Principles of Robot Autonomy I

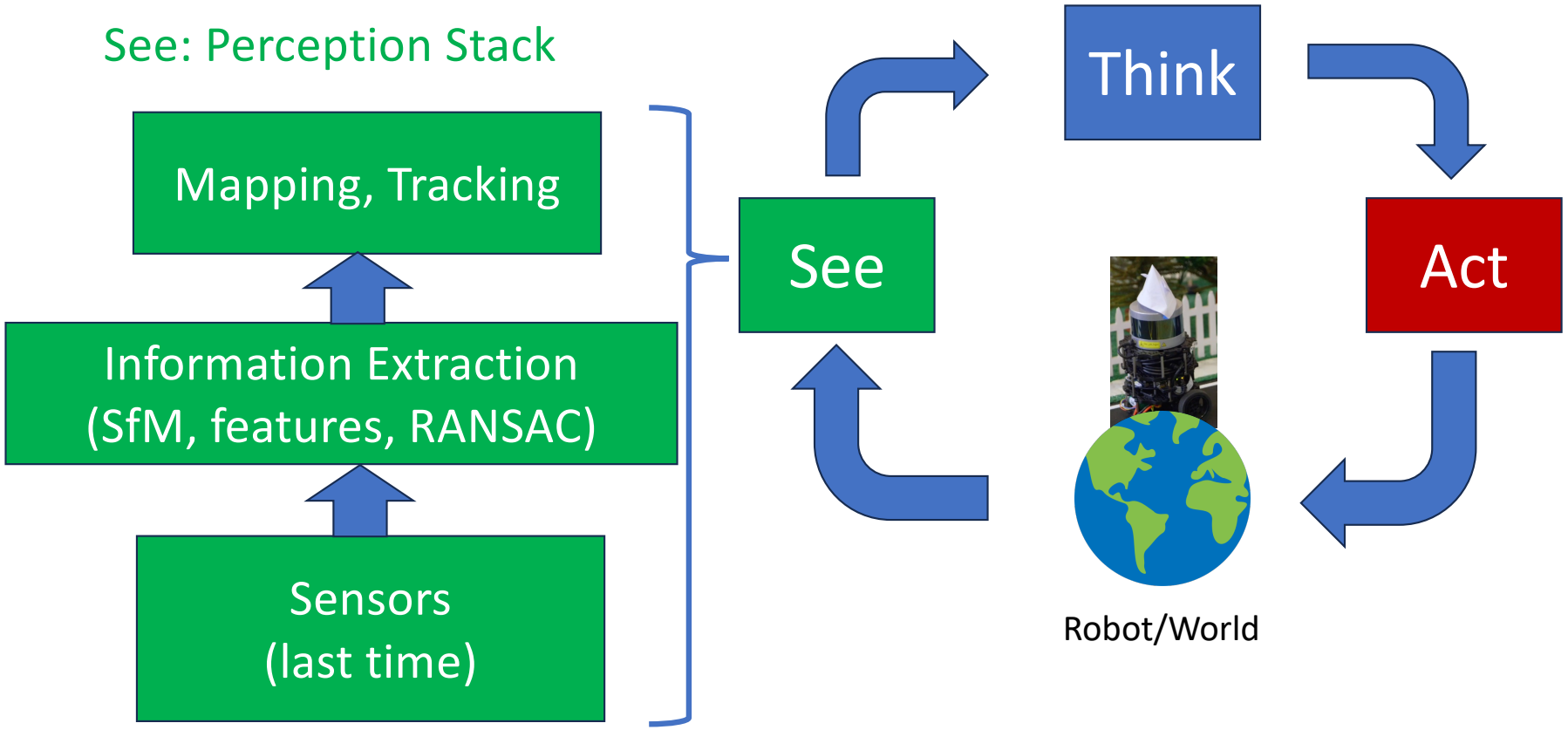
Features, Image convolutions, CNNs, Semantic Perception

Logistics

- Homework 3: due Tues, Oct 28
- Homework 4: out Tues Nov 4 (After the Midterm)
- Midterm window: Wed, Oct 29, 5pm – Sat, Nov 1
  - Take home, 72 hour window
  - Check out exam on gradescope
  - You will have personal 5 hour time slot
  - Open notes, book, HW solutions
  - No internet, no GenAI, no working with others
- Lecture 10:
  - Features
  - Image convolutions
  - CNNs and learning-based semantic CV

Robot Perception

See: Perception Stack



Perception Stack: Computer vision, filtering, SLAM

Use sensor data to update the models

Localization, Mapping, Tracking:

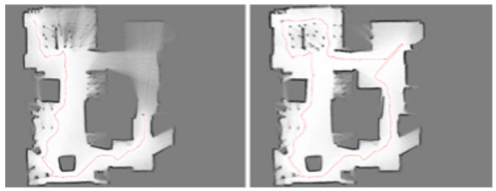
- EKF/Monte Carlo localization
- Occupancy grid mapping
- Pose graph optimization
- Tracking (EKF and Particle Filter)
- **AA273: Filtering (Schwager)**
- **AA275: Navigation (Gao)**

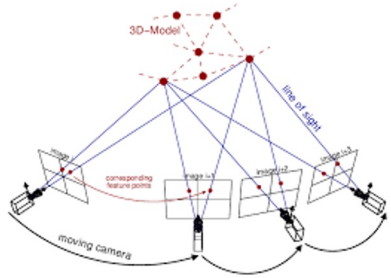
Information extraction

- Computer vision: features, correspondences, Structure from Motion (SfM), depth
- Lidar scan matching, ICP
- **CS231A: Comp Vision**

Sensors:

- RGB Cameras, RGB-D/stereo cameras, Lidar
- IMU, GPS, wheel encoders





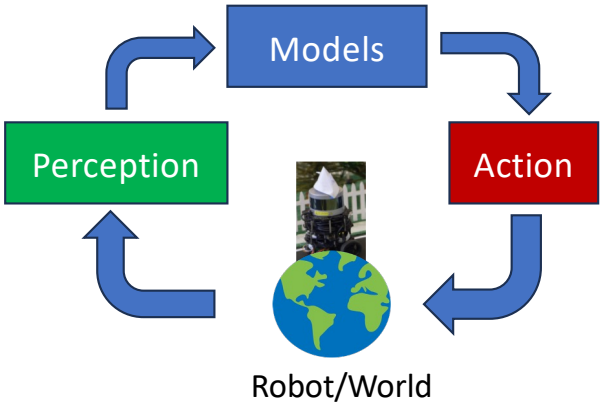


Image Feature Correspondences

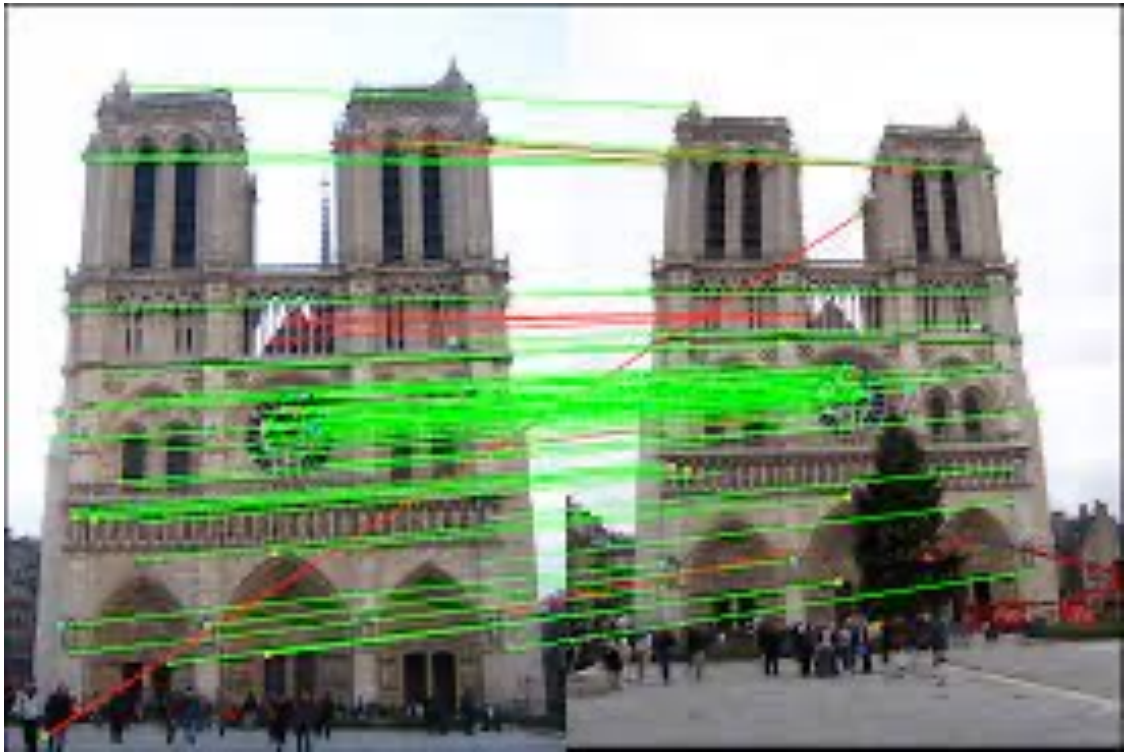
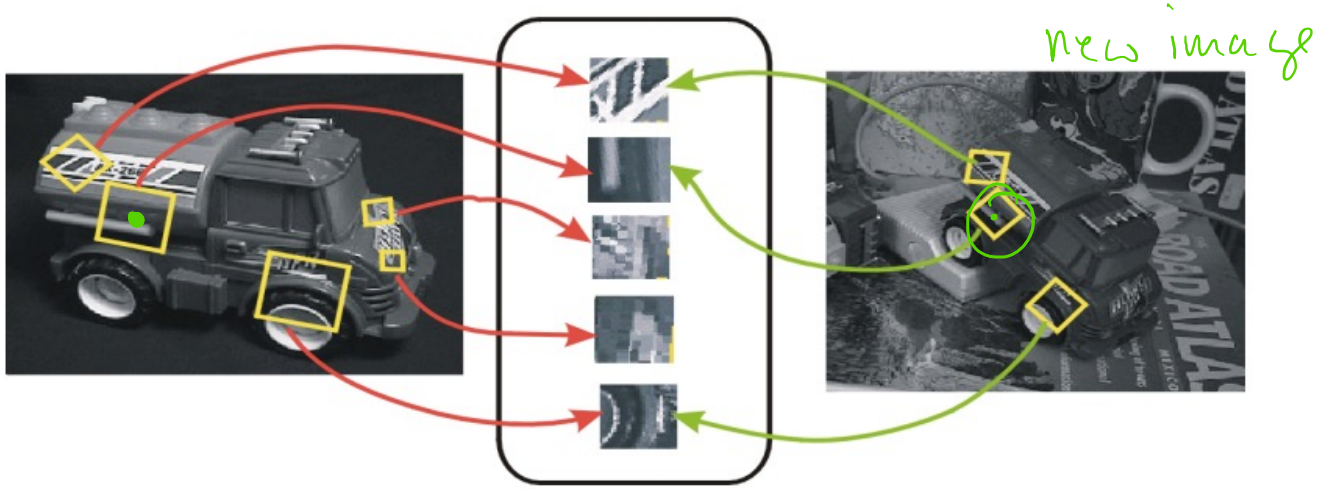


