Lecture 9:
Generating Supervision

Visual Computing Systems
Stanford CS348K, Spring 2023
Today’s agenda

- Much of this class involved discussing the Snorkel paper(s) from the reading list
PyTorch/TensorFlow/MX.Net

data-flow graphs

- Key abstraction: a program is a DAG of (large granularity) operations that consume and produce N-D tensors
Services provided by ML “frameworks”

- **Functionality:**
  - Implementations of wide range of useful operators
    - Conv, dilated conv, relu, softmax, pooling, separable conv, etc.
  - Implementations of various optimizers:
    - Basic SGD, with momentum, Adagrad, etc.
  - Ability to compose operators into large graphs to create models
  - Carry out back-propagation

- **Performance:**
  - High-performance implementation of operators (layer types)
  - Scheduling onto multiple GPUs, parallel CPUs (and sometimes multiple machines)
  - Automatic sparsification and pruning

- **Meta-optimization:**
  - Hyper-parameter search
  - More recently: neural architecture search
How to improve system support for ML?

Hardware/software for...
faster inference?
faster training?

Compilers for fusing layers, performing code optimizations?

List of papers at MLSys 2020 Conference
Today’s theme

- Today, in many domains large collections of *unlabeled data* are readily accessible

- But labels for the data (supervision) 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 (for other tasks)
- Huge corpus of unlabeled data
  Perhaps with a sparse set of human labels
- Abundant Compute

Goal: generate large amounts of supervision for use in training a model for a new task of interest
One research thrust: 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.

[Image credit: Ho et al. ICML 2019]

[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?
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

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

Priors from previous examples:

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

2. Certain transformations on data point will not change its 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\textsubscript{50}</th>
<th>AP\textsubscript{75}</th>
<th>AP\textsubscript{S}</th>
<th>AP\textsubscript{M}</th>
<th>AP\textsubscript{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

- Example: domain-knowledge prior: objects like cars cannot overlap in space
DB queries as concept “detectors”

(find elements in database matching this predicate)

[Source: Fu et al. 2019]
Example: 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”
A few more thoughts on systems for ML model development
ML model development is an iterative process
Example: does TensorFlow help with data curation?

“We cannot stress strongly enough the importance of good training data for this segmentation task: choosing a wide enough variety of poses, discarding poor training images, cleaning up inaccurate [ground truth] polygon masks, etc. With each improvement we made over a 9-month period in our training data, we observed the quality of our defocused portraits to improve commensurately.”

Synthetic Depth-of-Field with a Single-Camera Mobile Phone

NEAL WADHWA, RAHUL GARG, DAVID E. JACOBS, BRYAN E. FELDMAN, NORI KANAZAWA, ROBERT CARROLL, YAIR MOVSHOVITZ-ATTIAS, JONATHAN T. BARRON, YAEL PRITCH, and MARC LEVOY, Google Research

(a) Input image with detected face  (b) Person segmentation mask  (c) Mask + disparity from DP  (d) Our output synthetic shallow depth-of-field image
Thought experiment:
I ask you to train a car or person detector for a specific intersection.
Suggested “going further” readings

- See Overton (from Apple) and Ludwig (from Uber) papers listed under suggested readings for today’s lecture.