Fast and Three-rious: Speeding Up Weak Supervision with Triplet Methods

Daniel Y. Fu [*1] **Mayee F. Chen** [*1] **Frederic Sala** [1] **Sarah M. Hooper** [2] **Kayvon Fatahalian** [1] **Christopher Ré** [1]

Abstract

Weak supervision is a popular method for building machine learning models without relying on ground truth annotations. Instead, it generates probabilistic training labels by estimating the accuracies of multiple noisy labeling sources (e.g., heuristics, crowd workers). Existing approaches use latent variable estimation to model the noisy sources, but these methods can be computationally expensive, scaling superlinearly in the data. In this work, we show that, for a class of latent variable models highly applicable to weak supervision, we can find a *closed-form solution* to model parameters, obviating the need for iterative solutions like stochastic gradient descent (SGD). We use this insight to build FLYINGSQUID, a weak supervision framework that runs *orders of magnitude* faster than previous weak supervision approaches and requires fewer assumptions. In particular, we prove bounds on generalization error without assuming that the latent variable model can exactly parameterize the underlying data distribution. Empirically, we validate FLYINGSQUID on benchmark weak supervision datasets and find that it achieves the same or higher quality compared to previous approaches without the need to tune an SGD procedure, recovers model parameters 170 times faster on average, and enables new video analysis and online learning applications.

1. Introduction

Modern machine learning systems require large amounts of labeled training data to be successful. Weak supervision is a class of popular methods for building models without resorting to manually labeling training data (Dehghani et al., 2017b;a; Jia et al., 2017; Mahajan et al., 2018; Niu et al., 2012); it drives applications used by billions of people every day, ranging from Gmail (Sheng et al., 2020) to AI products at Apple (Ré et al., 2020) and search products (Bach et al., 2019). These approaches use noisy sources, such as heuristics, crowd workers, external knowledge bases, and user-defined functions (Gupta & Manning, 2014; Ratner et al., 2019; Karger et al., 2011; Dawid & Skene, 1979; Mintz et al., 2009; Zhang et al., 2017; Hearst, 1992) to generate probabilistic training labels without hand-labeling.

The major technical challenge in weak supervision is to efficiently estimate the accuracies of—and potentially the correlations among—the noisy sources without any labeled data (Guan et al., 2018; Takamatsu et al., 2012; Xiao et al., 2015; Ratner et al., 2018). Standard approaches to this problem, from classical crowdsourcing to more recent methods, use latent variable probabilistic graphical models (PGMs) to model the primary sources of signal—the agreements and disagreements between sources, along with known or estimated source independencies (Dawid & Skene, 1979; Karger et al., 2011; Ratner et al., 2016).

However, latent variable estimation is challenging, and the techniques are often sample- and computationally-complex. For example, Bach et al. (2019) required multiple iterations of a Gibbs-based algorithm, and Ratner et al. (2019) required estimating the full inverse covariance matrix among the sources, while Sala et al. (2019) and Zhan et al. (2019) required the use of multiple iterations of stochastic gradient descent (SGD) to learn accuracy parameters. These limitations make it difficult to use weak supervision in applications that require modeling complex temporal or spatial dependencies, such as video and image analysis, or in streaming applications that have strict latency requirements. In contrast, our solution is motivated by a key observation: that by breaking the problem into minimal subproblems—solving parameters for triplets of sources at a time, similar to Joglekar et al. (2013) and Chaganty & Liang (2014)—we can reduce parameter estimation into solving systems of equations that have simple, *closed-form solutions*.

Concretely, we show that, for a class of binary Ising models, we can reduce the problem of accuracy and correlation estimation to solving a set of systems of equations whose size is linear in the number of sources. These systems admit a closed-form solution, so we can estimate the model

[*]Equal contribution [1]Department of Computer Science, Stanford University [2]Department of Electrical Engineering, Stanford University. Correspondence to: Daniel Y. Fu <danfu@cs.stanford.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

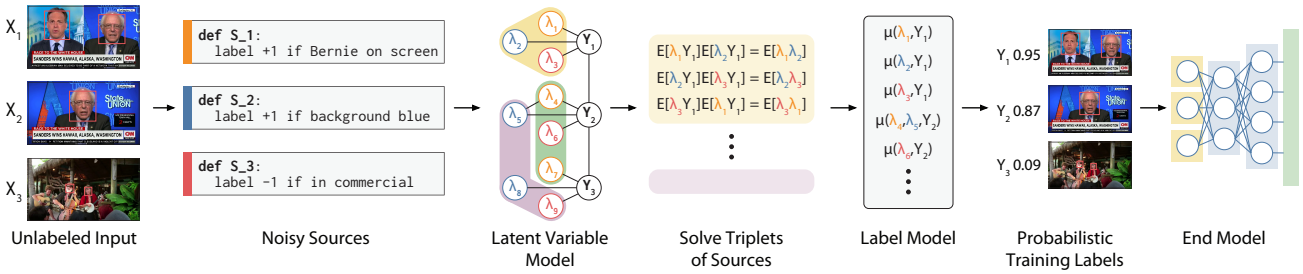

Figure 1. The FLYINGSQUID pipeline. Users provide weak supervision sources, which generate noisy labels for a set of unlabeled data. FLYINGSQUID uses a latent variable model and constructs triplets of sources to turn model parameter estimation into a set of minimal subproblems with closed-form solutions. The label model then generates probabilistic training labels to train a downstream end model.

parameters in time linear in the data with provable bounds, even though inference is NP-hard in general Ising models (Chandrasekaran et al., 2008; Koller & Friedman, 2009). Critically, the class of Ising models we use captures many weak supervision settings and is larger than that used in previous efforts. We use these insights to build FLYINGSQUID, a new weak supervision framework that learns label source accuracies with a closed-form solution.

We analyze the downstream performance of end models trained with labels generated by FLYINGSQUID, and prove the following results:

- We prove that the generalization error of a model trained with labels generated by FLYINGSQUID scales at the same asymptotic rate as supervised learning.
- We analyze model misspecification using KL divergence, a more fine-grained result than Ratner et al. (2019).
- We show that our parameter estimation approach can be sample optimal up to constant factors via an information-theoretic lower bound on minimax risk.
- We prove a first-of-its-kind result for downstream generalization of a window-based online weak supervision algorithm, accounting for distributional drift.

Next, we empirically validate FLYINGSQUID on three benchmark weak supervision datasets that have been used to evaluate previous state-of-the-art weak supervision frameworks (Ratner et al., 2018), as well as on four video analysis tasks, where labeling training data is particularly expensive and modeling temporal dependencies introduces significant slowdowns in learning graphical model parameters. We find that FLYINGSQUID achieves the same or higher quality as previous approaches while learning parameters orders of magnitude faster. Since FLYINGSQUID runs so fast, we can learn graphical model parameters *in the training loop* of a discriminative end model. This allows us to extend FLYINGSQUID to the online learning setting with a window-based algorithm, where we update model parameters at the same time as we generate labels for an end model. In summary, we observe the following empirical results:

- We replicate evaluations of previous approaches and match or exceed their accuracy (up to 4.9 F1 points).
- On tasks with relatively simple graphical model structures, FLYINGSQUID learns model parameters 170 times faster on average; on video analysis tasks, where there are complex temporal dependencies, FLYINGSQUID learns up to 4,000 times faster.
- We demonstrate that our window-based online weak supervision extension can both update model parameters and train an end model completely online, outperforming a majority vote baseline by up to 15.7 F1 points.

