Carnegie Mellon University

Auton Lab

End-to-End Weak Supervision

Neural Information Processing Systems (NeurIPS), 2021



Salva Rühling Cachay[1,2], Benedikt Boecking[1], Artur Dubrawski[1]

[1]Auton Lab, Carnegie Mellon University
[2]Technical University of Darmstadt

salvaruehling@gmail.com

Successful Machine Learning methods require large amounts of labeled data

https://medium.com/syncedreview/sensetime-trains-imagenet-alexnet-in-record-1-5-minutes-e944ab049b2c

Hand labeling, however, is expensive both in terms of time and cost

https://medium.com/syncedreview/sensetime-trains-imagenet-alexnet-in-record-1-5-minutes-e944ab049b2c

Alternative:
(Multi-source) Weak supervision [1]

[1] A. Ratner, C. De Sa, S. Wu, D. Selsam, C. Ré, “Data programming: Creating large training sets, quickly”, NeurIPS 2016.

Weak Supervision

Multiple noisy heuristics that cheaply apply to unlabeled data!

= Labeling functions (LFs)

Domain Heuristics

```
def f1(text):
  return (SPAM
          if `money`
          in text
          else ABSTAIN)
```

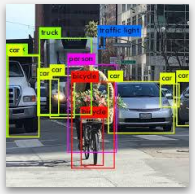
e.g. Hearst (1992), Dunnmon et al. (2020)

Distant Supervision

Freebase™

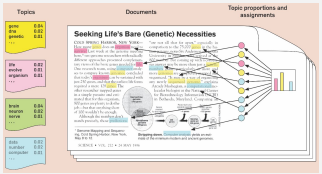
e.g. Mintz et al. (2019), Bach et al. (2019)

Pretrained Models



e.g. Chen et al. (2019)

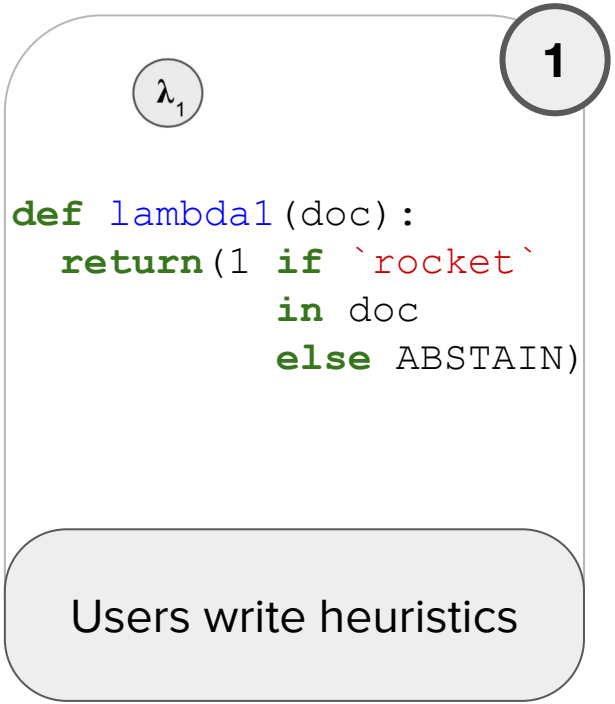
Unsupervised Models



e.g. Hingmire et al. (2014), Bach et al. (2019)

Image credit: Blei, David M. "Probabilistic topic models." Communications of the ACM 55, no. 4 (2012): 77-84.

The usual approach



The usual approach

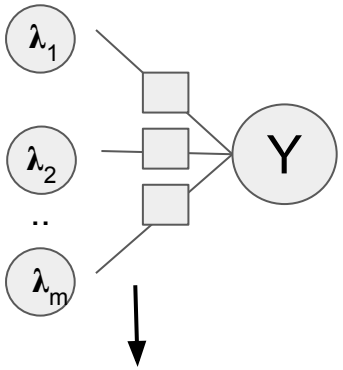
1

λ_1

```
def lambda1(doc):
  return(1 if `rocket`
            in doc
            else ABSTAIN)
```

Users write heuristics

2

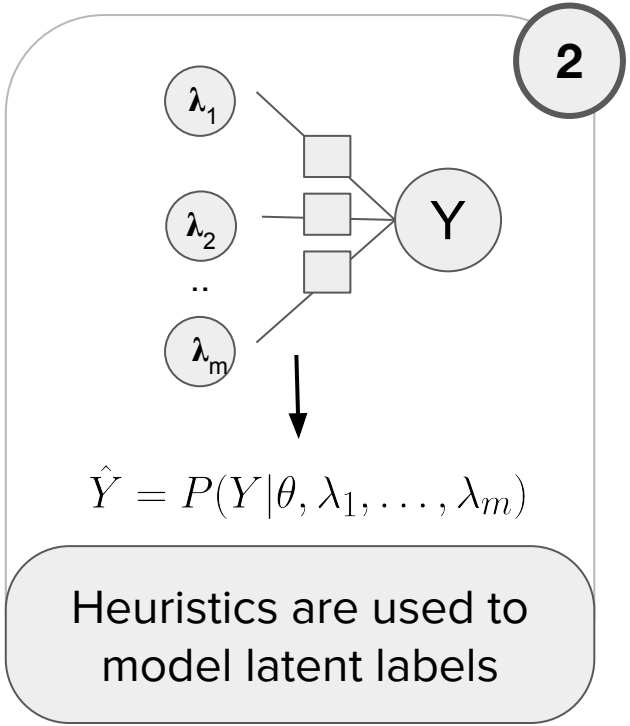


$$\hat{Y} = P(Y|\theta, \lambda_1, \ldots, \lambda_m)$$

Heuristics are used to model latent labels

But...

- Statistical dependencies are hard to model (efficiently)
  - Thus, they are often simply ignored!
- No data features/representations are considered!



→ This & more (often) violates assumptions needed for theoretical results

The usual approach

Two separate modeling steps!

