WeaSuL 2021

Proceedings of the First Workshop on Weakly Supervised Learning (WeaSuL)

May 7, 2021
Co-located with ICLR (Online)

Introduction

Welcome to WeaSuL 2021, the First Workshop on Weakly Supervised Learning, co-located with ICLR 2021.

Deep learning relies on massive training sets of labeled examples to learn from - often tens of thousands to millions to reach peak predictive performance, but large amounts of training data are only available for very few standardized learning problems. Even small variations of the problem specification or changes in the data distribution would necessitate re-annotation of large amounts of data.

However, domain knowledge can often be expressed by sets of prototypical descriptions: For example, vision experts can exploit meta information for image labeling, linguists can describe discourse phenomena by prototypical realization patterns, social scientists can specify events of interest by characteristic key phrases, and bio-medical researchers have databases of known interactions between drugs or proteins that can be used for heuristic labeling. These knowledge-based descriptions can be either used as rule-based predictors or as labeling functions for providing partial data annotations. The growing field of weak supervision provides methods for refining and generalizing such heuristic-based annotations in interaction with deep neural networks and large amounts of unannotated data.

In this workshop, we want to advance theory, methods and tools for allowing experts to express prior coded knowledge for automatic data annotations that can be used to train arbitrary deep neural networks for prediction. The ICLR 2021 Workshop on Weak Supervision aims at advancing methods that help modern machine-learning methods to generalize from knowledge provided by experts, in interaction with observable (unlabeled) data.

We called for both long and short papers and received 26 submissions, all of which were double-blindly reviewed by a pool of 29 reviewers. In total, 15 papers were accepted. All the accepted contributions are listed in these Proceedings and those submitted as archival are included in full text.

Learning with weak supervision is both studied from a theoretical perspective as well as applied to a variety of tasks from areas like natural language processing and computer vision. Therefore, the workshop brought together researchers from a wide range of fields, also bridging innovations from academia and the requirements of industry settings.

The program of the workshop, besides 3 oral paper presentations and 12 posters in 2 poster sessions, included invited talks by Marine Carpuat, Heng Ji, Lu Jiang, Dan Roth and Paroma Varma. It closed with a panel discussion with the invited speakers. Snorkel AI provided funding to sponsor ICLR registrations to increase diversity.

The WeaSuL Workshop Organizers

Organizers:

Michael A. Hedderich, Saarland University (Germany)
Benjamin Roth, University of Vienna (Austria)
Katharina Kann, University of Colorado Boulder (USA)
Barbara Plank, IT University of Copenhagen (Denmark)
Alex Ratner, University of Washington (USA)
Dietrich Klakow, Saarland University (Germany)

Program Committee:

Abhijeet Awasthi, Indian Institute of Technology Bombay
Andreas Baumann, University of Vienna
Bo Han, Hong Kong Baptist University
Chaojun Xiao, Tsinghua University
Curtis G. Northcutt, Massachusetts Institute of Technology
Daniel Y. Fu, Stanford University
David Adelani, Saarland University
Dawei Zhu, Saarland University
Edwin Simpson, University of Bristol
Erion Çano, University of Vienna
Ivan Habernal, TU Darmstadt
Jacob Goldberger, Bar-Ilan Univeristy
Judith Gaspers, Amazon
Julia Hockenmaier, University of Illinois at Urbana-Champaign
Khaled K. Saab, Stanford University
Lukas Lange, Bosch Center for Aritificial Intelligence
Mahaveer Jain, Facebook
Marina Speranskaya, LMU Munich
Nils Rethmeier, DFKI Berlin
Pierre Lison, University of Oslo
Quanming Yao, 4Paradigm
Sarah Hooper, Stanford University
Seffen Eger, TU Darmstadt
Shiran Dudy, Oregon Health & Science University
Stephen H. Bach, Brown University
Thomas Trost, Saarland University
Tongliang Liu, University of Sydney
Vincent Chen, Snorkel AI
Xiang Dai, University of Sydney

Invited Speaker:

Marine Carpuat, University of Maryland
Heng Ji, University of Illinois at Urbana-Champaign
Lu Jiang, Google Research
Dan Roth, University of Pennsylvania
Paroma Varma, Snorkel AI

Accepted Papers

TADPOLE: Task ADapted Pre-training via anOmaLy dEtection
Vivek Madan, Ashish Khetan and Zohar Karnin

CIGMO: Learning categorical invariant deep generative models from grouped data
Haruo Hosoya

Handling Long-Tail Queries with Slice-Aware Conversational Systems
Cheng Wang, Sun Kim, Taiwoo Park,Sajal Choudhary, Sunghyun Park, Young-Bum Kim, Ruhi Sarikaya and Sungjin Lee

Tabular Data Modeling via Contextual Embeddings
Xin Huang, Ashish Khetan, Milan Cvitkovic and Zohar Karnin
https://arxiv.org/abs/2012.06678

Pre-Training by Completing Points Cloud
Hanchen Wang, Liu Qi, Xiangyu Yue, Matt Kusner and Joan Lasenby | (non-archival)

AutoTriggER: Named Entity Recognition with Auxiliary Trigger Extraction
Dong-Ho Lee, Ravi Kiran Selvam, Sheikh Muhammad Sarwar, Bill Yuchen Lin, Fred Morstatter, Jay Pujara, Elizabeth Boschee, James Allan and Xiang Ren | (non-archival)

Active WeaSuL: Improving Weak Supervision with Active Learning
Samantha R Biegel, Rafah El-Khatib, Luiz Otavio Vilas Boas Oliveira, Max Baak and Nanne Aben
https://arxiv.org/abs/2104.14847

Dependency Structure Misspecification in Multi-Source Weak Supervision Models
Salva Rühling Cachay, Benedikt Boecking and Artur Dubrawski
https://arxiv.org/abs/2106.10302

Weakly-Supervised Group Disentanglement using Total Correlation
Linh Tran, Saeid Asgari Taghanaki, Amir Hosein Khasahmadi, Aditya Sanghi

Better adaptation to distribution shifts with Robust Pseudo-Labeling
Evgenia Rusak, Steffen Schneider, Peter Gehler, Oliver Bringmann, Bernhard Schölkopf, Wieland Brendel and Matthias Bethge | (non-archival)

Transformer Language Models as Universal Computation Engines
Kevin Lu, Aditya Grover, Pieter Abbeel and Igor Mordatch | (non-archival)

Using system context information to complement weakly labeled data
Matthias Meyer, Michaela Wenner, Clément Hibert, Fabian Walter and Lothar Thiele

Is Disentanglement all you need? Comparing Concept-based & Disentanglement Approaches
Dmitry Kazhdan, Botty Dimanov, Helena Andres Terre, Pietro Lió, Mateja Jamnik and Adrian Weller | (non-archival)

Weakly Supervised Multi-task Learning for Concept-based Explainability
Vladimir Balayan, Catarina G Belém, Pedro Saleiro and Pedro Bizarro
https://arxiv.org/abs/2104.12459

Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks
Curtis G Northcutt, Anish Athalye and Jonas Mueller
https://arxiv.org/abs/2103.14749

TADPOLE: TASK ADAPTED PRE-TRAINING VIA ANOMALY DETECTION

Vivek Madan
AWS AI Labs
vivmadan@amazon.com

Ashish Khetan
AWS AI Labs
khetan@amazon.com

Zohar Karnin
AWS AI Labs
zkarnin@amazon.com

ABSTRACT

The paradigm of pre-training followed by finetuning has become a standard procedure for NLP tasks, with a known problem of domain shift between the pre-training and downstream corpus. Previous works have tried to mitigate this problem with additional pre-training, either on the downstream corpus itself when it is large enough, or on a manually curated unlabeled corpus of a similar domain. In this paper, we address the problem for the case when the downstream corpus is too small for additional pre-training. We propose TADPOLE, a task adapted pre-training framework based on data selection techniques adapted from *Domain Adaptation*. We formulate the data selection as an anomaly detection problem that unlike existing methods works well when the downstream corpus is limited in size. It results in a scalable and efficient unsupervised technique that eliminates the need for any manual data curation. We evaluate our framework on eight tasks across four different domains: Biomedical, Computer Science, News, and Movie reviews, and compare its performance against competitive baseline techniques from the area of Domain Adaptation. Our framework outperforms all the baseline methods. On large datasets we get an average gain of 0.3% in performance but on small datasets with less than 5K training examples, we get a much higher gain of 1.8%. This shows the efficacy of domain adapted finetuning when the task dataset is small.

1 INTRODUCTION

Pre-trained language models such as ELMo (Peters et al., 2018), GPT (Radford et al., 2018), BERT (Devlin et al., 2018), Transformer-xl (Dai et al., 2019) and XLNet (Yang et al., 2019) have become a key component in solving virtually all natural language tasks. These models are pre-trained on large amount of cross-domain data ranging from Wikipedia to Book corpus to news articles to learn powerful representations. A generic approach for using these models consists of two steps: (a) Pre-training: train the model on an extremely large general domain corpus, e.g. with masked language model loss; (b) Finetuning: finetune the model on labeled task dataset for the downstream task.

Even though the approach of pre-training followed by fine-tuning has been very successful, it suffers from *domain shift* when applied to tasks containing text from a domain that is not sufficiently represented in the pre-training corpus. An immediate way of solving the problem is to pre-train the model on task domain data instead of the general domain data. For a handful of very popular task domains, the research community invested time and resources to collect a large domain-specific data corpus and pre-train a language model on it. The models include BioBERT pre-trained on biomedical text (Lee et al., 2020), ClinicalBERT pre-trained on clinical notes (Huang et al., 2019), SciBERT pre-trained on semantic scholar corpus (Beltagy et al., 2019), and FinBERT pre-trained on financial documents (Araci, 2019). These models achieve significant gain in performance over a model trained on general domain data, when the downstream task belongs to the respective domains.

These papers demonstrate how useful it can be to shift the domain of the pre-trained model. However, the approach is expensive and time consuming as it requires collecting gigabytes of domain data for each new task. The long-tail of domains remains left behind without a realistic solution. To mitigate this, in absence of the huge task domain data, a different known approach is to collect a medium (MBs, not GBs) amount of unlabeled task data, and adapt the pre-trained (on general data) model by e.g. extending the pre-training procedure on the unlabeled data (Howard & Ruder, 2018; Gururangan

et al., 2020). Such task adapted pre-training approach achieves relatively smaller gain but is less expensive. Although this approach is cheaper in terms of manual labor when compared to domain adapted BERT, it still requires an effort to collect unlabeled data. It requires much more data than what is needed for only fine-tuning. This is often impossible to achieve, for example when data is highly sensitive. In this paper we propose a solution to this challenging problem, providing domain adaptation to the pre-trained model without the need for any manual effort of data collection.

The high level idea is quite intuitive. Given a generic pre-training data containing text from multiple domains, we filter the available general domain to contain only pieces that are similar to the downstream task corpus. By continuing the pre-training process on this adapted corpus we achieve a better tuned pre-trained model. Figure 1 illustrate the feasibility of this approach with an example downstream task from a medical domain and highlighted text from a news article available in a general domain corpus. The key for a successful implementation is finding the best way of evaluating the similarity of a given snippet to the downstream task.

RCT20K TASK DATA
To investigate the efficacy of 6 weeks of daily low-dose oral prednisolone in improving pain, mobility, and systemic los-grade . . . [OBJECTIVE]
A total of 125 patients with primary knee OA were randomize . . . [METHODS]
Outcome measures included pain reduction and systemic inflammation markers . . . [METHODS]

News Article
For as much as we workout warriors recite that whole "no pain, no gain" mantra, we sure do pop a lot of painkillers. A recent article published in. . . These popular medicines, known as non-steroidal anti-inflammatory drugs, or NSAIDs, work by suppressing inflammation. . . the article kind of blows past is the fact plenty of racers . . .

Figure 1: Identification of task-data (top panel, medical data) in general domain corpus (bottom panel).

Although not many methods exist to solve the problem of domain shift in the context of pretraining, literature on Domain Adaptation provides several methods for the core task of evaluating the above mentioned similarity. These previous approaches use either a simple language model (LM) (Moore & Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013; Wang et al., 2017b; van der Wees et al., 2017), or a hand crafted similarity score (Wang et al., 2017a; Plank & Van Noord, 2011; Remus, 2012; Van Asch & Daelemans, 2010). The LM based technique are often both over simplistic, and require a fairly large corpus of task data to create a reasonable LM. The hand crafted similarity scores can be seen as ad-hoc methods for distinguishing inliers from outliers (i.e., anomaly detection); they tend to be focused on individual tasks and do not generalize well.

We formulate the similarity evaluation task as that of anomaly detection and propose a Task ADapted Pre-training via anOmaLy dEtection (TADPOLE) framework. Indeed, anomaly detection methods given a domain of instance are able to provide a score for new instances assessing how likely they are to belong to the input domain. We exploit pre-trained models to get sentence representations that are in turn used to train an anomaly detection model. By using pre-trained models, our method is effective even for small text corpora. By taking advantage of existing anomaly detection methods, we replace hand-crafted rules with techniques proven to generalize well.

