EE655: Computer Vision & Deep Learning

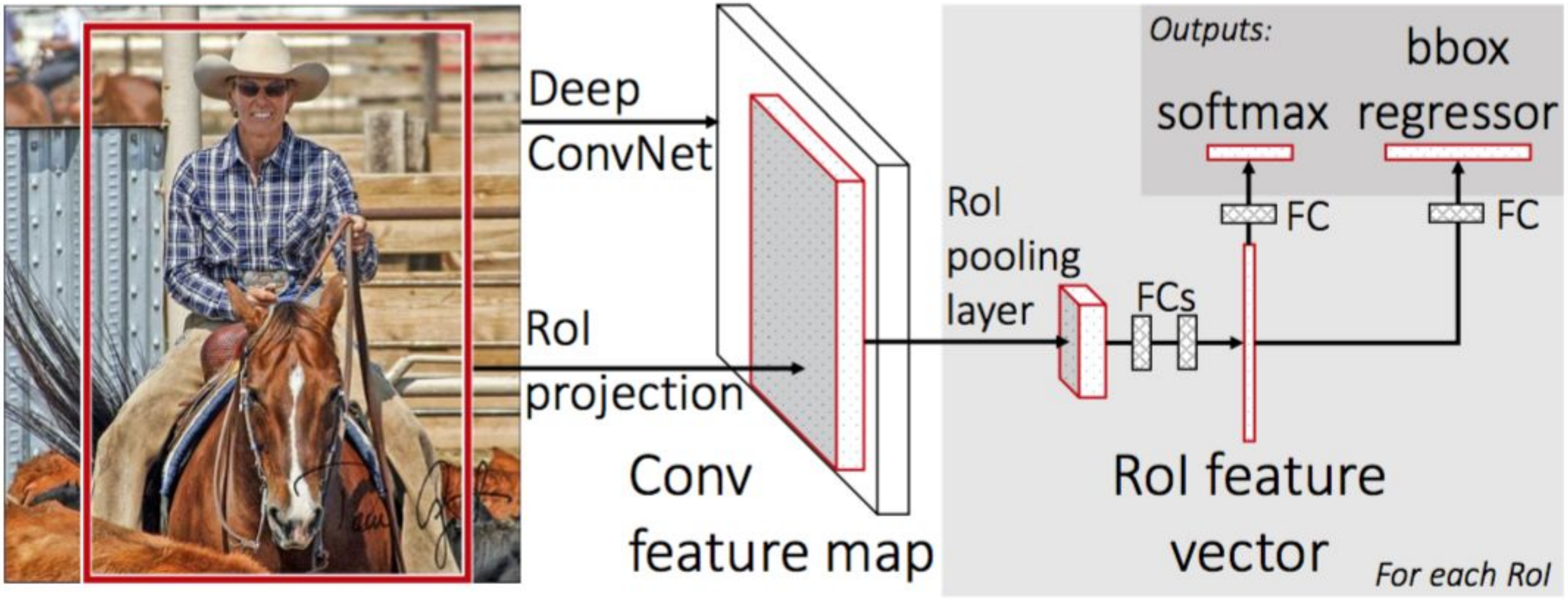
Lecture 13

Koteswar Rao Jerripothula, PhD
Department of Electrical Engineering
IIT Kanpur

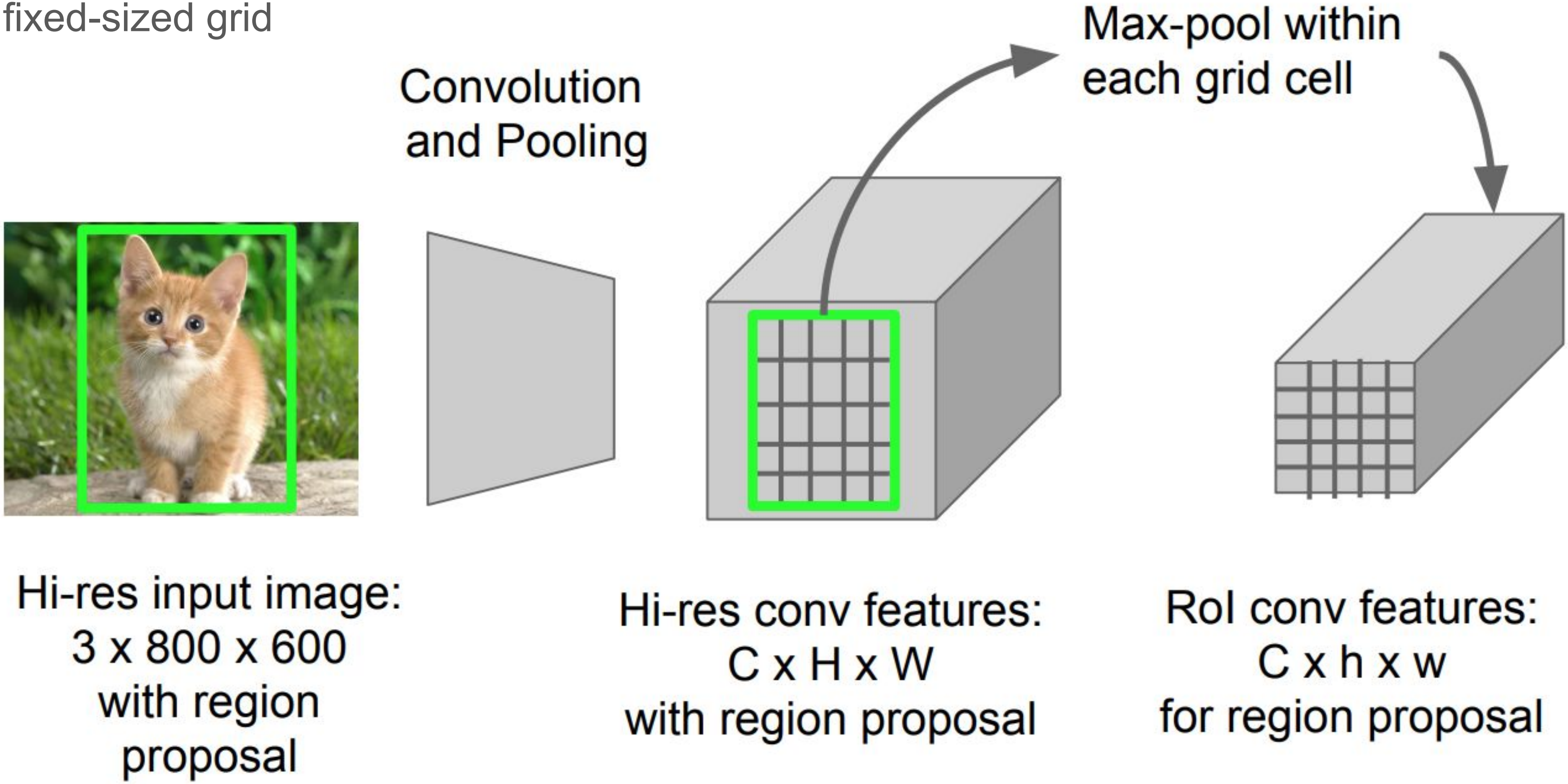
