CS 4803 / 7643: Deep Learning

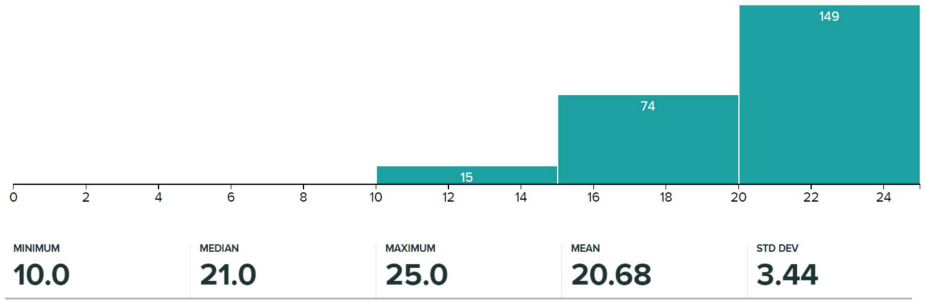
Topics:

- Specifying Layers
- Forward & Backward autodifferentiation
- (Beginning of) Convolutional neural networks

Zsolt Kira
Georgia Tech

Administrivia

- PS0 released
  - mean of 20.7
  - standard deviation of 3.4
  - median of 21
  - max of 25
  - See me if you did not pass



- PS1/HW1 out

- Start thinking about project topics/teams
  - More details on project next time

Recap from last time

Gradient Descent Pseudocode

```
for i in {0,…,num_epochs}:
    for x, y in data:
```

$$\hat{y} = SM(Wx)$$

$$L = CE(\hat{y}, y)$$

$$\frac{dL}{dW} = ???$$

$$W := W - \alpha \frac{dL}{dW}$$

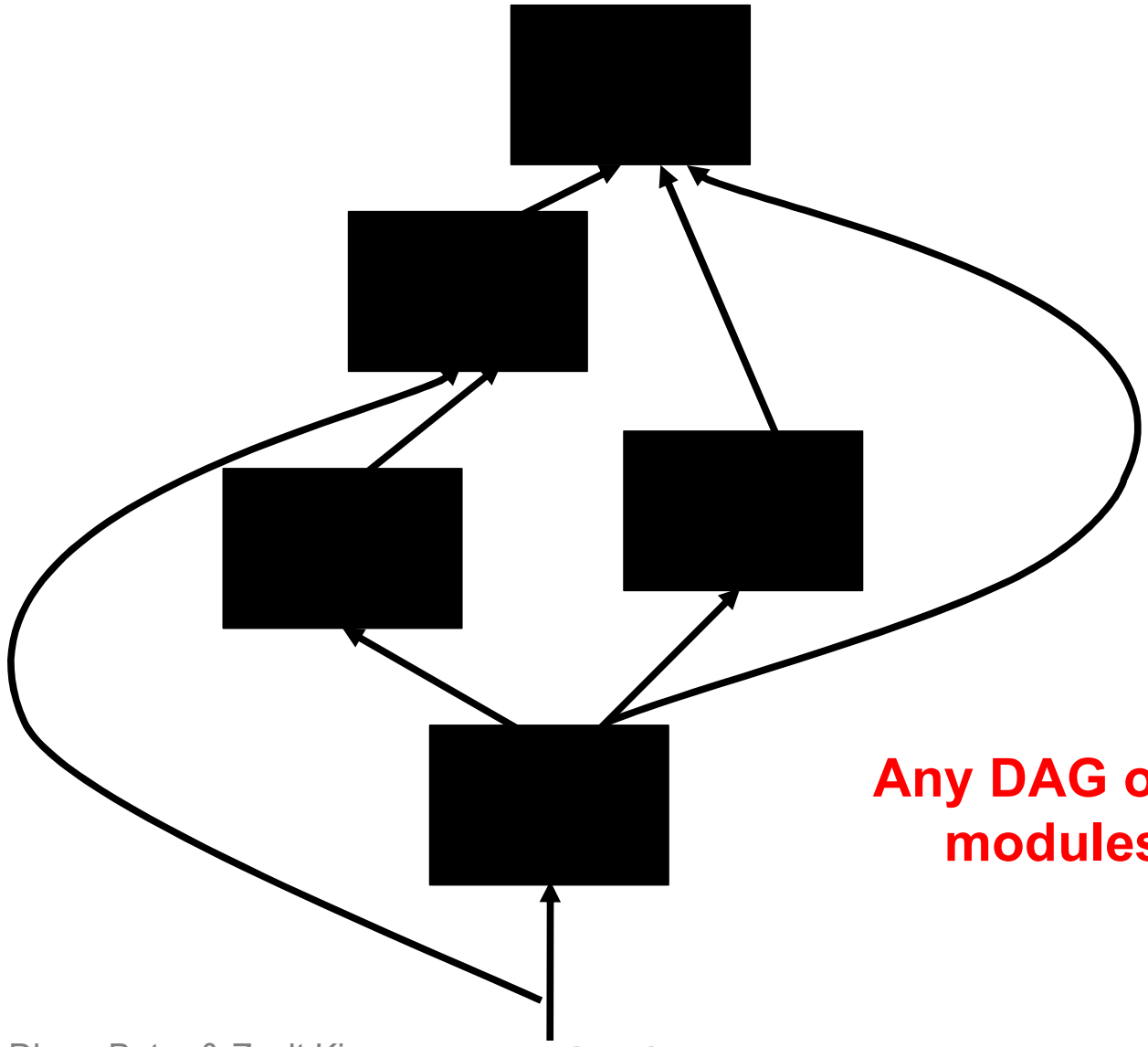
```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Some design decisions:

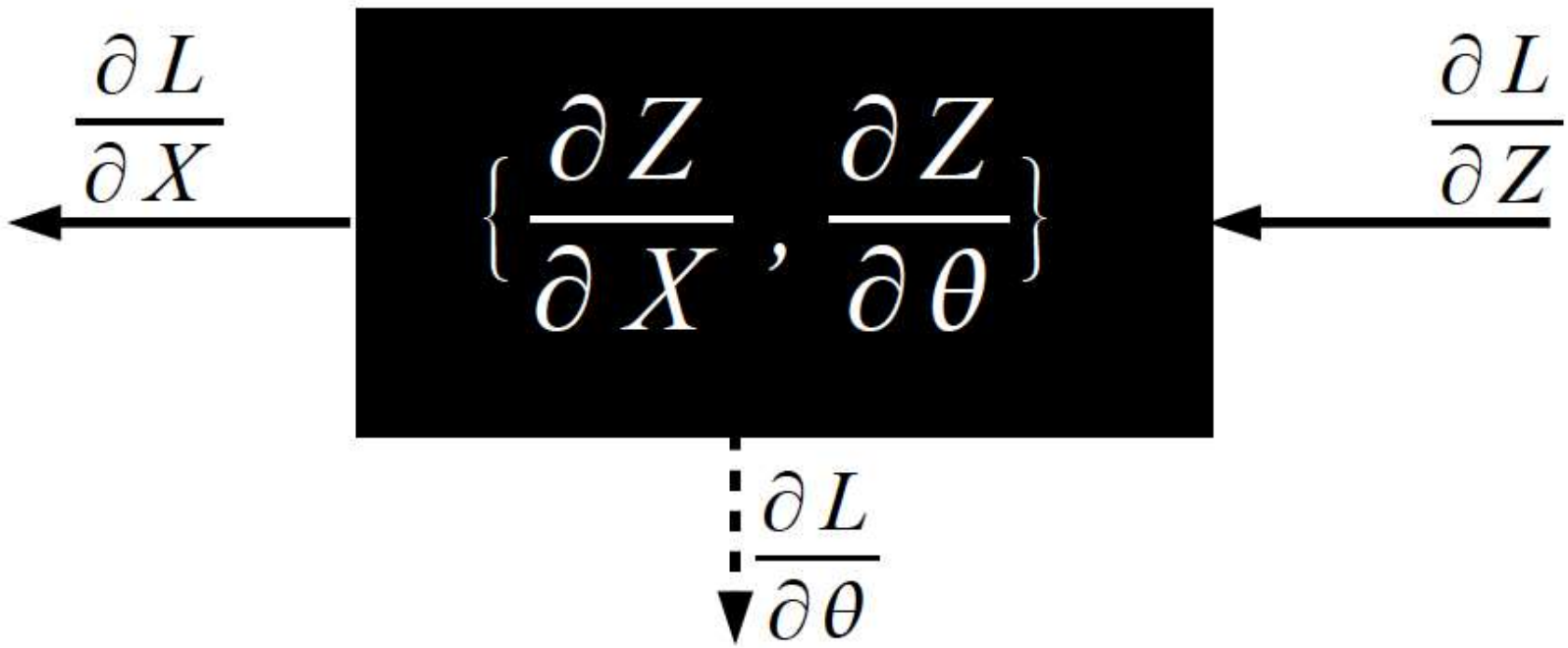
- How many examples to use to calculate gradient per iteration?
- What should alpha (learning rate) be?
  - Should it be constant throughout?
- How many epochs to run to?

Computational Graph



Any DAG of differentiable modules is allowed!

Key Computation: Back-Prop



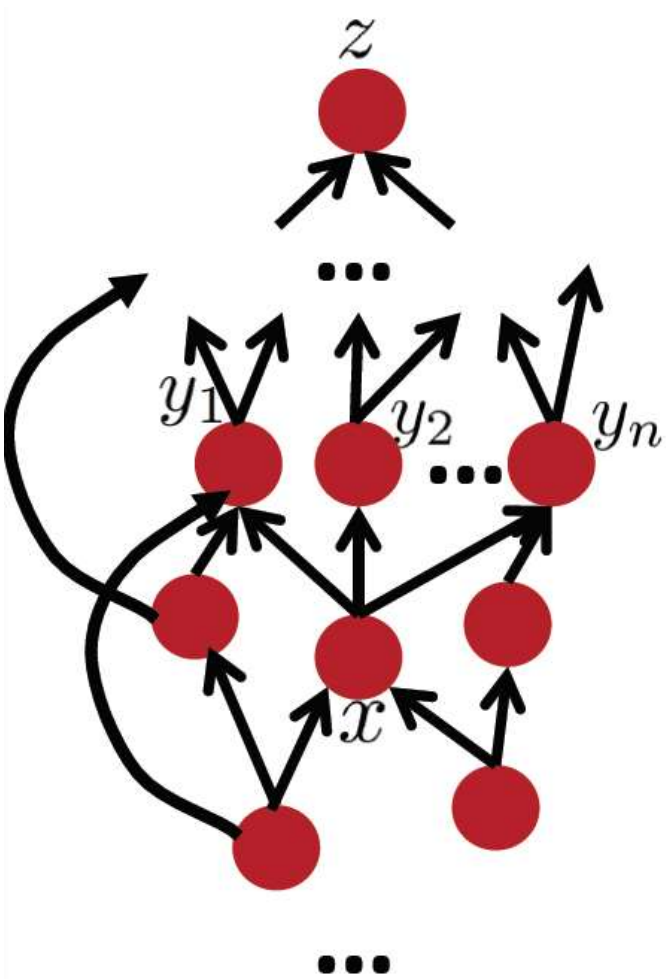
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]

Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]

General Flow Graphs

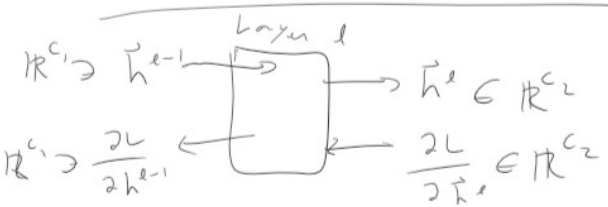


Flow graph: any directed acyclic graph
node = computation result
arc = computation dependency

$\{y_1, y_2, \ldots y_n\}$ = successors of x

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

"Deep Learning" book, Bengio



Forward Pass $\quad h_i^{\ell} = \sum_{j=1}^{c_1} w_{ij}\, h_j^{\ell-1}$

$$= \vec{w}^{\,T} \vec{h}^{\ell-1}$$

Vector

$$\underset{c_2 \times 1}{\begin{bmatrix} \vec{h}^{\ell} \\ h_1^{\ell} \\ \vdots \\ h_{c_2}^{\ell} \end{bmatrix}} = \underset{c_2 \times c_1}{\begin{bmatrix} \leftarrow & w_1^T & \rightarrow \\ & \vdots & \\ \leftarrow & w_{c_2}^T & \rightarrow \end{bmatrix}} \underset{c_1 \times 1}{\begin{bmatrix} h^{\ell-1} \\ h_1^{\ell-1} \\ \vdots \\ h_{c_1}^{\ell-1} \end{bmatrix}}$$

$$\vec{h}^{\ell} = W^{\ell} \vec{h}^{\ell-1}$$

$$\frac{\partial \vec{h}^{\ell}}{\partial \vec{h}^{\ell-1}} = W^{\ell}$$

$$\frac{\partial h_i^{\ell}}{\partial \vec{w}_i} = \vec{h}^{\ell-1\,T}$$

$$\frac{\partial L}{\partial h^{\ell-1}} = \frac{\partial L}{\partial \vec{h}^{\ell}} \cdot \frac{\partial \vec{h}^{\ell}}{\partial \vec{h}^{\ell-1}} = \frac{\partial L}{\partial \vec{h}^{\ell}} \times W^{\ell}$$

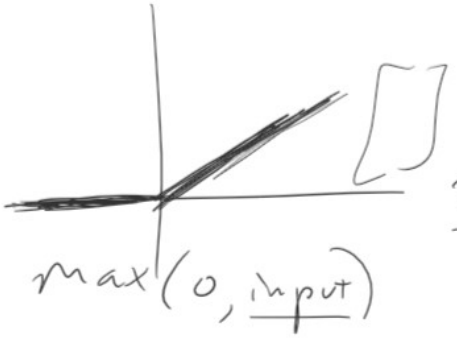
$$\frac{\partial L}{\partial \vec{w}_i} = \frac{\partial L}{\partial \vec{h}_i^{\ell}} \cdot \frac{\partial h_i^{\ell}}{\partial \vec{w}_i} = \frac{\partial L}{\partial h_i^{\ell}} \cdot \vec{h}^{\ell-1\,T}$$

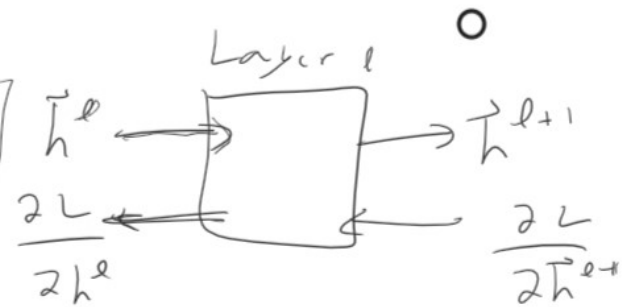
$10 \times 32 \times 1$

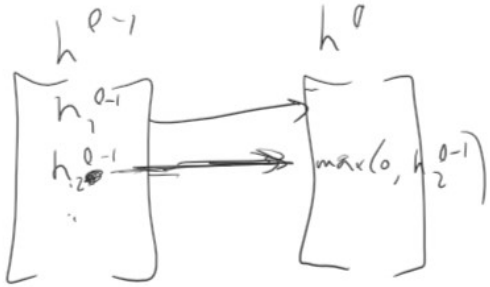
$10 \times 100 \times 100 \longrightarrow$

$(10 \times 100 \times 100) \times 1$

ReLU







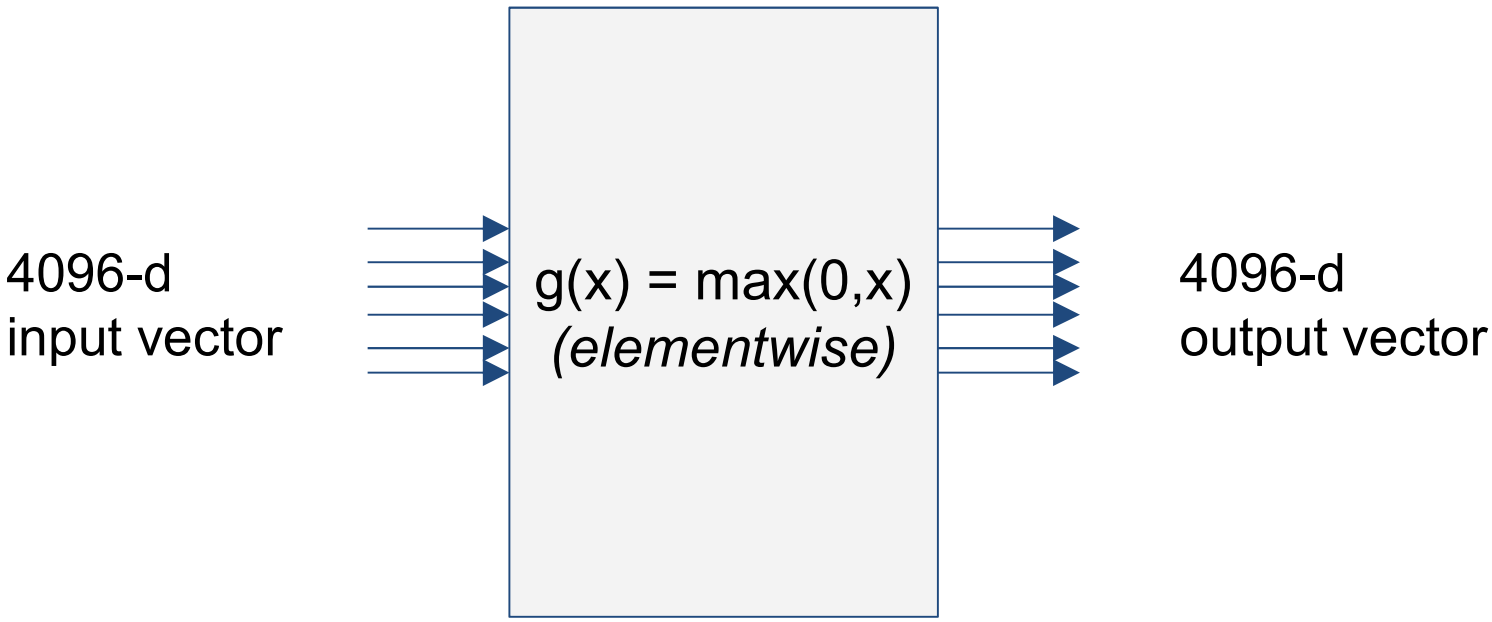
Forward pass $h_i^{\ell+1} = \max\{0, h_i^{\ell}\}$

Backwards pass $\boxed{\frac{\partial h_i^{\ell+1}}{\partial h_i^{\ell}} = \begin{cases} 1 & \text{if } h_i^{\ell} \geq 0 \\ 0 & \text{otherwise} \end{cases}}$

$$\frac{\partial \vec{h}^{\ell}}{\partial \vec{h}^{\ell-1}}$$

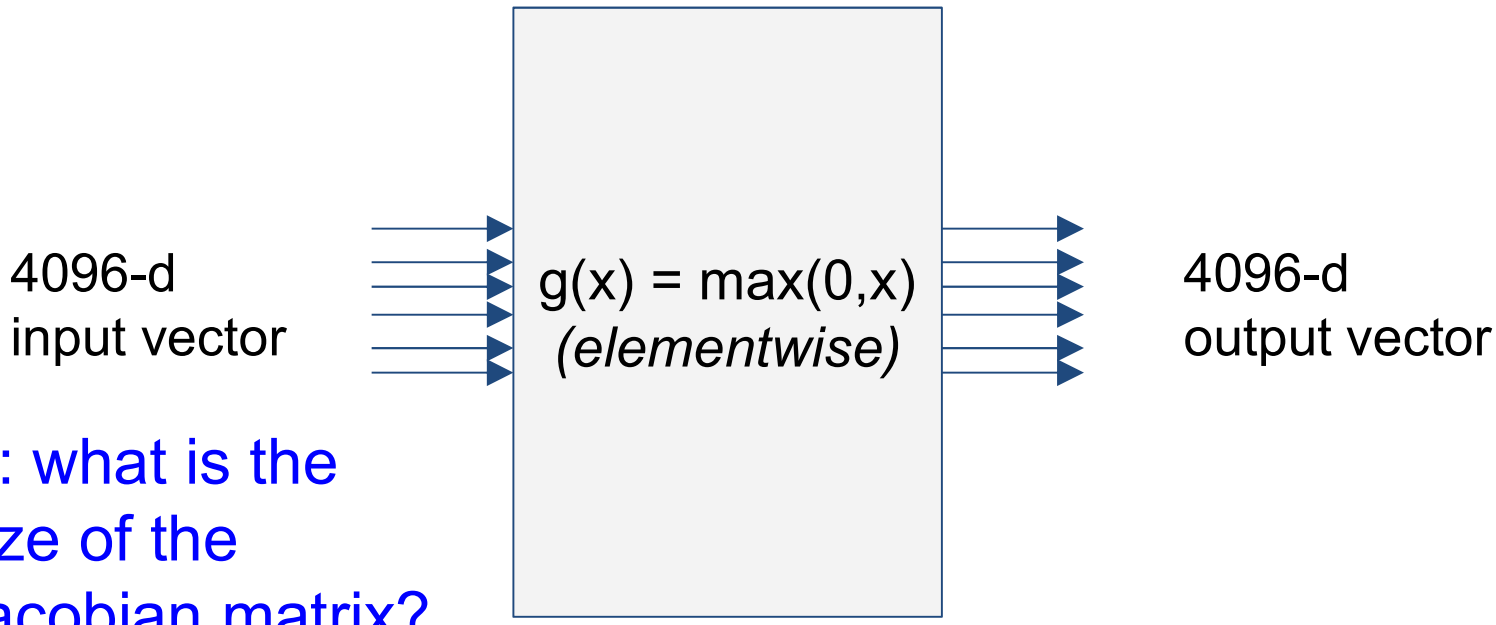
$$\frac{\partial L}{\partial h_i^{\ell}} = \frac{\partial L}{\partial h_i^{\ell+1}} \times \underline{\mathbb{1}\{h_i^{\ell} \geq 0\}}$$

Jacobian of ReLU



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

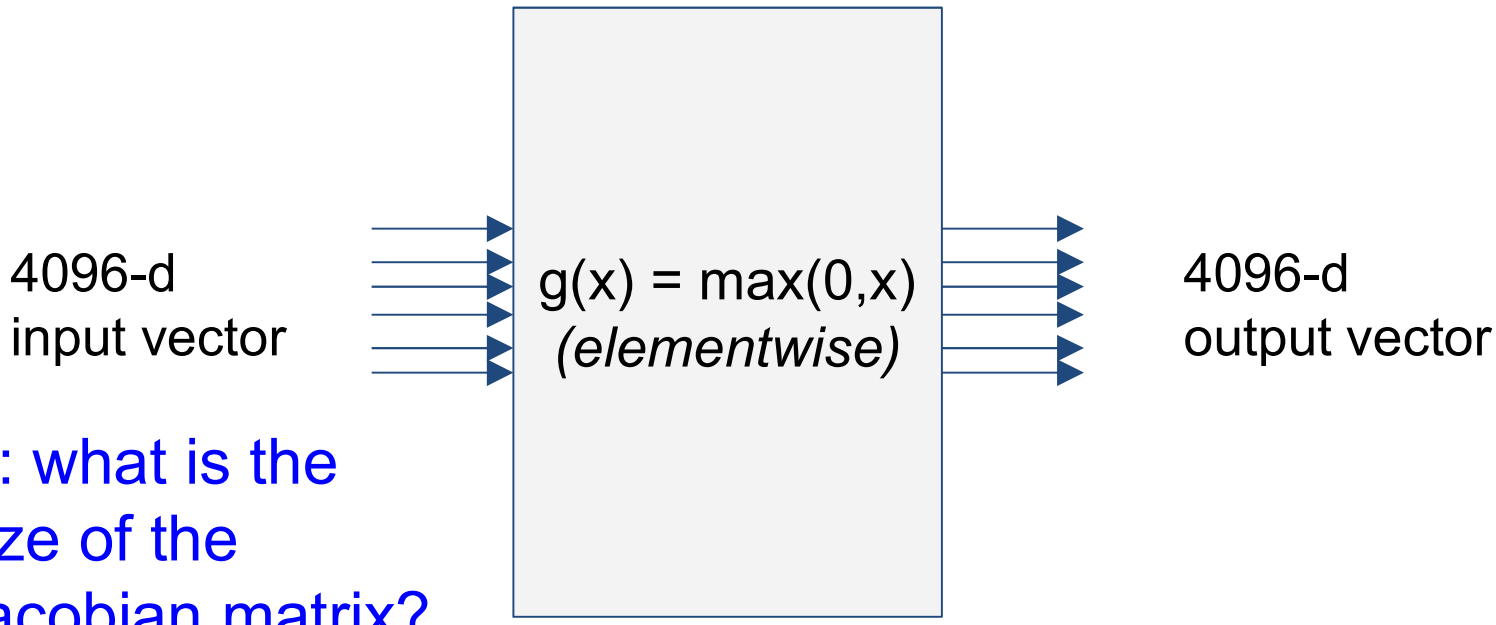
Jacobian of ReLU



Q: what is the size of the Jacobian matrix?

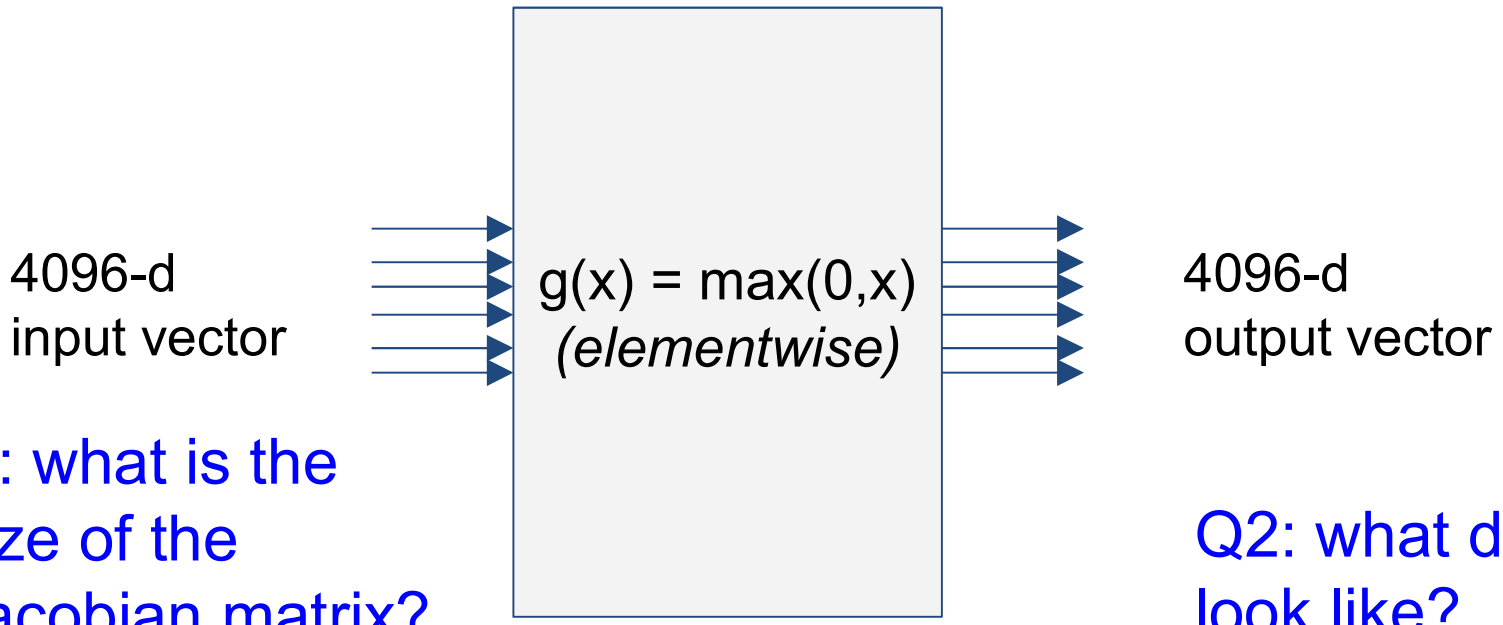
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Jacobian of ReLU



Q: what is the size of the Jacobian matrix? [4096 x 4096!]

Jacobian of ReLU



Q: what is the size of the Jacobian matrix? [4096 x 4096!]

Q2: what does it look like?

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Plan for Today

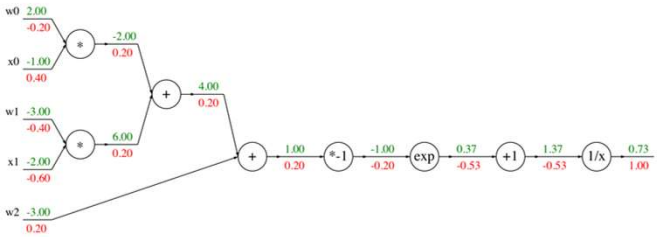
- Specifying Layers
- Forward & Backward auto-differentiation
- (Beginning of) Convolutional neural networks

Deep Learning = Differentiable Programming

- Computation = Graph
  - Input = Data + Parameters
  - Output = Loss
  - Scheduling = Topological ordering

- What do we need to do?
  - Generic code for representing the graph of modules
  - Specify modules (both forward and backward function)

Modularized implementation: forward / backward API

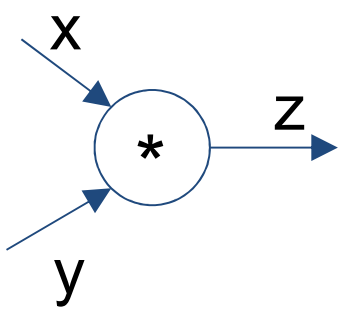


Graph (or Net) object *(rough psuedo code)*