In what follows we discuss how we implement our technique and compare it with other data selection methods based on extensive experimental results. We start by filtering out the subset of general domain corpus most relevant to the task. To do this, we explore several anomaly detection methods and give a quantitative criterion to identify the best method for a given task data. Then, we start with a pre-trained model on the general domain corpus and run additional pre-training for only 5% more steps on the filtered corpus from different methods. This is followed by the regular finetuning on the labeled task data. We measure the performance gain as an improvement in accuracy of finetuned model with additional pre-training vs the accuracy of finetuned model without additional pre-training. To establish the performance gain of TADPOLE, we evaluate it on eight tasks across four domains: Biomedical, Computer Science, News, and Movie reviews. We investigate all aspects of TADPOLE by comparing its performance with various baselines based on its variants and the competitive methods available in literature. The main highlights of our work are as follows:

- We provide TADPOLE, a novel anomaly detection based framework for adapting pre-training for the downstream task. The framework is explained in detail and all its steps are justified via extensive ablation studies.

- TADPOLE is superior to all the baseline methods including (i) LM based relevance score (iii) Distance based relevance score (iii) Continued pre-training on the task data.
- Our method achieves an average 1.07% lift in accuracy over eight tasks from four different domains whereas the baselines achieve no more than 0.34% gain. In all individual tasks, our method is either on par or (statistical) significantly better than all alternatives.
- For tasks with small labeled dataset (less than 5K examples), our method achieves an even higher average lift of 1.82% in accuracy.
- For a task requiring little domain adaptation, GLUE sentiment analysis, our method achieves an improvement of 0.4% in accuracy.

2 Related Work

Since our focus is on Data Selection Methods, we only discuss the related work on Data Selection in Domain Adaptation here. We discuss the other Domain Adaptation techniques in Appendix A.

Data Selection: As discussed above, core of data selection is to determine the relevance weights that in turn modify the source domain to become more similar to the target domain. There has been a sequence of works in trying to find the relevance weights via language models Moore & Lewis (2010); Wang et al. (2017b); van der Wees et al. (2017). For instance, Moore & Lewis (2010), Axelrod et al. (2011) and Duh et al. (2013) train two language models, an in-domain language model on the target domain dataset (same as task domain in our case) and an out-of-domain language model on (a subset of) general domain corpus. Then, relevance score is defined as the difference in the cross-entropy w.r.t. two language models. These methods achieve some gain but have a major drawback. A crucial assumption they rely on: *there is enough in-domain data to train a reasonable in-domain language model.* This assumption is not true in most cases. For most tasks, we only have access to a few thousands or in some cases a few hundreds of examples which is not enough to train a reasonably accurate language model. Our techniques rely on text representations based on the available pre-trained model. As such, our similarity score does not rely on models that can be trained with a small amount of data.

Another line of work defines hand crafted domain similarity measures to assign relevance score and filter out text from a general domain corpus (Wang et al., 2017a; Plank & Van Noord, 2011; Remus, 2012; Van Asch & Daelemans, 2010; Gururangan et al., 2020). For instance, Wang et al. (2017a) define the domain similarity of a sentence as the difference between Euclidean distance of the sentence embedding from the mean of in-domain sentence embeddings and the mean of out-of-domain sentence embeddings. Plank & Van Noord (2011) and Remus (2012) define the similarity measure as the Kullback-Leibler (KL) divergence between the relative frequencies of words, character tetra-grams, and topic models. Van Asch & Daelemans (2010) define domain similarity as Rényi divergence between the relevant token frequencies. These are adhoc measures suitable only for the respective tasks, and can be seen as a manual task-optimized anomaly detection. They fail to generalize well for new tasks and domains. Ruder & Plank (2017) attempts to remedy this issue and tries to learn the correct combination of these metrics for each task. They learn the combination weight vector via Bayesian optimization. However, Bayesian optimization is infeasible for deep networks like BERT. Each optimization step of this process amounts to pre-training the model and finetuning it for the task. For Bayesian optimization to work well it requires repeating this process multiple times, which is prohibitively computationally expensive. Thus, they use models such as linear SVM classifier and LDA which do not yield state-of-the-art performance. In contrast, we propose a lightweight method - based on anomaly detection - that can be applied to state-of-the-art deep language models like BERT.

3 TADPOLE: Task ADapted Pre-training via anOmaLy dEtection

Language model and Downstream Tasks. A generic approach for using state-of-the-art language models such as ELMo, GPT, BERT, and XLNet is to pre-train them on an extremely large general domain corpus and then finetune the pre-trained model on the downstream labeled task data. There is evident correlation between model's pre-training loss and its performance on the downstream task after finetuning (Devlin et al., 2018). Our design is motivated by an observation, backed by empirical evidence, that the correlation is even stronger if we consider the pre-training loss not on the pre-training data but the downstream task data.

To make this distinction formal, let $\mathcal{D}$, $\mathcal{D}_{\text{in}}$ be the pre-training and task data. Let Θ denote the parameters of the language model and ℓ_{LM} denote the language model loss function. The pre-training loss on pre-training data ($L_{\text{LM}}(\Theta)$) and target data ($L^{\text{in}}_{\text{LM}}(\Theta)$) are defined as follows: $L_{\text{LM}}(\Theta) = \sum_{x \in \mathcal{D}} \ell_{\text{LM}}(x; \Theta)$, $L^{\text{in}}_{\text{LM}}(\Theta) = \sum_{x \in \mathcal{D}_{\text{in}}} \ell_{\text{LM}}(x; \Theta)$. To show that $L^{\text{in}}_{\text{LM}}(\Theta)$ is better correlated with the performance of the downstream task we consider several BERT language models pre-trained on random combinations of datasets from different domains mentioned in Section 4. These models are selected such that *for all the models $L_{LM}(\Theta)$ is roughly the same.* For each model Θ, we estimate $L^{\text{in}}_{\text{LM}}(\Theta)$ and contrast it with the accuracy/f1 score of the finetuned model on the task data.

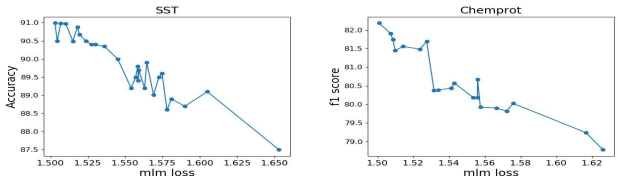


Figure 2: MLM loss of pre-trained BERT on the *task data* vs f1 score of corresponding finetuned BERT. Different points correspond to different BERT models, pre-trained on random combination of different datasets. MLM loss on the general domain corpus for all the pre-trained BERT models considered here is roughly the same.

Figure 2 provides the plots corresponding to this experiment and shows clear evidence that if the language model performs better on the task domain data, then the performance (accuracy/f1 score) of the finetuned model improves. We conclude that in order to ensure success in the downstream task, we should aim to minimize $L^{\text{in}}_{\text{LM}}(\Theta)$. A first attempt would be to pre-train or finetune the language model on $\mathcal{D}_{\text{in}}$. However, training a language model such as ELMo, GPT, BERT or XLNet requires a large corpus with several GBs of text and the available domain specific corpus $\mathcal{D}_{\text{in}}$ is often just the task data which has few MBs of text. Training on such a small dataset would introduce high variance. We reduce this variance by taking training examples from the general domain corpus $\mathcal{D}$, but control the bias this incurs by considering only elements having high relevance to the domain $\mathcal{D}_{\text{in}}$. Formally, we optimize a weighted pre-training loss function

$$L^{\Lambda}_{\text{LM}}(\Theta) = \sum_{x \in D} \lambda(x, \mathcal{D}_{\text{in}}) \cdot \ell_{\text{LM}}(x; \Theta), \tag{1}$$

where $\lambda(x, \mathcal{D}_{\text{in}})$ are relevance weights of instance x for domain $\mathcal{D}_{\text{in}}$. $\lambda(x, \mathcal{D}_{\text{in}})$ is (close to) 1 if x is relevant to $\mathcal{D}_{\text{in}}$ and (close to) 0 otherwise. We compute these weights using an anomaly detection model fitted on $\mathcal{D}_{\text{in}}$.

3.1 Anomaly Detection to solve the Domain Membership Problem

Detecting whether an instance x is an in-domain instance is equivalent to solving the following problem: *Given task data $\mathcal{T}$ and a sentence s, determine if s is likely to come from the distribution generating $\mathcal{T}$ or if s is an anomaly.*

This view helps us make use of a wide variety of anomaly detection techniques developed in literature (Noble & Cook, 2003; Chandola et al., 2009; Chalapathy & Chawla, 2019). To make use of these techniques, we first need a good numeric representation (embedding) with domain discrimination property. We use pre-trained BERT to embed each sentence into a 768 dimensional vector. Once the data is embedded, we need to decide which among the many anomaly detection algorithms proposed in literature should be applied on the embeddings. To decide the anomaly detection method, we propose an evaluation method ranking the techniques based on their discriminative properties.

Ranking anomaly detection algorithms: The idea is to treat the anomaly score as the prediction of a classifier distinguishing between in-domain and out-of-domain data. By doing so, we can consider classification metrics such as the f1_score as the score used to rank the anomaly detection algorithm. To do this, we split the in-domain data (the task data) into $\mathcal{D}^{\text{train}}_{\text{in}}$, $\mathcal{D}^{\text{test}}_{\text{in}}$ using a 90/10 split. We also create out-of-domain data $\mathcal{D}_{\text{out}}$ as a random subset of $\mathcal{D}$ of the same size as $\mathcal{D}^{\text{test}}_{\text{in}}$. We train an anomaly detection algorithm A with $\mathcal{D}^{\text{train}}_{\text{in}}$, and evaluate it's f1_score on the labeled test set composed of the union $\mathcal{D}^{\text{test}}_{\text{in}} \cup \mathcal{D}_{\text{out}}$, where the labels indicate which set the instance originated from. Note that anomaly detection algorithms considered do not require labeled samples for training. Thus, mixing data from $\mathcal{D}_{out}$ does not add much value.

Table 1 provides the results of this evaluation on six anomaly detection algorithms. Details of the tasks can be found in Section 4. We can see that Isolation Forest consistently performs well for most of the tasks. Local Outlier Factor performs almost equally well but is slower in prediction. Although it is possible to adaptively choose for every task the anomaly detection algorithm maximizing the f1_score, we chose to use a single algorithm, Isolation Forests, for the sake of having a simpler technique and generalizable results.

Task	RC	kNN	PCA	OCS	LOF	IF	Task	RC	kNN	PCA	OCS	LOF	IF
CHEMPROT	0.89	0.85	0.92	0.87	0.92	**0.96**	IMDB	0.88	0.96	0.87	0.81	**0.96**	0.94
ACL-ARC	0.77	0.88	0.90	0.89	**0.91**	0.88	SCIERC	0.78	0.84	0.86	0.76	0.88	**0.92**
HYPERPARTISAN	0.86	0.86	0.95	0.98	0.91	**0.98**	HELPFULNESS	0.82	0.89	0.83	0.76	0.83	**0.92**
RCT20K	0.85	0.88	0.82	0.76	0.87	**0.93**	IMDB	0.84	0.89	0.80	0.73	**0.92**	0.87

Table 1: Scores of different anomaly detection algorithms for different tasks. RC: Robust Covariance (Nguyen & Welsch, 2010), kNN: Nearest neighbor (Gu et al., 2019), PCA: Principal Component Analysis (Harrou et al., 2015), OCS: One Class SVM (Schölkopf et al., 2000), LOF: Local Outlier Factor (Breunig et al., 2000), IF: Isolation Forest (Liu et al., 2008)

Isolation Forest (Liu et al., 2008): For completeness, we provide a brief description of the Isolation Forest algorithm. Isolation Forest is an unsupervised decision tree ensemble method that identifies anomalies by isolating outliers of the data. It isolates anomalies in data points instead of profiling the normal points. Algorithm works by recursively partitioning the data using a random split between the minimum and maximum value of a random feature. It works due to the observation that outliers are less frequent than the normal points and lie further away from normal points in the feature space. Thus, in a random partitioning, anomalous points would require fewer splits on features resulting in shorter paths and distinguishing from the rest of the points. Anomaly score of a point x is defined as $s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$, where $E[h(x)]$ is the expected path length of x in various decision trees, $c(n) = 2H(n-1) - 2(n-1)/n$ is the average path length of unsuccessful search in a Binary Tree and $H(n-1)$ is the $n-1$-th harmonic number and n is the number of external nodes.

Now that we chose the anomaly detection technique, we move to discuss the effectiveness of the algorithm in (i) identifying the domain from the task data (ii) identifying the domain related data from the general domain corpus.

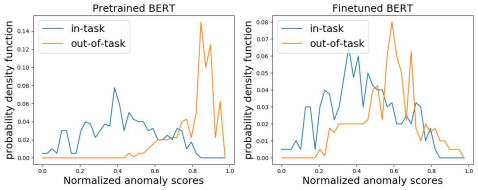


Figure 3: Sentence anomaly scores for SST with anomaly detection algorithm trained on embeddings from Left: pre-trained BERT, Right: finetuned BERT. In-task: sentences from the task data, out-of-task: sentences from general domain corpus.

