CS489/698: Intro to ML

Lecture 12: Training Neural Networks and CNNs part 1

focal Systems

UNIVERSITY OF WATERLOO

Outline

- Training
- Convolutional Layers

Outline

- Training
- Convolutional Layers

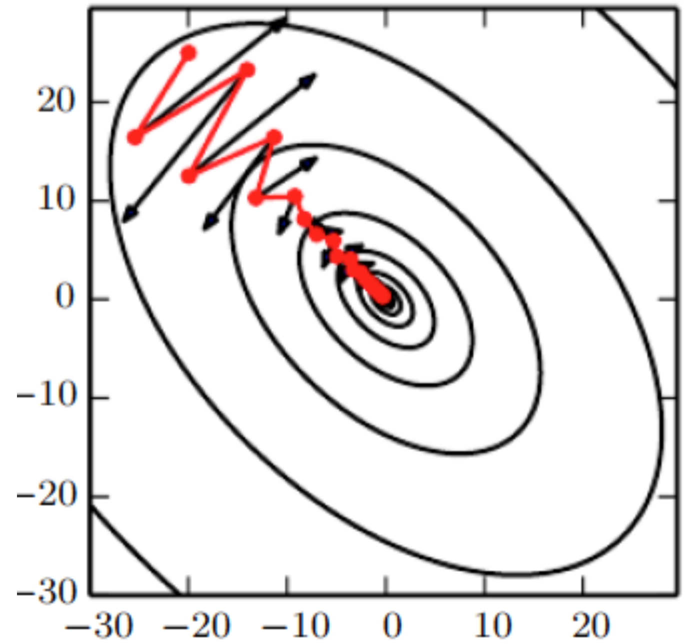
SGD Formalized

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.
Require: Initial parameter $\boldsymbol{\theta}$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\hat{\boldsymbol{g}}$
  end while

Momentum

- Adds a velocity term
- $$v \leftarrow \alpha v - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right),$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}.$$
- Adds a speedup of at most $\frac{\epsilon ||\boldsymbol{g}||}{1-\alpha}$.



- Momentum constant usually 0.9, 0.5, 0.99 corresponding to a 10x, 2x, 100x increase in max speed

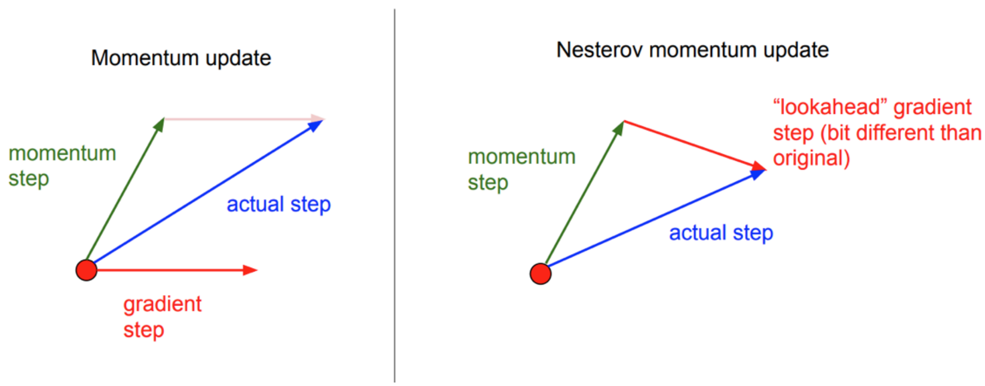
focal Systems UNIVERSITY OF WATERLOO

Nesterov Momentum

- Applies momentum first then gradient

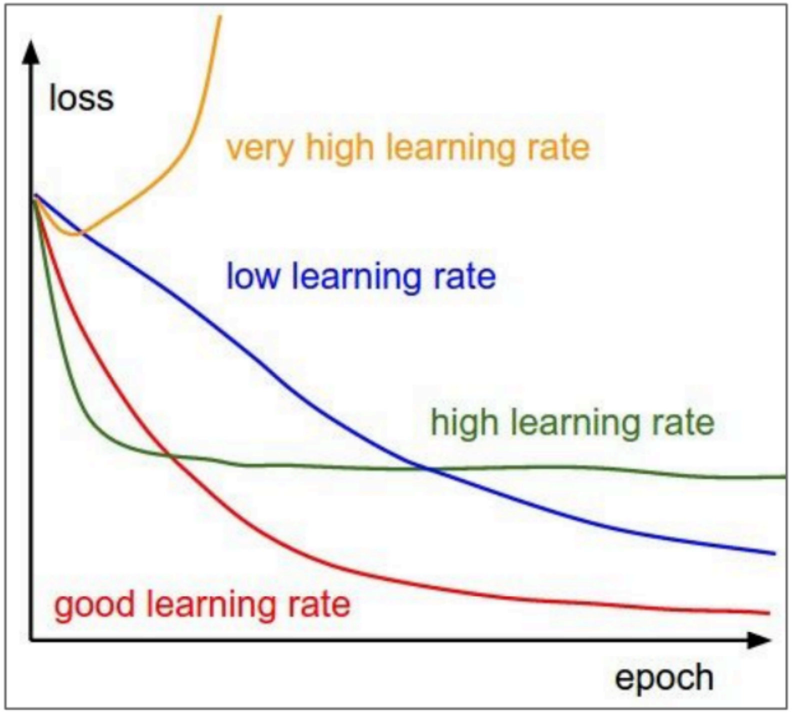
$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right),$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}.$$



focal Systems UNIVERSITY OF WATERLOO

Setting the learning rate



=> Learning rate decay over time!

step decay:
e.g. decay learning rate by half every few epochs.

exponential decay:

$$\alpha = \alpha_0 e^{-kt}$$

1/t decay:

$$\alpha = \alpha_0 / (1 + kt)$$

- Also good to try a setting for 100 iterations and see which is best on validation set

Adagrad

- Adapting the learning rate per parameter
- If gradient is always high, decreases learning rate
- If gradient is low, increases learning rate
- meant for fast convergence on convex problems
- suffers from premature decay

Adagrad

Algorithm 8.4 The AdaGrad algorithm

Require: Global learning rate ϵ

Require: Initial parameter $\boldsymbol{\theta}$

Require: Small constant δ, perhaps 10^{-7}, for numerical stability

Initialize gradient accumulation variable $\boldsymbol{r} = \mathbf{0}$

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$

Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. (Division and square root applied element-wise)

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

end while

RMSProp

- Replaces average with exponential decay
- Prevents premature decay

Algorithm 8.6 RMSProp algorithm with Nesterov momentum

Require: Global learning rate ϵ, decay rate ρ, momentum coefficient α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  Initialize accumulation variable $\boldsymbol{r} = \mathbf{0}$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
    Accumulate gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \frac{\epsilon}{\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while

Adam - Adaptive Momentum

- Applies momentum with parameterized learning rates

Algorithm 8.7 The Adam algorithm

Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$

Initialize 1st and 2nd moment variables $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
Initialize time step $t = 0$
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
  Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
  $t \leftarrow t + 1$
  Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
  Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
  Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$
  Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$
  Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}}+\delta}$ (operations applied element-wise)
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

How to choose?

- SGD < SGD+Momentum < SGD+Nesterov Momentum
- Adam is a good default
- RMSProp is good for RNNs, but also good default
- SGD + Nesterov momentum is best if you have time/ resources to optimize learning rate
- More of an Art

