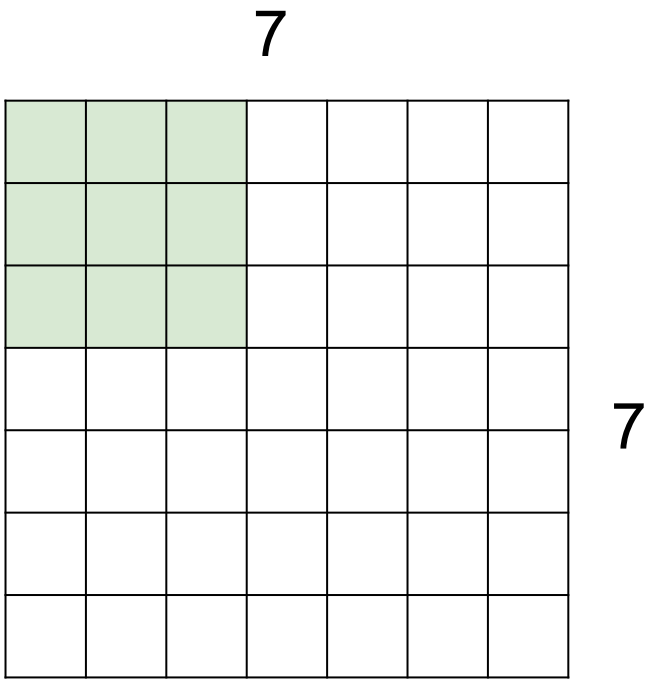


7x7 input (spatially)

assume 3x3 filter

applied **with stride 3?**

A closer look at spatial dimensions:



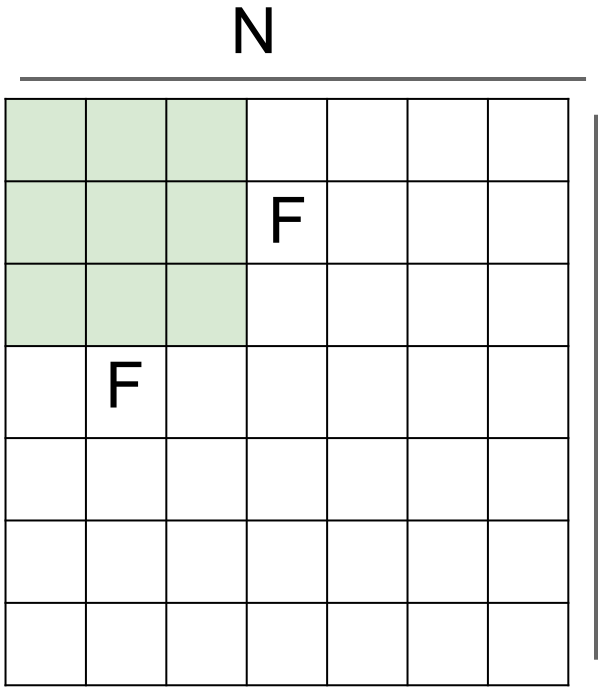
7x7 input (spatially)

assume 3x3 filter

applied **with stride 3?**

doesn't fit!

cannot apply 3x3 filter on 7x7 input with stride 3.



Output size:

(N - F) / stride + 1

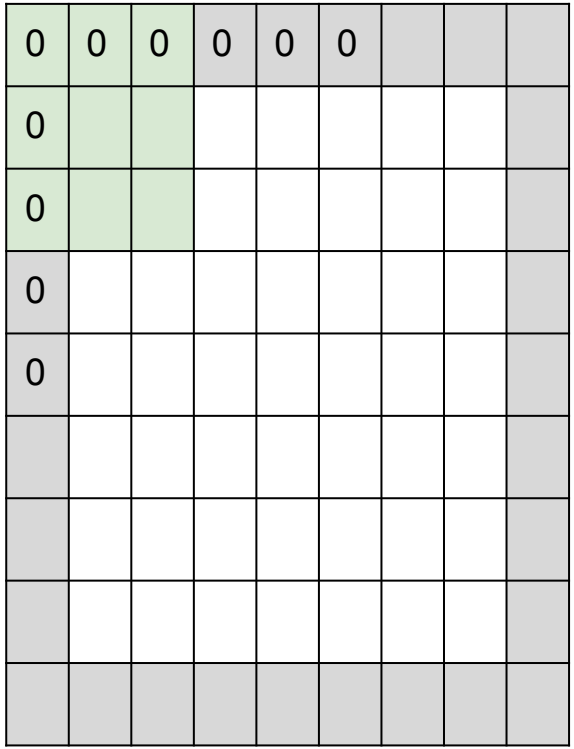
e.g. N = 7, F = 3:

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

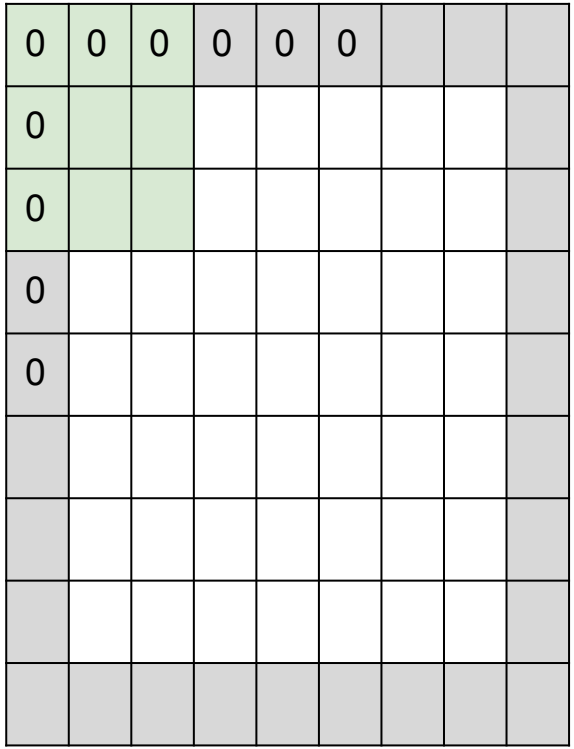


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)

(N - F) / stride + 1

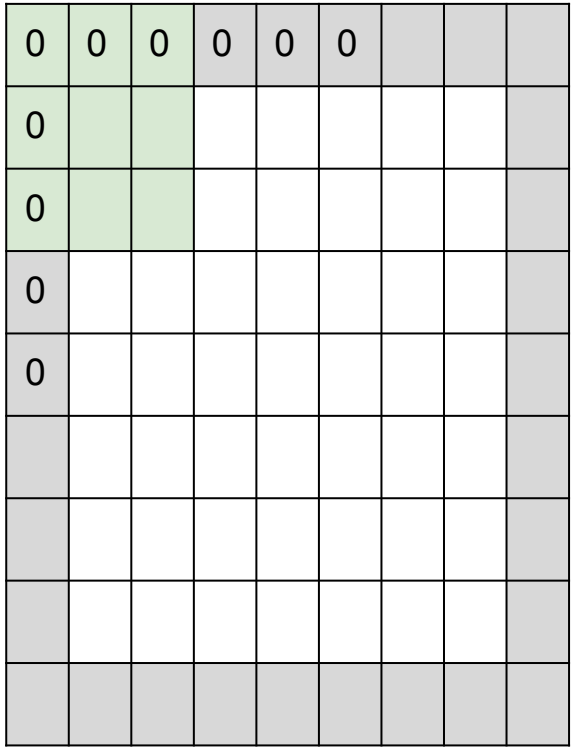
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

In practice: Common to zero pad the border

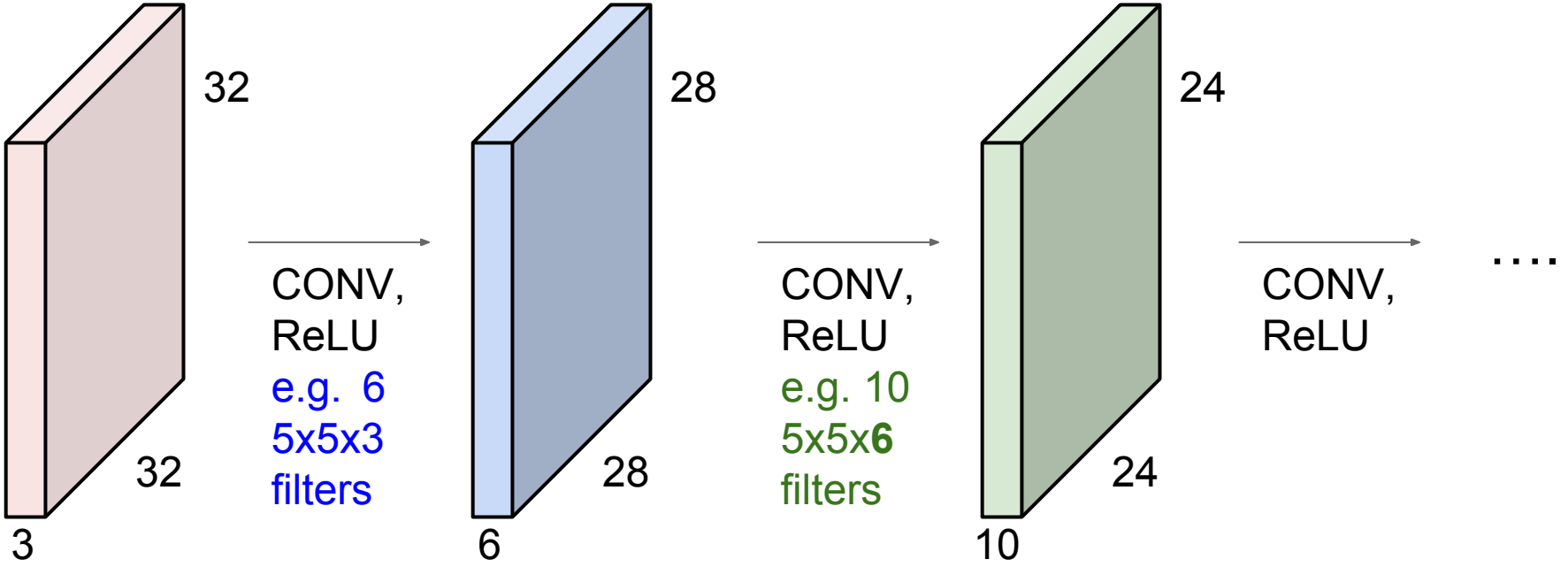


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, 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

