DEFORMABLE PARTS MODEL

Source: Ross Girshick

- Star Model: There is a "root" part, and you only have edges between the root part and all other parts.

\[ \text{Score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} \hat{m}(p_i) - \sum_{(i,j) \in E} d_{ij}(p_i, p_j) \]

\[ \text{Score}(p_0) = \max_{\langle p_1, p_2, \ldots, p_n \rangle} \hat{m}(p_0) + \sum_{i=1}^{n} m_i(p_i) - d_i(p_i, p_0) \]

- So you are searching over locations for the root part, and its score is given by maximizing over the relative locations of other parts.
- Corresponds to doing \( n \) independent maximizations over each \( p_i, i > 0 \).
DEFORMABLE PARTS MODEL

Source: Ross Girshick


- Star Model: There is a "root" part, and you only have edges between the root part and all other parts.

\[
\text{Score}(p_0, \ldots, p_n) = \sum_{j=0}^{n} m_j(p_j) - \sum_{(i,j) \in E} d_{ij}(p_i, p_j)
\]

\[
\downarrow
\]

\[
\text{Score}(p_0) = m_0(p_0) + \sum_{j=1}^{n} \max_{p_j} m_j(p_j) - d_i(p_i - p_0)
\]

- So you are searching over locations for the root part, and it's score is given by maximizing over the relative locations of other parts.

- Corresponds to doing \(n\) independent maximizations over each \(p_i, i > 0\).

- If \(d_j\) is set so that the cost is infinite beyond a search window \(P \times P\) (around \(p_0\)), then you only have to consider \(P^2\) candidates for each part, for a given root part location \(p_0\).

DEFORMABLE PARTS MODEL

Source: Ross Girshick


\[
S[x, y] = \text{Conv}(F, \Phi_0)[x, y] + \sum_{j=1}^{n} \max_{j' \not= j} \text{Conv}(F, \Phi_j)[x', y'] - d_j(x' - x, y' - y)
\]

- The \(m_j\) all come from linear classifiers on a HoG feature map \(F[x, y, :]\).

\[
m_j[x, y] = \text{Conv}(F, \Phi_j)
\]

Now, let’s assume that for each part, there is a 'canonical' optimal displacement \((\delta_i, \delta_j)\), and the penalty \(d_j\) is 0 if \((x' - x, y' - y)\) within a \(P_j \times P_j\) window around this displacement, and infinity otherwise.
DEFORMABLE PARTS MODEL

Source: Ross Girshick

Training: We need to learn the part filters \( m_i(\cdot) \) as well as the spring / distance functions \( d_i \).
- Would be simple if someone gave us explicit labels for parts. Then we could train each \( m_i, d_i \) separately.
- In practice, we only have labels for the whole object (windows that contain an object and those that do not).
- This is a "latent SVM" model: you don’t have outputs for the individual classifiers \( m_i \).
- Paper describes a way to learn this using alternating minimization (with careful initialization).

Score \( (p_0) = \max_{\psi_1(p_1) \cdots \psi_n(p_n)} m_0(p_0) + \sum_{j=1}^n m_j(p_j) - \sum_{j=1}^n d_j(p_j - p_0) \)

- What are components of this algorithm:
  - Hand-crafted SIFT feature extraction.
  - Computing responses to local filters \( m_i \).
  - Allowing 'slack' in part locations by maximizing scores in area around expected location of part.
  - Combining these individual part scores to score the entire object.
- Compare to a neural network
  - Many Conv Layers
  - Conv Layer
  - Max Pool
  - Conv / FC layer.

Girshick et al., "Deformable Part Models are Convolutional Neural Networks," CVPR 2015.
Given an input image, come up with a set of possible detection windows.
Crop out that window, reshape it to a standard size, and then provide it as input to a classification network.
Pick the highest scoring object, and if its score is higher than a threshold, report the detection.

Classification Network:
- First pre-train a network for image classification.
- Then fine-tune it for object detection.

How do you get region/bounding box proposals?
- Original R-CNN paper used a graphical model to setup a binary segmentation task.

Place "seeds" at regular intervals in the image.
Select a window around each seed.
Do a foreground/background segmentation where label of seed is set to be 1, and of boundary of box to 0. (Placed in unary cost).
Edge-wise costs like we talked about. Unary costs also include distribution of colors in foreground vs background.

Add a constant bias in unary favoring foreground to background. Different values of this bias will give you different segmentations.
The paper includes a classifier to 'rank' different segmentations.
R-CNN takes the top N proposals (using the smallest bounding box around the detected foreground pixels).
CNN-BASED OBJECT DETECTION


R-CNN also has a “bounding box regression” output that refines the object window.
- Outputs locations of the four corners of the bounding box (relative to the cropped and resized input).
- But this is expensive: need to re-run the full network on all bounding box proposals (~2k).

CNN-BASED OBJECT DETECTION


Have a common conv-net which gives you a ‘feature’ tensor for the entire image.
- Then for each proposed bounding box, do a max-pooling of features within that box to get a single vector.
- Have a bunch of fully-connected layers to produce the final predictions for that proposed box.

CNN-BASED OBJECT DETECTION

Ren et al., “Faster R-CNN,” NIPS 2015

Use a neural network to also output box proposals (called a Region Proposal Networks).
- At each location, the network considers \( k \) possible “templates” for the box, centered at that location, and of different sizes and aspect ratios.
- For each template, does a binary classification of “objectness”. (cls layer).
- Also outputs precise co-ordinates for each proposal.
- At test time, do non-maxima suppression to select proposals. Apply RoI pooling on the selection.
- RPN and the classification network share layers.

CNN-BASED OBJECT DETECTION


Basic idea, divide the image into an \( S \times S \) grid. \( S=7 \) in their paper).
- For each member of the grid, the network will produce proposals for \( B \) bounding boxes. \((B=2)\).
- The bounding box need not be contained in the grid-square. But it is the job of the grid that most overlaps a bounding box to predict it.
- Essentially means that you are limiting your bounding box size to be no larger than twice grid-box width and height.
CNN-BASED OBJECT DETECTION


- So at each grid, you predict properties of \( B \) bounding boxes. These properties include 4 co-ordinates, and a confidence score that it is a valid bounding box for any object.
- But at each grid, you also predict a class distribution. This is a common class distribution that applies to all bounding boxes.
  - Assumption: Two bounding boxes of different objects will not have the same ‘responsible’ grid square.

At test time, object probability of a box (from all grid members) is object probability from grid square \( x \) object-ness confidence of that particular box.
- Select all class+box predictions that cross a certain threshold.
- Strong assumptions about number / overlap of true bounding boxes. But is much faster.

IMAGE SEGMENTATION

- We have talked about the image segmentation task as labeling each pixel according to an object class.

So it’s labeled all car pixels as car.
- But there are multiple cars in this image!

Contrast this to object detection where you are able to detect multiple instances of the same object in one image.

IMAGE INSTANCE SEGMENTATION

- We want to get a label for each pixel, which tells us not only object category, but also a separate label for each pixel.

But how do we turn this into a classification task? What is the ground-truth label for each instance of a class in the image?

Simplest Solution: Get region proposals, do an independent segmentation inside each bounding box.

Box Proposals (these guys used CPMC) are based on "object-ness". Each box has one dominant object (there one instance of some class).

Network output is: a foreground / background segmentation, and a single class label for the box that applies to all pixels labeled foreground.

During training, you might have pixels of the same class but another instance in the same box. Treat those as background. Foreground only those that are of that class and of the "dominant" instance of the class.