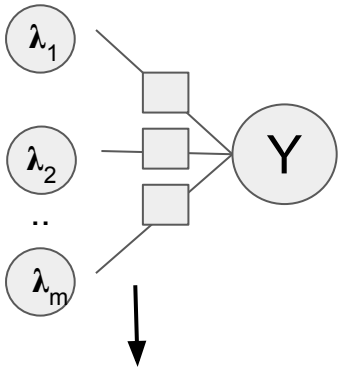
1

λ_1

```
def lambda1(doc):
  return(1 if `rocket`
           in doc
           else ABSTAIN)
```

Users write heuristics

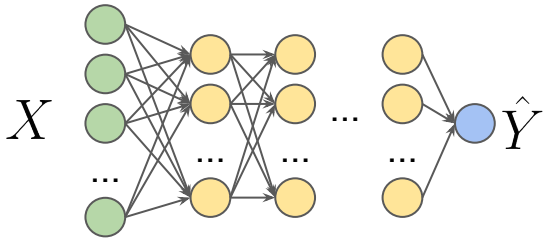
2



$$\hat{Y} = P(Y|\theta, \lambda_1, \ldots, \lambda_m)$$

Heuristics are used to model latent labels

3



The probabilistic labels are used to train an end-model

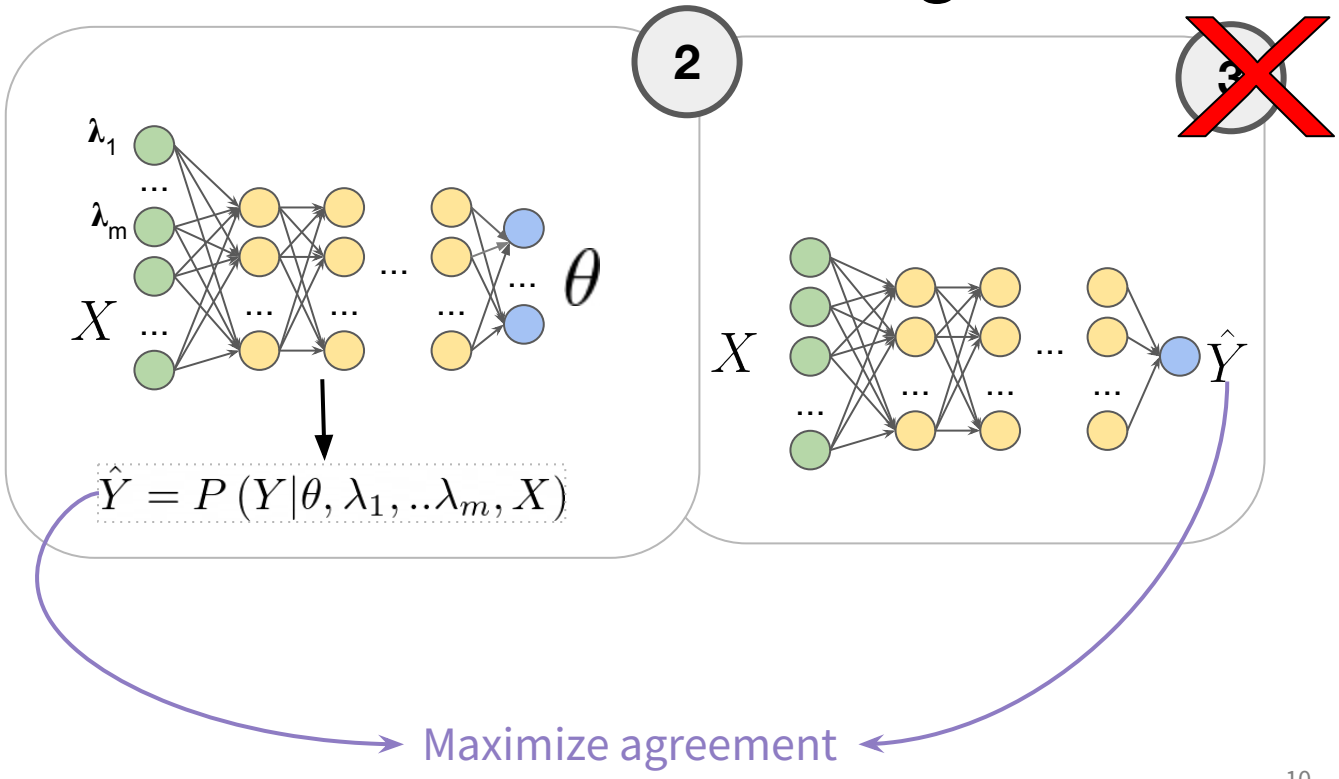
WeaSEL: Weakly Supervised End-to-end Learning

1

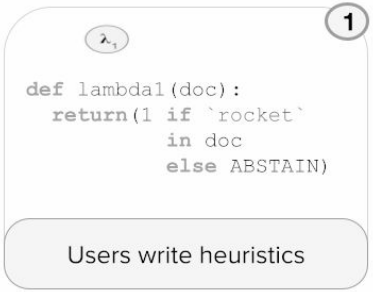
λ_1

```
def lambda1(doc):
  return(1 if `rocket`
           in doc
           else ABSTAIN)
```

