CSCI 497P/597P: Computer Vision

Convolutional Neural Networks
Application to other problems
Announcements

• Midterm and P3 grades are out
• Final Reminder: Friday is the last day to opt into letter grades for 497P students.
And so on and so forth...

• So we’ve beat the crap out of ImageNet... what now?
  – Can we do image classification on other datasets?
  – Can we do things other than image classification?
But first...

• A brief note about datasets.
• ImageNet is a collection of images with labels
  – The 1000 classes used for evaluation are a tiny subset of the tags available.
  – The labels were produced by humans.
Lil Uzi Hurt at Home
@lostblackboy

No matter what kind of image I upload, ImageNet Roulette, which categorizes people based on an AI that knows 2500 tags, only sees me as Black, Black African, Negroid or Negro.

Some of the other possible tags, for example, are “Doctor,” “Parent” or “Handsome.”
• The viral selfie app ImageNet Roulette seemed fun – until it called me a racist slur

*The Guardian, September 2019*

600,000 Images Removed from AI Database After Art Project Exposes Racist Bias

The image tagging system that went viral on social media was part of artist Trevor Paglen and AI researcher Kate Crawford’s attempts to publicize how prejudiced technology can be.

Dataset bias

LFW
[Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Huang et al.]
77.5% male
83.5% white

IJB-A
[Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus benchmark. Klare et al.]
79.6% lighter-skinned

Adience
86.2% lighter-skinned
### Error Rate (1-PPV) By Female x Skin Type

<table>
<thead>
<tr>
<th></th>
<th>TYPE I</th>
<th>TYPE II</th>
<th>TYPE III</th>
<th>TYPE IV</th>
<th>TYPE V</th>
<th>TYPE VI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FACE++</strong></td>
<td>11.9%</td>
<td>9.7%</td>
<td>8.2%</td>
<td>13.9%</td>
<td>32.4%</td>
<td>46.5%</td>
</tr>
<tr>
<td><strong>IBM</strong></td>
<td>5.1%</td>
<td>7.4%</td>
<td>8.2%</td>
<td>8.3%</td>
<td>33.3%</td>
<td>46.8%</td>
</tr>
</tbody>
</table>

Buolamwini & Gebru FAT* 2018, Slides from Joy Buolamwini

Slide: Timnit Gebru, Emily Denton
See also

• Writeup on ImageNet Bias: https://www.excavating.ai/

• ACM Conference on Fairness, Accountability, and Transparency: https://facctconference.org/index.html
Other Computer Vision Tasks

**Semantic Segmentation**
- GRASS, CAT, TREE, SKY
  - No objects, just pixels

**Classification + Localization**
- CAT
  - Single Object

**Object Detection**
- DOG, DOG, CAT
  - Multiple Object

**Instance Segmentation**
- DOG, DOG, CAT
Other Computer Vision Tasks

**Semantic Segmentation**
- **GRASS, CAT, TREE, SKY**
- No objects, just pixels

**Classification + Localization**
- **CAT**
- Single Object

**Object Detection**
- **DOG, DOG, CAT**
- Multiple Object

**Instance Segmentation**
- **DOG, DOG, CAT**
Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels
Semantic Segmentation Idea: Sliding Window

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

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung
Semantic Segmentation Idea: Sliding Window

Full image

Extract patch

Classify center pixel with CNN

Cow
Cow
Grass

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
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

Input: $3 \times H \times W$

Convolutions: $D \times H \times W$

Scores: $C \times H \times W$

Predictions: $H \times W$
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

Input: $3 \times H \times W$

Problem: convolutions at original image resolution will be very expensive ...