```
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Modularized implementation: forward / backward API



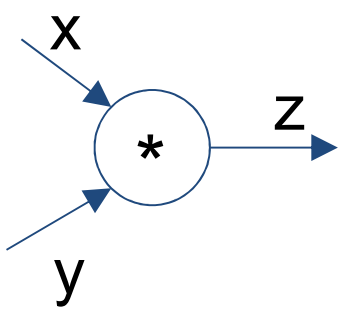
(x,y,z are scalars)

```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        return z
    def backward(dz):
        # dx = ... #todo
        # dy = ... #todo
        return [dx, dy]
```

$$\frac{\partial L}{\partial z}$$

$$\frac{\partial L}{\partial x}$$

Modularized implementation: forward / backward API



(x,y,z are scalars)

```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        self.x = x # must keep these around!
        self.y = y
        return z
    def backward(dz):
        dx = self.y * dz # [dz/dx * dL/dz]
        dy = self.x * dz # [dz/dy * dL/dz]
        return [dx, dy]
```

Example: Caffe layers

Branch: master | caffe / src / caffe / layers / | Create new file | Upload files | Find file | History

shelhamer committed on GitHub Merge pull request #4630 from BlGene/load_hdf5_fix ... Latest commit e687a71 21 days ago

..

absval_layer.cpp	dismantle layer headers	a year ago
absval_layer.cu	dismantle layer headers	a year ago
accuracy_layer.cpp	dismantle layer headers	a year ago
argmax_layer.cpp	dismantle layer headers	a year ago
base_conv_layer.cpp	enable dilated deconvolution	a year ago
base_data_layer.cpp	Using default from proto for prefetch	3 months ago
base_data_layer.cu	Switched multi-GPU to NCCL	3 months ago
batch_norm_layer.cpp	Add missing spaces besides equal signs in batch_norm_layer.cpp	4 months ago
batch_norm_layer.cu	dismantle layer headers	a year ago
batch_reindex_layer.cpp	dismantle layer headers	a year ago
batch_reindex_layer.cu	dismantle layer headers	a year ago
bias_layer.cpp	Remove incorrect cast of gemm int arg to Dtype in BiasLayer	a year ago
bias_layer.cu	Separation and generalization of ChannelwiseAffineLayer into BiasLayer	a year ago
bnll_layer.cpp	dismantle layer headers	a year ago
bnll_layer.cu	dismantle layer headers	a year ago
concat_layer.cpp	dismantle layer headers	a year ago
concat_layer.cu	dismantle layer headers	a year ago
contrastive_loss_layer.cpp	dismantle layer headers	a year ago
contrastive_loss_layer.cu	dismantle layer headers	a year ago
conv_layer.cpp	add support for 2D dilated convolution	a year ago
conv_layer.cu	dismantle layer headers	a year ago
crop_layer.cpp	remove redundant operations in Crop layer (#5138)	2 months ago
crop_layer.cu	remove redundant operations in Crop layer (#5138)	2 months ago
cudnn_conv_layer.cpp	dismantle layer headers	a year ago
cudnn_conv_layer.cu	Add cuDNN v5 support, drop cuDNN v3 support	11 months ago
cudnn_lcn_layer.cpp	dismantle layer headers	a year ago
cudnn_lcn_layer.cu	dismantle layer headers	a year ago
cudnn_lrn_layer.cpp	dismantle layer headers	a year ago
cudnn_lrn_layer.cu	dismantle layer headers	a year ago
cudnn_pooling_layer.cpp	dismantle layer headers	a year ago
cudnn_pooling_layer.cu	dismantle layer headers	a year ago
cudnn_relu_layer.cpp	Add cuDNN v5 support, drop cuDNN v3 support	11 months ago
cudnn_relu_layer.cu	Add cuDNN v5 support, drop cuDNN v3 support	11 months ago
cudnn_sigmoid_layer.cpp	Add cuDNN v5 support, drop cuDNN v3 support	11 months ago
cudnn_sigmoid_layer.cu	Add cuDNN v5 support, drop cuDNN v3 support	11 months ago
cudnn_softmax_layer.cpp	dismantle layer headers	a year ago
cudnn_softmax_layer.cu	dismantle layer headers	a year ago
cudnn_tanh_layer.cpp	Add cuDNN v5 support, drop cuDNN v3 support	11 months ago
cudnn_tanh_layer.cu	Add cuDNN v5 support, drop cuDNN v3 support	11 months ago
data_layer.cpp	Switched multi-GPU to NCCL	3 months ago
deconv_layer.cpp	enable dilated deconvolution	a year ago
deconv_layer.cu	dismantle layer headers	a year ago
dropout_layer.cpp	supporting N-D Blobs in Dropout layer Reshape	a year ago
dropout_layer.cu	dismantle layer headers	a year ago
dummy_data_layer.cpp	dismantle layer headers	a year ago
eltwise_layer.cpp	dismantle layer headers	a year ago
eltwise_layer.cu	dismantle layer headers	a year ago
elu_layer.cpp	ELU layer with basic tests	a year ago
elu_layer.cu	ELU layer with basic tests	a year ago
embed_layer.cpp	dismantle layer headers	a year ago
embed_layer.cu	dismantle layer headers	a year ago
euclidean_loss_layer.cpp	dismantle layer headers	a year ago
euclidean_loss_layer.cu	dismantle layer headers	a year ago
exp_layer.cpp	Solving issue with exp layer with base e	a year ago
exp_layer.cu	dismantle layer headers	a year ago

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Caffe Sigmoid Layer

```cpp
#include <cmath>
#include <vector>

#include "caffe/layers/sigmoid_layer.hpp"

namespace caffe {

template <typename Dtype>
inline Dtype sigmoid(Dtype x) {
  return 1. / (1. + exp(-x));
}

template <typename Dtype>
void SigmoidLayer<Dtype>::Forward_cpu(const vector<Blob<Dtype>*>& bottom,
    const vector<Blob<Dtype>*>& top) {
  const Dtype* bottom_data = bottom[0]->cpu_data();
  Dtype* top_data = top[0]->mutable_cpu_data();
  const int count = bottom[0]->count();
  for (int i = 0; i < count; ++i) {
    top_data[i] = sigmoid(bottom_data[i]);
  }
}

template <typename Dtype>
void SigmoidLayer<Dtype>::Backward_cpu(const vector<Blob<Dtype>*>& top,
    const vector<bool>& propagate_down,
    const vector<Blob<Dtype>*>& bottom) {
  if (propagate_down[0]) {
    const Dtype* top_data = top[0]->cpu_data();
    const Dtype* top_diff = top[0]->cpu_diff();
    Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
    const int count = bottom[0]->count();
    for (int i = 0; i < count; ++i) {
      const Dtype sigmoid_x = top_data[i];
      bottom_diff[i] = top_diff[i] * sigmoid_x * (1. - sigmoid_x);
    }
  }
}

#ifdef CPU_ONLY
STUB_GPU(SigmoidLayer);
#endif

INSTANTIATE_CLASS(SigmoidLayer);


}  // namespace caffe
```

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$(1-\sigma(x))\,\sigma(x)$$ * top_diff (chain rule)

Deep Learning = Differentiable Programming

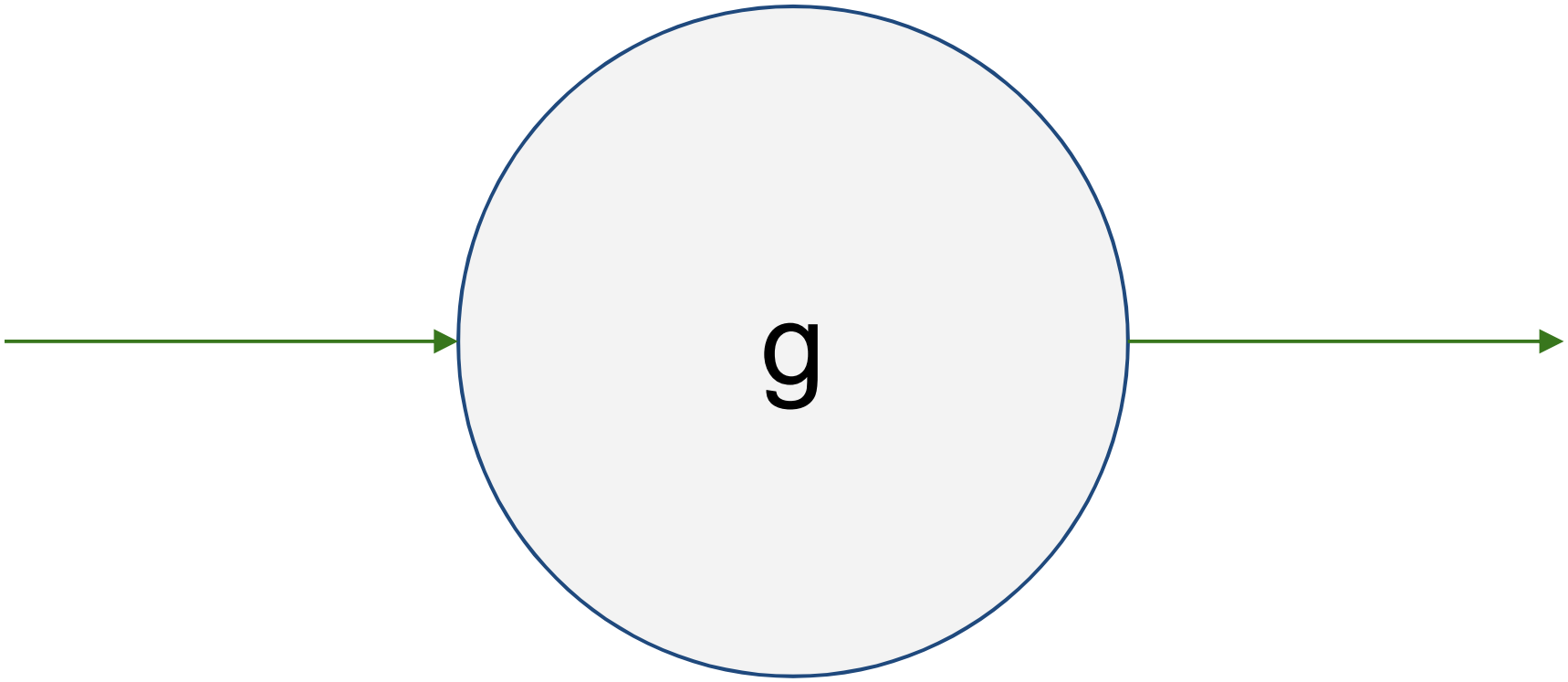
- Computation = Graph
  - Input = Data + Parameters
  - Output = Loss
  - Scheduling = Topological ordering

- **Auto-Diff**
  - A family of algorithms for implementing chain-rule on computation graphs

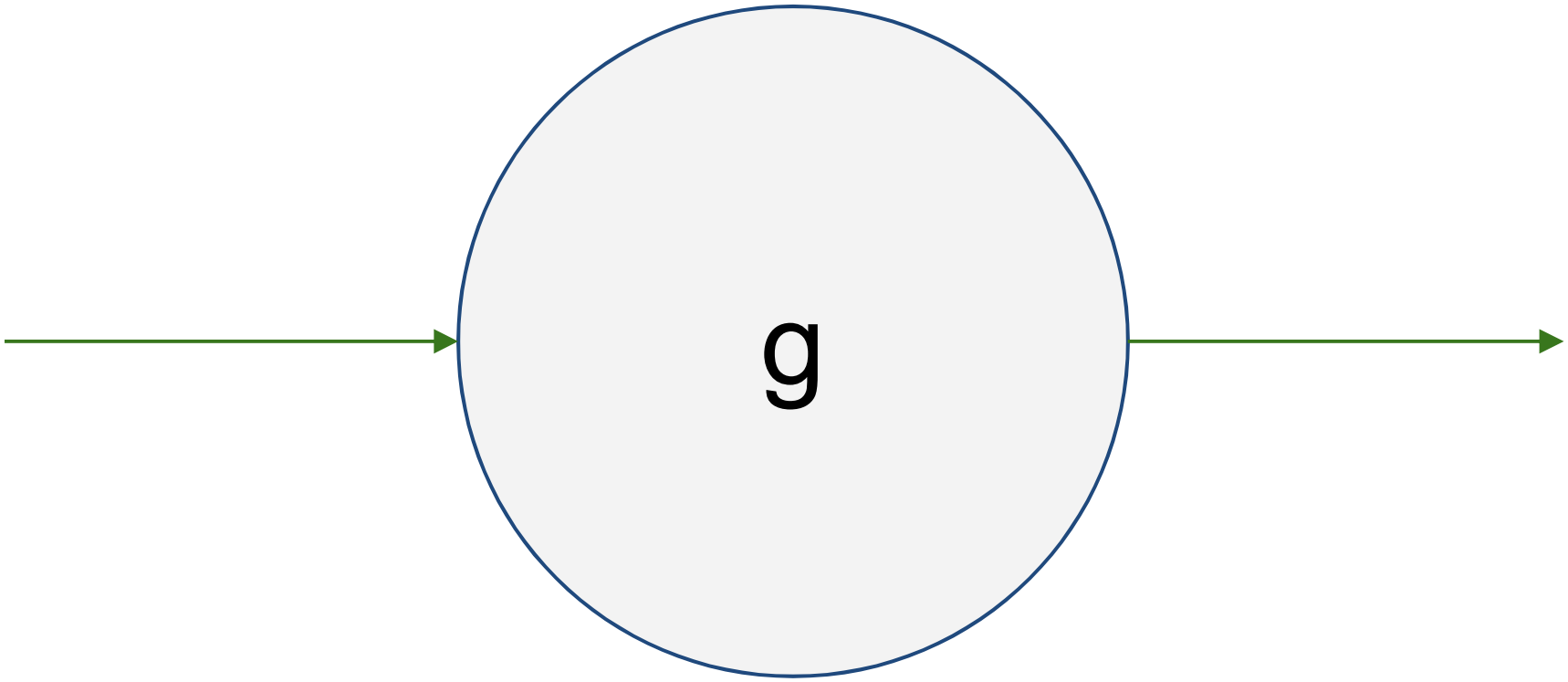
Forward mode vs Reverse Mode

- Key Computations

Forward mode AD

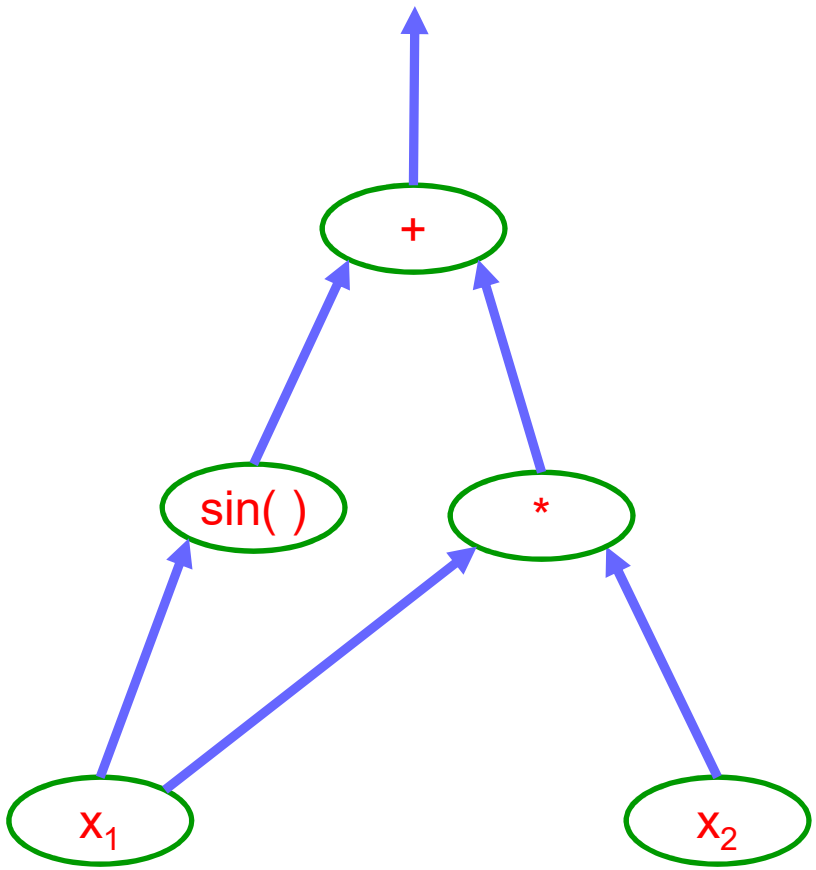


Reverse mode AD



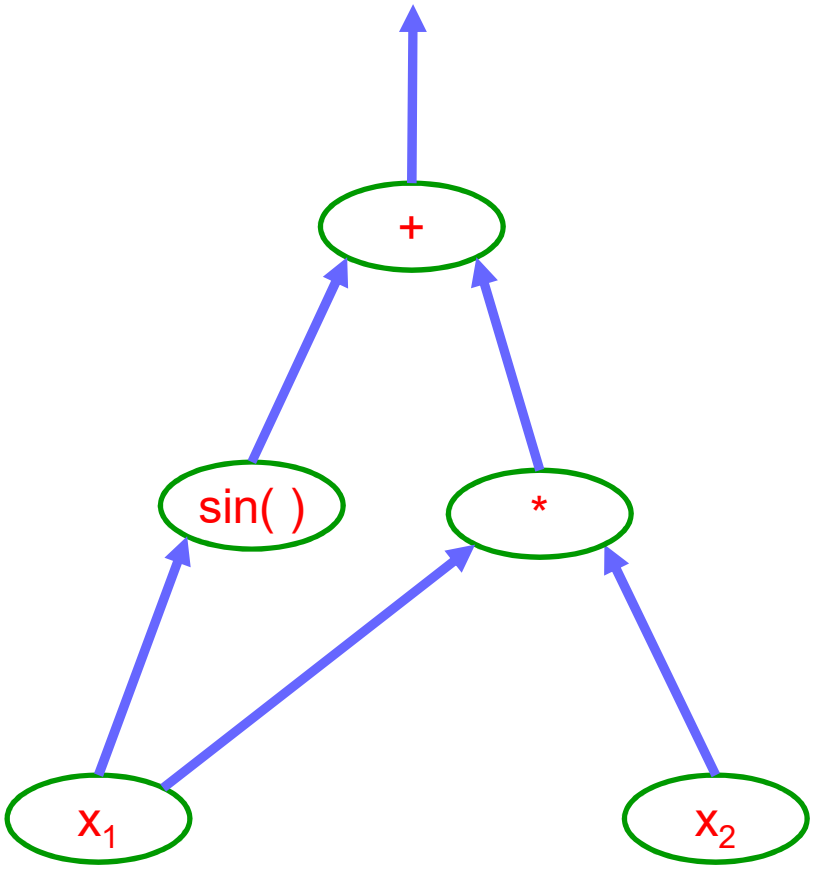
Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$