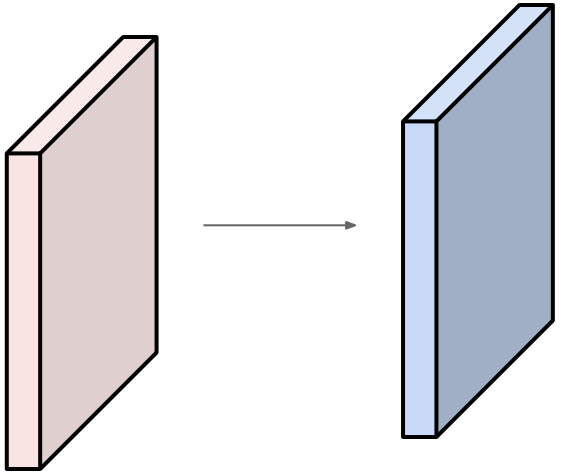
Remember back to…

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.

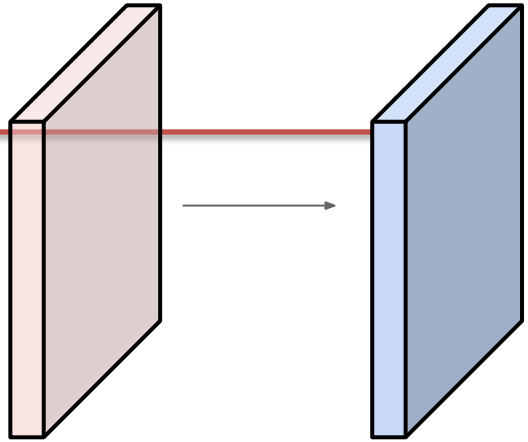


Examples time

- Input volume: **32x32x3**
- 10 5x5 filters with stride 1, pad 2
- Output volume size: ?



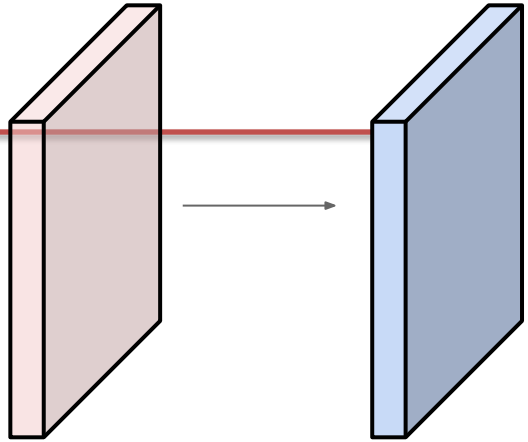
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

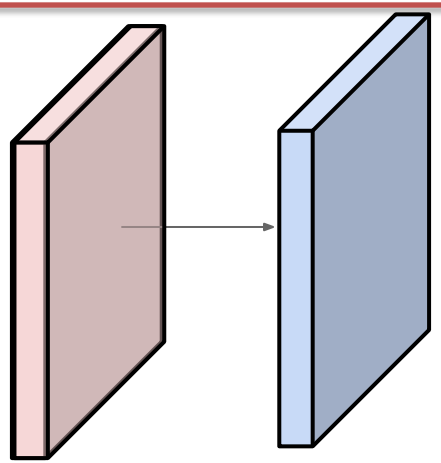
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

