Lecture 9:
System Support for Curating Supervision
Note

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

- I am posting these slides as some were used during parts of the discussion
Today’s theme

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

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

- Implication: ML engineers are interested in using any means necessary to acquire sources of supervision.
Obtaining supervision using knowledge contained in preexisting models
Using model to supervise itself

- Example: omni-supervised learning
- Evaluate existing model on different augmentations of unlabeled image
  - Original model trained using labeled training set
  - Ensemble predictions to estimate label for image
- Re-train model on both labeled images AND estimated label images

<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]
Key idea: data augmentation

Make machine-provided labels more robust by ensembling output from multiple perturbations of the image

Another augmentation example:

[Source: https://medium.com/@thimblot/data-augmentation-boost-your-image-dataset-with-few-lines-of-python-155c2dc1baec]
Label transfer via visual similarity

- If I know this image contains a cactus, then visually similar images likely also contain cacti as well.

  - Saguaro cactus

- What are good ways to define similar?

  - “Oversize load”

https://blog.waymo.com/2020/02/content-search.html
Providing supervision by writing programs
Encode external priors in programs

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

  ![Frame 1, SSD](image1.png) ![Frame 2, SSD](image2.png) ![Frame 3, SSD](image3.png)

  (a) Frame 1, SSD  (b) Frame 2, SSD  (c) Frame 3, SSD

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

  ![Example error 1](image4.png)

  (a) Example error 1.

  ![Example error 2](image5.png)

  (b) Example error 2.

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

```
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
```
Data model: spatiotemporal labels

All video annotations represented by labels associated with spatiotemporal volume in video

Per-frame face detection: Jake Tapper

And joining me now

Per-frame face detection: Bernie Sanders
Data model: spatiotemporal labels

Labels can be nested (hierarchical)
Programming model: create new labels via operations on sets of spatiotemporal labels

Spatiotemporal join:

video time

a

[diagram with time line and labels]
Programming model: create new labels via operations on sets of spatiotemporal labels

Spatiotemporal join:

\[
\begin{align*}
\text{video time} \\
\hline
a & \\
\hline
b &
\end{align*}
\]
Programming model: create new labels via operations on sets of spatiotemporal labels

Spatiotemporal join:
Three-person panels
(three faces, bounding boxes greater than 30% of screen height, in horizontal alignment)
Additional label composition operators

**Coalesce:** merging nearby labels

**video time**

<table>
<thead>
<tr>
<th>a</th>
<th>video time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="https://example.com/coalesceDiagram.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Minus:** remove volume of one label set from another

**video time**

<table>
<thead>
<tr>
<th>a</th>
<th>video time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="https://example.com/minusDiagram.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Interview event detection example

```
faces = rekkall.ingest(database.table("faces"), 3D)
```

Input: labels for per-frame face detections
Interview detection example

1. faces = `rekall.ingest(database.table("faces"), 3D)`

2. sanders = faces
   .filter(λ face: face.name == "Bernie Sanders")

3. sanders_segs = sanders
   .coalesce(
     predicate = time_gap < 30 seconds,
     merge = time_span)

4. tapper = faces
   .filter(λ face: face.name == "Jake Tapper")

5. tapper_segs = tapper
   .coalesce(
     predicate = time_gap < 30 seconds,
     merge = time_span)

6. sanders_and_tapper_segs = sanders_segs
   .join(tapper_segs,
     predicate = time_overlaps,
     merge = time_intersection)

7. sanders_alone_segs = sanders_segs
   .minus(sanders_and_tapper_segs)

8. interviews = sanders_and_tapper_segs
   .join(sanders_alone_segs,
     predicate = before or after,
     merge = time_span)

9. interviews = interviews
   .coalesce()
Interview detection example

```
faces = rekall.ingest(database.table("faces"), 3D)

sanders = faces
  .filter(\lambda face: face.name == "Bernie Sanders")

sanders_segs = sanders
  .coalesce(
    predicate = time_gap < 30 seconds,
    merge = time_span)

Tapper detections
```

```
tapper = faces
  .filter(\lambda face: face.name == "Jake Tapper")

tapper_segs = tapper
  .coalesce(
    predicate = time_gap < 30 seconds,
    merge = time_span)

Interview detection example
```
Interview detection example

: faces = \texttt{rekall.ingest(database.table("faces"), 3D)}

: sanders = faces
  .filter(\lambda \text{face}: \text{face.name} == "Bernie Sanders")

: sanders\_segs = sanders
  .coalesce(
      predicate = \texttt{time\_gap} < 30 \text{ seconds},
      merge = \texttt{time\_span})

: tapper = faces
  .filter(\lambda \text{face}: \text{face.name} == "Jake Tapper")