focal Systems

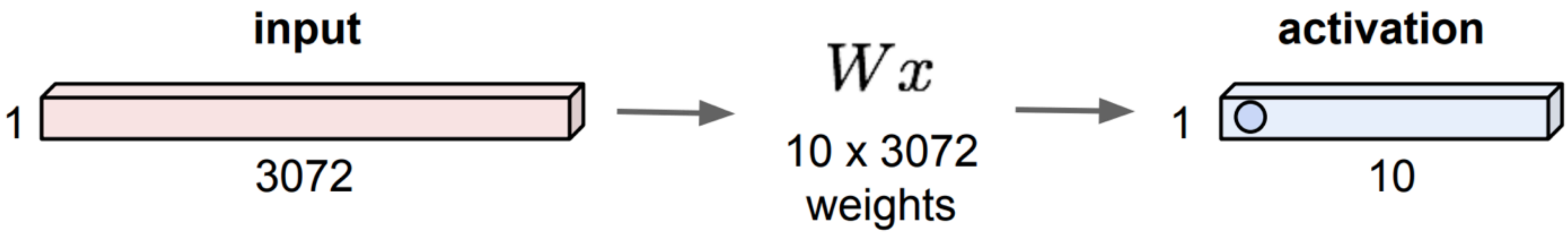
UNIVERSITY OF WATERLOO

Outline

- Training
- Convolutional Layers

Fully Connected Layer

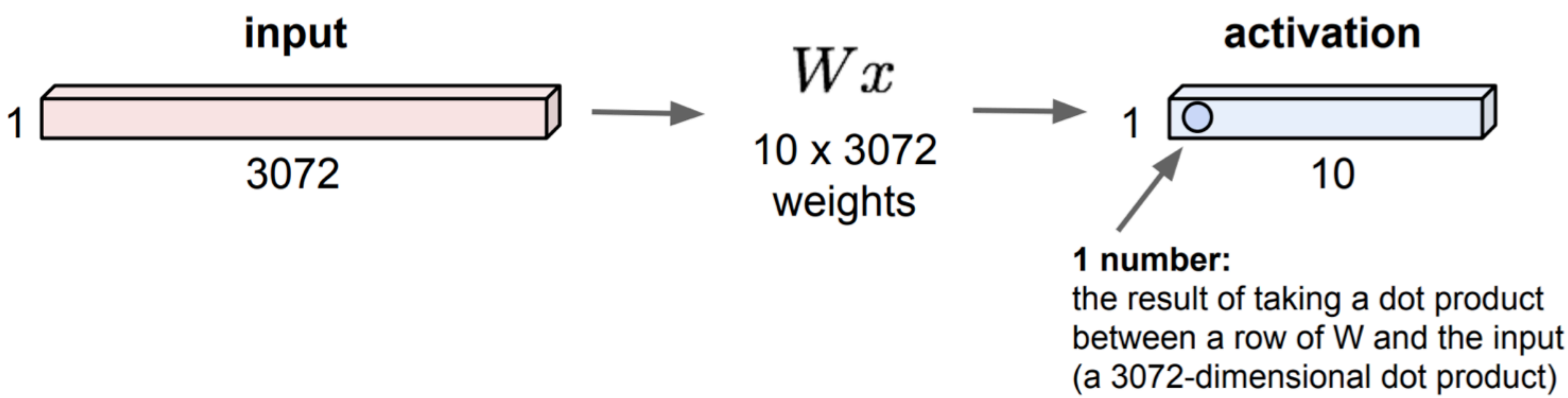
32x32x3 image -> stretch to 3072 x 1



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

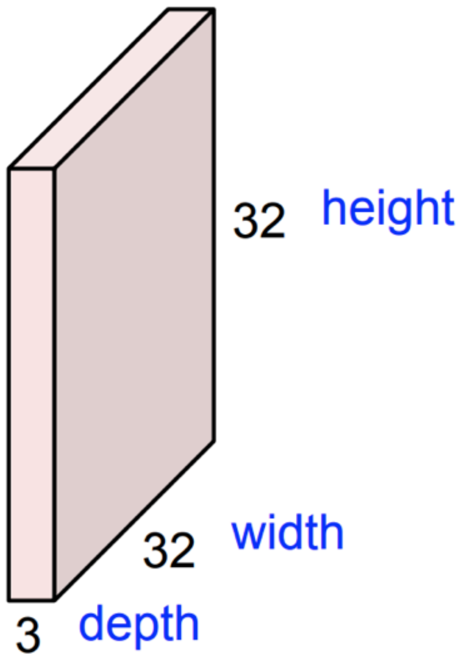


http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

focal Systems UNIVERSITY OF WATERLOO

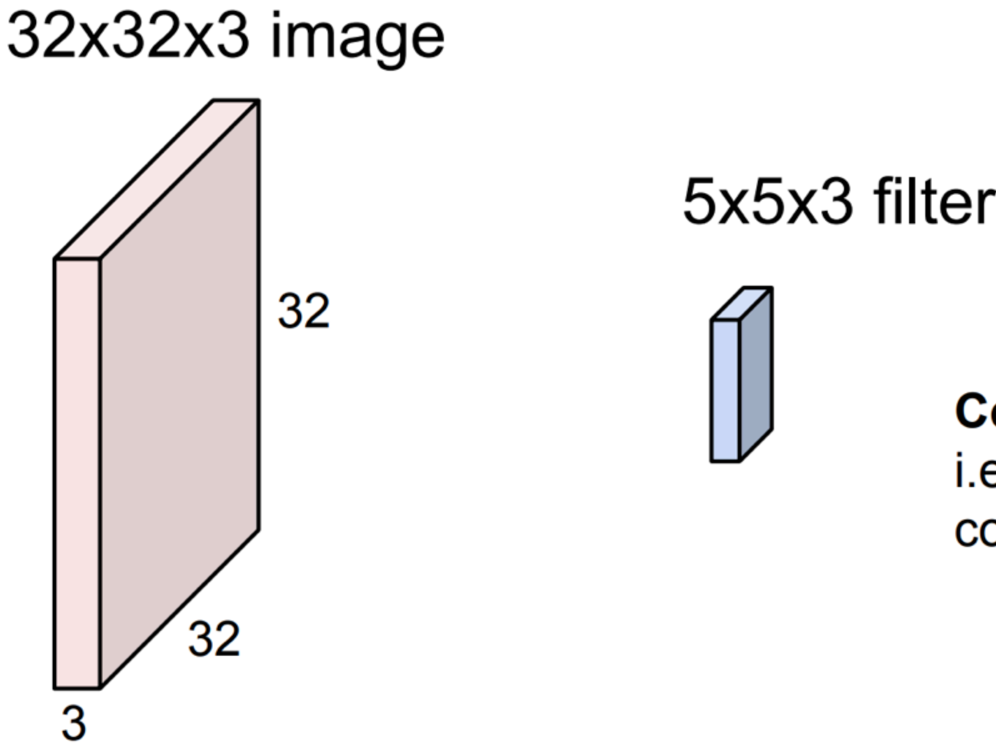
Convolutional Layer

32x32x3 image -> preserve spatial structure



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer



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

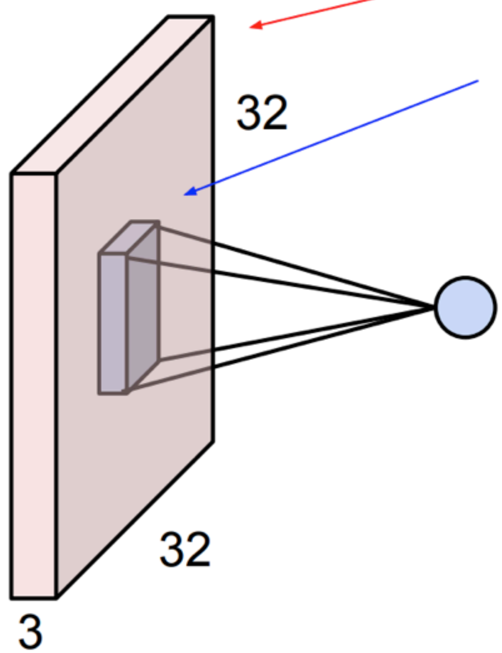
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

