Topics:
• Advanced Architectures
• Bias, Fairness, Calibration

CS 4644-DL / 7643-A
ZSOLOT KIRA
• Assignment 2 – We are in grace period!

• Projects
  • Project proposal due March 13th
  • March 8th: Come with project teams/ideas and run them by TAs!

• Meta Office Hours on Fairness/Bias today 2pm EST
  • NOT recorded!
Computer Vision Tasks

- **Classification**
  (Class distribution per image)

- **Semantic Segmentation**
  (Class distribution per pixel)

- **Object Detection**
  (List of bounding boxes with class distribution per box)

- **Instance Segmentation**
  (Class distribution per pixel with unique ID)
Given an image, output another image

- Each output contains class distribution per pixel
- More generally an image-to-image problem

Semantic Segmentation
  (Class distribution per pixel)

Instance Segmentation
  (Class distribution per pixel with unique ID)
Probability distribution over classes for this one pixel.
Fully connected layers no longer explicitly retain spatial information (though the network can still learn to do so)

**Idea: Convert fully connected layer to convolution!**
Each kernel has the size of entire input! (output is 1 scalar)

- This is equivalent to $Wx+b$!
- We have one kernel per output node
Same Kernel, Larger Input

Original:

\[ W = 5 \]
\[ H = 5 \]

Conv Kernel

\[ k_1 = 3 \]
\[ k_2 = 3 \]

Output

Larger:

\[ W = 7 \]
\[ H = 7 \]

Fully Convolutional Layer Kernel
Why does this matter?

- We can stride the “fully connected” classifier across larger inputs!
- Convolutions work on arbitrary input sizes (because of striding)

Idea 2: “De”Convolution and UnPooling

We can develop learnable or non-learnable upsampling layers!

Convolutional Neural Network (CNN)

Encoder

Decoder

Useful, lower-dimensional features

Useful, lower-dimensional features
**Example**: Max pooling

- Stride window across image but perform per-patch **max operation**

\[
X(0:1, 0:1) = \begin{bmatrix} 100 & 150 \\ 100 & 200 \end{bmatrix} \quad \Rightarrow \quad \text{max}(0:1,0:1) = 200
\]

**Idea**: Remember max elements in encoder! Copy value from equivalent position, rest are zeros

Copy value to position chosen as max in encoder, fill rest of this window with zeros.
Max Unpooling Example (one window)

Encoder

$X = \begin{bmatrix}
120 & 150 & 120 \\
100 & 50 & 110 \\
25 & 25 & 10 \\
\end{bmatrix}$

$Y = \begin{bmatrix}
150 & 150 \\
100 & 110 \\
\end{bmatrix}$

Decoder

$X = \begin{bmatrix}
300 & 450 \\
100 & 250 \\
\end{bmatrix}$

$Y = \begin{bmatrix}
0 & 300 & - \\
0 & 0 & - \\
- & - & - \\
\end{bmatrix}$
Max Unpooling Example

Encoder

\[
X_{\text{enc}} = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix} \xrightarrow{\text{max pool}} \begin{bmatrix} 150 \\ 100 \\ 110 \end{bmatrix} \quad \Rightarrow \quad Y_{\text{enc}} = \begin{bmatrix} 150 \\ 150 \\ 100 \\ 110 \end{bmatrix}
\]

Contributions from multiple windows are summed

Decoder

\[
X_{\text{dec}} = \begin{bmatrix} 300 & 450 \\ 100 & 250 \end{bmatrix} \xrightarrow{\text{max unpool}} Y_{\text{dec}} = \begin{bmatrix} 0 & 300 + 450 & 0 \\ 100 & 0 \\ 0 & 250 \\ 0 \end{bmatrix}
\]
We pull max indices from corresponding layers (requires symmetry in encoder/decoder).

Convolutional Neural Network (CNN)

Encoder

Image

Convolution + Non-Linear Layer

Pooling Layer

Convolution + Non-Linear Layer

Useful, lower-dimensional features

Decoder

(De)Convolution

Non-Linear Layer

(De)Convolution + Non-Linear Layer

(De)Convolution

Non-Linear Layer

(Un)Pooling Layer

Useful, lower-dimensional features

“Image”
How can we *upsample* using convolutions and learnable kernel?

**Normal Convolution**

$$H = 5$$  
$$W = 5$$

$$k_1 = 3$$  
$$k_2 = 3$$

$$H - k_1 + 1$$  
$$W - k_2 + 1$$

**Transposed Convolution (also known as “deconvolution”, fractionally strided conv)**

Idea: Take each input pixel, multiply by learnable kernel, “stamp” it on output

![Diagram of Transposed Convolution](image)
Transposed Convolution Example

Contributions from multiple windows are summed

\[ X = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix} \quad K = \begin{bmatrix} 1 \\ 2 \\ -1 \\ -2 \end{bmatrix} \]

Incorporate

\[
\begin{bmatrix}
120 & -120 & 0 & 0 \\
240 & -240 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

Incorporate

\[
\begin{bmatrix}
120 & -120 + 150 & -150 & 0 \\
240 & -240 + 300 & -300 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]
We can either learn the kernels, or take corresponding encoder kernel and rotate 180 degrees (no decoder learning)
We can start with a pre-trained trunk/backbone (e.g. network pretrained on ImageNet)!
U-Net

You can have skip connections to bypass bottleneck!