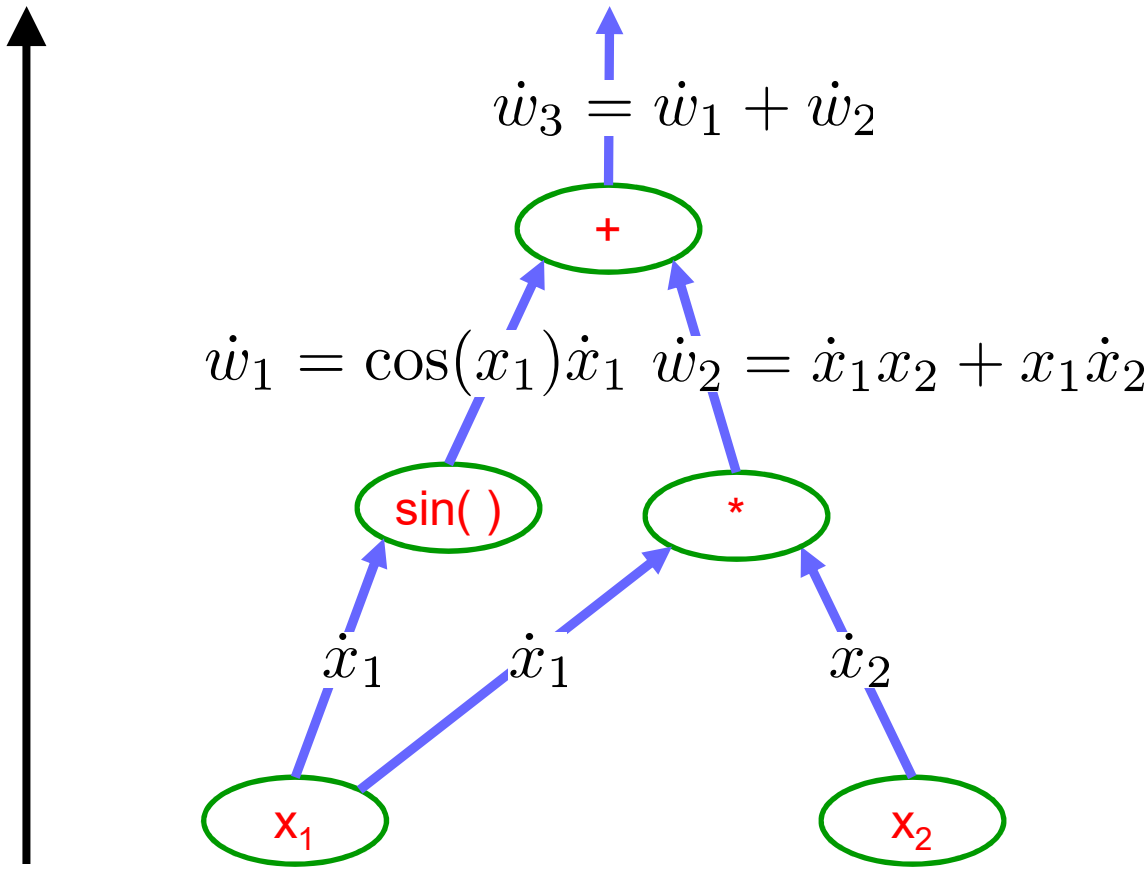
Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$



Example: Forward mode AD

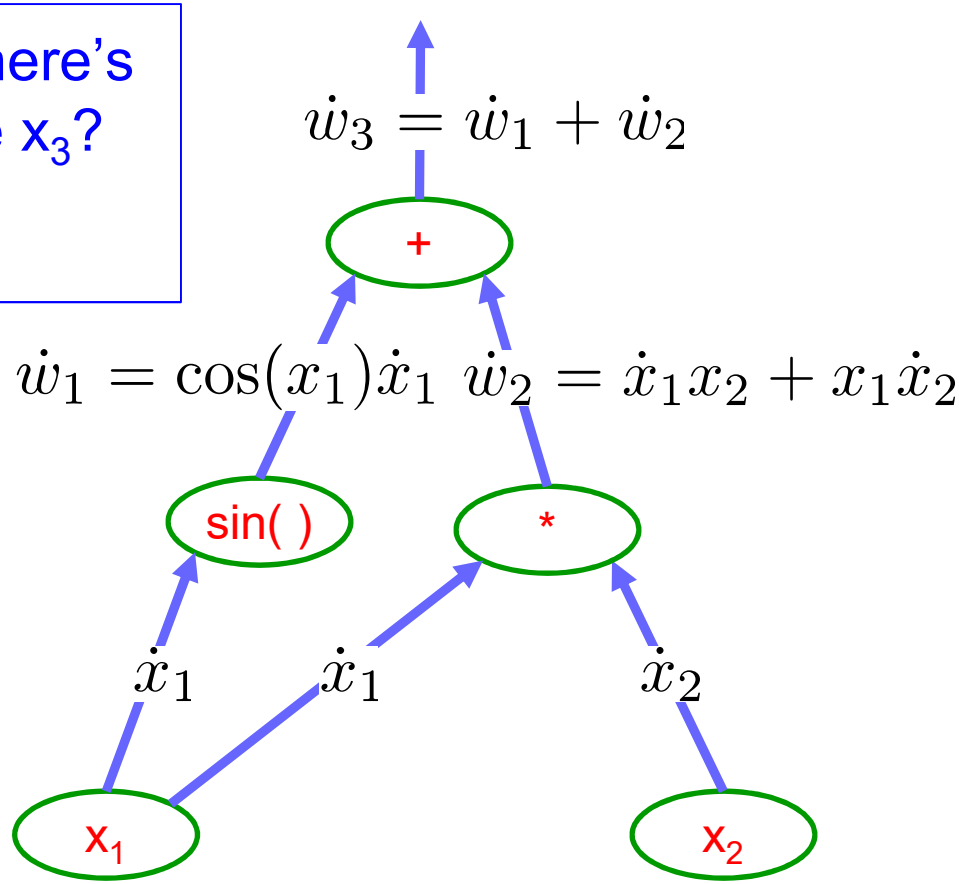
$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$



Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

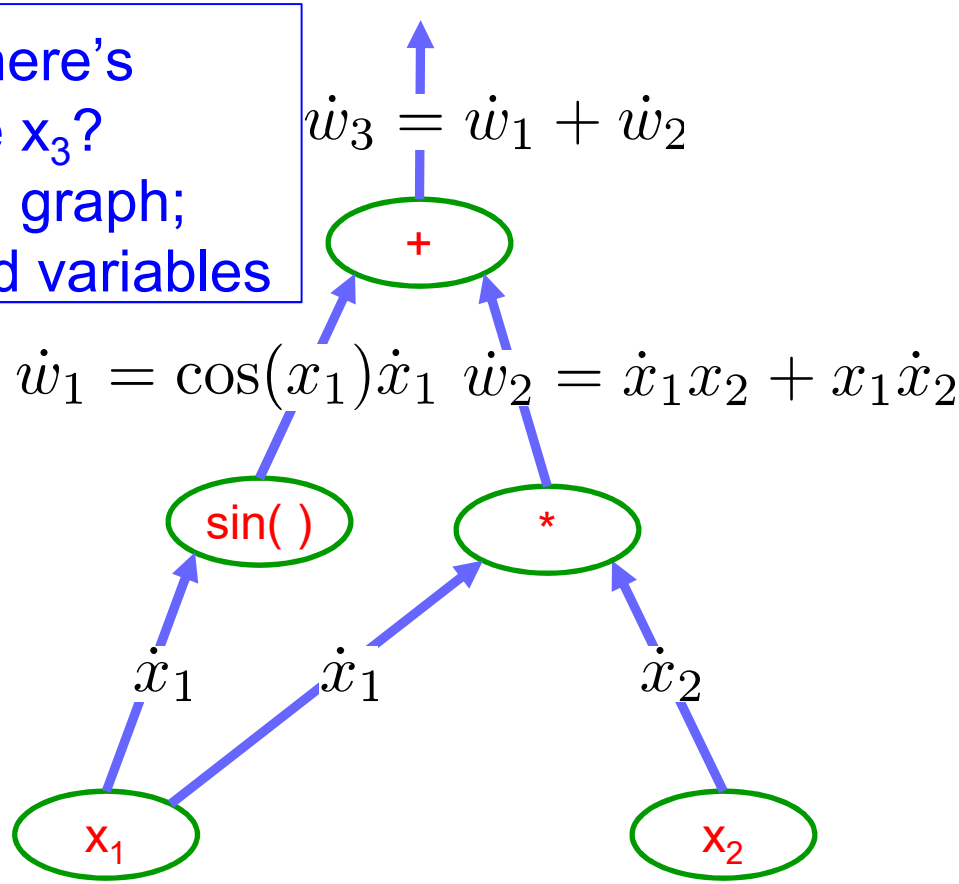
Q: What happens if there's another input variable x_3?



Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

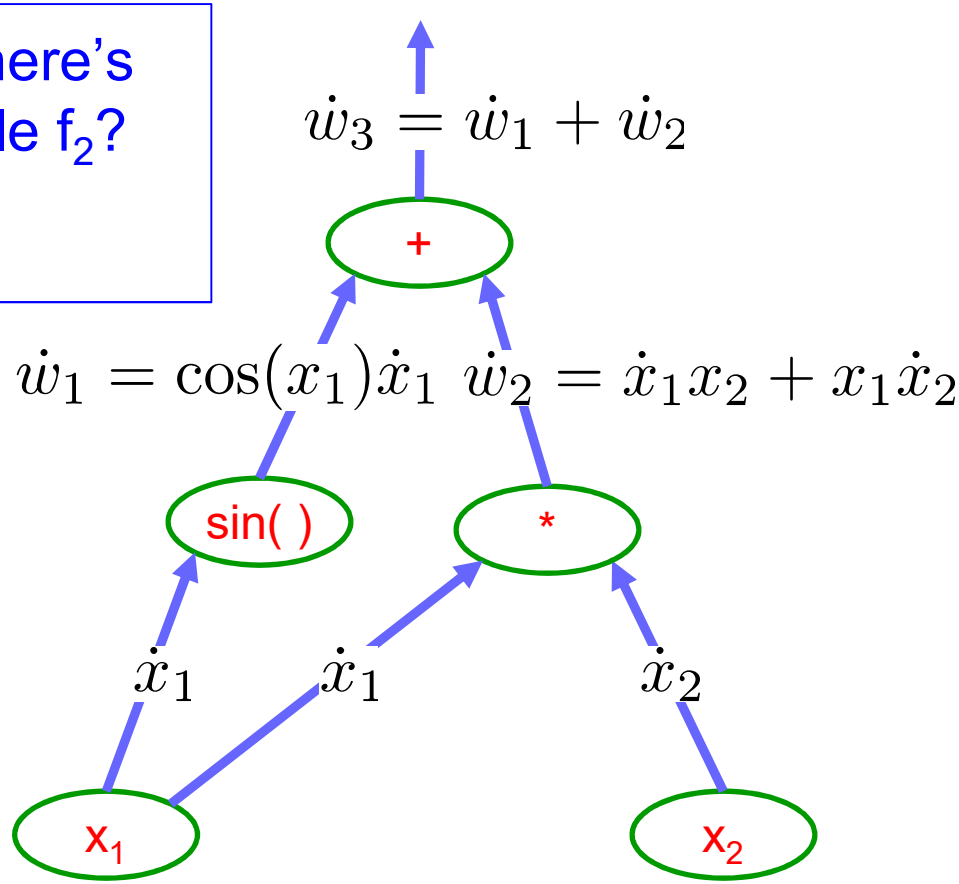
Q: What happens if there's another input variable x_3?
A: more sophisticated graph;
d "forward props" for d variables



Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

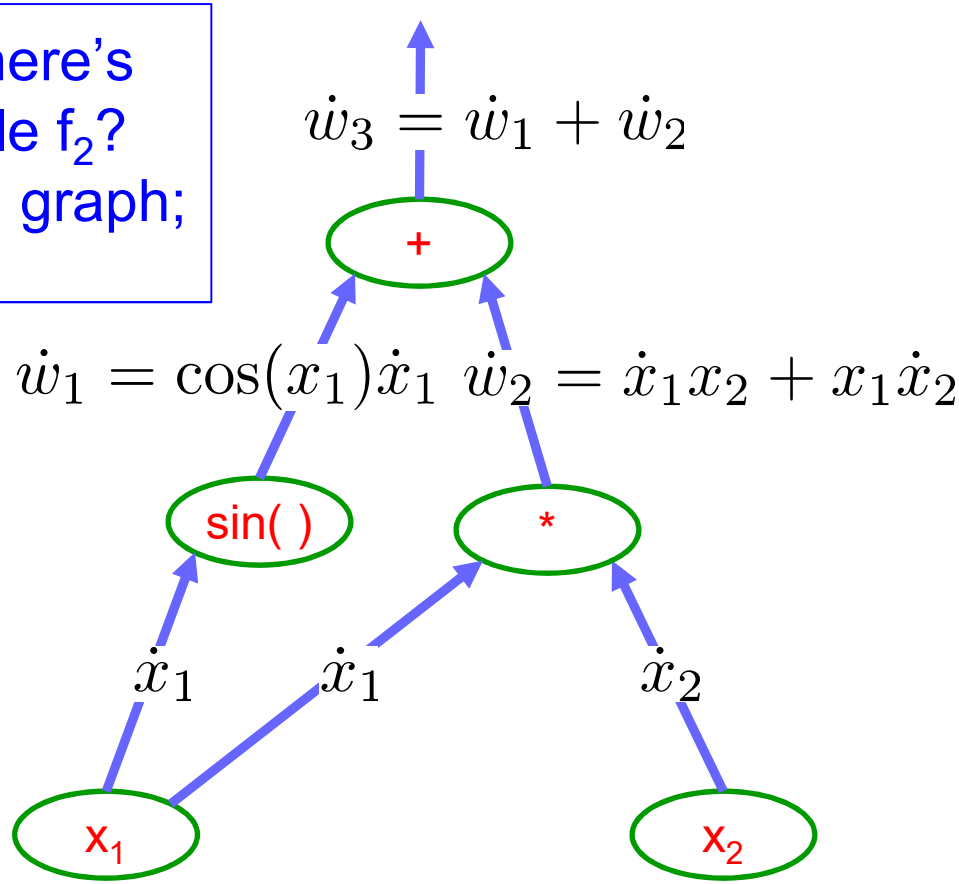
Q: What happens if there's another output variable f_2?



Example: Forward mode AD

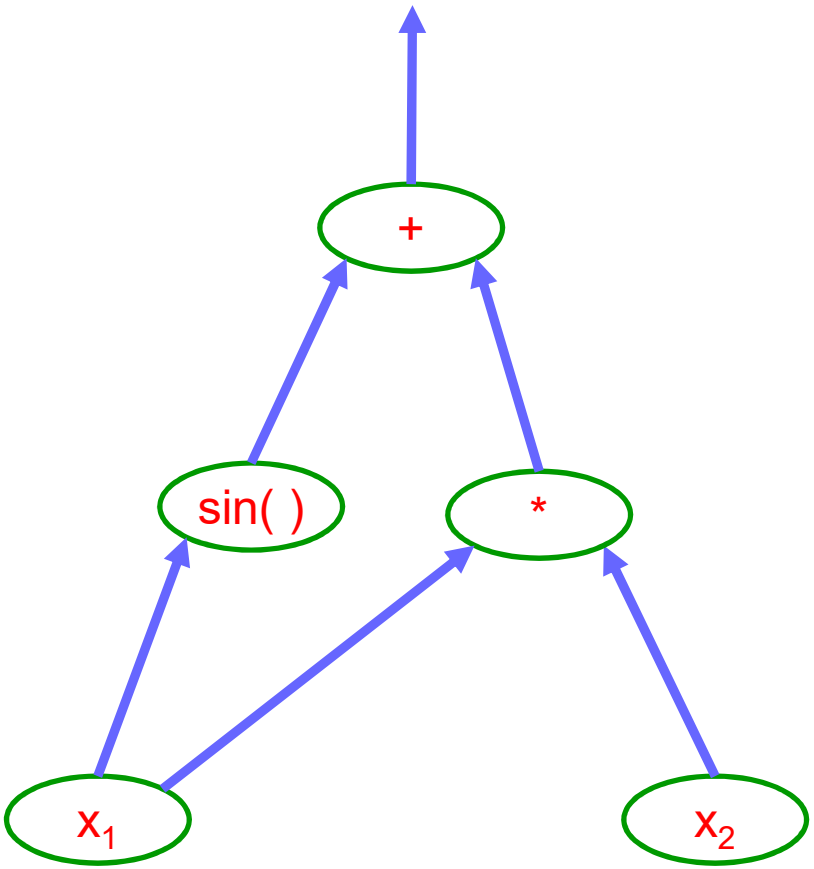
$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

Q: What happens if there's another output variable f_2?
A: more sophisticated graph; single "forward prop"



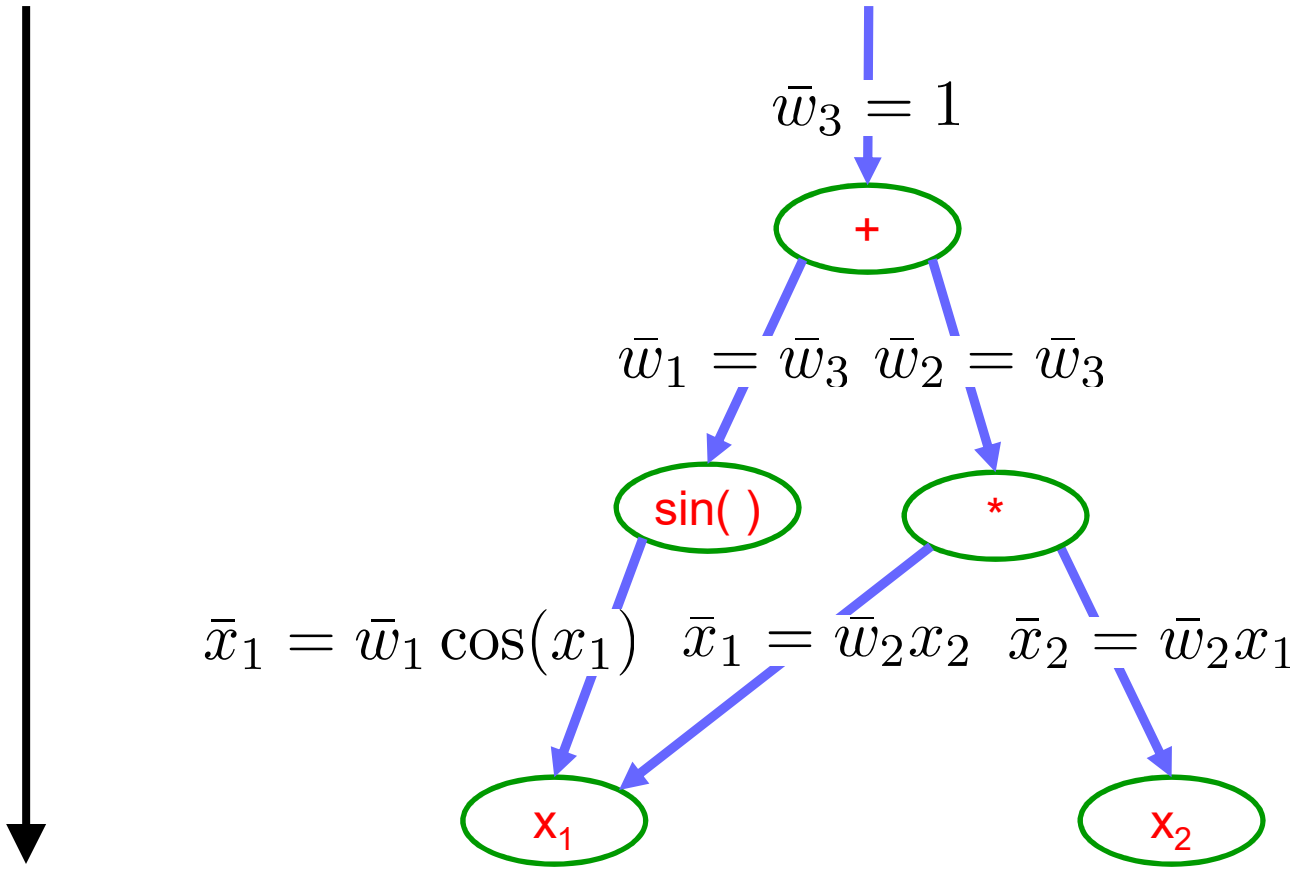
Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