Users write heuristics

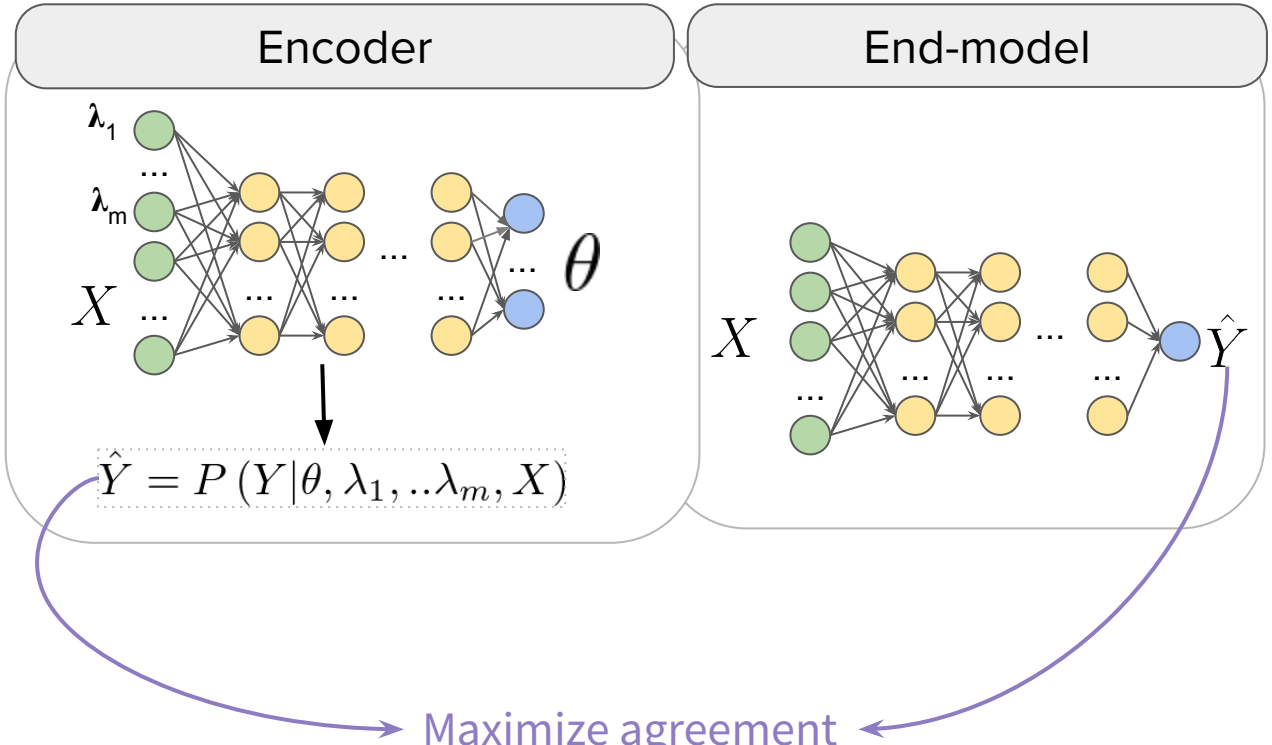


WeaSEL: Weakly Supervised End-to-end Learning



Algorithm 1 `WeaSEL`: The proposed Weakly Supervised End-to-end Learning algorithm for learning from multiple weak supervision sources.

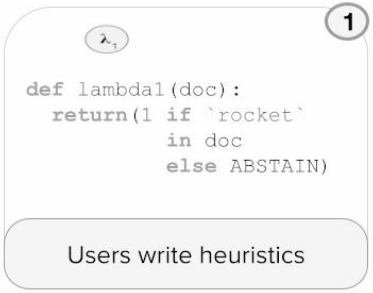
input: batch size n, networks e, f, inverse temperatures τ_1, τ_2, noise-aware loss function L, class balance $P(y)$.
for sampled minibatch $\{z^{(k)} = (\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)})\}_{k=1}^{n}$ **do**
  for all $k \in \{1, \ldots, n\}$ **do**
    # Produce accuracy scores for all weak sources
    $\theta\left(z^{(k)}\right) = \text{softmax}\left(e(z^{(k)})\tau_1\right)$
    # Generate probabilistic labels
    define $\mathbf{s}^{(k)}$ **as** $\mathbf{s}^{(k)} = \theta(z^{(k)})^T \bar{\boldsymbol{\lambda}}^{(k)}$
    $y_e^{(k)} = P_\theta(y|\boldsymbol{\lambda}^{(k)}) = \text{softmax}\left(\mathbf{s}^{(k)}\tau_2\right) \odot P(y)$
    # Downstream model forward pass
    $y_f^{(k)} = f(\mathbf{x}^{(k)})$
  end for
  $\mathcal{L}_f = \frac{1}{n}\sum_{k=1}^{n} L\left(y_f^{(k)}, \texttt{stop-grad}\left(y_e^{(k)}\right)\right)$
  $\mathcal{L}_e = \frac{1}{n}\sum_{k=1}^{n} L\left(y_e^{(k)}, \texttt{stop-grad}\left(y_f^{(k)}\right)\right)$
  update e to minimize $\mathcal{L}_e$, and f to minimize $\mathcal{L}_f$
end for
return downstream network $f(\cdot)$



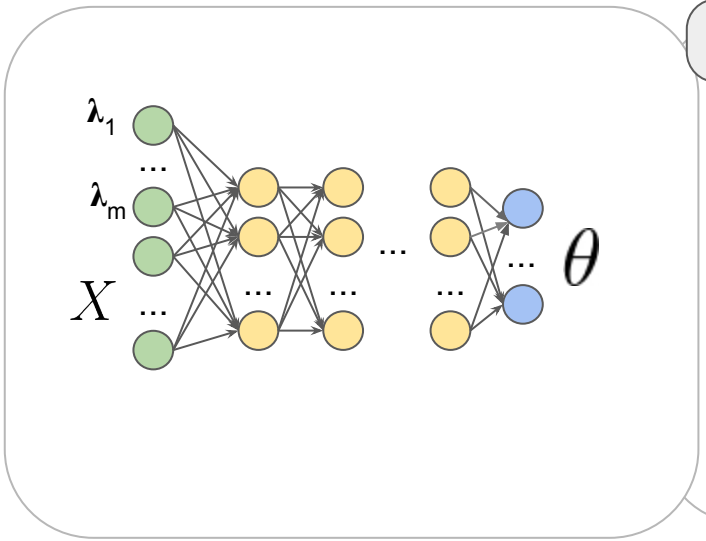
Our contributions

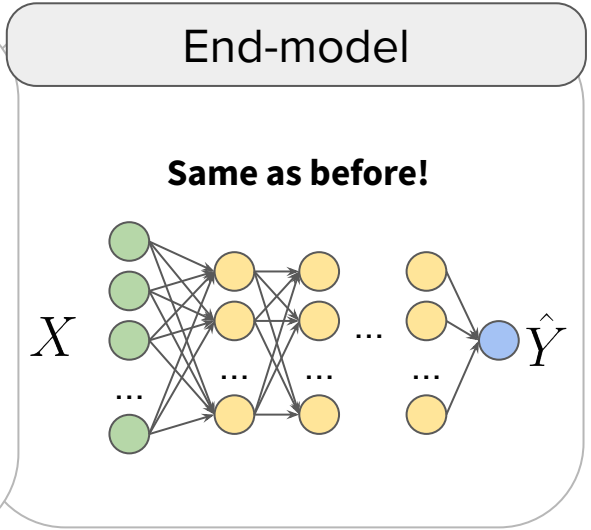
- introduce WeaSEL: A flexible, end-to-end method for learning models from multiple sources of weak supervision.
- empirically demonstrate that the method is robust to adversarial sources and highly correlated heuristics.
- release an open-source system for arbitrary PyTorch end-models
  - https://github.com/autonlab/weasel
- our method outperforms, by as much as 6.1 F1 points, state-of-the-art latent label modeling approaches on 4 out of 5 benchmark datasets, and achieves state-of-the-art performance on a crowdsourcing dataset against methods specifically designed for this setting