Outline

Fast R-CNN

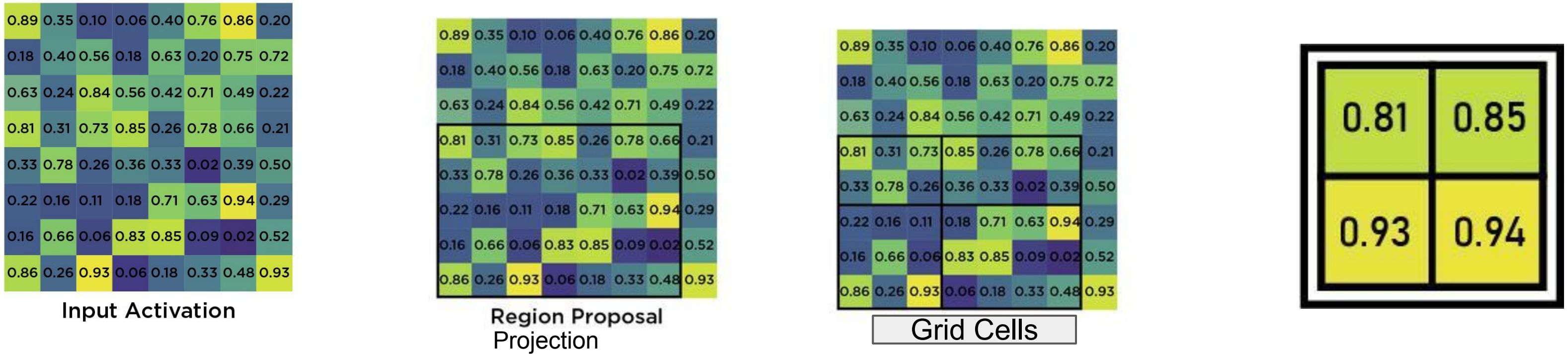
Video Processing


Deep
ConvNet
RoI
projection
Conv
feature map
Outputs:
softmax
bbox
regressor
FC
FC
RoI
pooling
layer
FCs
RoI feature
vector
For each RoI