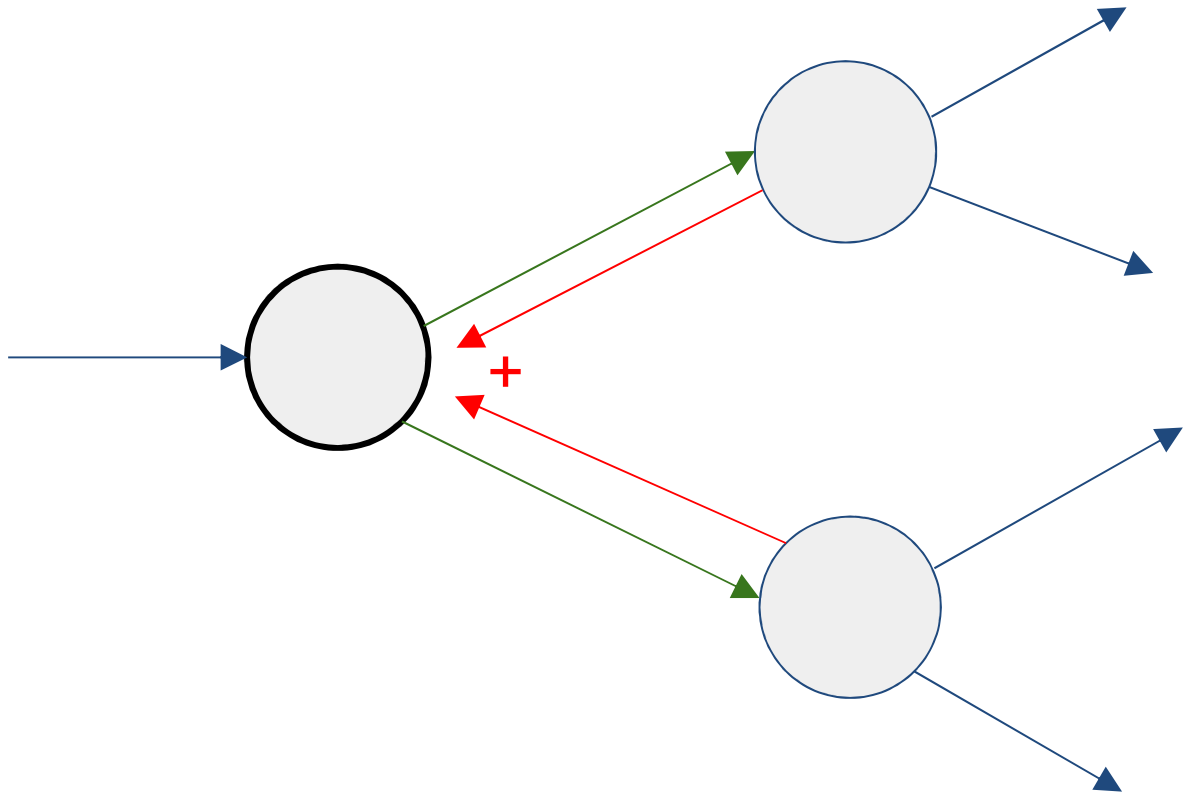


Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$



Gradients add at branches

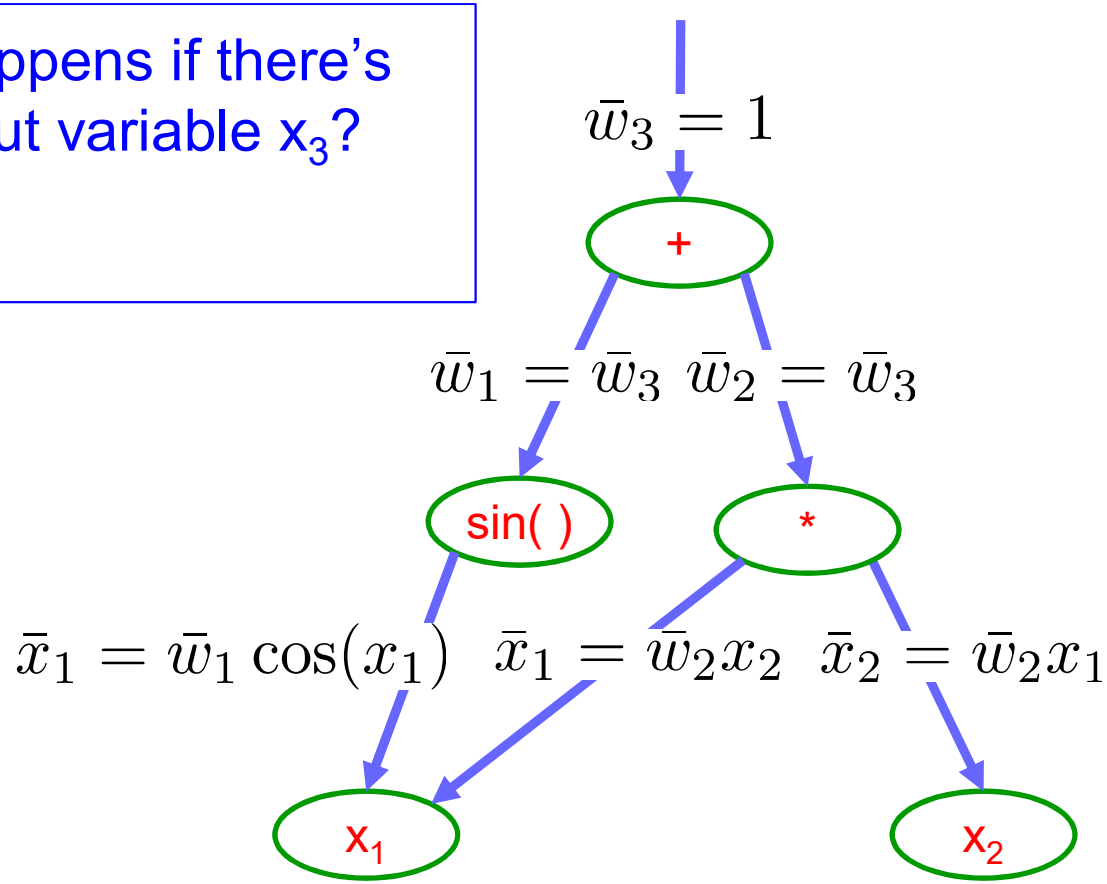


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

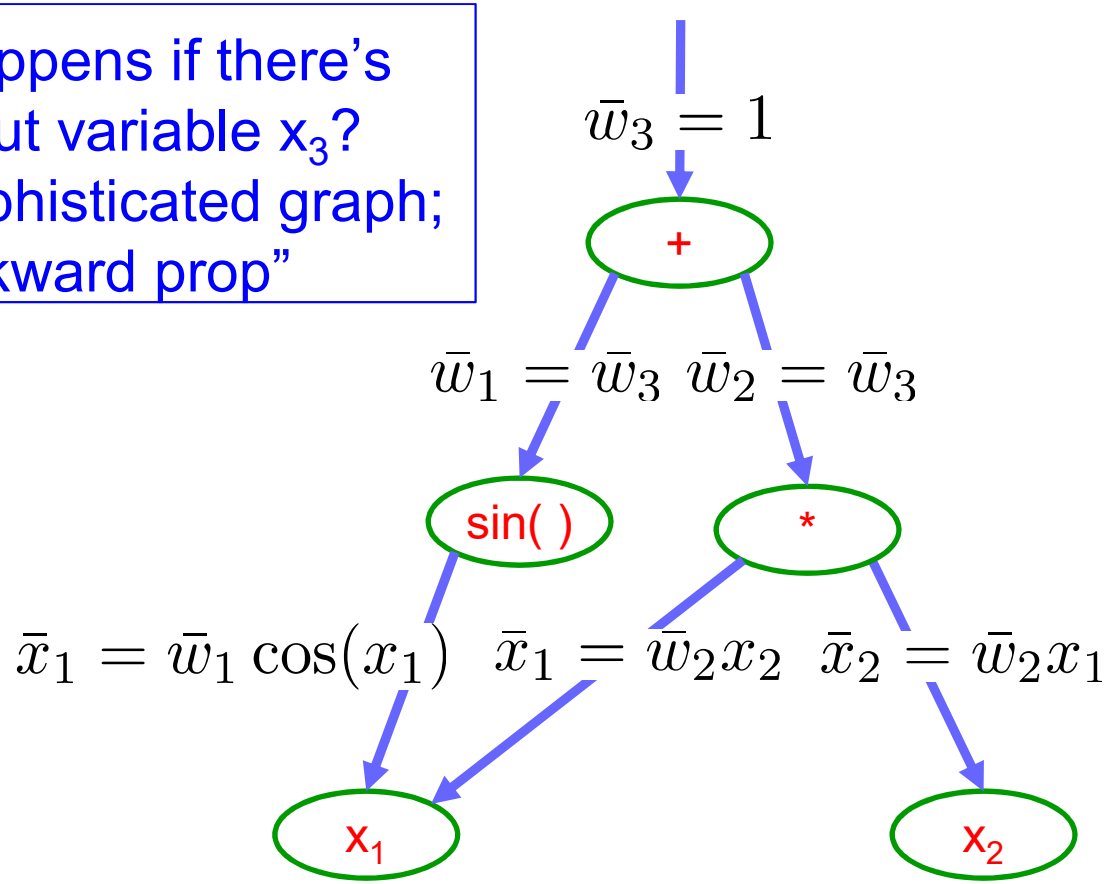
Q: What happens if there's another input variable x_3?



Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

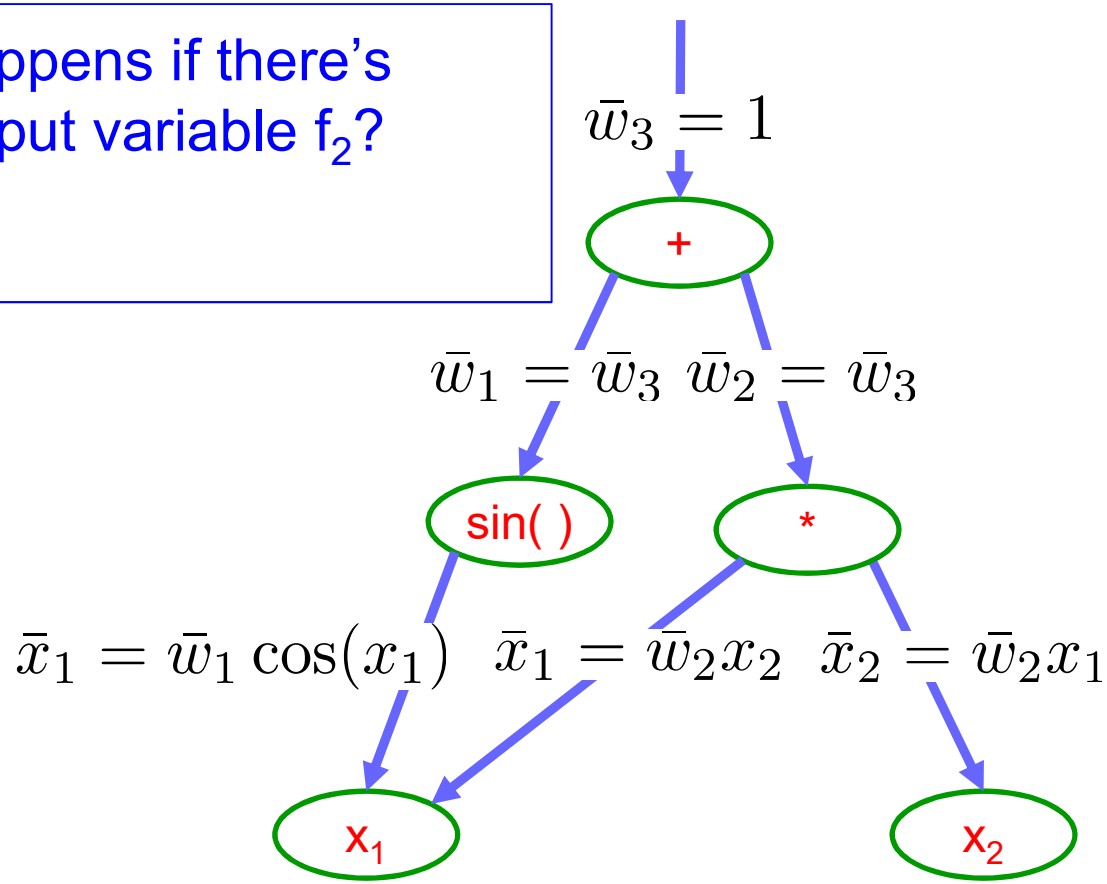
Q: What happens if there's another input variable x_3?
A: more sophisticated graph; single "backward prop"



Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

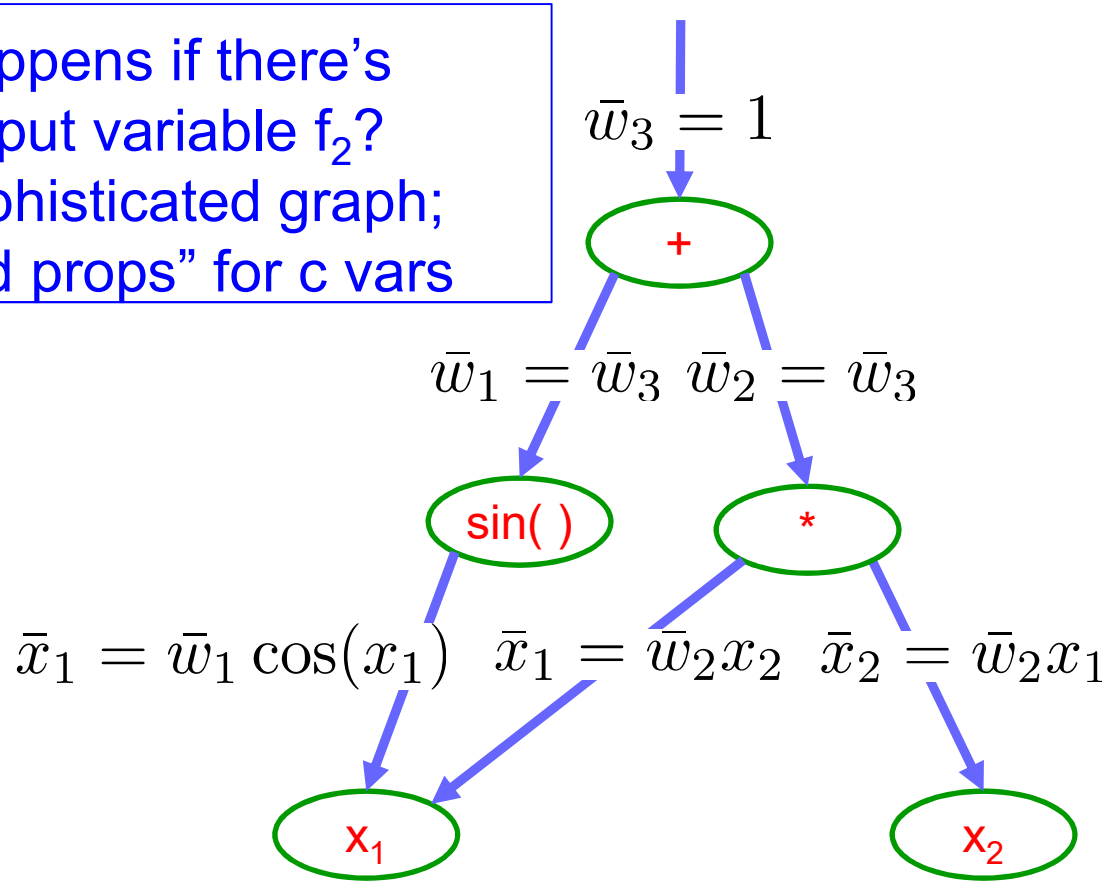
Q: What happens if there's another output variable f_2?



Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

Q: What happens if there's another output variable f_2?
A: more sophisticated graph; c "backward props" for c vars



Forward mode vs Reverse Mode

- x → Graph → L
- Intuition of Jacobian

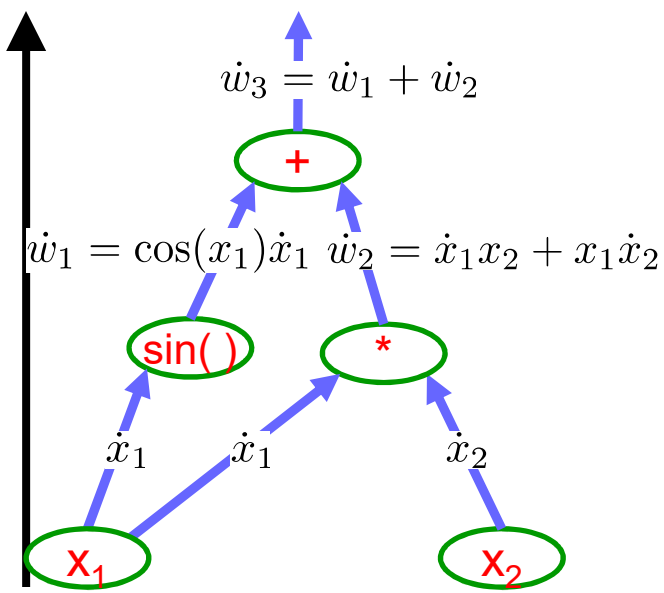
Forward mode vs Reverse Mode

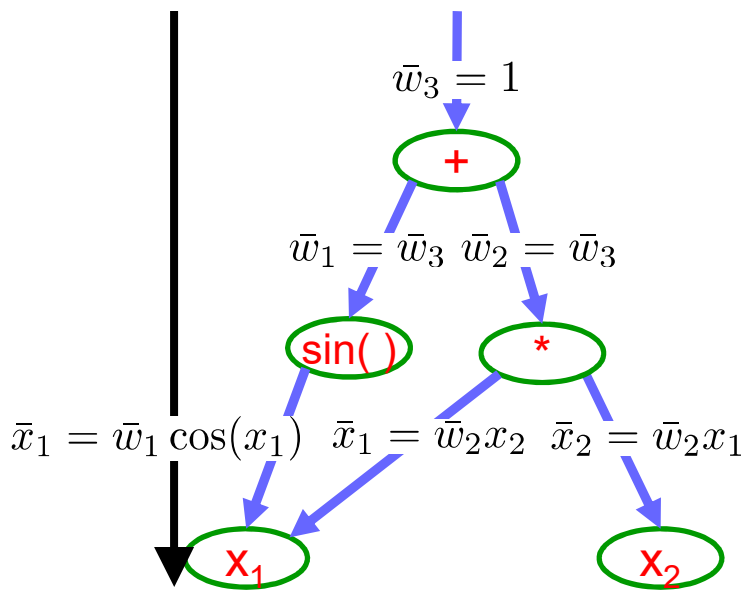
- What are the differences?

- Which one is faster to compute?
  - Forward or backward?

Forward mode vs Reverse Mode

- What are the differences?
- Which one is faster to compute?
  - Forward or backward?
- Which one is more memory efficient (less storage)?
  - Forward or backward?





Practical Note 2: Software Frameworks

Paddle
(Baidu)

Caffe
(UC Berkeley)

Caffe2
(Facebook)

A few weeks ago!

CNTK
(Microsoft)

Torch
(NYU / Facebook)

PyTorch
(Facebook)

MXNet
(Amazon)

Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

Theano
(U Montreal)

TensorFlow +Keras
(Google)

And others...

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

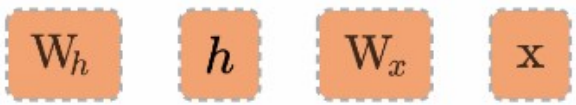
PyTorch

Package	Description
torch	a Tensor library like NumPy, with strong GPU support
torch.autograd	a tape based automatic differentiation library that supports all differentiable Tensor operations in torch
torch.nn	a neural networks library deeply integrated with autograd designed for maximum flexibility
torch.optim	an optimization package to be used with torch.nn with standard optimization methods such as SGD, RMSProp, LBFGS, Adam etc.
torch.multiprocessing	python multiprocessing, but with magical memory sharing of torch Tensors across processes. Useful for data loading and hogwild training.
torch.utils	DataLoader, Trainer and other utility functions for convenience
torch.legacy(.nn/.optim)	legacy code that has been ported over from torch for backward compatibility reasons

A graph is created on the fly

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```



Plan for Today (Cont.)

- Specifying Layers
- Forward & Backward auto-differentiation
- (Beginning of) Convolutional neural networks
  - What is a convolution?
  - FC vs Conv Layers

Recall: Linear Classifier

$$\underset{10\text{x}1}{\boxed{f(x,W)}} = \underset{10\text{x}3072}{\boxed{W}}\,\overset{3072\text{x}1}{\boxed{x}} + \underset{10\text{x}1}{\boxed{b}}$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

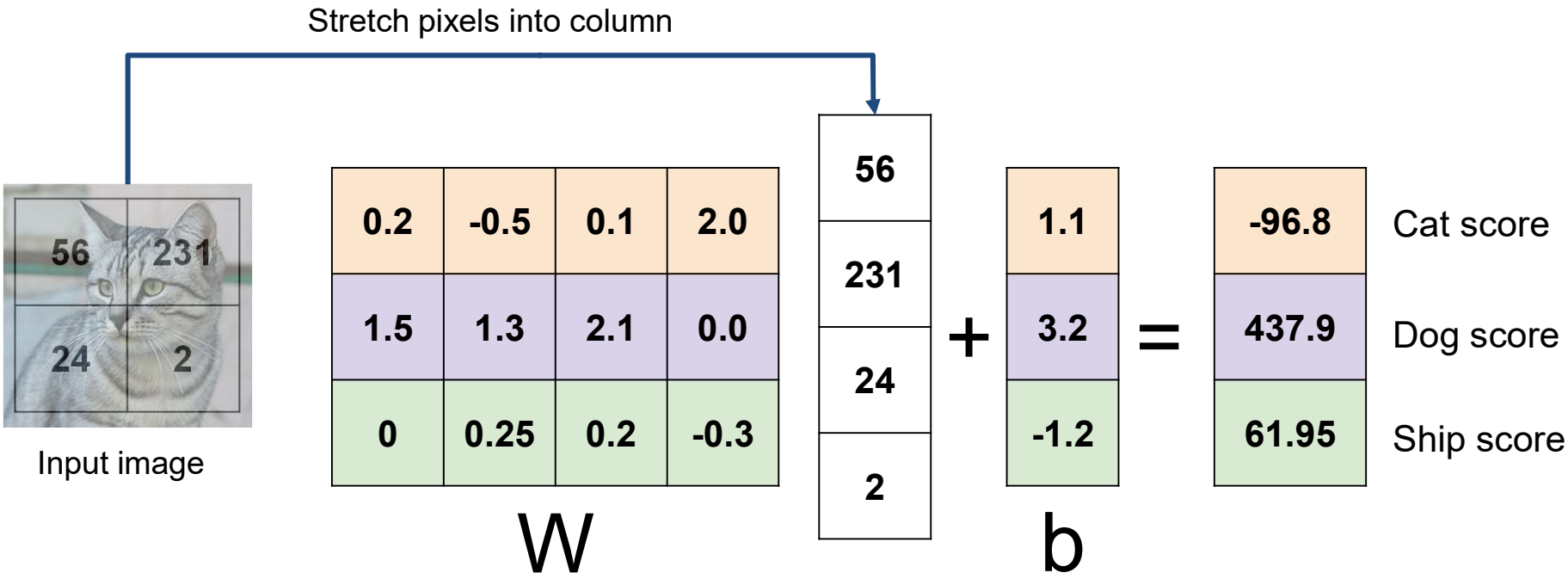
$f(\mathbf{x},\mathbf{W})$

10 numbers giving
class scores

W

parameters
or weights

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

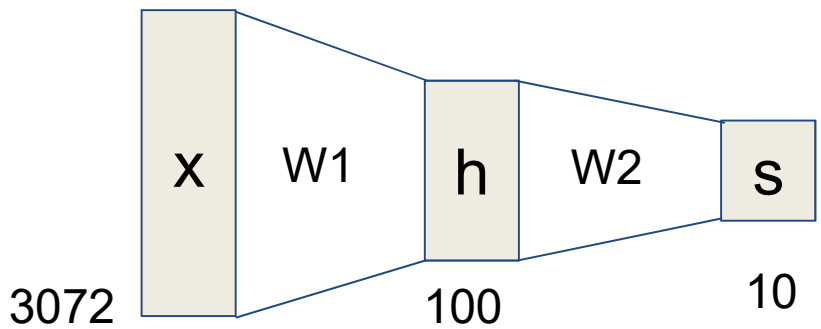


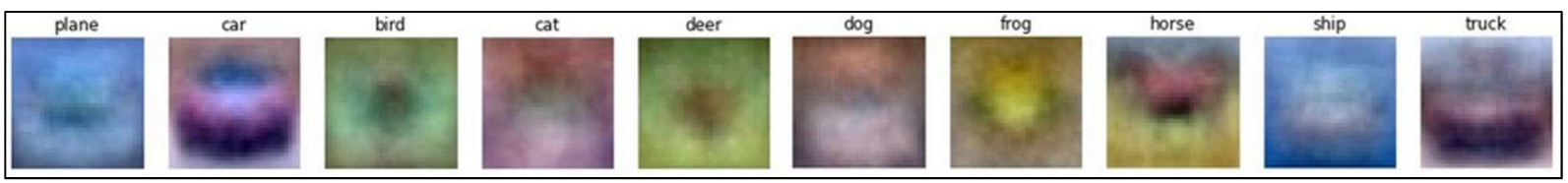
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Recall: (Fully-Connected) Neural networks

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$





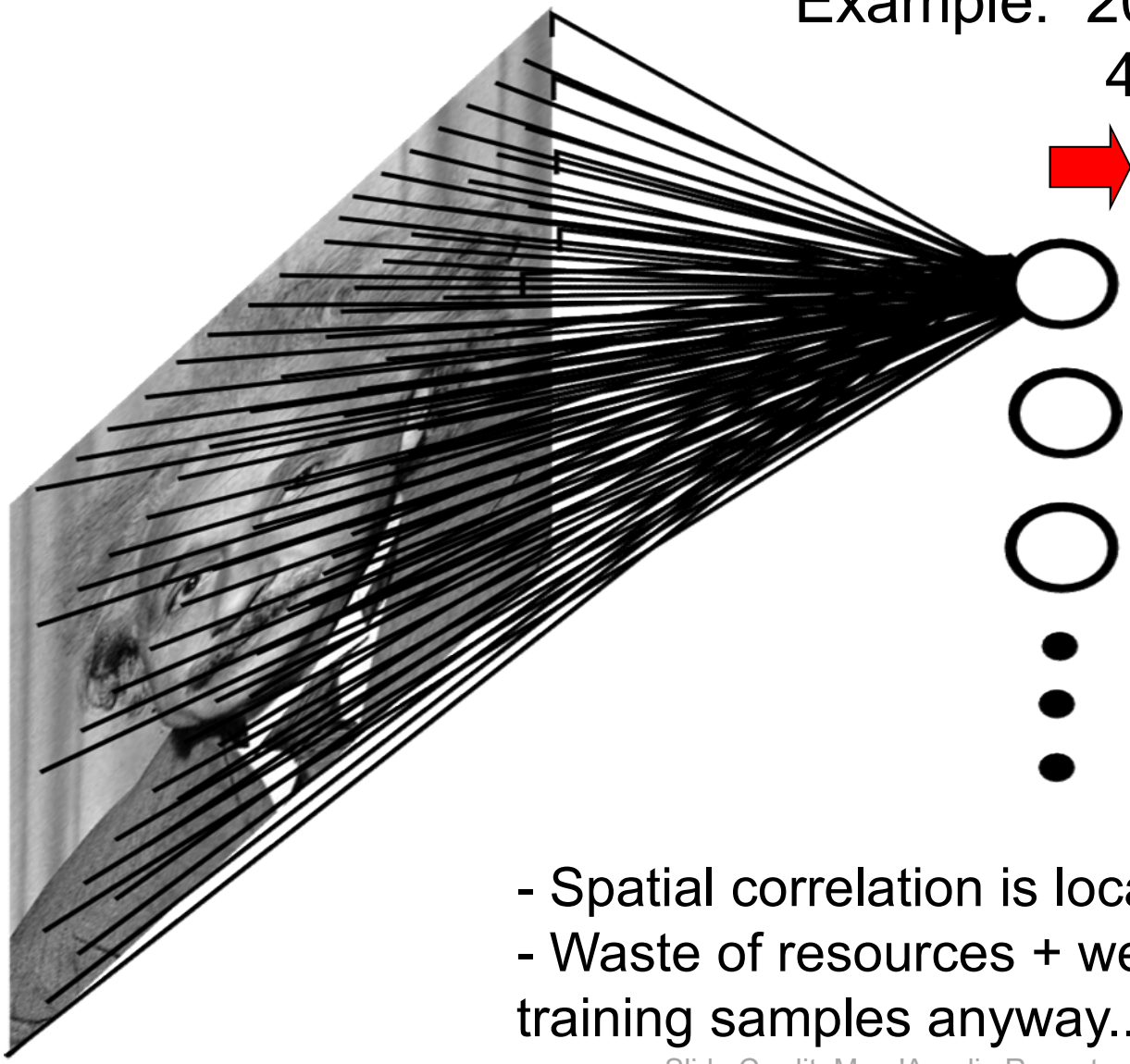
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Convolutional Neural Networks