We release FLYINGSQUID as a novel layer integrated into PyTorch.[1] This layer allows weak supervision to be integrated off-the-shelf into any deep learning model, learning the accuracies of noisy labeling sources in the same training loop as the end model. Our layer can be used in any standard training set up, enabling new modes of training from multiple label sources.

2. Weakly Supervised Machine Learning

In this section, we give an overview of weak supervision and our problem setup. In Section 2.1, we give an overview of the inputs to weak supervision from the user's perspective. In Section 2.2, we describe the formal problem setup. Finally, in Section 2.3, we show how the problem reduces to estimating the parameters of a latent variable PGM.

2.1. Background: Weak Supervision

We first give some background on weak supervision at a high level. In weak supervision, practitioners programmatically generate training labels through the process shown in Figure 1. Users build multiple weak supervision sources that assign noisy labels to data. For example, an analyst trying to detect interviews of Bernie Sanders in a corpus of cable TV news may use off-the-shelf face detection and identity

[1] https://github.com/HazyResearch/flyingsquid

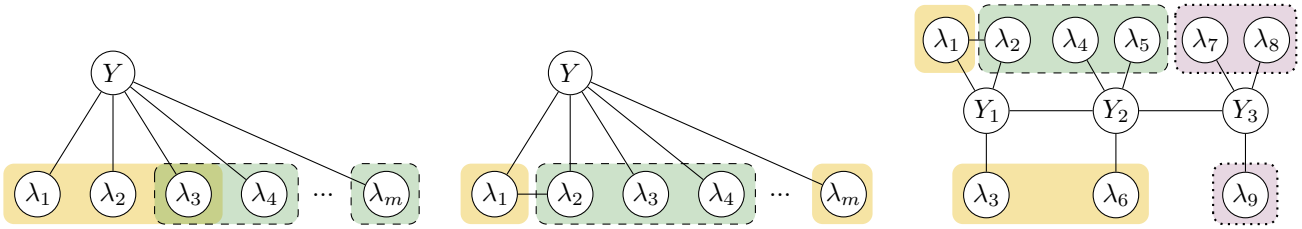

Figure 2. Example of dependency structure graphs and triplets (rectangles). Left: Conditionally independent sources; Middle: With dependencies. Right: Multiple temporally-correlated labels $\{Y_1, Y_2, Y_3\}$ with per-label sources.

classification networks to detect frames where Sanders is on screen, or she may write a Python function to search closed captions for instances of the text "Bernie Sanders." Critically, these weak supervision sources can vote or abstain on individual data points; this lets users express high-precision signals without requiring them to have high recall as well. For example, while the text "Bernie Sanders" in the transcript is a strong signal for an interview, the absence of the text is not a strong signal for the absence of an interview (once he is introduced, his name is not mentioned for most of the interview).

These sources are noisy and may conflict with each other, so a latent variable model, which we refer to as a *label model*, is used to express the accuracies of and correlations between them. Once its parameters are learned, the model is used to aggregate source votes and generate probabilistic training labels, which are in turn used to train a downstream discriminative model (*end model* from here on).

2.2. Problem Setup

Now, we formally define our learning problem. Let $\boldsymbol{X} = [X_1, X_2, \ldots, X_D] \in \mathcal{X}$ be a vector of D related elements (e.g., contiguous frames in a video, or neighboring pixels in an image). Let $\boldsymbol{Y} = [Y_1, Y_2, \ldots, Y_D] \in \mathcal{Y}$ be the vector of *unobserved* true labels for each element (e.g., the per-frame label for event detection in video, or a per-pixel label for a segmentation mask in an image). We refer to each Y_i as a *task*. We have $(\boldsymbol{X}, \boldsymbol{Y}) \sim \mathcal{D}$ for some distribution $\mathcal{D}$. We simplify to binary $Y_i \in \{\pm 1\}$ for ease of exposition (we discuss the multi-class case in Appendix C.2). Let m be the number of sources $S_1, \ldots, S_m$, each assigning a label $\lambda_j \in \{\pm 1\}$ to some single element X_i to vote on its respective Y_i, or abstaining ($\lambda_j = 0$).

The goal is to apply the m weak supervision sources to an unlabeled dataset $\{\boldsymbol{X}^i\}_{i=1}^n$ with n data points to create an $n \times m$ label matrix L, combine the source votes into element-wise probabilistic training labels, $\{\widetilde{\boldsymbol{Y}}^i\}_{i=1}^n$, and use them to train a discriminative classifier $f_w : \mathcal{X} \to \mathcal{Y}$, *all without observing any ground truth labels*.

2.3. Label Model

Now, we describe how we use a probabilistic graphical model to generate training data based on labeling function outputs. First, we describe how we use a graph to specify the conditional dependencies between label sources and tasks. Next, we describe how to represent the task labels $\boldsymbol{Y}$ and source votes $\boldsymbol{\lambda}$ using a binary Ising model from user-provided conditional dependencies between sources and tasks. Then, we discuss how to perform inference using the junction tree formula and introduce the label model parameters our method focuses on estimating.

Conditional Dependencies Let a graph G_{dep} specify conditional dependencies between sources and tasks, using standard technical notions from the PGM literature (Koller & Friedman, 2009; Lauritzen, 1996; Wainwright & Jordan, 2008). In particular, the lack of an edge in G_{dep} between a pair of variables indicates independence conditioned on a *separator set* of variables (Lauritzen, 1996). We assume that G_{dep} is user-provided; it can also be estimated directly from source votes (Ratner et al., 2019). Figure 2 shows three graphs, capturing different relationships between tasks and supervision sources. Figure 2 (left) is a single-task scenario where noisy source errors are conditionally independent; this case covers many benchmark weak supervision datasets. Here, $D = 1$, and there are no dependencies between different elements in the dataset (e.g., randomly sampled comments from YouTube for sentiment analysis). Figure 2 (middle) has dependencies between the errors of two sources (λ_1 and λ_2). Finally, Figure 2 (right) depicts a more complex scenario, where three tasks have dependencies between them. This structure is common in applications with temporal dependencies like video; for example, Y_1, Y_2, Y_3 might be contiguous frames (Sala et al., 2019).

Binary Ising Model We augment the dependency graph G_{dep} to set up a binary Ising model on $G = (V, E)$. Let the vertices $V = \{\boldsymbol{Y}, \boldsymbol{v}\}$ contain a set of hidden variables $\boldsymbol{Y}$ (one for every task Y_i) and observed variables $\boldsymbol{v}$, generated by augmenting $\boldsymbol{\lambda}$. We generate $\boldsymbol{v}$ by letting there be a pair of binary observed variables (v_{2i-1}, v_{2i}) for each label source λ_i, such that (v_{2i-1}, v_{2i}) is equal to $(1, -1)$ when $\lambda_i = 1$,

$(-1, 1)$ when $\lambda_i = -1$, and $(1, 1)$ or $(-1, -1)$ with equal probability when $\lambda_i = 0$. This mapping also produces an augmented label matrix $\mathcal{L}$ from the empirical label matrix L, which contains n samples of source labels.