Convolution Layer

32x32x3 image

5x5x3 filter w



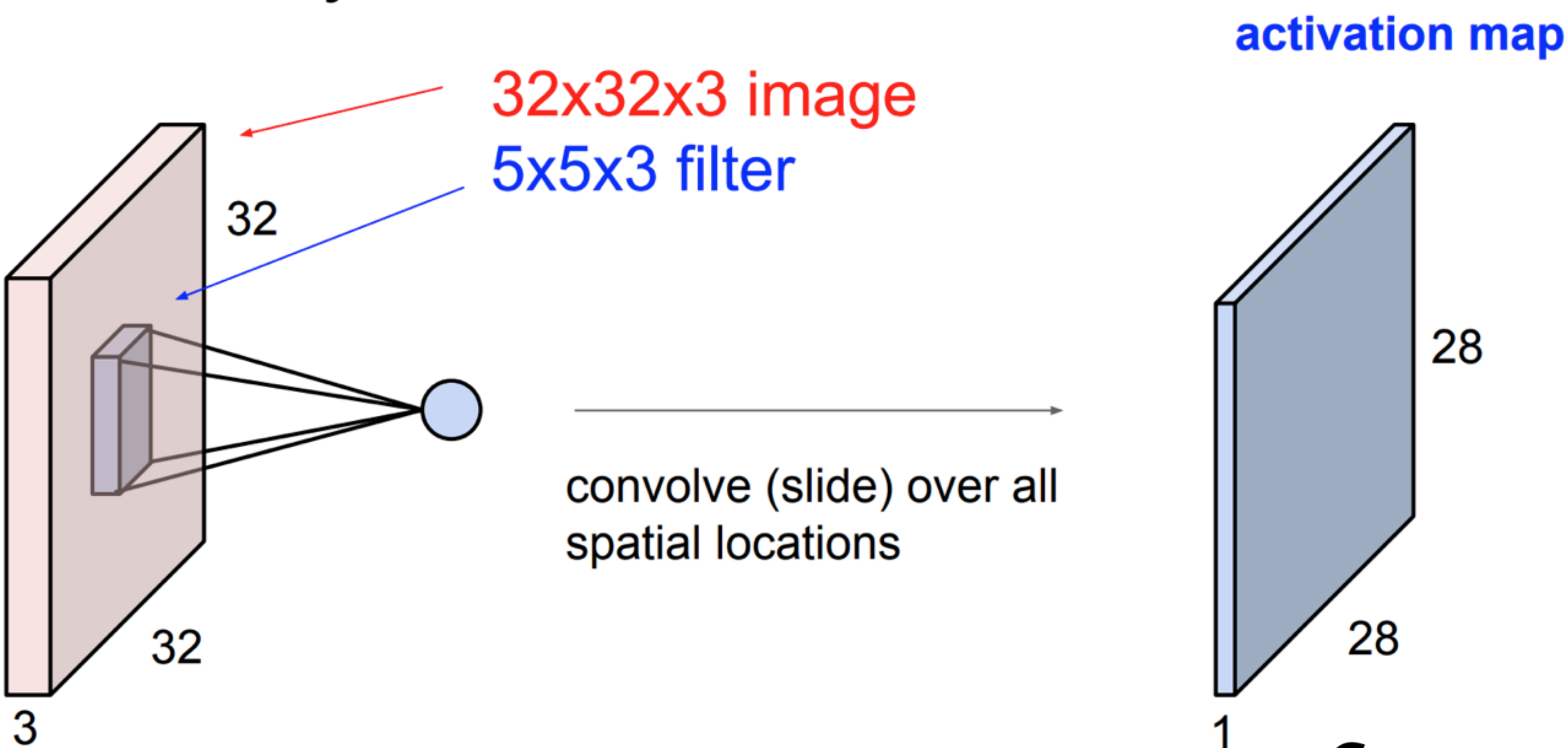
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

Convolution Layer

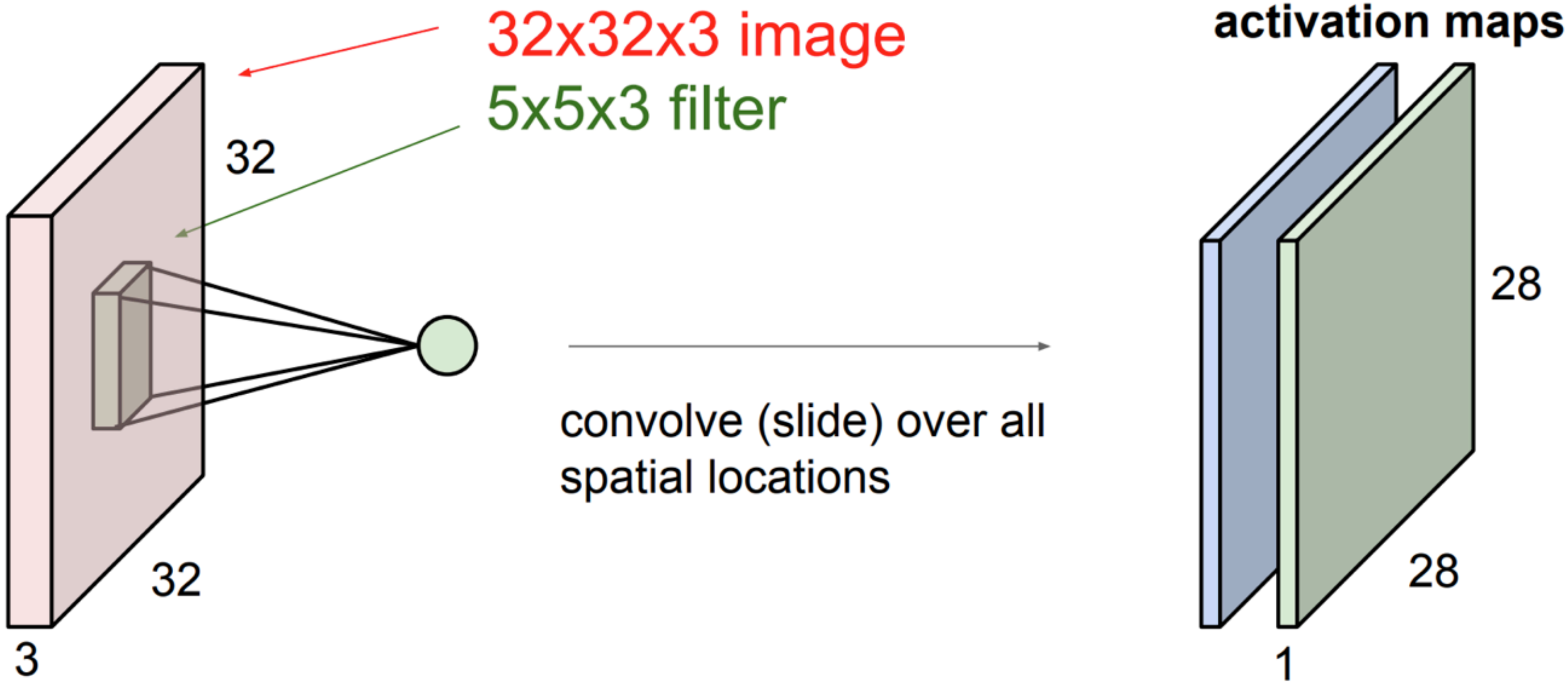


http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

Convolution Layer

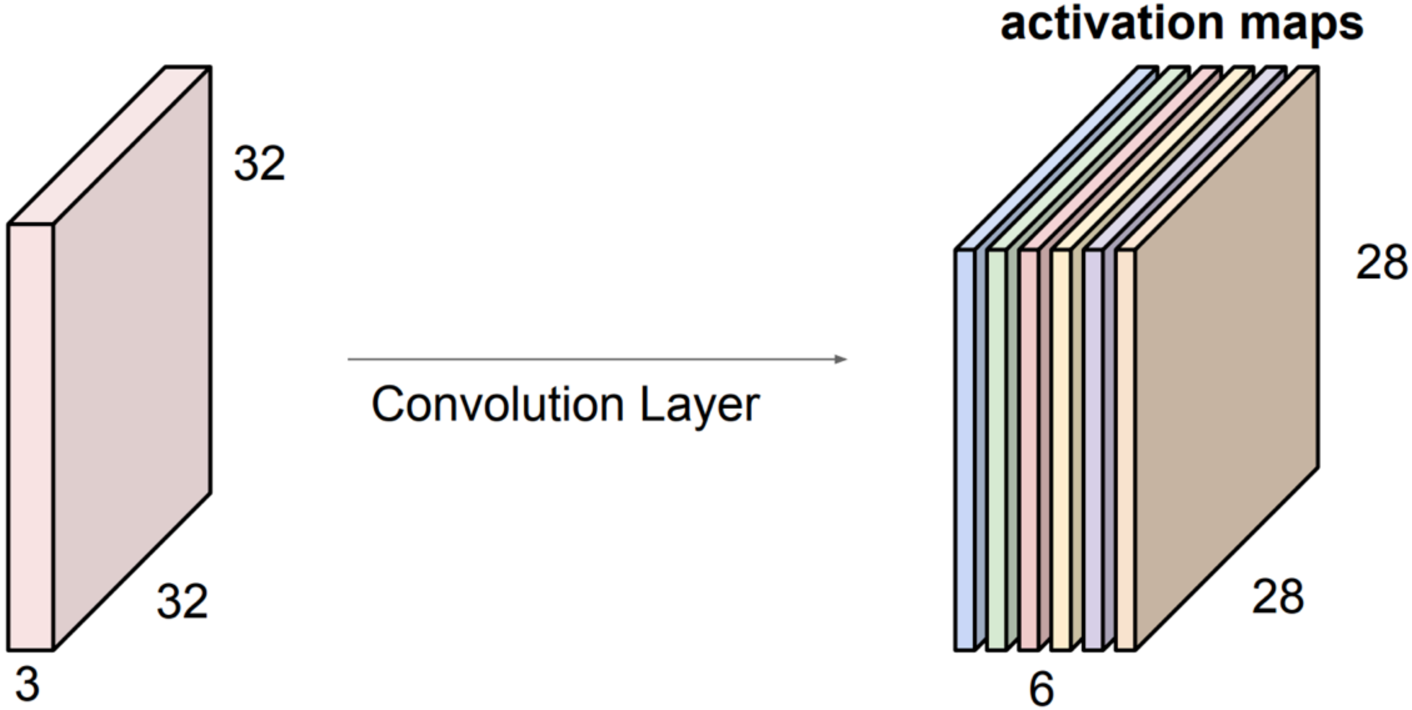
consider a second, green filter



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

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!

