Virtual classrooms

• Virtual lectures on Zoom
  – Only host shares the screen
  – Keep video off and microphone muted
  – But please do speak up (remember to unmute!)

• Virtual interactions on Zoom
  – Ask and answer plenty of questions
  – “Raise hand” feature on Zoom when you wish to speak
  – Post questions on chat window
  – TA will help keep track of raised hands and chat window

• Lectures recorded and upload on Kaltura
  – Available under “Media Gallery” on Canvas
Course details

• Class webpage:
  – http://cseweb.ucsd.edu/~mkchandraker/classes/CSE152B/Spring2020/

• Instructor email:
  – mkchandraker@eng.ucsd.edu

• TA: Rui Zhu
  – Emails: rzhu@eng.ucsd.edu

• Aim is to learn together, discuss and have fun!
Announcements

• Finals: will be a no-fault exam
  – Assign 40% to maximum of \{mid-term, final\}, 25% to the other

  ![Final Exam Scoring Preference](attachment:final_exam_scoring_preference.png)

• Assignments: we will retain highest 2 out of 3

  ![Assignment Scoring Preference](attachment:assignment_scoring_preference.png)
Announcements

• HW3 due date: no-penalty anytime by Jun 13, at 11pm PST

![Pie chart showing distribution of preferences for HW3 due date]

• Make-up exam if you have pressing needs
  – Sickness, unavailable resources, ....
  – Drop instructor an email (CC TA), with supporting material
  – Will schedule a second round with different questions

• Solutions to quizzes, assignments, mid-term on course webpage

CSE 152B, SP20: Manmohan Chandraker
Course Evaluation

• Course evaluation:
  • You should have received an email

• Link: http://cape.ucsd.edu

• Only 25% of class has submitted so far....

• First time the class is offered, your feedback is valuable

• Please take 5 minutes to do this!
Recap
Object Detection

Object 1: \((x_1, y_1, w_1, h_1)\), dog
Object 2: \((x_2, y_2, w_2, h_2)\), cat
Object 3: \((x_3, y_3, w_3, h_3)\), dog
Object 4: \((x_4, y_4, w_4, h_4)\), cat
Object 5: \((x_5, y_5, w_5, h_5)\), dog

Need to handle outputs of variable lengths
Sliding Windows

- Need to consider windows at several different positions and scales
- Either use a simple feature extractor, or evaluate only few candidates
Region Proposals

- Find blob-like regions of image that might be objects
- Do not consider object type and tolerate high rate of false positives
R-CNN

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

- Expensive
  - No feature reuse
  - Full forward pass of CNN for each region proposal
- Classification and regression are disjoint from feature extraction
  - CNN features not updated together with detection
- Complex training pipeline

[Girshick et al., Rich feature hierarchies]
R-CNN Regression

**Step 5** (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals.

Train a regressor to find offset function $d_i$ that reaches closest to regression target:

$$
\mathcal{L}_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} (t_i - d_i(p))^2 + \lambda \|w\|^2
$$

Regression targets:

$$

t_x = (g_x - p_x) / p_w
\quad
t_y = (g_y - p_y) / p_h
\quad
t_w = \log(g_w / p_w)
\quad
t_h = \log(g_h / p_h)
$$
Issues with R-CNN

- Very expensive for inference
  - Full forward pass of CNN for each region proposal

- Classification and regression are disjoint from feature extraction
  - CNN features not updated together with detection

- Complex training pipeline

- Need mechanism to “connect” CNN features to classifier and regressor
Fast R-CNN

- Extract convolutional features just once for whole image
- Need method to share computation of convolutional layers between proposals for an image
- Allows several advantages
  - Faster inference
  - Connects CNN to classifier
  - Easier training
- Proposals still from selective search

[Girshick, Fast R-CNN]
Fast R-CNN: RoI Pooling

Hi-res input image: 3 x 800 x 600 with region proposal

Convolution and Pooling

Hi-res conv features: C x H x W with region proposal

Problem: Fully-connected layers expect fixed size low-res features: C x h x w
Fast R-CNN: RoI Pooling

Hi-res input image: 3 x 800 x 600 with region proposal

Convolution and Pooling

Hi-res conv features: C x H x W with region proposal

Project region proposal onto conv feature map

Fully-connected layers

**Problem:** Fully-connected layers expect fixed size low-res features: C x h x w
Fast R-CNN: RoI Pooling

- Hi-res input image: 3 x 800 x 600 with region proposal
- Hi-res conv features: C x H x W with region proposal
- Divide projected region into h x w grid
- Problem: Fully-connected layers expect fixed size low-res features: C x h x w
Fast R-CNN: RoI Pooling