WeaSEL



WeaSEL: Weakly Supervised End-to-end Learning





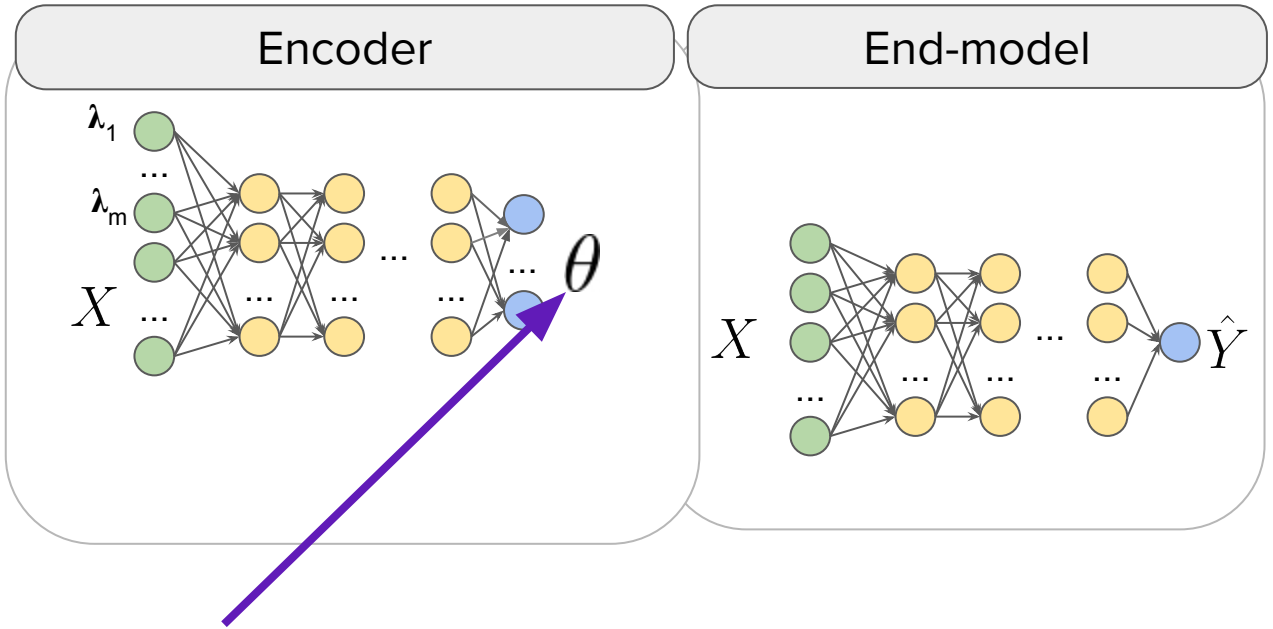
WeaSEL: Weakly Supervised End-to-end Learning

```
λ1                                  1
def lambda1(doc):
  return(1 if `rocket`
            in doc
            else ABSTAIN)
```

Users write heuristics

Algorithm 1 `WeaSEL`: The proposed Weakly Supervised End-to-end Learning algorithm for learning from multiple weak supervision sources.

input: batch size n, networks e, f, inverse temperatures τ_1, τ_2, noise-aware loss function L, class balance $P(y)$.
for sampled minibatch $\{z^{(k)} = (\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)})\}_{k=1}^{n}$ **do**
  for all $k \in \{1, \ldots, n\}$ **do**
    # Produce accuracy scores for all weak sources
    $\theta\left(z^{(k)}\right) = \text{softmax}\left(e(z^{(k)})\tau_1\right)$



Encoder predicts the accuracy of each heuristic

Accuracy may vary across samples!

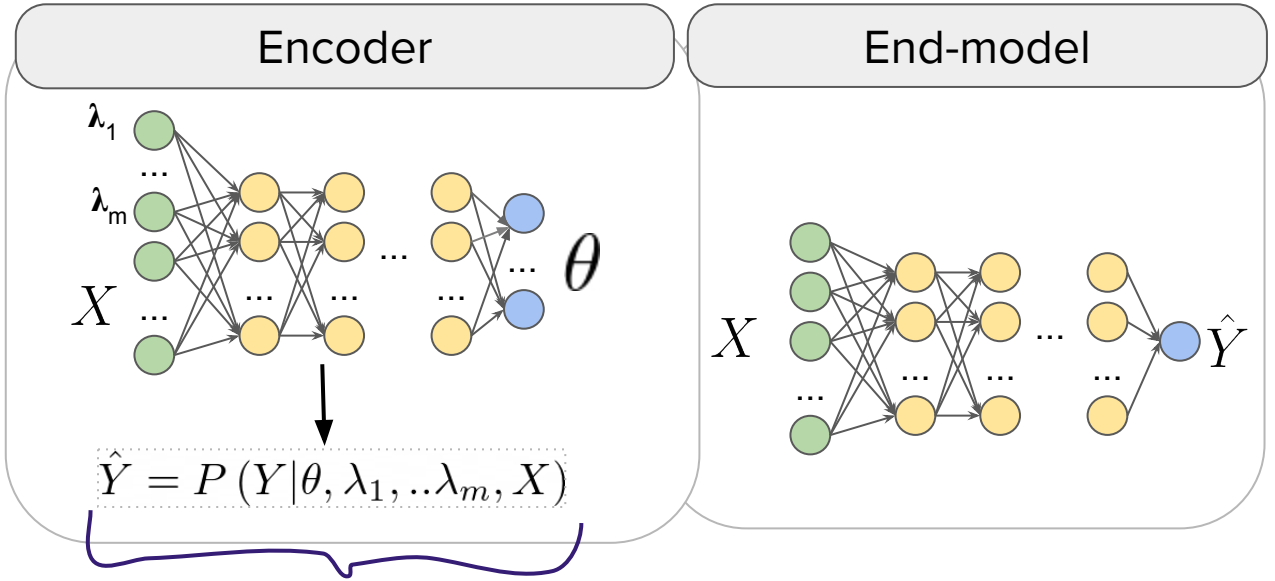
WeaSEL: Weakly Supervised End-to-end Learning

```
λ1                                  1
def lambda1(doc):
  return(1 if `rocket`
            in doc
            else ABSTAIN)
```

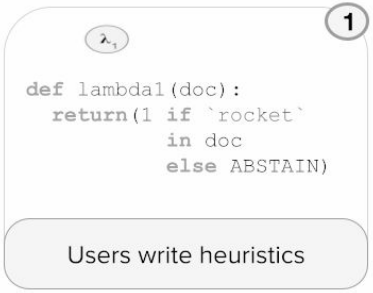
Users write heuristics

Algorithm 1 `WeaSEL`: The proposed Weakly Supervised End-to-end Learning algorithm for learning from multiple weak supervision sources.