Convolutions: $D \times H \times W$

Scores: $C \times H \times W$

Predictions: $H \times W$
Semantic Segmentation Idea: Fully Convolutional

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

- **Input:** \(3 \times H \times W\)
- **High-res:** \(D_1 \times H/2 \times W/2\)
- **Med-res:** \(D_2 \times H/4 \times W/4\)
- **Low-res:** \(D_3 \times H/4 \times W/4\)
- **High-res:** \(D_1 \times H/2 \times W/2\)
- **Predictions:** \(H \times W\)
Semantic Segmentation Idea: Fully Convolutional

**Downsampling:** Pooling, strided convolution

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

**Upsampling:** ???

**Input:** $3 \times H \times W$

**High-res:** $D_3 \times H/4 \times W/4$

**Med-res:** $D_2 \times H/4 \times W/4$

**Low-res:** $D_1 \times H/4 \times W/4$

**High-res:** $D_1 \times H/2 \times W/2$

**Predictions:** $H \times W$

In-Network upsampling: “Unpooling”

Nearest Neighbor

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

Input: 2 x 2

Output: 4 x 4
In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

Input: 4 x 4

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>6</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Max Unpooling
Use positions from pooling layer

Input: 2 x 2

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Output: 4 x 4

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rest of the network

Corresponding pairs of downsampling and upsampling layers
Learnable Upsampling: Transpose Convolution

Recall: Typical 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

Output: 4 x 4
Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

Dot product between filter and input

Output: 4 x 4
Learnable Upsampling: Transpose Convolution

**Recall:** Normal $3 \times 3$ convolution, stride 1 pad 1

Input: $4 \times 4$  
Output: $4 \times 4$

Dot product between filter and input
Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, **stride** 2 pad 1

![Input: 4 x 4](image1)

![Output: 2 x 2](image2)
Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, **stride 2** pad 1

- Input: 4 x 4
- Dot product between filter and input
- Output: 2 x 2

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung
Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, **stride 2** pad 1

Input: 4 x 4

Dot product between filter and input

Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input gives weight for filter

Input: 2 x 2

Output: 4 x 4
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

- Input gives weight for filter
- Sum where output overlaps
- Filter moves 2 pixels in the output for every one pixel in the input
- Stride gives ratio between movement in output and input
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

- Filter moves 2 pixels in the output for every one pixel in the input
- Stride gives ratio between movement in output and input
- Sum where output overlaps

Input gives weight for filter
Learnable Upsampling: Transpose Convolution

Other names:
- Deconvolution (bad)
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Sum where output overlaps

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input
Learnable Upsampling: 1D Example

Output contains copies of the filter weighted by the input, summing at where at overlaps in the output.

Need to crop one pixel from output to make output exactly 2x input.

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung
2D Object Detection

Semantic Segmentation

Object categories + 2D bounding boxes

GRASS, CAT, TREE, SKY
No objects, just pixels

2D Object Detection

DOG, DOG, CAT

3D Object Detection

Car
Object categories + 3D bounding boxes

This image is CC0 public domain
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? YES
Cat? NO
Background? NO
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.

Dog? YES
Cat? NO
Background? NO
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.

Dog? NO
Cat? YES
Background? NO
Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background.

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!
Region Proposals / Selective Search

- Find “bloppy” image regions that are likely to contain objects
- Relatively fast to run; e.g., Selective Search gives 2000 region proposals in a few seconds on CPU

Alexe et al., “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al., “Selective Search for Object Recognition”, IJCV 2013
Cheng et al., “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014
R-CNN

- Bbox reg
- SVMs
- ConvNet
- Linear Regression for bounding box offsets
- Classify regions with SVMs
- Forward each region through ConvNet
- Warped image regions
- Regions of Interest (RoI) from a proposal method (~2k)

Input image

Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung
Fast R-CNN

- Softmax classifier
- Linear + softmax
- Linear regressors
- Bounding-box regressors
- Fully-connected layers
- "RoI Pooling" layer
- "conv5" feature map of image
- Regions of Interest (RoIs) from a proposal method
- Forward whole image through ConvNet

Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.
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

Figure copyright 2015, Ross Girshick; reproduced with permission
Mask R-CNN

Classification Scores: C
Box coordinates (per class): 4 * C

Predict a mask for each of C classes

C x 14 x 14

He et al. "Mask R-CNN", arXiv 2017
Mask R-CNN: Very Good Results!

He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.
Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network!

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)

Output:
$7 \times 7 \times (5 \times B + C)$

Autoencoders and Generative Models
Generative Adversarial Networks
Other Problems

- Fine-grained recognition (e.g., dog/bird species)
- Instance segmentation
- Face detection and recognition
- Motion estimation
- Feature detection and description
- Depth estimation
- Novel view synthesis
- ...and many others