To summarize, the Conv Layer

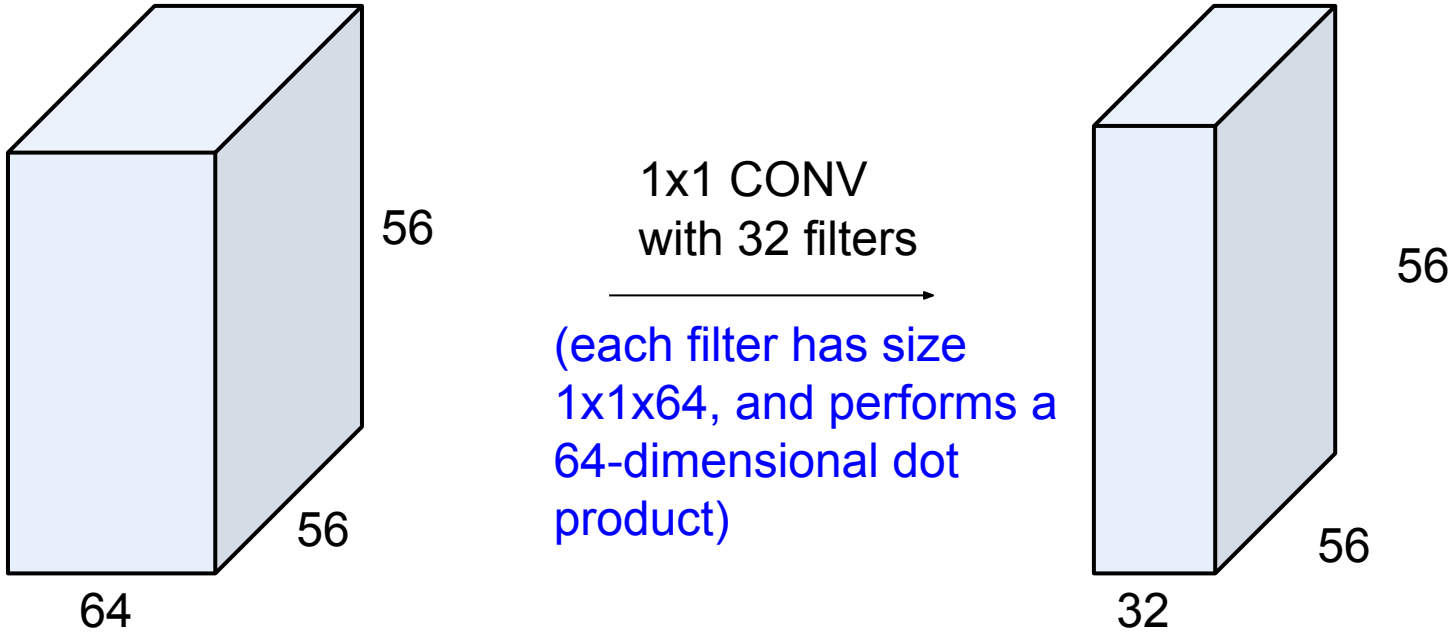
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - ✔ Number of filters K,
  - ✔ their spatial extent F,
  - ✔ the stride S,
  - ✔ the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - ✔ $W_2 = (W_1 - F + 2P)/S + 1$
  - ✔ $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - ✔ $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Common settings:

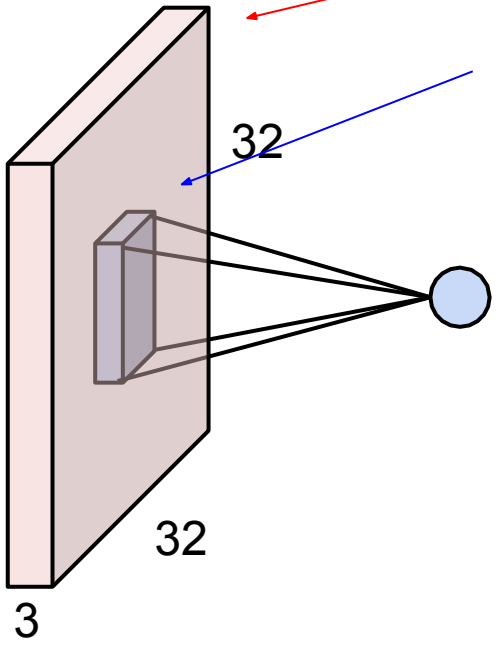
K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

(btw, 1x1 convolution layers make perfect sense)



The brain/neuron view of CONV Layer



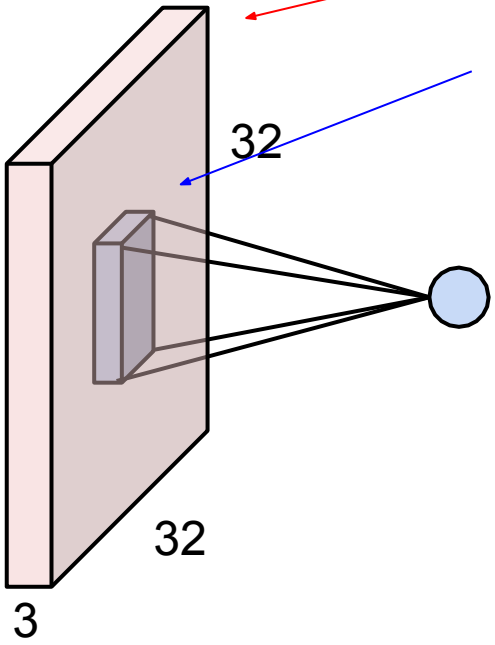
32x32x3 image

5x5x3 filter

1 number:

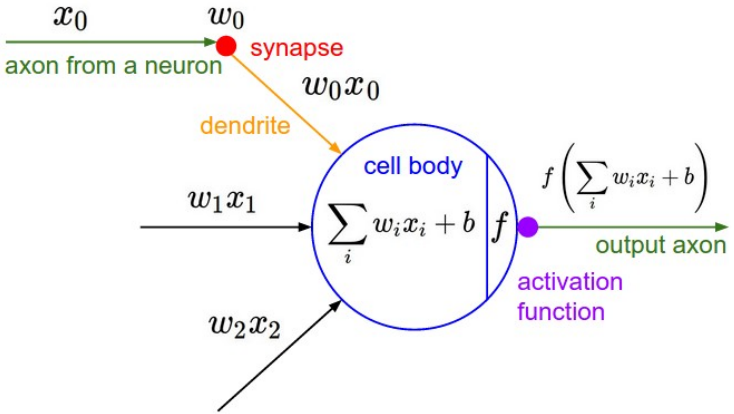
the result of taking a dot product between the filter and this part of the image (i.e. 5*5*3 = 75-dimensional dot product)

The brain/neuron view of CONV Layer



32x32x3 image

5x5x3 filter

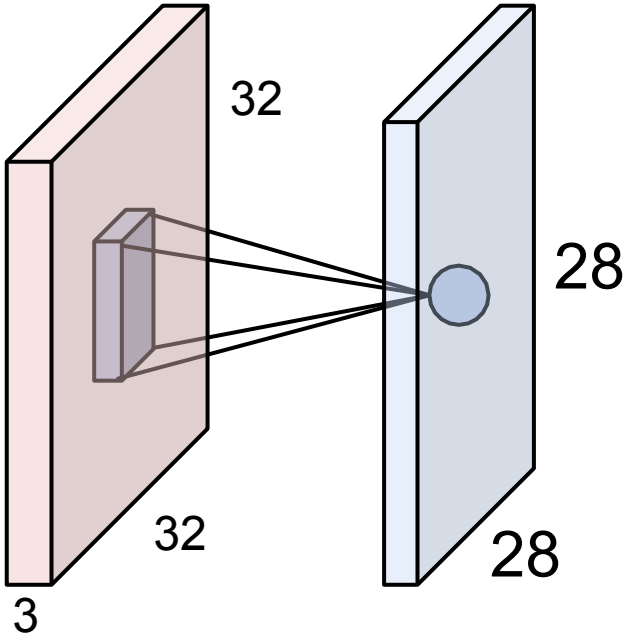


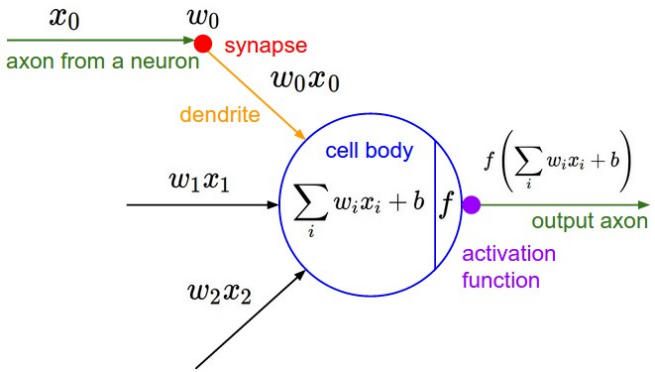
It's just a neuron with local connectivity...

1 number:

the result of taking a dot product between the filter and this part of the image (i.e. 5*5*3 = 75-dimensional dot product)