Next, let the edges E be constructed as follows. Let $Y^{dep}(i)$ denote the task that λ_i labels for all $i \in [1, m]$. Then for all i, there is an edge between each of (v_{2i-1}, v_{2i}) and $Y^{dep}(i)$ representing the accuracy of λ_i as well as an edge between v_{2i-1} and v_{2i} representing the abstain rate of λ_i. If there is an edge between λ_i and λ_j in G_{dep}, then there are four edges between (v_{2i-1}, v_{2i}) and (v_{2j-1}, v_{2j}). We also define $Y(j)$ as the hidden variable that v_j acts on for all $j \in [1, 2m]$; in particular, $Y(2i-1) = Y^{dep}(i)$.

Inference The Ising model defines a joint distribution $P(\boldsymbol{Y}, \boldsymbol{\lambda})$ (detailed in Appendix C.1), which we wish to use for inference. We can take advantage of the graphical model properties of G_{dep} for efficient inference. In particular, suppose that G_{dep} is triangulated; if not, edges can always be added to G_{dep} until it is. Then, G_{dep} admits a junction tree representation with maximal cliques $C \in \tilde{\mathcal{C}}_{dep}$ and separator sets $S \in \mathcal{S}_{dep}$. Inference is performed via a standard approach, using the junction tree formula

$$P(\boldsymbol{Y}, \boldsymbol{\lambda}) = \prod_{C \in \tilde{\mathcal{C}}_{dep}} \mu_C \Big/ \prod_{S \in \mathcal{S}_{dep}} \mu_S^{d(S)-1}, \qquad (1)$$

where μ_C is the marginal probability of a clique C, μ_S is the marginal probability of a separator set S, and $d(S)$ is the number of maximal cliques S is adjacent to (Lauritzen, 1996; Wainwright & Jordan, 2008). We refer to these marginals as the *label model parameters* $\boldsymbol{\mu}$.

We assume the distribution prior $P(\bar{\boldsymbol{Y}})$ is user-provided, but it can also be estimated directly by using source votes as in Ratner et al. (2019) or by optimizing a composite likelihood function as in Chaganty & Liang (2014). Some other marginals are directly observable from the votes generated by the sources $S_1, \ldots, S_m$. However, marginals containing elements from both $\boldsymbol{Y}$ and $\boldsymbol{\lambda}$ are not directly observable, since we do not observe $\boldsymbol{Y}$. The challenge is thus recovering this set of marginals $P(Y_i, \ldots, Y_j, \lambda_k, \ldots, \lambda_l)$.

3. Learning The Label Model

Now that we have defined our label model parameters $\boldsymbol{\mu}$, we need to recover the parameters directly from the label matrix L without observing the true labels $\boldsymbol{Y}$. First, we discuss how we recover the mean parameters of our Ising model using Algorithm 1 (Section 3.1). Then, we map the mean parameters to label model parameters (Section 3.2) by computing expectations over cliques of G and applying a linear transform to obtain $\boldsymbol{\mu}$. Finally, we discuss an extension to the online setting (Section 3.3).

Algorithm 1 Triplet Method (before averaging)

Input: Set of variables Ω_G, augmented label matrix $\mathcal{L}$
Initialize $A = \emptyset$
while $\exists\, v_i \in \Omega_G - A$ **do**
  Pick $v_j, v_k : v_i \perp\!\!\!\perp v_j | Y(i), v_i \perp\!\!\!\perp v_k | Y(i), v_j \perp\!\!\!\perp v_k | Y(i)$.
  Estimate $\hat{\mathbb{E}}[v_i v_j] = \frac{1}{n}\sum_t \mathcal{L}_{it}\mathcal{L}_{jt}$, $\hat{\mathbb{E}}[v_i v_k] = \frac{1}{n}\sum_t \mathcal{L}_{it}\mathcal{L}_{kt}$, and $\hat{\mathbb{E}}[v_j v_k] = \frac{1}{n}\sum_t \mathcal{L}_{jt}\mathcal{L}_{kt}$.
  $\hat{a}_i \leftarrow \sqrt{|\hat{\mathbb{E}}[v_i v_j] \cdot \hat{\mathbb{E}}[v_i v_k] \,/\, \hat{\mathbb{E}}[v_j v_k]|}$
  $\hat{a}_j \leftarrow \sqrt{|\hat{\mathbb{E}}[v_i v_j] \cdot \hat{\mathbb{E}}[v_j v_k] \,/\, \hat{\mathbb{E}}[v_i v_k]|}$
  $\hat{a}_k \leftarrow \sqrt{|\hat{\mathbb{E}}[v_i v_k] \cdot \hat{\mathbb{E}}[v_j v_k] \,/\, \hat{\mathbb{E}}[v_i v_j]|}$
  $A \leftarrow A \cup \{v_i, v_j, v_k\}$
end while
return ResolveSigns$(\hat{a}_i)\ \forall\, v_i \in V$

Inputs and Outputs As input, we take in a label matrix L that has, on average, better-than-random samples; dependency graph G_{dep}; and the prior $P(\bar{\boldsymbol{Y}})$. As output, we want to compute $\boldsymbol{\mu}$, which would enable us to produce probabilistic training data via (1).

3.1. Learning the Mean Parameters

We explain how to compute the mean parameters $\mathbb{E}[Y_i], \mathbb{E}[Y_i Y_j], \mathbb{E}[v_i Y(i)]$, and $\mathbb{E}[v_i v_j]$ of the Ising model. Note that all of these parameters can be directly estimated besides $\mathbb{E}[v_i Y(i)]$. Although we cannot observe $Y(i)$, we can compute $\mathbb{E}[v_i Y(i)]$ using a closed-form method by relying on notions of independence and rates of agreement between groups of three conditionally independent observed variables for the hidden variable $Y(i)$. Set $a_i := \mathbb{E}[v_i Y(i)]$, which can be thought of as the *accuracy* of the observed variable scaled to $[-1, +1]$. The following proposition produces sufficient signal to learn from:

Proposition 1. *If* $v_i \perp\!\!\!\perp v_j | Y(i)$, *then* $v_i Y(i) \perp\!\!\!\perp v_j Y(i)$.

Our proof is provided in Appendix C.1.1. This follows from a symmetry argument applied to the conditional independence of two variables v_i and v_j given $Y(i)$. Then

$$a_i a_j = \mathbb{E}[v_i Y(i)]\,\mathbb{E}[v_j Y(i)] = \mathbb{E}\left[v_i v_j Y(i)^2\right] = \mathbb{E}[v_i v_j],$$

where we used $Y(i)^2 = 1$. While we cannot observe a_i, the product of $a_i a_j$ is just $\mathbb{E}[v_i v_j]$, the observable rate at which a pair of variables act together. We can then utilize a third variable v_k such that $a_i a_k$ and $a_j a_k$ are also observable, and solve a system of three equations for the accuracies up to sign, e.g., $|a_i|, |a_j|, |a_k|$. We explain how to recover signs with the ResolveSigns function in Appendix C.1.5.