Image Features and their Descriptors

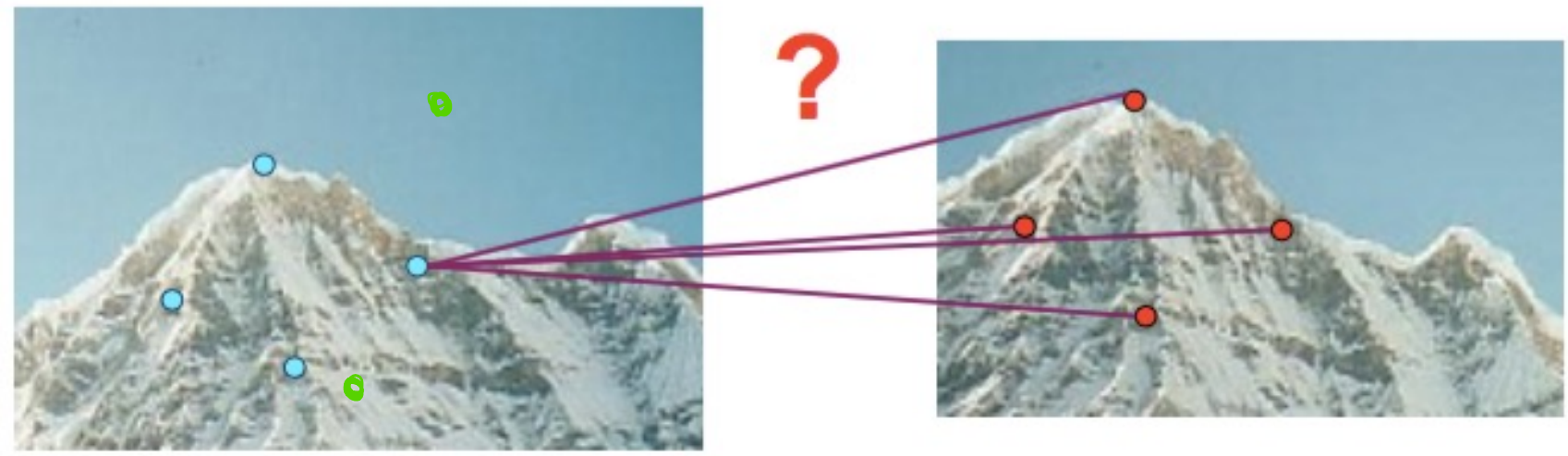
- Goal: *describe* keypoints so that we can compare them across images or use them for object detection or matching
- Desired properties:
  - Invariance with respect to pose, scale, illumination, etc.
  - Distinctiviness



Repeatability

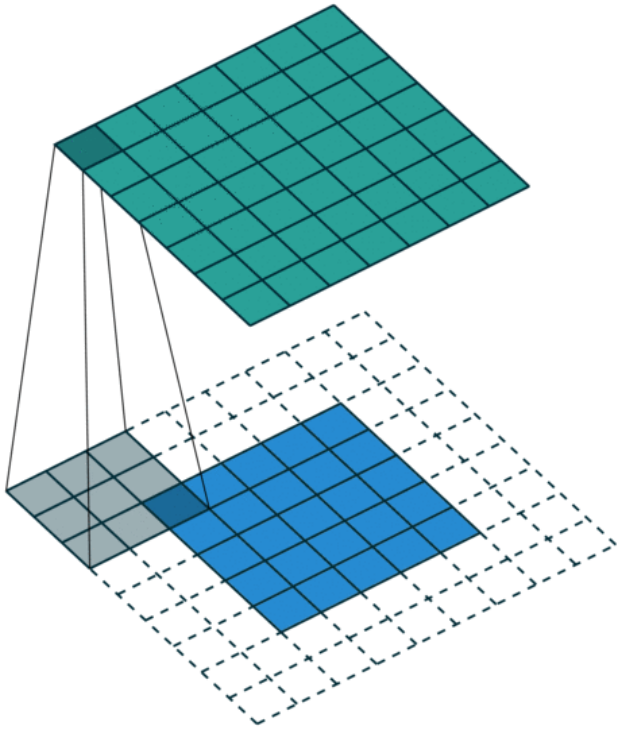
Without repeatability, matching is impossible

Distinctiveness



Without distinctiveness, it is not possible to establish reliable correspondences; distinctiveness is key for having a useful descriptor

Filter/Convolution Visualization



Source: Sanja Fidler

Image Convolutional Filters

- Filters can be linear or non-linear
- We will focus on linear spatial filters

$$I'(u,v) = F * I = \sum_{i=-n}^{n} \sum_{j=-m}^{m} F(i,j) I(u+i, v+j)$$

Filtered image

Filter mask

Original image

- Filter *F* (of size $(2n+1) \times (2m+1)$) is usually called a mask, kernel, or window
- Dealing with boundaries: e.g., pad, crop, extend, or wrap

Filter example #1: moving average

- The moving average filter returns the average of the pixels in the mask
- Achieves a smoothing effect (removes sharp features)
- E.g., for a *normalized* 3×3 mask

$$F = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Generated with a 5x5 mask

Filter example #2: Gaussian smoothing

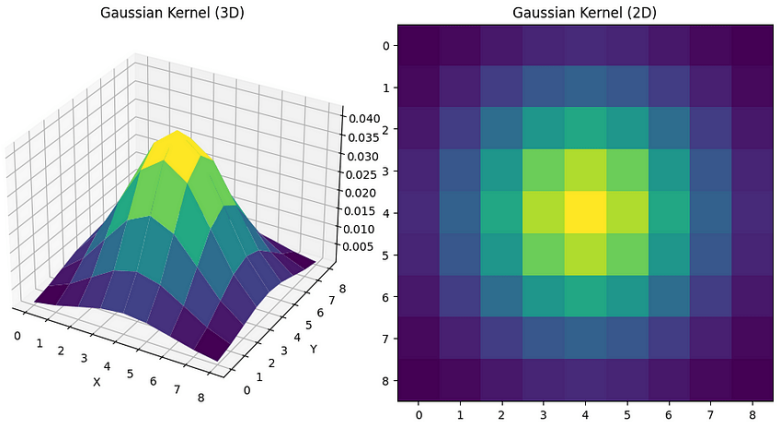
- Gaussian function

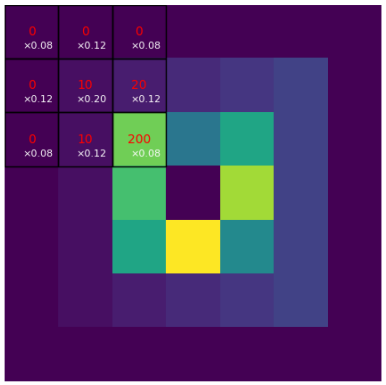
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

