# CS440/ECE448 Lecture 12 Computer Vision

Mark Hasegawa-Johnson, 2/2022

Slides may be redistributed under <u>CC-BY 4.0</u>. Most images are from other published sources; citations are provided inline.

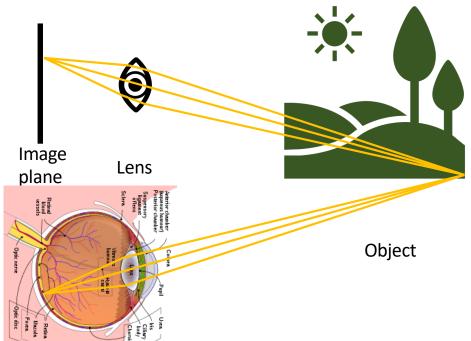
## Outline

#### • Cameras

- Pinhole camera equations
- Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

## Lenses and focus

- The lens in your eye collects light.
- Light that passes directly through the center of the lens is not bent.
- Light that passes above center is bent back toward center, and vice versa, so that it can all be collected in the same point on the image plane.

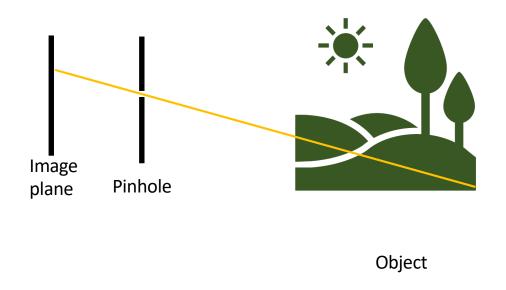


By Rhcastilhos. And Jmarchn. -

Schematic\_diagram\_of\_the\_human\_eye\_with\_English\_annotations.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1597930

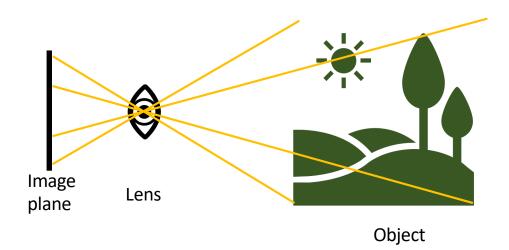
## The "pinhole camera" approximation

- A "pinhole camera" is a camera that only allows light through a very small hole.
- Disadvantage: A pinhole camera gets much less light than a lens (because the hole is smaller).
- Advantage: A pinhole camera focuses on all objects, at every distance, simultaneously.



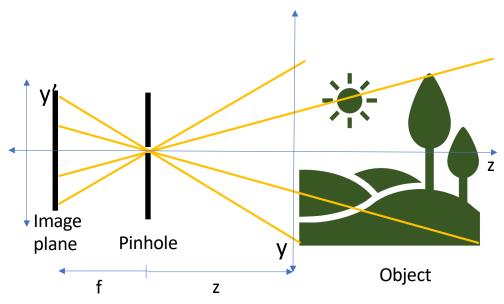
## Converting a 3D world to a 2D picture

- Different spots in the real world are projected onto different points in the image plane.
- Light that passes through the center of the lens is not bent.
- Therefore, we can use the pinhole camera approximation to analyze the relationship between real world position (x,y,z) and position on the image plane (x',y').



## The pinhole camera equations

- Define the origin (0,0,0) to be the pinhole.
- Define (x,y,z) as position of the object: x is horizontal (into the slide), y is vertical (upward), z is away from the camera.
- Define (x',y') as the position on the image plane where the light strikes (upside down).
- Define f as the distance from the pinhole to the image plane.



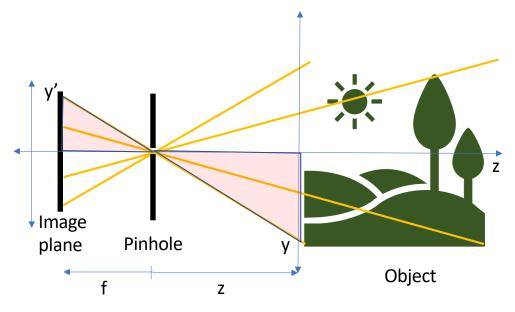
### The pinhole camera equations

• These are similar triangles! So  $y' - y \qquad x' - x$ 

$$\frac{y}{f} = \frac{-y}{z}, \qquad \frac{x}{f} = \frac{-x}{z}$$

• Solving for (x',y'), we get the pinhole camera equations:

$$x' = \frac{-fx}{z}, \qquad y' = \frac{-fy}{z}$$



## Outline

#### • Cameras

- Pinhole camera equations
- Vanishing point

### • Convolutions

- Blur
- Edge detection
- Convolutional neural net
- Imagenet

- When you take a picture, lines that are parallel in the real world appear to converge.
- The point toward which they converge is called the vanishing point. It lies on the horizon.
- The "horizon" is a line in the image, where a plane parallel to the ground passes through the pinhole.