Hi-res input image: 3 x 800 x 600 with region proposal

Convolution and Pooling

Hi-res conv features: C x H x W with region proposal

Max-pool within each grid cell

RoI conv features: C x h x w for region proposal

Fully-connected layers

Problem: Fully-connected layers expect fixed size low-res features: C x h x w

RoI Pooling

\[ y_{rj} = x_{i^*(r,j)} \]

(RoI r, output j)

\[ i^*(r,j) = \arg\max_{i' \in \mathcal{R}(r,j)} x_{i'} \]

Pooling region
Fast R-CNN: RoI Pooling

- **Convolution and Pooling**
  - Hi-res input image: 3 x 800 x 600 with region proposal
  - Hi-res conv features: C x H x W with region proposal
  - RoI conv features: C x h x w for region proposal

- **Fully-connected layers**
  - Can back propagate similar to max pooling

- **Problem**: Fully-connected layers expect fixed size low-res features: C x h x w

- **RoI Pooling**
  \[ y_{rj} = x_{i^*(r,j)} \]
  \[ i^*(r,j) = \text{argmax}_{i \in R(r,j)} x_i \]
  Pooling region

- **Backpropagation**
  \[ \frac{\partial L}{\partial x_i} = \sum_r \sum_j \left[ i = i^*(r,j) \right] \frac{\partial L}{\partial y_{rj}} \]
  Gradient accumulated if \( i \) is the index chosen for max pooling
Fast R-CNN: Improvements over R-CNN

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time:</td>
<td>84 hours</td>
<td>9.5 hours</td>
</tr>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
Object Detection Evaluation: Precision and Recall

Query

$$\text{precision} = \frac{\text{Number of relevant}}{\text{Number of returned}}$$

$$\text{recall} = \frac{\text{Number of relevant}}{\text{Number of total relevant}}$$

Database size: 10 images
Relevant (total): 5 images

Results (ordered):

[Ondrej Chum]
Object Detection Evaluation: Scoring a Bounding Box

If prediction and ground truth are bounding boxes, when do we have a correct detection?
We say the detection is correct (a “true positive”) if the intersection of the bounding boxes, divided by their union, is greater than a threshold.

\[ a_o = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \]

\[ a_o > 0.5 \quad \text{correct} \]
Object Detection Evaluation: Average Precision

- Ordering determined by detector confidence (say, softmax score)
- Choose the IoU threshold
- Plot the precision-recall curve
- Average precision: area under the curve
- Mean AP: average AP across all categories
- AP-k: AP value with k% IoU
- Another metric: report the average of AP-k, for various k = [50, 55, ..., 95].
Faster R-CNN

- Proposals still from separate mechanism in Fast R-CNN
- Insert a Region Proposal Network (RPN) after last convolutional layer
- RPN trained to produce region proposals directly, no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and regressor just like Fast R-CNN

[Ren et al., Faster R-CNN]
Faster R-CNN: Region Proposal Network

- Slide a small window on the feature map
- Build a small network for:
  - classifying object or not-object
  - regressing bounding box locations
- Position of sliding window gives location information with respect to the image
- Box regression gives finer localization with respect to this sliding window
Faster R-CNN: Need for Multiscale Proposals

- Problem with region proposal using a convolutional network
  - Multiple scales of objects possible

- Possible solutions
  - Multiple scaled images
  - Filters of different sizes (usually together with above)

- But such approaches are time-consuming.
Faster R-CNN: Anchors

• Consider single object localization
  • Simply have a classification and regression head

• But now consider multiple objects, possibly overlapping
  • Regression for objects will interfere with each other

• Anchors: a set of reference positions on the feature map
  • Anchor box with high ground truth overlap responsible for regressing position
  • Determines reference and spatial extent for predicting object

Typically: 3 scales and 3 aspect ratios
Faster R-CNN: Anchors

- A cost-efficient way to achieve multi-scale outputs
  - Relies on image and feature maps at single scale
  - Feature computation is shared across anchor boxes at different scales

- Translation invariance
  - Use same convolutional RPN, anchor boxes for spatial localization
  - A translated object will lead to accordingly shifted proposals

Typically: 3 scales and 3 aspect ratios
Training RPN

- Place anchors uniformly across image, n boxes at every position (typically n = 9)
  - For 40 x 60 feature map, about 21k anchor boxes
- RPN output: 4n regression (x, y, w, h) and 2n classification (object, background)
- Non-maximum suppression to pick about 2k boxes
  - Remove boxes that overlap with others of higher score
- Labels for RPN classification: compute IoU of anchor boxes with ground truth
  - Assign IoU > 0.7 as object and IoU < 0.3 as background
- Bounding box regression
  - Displacement target: distance between centers of ground truth and anchor boxes
  - Size target: log ratio of anchor and ground truth dimensions
- Sample 256 anchors to form mini-batch for training

$$L(\{p_i, t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Cross-entropy  Bounding box label  Smooth L1  Regression target
Training Overall Detector