Corpus	Input data	Filtered (Bio)	Filtered (CS)
News	7.1G (21.8%)	0.1G (2.0%)	0.0G (0.7%)
Finance	4.9G (15.0%)	0.0G (0.1%)	0.0G (0.3%)
CS	8.0G (24.5%)	1.0G (15.4%)	5.1G (78.0%)
Bio	12.6G (38.7%)	5.3G (82.4%)	1.4G (20.9%)

Figure 4: Filtering algorithm trained with Bio Abstracts and CS task data. We mix four corpora, filter out 80% of the data and retain the remaining 20% in both cases.

Figure 3 (left) shows that the anomaly detection algorithm is able to distinguish between the in-task-domain data and the out-of-task domain data. These experiments are done for the Sentiment Analysis task (SST) discussed in Section 4. Interestingly, we noticed in our experiments that a language model pre-trained on a diverse corpus is a better choice when compared to a model finetuned on the target domain. We conjecture that the reason is that a finetuned BERT is overly focused on the variations in the task data which are useful for task prediction and forgets information pertaining to different domains which is useful for domain discrimination. We exhibit this phenomenon more clearly in Figure 3 (right) where it is evident that the discriminating ability of the finetuned model is worse.

In order to assess the ability of our model to identify related text we perform the following experiment. First, we create a diverse corpus by taking the union of 4 datasets: News, Finance, CS abstracts and Biology abstracts. Figure 4, column 'Input data' contains their respective sizes. We then train two anomaly score based discriminators, one on CS task data and the other on Bio abstracts. For each

model we choose a threshold that would filter out 80% of the data, and observe the data eventually retained by it. The fraction of data retained from each corpus for each model is given in Figure 4, columns 'Filtered (Bio)' and 'Filtered (CS)'. We see that data from the News and Finance corpus is almost completely filtered as it is quite different than the text in abstracts of academic papers. We also see that a non-negligible percent of the filtered data for the Bio model comes from CS and vice versa. Since both corpora are abstracts of academic papers it makes sense that each corpus contains relevant data for the other. The details related to these corpora are given in Appendix B.

3.2 From Anomaly Detection Scores to Domain Adapted Pre-training

Once the anomaly detection object is trained, we use it to compute the relevance weights i.e. compute λ values defined in equation 1 . Let the sentences in the pre-training corpus be $s_1, \ldots, s_N$ with anomaly scores $\{A(s_1), \ldots, A(s_N)\}$. We explore two different strategies of λ value computation. First is when we normalize and transform the scores to compute continuous values and second when we use threshold and compute $0/1$ values.

Continuous λ values: We start by normalizing the anomaly scores to be mean zero and variance 1. Let $\mu = (\sum_{i=1}^{N} A(s_i))/N, \sigma = \sqrt{(\sum_{i=1}^{N}(A(s_i) - \mu)^2)/N}$. Then, for every $i \in \{1, \ldots, N\}$, normalized score is $\bar{A}(s_i) = (A(s_i) - \mu)/\sigma$. Using these normalized sentence anomaly scores, we compute the relevance weights as follows: $\lambda(s_i) = \frac{1}{1+e^{-C(\alpha - \bar{A}(s_i))}}$ where C and α are hyper-parameters. C controls the sensitivity of the weight in terms of anomaly score and α controls the fraction of target domain data present in the general domain corpus. $C \to \infty$ results in $0/1$ weights corresponding to discrete λ setting whereas $C = 0$ results in no task adaptation setting.

Discrete λ values: We sort the sentences as per anomaly scores, $A(s_{\sigma(1)}) \leq A(s_{\sigma(2)}) \leq \cdots \leq A(s_{\sigma(N)})$ and pick β fraction of the sentences with lowest anomaly scores, $\lambda(s_{\sigma(i)}) = 1$ for $i \in \{1, \ldots, \beta N\}$ and 0 otherwise. Even though this approach is less general than the continuous λ values case, it has an advantage of being model independent. We can filter out text, save it and use it to train any language model in a black box fashion. It does not require any change in pre-training or finetuning procedure. However, to utilize this option we need to make a change. Instead of filtering out sentences we need to filter out segments containing several consecutive sentences.

To understand why, suppose we filter out sentence 1 and sentence 10 and none of the sentences in between. When we save the text and construct input instances from it for a language model, then an input instance may contain the end of sentence 1 and the start of sentence 10. This is problematic as sentence 1 and sentence 10 were not adjacent to each other in the original corpus and hence, language model does not apply to them. It distorts the training procedure resulting in worse language models. To resolve this issue, we group sentences into segments and classify the relevance of each segment. Formally, let γ be a hyper-parameter and for all $j \in 1, \ldots, \lfloor N/\gamma \rfloor$ let the segment score be $y_j = \sum_{i=(j-1)*\gamma+1}^{j*\gamma} \frac{A(s_i)}{\gamma}$. We sort the segments according to their anomaly scores, $y_{\sigma'(1)} \leq \cdots \leq y_{\sigma'(N/\gamma)}$ and select the β fraction with lowest anomaly scores; save the sentences corresponding to these segments. To completely avoid the issue, we may set segment length very large. However, this is not feasible as the diverse nature of pre-training corpus makes sure that large enough segments rarely belong to a specific domain, meaning that the extracted data will no longer represent our target domain. We experimented with a handful of options for the segment length, and found the results to be stable when choosing segments of 15 sentences.

Continued pre-training instead of pre-training from scratch: Once we have computed the relevance weights $\lambda(s_i)$, we do not start pre-training the language model from scratch as this is not feasible for each new task/domain. Instead, we start with a language model pre-trained on the general domain corpus and perform additional pre-training for relatively fewer steps with the weighted loss function. In our case, we start with a BERT language model pre-trained for one million steps and continued pre-training with updated loss function for either $50,000$ or $100,000$ steps.

4 Experiments

We use datasets listed in Table 2 along with a general domain corpus consisting of 8GB of text from Wikipedia articles. We use $\text{BERT}_{\text{BASE}}$ model provided in the GluonNLP library for all our

Task	Train	Dev	Test	C	Task	Train	Dev	Test	C
HYPERPARTISAN	516	64	65	2	IMDB	20000	5000	25000	2
ACL-ARC	1688	114	139	6	SST	67349	872	1821	2
SCIERC	3219	455	974	7	AGNEWS	115000	5000	7600	4
CHEMPROT	4169	2427	3469	13	HELPFULNESS	115251	5000	25000	2
					RCT20K	180040	30212	30135	5

Table 2: Specification of task datasets. C refers to the number of classes. CHEMPROT (Kringelum et al., 2016) and RCT20K (Dernoncourt & Lee, 2017) are from biomedical domain. HYPERPARTISAN (Kiesel et al., 2019) and AGNEWS (Zhang et al., 2015) are from news domain. HELPFULNESS (McAuley et al., 2015) and IMDB (Maas et al., 2011) are from reviews domain. ACL-ARC (Jurgens et al., 2018) and SCIERC (Luan et al., 2018) are from CS domain. SST (Socher et al., 2013) is a general domain sentiment analysis task.

experiments. It has 12 layers, 768 hidden dimensions per token, 12 attention heads and a total of 110 million parameters. It is pre-trained with a sum of two objectives. First is the masked language model objective where model learns to predict masked tokens. Second is the next sentence prediction objective where sentence learns to predict if sentence B follows sentence A or not. We use learning rate of 0.0001, batch size 256 and warm-up ratio 0.01. For finetuning, we pass the final layer [CLS] token embedding through a task specific feed-forward layer for prediction. We use learning rate 3e-5, batch size 8, warm-up ratio 0.1 and finetune the network for five epochs. In all the experiments, we start with a BERT pre-trained for one million steps and continue pre-training for additional $50,000$ steps in case of discrete λ, and $100,000$ steps in case of continuous λ. Also, as mentioned in Section 3.2, we filter out segments instead of sentences and save them. We set the segment length to be 15 sentences and filter out 20% of the data. Pseudo-code of the end-to-end algorithm can be found in Appendix B.

4.1 BASELINE METHODS

For each baseline data selection method, we start with a BERT pre-trained on general domain corpus for one million steps as in case of TADPOLE. Then, we continue pre-training the baseline method for the same number of steps as in case of our method. In case of baseline methods which filter general domain corpus, we filter the same fraction of text as in case of our method.

General: *Continued pre-training on general domain corpus*. We know that in general longer pre-training leads to a better model. To estimate the impact of extra pre-training, we consider a baseline where we continue pre-training on the general domain corpus.

Random: *Continued pre-training on random subset of general domain corpus.*

Task (Gururangan et al., 2020): *Continued pre-training on task data*. We continue pre-training on the task data. Since task data is small, we can not pre-train on the task data for as many steps as in other cases. Instead we do 100 epochs, save the model after every 10 epoch and pick the best one.

LM (Moore & Lewis, 2010): *Continued pre-training on text filtered via language models trained on task data.* We train two language models, one on the task data and another on a subset of general domain corpus (same size as the task data). We select sentences with lowest scores given by the function $f(s) = H_I(s) - H_O(s)$, where $H_I(s)$ and $H_O(s)$ are the cross-entropy between the n-gram distribution and the language model distribution. More formally, cross entropy of a string s with empirical n-gram distribution p given a language model q_I is $H_I(s) = -\sum_x p(x) \log q_I(x)$.

Distance (Wang et al., 2017a): *Continued pre-training on data filtered via Euclidean distance scoring function.* For each sentence f, we consider BERT embedding v_f and compute vector centers $C_{F_{\text{in}}}$ and $C_{F_{\text{out}}}$ of the task data F_{in} and a random subset of general domain corpus F_{out}; $C_{F_{\text{in}}} = \frac{\sum_{f \in F_{\text{in}}} v_f}{|F_{\text{in}}|}, C_{F_{\text{out}}} = \frac{\sum_{f \in F_{\text{out}}} v_f}{|F_{\text{out}}|}$. We score a sentence f as per the scoring function:$\delta_f = d(v_f, C_{F_{\text{in}}}) - d(v_f, C_{F_{\text{out}}})$. We pick the text with lowest scores.

4.2 RESULTS

Table 3 shows the effectiveness of TADPOLE, automatically adapting pre-training to the task domain. Continuing pre-training on the unfiltered corpus (General or Random subset) yields an average gain

Task	Base	General	Random	Task	LM	Distance	TADPOLE
Small datasets: # training examples $<$ 5K							
HPRPARTISAN	$70.57_{3.04}$	$70.97_{2.03}$	$71.04_{2.32}$	$70.88_{2.63}$	$71.47_{2.56}$	$72.16_{2.14}$	$\mathbf{73.58}_{2.39}$
ACL-ARC	$72.31_{4.7}$	$72.38_{3.93}$	$72.42_{3.71}$	$72.46_{3.48}$	$72.40_{1.85}$	$72.47_{2.64}$	$\mathbf{72.81}_{3.83}$
SCIERC	$82.84_{1.39}$	$82.85_{1.38}$	$82.81_{1.13}$	$83.18_{1.09}$	$82.99_{2.75}$	$83.40_{2.17}$	$\mathbf{85.85}_{0.95}$
CHEMPROT	$81.62_{0.74}$	$81.59_{0.67}$	$81.62_{0.71}$	$81.63_{0.82}$	$81.83_{0.74}$	$81.64_{0.76}$	$\mathbf{82.41}_{0.62}$
Average Gain	-	$0.11_{0.05}$	$0.15_{0.05}$	$0.20_{0.04}$	$0.34_{0.08}$	$0.34_{0.06}$	$\mathbf{1.82}_{0.3}$
Large datasets: # training examples $\geq$ 5K							
IMDB	$88.65_{0.24}$	$88.53_{0.27}$	$88.63_{0.26}$	$88.77_{0.39}$	$88.67_{0.44}$	$88.69_{0.47}$	$\mathbf{89.29}_{0.22}$
SST	$92.02_{0.29}$	$92.21_{0.31}$	$92.14_{0.24}$	$92.21_{0.24}$	$92.25_{0.4}$	$92.15_{0.35}$	$\mathbf{92.42}_{0.32}$
AGNEWS	$93.99_{0.13}$	$94.06_{0.19}$	$\mathbf{94.09}_{0.11}$	$94.04_{0.08}$	$94.03_{0.13}$	$94.04_{0.11}$	$94.03_{0.16}$
HELPFULNESS	$69.30_{0.60}$	$69.39_{0.78}$	$69.34_{0.58}$	$69.41_{0.50}$	$69.58_{0.59}$	$69.42_{0.69}$	$\mathbf{69.70}_{0.92}$
RCT20K	$87.52_{0.16}$	$87.57_{0.16}$	$87.54_{0.17}$	$87.60_{0.18}$	$\mathbf{87.85}_{0.23}$	$87.62_{0.24}$	$87.82_{0.13}$
Average Gain	-	$0.02_{0.02}$	$0.04_{0.01}$	$0.11_{0.01}$	$0.18_{0.03}$	$0.09_{0.01}$	$\mathbf{0.36}_{0.04}$
All datasets							
Average Gain	-	$0.08_{0.05}$	$0.09_{0.05}$	$0.15_{0.04}$	$0.25_{0.08}$	$0.32_{0.17}$	$\mathbf{1.01}_{0.36}$

