Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 7: Convolutional Neural Networks (part 3)
Logistics:

Assignment 2 was due yesterday

Assignment 3 will be posted tonight …
Logistics:

**Assignment 2** was due *yesterday*

**Assignment 3** will be posted *tonight* …

**Final Projects** … poll results

… 3 more students voted for individual projects than for survey
we will start process of forming groups
Computer Vision Problems (no language for now)

**Categorization**
- Multi-class: Horse, Church, Toothbrush, Person

**Detection**
- Horse (x, y, w, h)
- Person (x, y, w, h)

**Segmentation**
- Horse
- Person

**Instance Segmentation**
- Horse1
- Horse2
- Person1
- Person2

ImageNet

Multi-label: Horse, Church, Toothbrush, Person

COCO: Common Objects in Context

Horse (x, y, w, h)
Person (x, y, w, h)
Detection

Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)
Datasets: Pascal VOC

20 classes: aeroplane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, train, TV

Real images downloaded from flickr, not filtered for “quality”
Datasets: Pascal VOC

20 classes: aeroplane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, train, TV

Real images downloaded from flickr, not filtered for “quality”
Datasets: COCO

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints
Object Detection

* plot from Ross Girshick, 2015

The graph shows the mean average precision (mAP) for object detection on the PASCAL VOC dataset from 2006 to 2016. The data is divided into two phases: before and after the use of deep convolutional neural networks (D-CNNs). Before D-CNNs, the mAP generally remained below 50%. After the introduction of D-CNNs, there was a significant increase in mAP, reaching values close to 80% by 2016.
Object Detection as Regression Problem

\[ \mathbf{w}_0 \]
\[ \mathbf{w}_1 \]
\[ \mathbf{w}_2 \]
\[ \mathbf{w}_{N-1} \]

\[ \text{LSTM} \] \[ \text{LSTM} \] \[ \text{LSTM} \] \[ \text{LSTM} \]

\[ \text{CAT} (x, y, w, h) \]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
Object **Detection** as Regression Problem

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
Object **Detection** as Regression Problem

![Image of a kitten on the grass](image1.png)

**Problem:** each image needs a different number of outputs

![Image of a duck with ducklings on the grass](image2.png)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background.

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background.

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background.

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
Object **Detection** as Classification Problem

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background.

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
Object **Detection** as Classification Problem

**Problem:** Need to apply CNN to many patches in each image

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
Region Proposals (older idea in vision)

Find image **regions that are likely contain objects** (any object at all)
- typically works by looking at histogram distributions, region aspect ratio, closed contours, coherent color

Relatively **fast to run** (Selective Search gives 1000 region proposals in a few seconds on a CPU)

Goal: Get “true” object regions to be in as few top K proposals as possible

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Alexe et al, TPAMI 2012 ]
[ Uijkings et al, IJCV 2013 ]
[ Cheng et al, CVPR 2014 ]
[ Zitnick and Dollar, ECCV 2014 ]
Input Image

[ Girshick et al, CVPR 2014 ]

* image from Ross Girshick
R-CNN

Girshick et al, CVPR 2014

Input Image

Regions of Interest from a proposal method (~2k)

* image from Ross Girshick
R-CNN

[ Girshick et al, CVPR 2014 ]

Input Image

Regions of Interest from a proposal method (~2k)

Warped image regions

* image from Ross Girshick
R-CNN

Forward each region through a CNN

Warped image regions

Regions of Interest from a proposal method (~2k)

Input Image

* image from Ross Girshick
R-CNN

Classify regions with SVM

Forward each region through a CNN

Warped image regions

Regions of Interest from a proposal method (~2k)

Input Image

[ Girshick et al, CVPR 2014 ]

* image from Ross Girshick
R-CNN

**Linear Regression** for bounding box offsets

* image from Ross Girshick

**Classify** regions with SVM

Forward each region through a **CNN**

**Warped** image regions

**Regions of Interest** from a proposal method (~2k)

**Input Image**
**R-CNN: Training**

*Fine-tuning ImageNet* CNN on object proposal patches

- >50% Intersection-over-Union overlap with GT considered “object” others “background”
- batches of 128 *(32 positives, 96 negatives)*
R-CNN: Issues

**Ad-hoc** training objectives

- Fine-tune network with softmax objective (**log** loss)
- Train post-hoc linear SVM (**hinge** loss)
- Train post-hoc bounding-box regression (**least squares**)

**Training** is slow

- 84 hours and takes a lot of disk space

**Inference / Detection is slow**

- 47 sec / image with VGG16 [Simonyan et al, ICLR 2015]
R-CNN vs. SPP

R-CNN
2000 nets on image regions

SPP-net
1 net on full image

[He et al, ECCV 2014]
Fast R-CNN

[ Girshick et al, ICCV 2015 ]

* image from Ross Girshick
Fast R-CNN

Forward prop the whole image through CNN

“conv5” feature map
Fast R-CNN

Regions of Interest from the proposal method

ConvNet

Input Image

“conv5” feature map

Forward prop the whole image through CNN

[ Girshick et al, ICCV 2015 ]

* image from Ross Girshick
Fast R-CNN

[ Girshick et al, ICCV 2015 ]

* image from Ross Girshick

Regions of Interest from the proposal method

Forward prop the whole image through CNN

"RoI Pooling" layer
"conv5" feature map
RoI Align

15 x 15 pixel Region of Interest in the original image

Original Image: 128 x 128

Corresponding region in the Feature Map (2.93 x 2.93)