• Train CNN + RPN
  • Initialize with ImageNet pretrained weights
  • Train end-to-end for region proposal task

• Train CNN + Detector
  • Initialize with ImageNet pretrained weights
  • Use fixed proposals from above RPN
  • Train Fast R-CNN for detection task

• Fine-tune RPN
  • Use CNN from above step
  • Fine-tune RPN for proposal task

• Fine-tune detector
  • Keep CNN fixed
  • Fine-tune Fast R-CNN layers for detection task

• Subsequently, a joint training is also available with all four losses
  • RPN: classification (object or background), regression (anchor to proposal)
  • Fast R-CNN: classification (object category), regression (proposal to bounding box)
## Faster R-CNN Improvements

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image (with proposals)</td>
<td>50 seconds</td>
<td>2 seconds</td>
<td>0.2 seconds</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
<td>66.9</td>
</tr>
</tbody>
</table>
Faster R-CNN Improvements

ImageNet Detection (mAP)

mAP

ResNet ensemble (2015) 62.07
ResNet single (2015) 58.85
NanoNet ensemble (2015) 53.57
Faster R-CNN single (2015) 42.94
GoogleNet ensemble (2014) 43.93
NUS ensemble (2014) 37.21
SPP ensemble (2014) 35.11
UVA-Euvision (2013) 22.58
Overfeat (2013) 19.4
Object Detection

Huang et al. Speed/Accuracy Tradeoffs for Modern Convolutional Object Detectors. CVPR 2017

CSE 152B, SP20: Manmohan Chandraker
Instance Segmentation

Image classification

Object detection

Semantic segmentation

Instance segmentation

[Lin et al., MS COCO]
Instance Segmentation

- Two choices:
  - Predict instance first, then classify
  - Segment first, then break into instances
  - Mask R-CNN is instance-first

- Decouple mask (FCN) and class prediction (Faster R-CNN)

[He et al., Mask R-CNN]
Mask R-CNN

- Human pose estimation with keypoints as one-hot masks
- Training (2 days on 8 GPUs) and inference (200ms) speed
- RoIAlign: quantization-free for better localization

[He et al., Mask R-CNN]
RoI-Pooling in Faster R-CNN

[Credit: deepsense.io]
RoI-Align in Mask R-CNN

[Credit: Silvio Galesso]
Decoupled segmentation and classification

- Decoupling achieved with per-class binary masks (sigmoid)
- Alternative in FCN: multinomial masks (softmax)

\[ L = L_{cls} + L_{box} + L_{mask} \]

- Generate mask for every class without competition between classes
- Dedicated classification branch to predict class label to select output mask

<table>
<thead>
<tr>
<th></th>
<th>AP</th>
<th>AP_{50}</th>
<th>AP_{75}</th>
</tr>
</thead>
<tbody>
<tr>
<td>softmax</td>
<td>24.8</td>
<td>44.1</td>
<td>25.1</td>
</tr>
<tr>
<td>sigmoid</td>
<td>30.3</td>
<td>51.2</td>
<td>31.5</td>
</tr>
</tbody>
</table>

\[ L_{mask}: \text{ mean binary cross-entropy} \]

\[ K \cdot (m \times m) \text{ sigmoid outputs:} \]
\[ \rightarrow \text{ pixel-wise binary classification} \]
\[ \rightarrow \text{ one mask for each class, no competition} \]
Single-Shot Detector

- Two step process of Faster R-CNN can be expensive
  - Box classification and regression are done twice
  - RPN produces boxes, but they are used to pool features
  - Separate classifier then used for evaluation

- Key motivation for SSD: achieve good accuracy-speed trade-off
  - Use something similar to RPN for directly scoring anchors
  - Extract anchors at multiple scales to get accuracy with minimal overhead

Faster R-CNN

SSD
Single-Shot Detector

- Standard VGG (or ResNet) base network

- Add further layers with progressively decreasing size
  - Used for predicting detections at multiple scales
  - Layers with wider receptive fields expected to detect larger objects

[Liu et al., SSD]
SSD: Default boxes (anchors)

- SSD uses default boxes, which are similar to anchors in Faster R-CNN.
- Default boxes located at each cell on the feature map:
  - Multiple boxes corresponding to different aspect ratios.
- Default boxes across feature layers for better discretization of scale.