Table 3: Performance of TADPOLE and five Baseline methods. Base corresponds to the pre-trained model on general domain corpus with no further pre-training. Baseline methods are mentioned in previous subsection. TADPOLE corresponds to our method with discrete relevance weights. Keeping in line with the previous works, we use the following metrics: accuracy for SST, micro f1 score for CHEMPROT and RCT20K, macro f1 score for ACL-ARC, SCIERC, HELPFULNESS, HPRPARTISAN, IMDB, and AGNEWS. Each model is finetuned eight times with different seeds and the mean value is reported. Subscript correspond to the standard deviation in the finetuned model performance. Average gain corresponds to the average improvement over Base for each of the baseline methods and TADPOLE. Subscript in Average Gain corresponds to the standard deviation in the estimate of the average gain.

less than 0.1%. Adapting pre-training by training on the task data only yields an average gain of 0.15%. Applying popular data selection methods known for domain adaptation including Language Model based relevance score or Distance based relevance score yields a maximum gain of 0.32%. TADPOLE beats all these methods and achieve an average gain of 1.01%. For four tasks with small number of labeled examples (less than 5k), we get a much higher gain of 1.82%. This shows the efficacy of domain adapted finetuning if the task dataset is small. For eight tasks from four domains (all except SST), TADPOLE achieves an average gain of 1.07% whereas the best baseline method achieves an average gain of 0.34%. Models pre-trained on each of the four domain specific corpus can achieve a higher gain (3.37%) over the base model. However, unlike these models, our method has the advantage that it does not require access to any large domain specific corpus. Instead we only need a small task dataset available for finetuning. So, it is applicable to any new task from any new domain. We can also observe in Table 3, that TADPOLE beats all the baseline methods in seven of the nine cases. For RCT20K, it achieves a performance gain (0.3%) on par with the best baseline method (0.33%). For AGNews, gain is insignificant for all the methods as well as models trained on domain specific corpus. We observe this correlation for other tasks as well. Performance boost is higher if the corresponding boost via additional pre-training on large domain specific corpus is higher. Results for this comparison can be found in Appendix E. In Table 3, results are presented for the discrete relevant weight case as they are better when the number of steps available to continue pre-training are small. Results for continuous weights case can be found in Appendix D.

5 CONCLUSION

Domain shift in finetuning from Pre-training can significantly impact the performance of deep learning models. We address this issue in the most reasonable setting when we only have access to the labeled task data for finetuning. We adapt data selection methods from Domain Adaptation to adapt pre-training for the downstream task. The existing methods either require sufficiently large task data, or are based on adhoc techniques that do not generalize well across tasks. Our major contribution is providing a new data selection technique that performs well even with very little task data, and generalizes well across tasks.

REFERENCES

Steven Abney. *Semisupervised learning for computational linguistics*. CRC Press, 2007.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. Construction of the literature graph in semantic scholar. *arXiv preprint arXiv:1805.02262*, 2018.

Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 355–362, 2011.

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

Eyal Ben-David, Carmel Rabinovitz, and Roi Reichart. Perl: Pivot-based domain adaptation for pre-trained deep contextualized embedding models. *Transactions of the Association for Computational Linguistics*, 8:504–521, 2020.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128, 2006.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

Xia Cui and Danushka Bollegala. Self-adaptation for unsupervised domain adaptation. *Proceedings-Natural Language Processing in a Deep Learning World*, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Franck Dernoncourt and Ji Young Lee. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 678–683, 2013.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Xiaoyi Gu, Leman Akoglu, and Alessandro Rinaldo. Statistical analysis of nearest neighbor methods for anomaly detection. In *Advances in Neural Information Processing Systems*, pp. 10923–10933, 2019.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Fouzi Harrou, Farid Kadri, Sondes Chaabane, Christian Tahon, and Ying Sun. Improved principal component analysis for anomaly detection: Application to an emergency department. *Computers & Industrial Engineering*, 88:63–77, 2015.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406, 2018.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 829–839, 2019.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016, 2016.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422. IEEE, 2008.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602*, 2018.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

Robert C. Moore and William Lewis. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pp. 220–224, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P10-2041`.

Tri-Dzung Nguyen and Roy E Welsch. Outlier detection and robust covariance estimation using mathematical programming. *Advances in data analysis and classification*, 4(4):301–334, 2010.

Caleb C Noble and Diane J Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, 2003.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pp. 751–760, 2010.

Nanyun Peng and Mark Dredze. Multi-task domain adaptation for sequence tagging. *arXiv preprint arXiv:1608.02689*, 2016.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Jason Phang, Thibault Févry, and Samuel R Bowman. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

Jason Phang, Phu Mon Htut, Yada Pruksachatkun, Haokun Liu, Clara Vania, Katharina Kann, Iacer Calixto, and Samuel R Bowman. English intermediate-task training improves zero-shot cross-lingual transfer too. *arXiv preprint arXiv:2005.13013*, 2020.

Barbara Plank and Gertjan Van Noord. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1566–1576, 2011.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Robert Remus. Domain adaptation using domain similarity-and domain complexity-based instance selection for cross-domain sentiment analysis. In *2012 IEEE 12th international conference on data mining workshops*, pp. 717–723. IEEE, 2012.

Sebastian Ruder and Barbara Plank. Learning to select data for transfer learning with bayesian optimization. *arXiv preprint arXiv:1707.05246*, 2017.

Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pp. 582–588, 2000.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Vincent Van Asch and Walter Daelemans. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pp. 31–36, 2010.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. Dynamic data selection for neural machine translation. *arXiv preprint arXiv:1708.00712*, 2017.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 560–566, 2017a.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1482–1488, 2017b.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5753–5763, 2019.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pp. 9054–9065, 2019.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.

APPENDIX

A RELATED WORK

Domain Adaptation: A typical set up for Domain Adaptation involves access to labeled data in source domain, very limited or no labeled data in the target domain and unlabeled data in both source and target domains. This is somewhat different than the setup for our paper where we have access to labeled data with no additional unlabeled data in the task domain and our objective is optimize performance for the same domain. Nevertheless, several techniques of Domain Adaptation have similarities or core components useful for our setup. There are two sets of approaches addressing Domain Adaptation problem: model-centric and data-centric. Model-centric approaches redesign parts of the model: the feature space, the loss function or regularization and the structure of the model (Blitzer et al., 2006; Pan et al., 2010; Ganin et al., 2016). A recent such approach, appropriate for our setting is called Pivot-based Domain Adaptation; it has recently been applied to Task Adaptive Pre-training when there is additional unlabeled task data available Ben-David et al. (2020). In a nutshell, the idea is to distinguish between pivot and non-pivot features, where pivot features behave similarly in both domains. Then, by converting the non-pivot to pivot features, one can make use of a model trained on the source data. This approach does not work well when the target data is small since the mapping of non-pivot to pivot features cannot be trained with a limited size dataset. Since our technique is data-centric and applies to the regime of a small target corpus, we do not further analyze this or any other model-centric approach.

Data-centric approaches for domain adaptation include pseudo-labeling, using auxiliary tasks and data selection. Pseudo-labeling apply a trained classifier to predict labels on unlabeled instances which are then treated as 'pseudo' gold labels for further training (Abney, 2007; Cui & Bollegala, 2019). Auxiliary-task domain adaptation use labeled data from auxiliary tasks via multi-task learning (Peng & Dredze, 2016) or intermediate-task transfer (Phang et al., 2018; 2020). The methods most relevant to us are those of data selection and are discussed above in detail.

B DATASETS IN ACCURACY ESTIMATION OF ANOMALY SCORE BASED DATA FILTRATION

CS task data: To train anomaly score discriminator for CS data, we use the tasks data from ACL-ARC and SCIERC. Details of these datasets are mentioned in Section 4.

CS and Bio Abstracts: Semantic Scholar corpus (Ammar et al., 2018) contains datasets from a variety of domain. We filter out text based on the domain field and only keep the abstracts from CS and bio domain.

News: We use REALNEWS (Zellers et al., 2019) corpus containing news articles from 500 news domains indexed by Google News. It is obtained by scraping dumps from Common Crawl.

Finance: We use the TRC2-financial dataset. This a subset of Reuters TRC24 corpus containing news articles published between 2008 and 2010. It can be obtained by applying here: `https://trec.nist.gov/data/reuters/reuters.html`

C PSEUDO CODE

Algorithm 1 shows the pseudo code for the case of continuous relevance weights. Discrete relevance weight setting is same as $C \to \infty$. As discussed in 3.2, in case of discrete relevance weights, we filter out segments containing several consecutive sentences. We experimented with several options for the segment length and found the stable segment length to be 15 sentences. Here, a sentence is a consecutive piece of text such that when applied through the BERT tokenizer, it results in 256 sentences.

Algorithm 1 Task Adaptive Pre-training

Input: Pre-trained model B, Pre-training instances $x_1, \ldots, x_N$, task data $\mathcal{T}$, (C, α), #steps
Stage 1: Instance weight computation
Let the sentences of the task be $s_1, \ldots, s_t$ with sentence embeddings $P = \{\text{Embed}(s_1), \ldots, \text{Embed}(s_t)\}$.
Let a random subset of pre-training instances (sentences of these instances) be $s'_1, \ldots, s'_{t/10}$ with BERT based sentence embeddings $N = \{\text{Embed}(s'_1), \ldots, \text{Embed}(s'_{t/10})\}$
Train an anomaly detection object, IF = IsolationForest($P \cup N$)
For $i \in [N]$, let $S(x_i) = \text{IF.score}(\text{Embed}(x_i))$
Let $\mu = \frac{1}{N}\sum_{i=1}^{N} S(x_i)$ and $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S(x_i) - \mu)^2}$.
For every $i \in [N]$, $\bar{S}(x_i) = \frac{S(x_i)-\mu}{\sigma}$.
For every $i \in [N]$, $\lambda(x_i) = \frac{1}{1+e^{-C(\alpha - \bar{S}(x_i))}}$
Stage 2: Adaptation of pre-training to target domain
Continue training language model B for #steps on instances $x_1, \ldots, x_N$ with instance weights $\lambda(x_1), \ldots, \lambda(x_N)$.
Finetune resulting model on the labeled task data $\mathcal{T}$

Task	Base	Discrete	Continuous-1	Continuous-3
CHEMPROT	$81.62_{0.74}$	$\mathbf{82.41}_{0.62}$	$81.74_{0.81}$	$81.64_{0.83}$
RCT20K	$87.52_{0.16}$	$\mathbf{87.82}_{0.13}$	$87.49_{0.28}$	$87.56_{0.22}$
HPRPARTISAN	$70.57_{3.04}$	$\mathbf{73.58}_{2.39}$	$70.94_{1.98}$	$71.29_{2.95}$
AGNEWS	$93.99_{0.13}$	$\mathbf{94.03}_{0.16}$	$94.01_{0.14}$	$94.01_{0.15}$
HELPFULNESS	$69.30_{0.60}$	$\mathbf{69.70}_{0.92}$	$69.35_{0.5}$	$69.37_{0.44}$
IMDB	$88.65_{0.24}$	$\mathbf{89.29}_{0.22}$	$88.63_{0.51}$	$88.71_{0.46}$
ACL-ARC	$72.31_{4.7}$	$\mathbf{72.81}_{3.83}$	$72.26_{2.33}$	$72.36_{2.12}$
SCIERC	$82.84_{1.39}$	$\mathbf{85.85}_{0.95}$	$83.14_{1.96}$	$83.13_{2.65}$
SST	$92.02_{0.29}$	$\mathbf{92.42}_{0.32}$	$92.11_{0.32}$	$92.13_{0.37}$

Table 4: Comparison of discrete vs continuous relevance weight setting. Base corresponds to the pre-trained model on general domain corpus with no further pre-training. Discrete refers to TADPOLE with discrete relevance weights/filtered out text and pre-trained additionally for 50000 steps. Continuous-x refers to TADPOLE with continuous relevance weights and pre-trained additionally for $x * 100,000$ more steps. Metrics used for different tasks: accuracy for SST, micro f1 score for CHEMPROT and RCT20K, macro f1 score for ACL-ARC, SCIERC, HELPFULNESS, HPRPARTISAN, IMDB, and AGNEWS. Each model is finetuned eight times with different seeds and the mean value is reported. Subscript correspond to the standard deviation in the finetuned model performance.

D CONTINUOUS RELEVANCE WEIGHTS

We see in Table 4 that a model additionally pre-trained for 50,000 with discrete λ values consistently over performs the continuous case even when we train with continuous relevance weights for far higher number of steps. This is because of the fact that many of those steps yield virtually no training at all. For instance, suppose the relevance weights are uniformly distributed between 0 and 1; $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$. Then, in discrete case we pick the top two sentences and thus two steps are sufficient to train on these most relevant sentences (assume batch size is 1). However, in continuous case, we need to train the model for ten steps to train on these top two relevant sentences. Thus, we need many more steps to achieve and beat the performance achieved in the Discrete case. An open question is to combine the two settings so as to benefit from the generality of Continuous case and efficiency of the discrete case.