Feature Map: 25 x 25

CNN

bilinear interpolation

variable size RoI
Fast R-CNN

Object classification

Input Image

Forward prop the whole image through CNN

Regions of Interest from the proposal method

“conv5” feature map

“RoI Pooling” layer

Bounding box regression

Multi-task loss

Log loss + Smooth L1 loss

Linear + softmax

Linear

FCs

ConvNet

[ Girshick et al, ICCV 2015 ]

* image from Ross Girshick
Fast R-CNN: Training

Input Image

ConvNet

Regions of Interest from the proposal method

“conv5” feature map

“RoI Pooling” layer

Bounding box regression

Object classification

Linear + softmax

Linear

Log loss + Smooth L1 loss

Multi-task loss

Forward prop the whole image through CNN
R-CNN vs. SPP vs. Fast R-CNN

Training time (Hours)

- R-CNN: 84 hours
- SPP-Net: 25.5 hours
- Fast R-CNN: 8.75 hours

Test time (seconds)

- Including Region proposals:
  - R-CNN: 49 seconds
  - SPP-Net: 4.3 seconds
  - Fast R-CNN: 0.32 seconds

- Excluding Region Proposals:
  - R-CNN: 47 seconds
  - SPP-Net: 2.3 seconds
  - Fast R-CNN: 0.32 seconds

*slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Girshick et al, CVPR 2014 ]
[ Girshick et al, ICCV 2015 ]
[ He et al, ECCV 2014 ]
R-CNN vs. SPP vs. Fast R-CNN

**Observation:** Performance dominated by the region proposals at this point!
Faster R-CNN

Make CNN do proposals!

Insert Region Proposal Network (RPN) to predict proposals from features

Jointly train with 4 losses:
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Mask R-CNN

[He et al., 2017]
Mask R-CNN

[He et al., 2017]
Summary of R-CNN Family of Models

R-CNN
- Box offset regressor
- SVM object classifier
- Region CNN features
- Deep CNN
- Region proposal

Fast R-CNN
- Box offset regressor
- Softmax classifier
- Region CNN features
- RoI pooling
- Deep CNN
- Region proposal

Faster R-CNN
- Box offset regressor
- Softmax classifier
- Region CNN features
- RoI pooling
- Deep CNN
- Region proposal

Mask R-CNN
- Box offset regressor
- Softmax classifier
- Region CNN features
- RoI Align
- Deep CNN

**LSDA:** Large Scale Detection through Adaptation

\[ W_{\text{DETECT}}^{\text{cat}} = W_{\text{CLASSIFY}}^{\text{cat}} + \delta W_{\text{cat}} \]

[Hoffman et al, NIPS 2014]
YOLO: You Only Look Once

Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Input image: 3 x H x W
Divide image into grid: 7 x 7
Image a set of base boxes centered at each grid cell
Here B = 3

Output: 7 x 7 x (5 * B + C)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
YOLO: You Only Look Once

[ Redmon et al, CVPR 2016 ]

Input image
3 x H x W

Divide image into grid
7 x 7

Image a set of base boxes
centered at each grid cell
Here B = 3
Feature **Pyramid** Networks

(a) Featurized image pyramid

(b) Single feature map

(c) Pyramidal feature hierarchy

(d) Feature Pyramid Network

(e) Similar Structure with (d)

[ Lin et al, CVPR 2017 ]
**Focal Loss**

\[
p_t = \begin{cases} 
  p & \text{if } y = 1 \\
  1 - p & \text{otherwise}
\end{cases}
\]

\[
\text{CE}(p_t) = -\log(p_t)
\]

\[
\text{FL}(p_t) = -(1 - p_t) \gamma \log(p_t)
\]

[Lin et al, ICCV 2017]
Review of CNNs
Review of CNNs

\[ W^T x + b \]
Review of CNNs

Convolutional Layer

Input

\[ W^T x + b \]

where \( W \) is a 2D weight tensor of size \( R \times 3072 \), and \( x \) is the input feature map. The activation map is produced by applying the activation function to the dot product of the weight tensor and the input feature map.

Fully Connected Layer
Review of **CNNs**

**Convolutional Layer**

- 32 width, 3 depth
- Convolutional layer
- Activation map

**Fully Connected Layer**

\[ W^T x + b \]

**Pooling Layer**

Max pool with 2 x 2 filter and stride of 2

\[
\begin{array}{cccc}
1 & 1 & 2 & 4 \\
5 & 6 & 7 & 8 \\
3 & 2 & 1 & 0 \\
1 & 2 & 3 & 4 \\
\end{array}
\]
Review of CNNs

Effective Techniques for Training

- **Regularization**: L1, L2, data augmentation
- **Transfer Learning**: fine-tuning networks
Effective Techniques for **Training**

- **Regularization:** L1, L2, data augmentation
- **Transfer Learning:** fine-tuning networks

**Vision Applications of CNNs**

- **Classification:** AlexNet, VGG, GoogleLeNet, ResNet
- **Segmentation:** Fully convolutional CNNs
- **Detection:** R-CNN, Fast R-CNN, Faster R-CNN, YOLO
Any CNN Could be Fully Convolutional

Image

224 x 224

VGG

1 x 1000
Any CNN Could be Fully Convolutional

Image

VGG

225 x 225

2 x 2 x 1000
Review of CNNs

Effective Techniques for Training

- Regularization: L1, L2, data augmentation
- Transfer Learning: fine-tuning networks

Vision Applications of CNNs

- Classification: AlexNet, VGG, GoogleLeNet, ResNet
- Segmentation: Fully convolutional CNNs
- Detection: R-CNN, Fast R-CNN, Faster R-CNN, YOLO