Various ways to get **image-like outputs**, for example to predict segmentations of input images

- Fully convolutional layers essentially apply the striding idea to the output classifiers, supporting arbitrary input sizes
  - (without output size depending on what the input size is)

- We can have various upsampling layers that actually increase the size

- Encoder/decoder architectures are popular ways to leverage these to perform general image-to-image tasks
Single-Stage Object Detection
Given an image, output a list of bounding boxes with probability distribution over classes per box

Problems:

- Variable number of boxes!
- Need to determine candidate regions (position and scale) first

Object Detection
(List of bounding boxes with class distribution per box)
We can use the same idea of fully-convolutional networks

- Use ImageNet pre-trained model as backbone (e.g. taking in 224x224 image)
- Feed in larger image and get classifications for different windows in image
We can have a *multi-headed architecture*
- One part predicting distribution over class labels (classification)
- One part predicting a bounding box for each image region (regression)
  - Refinement to fit the object better (outputs 4 numbers)
- Both heads *share features*! Jointly optimized (summing gradients)
Can also do this at multiple scales to result in a large number of detections

- Various tricks used to increase the resolution (decrease subsampling ratio)
- Redundant boxes are combined through **Non-Maximal Suppression (NMS)**

Single-shot detectors use a similar idea of grids as anchors, with different scales and aspect ratios around them.

- Various tricks used to increase the resolution (decrease subsampling ratio)

*Liu, et al., “SSD: Single Shot MultiBox Detector”, 2015*
Similar network architecture but single-scale (and hence faster for same size)

1. For each bounding box, calculate intersection over union (IoU)

2. Keep only those with IoU > threshold (e.g. 0.5)

3. Calculate precision/recall curve across classification probability threshold

4. Calculate average precision (AP) over recall of [0, 0.1, 0.2, …, 1.0]

5. Average over all categories to get mean Average Precision (mAP)

\[ mAP = \frac{1}{11} \sum_{i \in [0, 0.1, \ldots, 1.0]} AP_i \]
Results

EfficientDet

PP-YOLO

Long et al., “PP-YOLO: An Effective and Efficient Implementation of Object Detector”, 2020
Two-Stage Object Detectors
Instead of making dense predictions across an image, we can decompose the problem:

- Find regions of interest (ROIs) with object-like things
- Classifier those regions (and refine their bounding boxes)

We can use **unsupervised (non-learned!) algorithms** for finding candidates

**Downsides:**
- Takes 1+ second per image
- Return thousands of (mostly background) boxes

**Resize each candidate** to full input size and classify

*Uijlings, et al., “Selective Search for Object Recognition”, 2012*
What is the problem with this?

Computation for convolutions re-done for each image patch, even if overlapping!

Idea: **Reuse** computation by finding regions in **feature maps**

- Feature extraction only done once per image now!
- Problem: Variable input size to FC layers (different feature map sizes)

Girshick, “Fast R-CNN”, 2015
Given an arbitrarily-sized feature map, we can use pooling across a grid (ROI Pooling Layer) to convert to fixed-sized representation.
Fast R-CNN

We can now train this model end-to-end (i.e. backpropagate through entire model including ROI Pooling)!
**Idea:** Why not have the neural network *also* generate the proposals?

- Region Proposal Network (RPN) uses same features!

- Outputs **objectness score** and bounding box

- Top k selected for classification

- Note some parts (gradient w.r.t. bounding box coordinates) not differentiable so some complexity in implementation

RPN also uses notion of anchors in a grid

Boxes of various sizes and scales classified with objectness score and refined bounding boxes refined

Many new advancements have been made. For example, combining detection and segmentation.

- Extract foreground (object) mask per bounding box.

*He, et al., “Mask R-CNN”, 2018*

[https://paperswithcode.com/sota/object-detection-on-coco](https://paperswithcode.com/sota/object-detection-on-coco)
• A range of problems characterized by **density and type of output**

• **Semantic/instance segmentation:** Dense, spatial output
  - Leverage encoder/decoder architectures

• **Object detection:** Variable-length list of objects
  - Two-stage versus one-stage architectures
  - (Not covered): Anchor-based versus anchor-free methods
Class Imbalance: Object Detection

# background boxes >>> # foreground boxes!
Class Imbalance: Focal Loss

Cross Entropy: easy examples incur a non-negligible loss, which in aggregate mask out the harder, rare examples

\[
CE(p, y) = \begin{cases} 
- \log(p) & \text{if } y = 1 \\
- \log(1 - p) & \text{otherwise.}
\end{cases}
\]

Focal Loss: down-weights easy examples, to give more attention to difficult examples

\[
FL(p_t) = -(1 - p_t)^\gamma \log(p_t).
\]

(Lin et al., 2017)
Class Imbalance: Focal Loss

$CE(p_t) = -\log(p_t)$

$FL(p_t) = -(1 - p_t)^\gamma \log(p_t)$

(Lin et al., 2017)
Bias & Fairness
ML and Fairness

• AI effects our lives in many ways
• Widespread algorithms with many small interactions  
  – e.g. search, recommendations, social media
• Specialized algorithms with fewer but higher-stakes interactions  
  – e.g. medicine, criminal justice, finance
• At this level of impact, algorithms can have unintended consequences
• Low classification error is not enough, need fairness
Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin

SAN FRANCISCO (Reuters) - Amazon.com Inc’s (AMZN.O) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

The team had been building computer programs since 2014 to review job applicants’ resumes with the aim of mechanizing the search for top talent, five people familiar with the effort told Reuters.