(without the brain stuff)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Fully Connected Layer



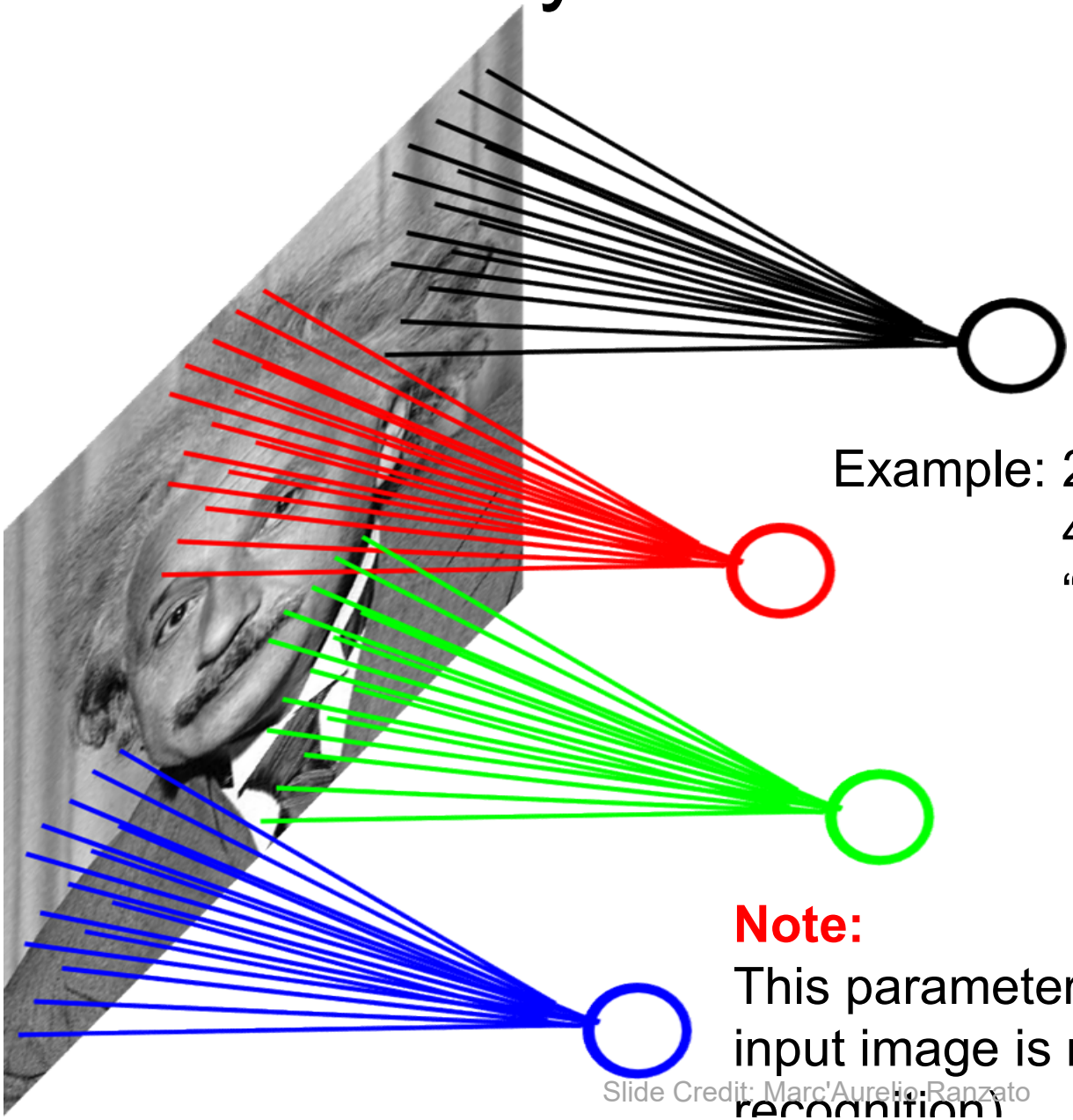
Example: 200x200 image
40K hidden units

~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Slide Credit: Marc'Aurelio Ranzato

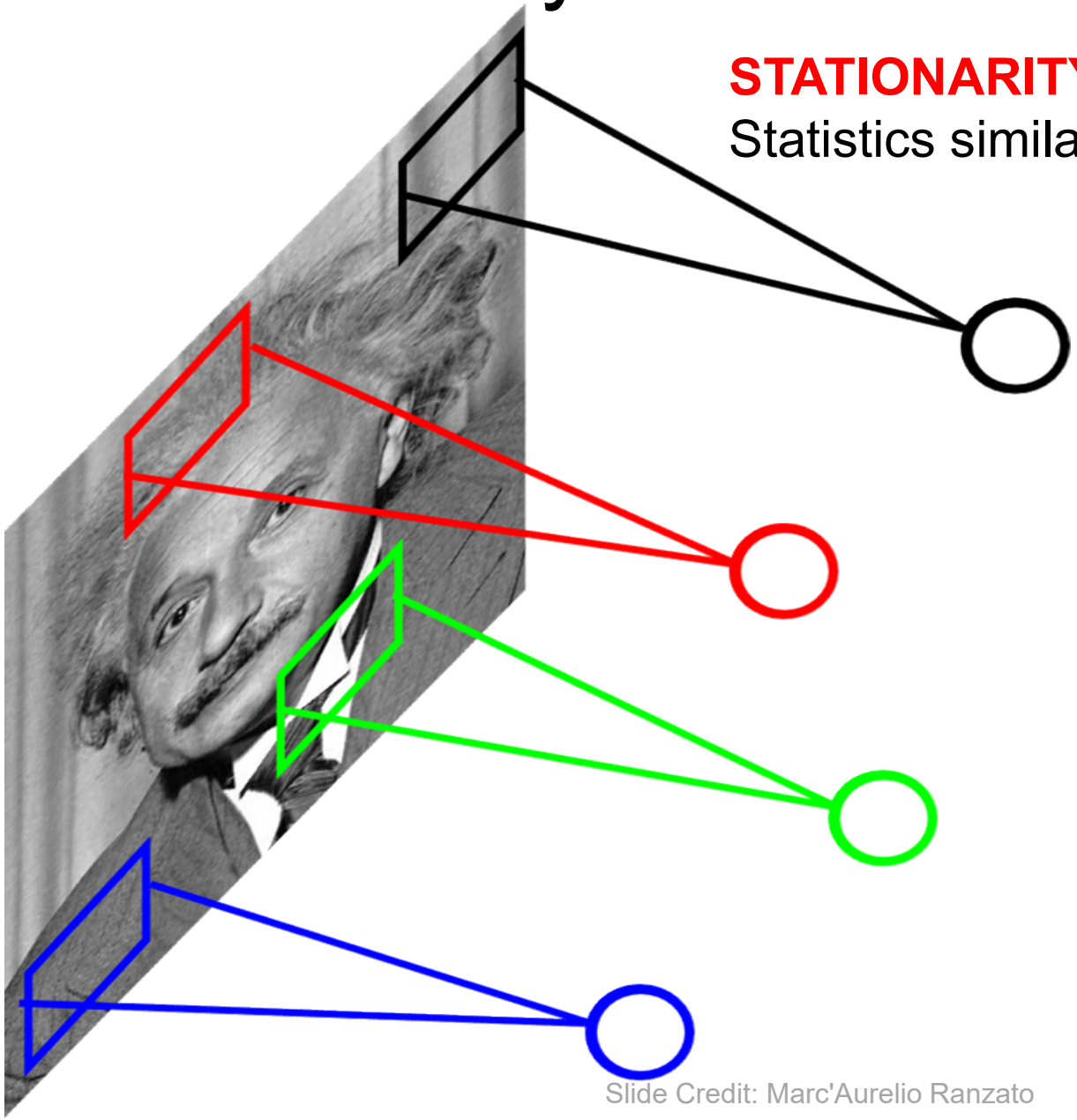
Locally Connected Layer



Example: 200x200 image
40K hidden units
"Filter" size: 10x10
4M parameters

Note:
This parameterization is good when input image is registered (e.g., face recognition)

Locally Connected Layer

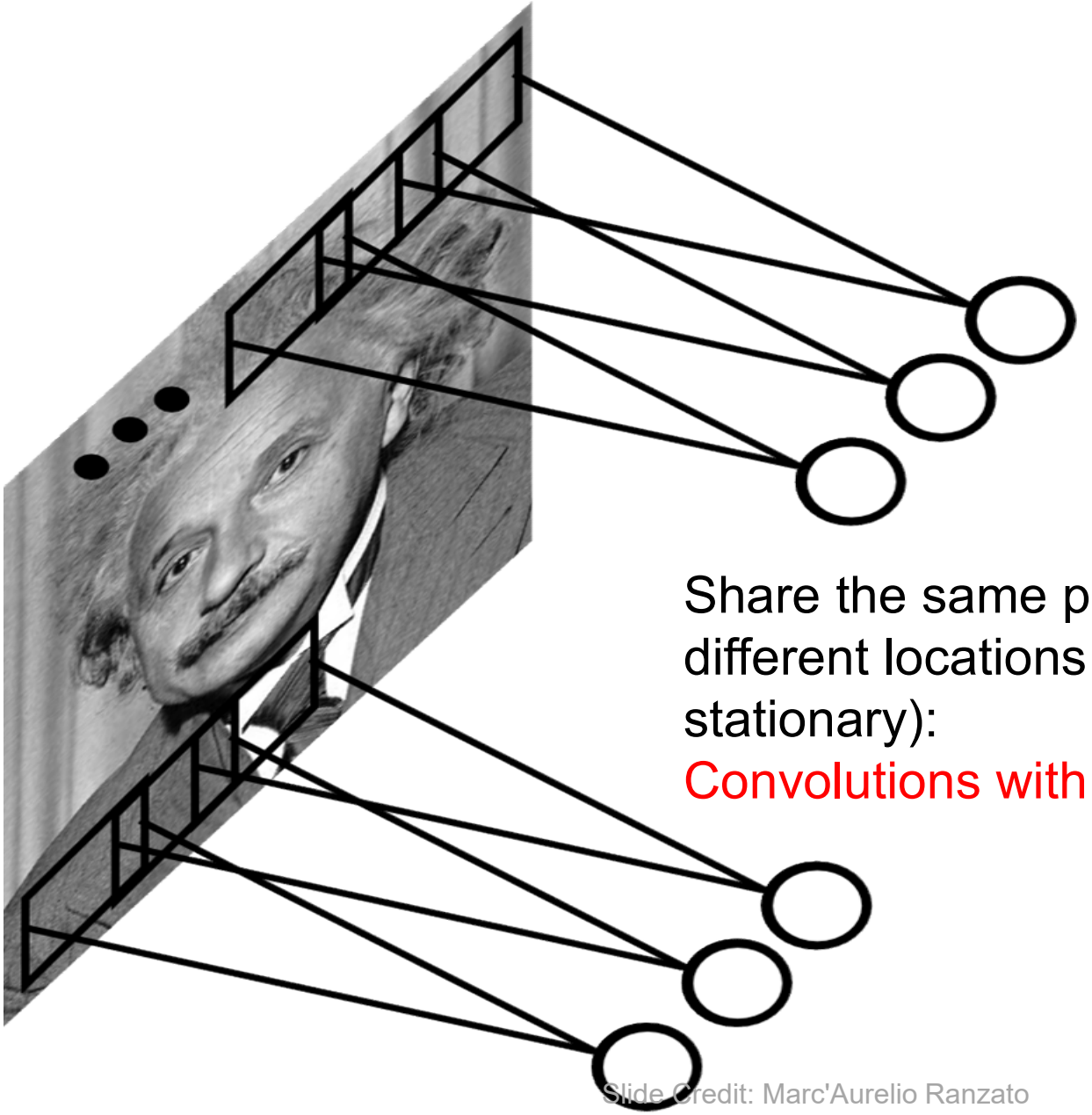


STATIONARITY?

Statistics similar at all locations

Slide Credit: Marc'Aurelio Ranzato

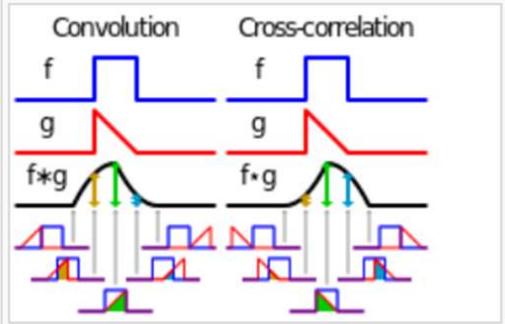
Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

What filter to use?

In mathematics and, in particular, functional analysis, **convolution** is a mathematical operation on two functions *f* and *g*, producing a third function that is typically viewed as a modified version of one of the original functions, giving the area overlap between the two functions as a function of the amount that one of the original functions is translated. Convolution is similar to cross-correlation. It has applications that include probability, statistics, computer vision, image and signal processing, electrical engineering, and differential equations.



Visual comparison of convolution, cross-correlation

Discrete convolution

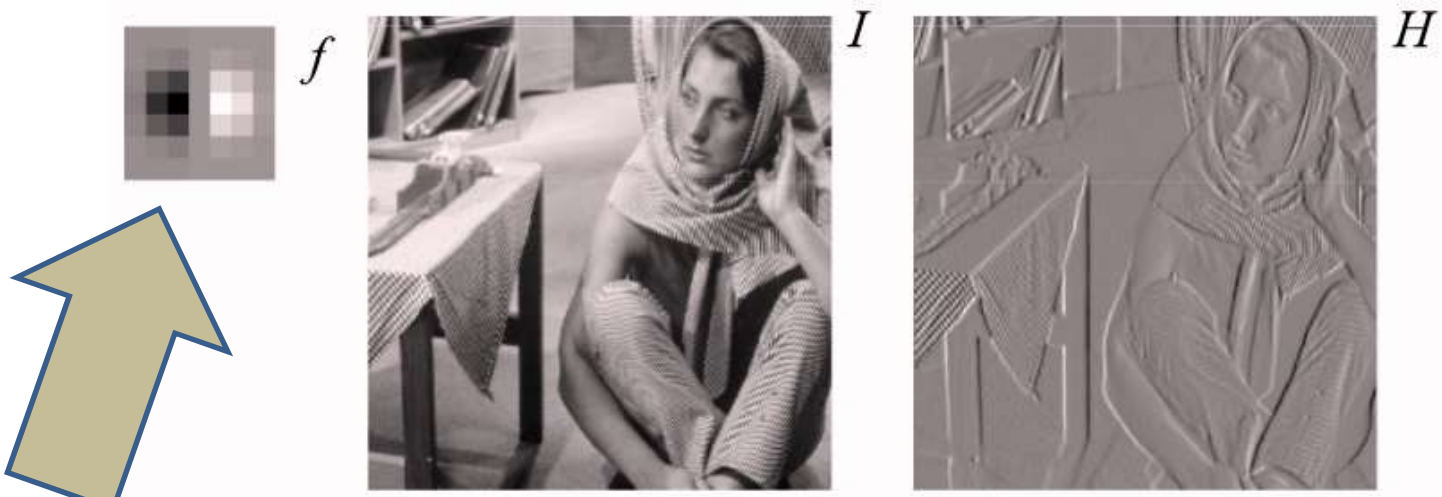
- Discrete Convolution!
  - Very similar to correlation but associative $F \otimes (G \otimes I) = (F \otimes G) \otimes I$

1D Convolution

$$y_k = \sum_{n=0}^{N-1} h_n \cdot x_{k-n}$$

$$\begin{aligned} y_0 &= h_0 \cdot x_0 \\ y_1 &= h_1 \cdot x_0 + h_0 \cdot x_1 \\ y_2 &= h_2 \cdot x_0 + h_1 \cdot x_1 + h_0 \cdot x_2 \\ y_3 &= h_2 \cdot x_1 + h_1 \cdot x_2 + h_0 \cdot x_3 \\ &\vdots \end{aligned}$$

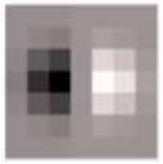
2D Convolution



Filter

$$H[m,n] = f \otimes I = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

A note on sizes



m

m

Filter

N

N

Input

N-*m*+1

N-*m*+1

Output

MATLAB to the rescue!

- conv2(x,w, ‘valid’)

Convolutions!

- Math vs. CS vs. programming viewpoints

Convolutions for mathematicians

- On operation on two functions x and w to produce a third function y
- E.g. input $x(t)$ and kernel or weighting function $w(\tau)$

$$
\begin{aligned}
y(t) &= (x * w)(t) \\
&= (w * x)(t) \\
&= \int_{\tau=-\infty}^{\infty} x(t-\tau)w(\tau)d\tau \\
&= \int_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau)d\tau
\end{aligned}
$$

Convolutions for mathematicians

- One dimension

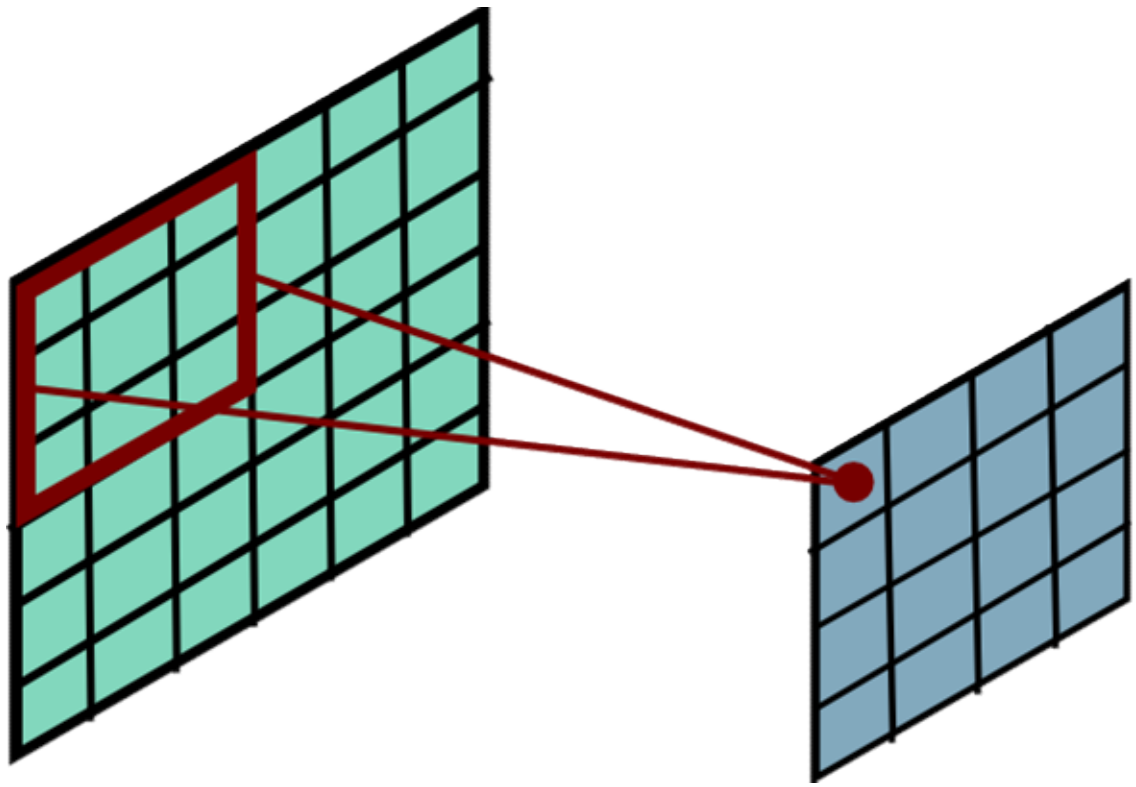
$$y(t) = (x * w)(t)$$
$$= \int_{\tau=-\infty}^{\infty} x(t-\tau)w(\tau)d\tau$$

- Two dimensions

$$y(t_1, t_2) = (x * w)(t_1, t_2)$$
$$= \int_{\tau=-\infty}^{\infty} \int_{\sigma=-\infty}^{\infty} x(t_1 - \tau, t_2 - \sigma)w(\tau, \sigma)\, d\tau d\sigma$$

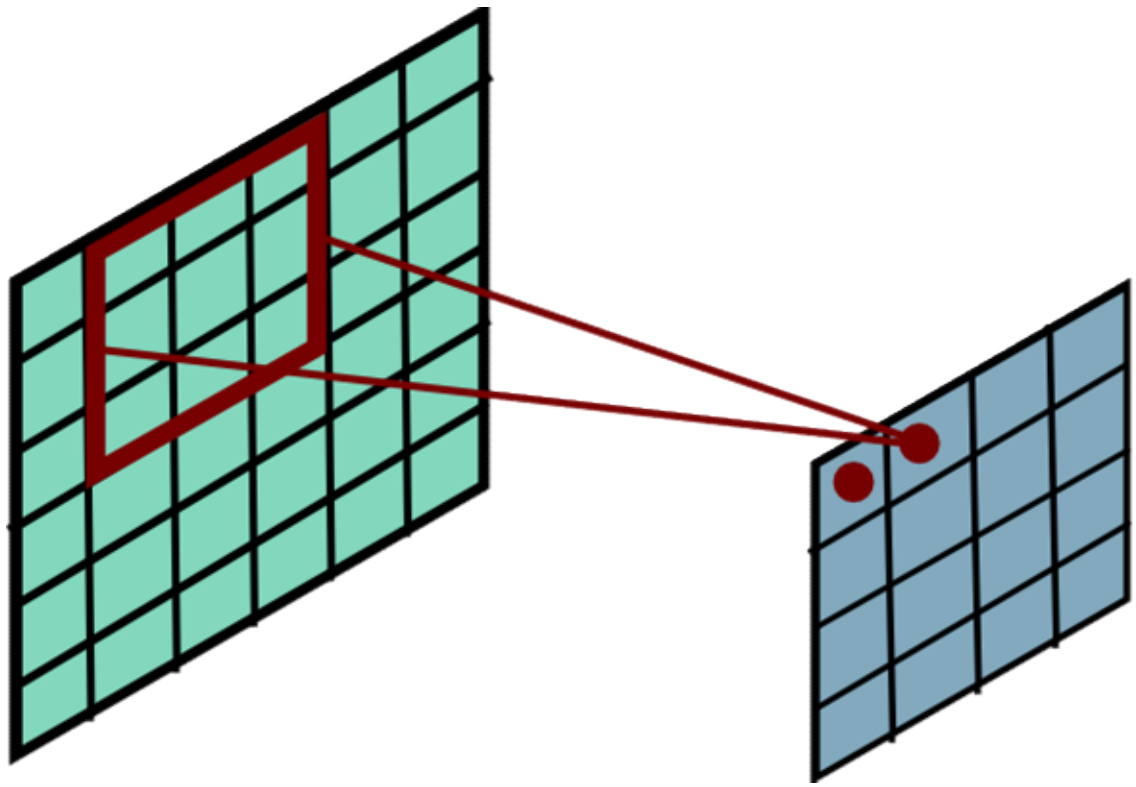
Convolutions for CS/Programmers

Convolutional Layer



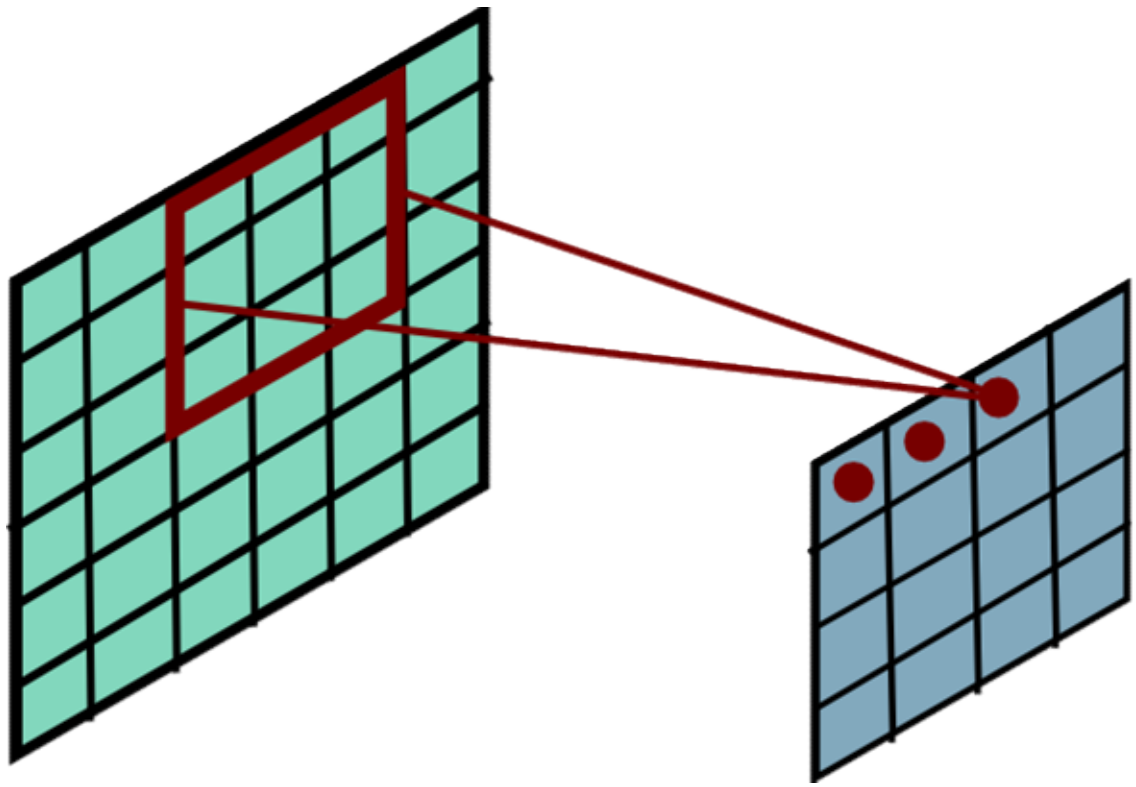
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