The brain/neuron view of CONV Layer



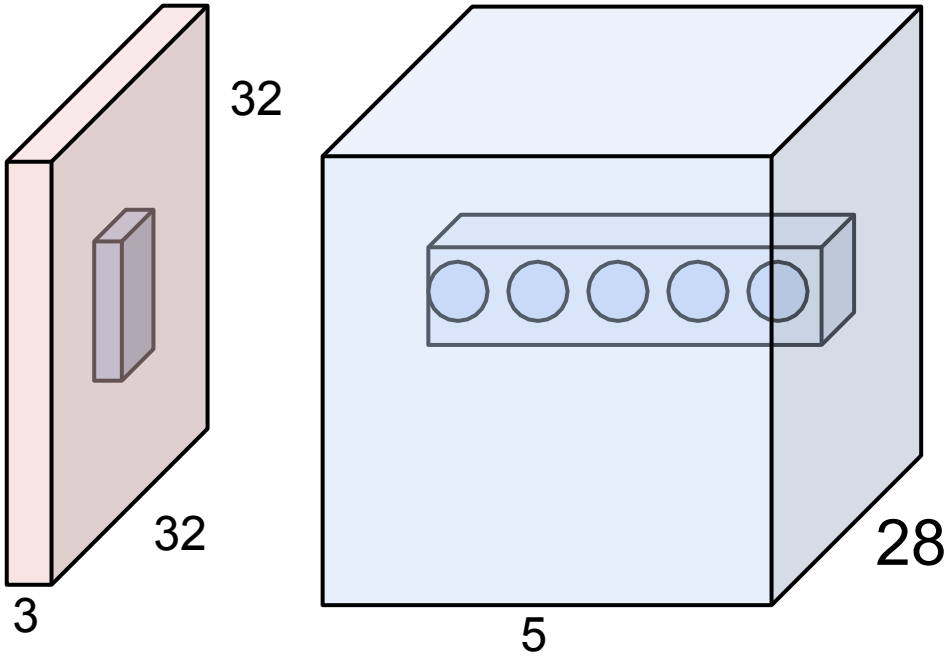


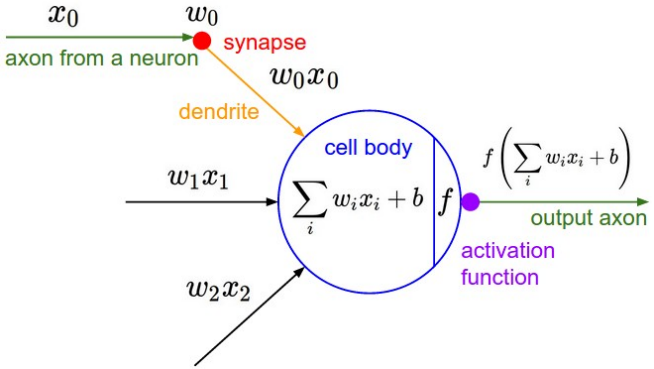
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

The Brain/Neuron view of CONV Layer





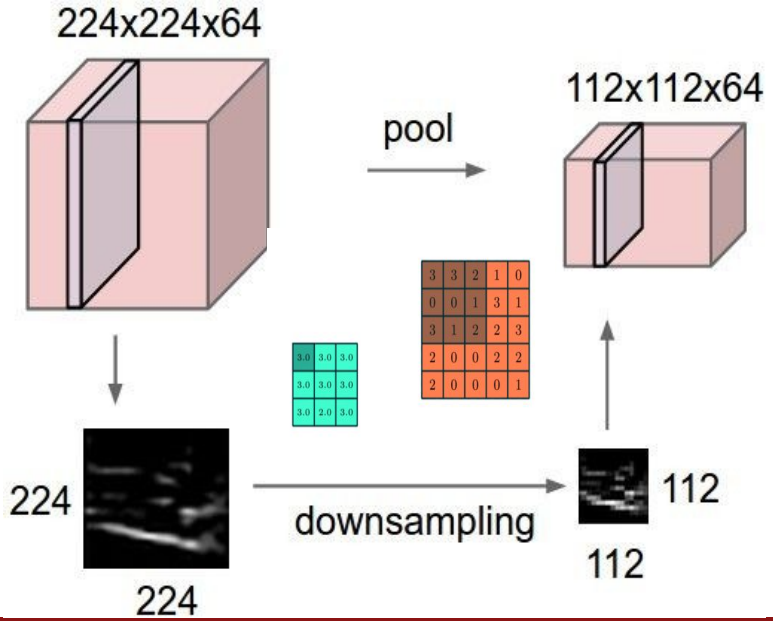
E.g. with 5 filters, CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume.

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently.
- Pooling layer is responsible for reducing the spatial size of the Convolved Feature.
- This is to **decrease the computational power required to process the data** through dimensionality reduction.
- Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

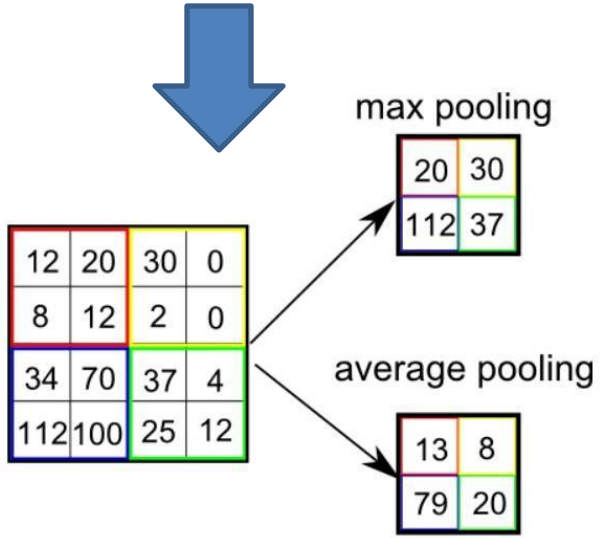
- **Pooling is two types: Max and Average**