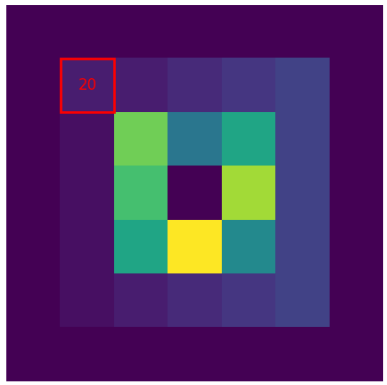
- To obtain the mask, sample the function about its center
- E.g., for a *normalized* 3×3 mask with $\sigma = 0.85$

$$G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

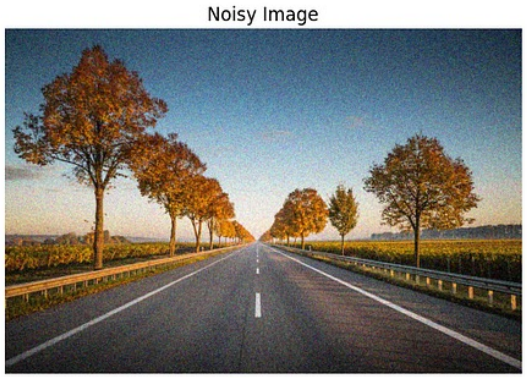
Filter example #2: Gaussian smoothing







$$G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



Visualizations by Vijay Vignesh

Differentiation

- Derivative of discrete function (centered difference)

$$\frac{\partial I}{\partial x} = I(x+1, y) - I(x-1, y) \qquad F_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$\frac{\partial I}{\partial y} = I(x, y+1) - I(x, y-1) \qquad F_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

- Derivative as a convolution operation; e.g., Sobel masks:

Along *x* direction

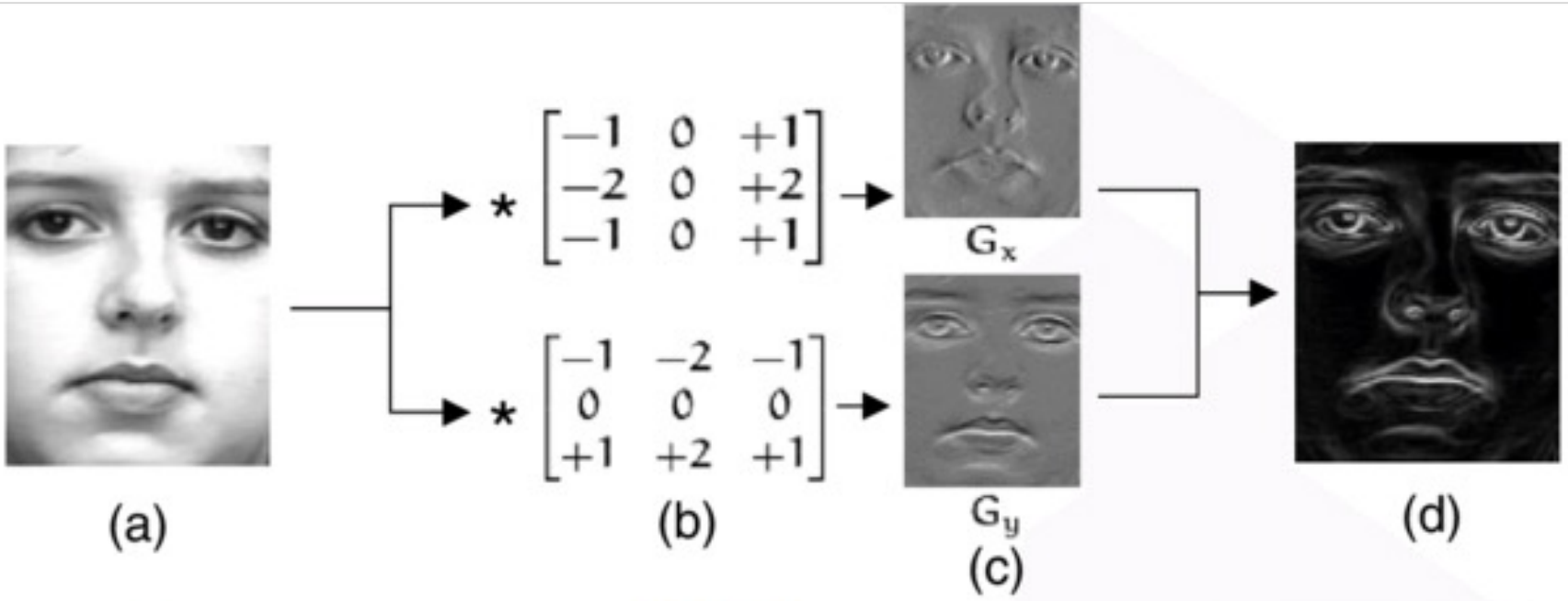
$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Along *y* direction

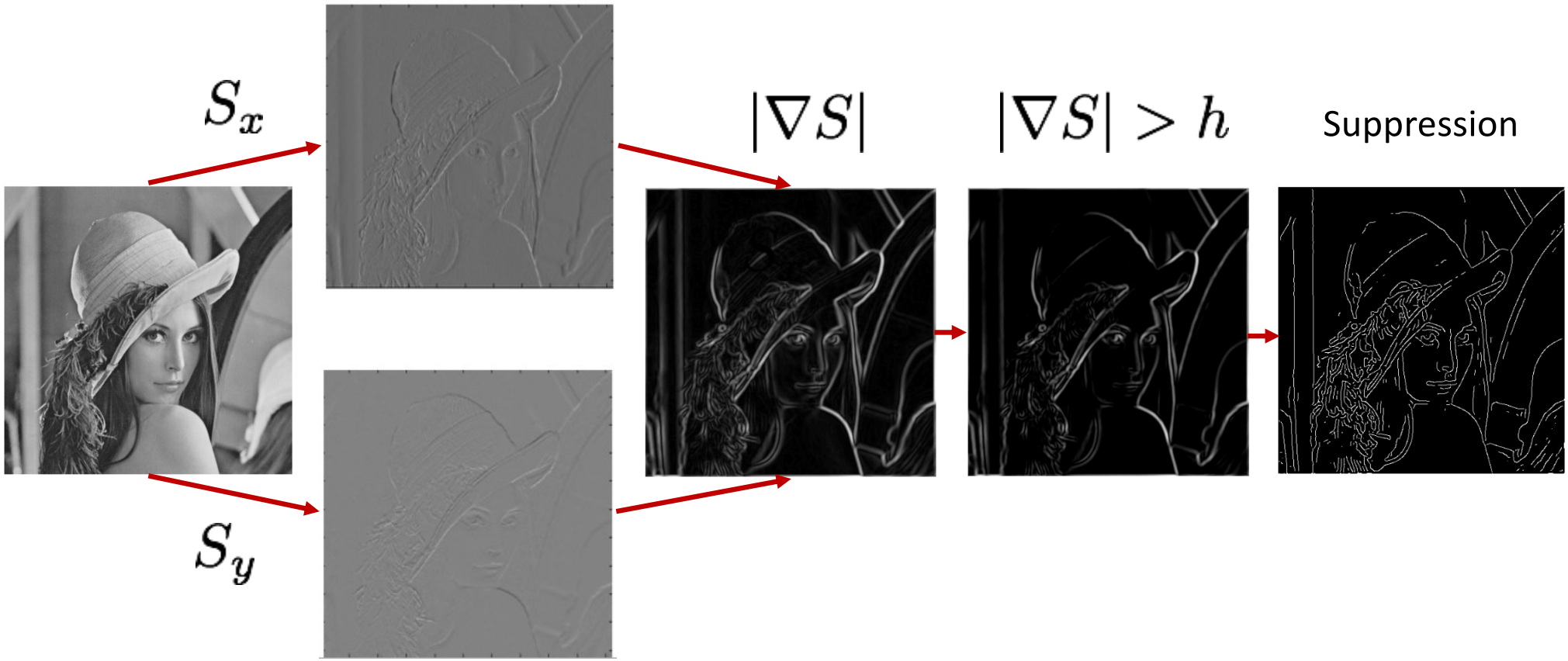
$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Note: masks are mirrored
In convolution