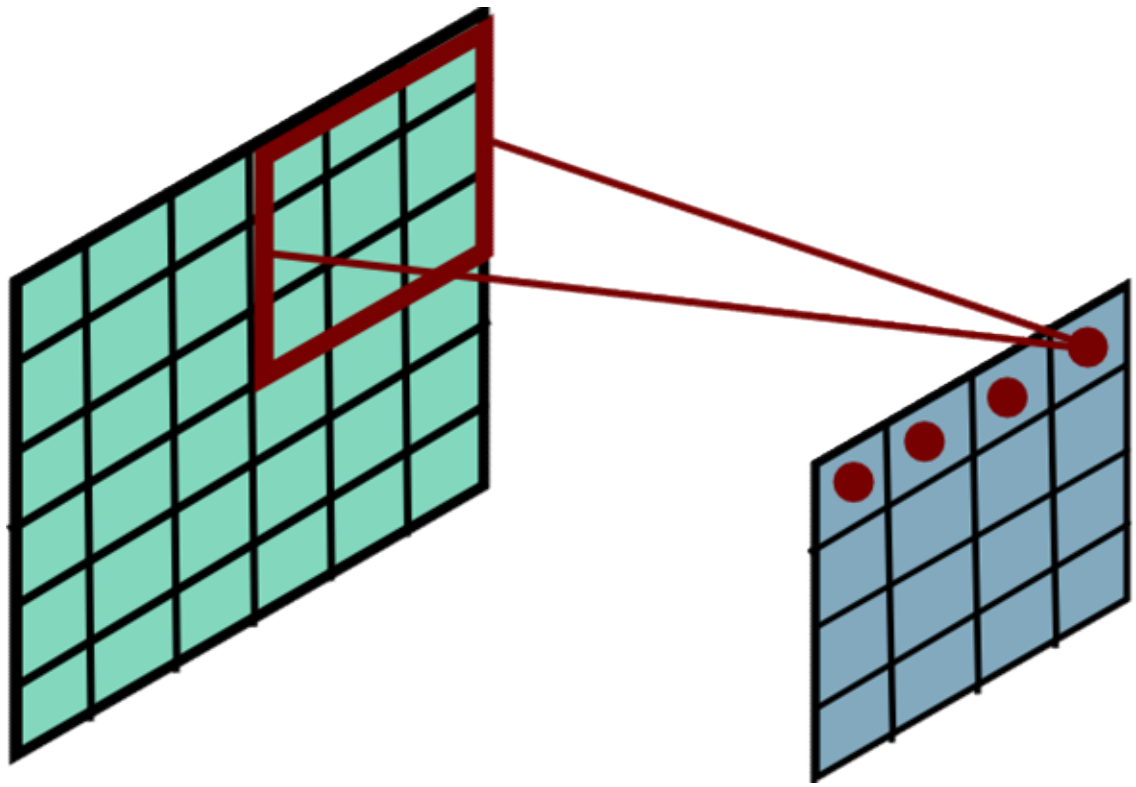
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



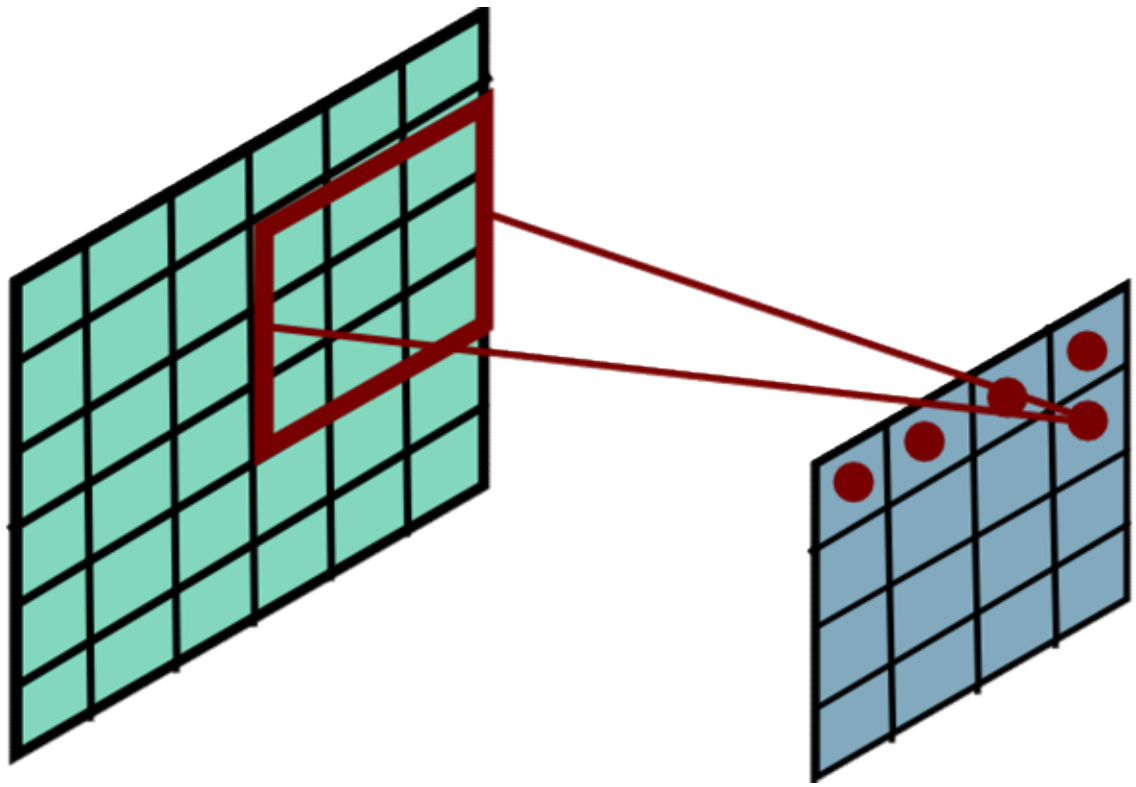
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



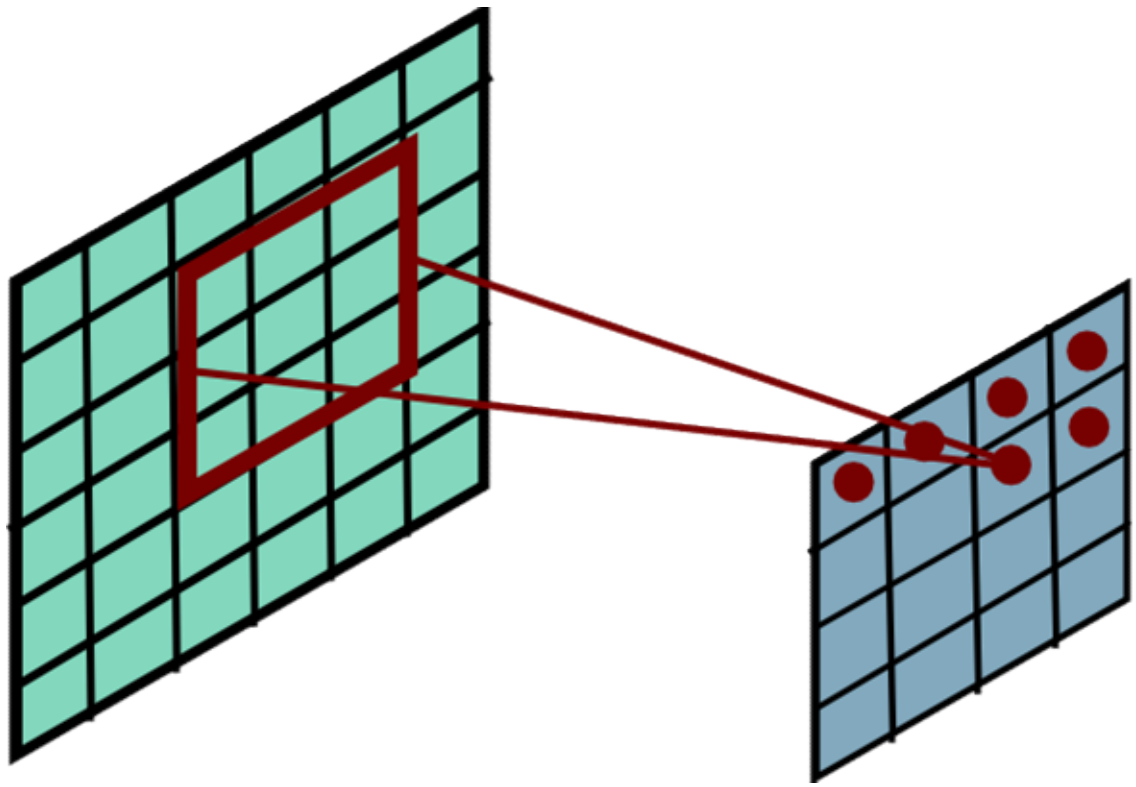
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



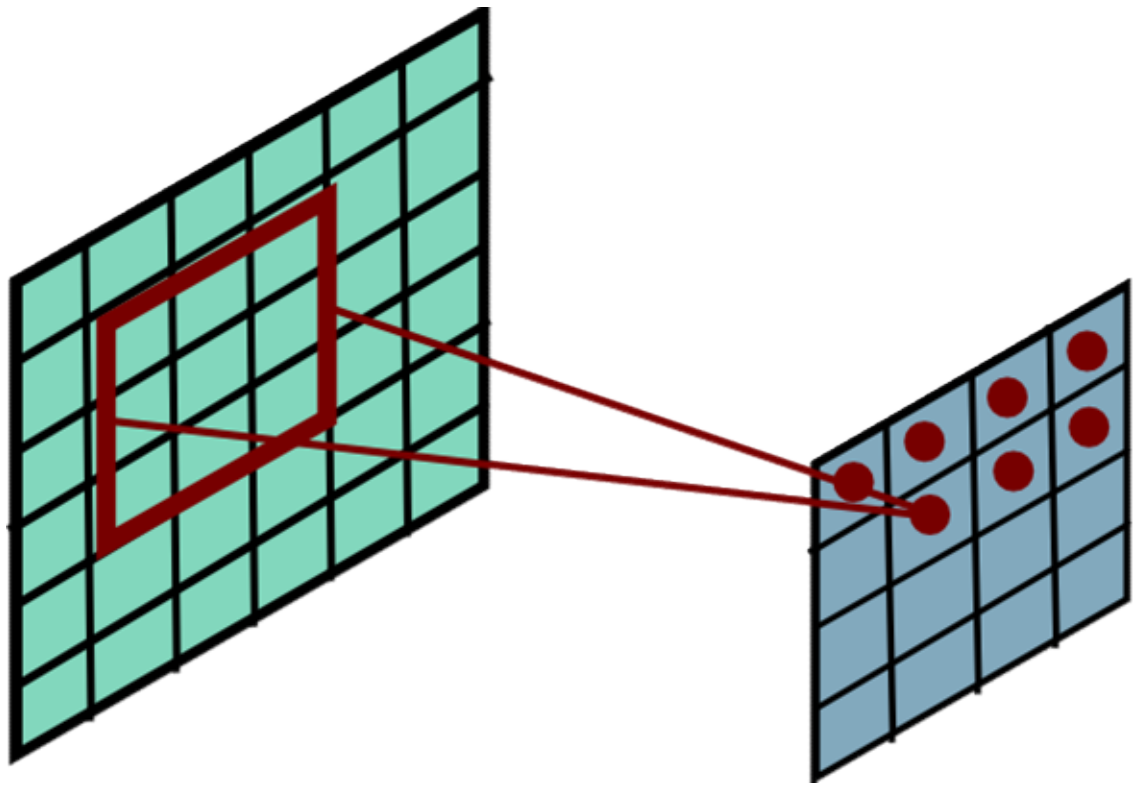
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



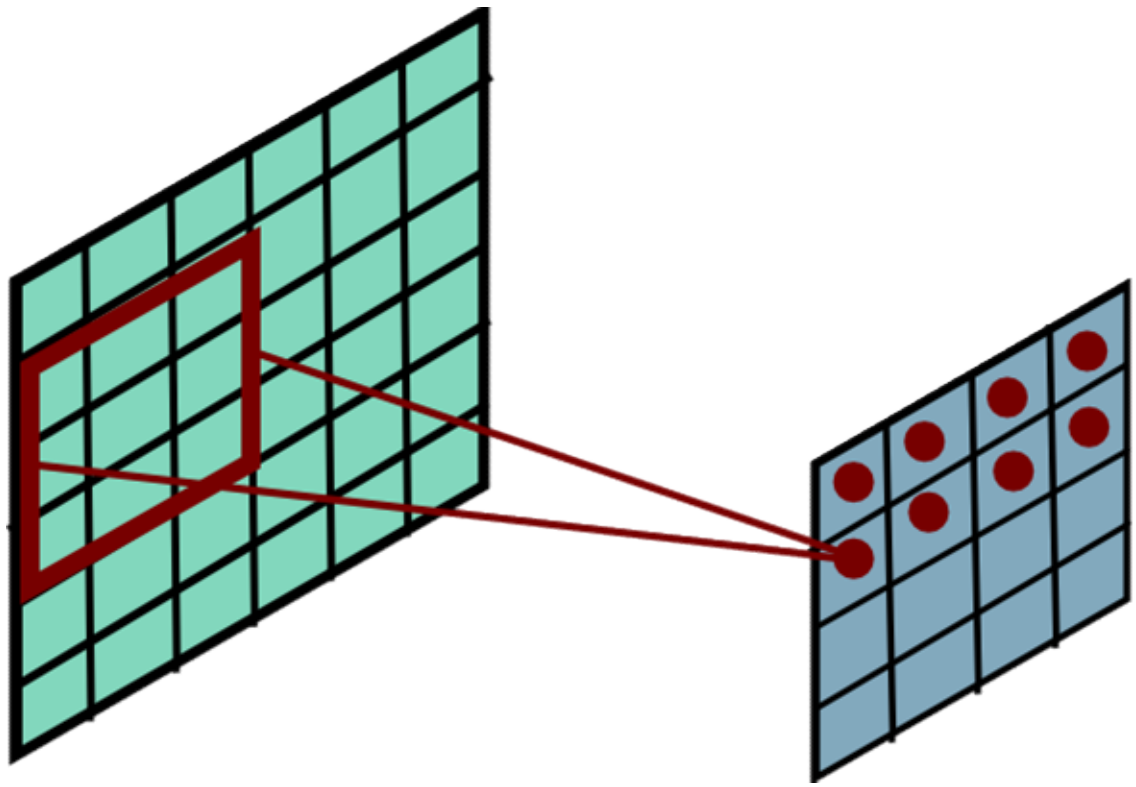
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



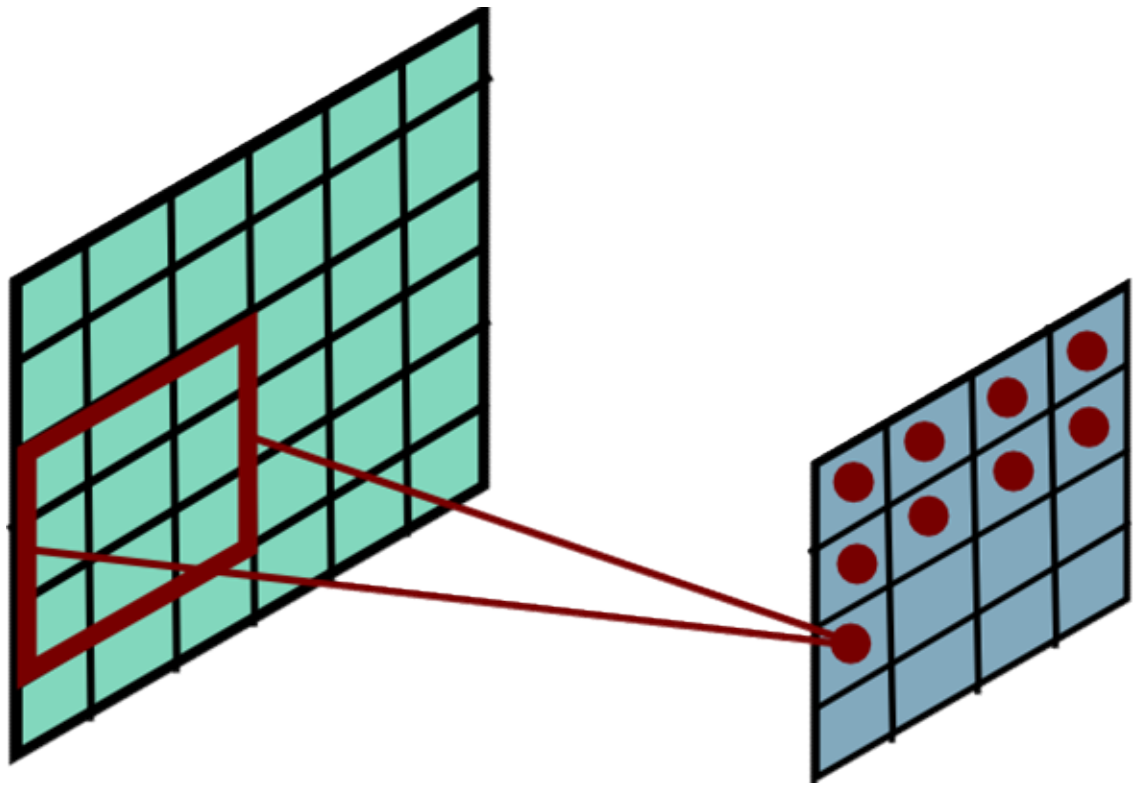
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



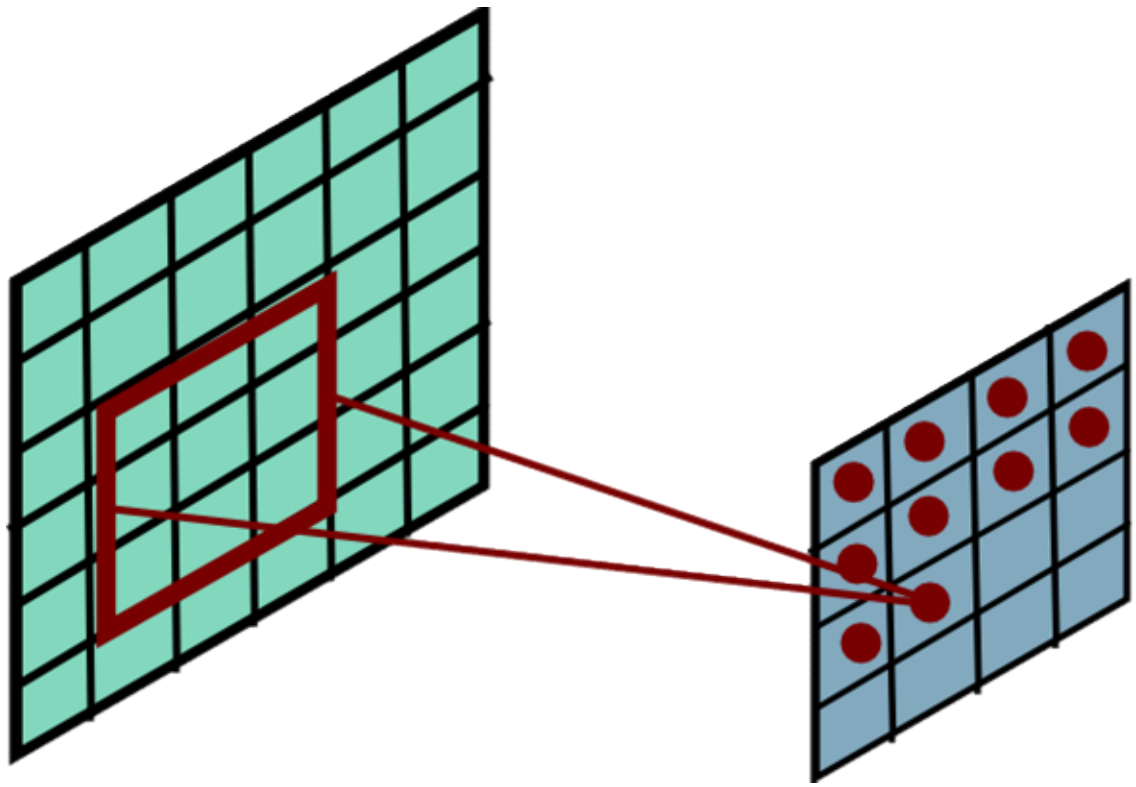
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



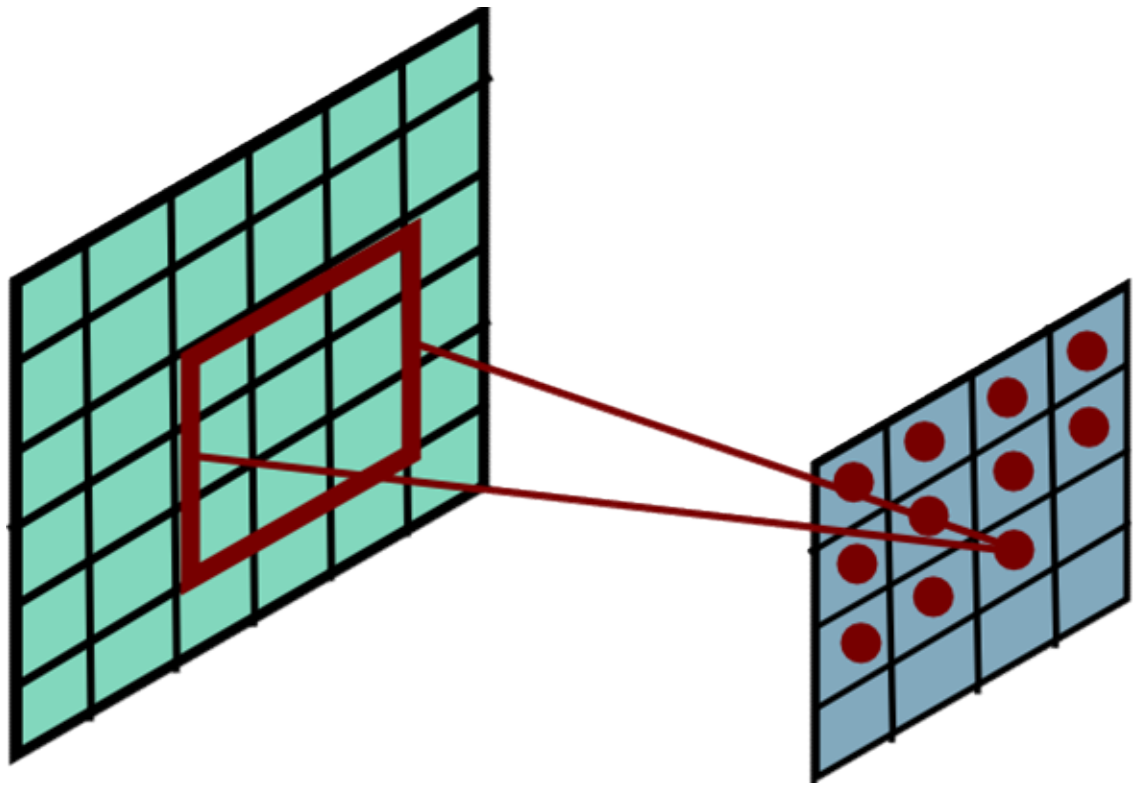
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



