Lecture 9:

Generating Supervision

Parallel Computing
Stanford CS348K, Spring 2021
Note

- Much of this class involved discussing the Snorkel paper(s)
Today’s theme

- Data alone is not precious. Today, in many domains large collections of *unlabeled data* are readily accessible.

- But labels (supervision) for this data is extremely precious.

- Implication: ML engineers are interested in using any means necessary to acquire sources of supervision.
Today’s problem setup

Given:

Pre-trained models (other tasks)

Huge corpus of unlabeled data
Perhaps with a sparse set of human labels

Goal: generate large amounts of supervision for use in training a model for a new task of interest
Making human labelers more efficient

- Example: “extreme clicking” is a faster way to define an object bounding box AND IT ALSO gives four points on the object’s silhouette

[Source: Papadopoulos et al. ICCV 2017]
Amplify sparse human labels: Automatically transfer labels from labeled data points to “similar” unlabeled data points
Data augmentation

Apply category-preserving transformations to images to increase size of labeled dataset.

[Source: https://medium.com/@thimblot/data-augmentation-boost-your-image-dataset-with-few-lines-of-python-155c2dc1baec]
Must be mindful of which transformations are label preserving for a task

Example: iNaturalist dataset

Is color change a good data augmentation?
Label transfer via visual similarity

If I know this image contains a cactus, then visually similar images in my unlabeled collection likely also contain a cactus as well.

What are good ways to define similar?

https://blog.waymo.com/2020/02/content-search.html
Label transfer via label propagation

- Given graph of unlabeled data points
  - e.g., nodes = images, edge weights given by visual similarity

- “Diffuse” sparse labels onto unlabeled nodes

![Label Propagation Algorithm]

[Image credit: https://www.cylynx.io/blog/efficient-large-graph-label-propagation-algorithm/]
Key idea: bringing in additional priors

Priors from previous examples:

1. similar images likely have same label (knn, label prop, clustering)

2. Certain transformations will not change the label
Using a trained model to supervise itself

- **Example:** omni-supervised learning
- **Train original model using smaller labeled training set**
- **Evaluate model on different augmentations of unlabeled image**
  
  - *Ensemble model's predictions to estimate “ground truth” label for image*
- **Re-train model on both labeled images AND estimated labels from ensemble**

<table>
<thead>
<tr>
<th>backbone</th>
<th>DD</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_S$</th>
<th>AP$_M$</th>
<th>AP$_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td></td>
<td>37.1</td>
<td>59.1</td>
<td>39.6</td>
<td>20.0</td>
<td>40.0</td>
<td>49.4</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>✓</td>
<td>37.9</td>
<td>60.1</td>
<td>40.8</td>
<td>20.3</td>
<td>41.6</td>
<td>50.8</td>
</tr>
<tr>
<td>ResNet-101</td>
<td></td>
<td>39.2</td>
<td>61.0</td>
<td>42.3</td>
<td>21.7</td>
<td>42.9</td>
<td>52.3</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>✓</td>
<td>40.1</td>
<td>62.1</td>
<td>43.5</td>
<td>21.7</td>
<td>44.3</td>
<td>53.7</td>
</tr>
<tr>
<td>ResNeXt-101-32×4</td>
<td></td>
<td>40.1</td>
<td>62.4</td>
<td>43.2</td>
<td>22.6</td>
<td>43.7</td>
<td>53.7</td>
</tr>
<tr>
<td>ResNeXt-101-32×4</td>
<td>✓</td>
<td>41.0</td>
<td>63.3</td>
<td>44.4</td>
<td>22.9</td>
<td>45.5</td>
<td>54.8</td>
</tr>
</tbody>
</table>

[Source: Radosavovic et al. CVPR 2018]
Modern trend: unsupervised pre-training

- Unsupervised pre-training at scale (using lots of data and using large models) learns good representations
- e.g. SimCLR, based on contrastive loss
- Give training image $x$, apply augmentation $t(x)$ (crop, resize, flip)

- Train DNN with contrastive loss that encourages projection of different transformations of the same image $x$ to be close ($g(f(t(x)))$ close to $g(f(t'(x)))$), transformations of different images to be far.

[Image credit: SimCLR paper, Chen et al. NeurIPS 2020]
Providing supervision by writing programs
Encode external priors in programs

- **Example: temporal consistency prior:** the state of world should not change significantly from frame to frame

  ![Frame 1](image1) ![Frame 2](image2) ![Frame 3](image3)

- **Example: domain-knowledge prior:** objects like cars cannot overlap in space

  ![Example error 1](image4) ![Example error 2](image5)

[Source: Kang et al. MLSys 2020]
DB queries as concept “detectors”
(find elements in database matching this predicate)

Video Collection

Basic Annotations

Face Detections
3:15–3:16: BERNIE...
5:18–5:20: THANK YOU...
9:15–9:17: TODAY IN...
Captions

def bernie_and_jake(faces):
    bernie = faces
    .filter(face.name == "Bernie")
    jake = faces
    .filter(face.name == "Jake")

    bernie_and_jake = bernie
    .join(jake,
        predicate = time_overlaps,
        merge_op = span)

    return bernie_and_jake

[Source: Fu et al. 2019]
Three-person panels
(three faces, bounding boxes greater than 30% of screen height, in horizontal alignment)
Today’s discussion: using weak supervision via “data programming”
Many, many ways to find, generate, and operationalize supervision

- Multiple-modalities of data, knowledge in prior models, weak sources of supervision, return to basic heuristics, etc.

- It does seem like better platform and system support would be helpful here! (more next class)