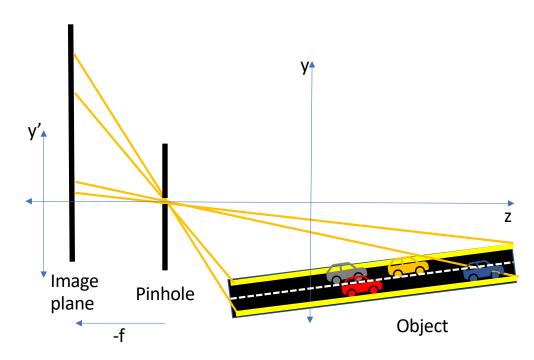


• Suppose we are imaging a couple of parallel lines, for which x and y depend on z:

Line 1:  $x = az + b_1$ ,  $y = cz + d_1$ 

Line 2:  $x = az + b_2$ ,  $y = cz + d_2$ 

- These are parallel lines, so they have the same slopes, a, c. For example, if the line are both horizontal, then c = 0.
- They are slightly offset from one another, so they have different offsets  $b_1, b_2, d_1, d_2$



Line 1:  $x = az + b_1$ ,  $y = cz + d_1$ Line 2:  $x = az + b_2$ ,  $y = cz + d_2$ • Remember the pinhole camera,

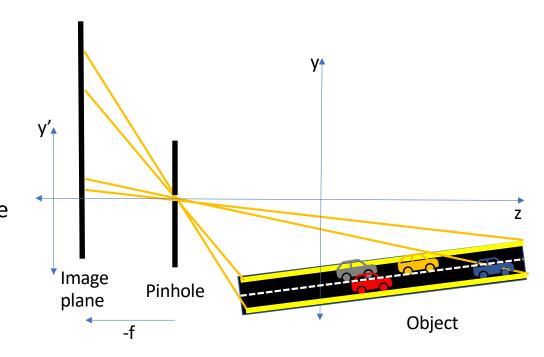
• Remember the pinhole camera equations:  $\frac{x}{z} = \frac{-x'}{f}$ , and  $\frac{y}{z} = \frac{-y'}{f}$ . In the (x', y') plane, therefore, the lines are

Line 1:

$$\frac{-x'z}{f} = az + b_1, \qquad \frac{-y'z}{f} = cz + d_1$$

Line 2:

$$\frac{-x'z}{f} = az + b_2, \qquad \frac{-y'z}{f} = cz + d_2$$



Line 1:

$$\frac{-x'z}{f} = az + b_1, \qquad \frac{-y'z}{f} = cz + d_1$$

Line 2:

$$\frac{-x'z}{f} = az + b_2, \qquad \frac{-y'z}{f} = cz + d_2$$

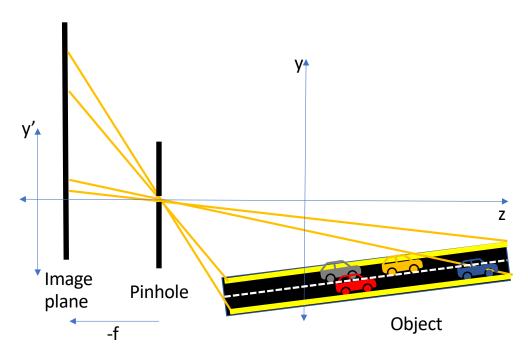
Now notice that, as  $z \to \infty$ , the first two terms become infinitely larger than the third term. If we divide through by z, and allow  $z \to \infty$ , we find that both lines converge to the same point:

Vanishing Point of Line 1:

$$x' = -af, \qquad y' = -cf$$

Vanishing Point of Line 2:

$$x' = -af, \qquad y' = -cf$$



Vanishing Point of Line 1: x' = -af, y' = -cfVanishing Point of Line 2: x' = -af, y' = -cf



For example, if the lines in the real world are horizontal (c = 0) and parallel to the z axis (a = 0), then in the image, they converge to the point (x', y') = (0,0).