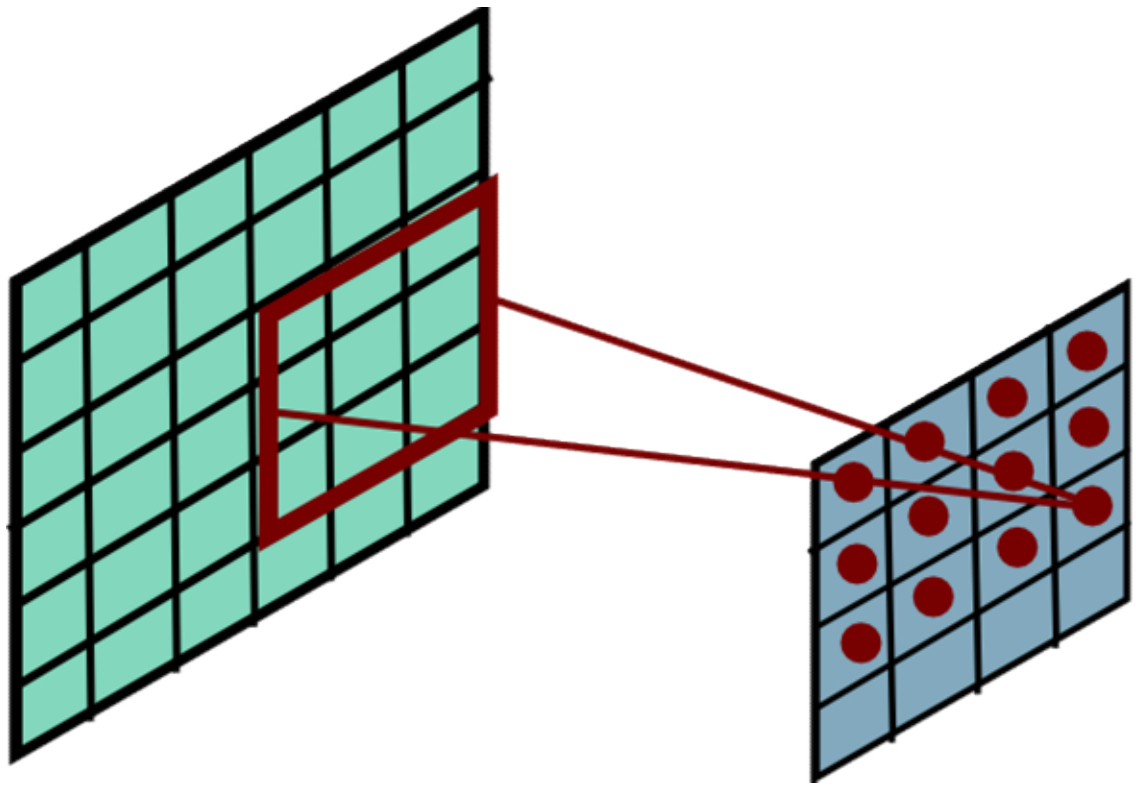
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



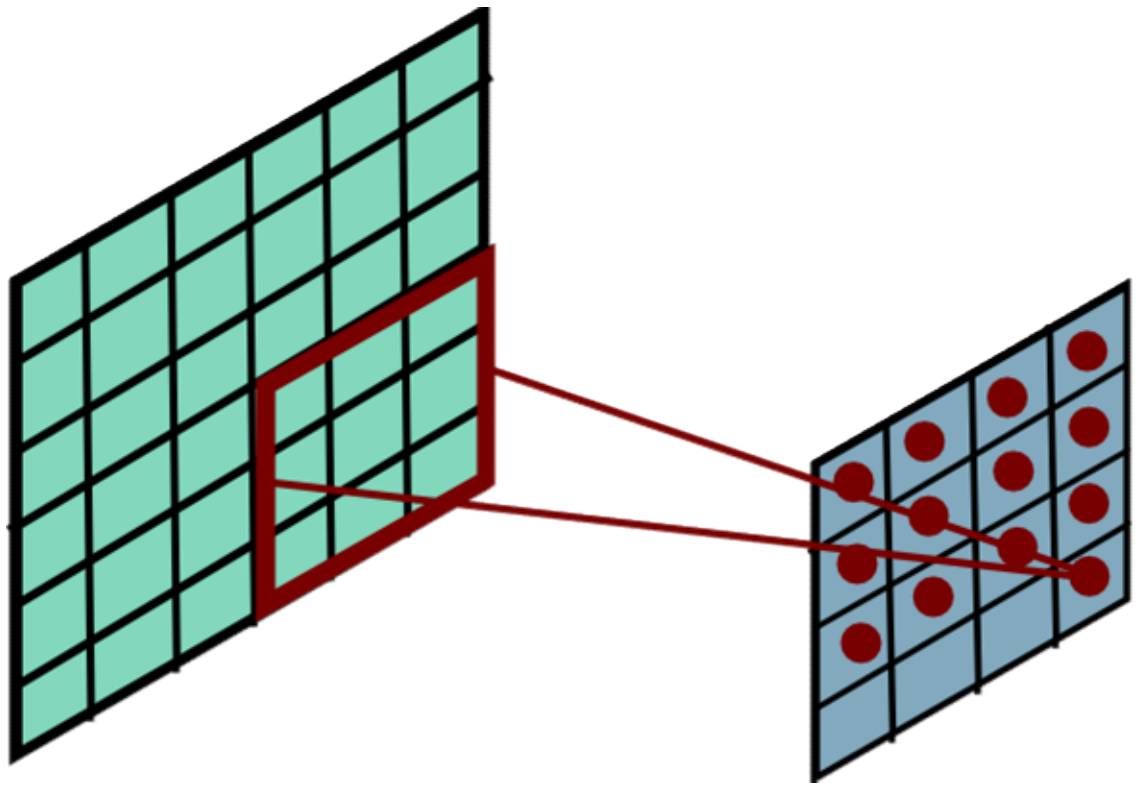
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



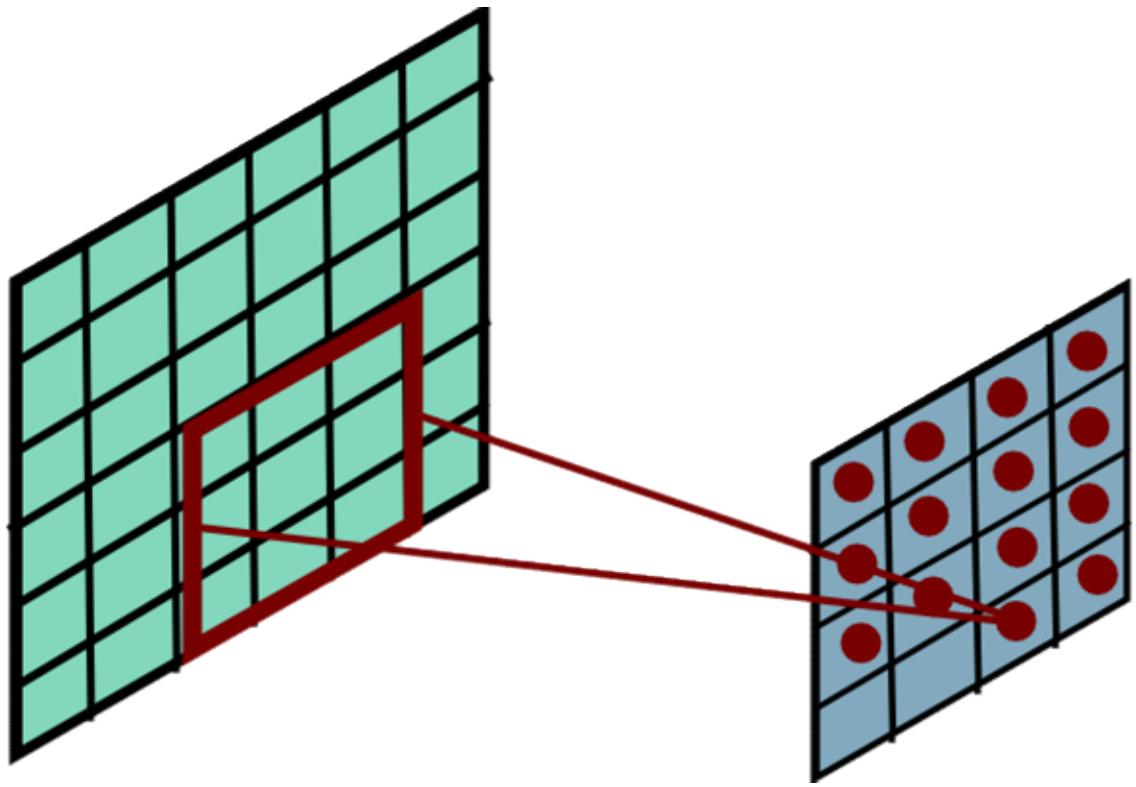
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



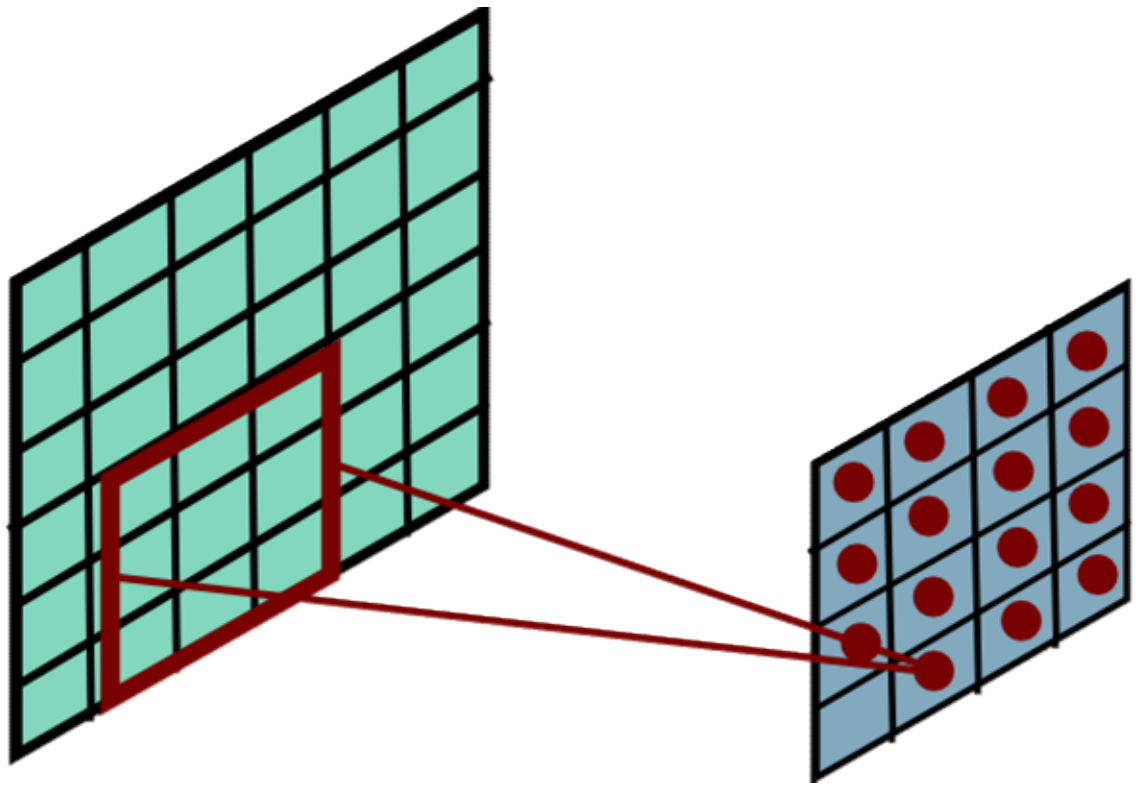
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



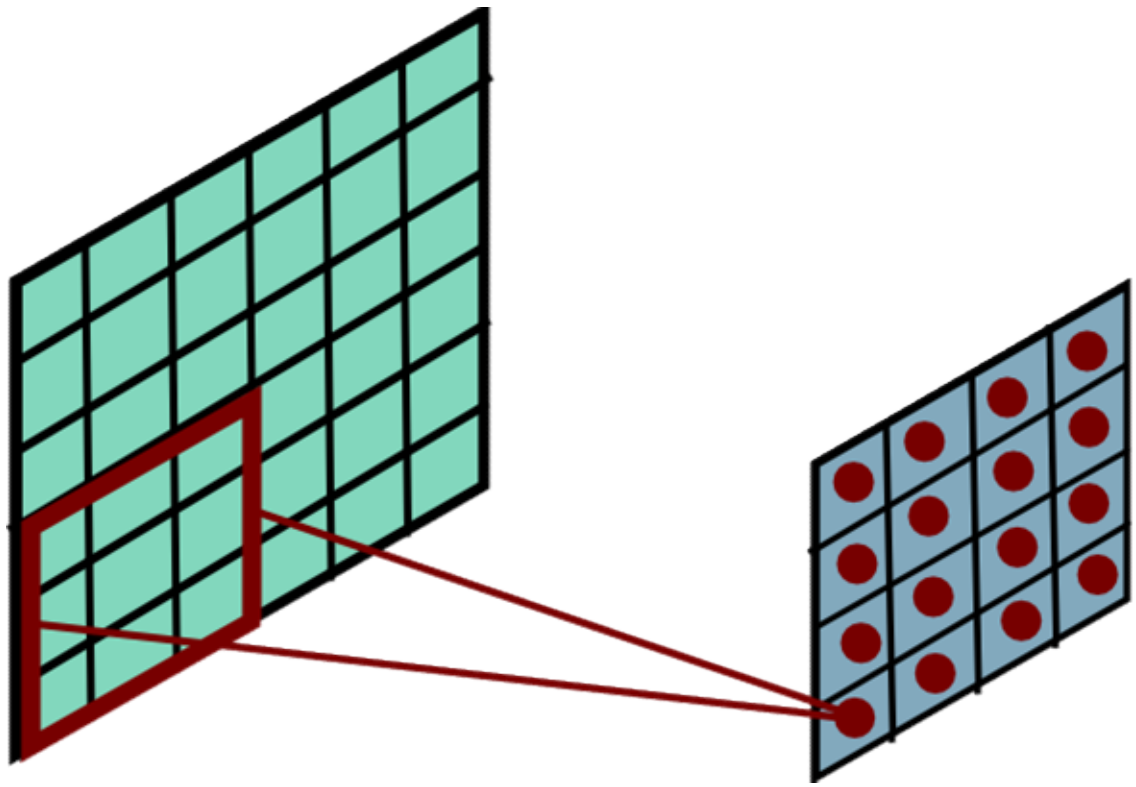
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

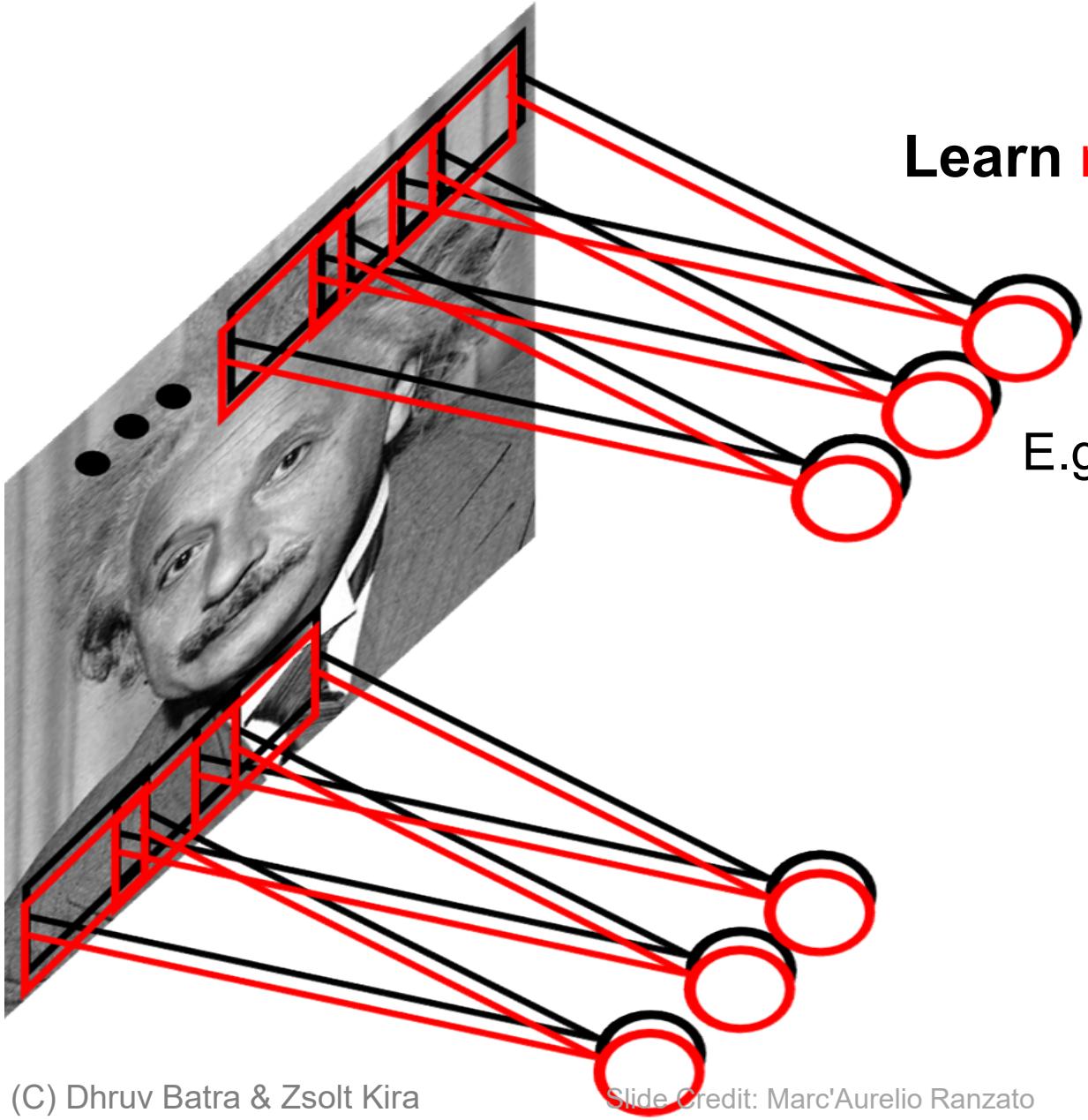


Slide Credit: Marc'Aurelio Ranzato

Convolution Explained

- http://setosa.io/ev/image-kernels/
- https://github.com/bruckner/deepViz

Convolutional Layer

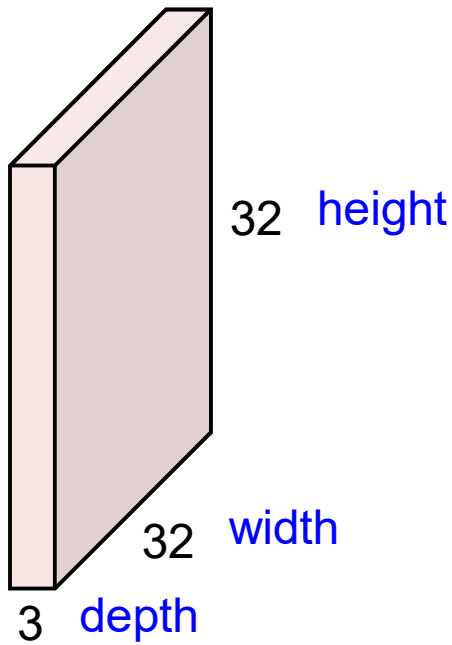


Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Convolution Layer

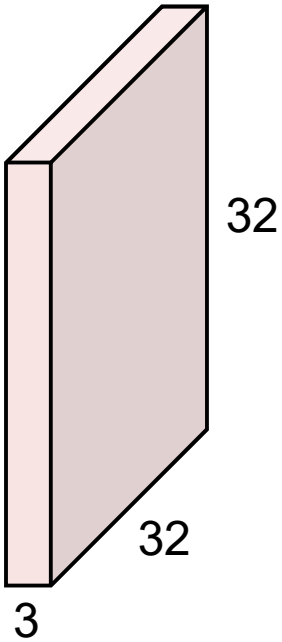
32x32x3 image -> preserve spatial structure



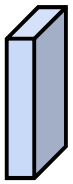
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Convolution Layer

32x32x3 image



5x5x3 filter

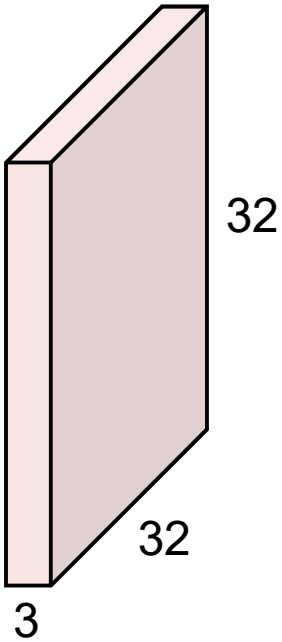


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

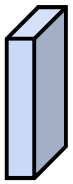
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