RoI pooling idea: Pooling within grid cells of fixed-sized grid



Let the grid-size be 2x2



Training and Loss Function

First, we take each training region of interest labeled with ground truth class u and ground truth bounding box v. Then we take the output generated by the softmax classifier and bounding box regressor and apply the loss function to them. We defined our loss function such that it takes into account both the classification and bounding box localization. This loss function is called multi-task loss. This is defined as follows:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v),$$

Multi-task Loss

Square bracket

where $\mathbf{L_{cls}}$ is classification loss, and $\mathbf{L_{loc}}$ is localization loss. lambda is a balancing parameter and ▯ is a function (the value of ▯ =0 for background, otherwise ▯ =1) to make sure that loss is only calculated when we need to define the bounding box. Here, $\mathbf{L_{cls}}$ is the log loss and $\mathbf{L_{loc}}$ is defined as

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Loss function of Fast R-CNN model

Predicting the same transformation parameters as R-CNN

Video Processing

- Background Subtraction
- Optical Flow

Background Subtraction

- ❑Frame – Initial Frame
- ❑Frame – Average Frame
- ❑Frame Difference

❑Post Processing:

Take absolute

Apply Otsu algorithm

(Frame – Initial Frame)

- Assumes object doesn't exist in the initial frame
- Assumes only changes is object's presence

(Frame-Average Frame)

- Assumes objects are mostly moving, so that average comprises of only the background
- Assumption: Object is sufficiently different from the background in appearance

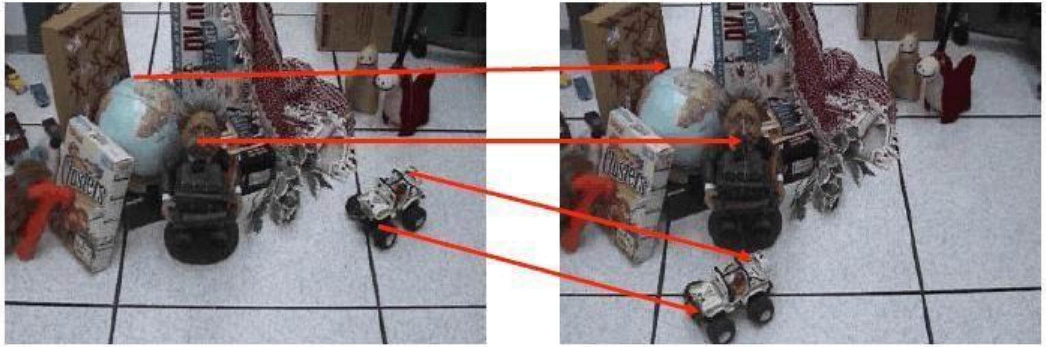
Frame Difference

- Highlights any motion between two frames
- Good for heterogeneous regions
- Struggles with homogeneous regions

Optical flow

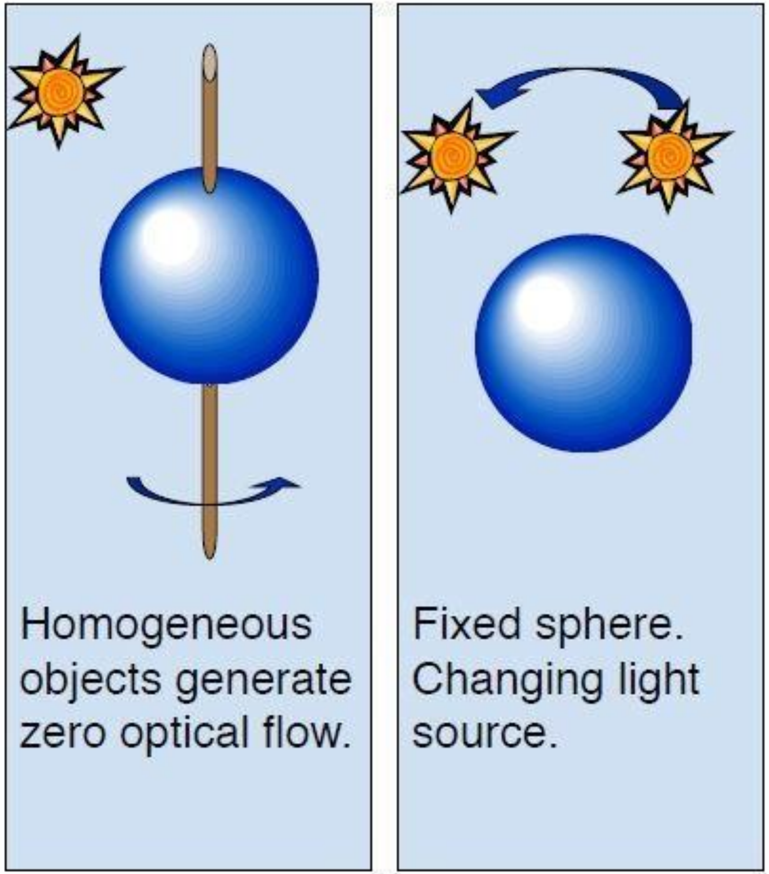
- Definition: optical flow is the *apparent* motion of brightness patterns in the image