## Outline

#### • Cameras

- Pinhole camera equations
- Vanishing point

#### • Convolutions

- Blur
- Edge detection
- Convolutional neural net
- Imagenet

- An RGB image is a signal in three dimensions: f[i, j, k] = intensity of the signal in the i<sup>th</sup> row, j<sup>th</sup> column, and k<sup>th</sup> color.
- f[i, j, k], for each (i, j, k), is either stored as an integer or a floating point number:
  - Floating point: usually x ∈ [0, 1], so x = 0 means dark, x = 1 means bright.
  - Integer: usually x ∈ {0,..., 255}, so x = 0 means dark, x = 255 means bright.
- The three color planes are usually:
  - *k* = 0: Red
  - *k* = 1: Blue
  - *k* = 2: Green

- "Local averaging" means that we create an output image, y[i, j, k], each of whose pixels is an average of nearby pixels in f[i, j, k].
- For example, if we average along the rows:

$$y[i,j,k] = \frac{1}{2M+1} \sum_{j'=j-M}^{j+M} f[i,j',k]$$

• If we average along the columns:

$$y[i,j,k] = \frac{1}{2M+1} \sum_{i'=i-M}^{i+M} f[i',j,k]$$





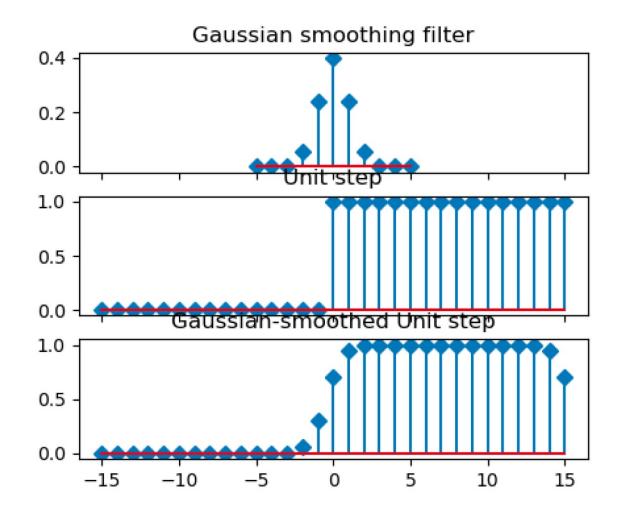
- Suppose we don't want the edges quite so abrupt. We could do that using "weighted local averaging:" each pixel of y[i, j, k] is a weighted average of nearby pixels in f[i, j, k], with some averaging weights g[n].
- For example, if we average along the rows:

$$y[i, j, k] = \sum_{m=j-M}^{j+M} g[j-m]f[i, m, k]$$

• If we average along the columns:

$$y[i,j,k] = \sum_{i'=i-M}^{i+M} g[i-m]f[m,j,k]$$

The top row are the averaging weights, g[n]. The middle row shows the input, f[n]. The bottom row shows the output, y[n].



## Gaussian blur

Weighted averaging is a little better than unweighted averaging. We just need to make sure that the weights add up to one. For example:

$$y[i, j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} g[m, n] f[i - m, j - n]$$

$$g[m,n] = \frac{1}{2\pi\sigma^2} e^{-\left(\left(\frac{m}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right)}$$

The Gaussian blur filter h(m, n) results in different degrees of smoothness of g(x', y'), depending on how we set the hyperparameter  $\sigma$  (the StDev):



By IkamusumeFan - Own work, CC BY-SA 4.0, https://commons.wikime dia.org/w/index.php?curi d=41790217

 A convolution is exactly the same thing as a weighted local average. We give it a special name, because we will use it very often. It's defined as:

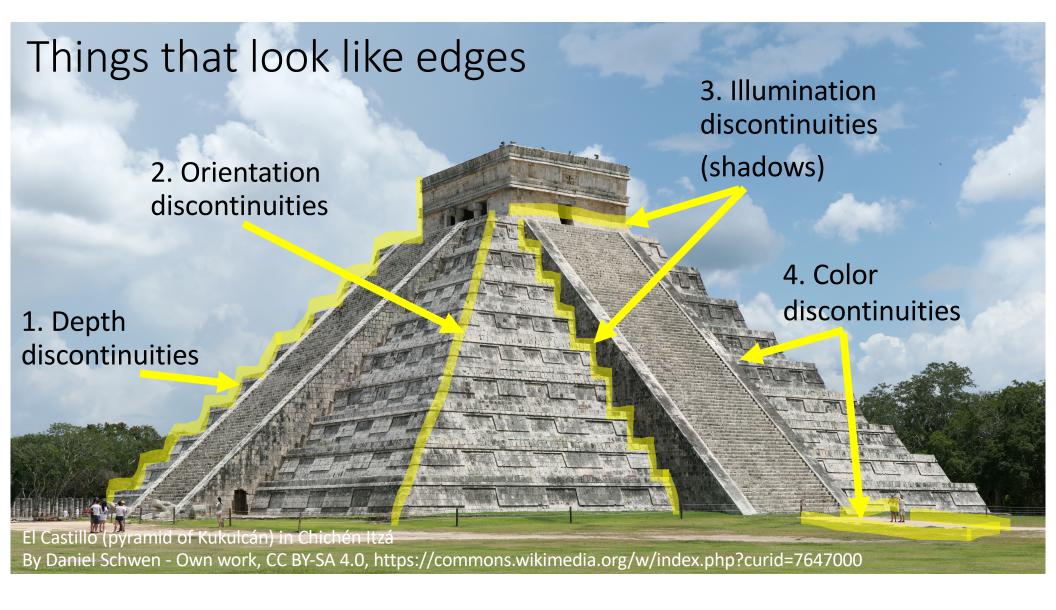
$$y[n] = \sum_{m} g[m]f[n-m] = \sum_{m} g[n-m]f[m]$$