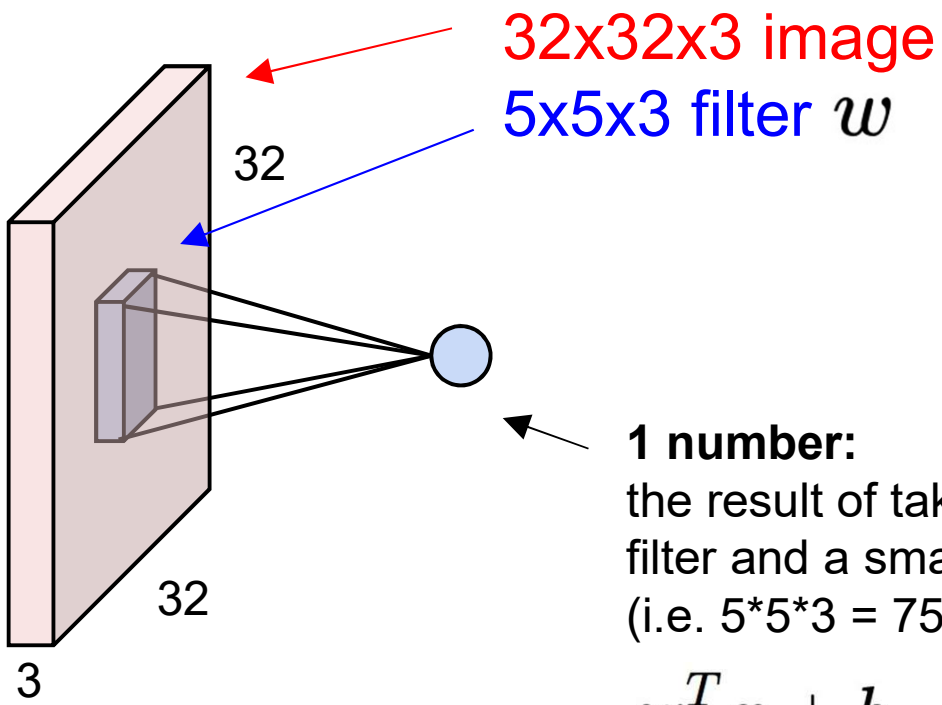


5x5x3 filter



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

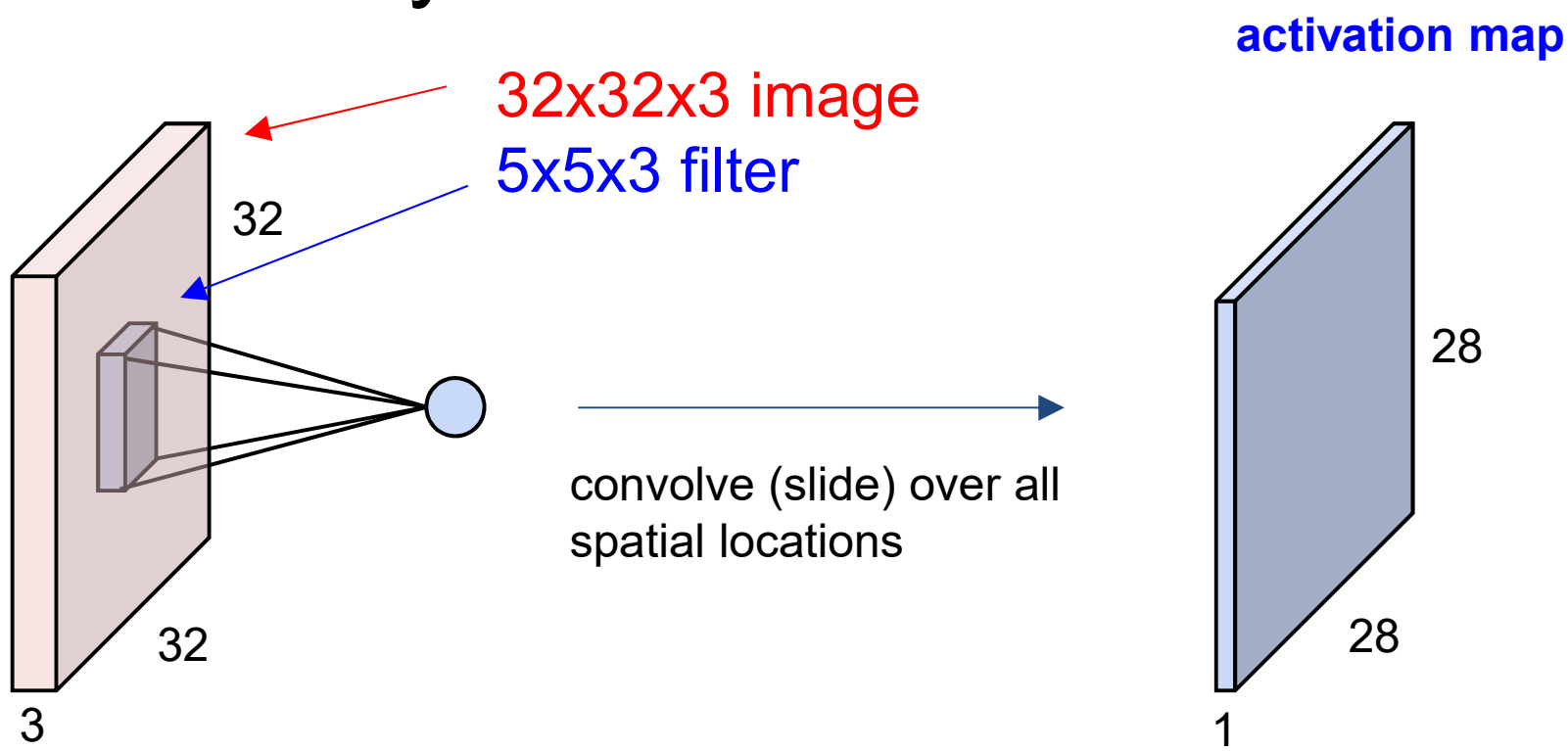
Convolution Layer



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)

$$w^T x + b$$

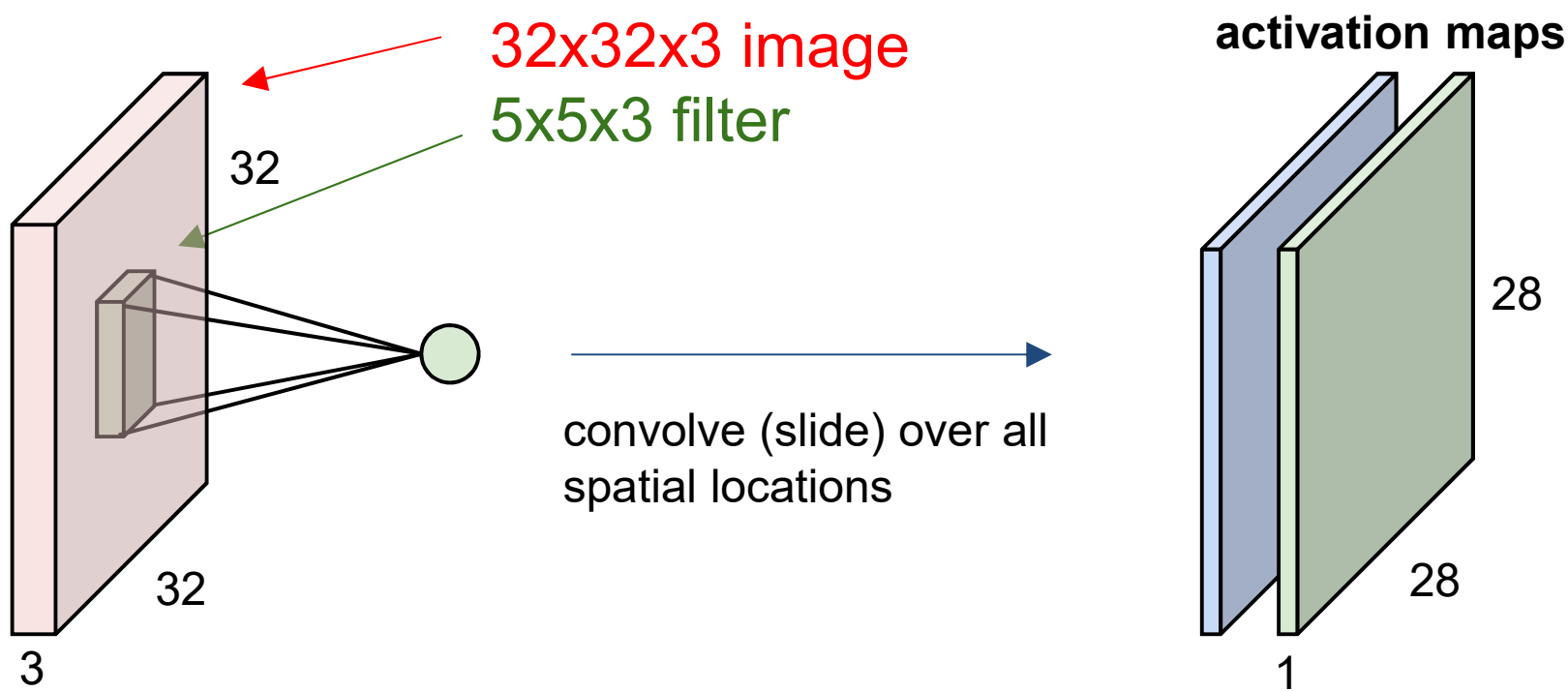
Convolution Layer



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

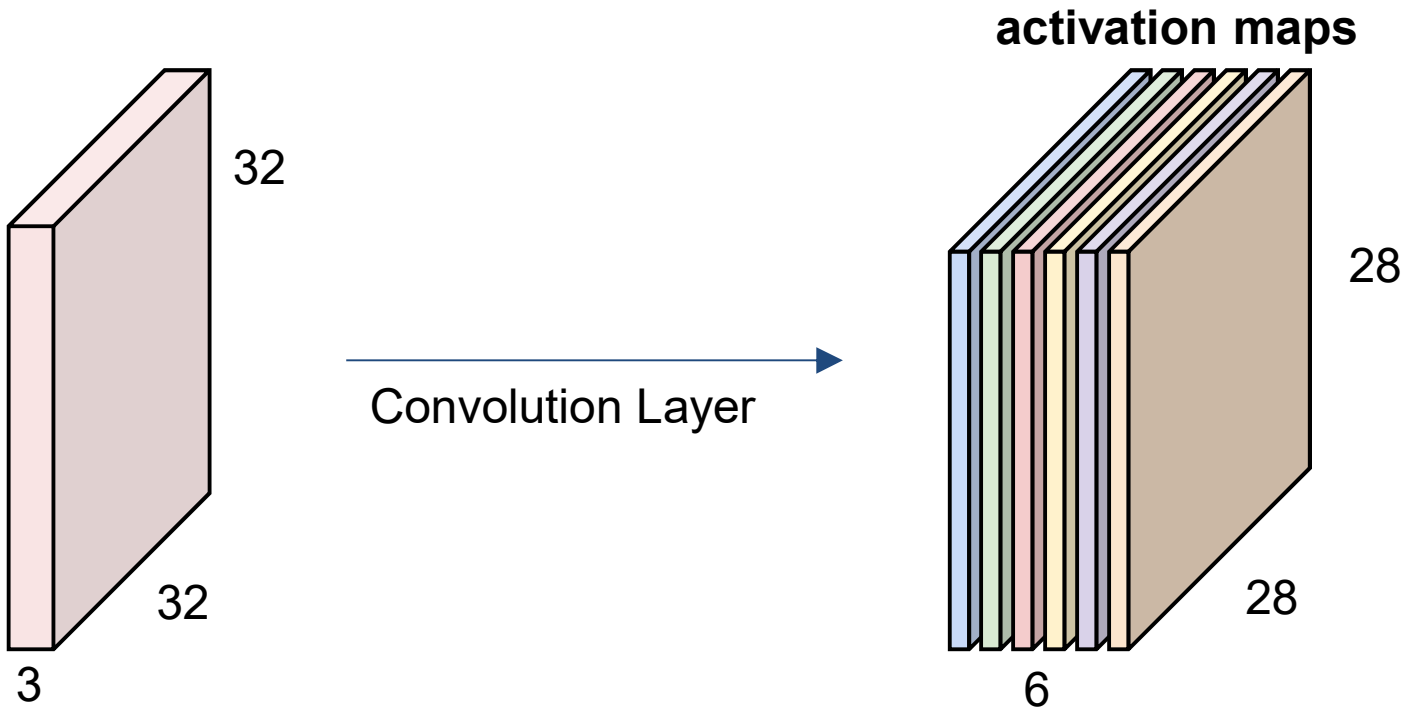
Convolution Layer

consider a second, green filter



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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!

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Im2Col

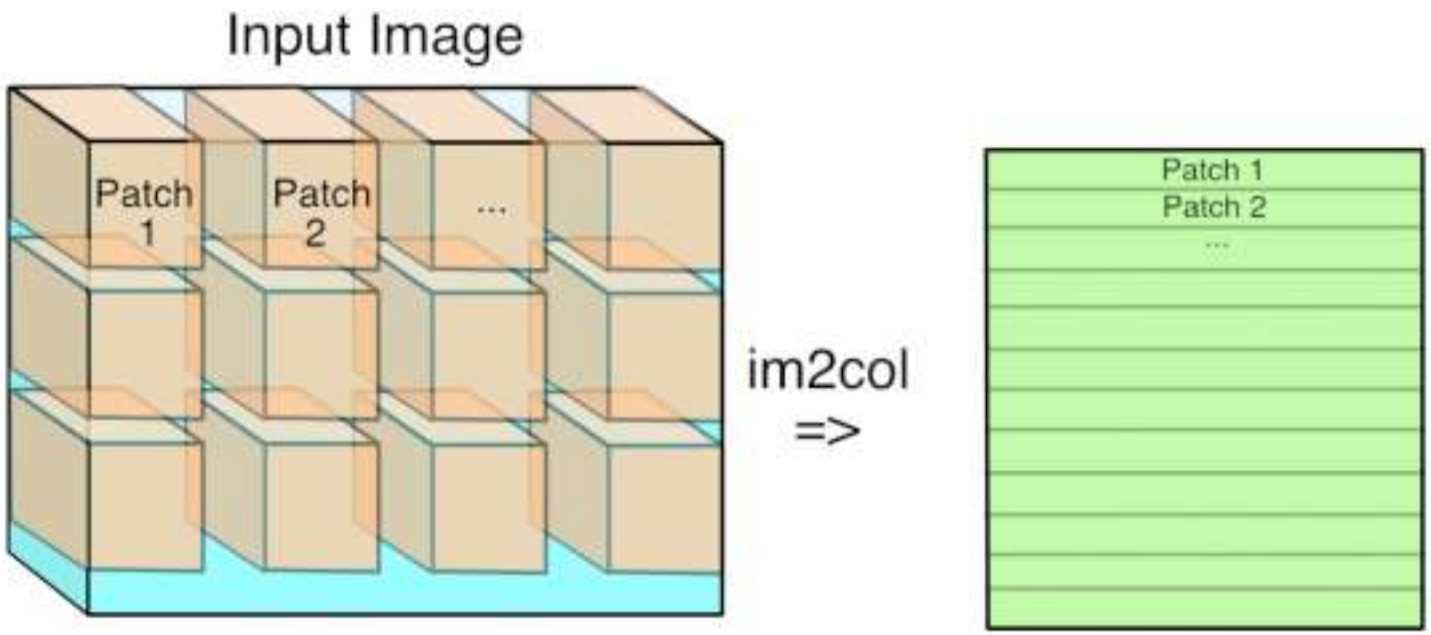


Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

General Matrix Multiply (GEMM)

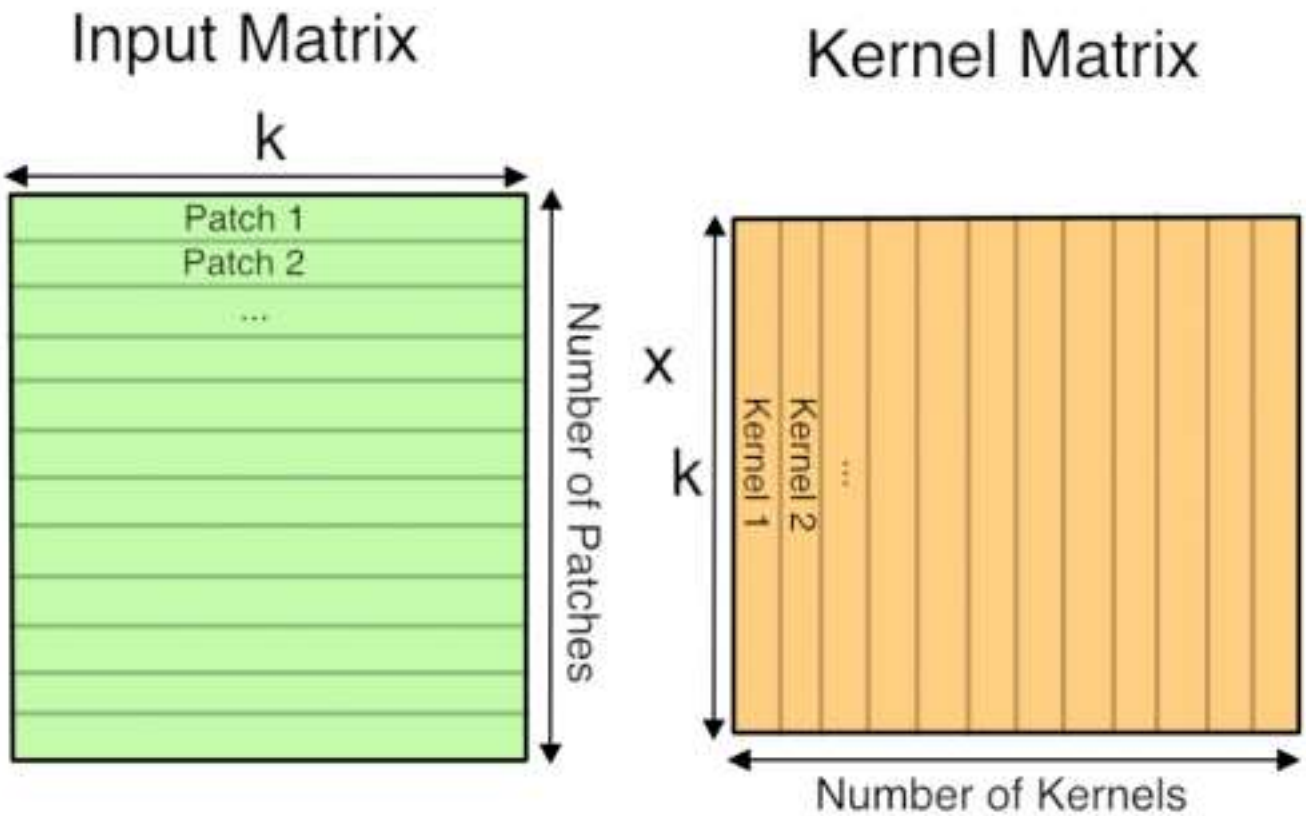
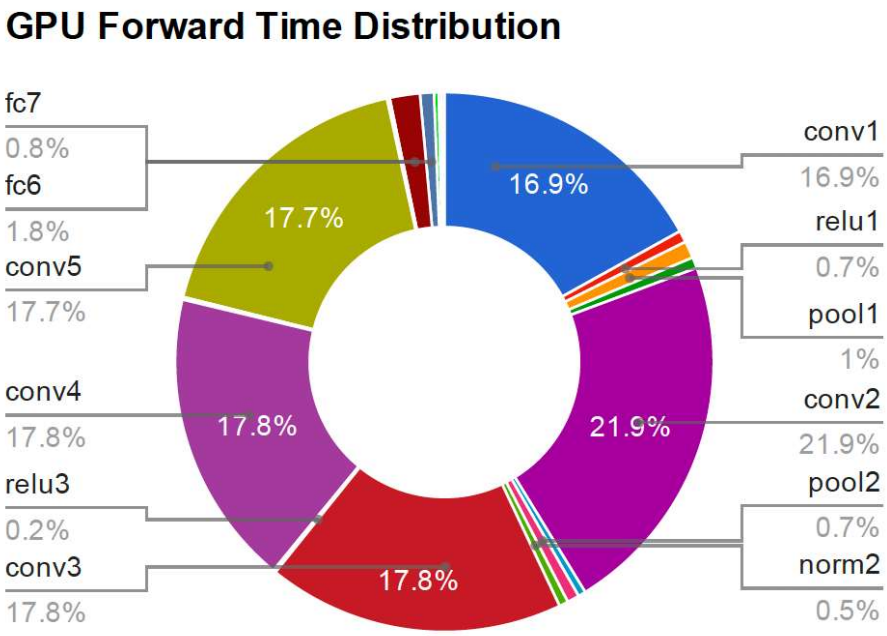


Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Time Distribution of AlexNet



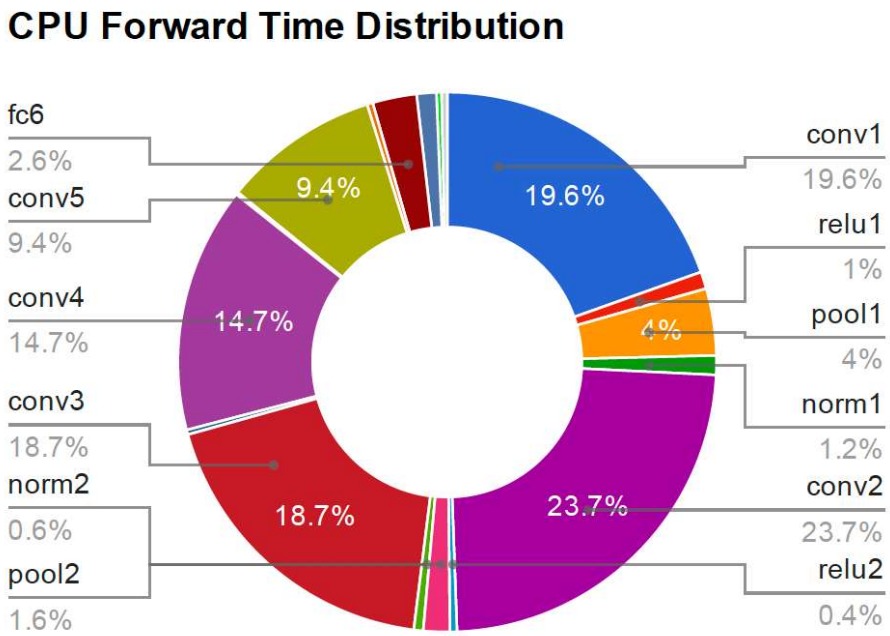
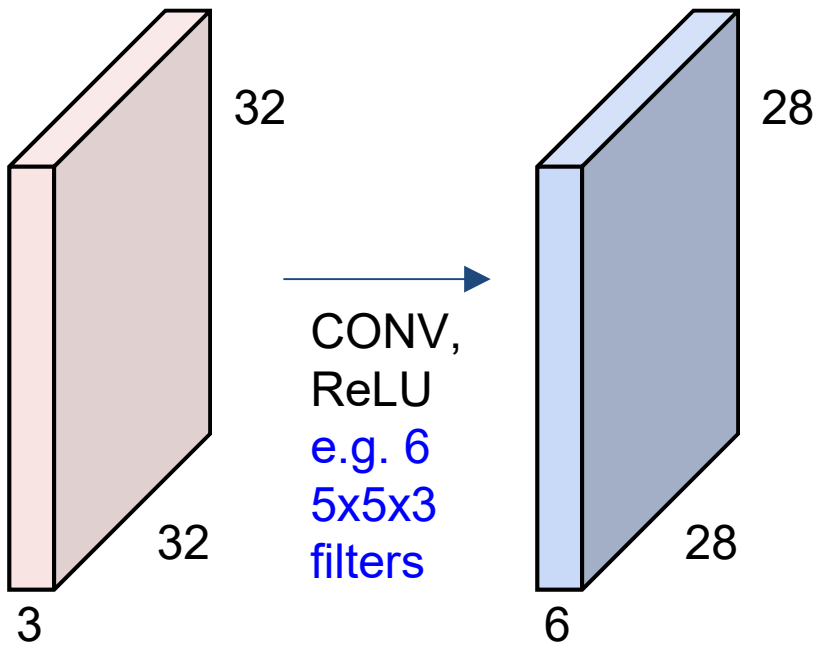


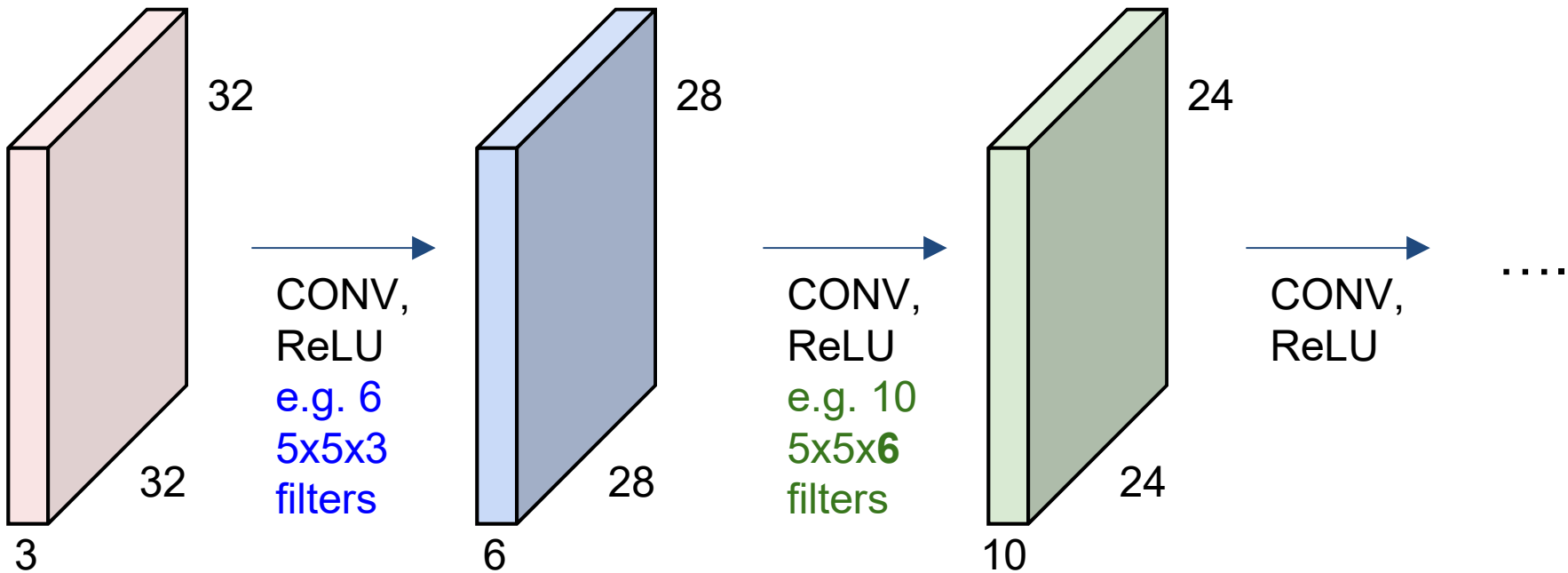
Figure Credit: Yangqing Jia, PhD Thesis

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



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n