Formally, define $\Omega_G = \{v_i \in V : \exists\, v_j, v_k \text{ s.t. } v_i \perp\!\!\!\perp v_j | Y(i), v_j \perp\!\!\!\perp v_k | Y(i), v_i \perp\!\!\!\perp v_k | Y(i)\}$ to be the set of variables that can be grouped into triplets in this way. For each

Algorithm 2 Label Model Parameter Recovery

Input: G_{dep}, distribution prior $P(\bar{\boldsymbol{Y}})$, label matrix L.
Augment G_{dep} and L to generate $G = (V, E)$ with clique-set $\mathcal{C}$ and augmented label matrix $\mathcal{L}$.
Obtain set of variables Ω_G with solvable accuracies.
Compute mean parameters and estimate all $\hat{a}_i = \hat{\mathbb{E}}\left[v_i Y(i)\right]$ using Algorithm 1.
for clique $C \in \mathcal{C}$ of observed variables **do**
  Compute $\hat{a}_C = \hat{\mathbb{E}}\left[\prod_{k \in C} v_k Y(C)\right]$ by factorizing into observable averages and mean parameters.
  Map $\hat{a}_C$ in G to $\hat{a}_{C_{dep}}$ in G_{dep}.
  Linearly transform $\hat{a}_{C_{dep}}$ to $\hat{\mu}_{C_{dep}}$.
end for
return Label model parameters $\hat{\boldsymbol{\mu}}$

variable $v_i \in \Omega_G$, we can compute the accuracy a_i by solving the system $a_i a_j = \mathbb{E}\left[v_i v_j\right], a_i a_k = \mathbb{E}\left[v_i v_k\right], a_j a_k = \mathbb{E}\left[v_j v_k\right]$. In many practical settings, $\Omega_G = V$, so the *triplet method* of recovery applies to each v_i, motivating Algorithm 1 (some examples of valid triplet groupings shown in Figure 2). Note that variables can appear in multiple triplets, and variables do not necessarily need to vote on the same task $Y(i)$ as long as they are conditionally independent given $Y(i)$. Different triplets give different accuracy values, so we compute accuracy values from all possible triplets and use the mean or median over all triplets. In cases where Ω_G is not equal to V, we supplement the triplet method with other independence properties to recover accuracies on more complex graphs, detailed in Appendix C.2.

3.2. Mapping to the Label Model Parameters

Now we map the mean parameters of our Ising model to label model parameters. We use the mean parameters to compute relevant expectations over the set $\mathcal{C}$ of all cliques in G, map them to expectations over cliques $\mathcal{C}_{dep}$ in G_{dep}, and linearly transform them into label model parameters. Define $Y(C)$ as the hidden variable that the entire clique $C \in \mathcal{C}$ of observed variables acts on. Each expectation over a clique of observed variables C and $Y(C)$, denoted $a_C := \mathbb{E}\left[\prod_{k \in C} v_k Y(C)\right]$, can be factorized in terms of the mean parameters and directly observable expectations (Appendix C.1.2). For instance, $v_i v_j \perp\!\!\!\perp Y(i,j)$ for $(v_i, v_j) \in E$, such that $\mathbb{E}\left[v_i v_j Y(i,j)\right] = \mathbb{E}\left[v_i v_j\right] \cdot \mathbb{E}\left[Y(i,j)\right]$.

Next, we convert the expectations over cliques in G back into expectations over cliques in G_{dep}. Denote $a_{C_{dep}} := \mathbb{E}\left[\prod_{k \in C_{dep}} \lambda_k Y^{dep}(C_{dep})\right]$ for each source clique $C_{dep} \in \mathcal{C}_{dep}$; then, there exists a $C \in \mathcal{C}$ over $\{v_{2k-1}\}_{k \in C_{dep}}$ such that $a_C = \mathbb{E}\left[\prod_{k \in C_{dep}} v_{2k-1} Y^{dep}(C_{dep})\right] = a_{C_{dep}}$ (Appendix C.1.3).

Finally, the label model parameters, which are marginal distributions over maximal cliques and separator sets, can be expressed as linear combinations of $a_{C_{dep}}$ and probabilities that can be estimated directly from the data. Below is an example of how to recover $\mu_i(a,b) = P(Y^{dep}(i) = a, \lambda_i = b)$ from $\mathbb{E}\left[\lambda_i Y^{dep}(i)\right]$:

$$\begin{bmatrix} 1&1&1&1&1&1 \\ 1&0&1&0&1&0 \\ 1&1&0&0&0&0 \\ 1&0&0&0&0&1 \\ 0&0&1&1&0&0 \\ 0&0&1&0&0&0 \end{bmatrix} \begin{bmatrix} \mu_i(1,1) \\ \mu_i(-1,1) \\ \mu_i(1,0) \\ \mu_i(-1,0) \\ \mu_i(1,-1) \\ \mu_i(-1,-1) \end{bmatrix} = \begin{bmatrix} 1 \\ P(Y^{dep}(i)=1) \\ P(\lambda_i=1) \\ P(\lambda_i Y^{dep}(i)=1) \\ P(\lambda_i=0) \\ P(\lambda_i=0, Y^{dep}(i)=1) \end{bmatrix}. \quad (2)$$

$P(\lambda_i Y^{dep}(i) = 1)$ can be written as $\frac{1}{2}(\mathbb{E}\left[\lambda_i Y^{dep}(i)\right] - P(\lambda_i = 0) + 1)$ and $P(\lambda_i = 0, Y^{dep}(i) = 1)$ is factorizable due to the construction of G, so all values on the right of (2) are known, and we can solve for μ_i. Extending this example to larger cliques requires computing more a_C values and more directly estimatable probabilities; we detail the general case in Appendix C.1.4.

3.3. Weak Supervision in Online Learning

Now we discuss an extension to online learning. Online learning introduces two challenges: first, samples are introduced one by one, so we can only see each $\boldsymbol{X}^t$ once before discarding it; second, online learning is subject to *distributional drift*, meaning that the distribution P_t each $(\boldsymbol{X}^t, \boldsymbol{Y}^t)$ is sampled from changes over time. Our closed-form approach is fast, both in terms of sample complexity and wall-clock time, and only requires computing the averages of observable summary statistics, so we can learn $\boldsymbol{\mu}_t$ online with a rolling window, interleaving label model estimation and end model training. We describe this online variant of our method and how window size can be adjusted to optimize for sampling noise and distributional drift in Appendix C.3.

4. Theoretical Analysis

In this section, we analyze our method for label model parameter recovery and provide bounds on its performance. First, we derive a $\mathcal{O}(1/\sqrt{n})$ bound for the sampling error $\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2$ in Algorithm 2. Next, we show that this sampling error has a tight minimax lower bound for certain graphical models, proving that our method is information-theoretically optimal. Then, we present a generalization error bound for the end model that scales in the sampling error and a *model misspecification* term, which exists when the underlying data distribution $\mathcal{D}$ cannot be represented with our graphical model. Lastly, we interpret these results, which are more fine-grained than prior weak supervision analyses, in terms of end model performance and label model tradeoffs. All proofs are provided in Appendix D.

In Appendix C.3.1, we give two further results for the online variant of the algorithm: selecting an optimal window size to

minimize sampling error, and providing a guarantee on end model performance even in the presence of distributional drift, sample noise, and model misspecification.