• We use the symbol \* to mean "convolution:"

$$y[n] = g[n] * f[n] = \sum_{m} g[m]f[n-m] = \sum_{m} g[n-m]f[m]$$

## Outline

- Cameras
  - Pinhole camera equations
  - Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet



## Edge detection by subtraction

Subtract neighboring pixels, to compute an "image gradient:" Image 0 Gx Image 0 Gy 0 X gradient:  $\nabla_x f[i,j] \approx \frac{f[i,j+1] - f[i,j-1]}{2}$ 100 0 Y gradient: 100  $\nabla_y f[i,j] \approx \frac{f[i+1,j] - f[i-1,j]}{2}$ 0 100

100

0

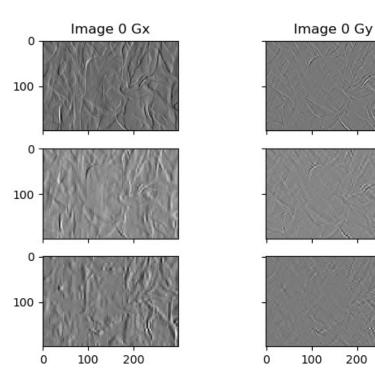
200

100 200

0

## A problem with "edge detection by subtraction"

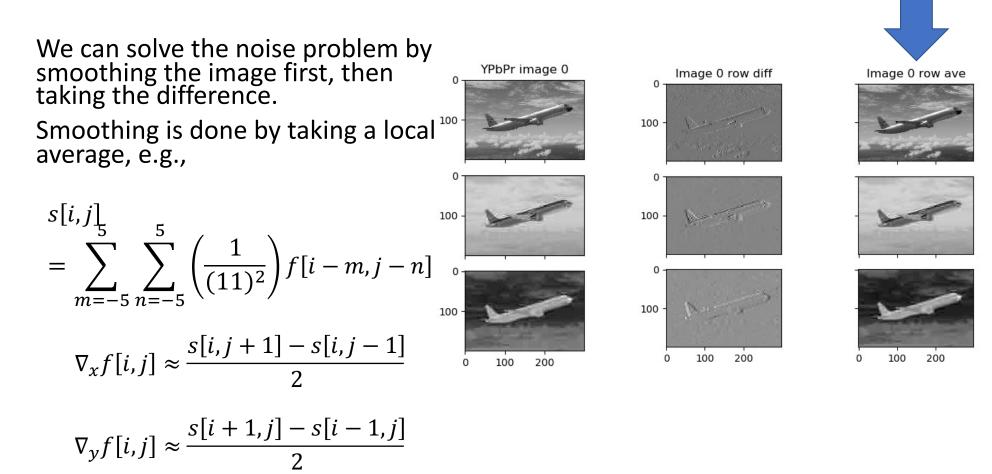
It tends to exaggerate noise.





200

## Solving the noise problem



### Gaussian blur

Weighted averaging is a little better than unweighted averaging. We just need to make sure that the weights add up to one. For example:

$$s[i,j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} h[m,n]f[i-m,j-n]$$

$$h[m,n] = \frac{1}{2\pi\sigma^2} e^{-\left(\left(\frac{m}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right)}$$

$$\nabla_x f[i,j] \approx \frac{s[i,j+1] - s[i,j-1]}{2}$$

$$\nabla_y f[i,j] \approx \frac{s[i+1,j] - s[i-1,j]}{2}$$

The Gaussian blur filter h(m, n) results in different degrees of smoothness of S(x', y'), depending on how we set the hyperparameter  $\sigma$  (the StDev):