Automation has been key to Amazon’s e-commerce dominance, be it inside warehouses or driving pricing decisions. The company’s experimental hiring tool used artificial intelligence to give job candidates scores ranging from one to five stars - much like
Gender and racial bias found in Amazon’s facial recognition technology (again)

Research shows that Amazon’s tech has a harder time identifying gender in darker-skinned and female faces

By James Vincent | Jan 25, 2019, 9:45am EST

MOST READ

My Samsung Galaxy Fold screen broke after just a day

We finally know why the Instagram founders really quit

Command Line

Command Line delivers daily updates from the near-future.
ML and Fairness

• Fairness is morally and legally motivated
• Takes many forms
• Criminal justice: recidivism algorithms (COMPAS)
  – Predicting if a defendant should receive bail
  – Unbalanced false positive rates: more likely to wrongly deny a black person bail

<table>
<thead>
<tr>
<th></th>
<th>White</th>
<th>Black</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrongly Labeled High-Risk</td>
<td>23.5%</td>
<td>44.9%</td>
</tr>
<tr>
<td>Wrongly Labeled Low-Risk</td>
<td>47.7%</td>
<td>28.0%</td>
</tr>
</tbody>
</table>

Why Fairness is Hard

- Suppose we are a bank trying to fairly decide who should get a loan
  - i.e. Who is most likely to pay us back?
- Suppose we have two groups, A and B (the sensitive attribute)
  - This is where discrimination could occur
- The simplest approach is to remove the sensitive attribute from the data, so that our classifier doesn't know the sensitive attribute.

<table>
<thead>
<tr>
<th>Age</th>
<th>Gender</th>
<th>Postal Code</th>
<th>Req Amt</th>
<th>A or B?</th>
<th>Pay</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>F</td>
<td>M5E</td>
<td>$300</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>M</td>
<td>M4C</td>
<td>$1000</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>M</td>
<td>M3H</td>
<td>$250</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>F</td>
<td>M9C</td>
<td>$2000</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>71</td>
<td>F</td>
<td>M3B</td>
<td>$200</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>M</td>
<td>M5W</td>
<td>$1500</td>
<td>B</td>
<td>0</td>
</tr>
</tbody>
</table>
Why Fairness is Hard

• However, if the sensitive attribute is correlated with the other attributes, this isn't good enough
• It is easy to predict race if you have lots of other information (e.g. home address, spending patterns)
• More advanced approaches are necessary

Table 3: To Loan or Not to Loan? (masked)

<table>
<thead>
<tr>
<th>Age</th>
<th>Gender</th>
<th>Postal Code</th>
<th>Req Amt</th>
<th>A or B?</th>
<th>Pay</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>F</td>
<td>M5E</td>
<td>$300</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>M</td>
<td>M4C</td>
<td>$1000</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>M</td>
<td>M3H</td>
<td>$250</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>F</td>
<td>M9C</td>
<td>$2000</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>71</td>
<td>F</td>
<td>M3B</td>
<td>$200</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>M</td>
<td>M5W</td>
<td>$1500</td>
<td>?</td>
<td>0</td>
</tr>
</tbody>
</table>
Definitions of Fairness – Group Fairness

• So we've built our classifier . . . how do we know if we're being fair?
• One metric is demographic parity | requiring that the same percentage of A and B receive loans
  – What if 80% of A is likely to repay, but only 60% of B is?
  – Then demographic parity is too strong
• Could require equal false positive/negative rates
  – When we make an error, the direction of that error is equally likely for both groups

\[
P(\text{loan} | \text{no repay}, A) = P(\text{loan} | \text{no repay}, B) \\
P(\text{no loan} | \text{would repay}, A) = P(\text{no loan} | \text{would repay}, B)
\]

• These are definitions of group fairness
• Treat different groups equally"
Definitions of Fairness – Individual Fairness

• Also can talk about individual fairness | “Treat similar examples similarly"

• Learn fair representations
  – Useful for classification, not for (unfair) discrimination
  – Related to domain adaptation
  – Generative modelling/adversarial approaches

Figure 1: “The Variational Fair Autoencoder” (Louizos et al., 2016)
Conclusion

• This is an exciting field, quickly developing
• Central definitions still up in the air
• AI moves fast | lots of (currently unchecked) power
• Law/policy will one day catch up with technology
• Those who work with AI should be ready
  – Think about implications of what you develop!