MAX and AVERAGE POOLING

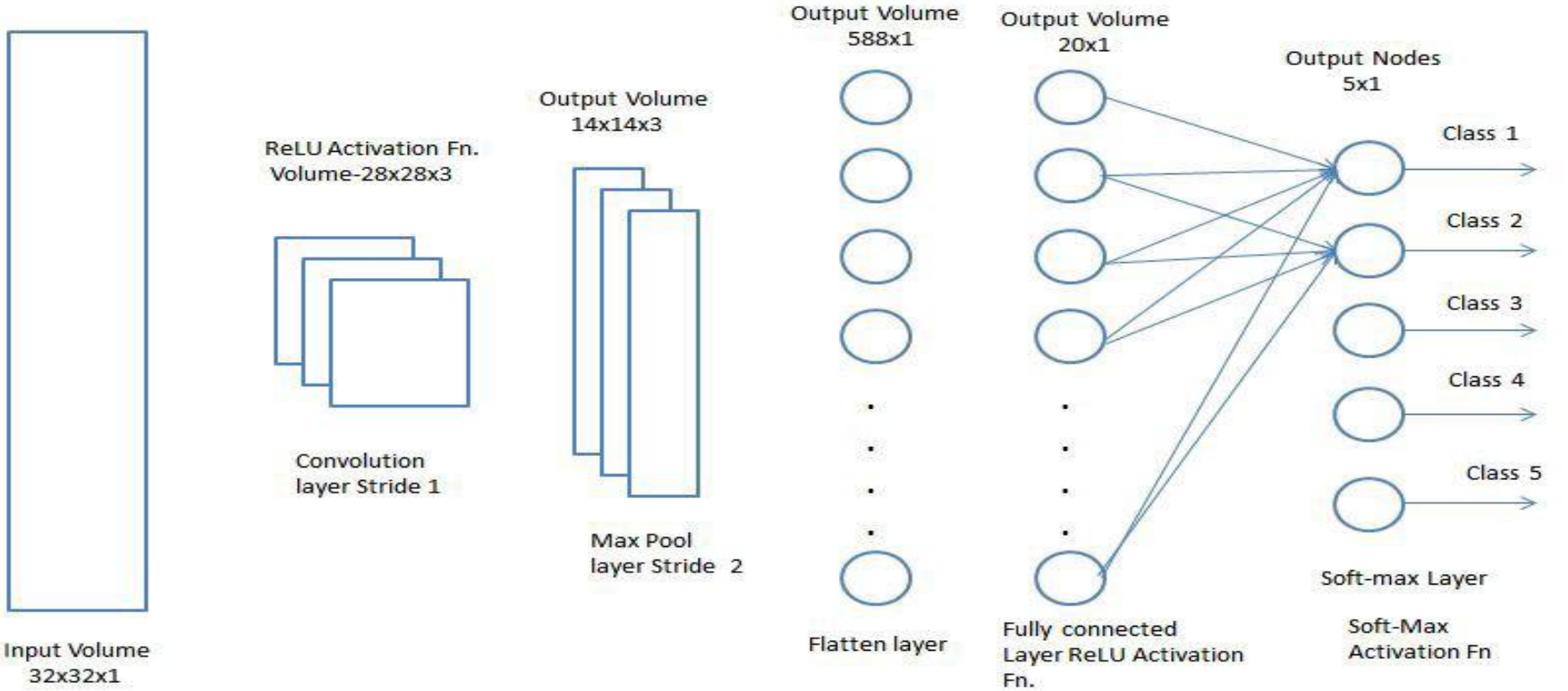
- **Max Pooling** returns the **maximum value** from the portion of the image covered by the Kernel.
- **Average Pooling** returns the **average of all the values** from the portion of the image covered by the Kernel.
- **Max Pooling** also performs as a **Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.
- On the other hand, **Average Pooling** simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.
- **The Convo Layer and the Pooling Layer, together form the i-th layer of a CNN. These layers may be increased to have low level details but computational complexity increases.**

Max and Average pool with 2x2 filters and stride 2



Fully Connected Layer

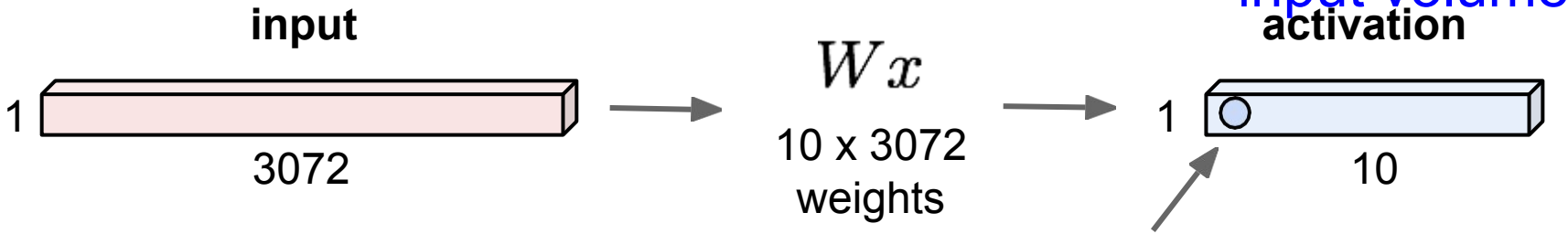
Flatten the final output and feed it to a regular NN for classification purposes



Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

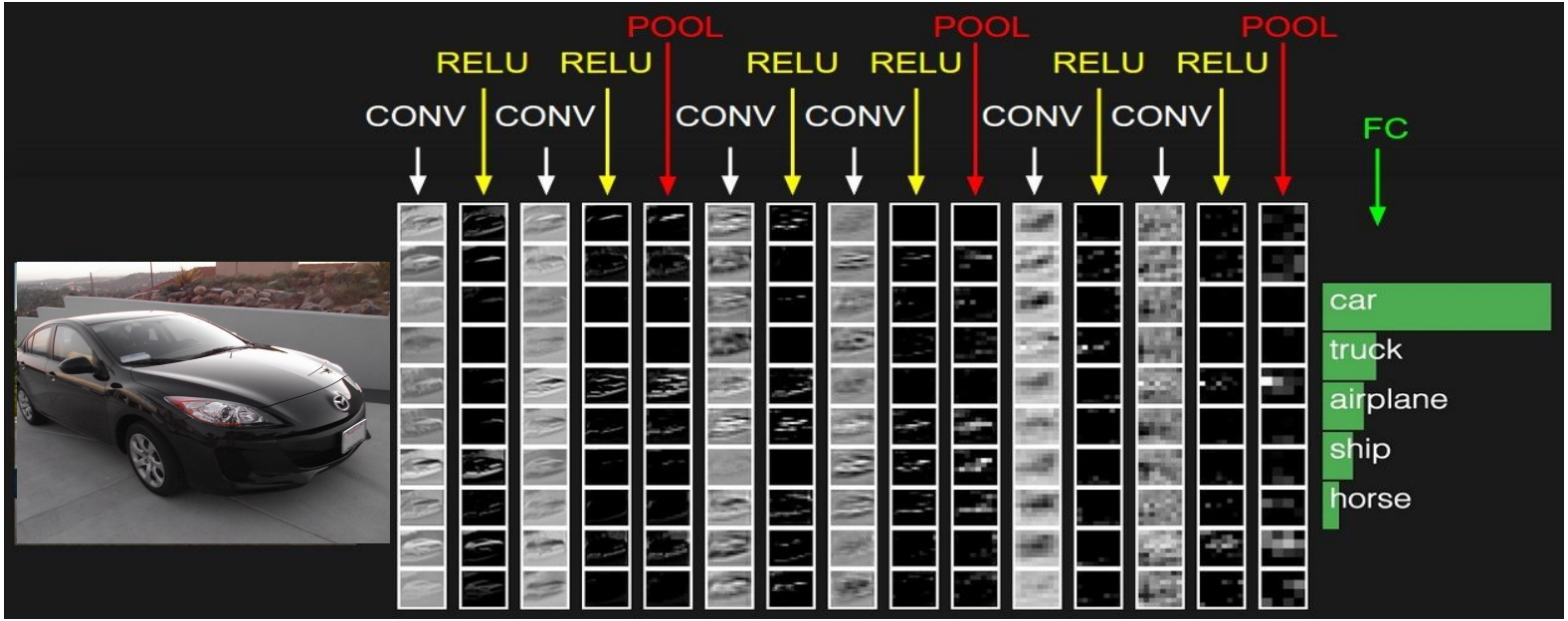
Each neuron looks at the full input volume



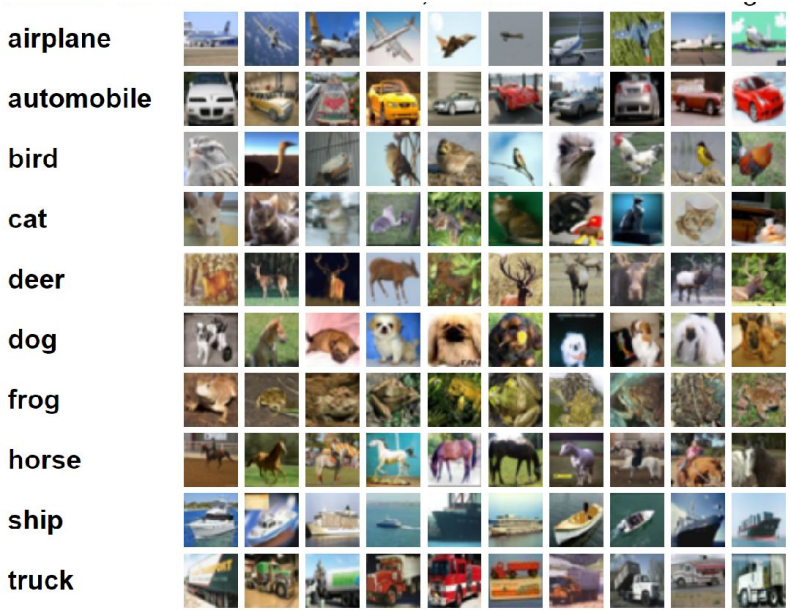
1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

Fully Connected Layer (FC layer)

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



[ConvNetJS demo: training on CIFAR-10]



http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Summary

- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like

  [(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX

  where N is usually up to ~5, M is large, 0 <= K <= 2.
  - but recent advances such as ResNet/GoogLeNet challenge this paradigm

Reference

- Fei-Fei Li & Justin Johnson & Serena Yeung, Lecture Series
- https://towardsdatascience.com/