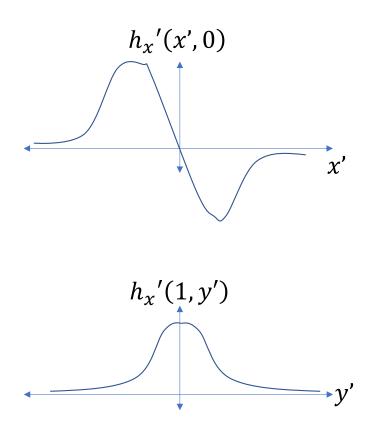
By IkamusumeFan - Own work, CC BY-SA 4.0, https://commons.wikime dia.org/w/index.php?curi d=41790217

### Difference of Gaussians

A "difference-of-Gaussians" filter is created by subtracting two Gaussian-blur filters, like this:

$$h[m,n] = \frac{1}{2\pi\sigma^2} e^{-\left(\left(\frac{m}{\sigma}\right)^2 + \left(\frac{n}{\sigma}\right)^2\right)}$$
$$h_x'[m,n] = \frac{h[m,n+1] - h[m,n-1]}{2}$$
$$h_y'[m,n] = \frac{h[m+1,n] - h[m-1,n]}{2}$$

A "difference-of-Gaussians" filter looks kind of like this:



## Difference of Gaussians

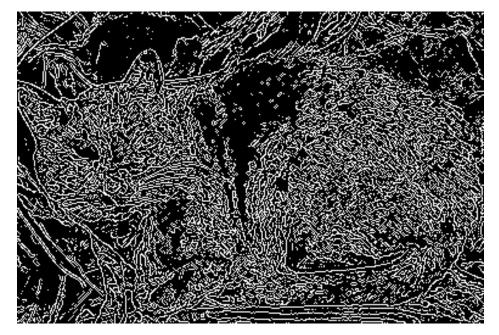
If we pre-compute the difference-of-Gaussians filters, then we can combine the weighted-average and the subtraction into just one operation, to save computation:

$$\nabla_{x} f[i,j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} h_{x}'[m,n] f[i-m,j-n]$$

$$\nabla_{y} f[i, j] = \sum_{m=-5}^{5} \sum_{n=-5}^{5} h_{y}'[m, n] f[i - m, j - n]$$

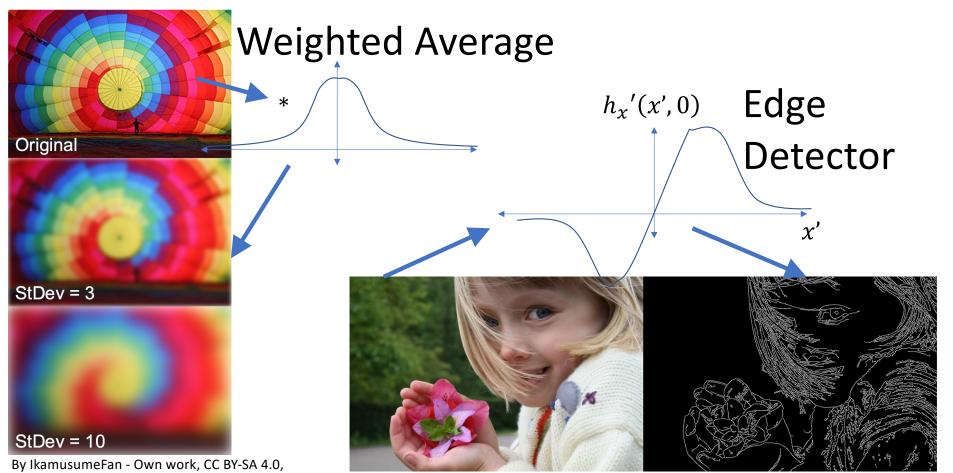
The difference-of-Gaussians filters,  $h_x'[m,n]$  and  $h_y'[m,n]$ , detect more or less edges, depending on how we set the hyperparameter  $\sigma$ . Here is

 $\sqrt{\nabla_x f[i,j]^2 + \nabla_y f[i,j]^2}$ , thresholded to make it black and white:



By Overremorto - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10581259

Example Convolutions: Weighted Average, Edge Detector



https://commons.wikimedia.org/w/index.php?curid=41790217one, CC-SA 3.0, https://commons.wikimedia.org/wiki/File:%C3%84%C3%A4retuvastuse\_n%C3%A4ide.png

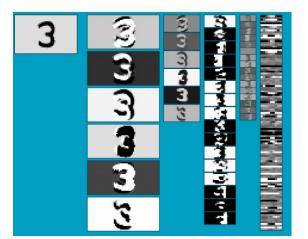
## Outline

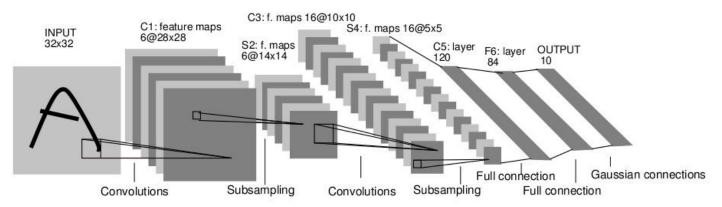
#### • Cameras

- Pinhole camera equations
- Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