: tapper\_segs = tapper
  .coalesce(
      predicate = \texttt{time\_gap} < 30 \text{ seconds},
      merge = \texttt{time\_span})
faces = rekkall.ingest(database.table("faces"), 3D)

sanders = faces
    .filter(λ face: face.name == "Bernie Sanders")

sanders_segs = sanders
    .coalesce(
        predicate = time_gap < 30 seconds,
        merge = time_span)

tapper = faces
    .filter(λ face: face.name == "Jake Tapper")

tapper_segs = tapper
    .coalesce(
        predicate = time_gap < 30 seconds,
        merge = time_span)

sanders_and_tapper_segs = sanders_segs
    .join(
        tapper_segs,
        predicate = time_overlaps,
        merge = time_intersection)

sanders_alone_segs = sanders_segs
    .minus(sanders_and_tapper_segs)

interview_segs = sanders_and_tapper_segs
    .join(
        sanders_alone_segs,
        predicate = before or after,
        merge = time_span)

interviews = interview_segs
    .coalesce()
Interview detection example

```python
faces = rekall.ingest(database.table("faces"), 3D)
sanders = faces
    .filter(λ face: face.name == "Bernie Sanders")
sanders_segs = sanders
    .coalesce(
        predicate = time_gap < 30 seconds,
        merge = time_span)
tapper = faces
    .filter(λ face: face.name == "Jake Tapper")
tapper_segs = tapper
    .coalesce(
        predicate = time_gap < 30 seconds,
        merge = time_span)
sanders_and_tapper_segs = sanders_segs
    .join(
        tapper_segs,
        predicate = time_overlaps,
        merge = time_intersection)
sanders_alone_segs = sanders_segs
    .minus(sanders_and_tapper_segs)
```

Segments with only Sanders (MINUS)
faces = rekall.ingest(database.table("faces"), 3D)

sanders = faces
    .filter(λ face: face.name == "Bernie Sanders")

sanders_segs = sanders
    .coalesce(
        predicate = time_gap < 30 seconds,
        merge = time_span)

tapper = faces
    .filter(λ face: face.name == "Jake Tapper")

tapper_segs = tapper
    .coalesce(
        predicate = time_gap < 30 seconds,
        merge = time_span)

sanders_and_tapper_segs = sanders_segs
    .join(
        tapper_segs,
        predicate = time_overlaps,
        merge = time_intersection)

sanders_alone_segs = sanders_segs
    .minus(sanders_and_tapper_segs)

interview_segs = sanders_and_tapper_segs
    .join(
        sanders_alone_segs,
        predicate = before or after,
        merge = time_span)

interviews = interview_segs.coalesce()
### Value of leveraging domain knowledge

In label-data poor scenarios, Rekall queries often on par with, and sometimes significantly more accurate than, deep learned approaches.

<table>
<thead>
<tr>
<th>Task</th>
<th>RN-50</th>
<th>RN-50, Smoothed</th>
<th>Conv3D</th>
<th>Rekall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interview</td>
<td>78.3 +/- 7.6</td>
<td>88.6 +/- 5.3</td>
<td>17.7 +/- 18.3</td>
<td>95.5</td>
</tr>
<tr>
<td>Commercial</td>
<td>90.9 +/- 1.0</td>
<td>90.0 +/- 0.9</td>
<td>88.6 +/- 0.4</td>
<td>94.9</td>
</tr>
<tr>
<td>Conversation</td>
<td>65.2 +/- 3.5</td>
<td>66.1 +/- 3.5</td>
<td><strong>79.4 +/- 2.3</strong></td>
<td>71.8</td>
</tr>
<tr>
<td>Shot Detect</td>
<td>--</td>
<td>--</td>
<td>83.2 +/- 1.0</td>
<td>84.1</td>
</tr>
<tr>
<td>Shot Scale</td>
<td>67.3 +/- 1.0</td>
<td>68.1 +/- 1.2</td>
<td>70.1 +/- 0.8</td>
<td>96.2</td>
</tr>
</tbody>
</table>

... and often can be written by domain experts (with little Rekall programming experience) in an afternoon.
Human-in-the-loop curation tasks

- Queries for curating training data for data labelers
  - Focus labeling effort on important scenes

- Queries for debugging model failures
  - Find situations where a model is likely wrong (disagreement, "flicker", etc.)

- Queries for curating video for content creation
  - Video designer looking for appropriate “B-roll”
Better interfaces for human labeling

- Example: extreme clicking defines an object bounding box AND ALSO gives four points on the object’s silhouette

5x faster for humans to label

[Source: Papadopoulos et al. ICCV 2017]
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.

- Programming is going into curating and generating training data!

- Programming is going into curating and generating training data! It does seem like better platform and system support would be helpful here!