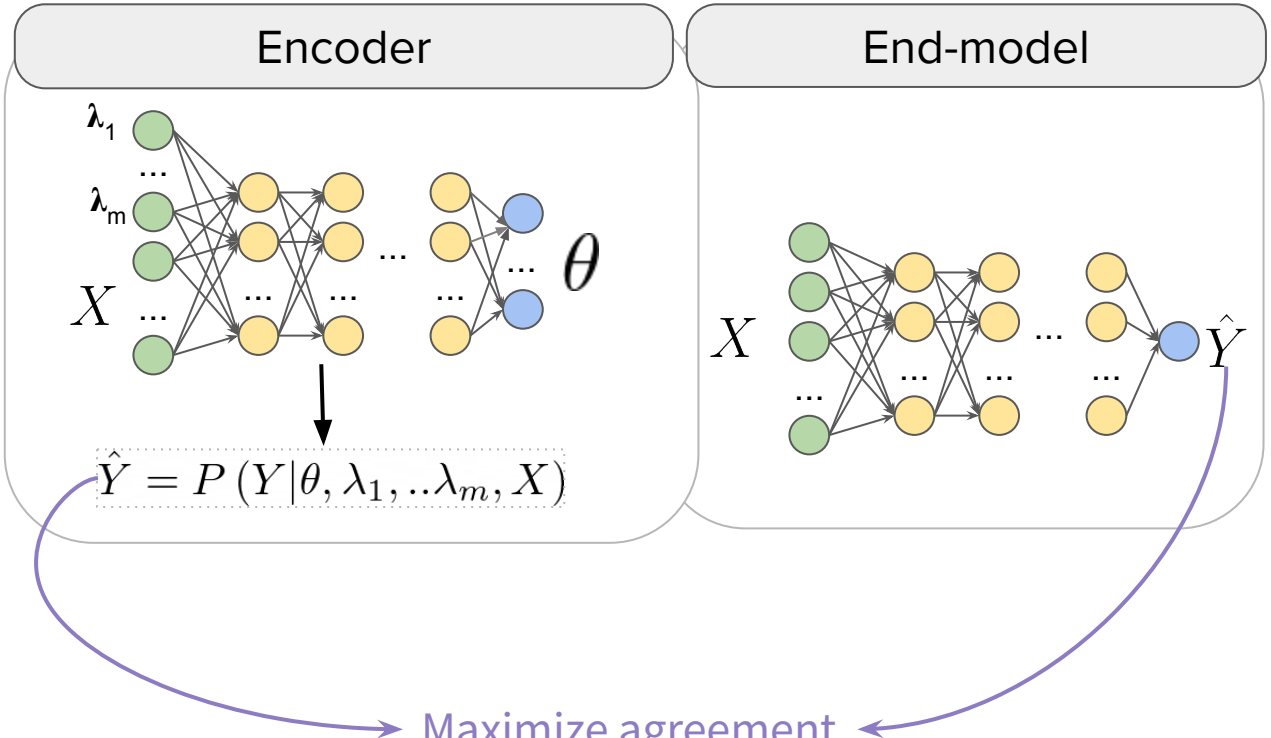
input: batch size n, networks e, f, inverse temperatures τ_1, τ_2, noise-aware loss function L, class balance $P(y)$.
for sampled minibatch $\{z^{(k)} = (\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)})\}_{k=1}^{n}$ **do**
  for all $k \in \{1, \ldots, n\}$ **do**
    # Produce accuracy scores for all weak sources
    $\theta\left(z^{(k)}\right) = \text{softmax}\left(e(z^{(k)})\tau_1\right)$
    # Generate probabilistic labels
    define $\mathbf{s}^{(k)}$ **as** $\mathbf{s}^{(k)} = \theta(z^{(k)})^T \bar{\lambda}^{(k)}$
    $y_e^{(k)} = P_\theta(y|\boldsymbol{\lambda}^{(k)}) = \text{softmax}\left(\mathbf{s}^{(k)}\tau_2\right) \odot P(y)$



Same weighted aggregation as before!



WeaSEL: Weakly Supervised End-to-end Learning



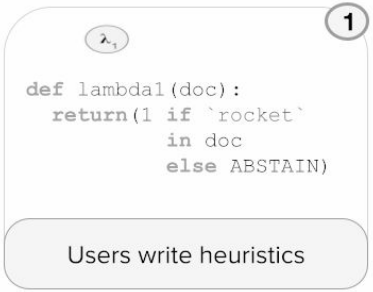
Algorithm 1 `WeaSEL`: The proposed Weakly Supervised End-to-end Learning algorithm for learning from multiple weak supervision sources.