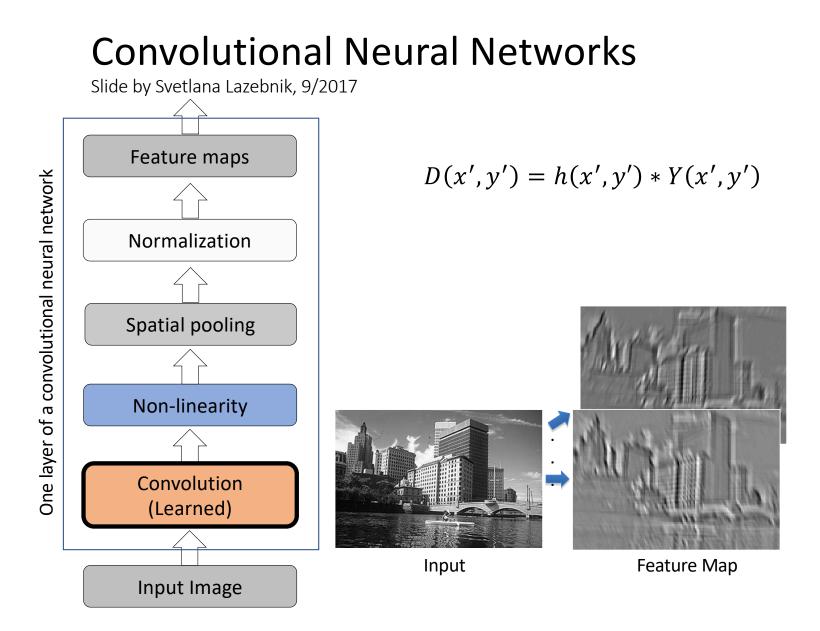
## **Convolutional Neural Networks**

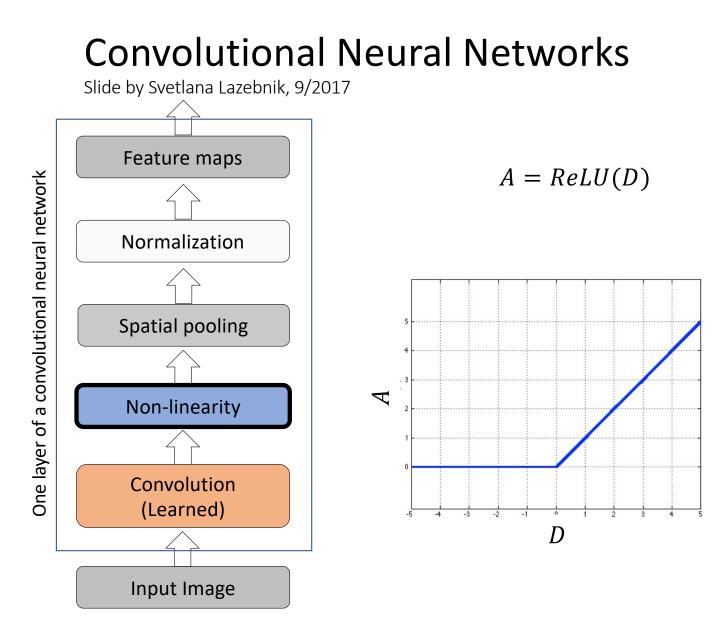
- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors (trainable convolutions!!)
- Higher stages compute more global, more invariant features
- Classification layer at the end





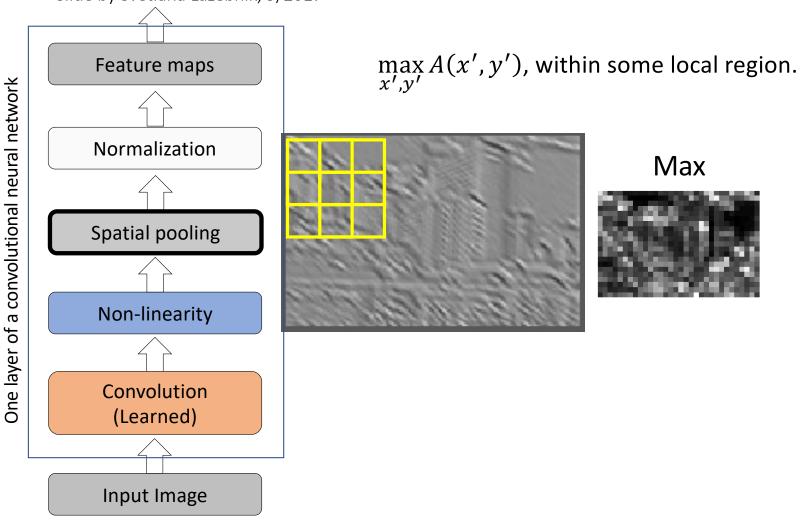
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, <u>Gradient-based learning applied to document recognition</u>, Proc. IEEE 86(11): 2278–2324, 1998.



