CS348K: Visual Computing Systems
Class/Reading Response Template

Reminder: please make sure the PDF you submit to Gradescope DOES NOT have your name on it. We will concatenate all responses and give everyone in the class a PDF of all responses.

Part 1: Top N takeaways from discussions in the last class. Note: this part of the response is unrelated to the current reading, but should pertain to the discussion of the prior reading in class (or just discussion in the class in general, if there was no reading):

- What was the most surprising/interesting thing you learned?
  - I think the most interesting is how TPU avoids memory-intensive operations by performing systolic computations. This hardware architecture reduces data load and store and allows for highly efficient pipelined processing.
- Is there anything you feel passionate about (agreed with, disagreed with?) that you want to react to?
  - I like sparse optimization. One widely used technique is to use a special sparse formats, such as Compressed Sparse Row or Compressed Sparse Column, which store only the non-zero elements of the matrix and their indices. I think some image compression techniques also have the same idea.
- Did class cause you to do any additional reading on your own? If so, what did you learn?
  - I read more about sparse optimization. Recent TVM add a new dialect SparseTIR for sparse matrix computation which supports for sparse tensors and operations, such as sparse matrix multiplication and sparse tensor slicing.
- Major takeaways in general?
  - Memory is the most important.
  - Hardware intrinsics are effective for acceleration.
  - Sparsity and low precision. Actually I think we can add update model architecture and distillation as two approaches that accelerate machine learning more from the algorithm side.

Part 2: Answers/reactions to instructor's specific prompts for this reading. (Please see course website for prompts).

First let's get our terminology straight. What is meant by "weak supervision", and how does it differ from the traditional supervised learning scenario where a training procedure is given a training set consisting of (1) a set of data examples and (2) a corresponding set of "ground truth" labels for each of these examples?

Weak supervision means that the labeling of training data is relatively bad. Compared to a traditional supervised learning scenario, weak supervision consists of a set of data examples, but not a perfect set of "ground truth" labels for each of these examples. The labels may be noisy or incomplete.

Like in all systems, I'd like everyone to pay particular attention to the design principles described in Section 1 of the Ratner et al. paper, as well as the principles defined in the Drybell paper. If you had to boil the entire philosophy of Snorkel down to one thing, what would you say it is? Hint: look at principle 1 in the Ratner et al. paper. "If accurate labels are expensive, but data is cheap, find a way to use all the sources of ___________ you can get your hands on."

I would say it's weak supervision. By combining sources of weak supervision, Snorkel can generate training data quickly when accurate labels are difficult to get.

Previously humans injected their knowedge by labeling data examples. Under the Snorkel paradigm, now humans inject their knowledge now?

Humans provide heuristics, instructions and rules for indirect supervision. These weak supervision signals are used to generate probabilistic labels for the training data, which are then used to train a model.

The main abstraction in Snorkel is the labeling function. Please describe what the output interface of a labeling function is. Then, and most importantly, what is the value of this abstraction. Hint: you probably want to refer to the key principle of Snorkel in your answer.

The output interface of a labeling function is a probability distribution for each datapoint. The value is the probability of some labeling function to contribute to the final result.

What is the role of the final model training part of Snorkel? (That is, training an off-the-shelf DNN architecture on the probabalistic labels produced by Snorkel.) Why not just use the probablistic labels as the model itself?

The role of the final model training is to learn a mapping from the input data to the output labels using the probabilistic labels. The reason why not just use the labels is that they are just an estimation and are too coarse and noisy for training. This DNN allows to get some probabilistic based insights above the labels.

One interesting aspect of Snorkel is the notion of learning from "non-servable assets". (This is more clearly articulated in the Snorkel DryBell paper, so take a look there). This is definitely not an issue that would be high on the list of academic concerns, but it is quite important in industry. What is a non-servable asset and how does the Snorkel approach make it possible to "learn from" non-servable assets even if they are not available at later runtimes.

Non-servable are data that can’t be used, maybe because of expensive or sensitive. Snorkel DryBell creates classifiers of comparable quality to ones trained with tens of thousands of hand-labeled examples, which converts non-servable organizational resources to servable. This is done in the data modeling, so it still work even if non-servable assets are not available at later runtimes.

From the ML perspective, the key technical insight of Snorkel (and of most follow on Snorkel papers, see here, here, and here for some examples) is the mathematical modeling of correlations between labeling functions in order to more accurately estimate probabilistic labels. We will not talk in detail about these generative algorithms in class, but many of you will enjoy learning about them. In general, I'd like you to reflect on Snorkel and (if time) some of the recommended papers below. (see the Rekall blog post, or the Model Assertions paper for different takes.) I'm curious about your comments.

Snorkel is a practical approach to apply weak supervision. It leverages sources and informations through a probabilistic model to get labels. It can be widely applied to all kinds and models and it’s useful in both academia and industry. The paper also provides a DSL, which makes it even easier to use and deploy.

Part 3: [Optional] Questions I’d like to have specifically addressed via in class. (We also encourage you to just post these questions on Ed immediately so anyone can answer!)