E PERFORMANCE BOOST WITH DOMAIN-SPECIFIC CORPUS VS TADPOLE

We compare the performance boost we achieved due to TADPOLE with the performance boost we achieve if we have access to large pre-training corpus. In Table 5, we list the gain in performance in both cases over eight tasks from four domains. We see that the performance boost is higher with

Task	TADPOLE	Domain Corpus
CHEMPROT	0.79	2.3
RCT20K	0.3	0.4
HYPERPARTISAN	3.01	1.6
AGNEWS	0.04	0.0
HELPFULNESS	0.4	1.4
IMDB	0.64	5.4
ACL-ARC	0.5	3.5
SCIERC	3.01	12.4

Table 5: Performance boost via TADPOLE vs pre-training on domain specific corpus. TADPOLE corresponds to our method with discrete relevance weights/filtered out text and pre-trained additionally for 50000 steps. Domain Corpus refers to the model trained in Gururangan et al. (2020) over the domain same as the downstream task. Metrics used for different tasks: accuracy for SST, micro f1 score for CHEMPROT and RCT20K, macro f1 score for ACL-ARC, SCIERC, HELPFULNESS, HPRPARTISAN, IMDB, and AGNEWS. Each model is finetuned eight times with different seeds. We report the difference in the mean value of performance between the model with additional pre-training and base model with no additional pre-training.

TADPOLE if the corresponding boost is higher with domain specific corpus. Thus if there is a large domain shift between the general domain corpus and the task data, as can be measured by the performance boost via large pre-training corpus, then TADPOLE is able to achieve large performance boost via Task Adaptation. Scale of numbers in the two columns are not directly comparable due to the following two reasons. First is that additional pre-training done is Gururangan et al. (2020) is for almost as many steps as the number of steps required to pre-train a network from scratch. However, in our case additional pre-training is done for only 5% of the number of steps required to pre-train a network from scratch. Second reason is that the model used in (Gururangan et al., 2020) is different, ROBERTA. Also, the general domain corpus is different and thus the domain shift is not exactly the same as in our case. The point however remains the same, which is that as the target domain is further away from the pre-training corpus, the benefits of TADPOLE increase.

CIGMO: LEARNING CATEGORICAL INVARIANT DEEP GENERATIVE MODELS FROM GROUPED DATA

Haruo Hosoya
ATR International, Hikaridai 2-2-2, Sourakugun Seikacho, Kyoto, Japan
hosoya@atr.jp

ABSTRACT

Images of general objects are often composed of three hidden factors: category (e.g., car or chair), shape (e.g., particular car form), and view (e.g., 3D orientation). While existing disentangling models can discover either a category or shape factor separately from a view factor, such models typically cannot capture the structure of general objects that the diversity of shapes is much larger across categories than within a category. Here, we propose a novel generative model called CIGMO, which can learn to represent the category, shape, and view factors at once only with weak supervision. Concretely, we develop mixture of disentangling deep generative models, where the mixture components correspond to object categories and each component model represents shape and view in a category-specific and mutually invariant manner. We devise a learning method based on variational autoencoders that does not explicitly use label information but uses only grouping information that links together different views of the same object. Using several datasets of 3D objects including ShapeNet, we demonstrate that our model often outperforms previous relevant models including state-of-the-art methods in invariant clustering and one-shot classification tasks, in a manner exposing the importance of categorical invariant representation.

1 INTRODUCTION

In everyday life, we see objects in a great variety. Categories of objects are numerous and their shape variations are tremendously rich; different views make an object look totally different (Figure 1(A)). Recent neuroscientific studies have revealed how the primate brain organizes representation of complex objects in the higher visual cortex (Freiwald & Tsao, 2010; Srihasam et al., 2014; Bao et al., 2020). According to these, it comprises multi-stream networks, each of which is specialized to a particular object category, encodes category-specific visual features, and undergoes multiple stages with increasing view invariance. These biological findings inspire us a new form of learning model that has multiple modules of category-specific invariant feature representations.

More specifically, our goal is, given an image dataset of general objects, to learn a generative model representing three latent factors: (1) category (e.g., cars, chairs), (2) shape (e.g., particular car or chair types), and (3) view (e.g., 3D orientation). A similar problem has been addressed by recent disentangling models that discover complex factors of input variations in a way invariant to each other (Tenenbaum & Freeman, 2000; Kingma et al., 2014; Chen et al., 2016; Higgins et al., 2016; Bouchacourt et al., 2018; Hosoya, 2019a). However, although these models can effectively infer a category or shape factor separately from a view factor, these typically cannot capture the structure in general object images that the diversity of shapes is much larger across categories than within a category.

In this study, we propose a novel model called CIGMO (Categorical Invariant Generative MOdel), which can learn to represent all the three factors (category, shape, and view) at once only with weak supervision. Our model has the form of mixture of deep generative models, where the mixture components correspond to categories and each component model gives a disentangled representation of shape and view for a particular category. We develop a learning algorithm based on variational autoencoders (VAE) method (Kingma & Welling, 2014) that does not use explicit labels, but uses

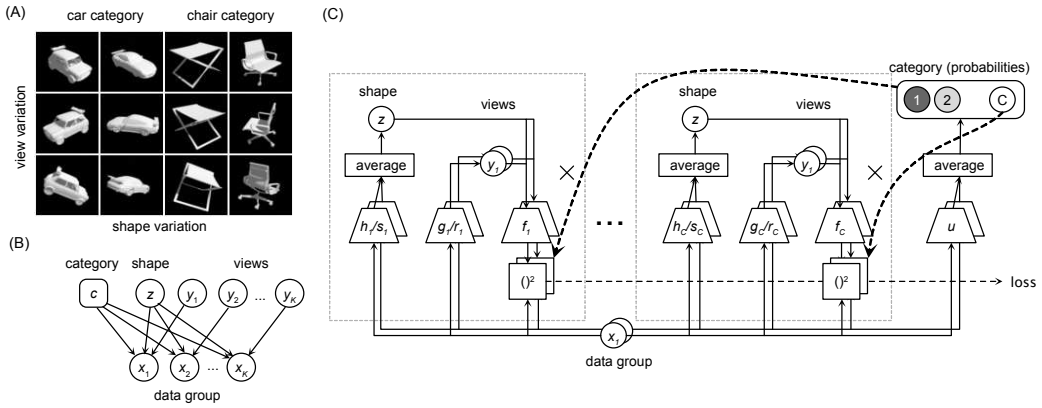


Figure 1: (A) Examples of general object images. These include two categories (car and chair) each with two shape variations. In addition, the object of each shape is shown in three different views. (B) The graphical model. Each instance x_k in a data group is generated from a category c, a shape z, and a view y_k. Round boxes are discrete variables and circles are continuous variables. (C) A diagrammatic outline of CIGMO learning algorithm. The entire workflow consists of C modules (light-gray boxes) of group-based VAE corresponding to C categories and a classifier network (rightmost). Given an input group of instances x_k (bottom), each module c estimates an instance-specific view y_k using encoders g_c/r_c and a group-common shape z by using encoders h_c/s_c followed by averaging. Then, new data instances generated by decoder f_c are compared with the original data for obtaining the reconstruction error (loss). This process is repeated for all modules. In parallel, the posterior probability for category c is computed by the classifier u followed by averaging and multiplied with the reconstruction error for the corresponding module. Other probabilistic mechanisms (e.g., priors) are omitted here for brevity.

only grouping information that links together different views of the same object (Bouchacourt et al., 2018; Chen et al., 2018; Hosoya, 2019a; Locatello et al., 2020).

Using two image datasets of 3D objects (one derived from ShapeNet (Chang et al., 2015)), we demonstrate that CIGMO can solve multiple unconventional visual tasks on objects that are unseen during training. These include invariant clustering, i.e., clustering of objects regardless of the view, one-shot classification, i.e., object recognition given one example per class, and other various feature manipulations using the disentangled representation. Quantitative comparison indicates that our model often outperforms many existing approaches including state-of-the-art methods.

Our key contributions are (1) proposal of the new approach of modeling data with category, shape, and view variables, (2) development of the new deep probabilistic generative model CIGMO, equipped with the VAE-based weakly supervised learning algorithm, (3) experiments of the model on two 3D object image datasets with quantitative comparisons that show performance advantages over several alternative models.

2 Related work

The present work is closely related to recently proposed disentangling models for discovering mutually invariant factors of variation in the input. In one direction, some models have used unsupervised learning with certain constraints on the latent variable, though these seem to be effective only in limited cases (Higgins et al., 2016; Chen et al., 2016). Thus, more practical approaches have made use of explicit labels, such as semi-supervised learning for a part of dataset (Kingma et al., 2014; Siddharth et al., 2017) or adversarial learning to promote disentanglement (Lample et al., 2017; Mathieu et al., 2016). However, labels are often expensive.

To find out a good compromise, weaker forms of supervision have been investigated. One such direction is group-based learning, which assumes inputs with the same shape to be grouped together (Bouchacourt et al., 2018; Chen et al., 2018; Hosoya, 2019a). However, these existing group-

based methods are fundamentally limited in that the factors that can be separated are two—a group-common factor (shape) and an instance-specific factor (view)—and there is no obvious way to extend it to more than two. Thus, our novelty here is to introduce a mixture model comprising multiple shape-view disentangling models, so that fitting the mixture model to a grouped dataset can give rise to the third factor, categories, as mixture components. In Section 4, we examine the empirical merits of this technique in several tasks. Note that grouping information can most naturally be found in temporal data (like videos) since the object identity is often stable over time, cf. classical temporal coherence principle (Földiák, 1991). Indeed, some weakly supervised disentangling approaches have explicitly used such temporal structure (Yang et al., 2015; Denton & Birodkar, 2017).

Some recent work has used deep nets for clustering of complex object images. The most typical approach is a simple combination of a deep generative model (e.g., VAE) and a conventional clustering method (e.g., Gaussian mixture), although such approach seems to be limited in capturing large object view variation (Jiang et al., 2017). A latest approach proposes a feedforward approach that takes pairs of image data, similarly to ours, and maximizes the mutual information between the categorical posterior probability distributions for such paired image; this has shown remarkable clustering performance on natural images under various view variation (Ji et al., 2019). In Section 4, we experimentally compare their method with ours. Note, however, that these methods are specialized to clustering and throw away all information other than the category.

3 CIGMO: Categorical Invariant Generative Model

3.1 Model

In our framework, we assume a grouped dataset

$$\mathbb{D} = \{(\boldsymbol{x}_1^{(n)}, \ldots, \boldsymbol{x}_K^{(n)}) \mid \boldsymbol{x}_k^{(n)} \in \mathbb{R}^D, n = 1, \ldots, N\}$$

where each data point is a group (tuple) of K data instances (e.g., images); we assume independence between groups but not instances within a group. For a data group $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K)$, we consider three types of hidden variables: category $c \in \{1, \ldots, C\}$, shape $\boldsymbol{z} \in \mathbb{R}^M$, and views $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K \in \mathbb{R}^L$ (omitting the superscript (n) for brevity), where the category and shape are common for the group while the views are specific to each instance. We consider the following generative model (Figure 1(B)):

$$\begin{aligned} p(c) &= \pi_c \\ p(\boldsymbol{z}) &= \mathcal{N}_M(0, \boldsymbol{I}) \\ p(\boldsymbol{y}_k) &= \mathcal{N}_L(0, \boldsymbol{I}) \\ p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{z}, c) &= \mathcal{N}_D(f_c(\boldsymbol{y}_k, \boldsymbol{z}), \boldsymbol{I}) \end{aligned}$$

for $c = 1, \ldots, C$ and $k = 1, \ldots, K$. Here, f_c is a decoder deep net defined for each category c and π_c is a category prior with $\sum_{c=1}^{C} \pi_c = 1$. In the generative process, first, the category c is drawn from the categorical distribution $(\pi_1, \ldots, \pi_C)$, while the shape $\boldsymbol{z}$ and views $\boldsymbol{y}_k$ are drawn from standard Gaussian priors. Then, each data instance $\boldsymbol{x}_k$ is generated by the decoder deep net f_c for category c applied to the group-common shape $\boldsymbol{z}$ and the instance-specific view $\boldsymbol{y}_k$ (added with Gaussian noise of unit variance). In other words, the decoder f_c generates different data instances in a group from the same shape and different views. Having defined a mixture of deep generative models as above, we expect that, after fitting it to a view-grouped object image dataset, object categories will arise as mixture components and category-specific shapes and views will be represented in each component model.

3.2 Learning

We construct a learning algorithm based on the VAE approach (Kingma & Welling, 2014). As the most important step, we specify inference models to encode approximate posterior distributions. First, we estimate the posterior probability for category c as follows:

$$q(c|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K) = \frac{1}{K} \sum_{k=1}^{K} u^{(c)}(\boldsymbol{x}_k) \tag{1}$$

Here, u is a classifier deep net that takes an individual instance $\boldsymbol{x}$ and outputs a probability distribution over the categories ($\sum_{c=1}^{C} u^{(c)}(\boldsymbol{x}) = 1$). We then take the average over the instance-specific probability distributions and use it as the group-common distribution. This is a simple approach to make the instance-specific distributions converge to equal values, i.e., $u(\boldsymbol{x}_1) \approx u(\boldsymbol{x}_2)$. (We can use other approaches than average such as normalized product or softmax over averaged logits. However, we could not find a difference in performance.) Note that, although the use of a deep net for estimating a categorical distribution is similar to Kingma et al. (2014), the technique of computing the group-common category distribution (equation 1) specifically arises from the combination of mixture and group-based learning and is therefore somewhat new here.