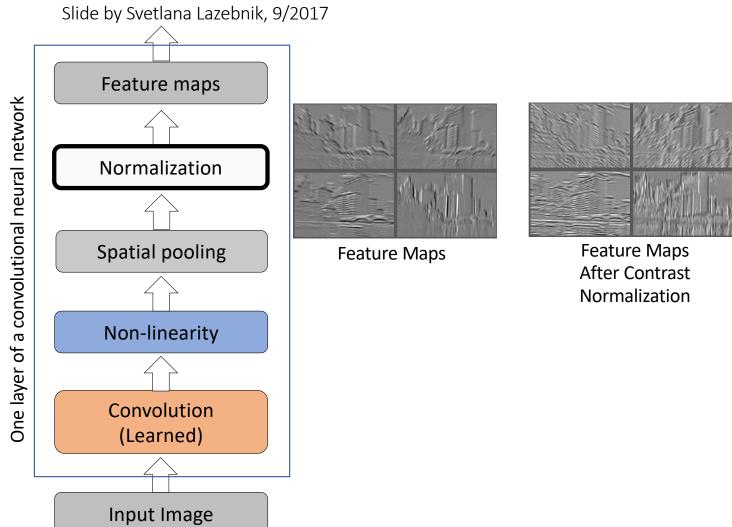




Slide by Svetlana Lazebnik, 9/2017

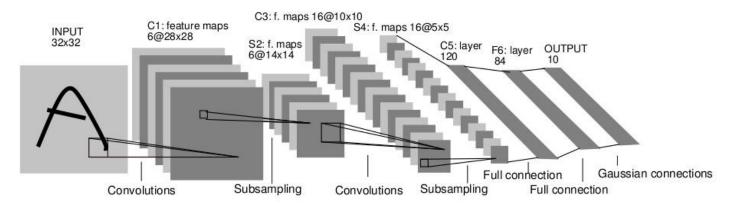


### **Convolutional Neural Networks**



### LeNet, 1998

- (Convolution, Nonlinearity, Max-Pooling, Normalization) X 2 layers
- After the second max-pooling, the resulting features are concatenated to form a vector, which is then classified.



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, <u>Gradient-based learning applied to document recognition</u>, Proc. IEEE 86(11): 2278–2324, 1998.

# Outline

#### • Cameras

- Pinhole camera equations
- Vanishing point
- Convolutions
  - Blur
  - Edge detection
  - Convolutional neural net
  - Imagenet

# ImageNet Challenge

# IM GENET



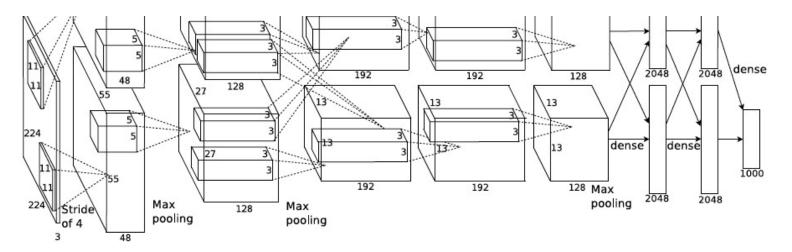
[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

A. Krizhevsky, I. Sutskever, and G. Hinton, <u>ImageNet Classification with Deep Convolutional</u> <u>Neural Networks</u>, NIPS 2012

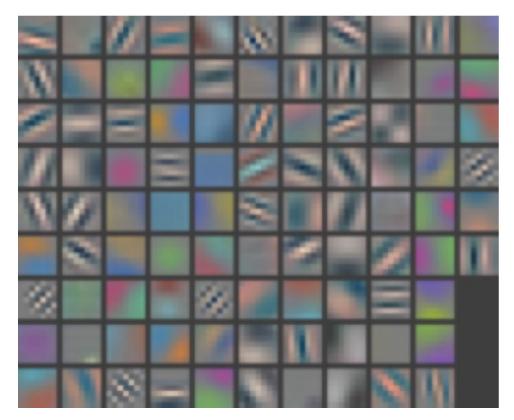
### AlexNet

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10<sup>6</sup> vs. 10<sup>3</sup> images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