Sampling Error We first control the error in estimating the label model parameters $\hat{\boldsymbol{\mu}}$. The noise comes from sampling in the empirical estimates of moments and probabilities used by Algorithm 2.

Theorem 1. *Let $\hat{\boldsymbol{\mu}}$ be an estimate of $\boldsymbol{\mu}$ produced by Algorithm 2 using n unlabeled data points. Then, assuming that cliques in G_{dep} are limited to 3 vertices,*

$$\mathbb{E}\left[\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2\right] \leq \frac{1}{a_{\min}^5}\left(3.19C_1\sqrt{\frac{m}{n}} + \frac{6.35C_2}{\sqrt{r}}\frac{m}{\sqrt{n}}\right),$$

where $a_{\min} > 0$ is a lower bound on the absolute value of the accuracies of the sources, and r is the minimum frequency at which sources abstain, if they do so.

If no sources abstain, $\sqrt{r}$ is not present in the bound. For higher-order cliques, the error scales in m with the size of the largest clique. In the case of full conditional independence, only the first term in the bound is present, so the error scales as $\mathcal{O}\left(\sqrt{\frac{m}{n}}\right)$.

Optimality We show that our method is sample optimal in both n and m up to constant factors for certain graphical models. We bound the minimax risk for the parameter estimates to be $\Omega\left(\frac{m}{\sqrt{n}}\right)$ via Assouad's Lemma (Yu, 1997). This bound holds for any binary Ising model used in our framework, but in particular it is tight when our observed variables are all conditionally independent and do not abstain.

Theorem 2. *Let $\mathcal{P} = \left\{P(Y, \boldsymbol{v}) = \frac{1}{Z}\exp\left(\theta_Y Y + \sum_{i=1}^{m}\theta_i v_i Y\right), \theta \in \mathbb{R}^{m+1}\right\}$ be a family of distributions. Using L_2 norm estimation of the minimax risk, the sampling error is lower bounded as*

$$\inf_{\hat{\boldsymbol{\mu}}}\sup_{P\in\mathcal{P}}\mathbb{E}_P\left[\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}(P)\|_2\right] \geq \frac{e_{min}}{8}\sqrt{\frac{m}{n}}.$$

Here $\boldsymbol{\mu}(P)$ is the set of label model parameters corresponding to a distribution P, and e_{min} is the minimum eigenvalue of $\boldsymbol{Cov}\left[Y, \boldsymbol{v}\right]$ for distributions in $\mathcal{P}$.

Generalization Bound We provide a bound quantifying the performance gap between the end model parametrization that uses outputs of our label model and the optimal end model parametrization over the true distribution of labels.

Let $P_{\hat{\boldsymbol{\mu}}}(\cdot|\boldsymbol{\lambda})$ be the probabilistic output of our learned label model parametrized by $\hat{\boldsymbol{\mu}}$ given some source labels $\boldsymbol{\lambda}$. Define a loss function $L(w, \boldsymbol{X}, \boldsymbol{Y}) \in [0, 1]$, where w parametrizes the end model $f_w \in \mathcal{F} : \mathcal{X} \to \mathcal{Y}$, and choose $\hat{w}$ such that

$$\hat{w} = \text{argmin}_w \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\widetilde{\boldsymbol{Y}}\sim P_{\hat{\boldsymbol{\mu}}}(\cdot|\boldsymbol{\lambda}(\boldsymbol{X}^i))}\left[L(w, \boldsymbol{X}^i, \widetilde{\boldsymbol{Y}})\right].$$

While previous approaches (Ratner et al., 2019) make the strong assumption that there exists some $\boldsymbol{\mu}$ such that sampling $(\boldsymbol{X}, \widetilde{\boldsymbol{Y}})$ from $P_{\boldsymbol{\mu}}$ is equivalent to sampling from $\mathcal{D}$, our generalization error bound accounts for potential model misspecification:

Theorem 3. *Let $w^* = \text{argmin}_w \mathbb{E}_{(\boldsymbol{X},\boldsymbol{Y})\sim\mathcal{D}}\left[L(w, \boldsymbol{X}, \boldsymbol{Y})\right]$. There exists a $\hat{w}$ computed from the outputs of our label model such that the generalization error for $\boldsymbol{Y}$ satisfies*

$$\mathbb{E}_{\mathcal{D}}\left[L(\hat{w}, \boldsymbol{X}, \boldsymbol{Y}) - L(w^*, \boldsymbol{X}, \boldsymbol{Y})\right] \leq \gamma(n) + \frac{8|\mathcal{Y}|}{e_{min}}||\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}||_2 + \delta(\mathcal{D}, P_{\boldsymbol{\mu}}),$$

where $\delta(\mathcal{D}, P_{\boldsymbol{\mu}}) = 2\sqrt{2\,KL(\mathcal{D}(\boldsymbol{Y}|\boldsymbol{X}) \,||\, P_{\boldsymbol{\mu}}(\boldsymbol{Y}|\boldsymbol{X}))}$, e_{min} is the minimum eigenvalue of $\boldsymbol{Cov}\left[\boldsymbol{Y}, \boldsymbol{v}\right]$ over the construction of the binary Ising model, and $\gamma(n)$ is a decreasing function that bounds the error from performing empirical risk minimization to learn $\hat{w}$.

Interpreting the Bounds The generalization error in Theorem 3 has two components, involving the noise awareness of the model and the model misspecification. Using the sampling error result, the first two terms $\gamma(n)$ and $||\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}||_2$ scale in $\mathcal{O}(1/\sqrt{n})$, which can be tight by Theorem 2 and is the same asymptotic rate as supervised approaches.

The third term $\delta(\mathcal{D}, P_{\boldsymbol{\mu}})$ is a divergence between our model and $\mathcal{D}$. Richer models can represent more distributions and have a smaller KL term, but may suffer a higher sample complexity. This tradeoff suggests the importance of selecting an appropriately constrained graphical model in practice.

5. Evaluation

The primary goal of our evaluation is to validate that FLYINGSQUID can achieve the same or higher quality as state-of-the-art weak supervision frameworks (Section 5.1) while learning label model parameters orders of magnitude faster (Section 5.2). We also evaluate the online extension and discuss how online learning can be preferable to offline learning in the presence of distributional shift over time (Section 5.3).