Then, given the estimated category c, we infer each instance-specific view $\boldsymbol{y}_k$ from the input $\boldsymbol{x}_k$ as follows:

$$q(\boldsymbol{y}_k|\boldsymbol{x}_k, c) = \mathcal{N}_L\left(g_c(\boldsymbol{x}_k), \mathrm{diag}(r_c(\boldsymbol{x}_k))\right) \tag{2}$$

where g_c and r_c are encoder deep nets that are defined for each category c to specify the mean and variance, respectively. To estimate shape $\boldsymbol{z}$, we compute the following:

$$q(\boldsymbol{z}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K, c) = \mathcal{N}_M\left(\frac{1}{K}\sum_{k=1}^{K} h_c(\boldsymbol{x}_k), \frac{1}{K}\sum_{k=1}^{K} \mathrm{diag}(s_c(\boldsymbol{x}_k))\right) \tag{3}$$

Here, again, encoder deep nets h_c and s_c are defined for each category c. These compute the mean and variance, respectively, of the posterior distribution for the individual shape for each instance $\boldsymbol{x}_k$. Then, the group-common shape $\boldsymbol{z}$ is obtained as the average over all the individual shapes (Hosoya, 2019a). In this way, the instance-specific shape representations are expected to converge to an equal value in the course of training, i.e., $h_c(\boldsymbol{x}_1) \approx h_c(\boldsymbol{x}_2)$. Note that CIGMO is precisely equal to GVAE when $C = 1$.

For training, we define the following variational lower bound of the marginal log likelihood for a data point: $\mathcal{L}(\phi; \vec{\boldsymbol{x}}) = \mathcal{L}_{\mathrm{recon}} + \mathcal{L}_{\mathrm{KL}}$ with

$$\mathcal{L}_{\mathrm{recon}} = \mathbb{E}_{q(\vec{\boldsymbol{y}}, \boldsymbol{z}, c|\vec{\boldsymbol{x}})}\left[\sum_{k=1}^{K} \log p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{z}, c)\right]$$
$$\mathcal{L}_{\mathrm{KL}} = -D_{\mathrm{KL}}(q(\vec{\boldsymbol{y}}, \boldsymbol{z}, c|\vec{\boldsymbol{x}})\|p(\vec{\boldsymbol{y}}, \boldsymbol{z}, c))$$

where $\vec{\boldsymbol{x}}$ stands for $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K)$, etc., and ϕ is the set of all weight parameters in the classifier, encoder, and decoder deep nets. We compute the reconstruction term $\mathcal{L}_{\mathrm{recon}}$ as follows:

$$\mathcal{L}_{\mathrm{recon}} = \sum_{c=1}^{C} q(c|\vec{\boldsymbol{x}}) \mathbb{E}_{q(\vec{\boldsymbol{y}}, \boldsymbol{z}|\vec{\boldsymbol{x}}, c)}\left[\sum_{k=1}^{K} \log p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{z}, c)\right] \approx \sum_{c=1}^{C} q(c|\vec{\boldsymbol{x}}) \sum_{k=1}^{K} \log p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{z}, c)$$

where we approximate the expectation using one sample $\boldsymbol{z} \sim q(\boldsymbol{z}|\vec{\boldsymbol{x}}, c)$ and $\boldsymbol{y}_k \sim q(\boldsymbol{y}_k|\boldsymbol{x}_k, c)$ for each k, but directly use the probability value $q(c|\vec{\boldsymbol{x}})$ for c. The latter is crucial for making the loss function differentiable. The KL term $\mathcal{L}_{\mathrm{KL}}$ is computed as follows:

$$\mathcal{L}_{\mathrm{KL}} = -D_{\mathrm{KL}}(q(c|\vec{\boldsymbol{x}})\|p(c)) - \sum_{c=1}^{C} q(c|\vec{\boldsymbol{x}}) \left[\sum_{k=1}^{K} D_{\mathrm{KL}}(q(\boldsymbol{y}_k|\boldsymbol{x}_k, c)\|p(\boldsymbol{y}_k)) + D_{\mathrm{KL}}(q(\boldsymbol{z}|\vec{\boldsymbol{x}}, c)\|p(\boldsymbol{z}))\right]$$

Finally, our training procedure is to maximize the lower bound for the entire dataset with respect to the weight parameters: $\hat{\phi} = \arg\max_{\phi} \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(\phi; \boldsymbol{x}_1^{(n)}, \ldots, \boldsymbol{x}_K^{(n)})$. A diagrammatic outline of the algorithm is given in Figure 1(C).

Optionally, we sometimes incorporate a regularization to maximize mutual information between the category probability distributions within each group (Ji et al., 2019):

$$\mathcal{L}'(\phi; \vec{\boldsymbol{x}}) = \mathcal{L}_{\mathrm{recon}} + \mathcal{L}_{\mathrm{KL}} + \alpha \sum_{k<k'} I(u(\boldsymbol{x}_k), u(\boldsymbol{x}_{k'})) \tag{4}$$

4 EXPERIMENTS

We have applied the model described in Section 3 to two image datasets: ShapeNet (general objects) and MultiPie (natural faces). Below, we outline the experimental set-up and show the results.

4.1 SHAPENET

For the first set of experiment, we created a dataset of multi-viewed object images derived from 3D models in ShapeNet database (Chang et al., 2015). We selected 10, out of 55, pre-defined object classes with a relatively large number of object identities (car, chair, table, airplane, lamp, boat, box, display, truck, and vase); we avoided heavily overlapping classes with many visually similar objects, e.g., chair, sofa, and bench. We then rendered each object in 30 views in a single lighting condition. We split the training and test sets, which consisted of 21888 and 6210 object identities, respectively. We also created subset versions with 3 or 5 object classes. For training data, we formed groups of images of the same object in random 3 views ($K = 3$). We used object identity labels (not class labels) for grouping, but, after this step, we never used any label during the training. Supplementary Materials give more details on the dataset.

To train a CIGMO model, we used the following architecture. First, we set the number of categories in the model to the number of classes in the data ($C = 3, 5$, or 10). We set the shape dimension $M = 100$ and the view dimension $L = 3$. Here, using a very low view dimension was crucial since otherwise the view variable $\boldsymbol{y}_k$ would take over all the information in the input and the shape variable $\boldsymbol{z}$ would become degenerate (Hosoya, 2019a). The classifier deep net u consisted of three convolutional layers and two fully connected layers and ended with a Softmax layer. The shape and view encoder deep nets had a similar architecture, except that the last layer was linear for mean encoding (g_c and h_c) and ended with Softplus for variance encoding (r_c and s_c). The decoder deep nets f_c had an inverse architecture ending with Sigmoid. Since the model had so many deep nets, a large part of the networks was shared to save the memory space. For simplicity, we fixed the category prior $\pi_c = 1/C$. We turned off the optional regularization (equation 4) as we did not find it effective. Supplementary Materials give more details on the architecture. For optimization, we used Adam (Kingma & Ba, 2015) with mini-batches of size 100 and ran 20 epochs.

For comparison, we trained a number of related models. To investigate the effect of decoupled representation of category and shape, we trained GVAE models (Hosoya, 2019a), which can be obtained as a special case of CIGMO with a single category ($C = 1$). To examine the effect of disentangling of shape and view, we trained mixture of VAEs, again obtained as CIGMO with no grouping ($K = 1$; the shape and view variables are integrated). We also trained plain VAEs for basic comparison ($C = K = 1$). For a part of evaluation below, since GVAE and VAE themselves have no clustering capability, we performed k-means on the shape variable of GVAE or the latent variable of VAE. In addition, we incorporated two completely different methods: Multi-Level VAE (MLVAE) (Bouchacourt et al., 2018), another group-based generative model, and Invariant Information Clustering (IIC) (Ji et al., 2019), a method specialized to invariant clustering. We tested two versions of IIC, with and without regularization using 5-times overclustering (Ji et al., 2019). For each method, we trained 10 model instances from the scratch.

We evaluated the trained models using test data, which were ungrouped and contained objects of the same classes as training data but of different identities. The evaluation involved three tasks: (1) invariant clustering, (2) one-shot classification, and (3) feature manipulation.

4.1.1 INVARIANT CLUSTERING

In this task, we simply inferred the most probable category from a given image, $\hat{c} = \arg\max_c q(c|\boldsymbol{x})$. Figure 2(A) illustrates results from a CIGMO model with 3 categories, where each box shows random test images belonging to each estimated category. This demonstrates a very precise clustering of objects achieved by the model, which is quite remarkable given the large view variation and no category label used during training. Figure 2(B) shows analogous examples for a mixture of VAEs. The result shows a clear degrade of performance.

For comparison of the methods, we quantified the performance of invariant clustering in terms of classification accuracy. Here, we used the best category-to-class assignment computed by the Hungarian algorithm (Munkres, 1957). Table 1 summarizes the results. Generally, CIGMO outperformed the other methods in all cases with a large margin. More specifically, first, CIGMO always performed better than mixture of VAEs, showing the importance of shape-view disentangling. Second, CIGMO was also always better than GVAE (and MLVAE) plus k-means, also showing the importance of category-shape decoupling. In other words, if shapes of all categories were packed into a single latent variable, category information could not be clearly represented. These two points,

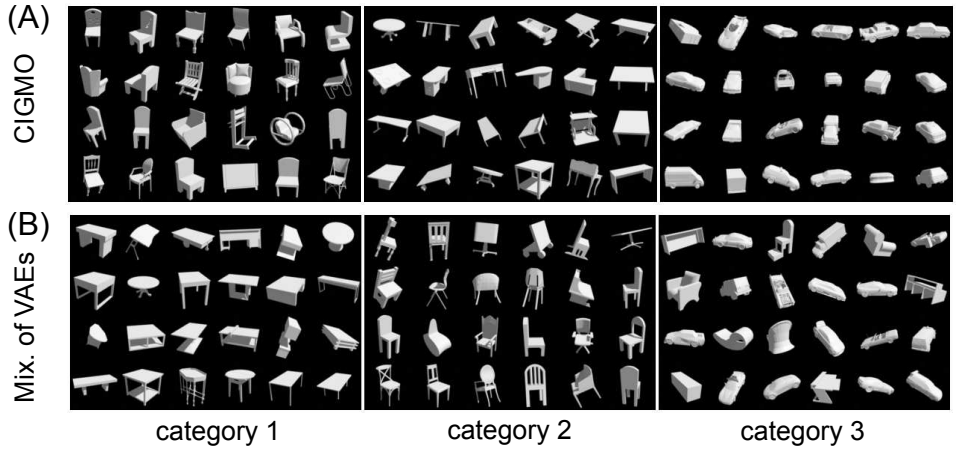


Figure 2: Examples of invariant clustering from (A) a CIGMO, (B) a mixture of VAEs, for ShapeNet, in the case of 3 categories. Random 24 images belonging to each estimated category are shown in a box. Note that the categories quite precisely correspond to the chair, table, car image classes in (A). Such correspondence is less clear in (B); in particular, cars are mixed with many other objects (category 3).

# of categories	3	5	10
chance level	33.33	20.00	10.00
IIC	85.25 ± 13.74	81.10 ± 7.33	60.84 ± 1.45
IIC (overcluster.)	79.86 ± 13.78	81.87 ± 4.57	59.73 ± 1.49
VAE*	66.41 ± 5.69	50.83 ± 3.85	37.07 ± 1.00
Mix. of VAEs	82.35 ± 5.66	65.73 ± 6.24	40.86 ± 3.58
MLVAE*	82.04 ± 7.78	70.68 ± 5.04	54.47 ± 1.92
GVAE*	73.20 ± 10.93	69.42 ± 3.47	52.55 ± 2.74
CIGMO	**94.83** ± 6.06	**89.36** ± 4.53	**68.53** ± 4.24

Table 1: Invariant clustering accuracy (%) for ShapeNet. The mean and SD over 10 model instances are shown. The method name with asterisk involves k-means clustering.

taken together, emphasize the categorical and invariant nature of our model. Third, CIGMO gave performance superior to IIC, surpassing the state-of-the-art method for this task. This was the case both with and without overclustering; in fact, we could not find a consistent improvement by overclustering in IIC, contrary to the claim by Ji et al. (2019).

4.1.2 ONE-SHOT CLASSIFICATION

In this task, we split test data into gallery and probe, where gallery holds one image for each object and probe the rest, and then identify the object of each probe image. Note that our purpose here is not to infer the class but the object identity, unlike invariant clustering. Note also that, since the test objects are disjoint from the training objects, both gallery and probe images contain only unseen objects for the model. We used this task here since its performance can serve as a criterion for evaluating disentangled representations (Hosoya, 2019a). The rationale is that, if shape code is perfectly invariant in view, then all images of the same object should be mapped to an identical point in the shape space.