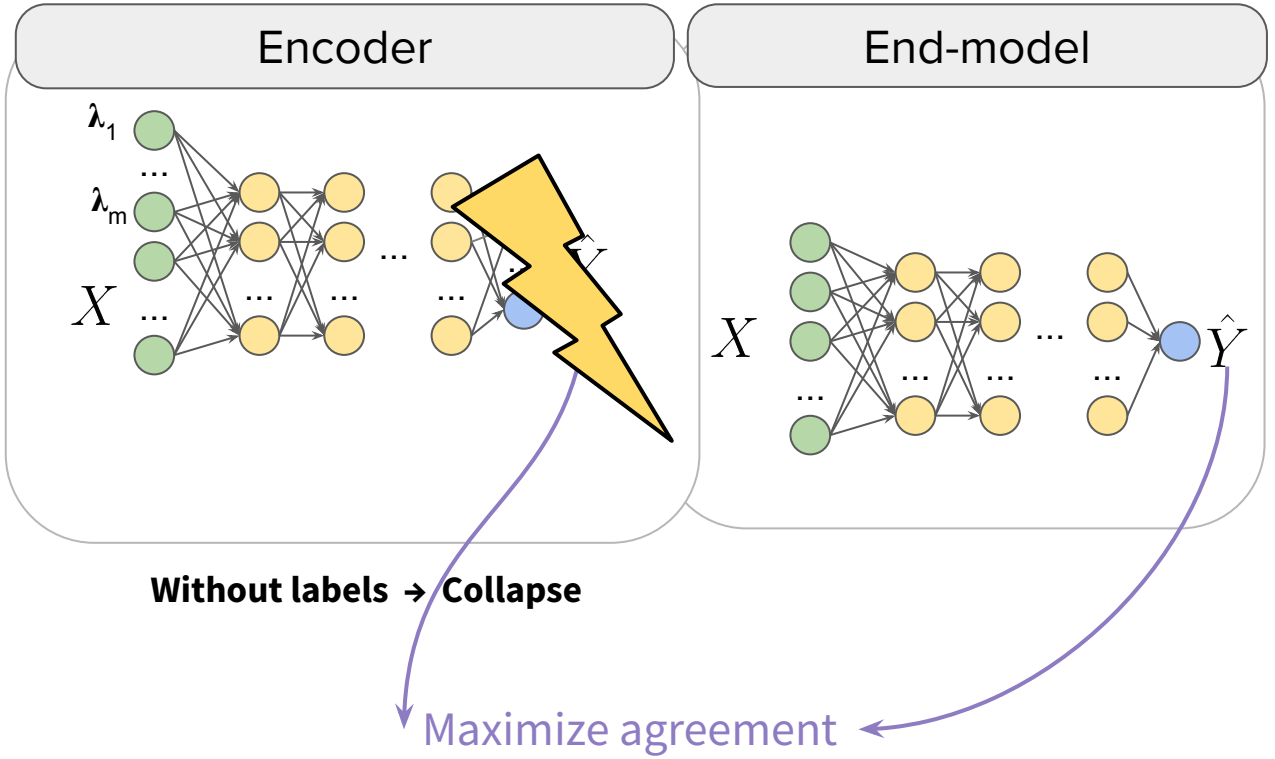
input: batch size n, networks e, f, inverse temperatures τ_1, τ_2, noise-aware loss function L, class balance $P(y)$.
for sampled minibatch $\{z^{(k)} = (\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)})\}_{k=1}^{n}$ **do**
  for all $k \in \{1, \ldots, n\}$ **do**
    # Produce accuracy scores for all weak sources
    $\theta\left(z^{(k)}\right) = \text{softmax}\left(e(z^{(k)})\tau_1\right)$
    # Generate probabilistic labels
    define $\mathbf{s}^{(k)}$ **as** $\mathbf{s}^{(k)} = \theta(z^{(k)})^T \overline{\boldsymbol{\lambda}}^{(k)}$
    $y_e^{(k)} = P_\theta(y|\,\boldsymbol{\lambda}^{(k)}) = \text{softmax}\left(\mathbf{s}^{(k)}\tau_2\right) \odot P(y)$
    # Downstream model forward pass
    $y_f^{(k)} = f(\mathbf{x}^{(k)})$
  end for
  $\mathcal{L}_f = \frac{1}{n}\sum_{k=1}^{n} L\left(y_f^{(k)}, \texttt{stop-grad}\left(y_e^{(k)}\right)\right)$
  $\mathcal{L}_e = \frac{1}{n}\sum_{k=1}^{n} L\left(y_e^{(k)}, \texttt{stop-grad}\left(y_f^{(k)}\right)\right)$
  update e to minimize $\mathcal{L}_e$, and f to minimize $\mathcal{L}_f$
end for
return downstream network $f(\cdot)$

Maximize agreement:

Experiments

Predict accuracy scores, not labels





Datasets

Table 3: Dataset details, where training, validation and test set sizes are N_{train}, N_{val}, N_{test} respectively, and f denotes the downstream model type. We also report the total coverage Cov. of all LFs, which refers to the percentage of training samples which are labeled by at least one LF (the rest is not used). For IMDB we used two different sets of labeling functions of sizes 12 and 136.

Dataset	#LFs	N_{train}	Cov. (in %)	N_{val}	N_{test}	f
Spouses	9	$22,254$	25.8	2811	2701	LSTM
BiasBios	99	$12,294$	81.8	250	$12,044$	MLP
IMDB	12	$25k$	88.0	250	$24,750$	MLP
IMDB	136	$25k$	83.1	250	$24,750$	MLP
Amazon	175	$160k$	65.5	500	$39,500$	MLP

Results

Table 1: Test F1 performance of various label models over seven runs using different random seeds, are averaged out ± standard deviation. The top 2 performance scores are highlighted as **First**, **Second**. Triplet-median [10] is not listed as it only converged for IMDB with 12 LFs (F1 = 73.0 ± 0.22), and Spouses (F1 = 48.7 ± 1.0). Sup. (Val. set) is the performance of the downstream model trained in a supervised manner on the labeled validation set. The rest are state-of-the-art latent label models. For reference, we also report the *Ground truth* performance of a fully supervised model trained on true training labels (which are unused by all other models, and not available for Spouses).

Model	**Spouses** (9 LFs)	**ProfTeacher** (99 LFs)	**IMDB** (136 LFs)	**IMDB** (12 LFs)	**Amazon** (175 LFs)
Ground truth	–	90.65 ± 0.29	86.72 ± 0.40	86.72 ± 0.40	92.93 ± 0.68
Sup. (Val. set)	20.4 ± 0.2	73.34 ± 0.00	68.76 ± 0.00	68.76 ± 0.00	84.18 ± 0.00
Snorkel	48.79 ± 2.69	85.12 ± 0.54	**82.22 ± 0.18**	**74.45 ± 0.58**	80.54 ± 0.41
Triplet	45.88 ± 3.64	74.43 ± 10.59	75.36 ± 1.92	73.15 ± 0.95	75.44 ± 3.21
Triplet-Mean	**49.94 ± 1.47**	82.58 ± 0.32	79.03 ± 0.26	73.18 ± 0.23	79.44 ± 0.68
Majority vote	40.67 ± 2.01	**85.44 ± 0.37**	80.86 ± 0.28	74.13 ± 0.31	**84.20 ± 0.52**
`WeaSEL`	**51.98 ± 1.60**	**86.98 ± 0.45**	**82.10 ± 0.45**	**77.22 ± 1.02**	**86.60 ± 0.71**

Evaluation on a crowdsourcing-worker aggregation dataset

Table 2: Test accuracy scores on the crowd-sourced, multi-class LabelMe image classification dataset.