Datasets We evaluate FLYINGSQUID on three benchmark datasets and four video analysis tasks. Each dataset consists of a large (187–64,130) unlabeled training set, a smaller (50–9,479) hand-labeled *development set*, and a held-out test set. We use the unlabeled training set to train the label model and end model, and use the labeled development set

					End Model Performance (F1), Label Model Training Time (s)					Lift, Speedup			
	Task	D	m	Prop	TS	MV	DP	SDP	FLYINGSQUID (l.m. in paren.)	TS	MV	DP	SDP
Benchmarks	**Spouse**	1	9	0.07	20.4 ± 0.2 –	19.3 ± 0.01 –	44.7 ± 1.7 7.5 ± 0.9	– –	**49.6 ± 2.4** (47.0) **0.017 ± 0.003**	+29.3 –	+30.3 –	+4.9 440×	– –
	Spam	1	10	0.49	91.5 –	88.3 –	91.8 0.76 ± 0.1	– –	**92.3** (89.1) **0.014 ± 0.002**	+0.8 –	+4.0 –	+0.5 54×	– –
	Weather	1	103	0.53	74.6 –	87.3 –	87.3 0.78 ± 0.1	– –	**88.9** (77.6) **0.150 ± 0.03**	+14.3 –	+1.6 –	+1.6 5.2×	– –
Video Analysis	**Interview**	6	24	0.03	80.0 ± 3.4 –	58.0 ± 5.3 –	8.7 ± 0.2 31.5 ± 1.0	92.0 ± 2.2 256.6 ± 5.4	91.9 ± 1.6 (**93.0**) **0.423 ± 0.04**	+11.9 –	+33.9 –	+83.2 74.5×	-0.1 607×
	Commercial	6	24	0.32	90.9 ± 1.0 –	91.8 ± 0.2 –	90.5 ± 0.4 23.3 ± 1.0	89.8 ± 0.5 265.6 ± 6.2	**92.3 ± 0.4** (88.4) **0.067 ± 0.01**	+1.4 –	+0.5 –	+1.8 350×	+2.5 4,000×
	Tennis Rally	14	84	0.34	57.6 ± 3.4 –	80.2 ± 1.0 –	82.5 ± 0.3 41.1 ± 1.9	80.6 ± 0.7 398.4 ± 7.5	**82.8 ± 0.4** (82.0) **0.199 ± 0.04**	+25.2 –	+2.6 –	+0.3 210×	+2.2 2,000×
	Basketball	8	32	0.12	26.8 ± 1.3 –	8.1 ± 5.4 –	7.7 ± 3.3 28.7 ± 2.0	**38.2 ± 4.1** 248.6 ± 7.7	37.9 ± 1.9 (27.9) **0.092 ± 0.03**	+11.1 –	+29.8 –	+30.2 310×	-0.3 2,700×

Table 1. FLYINGSQUID performance in terms of F1 score (first row of each task), and label model training time in seconds (second row). We report mean ± standard deviation across five random weight initializations of the end model (except for **Spam** and **Weather**, which use logistic regression). Improvement in terms of mean end model lift, speedup in terms of mean runtime. We compare FLYINGSQUID's end model and label model (label model in parentheses) against traditionally supervised (TS) end models trained on the labeled dev set, majority vote (MV), data programming (DP) and sequential data programming (SDP). D: number of related elements modeled (contiguous sequences of frames for video tasks). m: number of supervision sources. Prop: proportion of positive examples.

for a) training a traditional supervision baseline, and b) for hyperparameter tuning of the label and end models. More details about each task and the experiments in Appendix E.

Benchmark Tasks. We draw three benchmark weak supervision datasets from a previous evaluation of a state-of-the-art weak supervision framework (Ratner et al., 2018). **Spouse** seeks to identify mentions of spouse relationships in a set of news articles (Corney et al., 2016), **Spam** classifies whether YouTube comments are spam (Alberto et al., 2015), and **Weather** is a weather sentiment task from Crowdflower (Cro, 2018).

Video Analysis Tasks. We use video analysis as another driving task: video data is large and expensive to label, and modeling temporal dependencies is important for quality but introduces significant slowdowns in label model parameter recovery (Sala et al., 2019). **Interview** and **Commercial** identify interviews with Bernie Sanders and commercials in a corpus of TV news, respectively (Fu et al., 2019; Int, 2018). **Tennis Rally** identifies tennis rallies during a match from broadcast footage. **Basketball** identifies basketball videos in a subset of ActivityNet (Caba Heilbron et al., 2015).

5.1. Quality

We now validate that end models trained with labels generated by FLYINGSQUID achieve the same or higher quality as previous state-of-the-art weak supervision frameworks. We also discuss the relative performance of FLYINGSQUID's label model compared to the end model, and ablations of our method.

End Model Quality To evaluate end model quality, we use FLYINGSQUID to generate labels for the unlabeled training set and compare the end models trained with these labels against four baselines:

1. *Traditional Supervision* **[TS]**: We train the end model using the small hand-labeled development set.
2. *Majority Vote* **[MV]**: We generate training labels over the unlabeled training set using majority vote.
3. *Data Programming* **[DP]**: We use data programming, a state-of-the-art weak supervision framework that models each data point separately (Ratner et al., 2019).
4. *Sequential Data Programming* **[SDP]**: For the video tasks, we also use a state-of-the-art sequential weak supervision framework, which models sequences of frames (Sala et al., 2019).

Table 1 shows our results. We achieve the same or higher end model quality compared to previous weak supervision frameworks. Since FLYINGSQUID does not rely on SGD to learn label model parameters, there are fewer hyperparameters to tune, which can help us achieve higher quality than previous reported results.

Label Model vs. End Model Performance Table 1 also shows the performance of FLYINGSQUID's label model. In four of the seven tasks, the end model outperforms the label model, since it can learn new features directly from the input data that are not available to the noisy sources. For example, the sources in the **Commercial** task rely on simple visual heuristics like the presence of black frames (in our dataset, commercials tend to be book-ended on either side by black frames); the end model, which is a deep network, is able to pick up on subtler features over the pixel space. In three tasks, however, the label model nearly matches

	Streaming End Model (F1)			Improvement	
Task	**TS**	**MV**	**FLYINGSQUID**	**TS**	**MV**
Interview	41.9 ± 4.0	37.8 ± 9.5	**53.5 ± 0.5**	+11.6	+15.7
Commercial	56.5 ± 1.7	78.9 ± 14.5	**93.0 ± 0.5**	+36.5	+14.1
Tennis Rally	41.5 ± 1.7	81.6 ± 0.6	**82.7 ± 0.4**	+25.2	+1.1
Basketball	20.7 ± 4.2	22.0 ± 11.3	**26.7 ± 0.3**	+6.0	+4.7

Table 2. We compare performance of an end model trained with an online pass over the training set, and then the test set with labels from FLYINGSQUID, against a model trained with majority vote (MV) labels over the training and test set, and a traditionally supervised (TS) model trained with ground truth labels over the test set. We report mean ± standard deviation from five random weight initializations.

or slightly outperforms the end model. In these cases, the sources have access to features that are difficult for an end model to learn with the amount of unlabeled data available. For example, the sources in the **Interview** task rely on an identity classifier that has learned to identify Bernie Sanders from thousands of examples.

Ablations We describe the results of two ablation studies (detailed results in Appendix E.4). In the first study, we replace abstentions with random votes instead of augmenting G_{dep}. This results in a degradation of 25.6 points, demonstrating the importance of allowing supervision sources to abstain. In the second study, we examine the effect of using individual triplet assignments instead of taking the median or mean over all possible assignments. On average, taking random assignments results in a degradation of 23.8 points compared to taking an aggregate. Furthermore, there is a large degree of variance in label model performance when using individual triplet assignments. While the best assignments can match FLYINGSQUID, bad assignments result in significantly worse performance.