Visualization of Sobel Masks



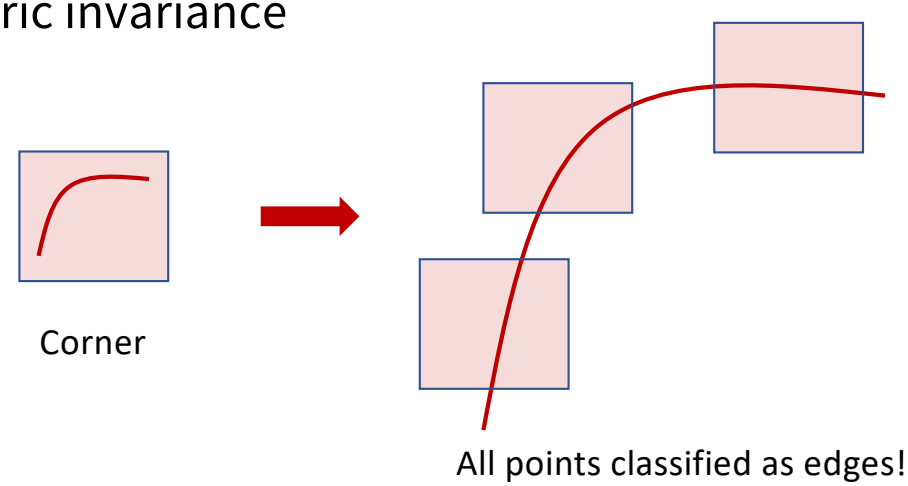
Canny edge detector



Properties of Harris detectors

- Widely used
- Detection is invariant to
  - Rotation -> geometric invariance
  - Linear intensity changes -> photometric invariance

- Detection is not invariant to
  - Scale changes
  - Geometric affine changes

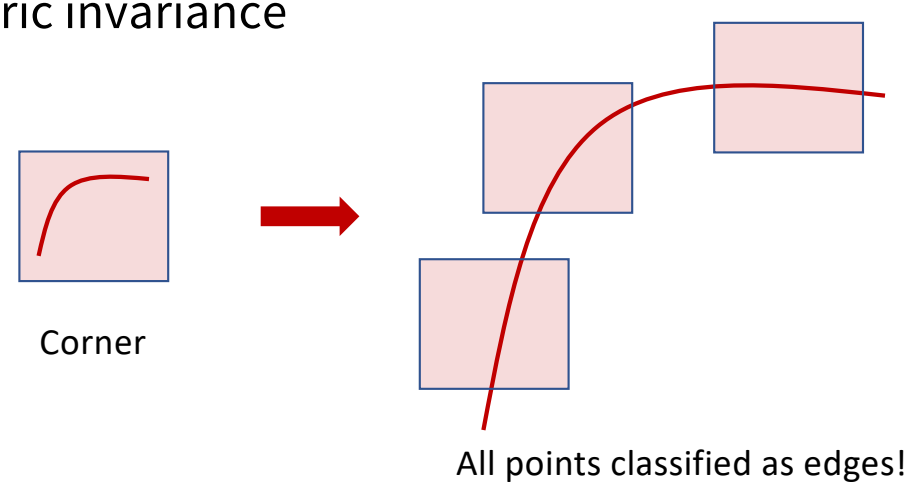


Harris detector: example

Properties of Harris detectors

- Widely used
- Detection is invariant to
  - Rotation -> geometric invariance
  - Linear intensity changes -> photometric invariance

- Detection is not invariant to
  - Scale changes
  - Geometric affine changes



➡ Scale-invariant detection, such as

1. Harris-Laplacian
2. in SIFT (specifically, Difference of Gaussians (DoG))

Popular detectors / descriptors

- SIFT (Scale-Invariant Feature Transformation)
  - Invariant to rotation and scale, but computationally demanding
  - SIFT descriptor is a 128-dimensional vector!
- SURF
- FAST
- BRIEF
- ORB
- BRISK
- LIFT

Classical vs Learning based Features

- Hand engineered features/filter masks (SIFT, ORB, etc)
- SVM for learning based image classification

2012: AlexNet 2012

- Feature masks learned from examples (CNNs)
- Multi-layer CNNs (deep networks)
- For specific classification/regression tasks
- small scale training data

YOLO: 2015

Transformer: 2017
"Attn is all you need"

2018: BERT (Google), GPT-1 (OpenAI)

2021: CLIP 2021 (OpenAI)

- Foundation models
- Very large transformers
- Pre- trained on internet scale data
- Fine-tuned (post-trained) to accomplish specific tasks

2024/2025: SAMv2 (Meta), DINO v3 (Meta), ChatGPT-5 (OpenAI), Imagen 4 (Google), Sora (OpenAI), …

Learning Image Features

Task: Classify whether an image has a cat

Labeled data:

( , 1) $= (x_i, y_i)$

( , 0) $= (x_j, y_j)$

$$D = \{(x_i, y_i)\}_{i=1}^{N} \qquad N \approx 10k$$

Learning Image Features

Task: Classify whether an image has a cat

Labeled data: $D = \{(x_i, y_i)\}_{i=1}^{N}$

Model Architecture: $p(y = 1) = f_\theta(x)$

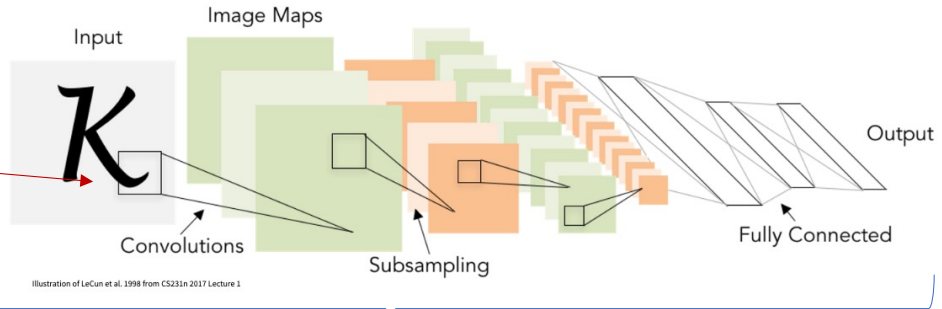
Model

Free parameters (NN weights)

$$I'(u,v) = F * I = \sum_{i=-n}^{n} \sum_{j=-m}^{m} F(i,j) I(u+i, v+j)$$

$$I''(u,v) = \mathrm{ReLU}(I'(u,v))$$

Filter mask parameters



Free parameters θ

Learning Image Features

Task: Classify whether an image has a cat

Labeled data: $D = \{(x_i, y_i)\}_{i=1}^{N}$

Model Architecture: $p(y = 1) = f_\theta(x)$

Training loss function: $l(f_\theta(x_i), y_i)$

E.g., cross entropy loss (classification):

$$l = -\big(y_i \log(\underbrace{f_\theta(x_i)}_{p(y_i=1)}) + (1 - y_i) \log(\underbrace{1 - f_\theta(x_i)}_{p(y_i=0)})\big)$$

E.g., l2 loss (regression):

$$l = \|f_\theta(x_i) - y_i\|^2$$

Learning Image Features

Task: Classify whether an image has a cat

Labeled data: $D = \{(x_i, y_i)\}_{i=1}^{N}$

Model Architecture: $p(y = 1) = f_\theta(x)$

Training loss function: $l(f_\theta(x_i), y_i)$

Optimization problem:

$$\min_\theta \sum_{(x_i, y_i) \in D} l(f_\theta(x_i), y_i)$$

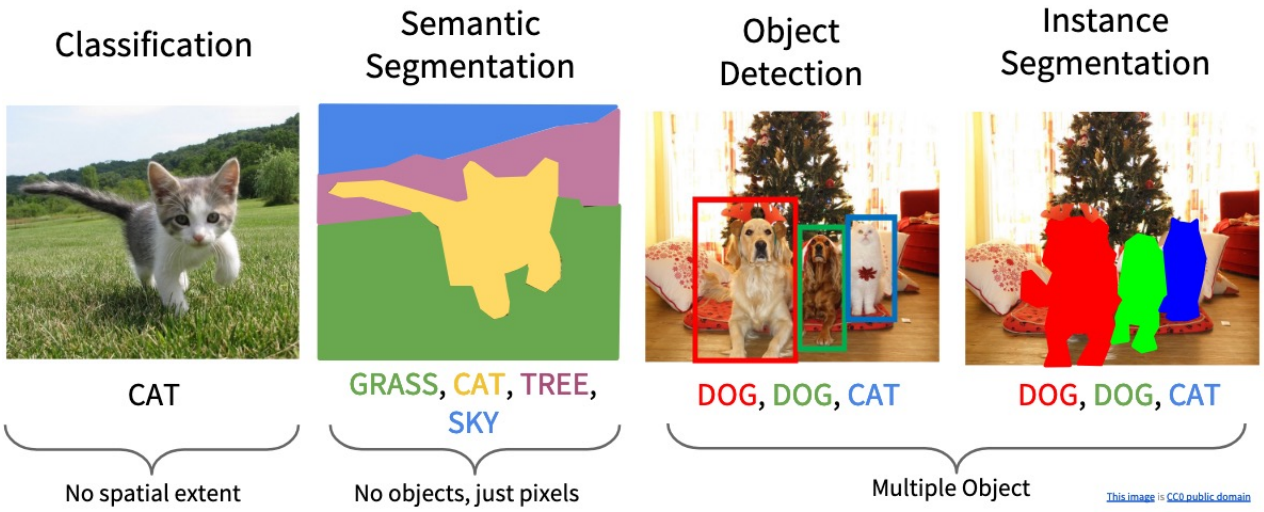
Solve to get: θ^* e.g., with stoch. grad. descent (e.g. Adam) in Pytorch