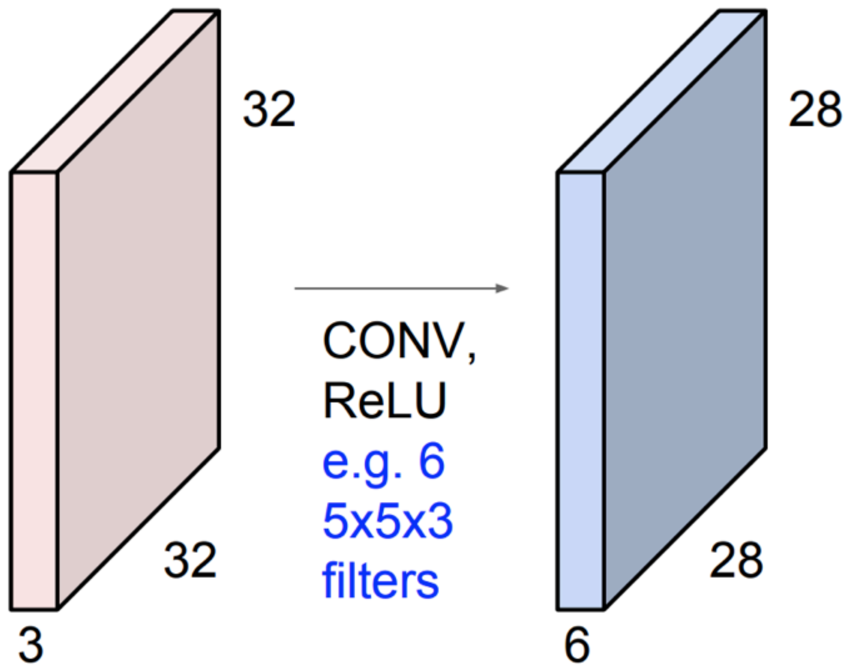
focal Systems

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

UNIVERSITY OF WATERLOO

Convolutional Layer

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



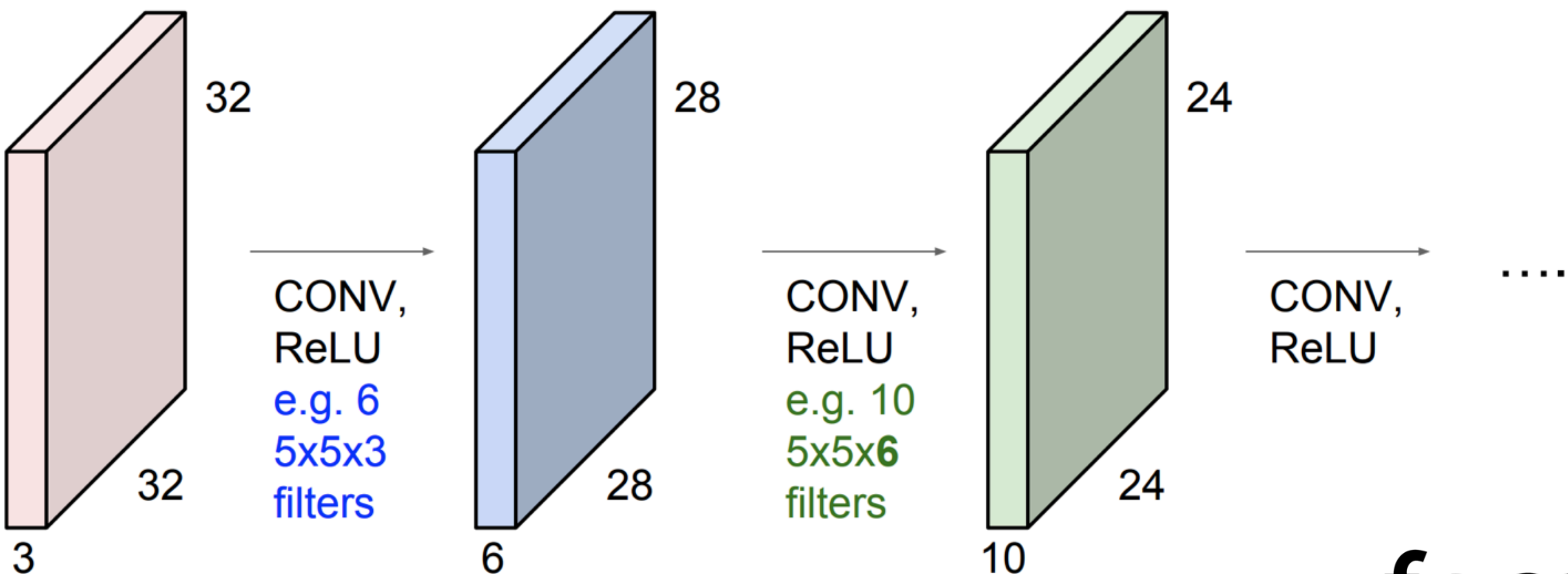
focal Systems

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

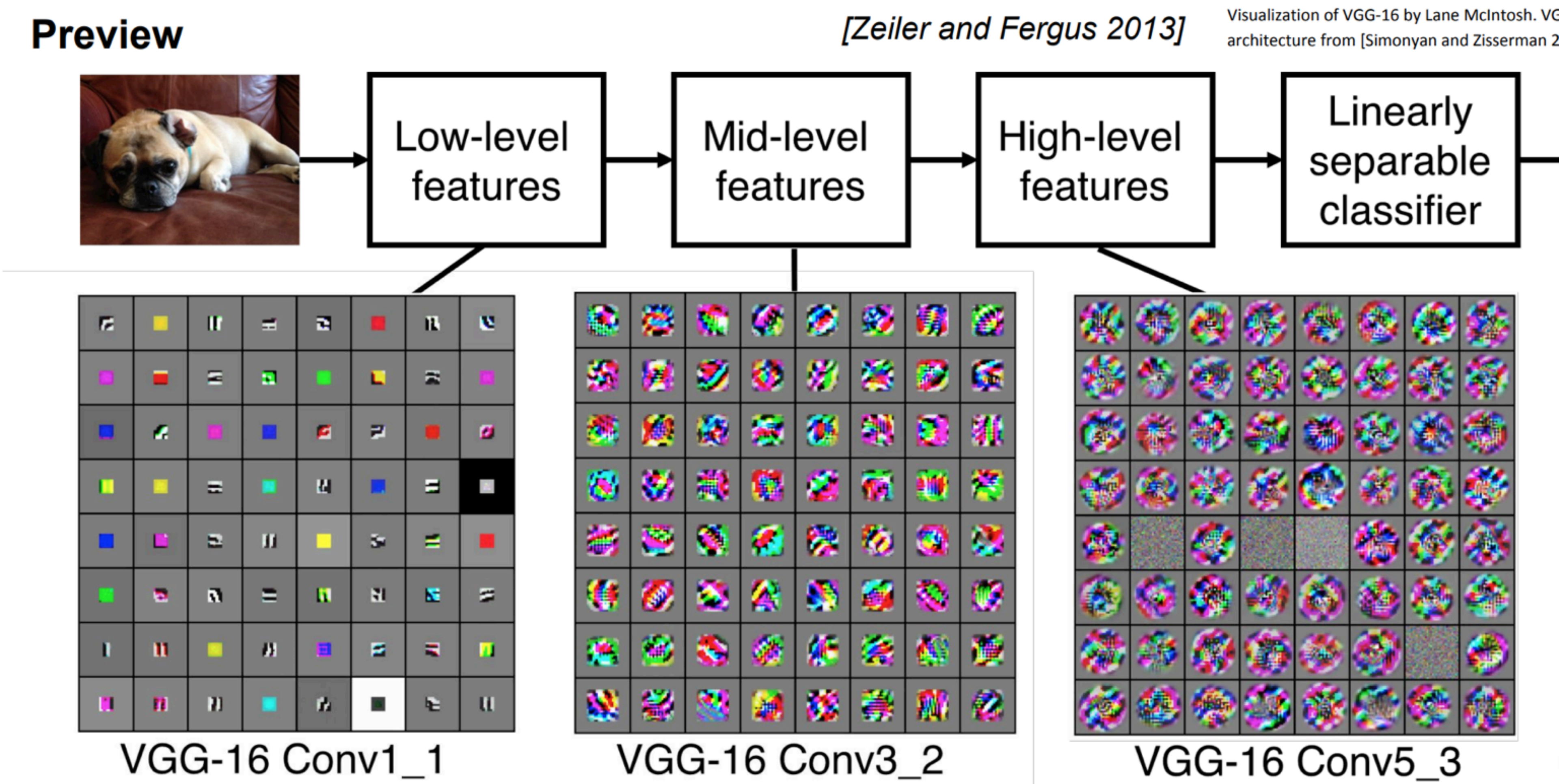
Convolutional Layer

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



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

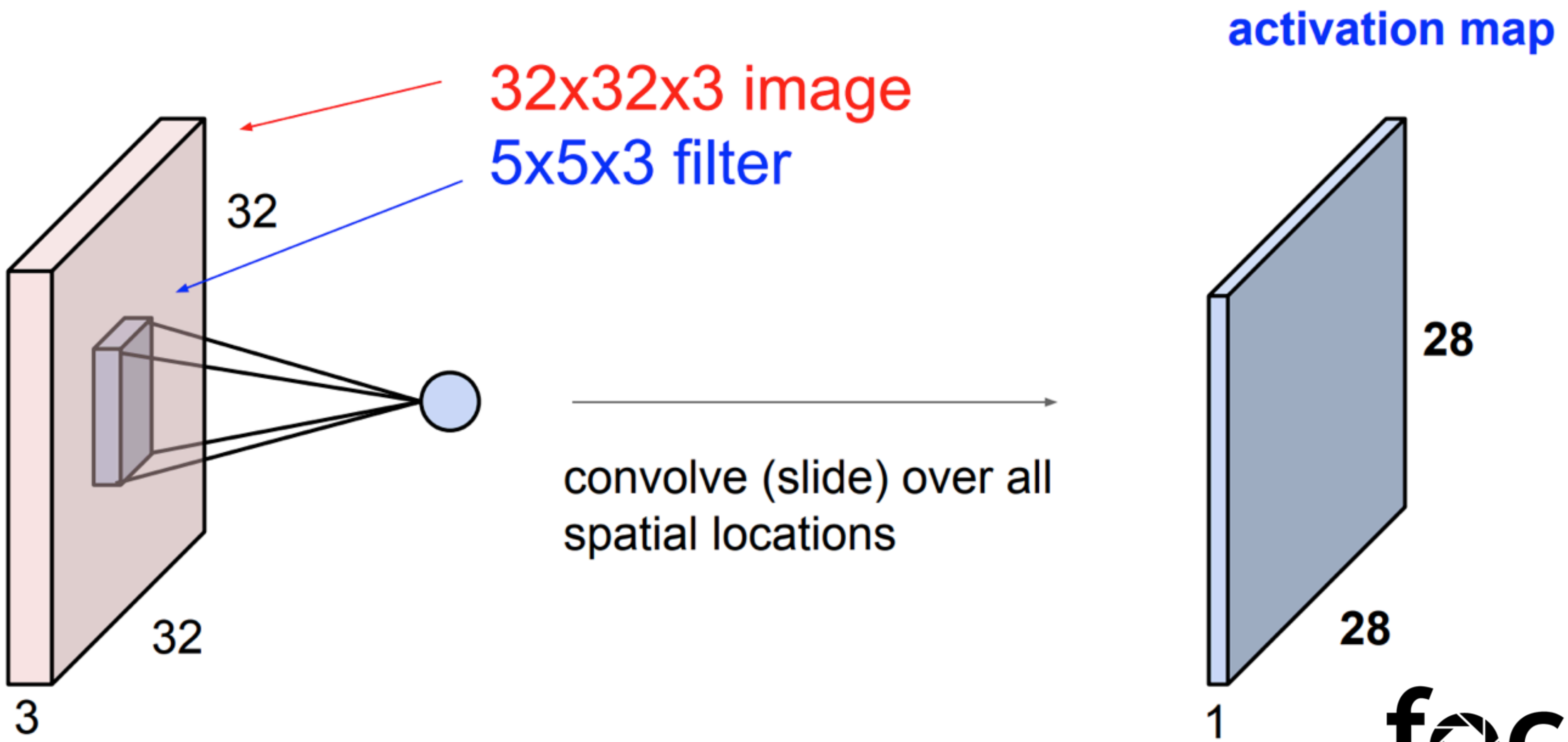