5.2. Speedup

We now evaluate the speedup that FLYINGSQUID provides over previous weak supervision frameworks. Table 1 shows measurements of how long it takes to train each label model. Since FLYINGSQUID learns source accuracies and correlations with a closed-form solution, it runs orders of magnitude faster than previous weak supervision frameworks, which rely on multiple iterations of stochastic gradient descent and thus scale superlinearly in the data. Speedup varies due to the optimal number of iterations for DP and SDP, which are SGD-based (number of iterations is tuned for accuracy), but FLYINGSQUID runs up to 440 times faster than data programming on benchmark tasks, and up to 4,000 times faster than sequential data programming on the video tasks (where modeling sequential dependencies results in much slower performance).

5.3. Online Weak Supervision

We now evaluate the ability of our online extension to simultaneously train a label model and end model online for our video analysis tasks. We also use synthetic experiments to demonstrate when training a model online can be preferable to training a model offline.

Core Validation We first validate our online extension by using the FLYINGSQUID PyTorch layer to simultaneously train a label model and end model online for our video analysis tasks. We train first on the training set and then on the test set (using probabilistic labels for both). We compare against online traditional supervision (TS) and majority vote (MV) baselines. Since the training set is unlabeled, the TS model is trained only on the ground-truth test set labels, while the MV baseline uses majority vote to label the training and test sets. To mimic the online setting, each datapoint is only seen once during training.

Table 2 shows our results. Our method outperforms MV by up to 15.7 F1 points, and TS by up to 36.5 F1 points. Even though TS is trained on ground-truth test set labels, it underperforms both other methods because it only does a single pass over the (relatively small) test set. MV and FLYINGSQUID, on the other hand, see many more examples in the weakly-labeled training set before having to classify the test set.

The online version of FLYINGSQUID often underperforms its offline equivalent (Table 1), since the online model can only perform a single iteration of SGD with each datapoint. However, in 2 cases, the online model overperforms the offline model, for two reasons: a) the training set is large enough to make up the difference in having multiple epochs with SGD, and b) online training over the test set enables continued specialization to the test set.

Distributional Drift Over Time We also study the effect of distributional drift over time using synthetic experiments. Distributional drift can mean that label model parameters learned on previous data points may not describe future data points. Figure 3 shows the results of online vs. offline training in two settings with different amounts of drift. On the left is a setting with limited drift; in this setting, the offline model learns better parameters than the online model, since it has access to more data, all of which is representative of the test set. On the right is a setting with large amounts of periodic drift; in this setting, the offline model cannot learn parameters that work for all data points. But the online model, which only learns parameters for a recent window of data points, is able to specialize to the periodic shifts.

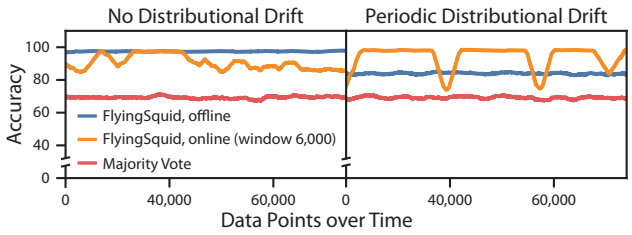

Figure 3. When there is large distributional drift, online learning can outperform offline learning by adapting over time (synthetic).

6. Related Work

Latent Variable Estimation Latent variable estimation is a classic problem in machine learning, used for hidden Markov Models, Markov random fields, topic modeling, and more (Wainwright & Jordan, 2008; Koller & Friedman, 2009). General algorithms do not admit closed-form solutions; classical techniques like expectation maximization and Gibbs sampling can require many iterations to converge, while techniques like tensor decomposition run the expensive power method (Anandkumar et al., 2014). We show that the weak supervision setting allows us to break down the parameter estimation problem into subproblems with closed-form solutions.

Our solution is similar to previous methods that have exploited triplets of conditionally-independent variables to solve latent variable estimation (Joglekar et al., 2013; Chaganty & Liang, 2014). Joglekar et al. (2013) focuses on the explicit context of crowdsourcing and is equivalent to a simplified version of Algorithm 1 when all the label sources are conditionally independent from each other and do not abstain. In contrast, our work handles a wider variety of use cases critical for weak supervision (such as sources that can abstain) and develops theoretical characterizations for downstream model behavior. Chaganty & Liang (2014) shows how to estimate the canonical parameters of a wide class of graphical models by applying tensor decomposition to recover conditional parameters. By comparison, our work is more specialized, which lets us replace tensor decomposition with a non-iterative closed-form solution, even for non-binary variables. A more detailed comparison against both of these methods is available in Appendix A.

Weak Supervision Our work is related to several such techniques, such as distant supervision (Mintz et al., 2009; Craven et al., 1999; Hoffmann et al., 2011; Takamatsu et al., 2012), co-training methods (Blum & Mitchell, 1998), pattern-based supervision (Gupta & Manning, 2014) and feature annotation (Mann & McCallum, 2010; Zaidan & Eisner, 2008; Liang et al., 2009). Recently, weak supervision frameworks rely on latent graphical models and other methods to systematically integrate multiple noisy sources (Ratner et al., 2016; 2018; Bach et al., 2017; 2019; Guan et al., 2018; Khetan et al., 2018; Sheng et al., 2020; Ré et al., 2020). Two recent approaches have proposed new methods for modeling sequential dependencies in particular, which is important in applications like video (Zhan et al., 2019; Sala et al., 2019; Safranchik et al., 2020). These approaches largely rely on iterative methods like stochastic gradient descent, and do not run closed-form solutions to latent variable estimation.

Crowdsourcing Our work is related to crowdsourcing (crowd workers can be thought of as noisy label sources). A common approach in crowdsourcing is filtering crowd workers using a small set of gold tasks, or filtering based on number of previous tasks completed or with monetary incentives (Rashtchian et al., 2010; Shaw et al., 2011; Sorokin & Forsyth, 2008; Downs et al., 2010; Mitra et al., 2015; Kittur et al., 2008). In contrast, in our setting, we do not have access to ground truth data to estimate source accuracies, and we cannot filter out noisy sources *a priori*. Other techniques can estimate worker accuracies without ground truth annotations, but assume that workers are independent (Karger et al., 2011). We can also directly model crowd workers using our label model, as in the **Weather** task.

Online Learning Training models online traditionally requires hand labels (Cesa-Bianchi & Lugosi, 2006; Shalev-Shwartz et al., 2012), but recent approaches like Mullapudi et al. (2019) train models online using a student-teacher framework (training a student network online based on the outputs of a more powerful teacher network). In contrast, our method does not rely on a powerful network that has been pre-trained to carry out the end task. In both traditional and newer distillation settings, a critical challenge is updating model parameters to account for domain shift (Shalev-Shwartz et al., 2012). For our online setting, we deal with distributional drift via a standard rolling window.

7. Conclusion

We have proposed a method for latent variable estimation by decomposing it into minimal subproblems with closed-form solutions. We have used this method to build FLYINGSQUID, a new weak supervision framework that achieves the same or higher quality as previous approaches while running orders of magnitude faster, and presented an extension to online learning embodied in a novel FLYINGSQUID layer. We have proven generalization and sampling error bounds and shown that our method can be sample optimal. In future work, we plan to extend our insights to more problems where closed-form latent variable estimation can result in faster algorithms or new applications—problems such as structure learning and data augmentation.