Learned model: $p(y = 1) = f_\theta^*(x)$

Prob. that a new image x has a cat

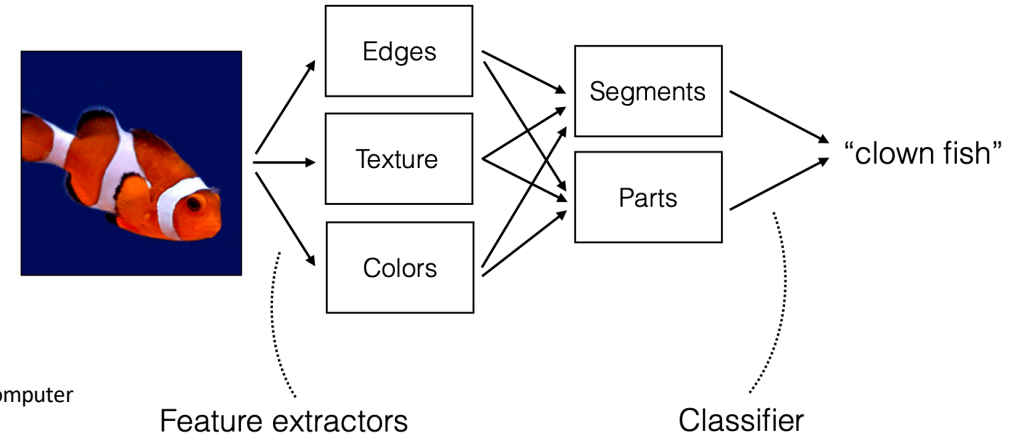
Classic Vision Problems: Semantic

- Image Classification: What's in the image?
- Object Detection: Draw bounding boxes around objects with labels
- Semantic Segmentation: Identify pixels belonging to object classes
- Instance Segmentation: Identify pixels belonging to object instance



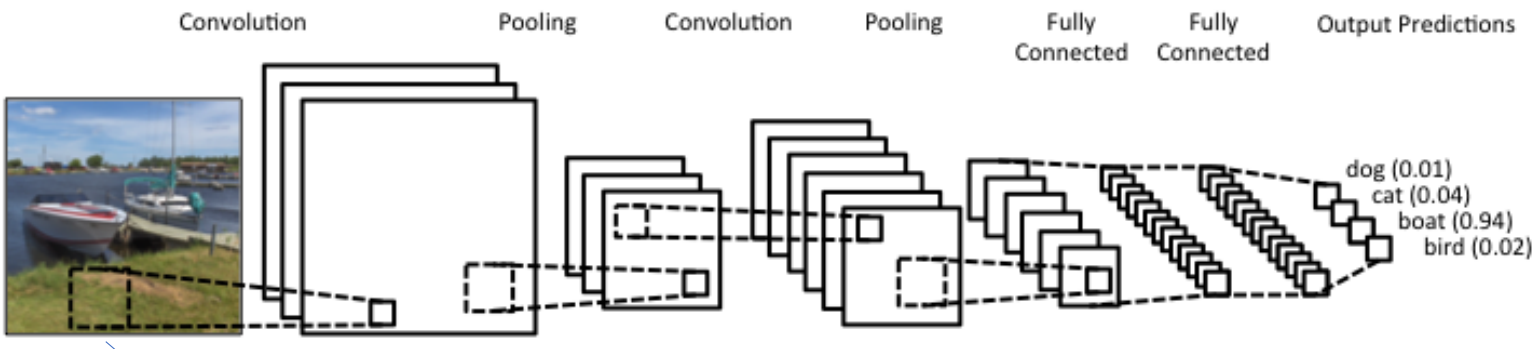
Learned Computer Vision Pipeline: Multiple layers of “feature extractors”

- Multiple layers of “feature extractors”
- Features of features of features – hierarchy of abstractions
- All learned from labeled examples



Example from Advances in Computer Vision – MIT – 6.869/6.819

What are these parameters θ?



$$I'(u,v) = F * I = \sum_{i=-n}^{n} \sum_{j=-m}^{m} F(i,j) I(u+i, v+j)$$

Hand-designed kernels:

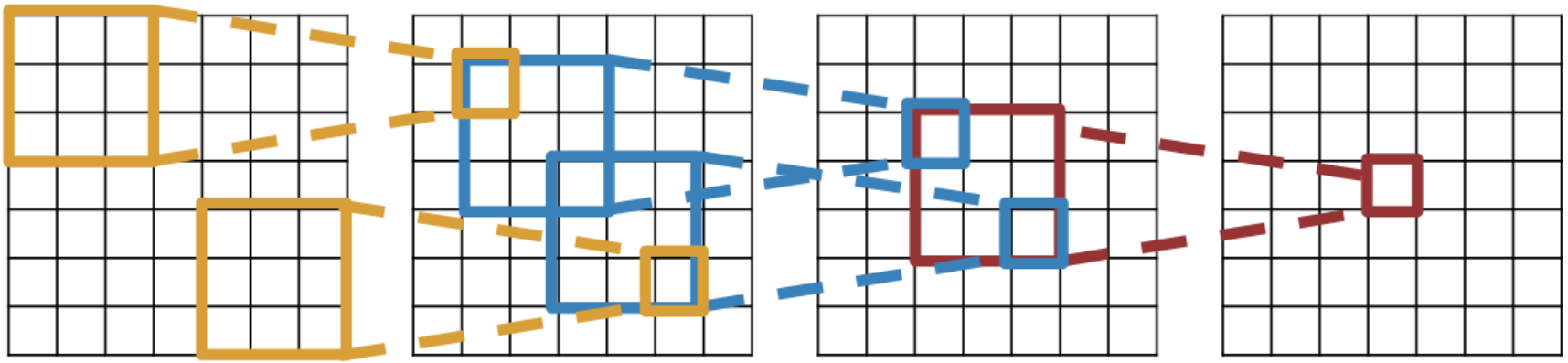
$$F = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Learned kernels:

$$F = \begin{matrix} \theta_{0,0} & \theta_{0,1} & \theta_{0,2} \\ \theta_{1,0} & \theta_{1,1} & \theta_{2,1} \\ \theta_{2,0} & \theta_{2,1} & \theta_{2,2} \end{matrix}$$

Receptive Fields

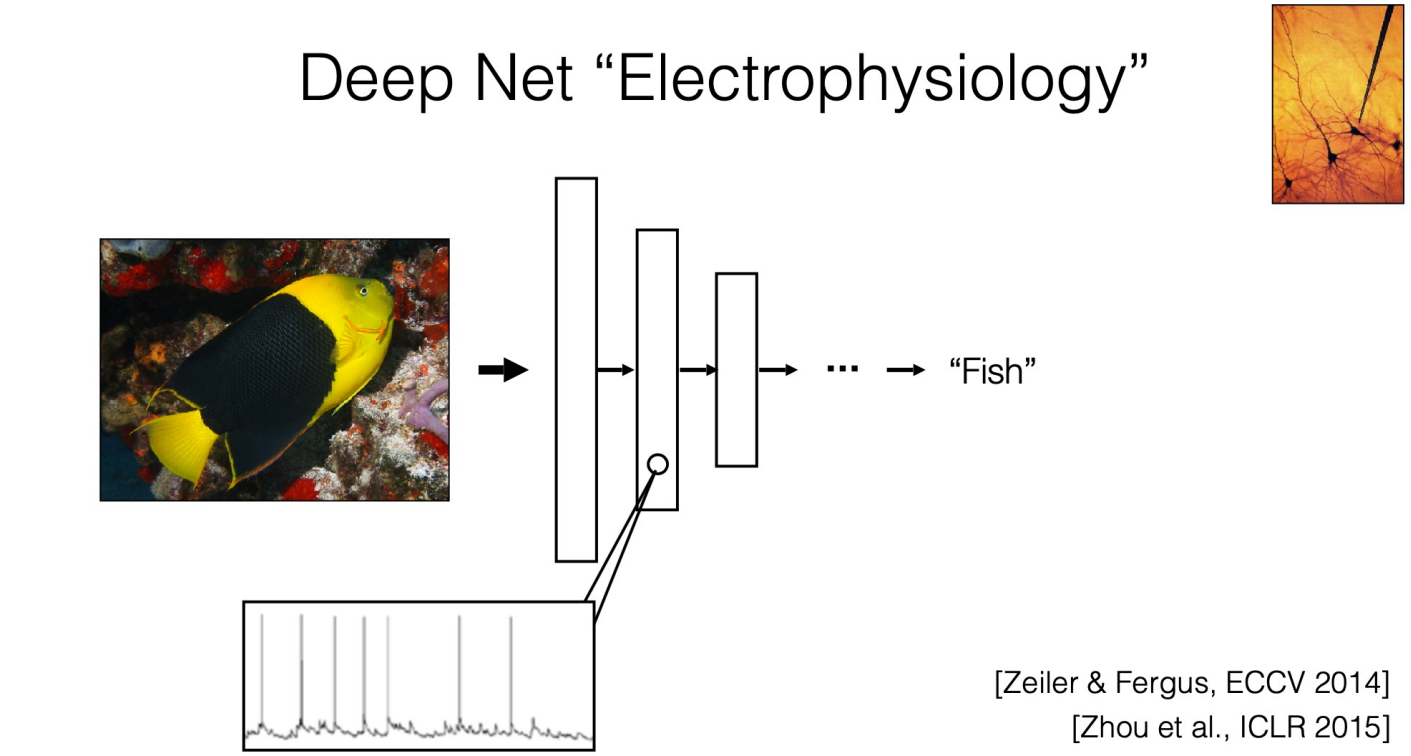
Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)



Be careful – “receptive field in the input” vs. “receptive field in the previous layer”

Slide inspiration: Justin Johnson
From Lecture 5 – CS231n

How do you interpret what the network has learned?