Thus, we compared overall accuracy of one-shot classification for CIGMO and other models. For this, we performed a nearest-neighbor method according to cosine distance in the shape space. Here, the shape space was defined depending on the method. For GVAE (or MLVAE), the shape variable $\boldsymbol{z} = h(\boldsymbol{x})$ directly defined the shape space. For CIGMO, since the shape representation depended on the category, we first constructed a $C \times M$ matrix $\boldsymbol{Z}$ such that $\boldsymbol{Z}_{c,*} = h_c(\boldsymbol{x})$ for $c = \hat{c}$ and $\boldsymbol{Z}_{c,*} = 0$ otherwise, and then flattened the matrix to a vector. This gave us category-dependent

# of categories	3	5	10
chance level	0.03	0.02	0.02
VAE	1.81 ± 0.04	3.09 ± 0.05	2.97 ± 0.02
Mix. of VAEs	1.91 ± 0.06	3.30 ± 0.07	3.15 ± 0.06
MLVAE	20.56 ± 0.50	17.69 ± 0.28	15.68 ± 0.28
GVAE	21.44 ± 0.54	17.85 ± 0.28	15.93 ± 0.18
CIGMO	$\mathbf{23.95} \pm 0.52$	$\mathbf{21.66} \pm 0.79$	$\mathbf{19.49} \pm 0.71$

Table 2: One-shot classification accuracy (%) for ShapeNet. The mean and SD over 10 model instances are shown.

# of categories	3	5	10
MLVAE	53.59 ± 0.57	47.37 ± 0.85	42.99 ± 0.63
GVAE	55.17 ± 0.80	48.23 ± 0.58	43.85 ± 0.29
CIGMO	57.69 ± 0.94	51.00 ± 0.89	46.30 ± 0.93
MLVAE	0.24 ± 0.02	0.62 ± 0.04	0.53 ± 0.09
GVAE	0.23 ± 0.04	0.55 ± 0.05	0.48 ± 0.05
CIGMO	0.26 ± 0.04	0.65 ± 0.05	0.66 ± 0.08

Table 3: Quality of shape-view disentanglement for ShapeNet, measured as neural network classification accuracy (%) for object identity from the shape (top rows; higher is better) or view variable (bottom rows; lower is better); weighted average over categories. The mean and SD over 10 model instances are shown.

shape vectors that could be directly compared and, in particular, those of different categories would give cosine distance 1, the maximum value. For VAE or mixture of VAEs, we used a similar scheme except that we used the entire latent variable in place of shape variable.

Table 2 summarizes the results. Overall, CIGMO performed the best among the compared methods in all cases. In particular, it outperformed, by far, mixture of VAEs, showing the success of shape information disentangled from view. Further, CIGMO also performed significantly better than GVAE and MLVAE, which indicates that shapes can be represented more efficiently with category specialization than without. These results, again, reveal the advantage of the categorical and invariant representations in modeling general object images. (Note also that the scores were remarkably high even for up to 6210-way classification by one shot.)

4.1.3 FEATURE MANIPULATION

CIGMO provides various ways of manipulating latent feature representations and generating new images. These include (1) swapping, to generate an image from the view of one image and the shape of another, (2) interpolation, to generate an image from the shape and view that linearly interpolates those of two images, and (3) random generation, to generate an image from shape and view randomly drawn from Gaussian distributions. Analogous manipulations have commonly been used in previous studies (Mathieu et al., 2016; Bouchacourt et al., 2018; Hosoya, 2019a), but our cases are conditioned on a category. Supplementary Materials give examples of these feature manipulations; we can qualitatively confirm the represented disentaglement, e.g., reasonably clear alignment of views in rows and shapes in columns in swapping.

To quantify the quality of the category-wise shape-view disentanglement, we measured how much information the shape or view variable contained on object identity. For this purpose, we trained two-layer neural networks on either variable for classification (Mathieu et al., 2016; Bouchacourt et al., 2018). A better disentangled representation was expected to give a higher accuracy from the shape variable and a lower accuracy from the view variable. We performed this for each category using the belonging test images and took the average accuracy weighted by the number of those images. Table 3 summarizes the results. Overall, CIGMO gives fairly comparable quality to the existing disentangling methods.

Figure 3: Results from a CIGMO model trained for MultiPie dataset. (A) Invariant clustering. Random images belonging to each estimated category are shown in a box. (B) Swapping. For each category, we generate a matrix of images from two lists of sample images, where each generated image has the view of an image in the left list and the shape of an image in the top list.

4.2 MULTIPIE

For the second set of experiment, we used a dataset of multi-viewed face images derived from MultiPie dataset (Gross et al., 2010). We followed the same data preparation as Hosoya (2019a), except that we included all lighting conditions. We split the training and test sets consisting of disjoint 795 and 126 identities, and grouped together the training images with the same identity, hair/cloth style, and expression. For this dataset, there was no pre-defined class unlike ShapeNet; we arbitrarily set the number of categories to 3 in CIGMO models. In addition, we turned on the regularization using mutual information (equation 4) with $\alpha = 1000$. Other training conditions were identical as before.

Figure 3(A) illustrates the result of invariant clustering for a CIGMO model. By inspection, category 2 seemed to represent faces with long hair, while categories 1 and 3 both represent faces with short hair. Although the difference between categories 1 and 3 was more subtle, category 1 seemed to include more round faces while category 3 oval faces. Note that we could not verify these observations since the dataset lacked relevant labels. Other CIGMO model instances trained in the same way had 1 to 3 effective categories (the remaining categories were degenerate to which no input belonged) and seemed to use more or less similar categorization strategies; in particular, we could not find categorization by expression. Figure 3(B) illustrates the result of category-wise swapping. We can observe that the view and shape representations were well disentangled in each category. However, CIGMO gave overall slightly lower performance for one-shot classification accuracy and shape-view disentangling quality compared to GVAE or MLVAE (Supplementary Materials). This indicates a lesser importance of category-shape decoupling in this task for this dataset. This is understandable since, in a sense, all faces look alike and therefore, in what way faces are categorized, cross-category diversity would not be so large compared to within-category diversity.

5 CONCLUSION

In this paper, we have proposed CIGMO as a deep generative model to discover category, shape, and view factors from general object images. The model has the form of mixture of group-based VAEs and comes with a weakly supervised algorithm that requires no explicit label but only view-grouping

information. We showed that, for two image datasets, CIGMO can learn to represent categories in the mixture components and category-specific disentangled information of shape and view in each component model. We demonstrated that our model can outperform existing methods including state-of-the-art methods in invariant clustering and one-shot classification tasks, emphasizing the importance of categorical invariant representation. Future investigation will include improvement in image generation quality, category degeneracy, and scalability, and application to more realistic datasets. Lastly, CIGMO's biological relationship to the primate inferotemporal cortex would be interesting to pursue, as our present work was originally inspired so (Hosoya & Hyvärinen, 2017; Hosoya, 2019b)

Acknowledgments This work has been supported by the Commissioned Research of National Institute of Information and Communications Technology and Grants-in-Aid for Scientific Research.

REFERENCES

Pinglei Bao, Liang She, Mason McGill, and Doris Y. Tsao. A map of object space in primate inferotemporal cortex. *Nature*, 583(7814):103–108, jul 2020.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-Level Variational Autoencoder: Learning Disentangled Representations from Grouped Observations. In *AAAI Conf. Artif. Intell.*, jan 2018.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. 2015.

Mickaël Chen, Ludovic Denoyer, and Thierry Artières. Multi-View Data Generation Without View Supervision. In *Int. Conf. Learn. Represent.*, nov 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN - Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *Adv. Neural Inf. Process. Syst.*, jan 2016.

Emily Denton and Vighnesh Birodkar. Unsupervised Learning of Disentangled Representations from Video. *Adv. Neural Inf. Process. Syst.*, may 2017.

P Földiák. Learning invariance from transformation sequences. *Neural Comput.*, 3(2):194–200, jan 1991.

Winrich A Freiwald and Doris Y Tsao. Functional Compartmentalization and Viewpoint Generalization Within the Macaque Face-Processing System. *Science*, 330(6005):845–851, nov 2010.

Ralph Gross, Iain Matthews, Jeff Cohn, Takeo Kanade, and Simon Baker. Multi-PIE. *Proc. Int. Conf. Autom. Face Gesture Recognition. IEEE Int. Conf. Autom. Face Gesture Recognit.*, 28(5): 807–813, may 2010.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *Int. Conf. Learn. Represent.*, nov 2016.

Haruo Hosoya. Group-based learning of disentangled representations with generalizability for novel contents. *Int. Jt. Conf. Artif. Intell.*, jan 2019a.

Haruo Hosoya. A deep generative model explaining tuning properties of monkey face processing patches. *2019 Conf. Cogn. Comput. Neurosci.*, 2019b.

Haruo Hosoya and Aapo Hyvärinen. A mixture of sparse coding models explaining properties of face neurons related to holistic and parts-based processing. *PLoS Comput. Biol.*, 13(7):e1005667, jul 2017.

Xu Ji, Andrea Vedaldi, and Joao Henriques. Invariant information clustering for unsupervised image classification and segmentation. *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 9864–9873, 2019.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised generative approach to clustering. *Int. Jt. Conf. Artif. Intell.*, pp. 1965–1972, 2017.

D Kingma and J Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learn. Represent.*, jan 2015.

D P Kingma and M Welling. Auto-encoding variational bayes. In *Int. Conf. Learn. Represent.*, jan 2014.

D P Kingma, S Mohamed, and D J Rezende. Semi-supervised learning with deep generative models. *Adv. Neural Inf. Process. Syst.*, jan 2014.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader Networks:Manipulating Images by Sliding Attributes. *Adv. Neural Inf. Process. Syst.*, pp. 5967–5976, jan 2017.

Francesco Locatello, Ben Poole, Gunnar Raetsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-Supervised Disentanglement Without Compromises. In *Proc. 37th Int. Conf. Mach. Learn.*, volume 119, pp. 6348–6359, 2020.

Michaël Mathieu, Junbo Jake Zhao, Pablo Sprechmann, Aditya Ramesh, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. *Adv. Neural Inf. Process. Syst.*, jan 2016.

James Munkres. Algorithms for the Assignment and Transportation Problems. *J. Soc. Ind. Appl. Math.*, 5(1):32–38, mar 1957.

N Siddharth, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Frank Wood, Noah D Goodman, Pushmeet Kohli, and Philip H S Torr. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. *Adv. Neural Inf. Process. Syst.*, jan 2017.

Krishna Srihasam, Justin L. Vincent, and Margaret S. Livingstone. Novel domain formation reveals proto-architecture in inferotemporal cortex. *Nat. Neurosci.*, 17(12):1776–1783, 2014.

J B Tenenbaum and W T Freeman. Separating style and content with bilinear models. *Neural Comput.*, 12(6):1247–1283, jun 2000.

Jimei Yang, Scott Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised Disentangling with Recurrent Transformations for 3D View Synthesis. *Adv. Neural Inf. Process. Syst.*, jan 2015.

SUPPLEMENTARY MATERIALS

A DATASET DETAILS

Our object image dataset stemmed from a core subset of ShapeNet database of 3D object models (Chang et al., 2015) used in SHREC2016 challenge[1]. The subset contained 55 object classes and did not include material data. Out of the 55 classes, we selected 10 classes: car, chair, table, airplane, lamp, boat, box, display, truck, and vase. Our criterion here was to select classes with a large number of object identities but avoid including visually similar classes, e.g., chair, sofa, and bench. We rendered each object in 30 views consisting of 15 azimuths (equally dividing $360°$) and 2 elevations ($0°$ and $22.5°$ downward) in a single lighting condition; the images were gray-scale and had size 64×64 pixels. All the rendering used Blender software[2]. We divided the data into training and test following the split given in the original database.

[1] `https://shapenet.cs.stanford.edu/shrec16/`

[2] `https://www.blender.org`

B ARCHITECTURE DETAILS

In a CIGMO model, the classifier deep net u consisted of three convolutional layers each with 32, 64, and 128 filters (kernel 5×5; stride 2; padding 2), followed by two fully connected layers each with 500 intermediate units and C output units. These layers were each intervened with Batch Normalization and ReLU nonlinearity, except that the last layer ended with Softmax. The shape and view encoder deep nets had a similar architecture, except that the last layer was linear for encoding the mean (g_c and h_c) or ended with Softplus for encoding the variance (r_c and s_c). The decoder deep nets f_c had two fully connected layers (103 input units and 500 intermediate units) followed by three transposed convolutional layers each with 128, 64, and 32 filters (kernel 6×6; stride 2; padding 2). These layers were again intervened with Batch Normalization and ReLU nonlinearity, but the last layer was Sigmoid.

To save the memory space, the shape encoders shared the first four layers for all categories and for mean and variance. The view encoders shared the entire architecture for all categories, but with a separate last layer for mean or variance specification. The decoders shared all but the first layer for all categories.

In the quantitative comparison, we obtained a mixture of VAEs, a GVAE model, and a VAE model as a special case of CIGMO model. Namely, a mixture of VAEs was a CIGMO with no grouping ($K = 1$), a GVAE was a CIGMO with a single category ($C = 1$), and a VAE was a CIGMO with both constraints ($K = 1$ and $C = 1$). In the case of no grouping, since no structure could differentiate the shape and view dimensions, we combined these into a single latent variable of 103 dimensions.

C ADDITIONAL RESULTS FOR SHAPENET

We performed the following feature manipulation tasks introduced in Section 4.

Swapping For a category c and for images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ belonging to c, obtain $\boldsymbol{x}_{\text{swap}} = f_c(\boldsymbol{y}_1, \boldsymbol{z}_2)$ where $\boldsymbol{y}_1 = g_c(\boldsymbol{x}_1)$ and $\boldsymbol{z}_2 = h_c(\boldsymbol{x}_2)$.