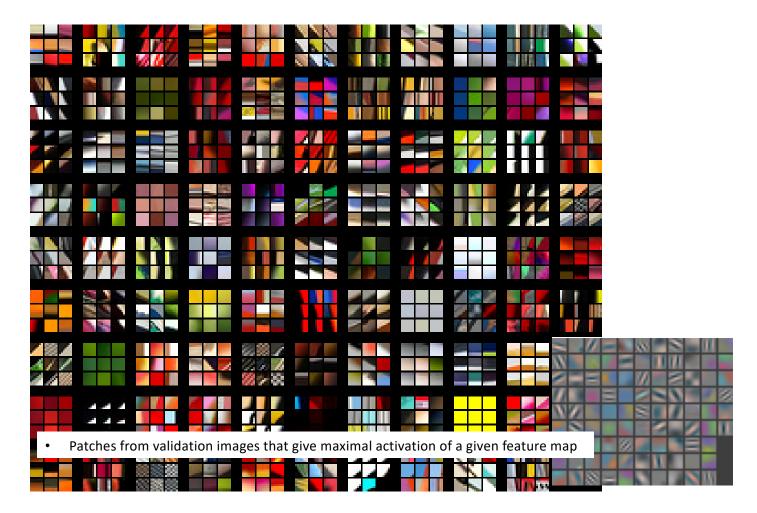
A. Krizhevsky, I. Sutskever, and G. Hinton, <u>ImageNet Classification with Deep Convolutional</u> <u>Neural Networks</u>, NIPS 2012

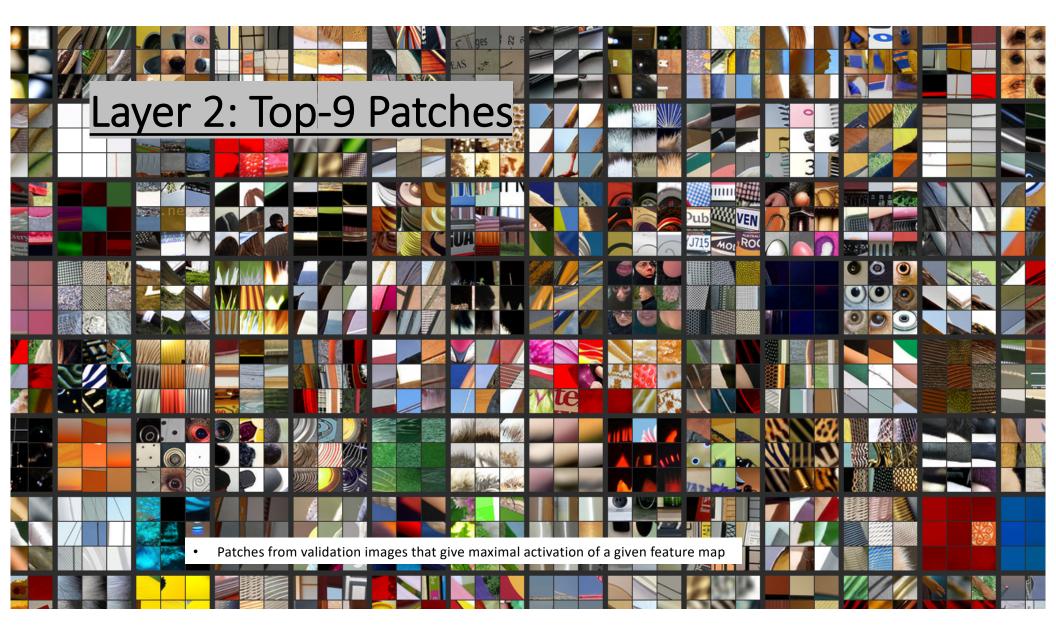
### Layer 1 Filters



M. Zeiler and R. Fergus, <u>Visualizing and Understanding Convolutional Networks</u>, arXiv preprint, 2013

### Layer 1: Top-9 Patches











## Conclusions

- Pinhole camera equations tell you the relationship between the position on the image, (x',y'), and the position in the real world, (x,y,z). In particular, they tell you why parallel lines seem to converge at the vanishing point.
- Weighted averaging (e.g., Gaussian blur) and edge detection are two examples of convolution.
- Edges are caused by discontinuities of depth, orientation, illumination, and color. It's useful to detect where such things happen! Subtracting pixels is noisy, so convolve with a difference-of-gaussians filter instead.
- A convolutional neural network is a series of layers, each of which contains a convolution, followed by a nonlinearity, followed by a local maximum.
- ImageNet is a database with millions of images, showing examples of over 1000 different objects. It has permitted people to train extremely complicated and highly accurate neural nets.