[Zeiler & Fergus, ECCV 2014]
[Zhou et al., ICLR 2015]

Visualizing and Understanding CNNs

[Zeiler and Fergus, 2014]

Gabor-like filters learned by **layer 1**

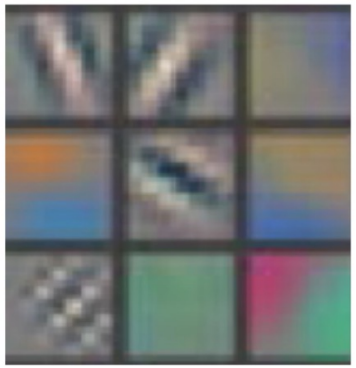
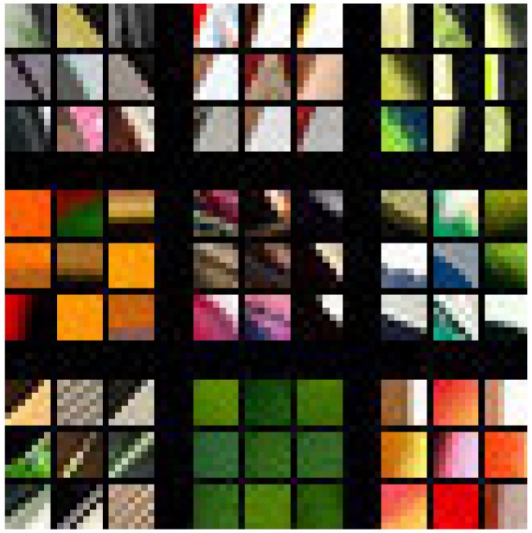


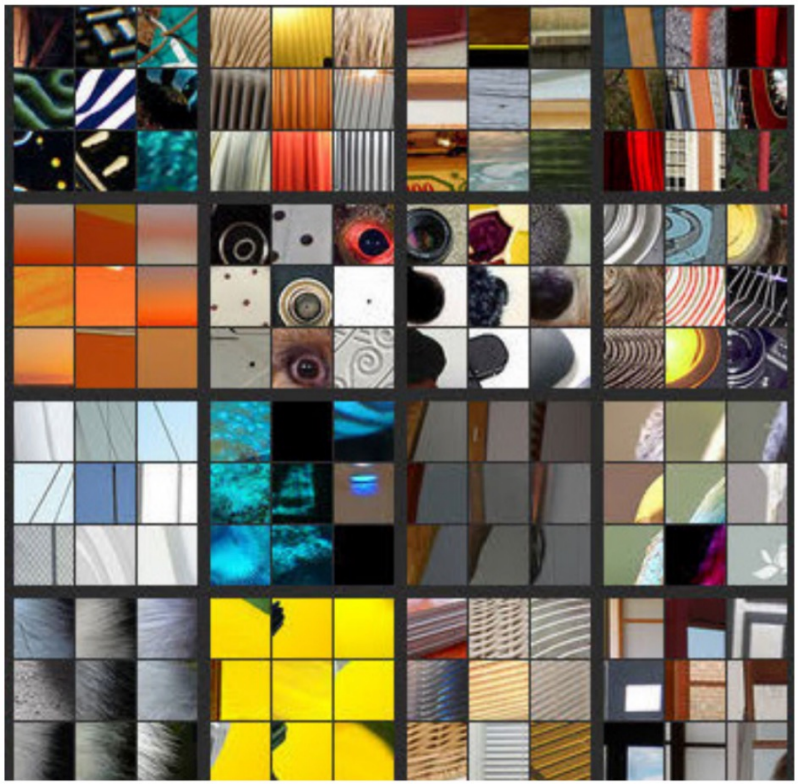
Image patches that activate each of the **layer 1** filters most strongly



Visualizing and Understanding CNNs

[Zeiler and Fergus, 2014]

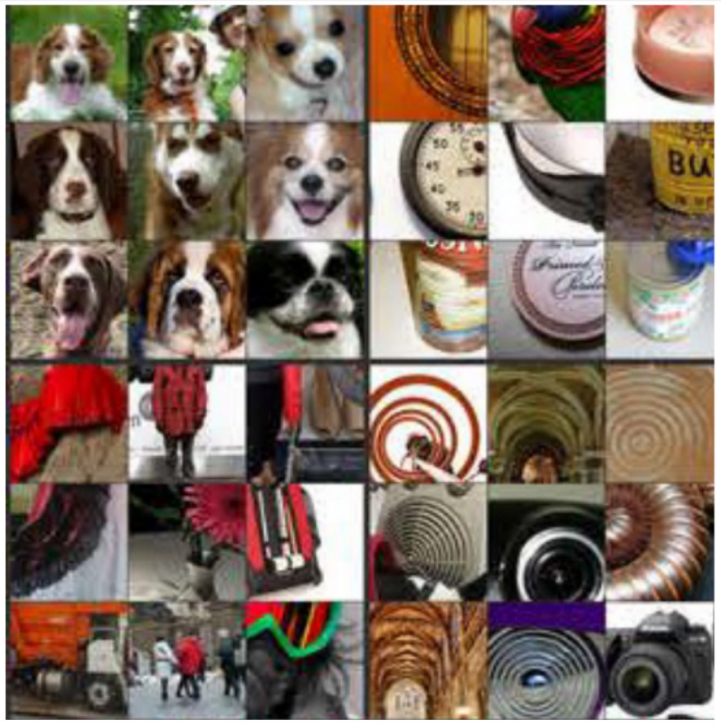
Image patches that activate each of the **layer 2** neurons most strongly



Visualizing and Understanding CNNs

[Zeiler and Fergus, 2014]

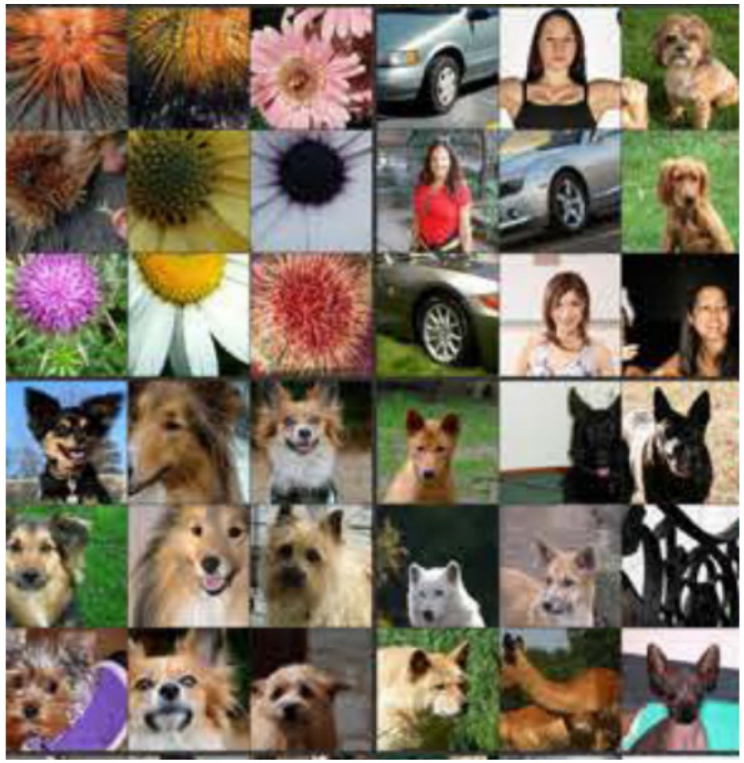
Image patches that activate each of the **layer 4** neurons most strongly