Model	Accuracy
Majority vote	79.23 ± 0.5
MBEM [26]	76.84 ± 0.4
DoctorNet [21]	81.31 ± 0.4
CrowdLayer [34]	82.83 ± 0.4
AggNet [1]	84.35 ± 0.4
MaxMIG [8]	$\mathbf{85.45 \pm 1.0}$
Snorkel+CE	82.89 ± 0.7
WeaSEL+CE	82.46 ± 0.8
Snorkel+MIG	85.15 ± 0.8
WeaSEL+MIG	$\mathbf{86.36 \pm 0.3}$

Key design choices

WeaSEL is more robust against “bad” LFs

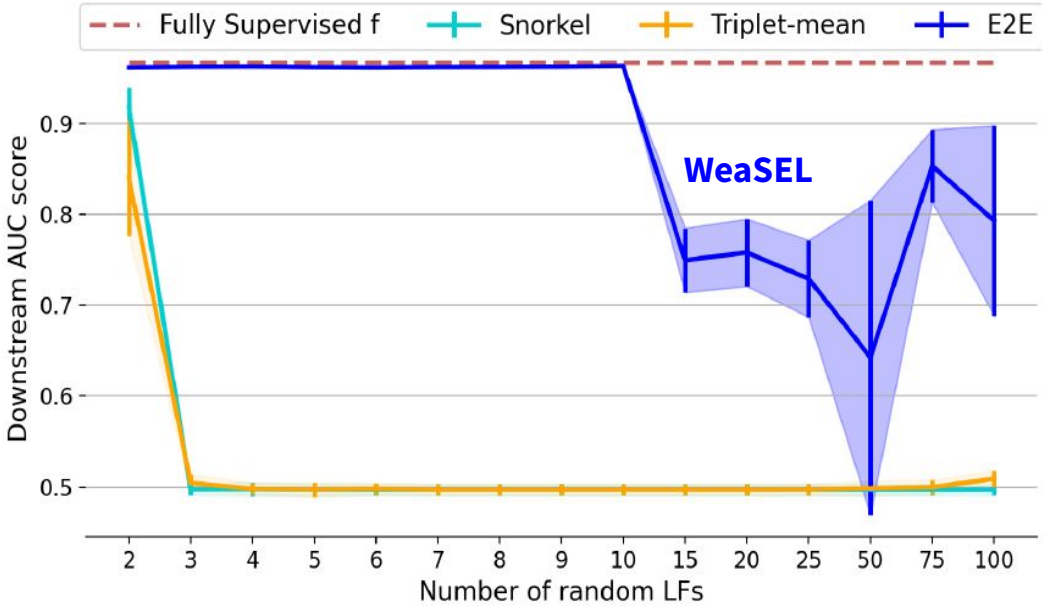


Figure 4: We start with a 100% accurate LF (i.e. ground truth labels) and incrementally add new, independent LFs that are no better than a random guess. `WeaSEL` recovers the performance of training directly on the ground truth labels (Fully Supervised f), for up to 10 such randomly voting LFs that are independent of each other. The PGM-based prior work, rapidly degrades in performance (AUC ≈ 0.5) and is not able to recover any of the 100% accurate signal of the true-labels-LF, as soon as the LF set is corrupted by three or more random LFs. Performances are averaged out over five random seeds, and the standard deviation is shaded. For more details, see F.2.2

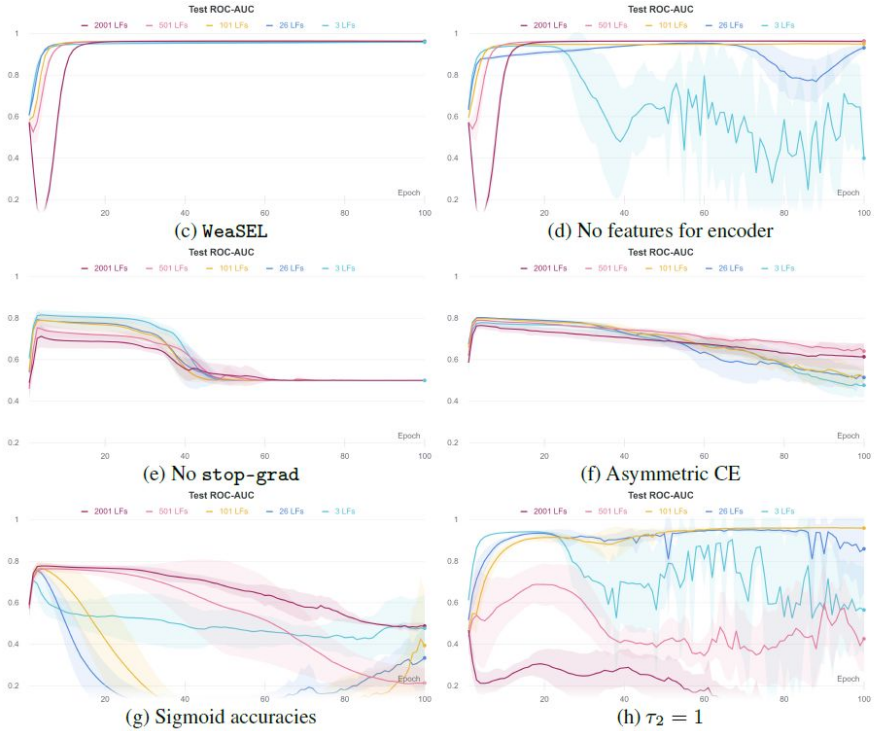


Figure 5: We start with a 100% accurate LF (i.e. ground truth labels) and plot test performances at each training epoch for a varying number of duplicates $\in \{2, 25, 100, 500, 2000\}$ of a LF that is no better than a coin flip. Performances are averaged out over five random seeds, and the standard deviation is shaded. More details are given in F.2.1.