- Have to be careful: apparent motion can be caused by lighting changes without any actual motion



Where did each pixel in image 1 go to in image 2

Where it breaks

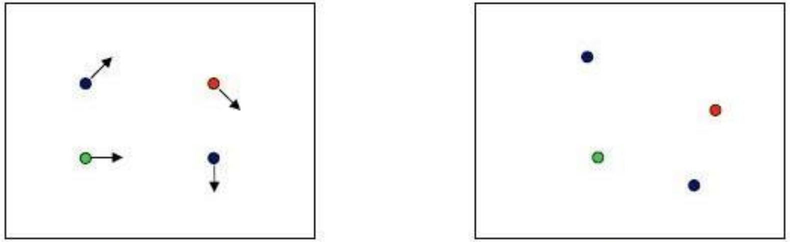


The Optical Flow Field

Still, in many cases it does work….

- Goal:
  Find for each pixel a velocity vector $\vec{\mathrm{u}} = (u, v)$
  which says:
  - How quickly is the pixel moving across the image
  - In which direction it is moving

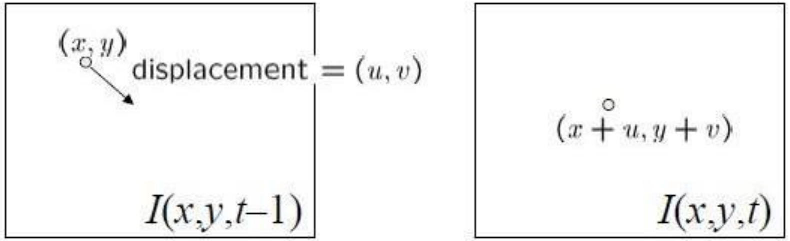
Estimating optical flow



$I(x,y,t-1)$ $\quad$ $I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field between them.
- Key assumptions
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

The brightness constancy constraint



- Brightness Constancy Equation:

$$I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$$

Can be written as:

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$I(x, y, t-1) \approx I(x, y, t) + I_x \cdot u(x, y) + I_y \cdot v(x, y)$$

So, $$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

The brightness constancy constraint

$$I_x \cdot u + I_y \cdot v + I_t = 0$$

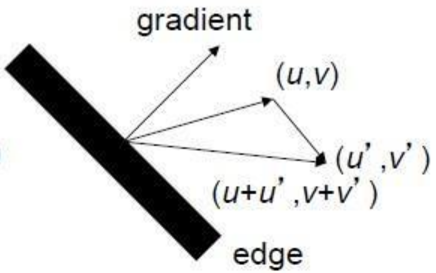
- How many equations and unknowns per pixel?
  - One equation, two unknowns

- Intuitively, what does this constraint mean?

$$\nabla I \cdot (u, v) + I_t = 0$$

- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown

If (u, v) satisfies the equation,
so does $(u+u', v+v')$ if $\nabla I \cdot (u', v') = 0$



- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \ v]$$

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

$$\underset{25\times2}{A} \ \underset{2\times1}{d} = \underset{25\times1}{b}$$

Prob: we have more equations than unknowns

$$\underset{25\times2}{A}\;\underset{2\times1}{d} = \underset{25\times1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

$$\underset{2\times2}{(A^T A)}\;\underset{2\times1}{d} = \underset{2\times1}{A^T b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lucas & Kanade (1981)