Visualizing and Understanding CNNs

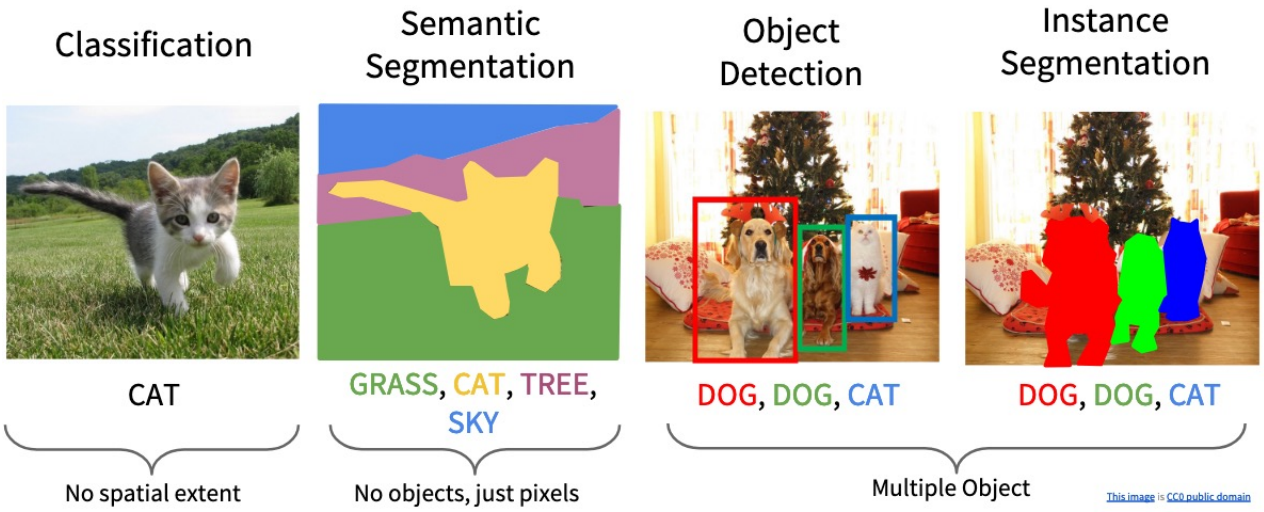
[Zeiler and Fergus, 2014]

Image patches that activate each of the **layer 5** neurons most strongly

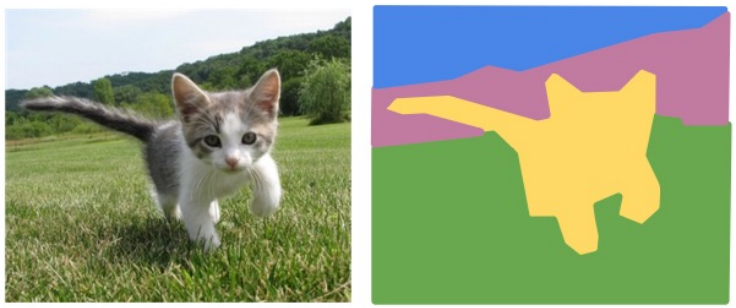


Semantic Perception

- Image Classification: What's in the image? ☑
- Object Detection: Draw bounding boxes around objects with labels
- Semantic Segmentation: Identify pixels belonging to object classes
- Instance Segmentation: Identify pixels belonging to object instance

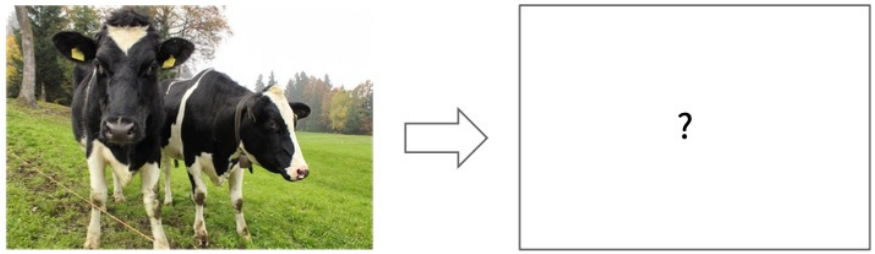


Semantic Segmentation: The Problem



GRASS, CAT, TREE,
SKY, ...

Paired training data: for each training image, each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

Semantic Segmentation Idea: Sliding Window

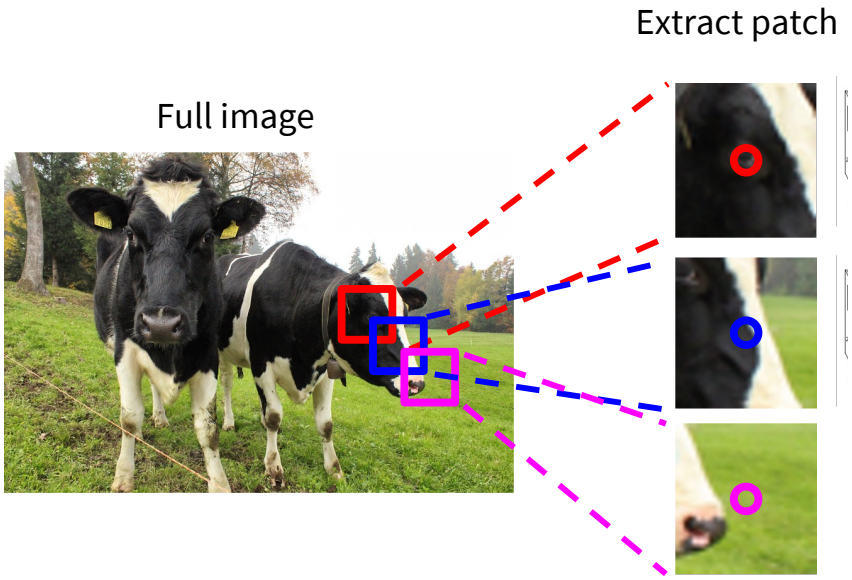
Full image

Impossible to Classify without Context!

Q: How do we include Context?

Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

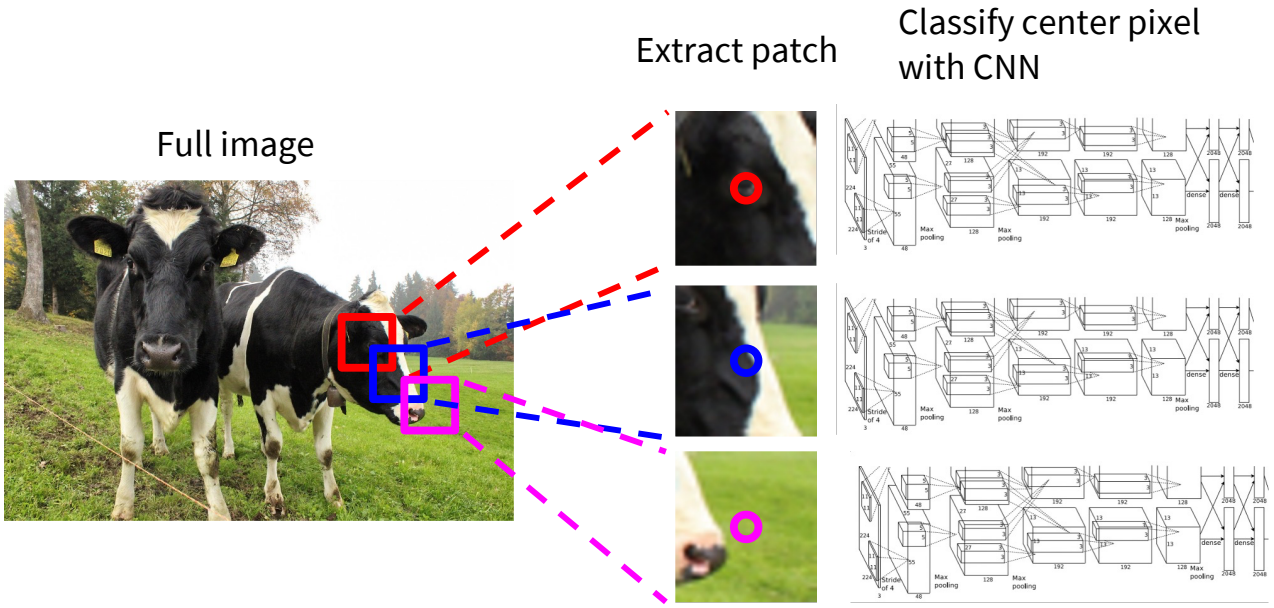
Semantic Segmentation Idea: Sliding Window



Q: How do we model this?

Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

Semantic Segmentation Idea: Sliding Window



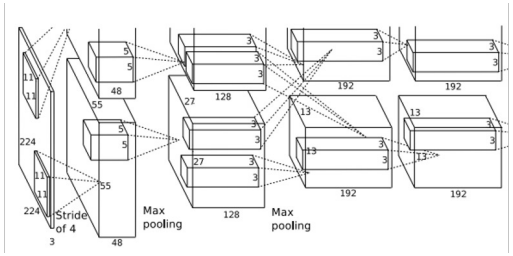
Problem: Very Inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

Semantic Segmentation Idea: Convolution

Full image



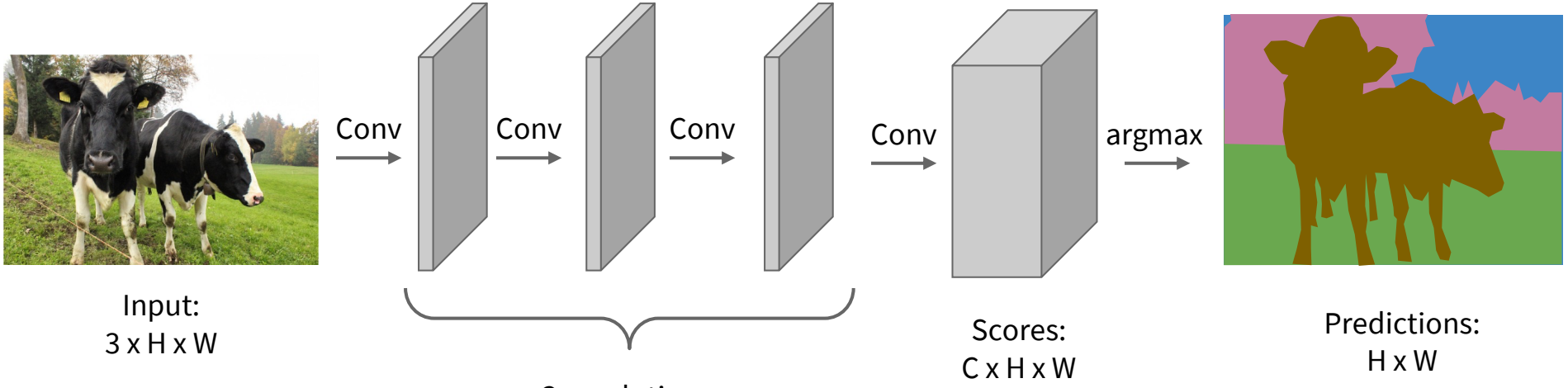
An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!

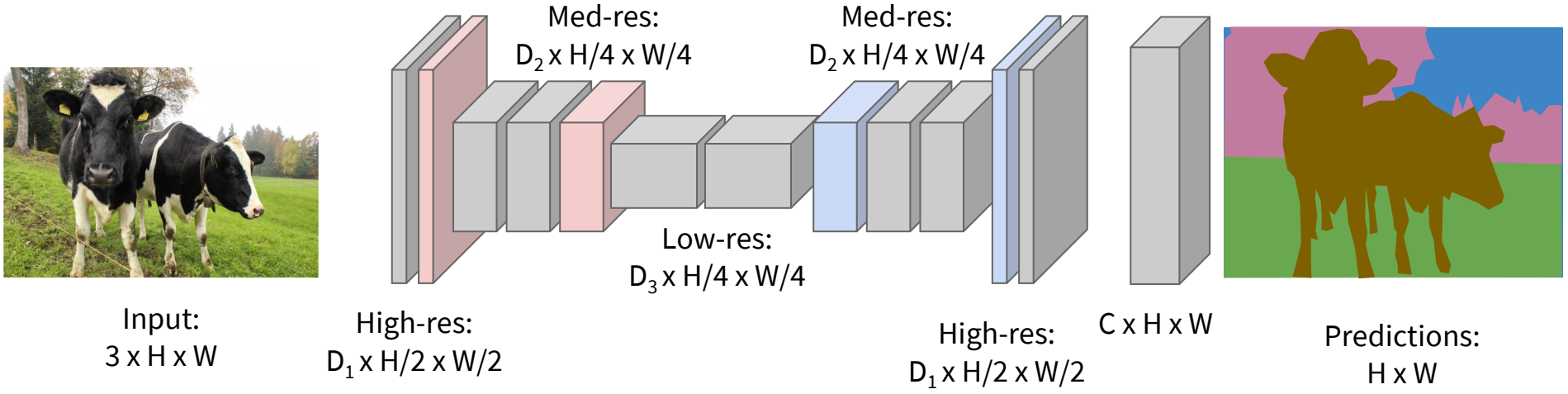


Problem: Convolutions at original image resolution will be very expensive

Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

Semantic Segmentation Idea: Fully Convolutional

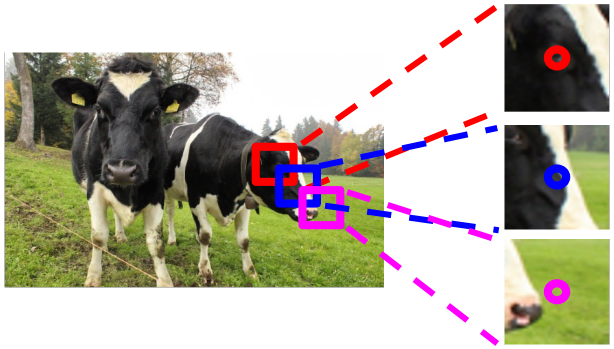
Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

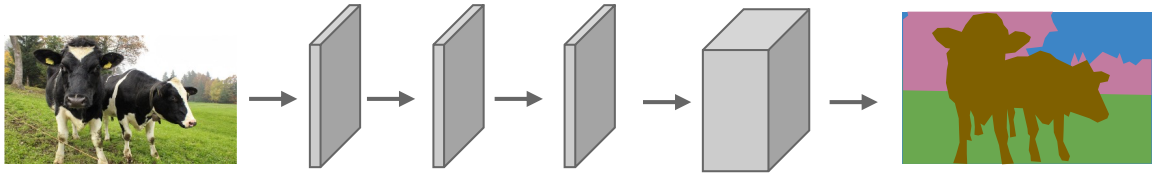


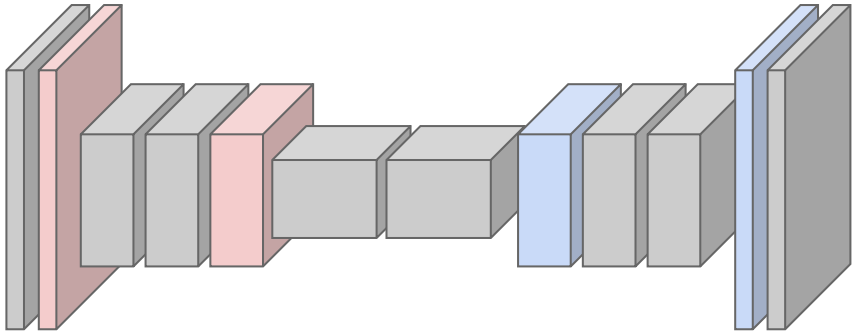
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

Semantic Segmentation: Summary



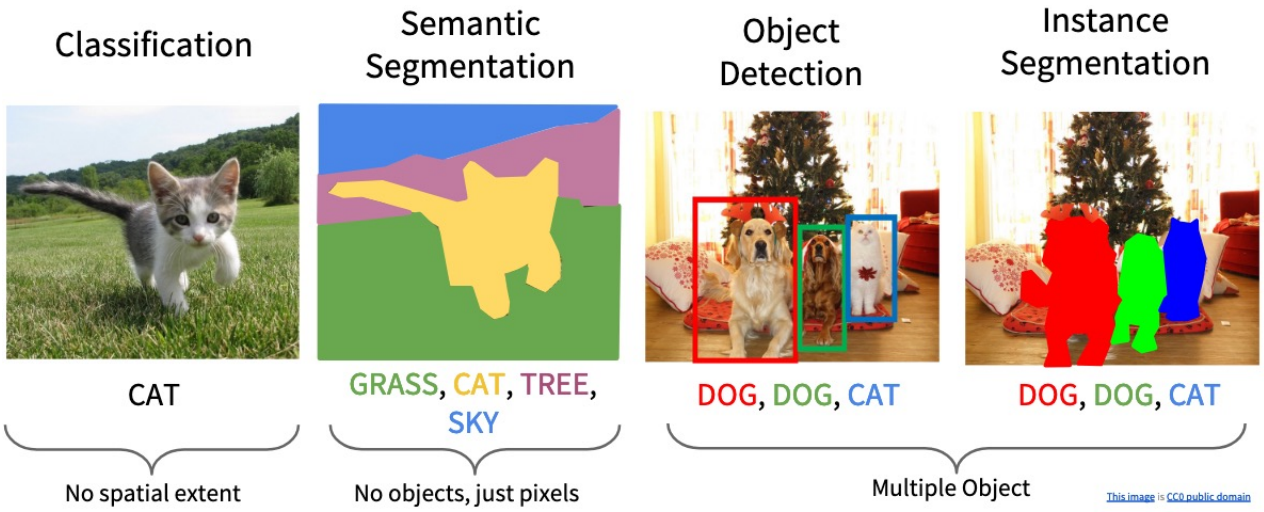




Adapted from Lecture 9 – CS231n. Fei-Fei Li, Ehsan Adeli

Semantic Perception

- Image Classification: What's in the image? ☑
- Object Detection: Draw bounding boxes around objects with labels
- Semantic Segmentation: Identify pixels belonging to object classes ☑
- Instance Segmentation: Identify pixels belonging to object instance



Semantic Perception

Take cs231n for more details and more advanced techniques!