Acknowledgments

We thank Avanika Narayan for helping with the Tennis dataset, and Avner May for helpful discussions. We gratefully acknowledge the support of DARPA under Nos. FA86501827865 (SDH) and FA86501827882 (ASED); NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, American Family Insurance, Google Cloud, Swiss Re, Brown Institute for Media Innovation, the HAI-AWS Cloud Credits for Research program, Department of Defense (DoD) through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program, Fannie and John Hertz Foundation, National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518, Texas Instruments Stanford Graduate Fellowship in Science and Engineering, and members of the Stanford DAWN project: Teradata, Facebook, Google, Ant Financial, NEC, VMWare, and Infosys. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.

References

Weather sentiment: Dataset in crowdflower. https://data.world/crowdflower/weather-sentiment, 2018.

Internet archive: Tv news archive. https://archive.org/details/tv, 2018.

Alberto, T. C., Lochter, J. V., and Almeida, T. A. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 138–143. IEEE, 2015.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

Bach, S. H., He, B., Ratner, A., and Ré, C. Learning the structure of generative models without labeled data. In *ICML*, 2017.

Bach, S. H., Rodriguez, D., Liu, Y., Luo, C., Shao, H., Xia, C., Sen, S., Ratner, A., Hancock, B., Alborzi, H., et al. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, pp. 362–375, 2019.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100. ACM, 1998.

Caba Heilbron, F., Escorcia, V., Ghanem, B., and Carlos Niebles, J. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge university press, 2006.

Chaganty, A. T. and Liang, P. Estimating latent-variable graphical models using moments and likelihoods. In *International Conference on Machine Learning*, pp. 1872–1880, 2014.

Chandrasekaran, V., Srebro, N., and Harsha, P. Complexity of inference in graphical models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 70–78. AUAI Press, 2008.

Corney, D., Albakour, D., Martinez-Alvarez, M., and Moussa, S. What do a million news articles look like? In *NewsIR@ ECIR*, pp. 42–47, 2016.

Craven, M., Kumlien, J., et al. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, pp. 77–86, 1999.

Dawid, A. P. and Skene, A. M. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics*, pp. 20–28, 1979.

Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. Learning to learn from weak supervision by full supervision. In *NIPS workshop on Meta-Learning (MetaLearn 2017)*, 2017a.

Dehghani, M., Zamani, H., Severyn, A., Kamps, J., and Croft, W. B. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 65–74. ACM, 2017b.

Downs, J. S., Holbrook, M. B., Sheng, S., and Cranor, L. F. Are your participants gaming the system? screening mechanical turk workers. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 2399–2402, 2010.

Fu, D. Y., Crichton, W., Hong, J., Yao, X., Zhang, H., Truong, A., Narayan, A., Agrawala, M., Ré, C., and Fatahalian, K. Rekall: Specifying video events using compositions of spatiotemporal labels. *arXiv preprint arXiv:1910.02993*, 2019.

Guan, M. Y., Gulshan, V., Dai, A. M., and Hinton, G. E. Who said what: Modeling individual labelers improves classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Gupta, S. and Manning, C. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pp. 98–108, 2014.

Hearst, M. A. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pp. 539–545. Association for Computational Linguistics, 1992.

Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., and Weld, D. S. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 541–550. Association for Computational Linguistics, 2011.

Jia, Z., Huang, X., Eric, I., Chang, C., and Xu, Y. Constrained deep weak supervision for histopathology image segmentation. *IEEE transactions on medical imaging*, 36 (11):2376–2388, 2017.

Joglekar, M., Garcia-Molina, H., and Parameswaran, A. Evaluating the crowd with confidence. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 686–694, 2013.

Karger, D. R., Oh, S., and Shah, D. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pp. 1953–1961, 2011.

Khetan, A., Lipton, Z. C., and Anandkumar, A. Learning from noisy singly-labeled data. In *International Conference on Learning Representations*, 2018.

Kittur, A., Chi, E. H., and Suh, B. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 453–456, 2008.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Lauritzen, S. *Graphical Models*. Clarendon Press, 1996.

Liang, P., Jordan, M. I., and Klein, D. Learning from measurements in exponential families. In *Proceedings of the 26th annual international conference on machine learning*, pp. 641–648. ACM, 2009.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 181–196, 2018.

Mann, G. S. and McCallum, A. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of machine learning research*, 11(Feb): 955–984, 2010.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pp. 1003–1011. Association for Computational Linguistics, 2009.

Mitra, T., Hutto, C. J., and Gilbert, E. Comparing person- and process-centric strategies for obtaining quality data on amazon mechanical turk. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 1345–1354, 2015.

Mullapudi, R. T., Chen, S., Zhang, K., Ramanan, D., and Fatahalian, K. Online model distillation for efficient video inference. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3573–3582, 2019.

Niu, F., Zhang, C., Ré, C., and Shavlik, J. W. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.

Rashtchian, C., Young, P., Hodosh, M., and Hockenmaier, J. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pp. 139–147. Association for Computational Linguistics, 2010.

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the 44th International Conference on Very Large Data Bases (VLDB)*, Rio de Janeiro, Brazil, 2018.

Ratner, A. J., Sa, C. M. D., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016.

Ratner, A. J., Hancock, B., Dunnmon, J., Sala, F., Pandey, S., and Ré, C. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, 2019.

Ré, C., Niu, F., Gudipati, P., and Srisuwananukorn, C. Overton: A data system for monitoring and improving machine-learned products. In *Proceedings of the 10th Annual Conference on Innovative Data Systems Research*, 2020.

Safranchik, E., Luo, S., and Bach, S. H. Weakly supervised sequence tagging from noisy rules. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Sala, F., Varma, P., Fries, J., Fu, D. Y., Sagawa, S., Khattar, S., Ramamoorthy, A., Xiao, K., Fatahalian, K., Priest, J., and Ré, C. Multi-resolution weak supervision for sequential data. In *Advances in Neural Information Processing Systems 32*, pp. 192–203, 2019.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Shaw, A. D., Horton, J. J., and Chen, D. L. Designing incentives for inexpert human raters. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pp. 275–284, 2011.

Sheng, Y., Vo, N. H., Wendt, J. B., Tata, S., and Najork, M. Migrating a privacy-safe information extraction system to a software 2.0 design. In *Proceedings of the 10th Annual Conference on Innovative Data Systems Research*, 2020.

Sorokin, A. and Forsyth, D. Utility data annotation with amazon mechanical turk. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8. IEEE, 2008.

Takamatsu, S., Sato, I., and Nakagawa, H. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pp. 721–729. Association for Computational Linguistics, 2012.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2691–2699, 2015.

Yu, B. Assouad, fano, and le cam. In *Festschrift for Lucien Le Cam*, pp. 423–435. Springer, 1997.

Zaidan, O. F. and Eisner, J. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 31–40. Association for Computational Linguistics, 2008.

Zhan, E., Zheng, S., Yue, Y., Sha, L., and Lucey, P. Generating multi-agent trajectories using programmatic weak supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Zhang, C., Ré, C., Cafarella, M., De Sa, C., Ratner, A., Shin, J., Wang, F., and Wu, S. DeepDive: Declarative knowledge base construction. *Commun. ACM*, 60(5): 93–102, 2017.