Adding duplicated, “bad” heuristics (up to 2000) can break ablated versions of WeaSEL, but not WeaSEL

Table 5: Ablative study on the subcomponents of our algorithm as in Alg. 1 (over 5 random seeds). In each row below we change exactly one component of `WeaSEL` and report the resulting F1 score. Note that the scores for `WeaSEL` are slightly different to the ones in the main results table, since they were run separately, with fewer seeds, and for only one learning rate (1e-4). Configurations that **outperform base `WeaSEL` are highlighted in bold font**, while the **four worst performing configurations** are highlighted in red for each dataset. Note that bold font does not indicate significant differences.

Change	ProfTeacher	IMDB-136 LFs	IMDB-12 LFs	Amazon
`WeaSEL`	86.8 ± 0.4	82.1 ± 0.7	77.3 ± 0.5	86.6 ± 0.5
$\theta(\boldsymbol{\lambda}, \mathbf{x}) = \theta(\boldsymbol{\lambda})$	85.6 ± 1.6	82.1 ± 0.5	75.9 ± 0.8	86.6 ± 0.4
Linear e	81.9 ± 0.7	80.0 ± 0.6	73.2 ± 0.6	82.6 ± 0.5
1 hidden layer e	$\mathbf{87.1 \pm 0.7}$	81.8 ± 0.6	76.8 ± 0.9	85.3 ± 0.8
75x50x25x50x75 e	84.3 ± 2.1	81.9 ± 0.6	75.8 ± 1.1	86.1 ± 0.6
$\tau_1 = 2$	86.7 ± 1.0	81.9 ± 0.3	77.3 ± 0.5	85.5 ± 1.0
$\tau_1 = 1/2$	86.5 ± 0.8	81.8 ± 0.5	76.0 ± 1.4	86.4 ± 0.3
$\tau_1 = 1/4$	84.5 ± 1.2	81.8 ± 0.2	73.9 ± 0.9	85.6 ± 1.0
$\tau_2 = 1$	85.2 ± 1.6	$\mathbf{82.2 \pm 0.4}$	76.6 ± 1.0	84.3 ± 1.2
$\tau_2 = m$	86.1 ± 0.7	81.2 ± 0.6	76.4 ± 0.4	85.7 ± 0.2
No BatchNorm	82.6 ± 1.4	81.9 ± 0.5	74.7 ± 0.7	85.3 ± 0.8
1e-4 weight decay	$\mathbf{87.4 \pm 0.4}$	80.9 ± 1.3	$\mathbf{77.9 \pm 0.6}$	85.2 ± 0.5
MIG loss	86.7 ± 0.4	78.7 ± 0.4	74.1 ± 0.4	84.7 ± 1.8
L1 loss	86.2 ± 0.6	81.1 ± 0.5	75.6 ± 0.9	84.1 ± 0.9
Squared Hellinger loss	$\mathbf{87.4 \pm 0.3}$	$\mathbf{82.2 \pm 0.6}$	75.7 ± 1.1	86.3 ± 0.4
CE(P_f, P_e) asymm. loss	$\mathbf{77.3 \pm 3.7}$	$\mathbf{77.7 \pm 1.1}$	71.7 ± 0.3	$\mathbf{78.7 \pm 1.2}$
CE(P_e, P_f) asymm. loss	$\mathbf{73.1 \pm 6.8}$	$\mathbf{71.9 \pm 1.9}$	$\mathbf{69.7 \pm 0.7}$	$\mathbf{70.1 \pm 1.1}$
No `stop-grad`	$\mathbf{80.4 \pm 2.1}$	$\mathbf{76.2 \pm 0.5}$	$\mathbf{71.0 \pm 0.6}$	79.3 ± 0.6
$\theta(\boldsymbol{\lambda}, \mathbf{x}) = \sqrt{m} \cdot \text{sigmoid}(e(\boldsymbol{\lambda}, \mathbf{x}))$	85.5 ± 0.6	81.8 ± 0.5	$\mathbf{78.0 \pm 0.7}$	$\mathbf{86.9 \pm 0.3}$
$\theta(\boldsymbol{\lambda}, \mathbf{x}) = \text{ReLU}(e(\boldsymbol{\lambda}, \mathbf{x}))+$ 1e-5	83.0 ± 2.3	78.3 ± 1.1	$\mathbf{69.1 \pm 2.1}$	$\mathbf{74.2 \pm 2.7}$
$\theta(\boldsymbol{\lambda}, \mathbf{x}) = \text{Tanh}(e(\boldsymbol{\lambda}, \mathbf{x}))$	$\mathbf{71.9 \pm 4.0}$	$\mathbf{67.0 \pm 0.8}$	$\mathbf{67.0 \pm 1.1}$	$\mathbf{67.3 \pm 1.1}$

Practical aspects

- Early-stopping on a small labeled validation set
- In binary classification: Tune decision threshold
- When the end-model is slow to train, the process of finding a “final” set of heuristics is slowed down with WeaSEL → use Snorkel or less complex end-model

Future work

- How to completely avoid collapses? How to detect them without validation set?
  - Use a small fraction of hard labels! But, how many? And, how to schedule them across epochs?
- Other training tricks, e.g.
  - label smoothing
  - label confidence thresholding
  - Use (exponential) moving average (EMA) weights to generate the target labels
  - Draw inspiration from GAN training
- Use probabilistic heuristics!
- Applicable to regression?

Conclusion

- We proposed WeaSEL, a new approach for end-to-end learning of neural network models for classification from, exclusively, multiple sources of weak supervision that streamlines prior latent variable models.
- Strong empirical performance and outperforms several state-of-the-art crowdsourcing methods on a crowdsourcing task.
- More robust to dependencies and correlations between the heuristics
- Works with discrete and probabilistic labeling functions and can utilize various neural network designs for probabilistic label generation.

Carnegie Mellon University

Auton Lab

Thanks! :)



End-to-End Weak Supervision,
Salva Rühling Cachay, Benedikt Boecking, Artur Dubrawski,
In Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS), 2021

Code: https://github.com/autonlab/weasel
Contact: *salvaruehling@gmail.com*