Convolutional Layer



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

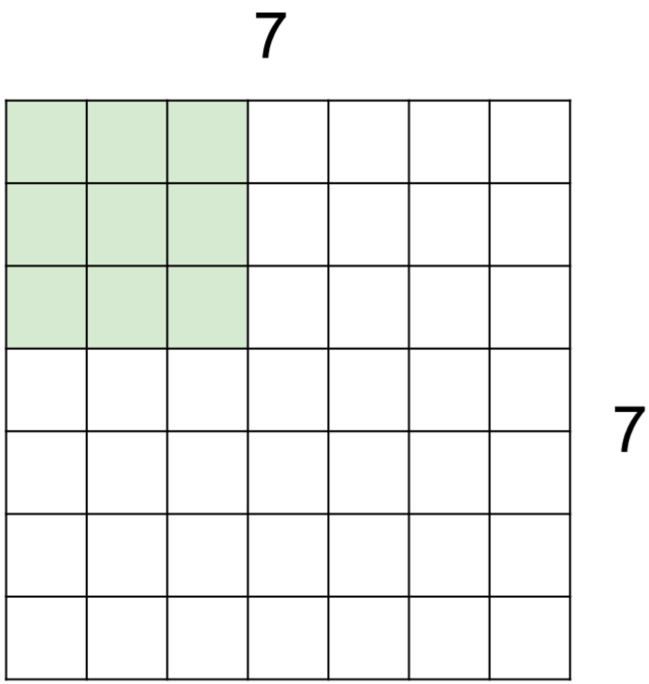
A closer look at spatial dimensions:



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

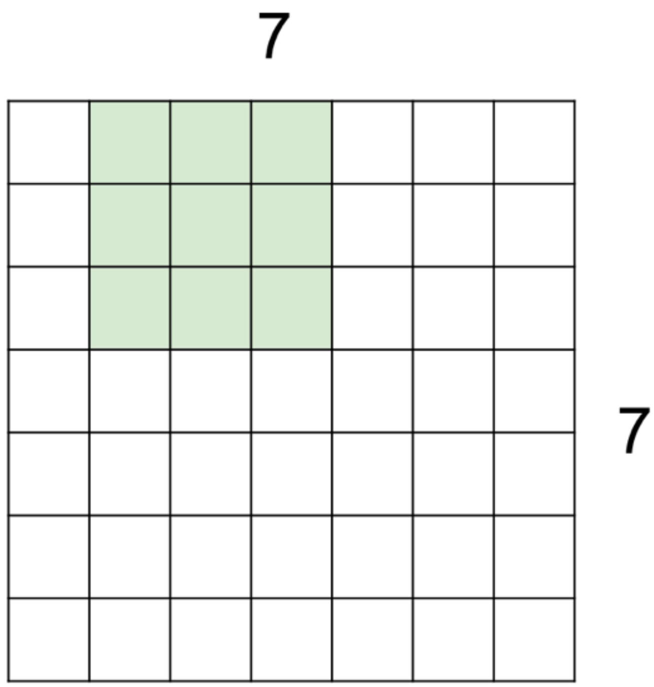
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

focal Systems

UNIVERSITY OF WATERLOO

Convolutional Layer

A closer look at spatial dimensions:

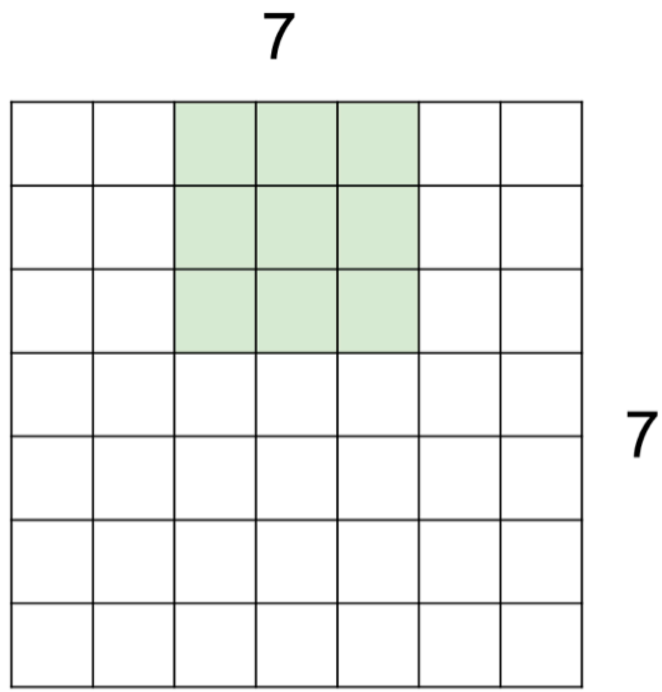


7x7 input (spatially)
assume 3x3 filter

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:



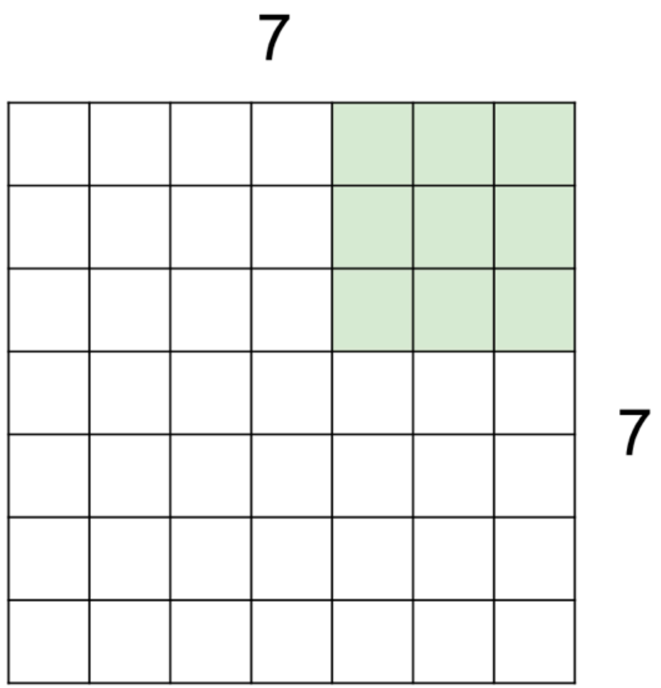
7x7 input (spatially)
assume 3x3 filter

focal Systems
UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:



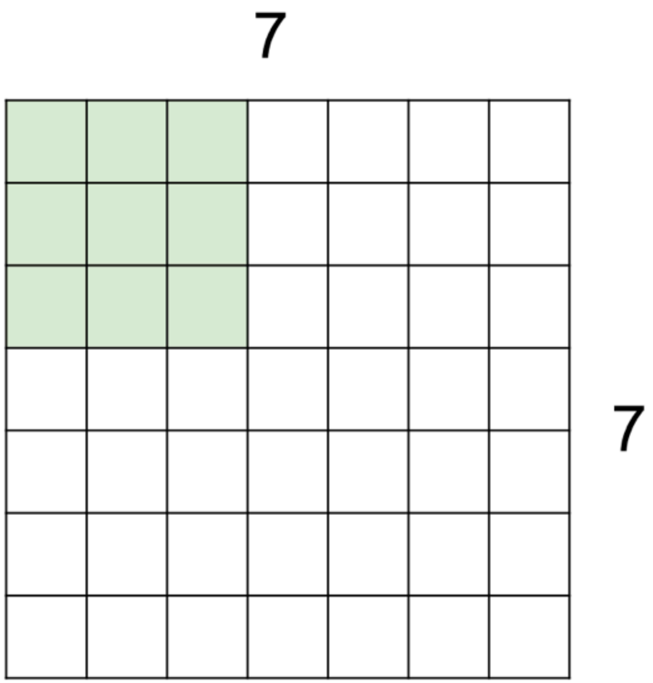
7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:

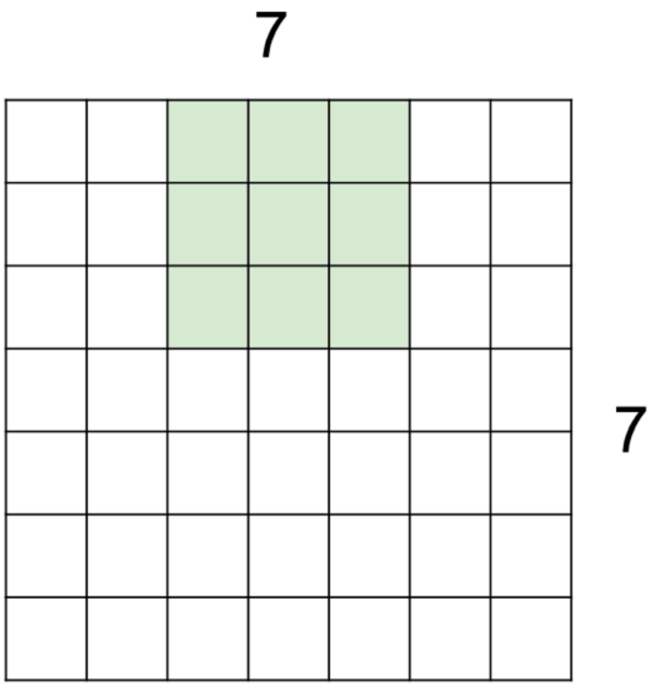


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:

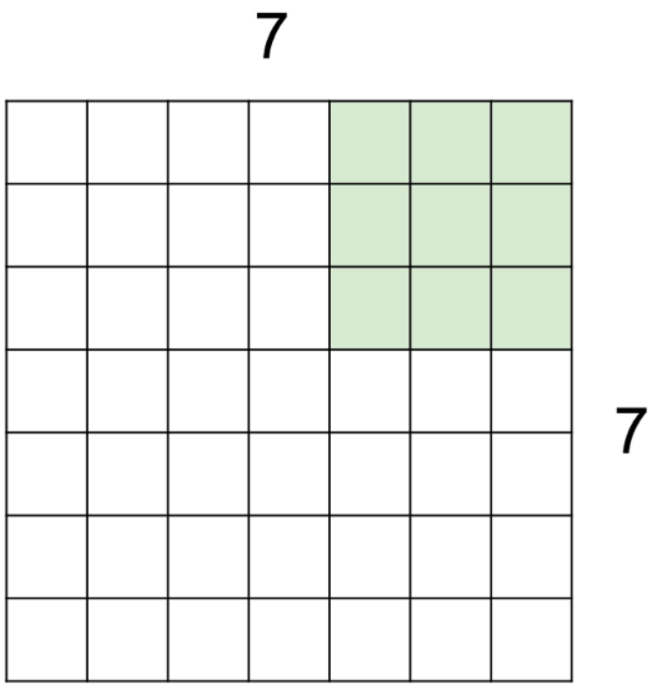


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:

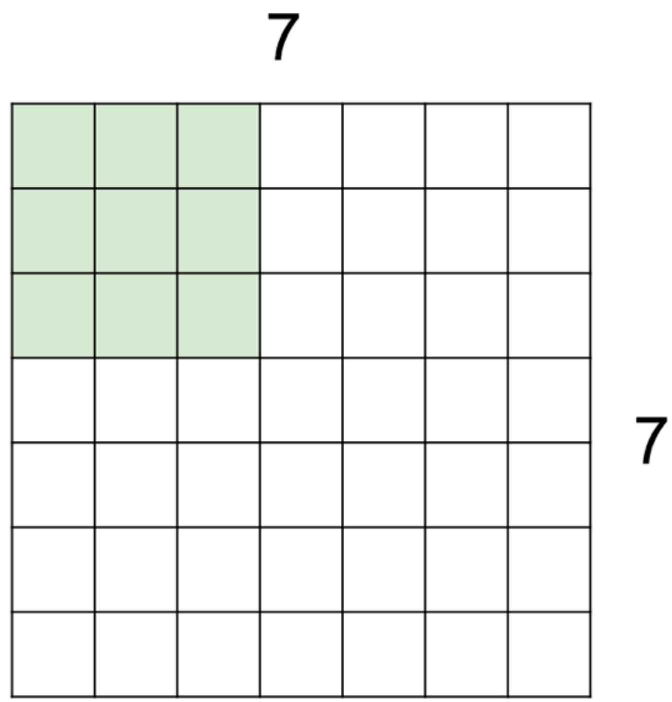


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:



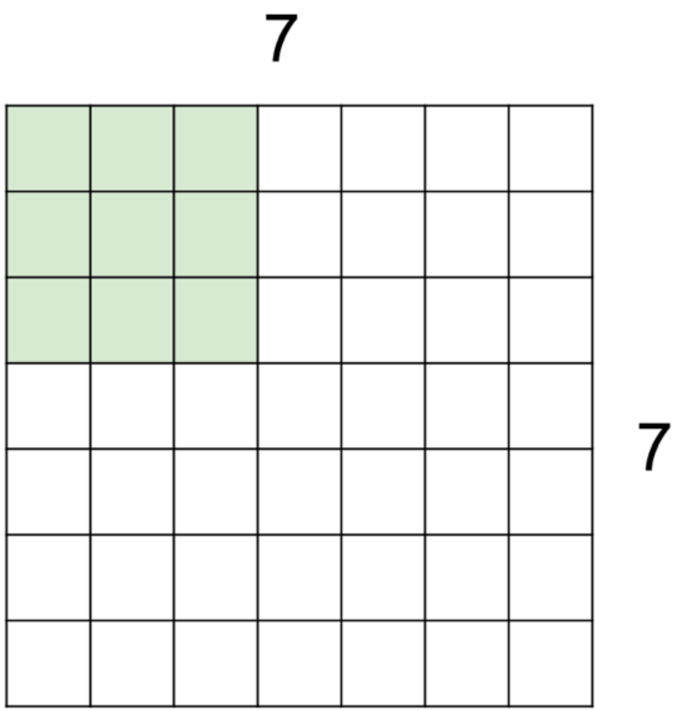
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Systems

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

A closer look at spatial dimensions:

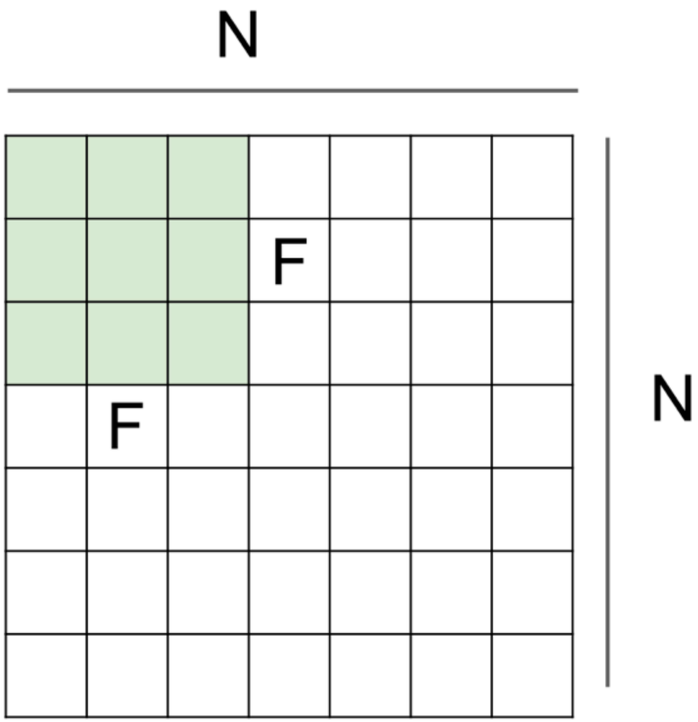


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer



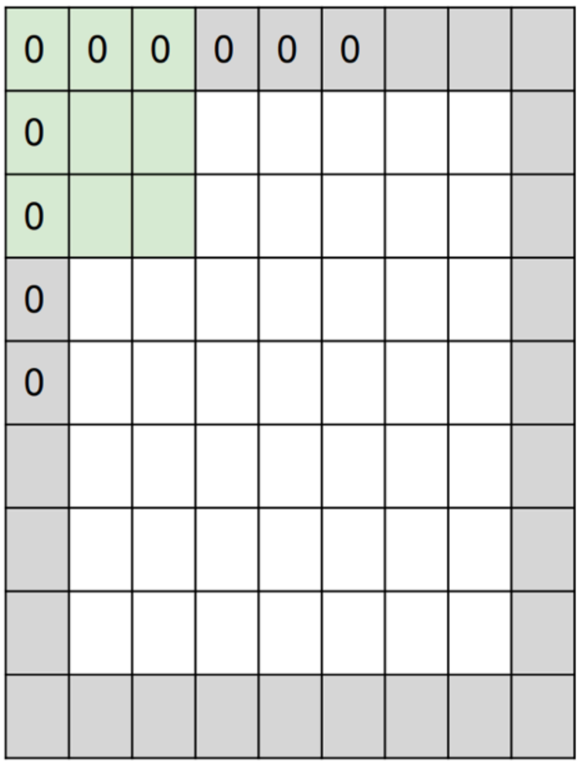
Output size:
(N - F) / stride + 1

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 :\

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

In practice: Common to zero pad the border



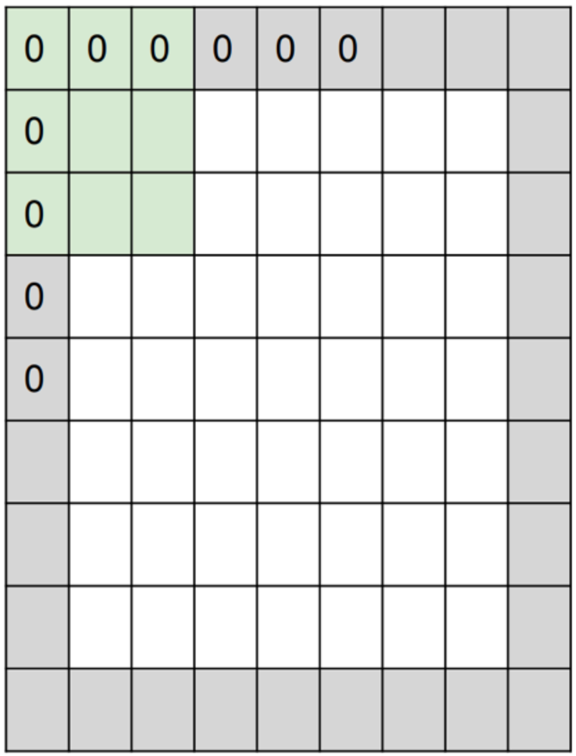
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

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?

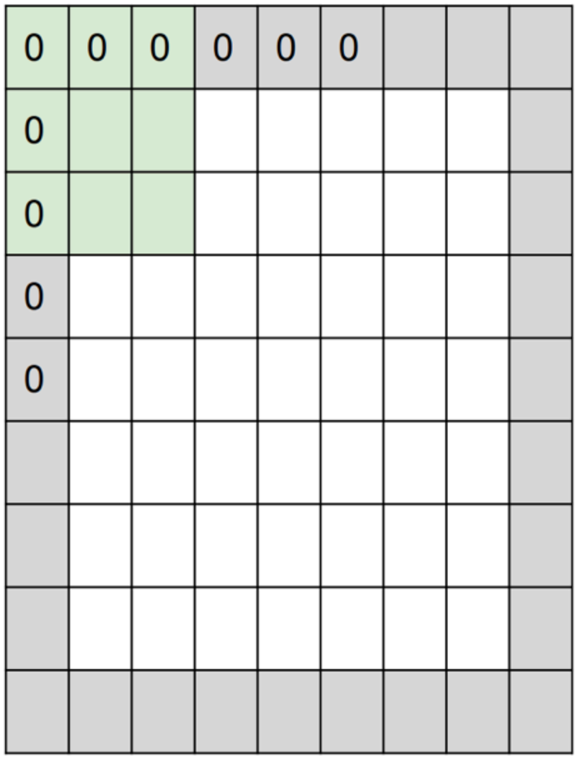
7x7 output!

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

focal Systems
UNIVERSITY OF WATERLOO

Convolutional Layer

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?