Each position has anchor boxes of different aspect ratio.

\[ S_k \times \frac{S_k}{\sqrt{a_r}} \times \frac{S_k}{\sqrt{a_r}} \times \frac{S_k}{\sqrt{a_r}} \]

\[ a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\} \]
SSD: Default boxes (anchors)

- Anchor boxes of different feature maps have different scales
  - Various feature maps responsible for detecting objects of different sizes
- Let $m$ be the number of feature maps used for prediction
- Let $s_k$ denote the scale of feature map $k$
- Define $s_{\text{min}} = 0.2$ and $s_{\text{max}} = 0.9$

$$s_k = s_{\text{min}} + \frac{s_{\text{max}} - s_{\text{min}}}{m - 1}(k - 1), \quad k \in [1, m]$$
SSD: Convolution prediction at multiple scales

- On each feature map, two types of convolutional filters applied
  - $c$ filters for class prediction, where $c$ is the number of object categories
  - 4 filters for bounding box regression, for coordinates $x, y, w, h$
- With $k$ default boxes at every cell, there are $(c+4)k$ filters for each feature map
- Output for $m \times n$ feature map is a $m \times n \times (c+4)k$ map
  - Represents category label and regression offsets for each default box
SSD: Training

- Ground truth positives: match ground truth to each default box with IoU > 0.5
- Ground truth negatives: all other default boxes
SSD: Training

- Ground truth positives: match ground truth to each default box with IoU > 0.5
- Ground truth negatives: all other default boxes
- Loss: Combination of classification confidence and localization accuracy
- Classification loss is softmax for multi-class prediction

\[
L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}
\]

\(x_{ij}^p = 1\) if default box \(i\) matches ground truth box \(j\) of class \(p\), otherwise 0

- Smooth L1 loss for localization

\[
L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)
\]

Offsets  Predicted location  Ground truth for box \(j\)
SSD: Training

• Ground truth positives: match ground truth to each default box with IoU > 0.5
• Ground truth negatives: all other default boxes
• Loss: Combination of classification confidence and localization accuracy
• Classification loss is softmax for multi-class prediction

\[ L_{conf}(x, c) = - \sum_{i \in Pos} x^p_{ij} \log(\hat{c}^p_i) - \sum_{i \in Neg} \log(\hat{c}^0_i) \quad \text{where} \quad \hat{c}^p_i = \frac{\exp(c^p_i)}{\sum_p \exp(c^p_i)} \]

\[ x^p_{ij} = 1 \text{ if default box } i \text{ matches ground truth box } j \text{ of class } p, \text{ otherwise } 0 \]

• Smooth L1 loss for localization

\[ L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x^k_{ij} \text{smooth}_{L1}(l^m_i - \hat{g}^m_j) \]

Offsets Predicted location Ground truth for box j

• Number of negatives far exceeds positives
  • Hard negative mining: choose default boxes with highest classification loss
SSD: Analysis

- Helps to use multiple layers for prediction

<table>
<thead>
<tr>
<th>Prediction source layers from:</th>
<th>mAP</th>
<th>use boundary boxes?</th>
<th># Boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv4_3 conv7 conv8_2 conv9_2 conv10_2 conv11_2</td>
<td>74.3</td>
<td>Yes</td>
<td>8732</td>
</tr>
<tr>
<td></td>
<td>74.6</td>
<td>Yes</td>
<td>8764</td>
</tr>
<tr>
<td></td>
<td>73.8</td>
<td>Yes</td>
<td>8942</td>
</tr>
<tr>
<td></td>
<td>70.7</td>
<td>Yes</td>
<td>9864</td>
</tr>
<tr>
<td></td>
<td>64.2</td>
<td>No</td>
<td>9025</td>
</tr>
<tr>
<td></td>
<td>62.4</td>
<td>No</td>
<td>8664</td>
</tr>
</tbody>
</table>

- In general, most errors in SSD are due to poorer localization
  - Explicit pooling of region proposals in Faster R-CNN prevents this
  - Possibly also cause for more confusion between similar categories in SSD

- SSD also tends to do better for larger objects compared to smaller ones
Object detection: Accuracy and speed

Huang et al. Speed/Accuracy Tradeoffs for Modern Convolutional Object Detectors. CVPR 2017
Ungraded Quiz 5

• Detection and segmentation
  – https://forms.gle/z32PoDX3x8FYNCmN9
Course Evaluation

• Course evaluation:
  • You should have received an email

• Link: http://cape.ucsd.edu

• Only 25% of class has submitted so far ....

• First time the class is offered, your feedback is valuable

• Please take 5 minutes to do this!