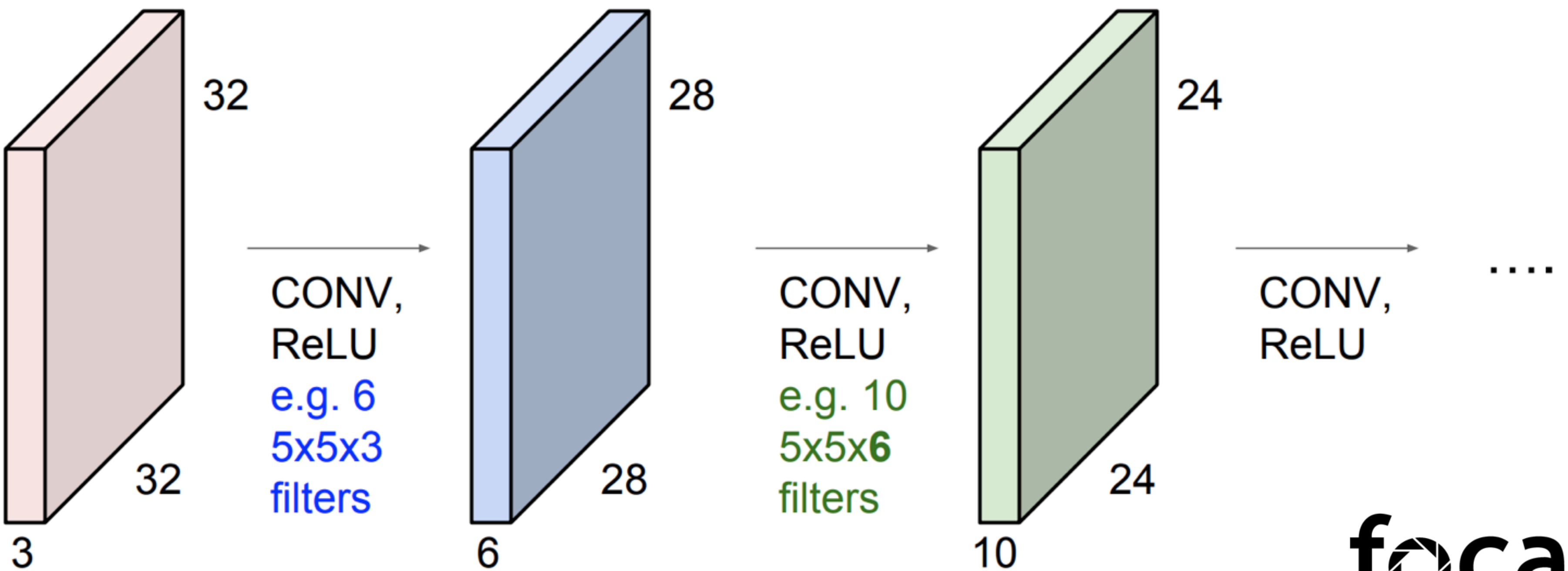
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

focal Systems
UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

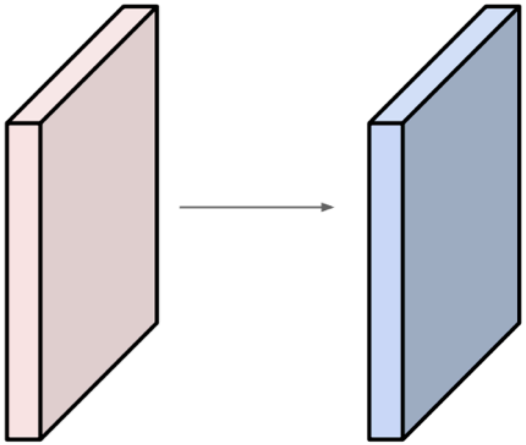
Remember back to…
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

Examples time:



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

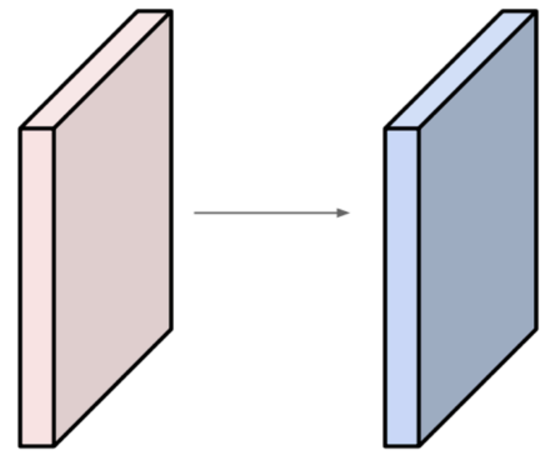
Output volume size: ?

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

focal Systems
UNIVERSITY OF WATERLOO

Convolutional Layer

Examples time:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

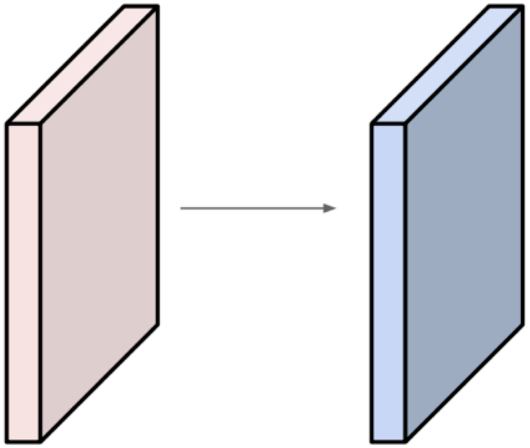
(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

Examples time:



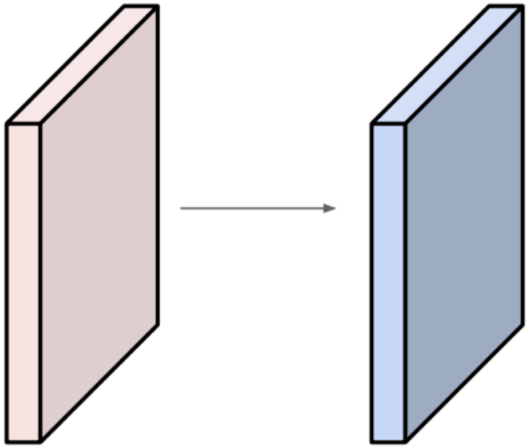
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional 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**

focal Systems

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

UNIVERSITY OF WATERLOO

Convolutional Layer

Summary. 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.

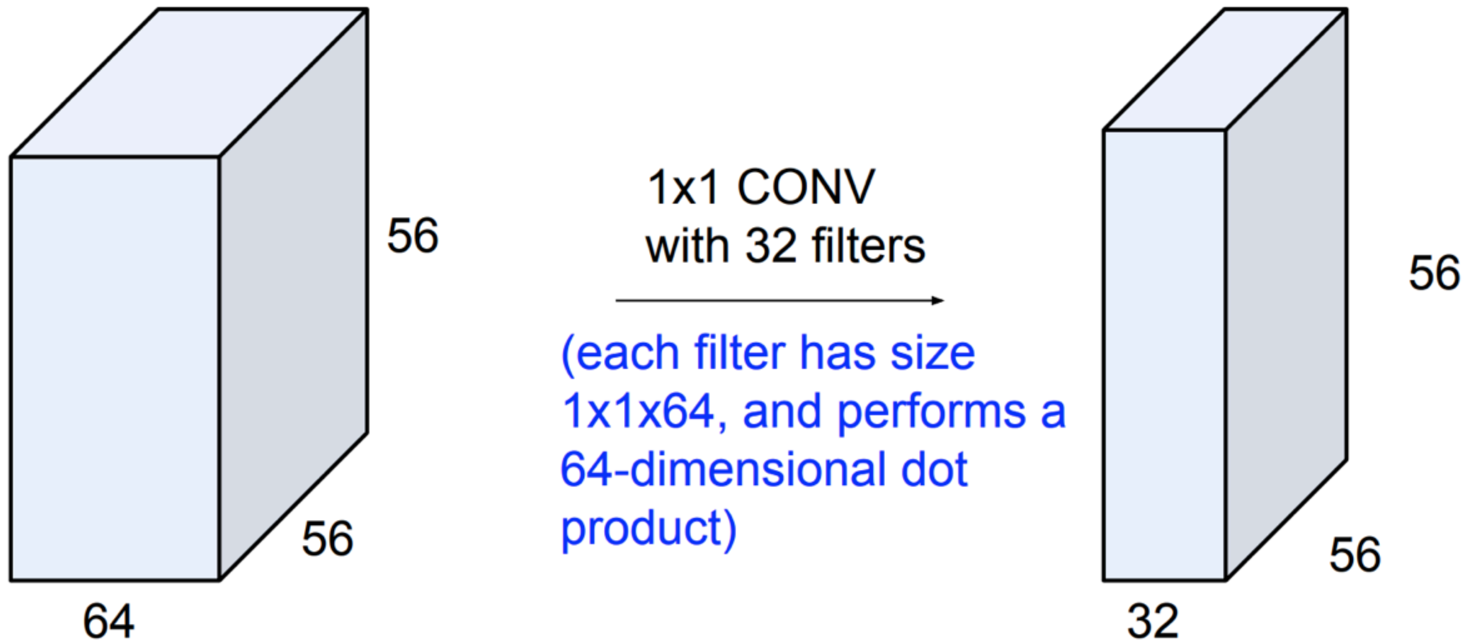
focal Systems

UNIVERSITY OF WATERLOO

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

Convolutional Layer

(btw, 1x1 convolution layers make perfect sense)



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

focal Systems

Deep Learning for Retail