Interpolation For a category c and for images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ belonging to c, obtain $\boldsymbol{x}_{\text{interp}} = f_c(\alpha\boldsymbol{y}_1 + (1-\alpha)\boldsymbol{y}_2, \beta\boldsymbol{z}_1 + (1-\beta)\boldsymbol{z}_2)$ where $\boldsymbol{y}_i = g_c(\boldsymbol{x}_i)$ and $\boldsymbol{z}_i = h_c(\boldsymbol{x}_i)$ with $0 \leq \alpha, \beta \leq 1$ and $i = 1, 2$.

Random generation For a category c, obtain $\boldsymbol{x}_{\text{rand}} = f_c(\boldsymbol{y}, \boldsymbol{z})$ with $\boldsymbol{y} \sim \mathcal{N}_L(0, \boldsymbol{I})$ and $\boldsymbol{z} \sim \mathcal{N}_M(0, \boldsymbol{I})$.

Figure 4 shows examples of these feature manipulations on a 3-category CIGMO model trained on ShapeNet. As we can see, swapping gives a clear alignment of views in rows and shapes in columns. Interpolation gives smooth changes of images from one image to another in both shape and view dimensions. Random generation gives new images most of which are recognizable as each category.

D ADDITIONAL RESULTS FOR MULTIPIE

Tables 4 and 5 show the quantitative results of one-shot classification accuracy and category-wise shape-view disentanglement.

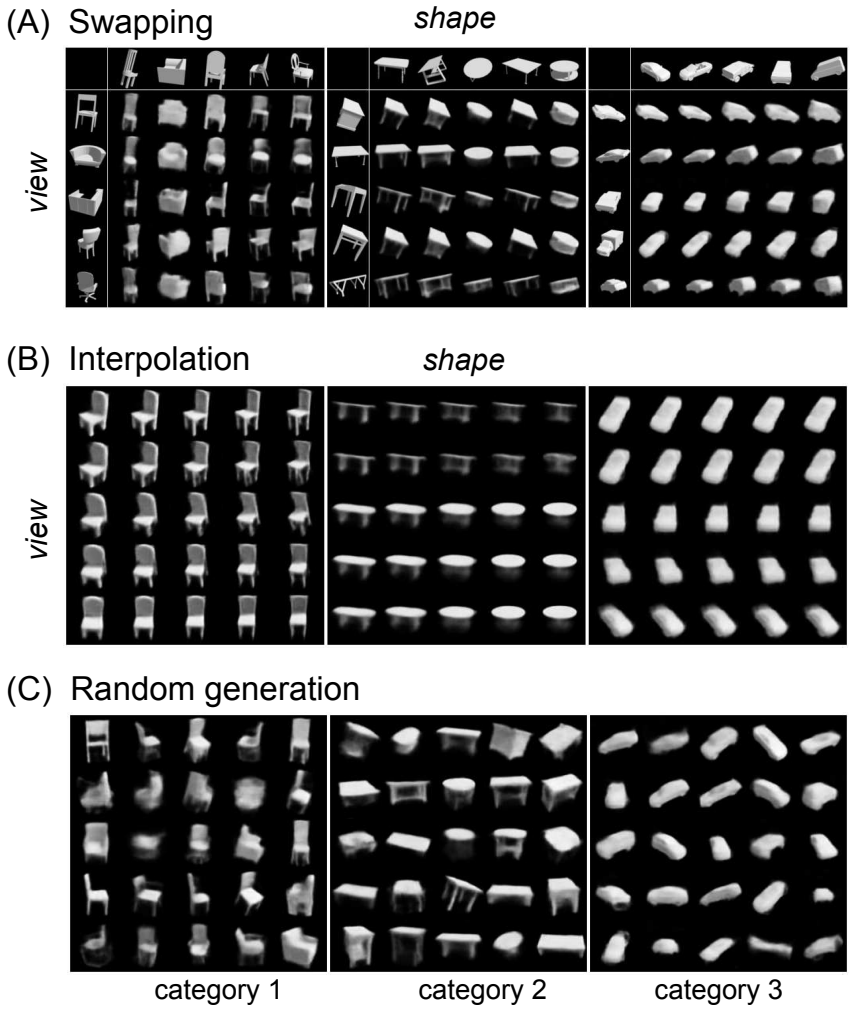


Figure 4: Examples of feature manipulation tasks from a 3-category CIGMO model for ShapeNet dataset. (A) Swapping. For each category, we generate a matrix of images from two lists of sample images, where each generated image has the view of an image in the left list and the shape of an image in the top list. (B) Interpolation. For each category, we generate a matrix of images from two sample images (corresponding to the top-left and bottom-right images), where each generated image has the view and shape that linearly interpolates those of the two images. (C) Random generation. For each category, we generate images from shapes and views that are drawn from Gaussian distributions.

	MultiPie
chance level	0.24
VAE	7.31 ± 0.25
Mix. of VAEs	6.64 ± 0.63
MLVAE	37.89 ± 0.60
GVAE	$\mathbf{37.96} \pm 0.37$
CIGMO	34.09 ± 1.62

Table 4: One-shot classification accuracy (%) for MultiPie. The mean and SD over 10 model instances are shown.

	MultiPie
MLVAE	92.29 ± 0.54
GVAE	92.76 ± 0.43
CIGMO	88.92 ± 1.46
MLVAE	0.50 ± 0.04
GVAE	0.51 ± 0.04
CIGMO	0.88 ± 0.22

Table 5: Quality of shape-view disentanglement for MultiPie, measured as neural network classification accuracy (%) for object identity from the shape (top rows; higher is better) or view variable (bottom rows; lower is better); weighted average over categories. The mean and SD over 10 model instances are shown.

HANDLING LONG-TAIL QUERIES WITH SLICE-AWARE CONVERSATIONAL SYSTEMS

Cheng Wang, Sun Kim, Taiwoo Park, Sajal Choudhary
Amazon Alexa AI
{cwngam, kimzs, parktaiw, sajalc}@amazon.com

Sunghyun Park, Young-Bum Kim, Ruhi Sarikaya, Sungjin Lee
Amazon Alexa AI
{sunghyu, youngbum, rsarikay, sungjinl}@amazon.com

ABSTRACT

We have been witnessing the usefulness of conversational AI systems such as Siri and Alexa, directly impacting our daily lives. These systems normally rely on machine learning models evolving over time to provide quality user experience. However, the development and improvement of the models are challenging because they need to support both high (head) and low (tail) usage scenarios, requiring fine-grained modeling strategies for specific data subsets or slices. In this paper, we explore the recent concept of slice-based learning (SBL) (Chen et al., 2019) to improve our baseline conversational skill routing system on the tail yet critical query traffic. We first define a set of labeling functions to generate weak supervision data for the tail intents. We then extend the baseline model towards a slice-aware architecture, which monitors and improves the model performance on the selected tail intents. Applied to de-identified live traffic from a commercial conversational AI system, our experiments show that the slice-aware model is beneficial in improving model performance for the tail intents while maintaining the overall performance.

1 INTRODUCTION

Conversational AI systems such as Google Assistant, Amazon Alexa, Apple Siri and Microsoft Cortana have become more prevalent in recent years (Sarikaya, 2017). One of the key techniques in those systems is to employ machine learning (ML) models to route a user's spoken utterance to the most appropriate skill that can fulfill the request. This requires the models to first capture the semantic meaning of the request, which typically involves assigning the utterance query to the candidate domain, intent, and slots (El-Kahky et al., 2014). For example, "Play Frozen" can be interpreted with *Music* as the domain, *Play Music* as the intent, and *Album Name:Frozen* as the slot key and value. Then, the models can route the request to a specific skill, which is an application that actually executes to deliver an experience (Li et al., 2021). For commercial conversational AI systems, there usually exists a large-scale dataset of user requests with ground-truth semantic interpretations and skills (e.g., through manual annotations and hand-crafted rules or heuristics). Along with various contextual signals, it is possible to train ML models (e.g., deep neural networks) with high predictive accuracy in routing a user request to the most appropriate skill, which then can continue to optimize towards better user experience through implicit or explicit user feedback (Park et al., 2020).

Nevertheless, developing such ML models or improving existing ones towards better user experience is still challenging. One hurdle is the imbalance in the distribution of the user queries with a long tail in terms of traffic volume. This often makes it difficult for the ML models to learn the patterns from the long-tail queries, some of which could be for critical features. Several approaches have been proposed to address such imbalance issue (Smith et al., 2014; He et al., 2008; Chawla et al., 2002). However, they are mainly based on applying reverse-discriminative sampling strategies,

for example, over-sampling minority and/or under-sampling majority. The sampling methods are usually insufficient in inspecting and improving model performance on pre-defined data subgroups.

In this work, we focus on the problem of imbalanced queries, specifically on tail but critical intents, in the context of the recently proposed slice-based learning (SBL) (Chen et al., 2019). SBL is a novel programming model that sits on top of ML systems. The approach first inspects particular data subsets (Ratner et al., 2019), which are called slices, and it improves the ML model performance on those slices. While the capability of monitoring specific slices is added to a pre-trained ML model (which is termed the *backbone* model), the approach has shown that overall performance across the whole traffic is comparable to those without SBL. Motivated by this idea, we propose to adopt the SBL concept to our baseline skill routing approach (we term the baseline model as $\mathbf{P}$; please refer to Sec. 3.1 for details) to improve its performance on tail yet critical intent queries while keeping the overall performance intact. First, we define slice functions (i.e., labeling functions) to specify the intents that we want to monitor. A pre-trained $\mathbf{P}$ is used as a backbone model for extracting the representation for each query. Then, we extend $\mathbf{P}$ to a slice-aware architecture, which learns to attend to the tail intent slices of interest.

We perform two experiments using a large-scale dataset with de-identified customer queries. First, we examine the attention mechanism in the extended model $\mathbf{P}$ with SBL. In particular, we test two attention weight functions with different temperature parameters in computing the probability distribution over tail intent slices. Second, we compare SBL to an upsampling method in $\mathbf{P}$ for handling tail intents. Our experiments demonstrate that SBL is able to effectively improve the ML model performance on tail intent slices as compared to the upsampling approach, while maintaining the overall performance.

We describe the related work in Section 2. In Section 3, we explain the baseline skill routing model, $\mathbf{P}$, and then elaborate how to extend it to a slice-aware architecture. The experiment results are reported in Section 4, and in Section 5, we discuss the advantages and potential limitations of applying SBL in our use case. We conclude this work in Section 6.

2 Related Work

2.1 Slice-Based Learning

Slice-based learning (SBL) (Chen et al., 2019) is a novel programming model that is proposed to improve ML models on critical data slices without hurting overall performance. A core idea of SBL is to represent a sample differently depending on the data subset or slice to which it belongs. It defines and leverages slice functions, i.e., pre-defined labeling functions, to generate weak supervision data for learning slice-aware representations. For instance, in computer vision (CV) applications, a developer can define object detection functions to detect whether an image contains a bicycle or not. In natural language understanding (NLP) applications, a developer can define intent-specific labeling functions such as for *Play Music* intent. SBL exhibits better performance than a mixture of experts (Jacobs et al., 1991) and multi-task learning (Caruana, 1997), with reduced run-time cost and parameters (Chen et al., 2019). Recently Gustavo et al. (Penha & Hauff, 2020) have employed the concept of SBL to understand failures of ranking models and identify difficult instances in order to improve ranking performance. Our work applies the idea to improve skill routing performance on low traffic but critical intents in conversational AI systems.

2.2 Weakly Supervised Learning

Weakly supervised learning attempts to learn predictive models with noisy and weak supervision data. Typically, there are three types of weak supervision: incomplete supervision, inexact supervision, and inaccurate supervision (Zhou, 2018). Various weakly supervised ML models are developed in NLP (Medlock & Briscoe, 2007; Huang et al., 2014; Wang & Manning, 2014) and in CV (Prest et al., 2011; Oquab et al., 2015; Peyre et al., 2017). Recently, promising approaches have been proposed to generate weak supervision data by programming training data (Ratner et al., 2016). In a large-scale industry setting, weak supervision data are highly desired given that human annotations are costly and time-consuming. Our work relates to weakly supervised learning in terms of inaccurate supervision. We split queries into different groups (slices) by defining labeling functions (slice functions). Each group is assigned with a group identity label. In practice, the slice functions may

not perfectly assign labels to input data as mentioned in SBL (Chen et al., 2019; Cabannnes et al., 2020).

2.3 CONVERSATIONAL SKILL ROUTING MODELS

In conversational AI systems, a skill refers to the application that actually executes on a user query or request to deliver an experience, such as playing a song or answering a question. The skills often comprise both first-party and third-party applications (Li et al., 2021). The skill routing is a mechanism that maps users' queries, given contextual information such as semantic interpretations and device types, to an appropriate application. The routing decision is usually determined by an ML model that is separate from typical natural language understanding (NLU) models for domain, intent, slot parsing. Please refer to section 3.1 for more details.

3 SLICE-AWARE CONVERSATIONAL SKILL ROUTING MODELS

This section explains our skill routing model (backbone model) and then explains how we extend the backbone model to a slice-aware architecture by adapting the concept from SBL (Chen et al., 2019).

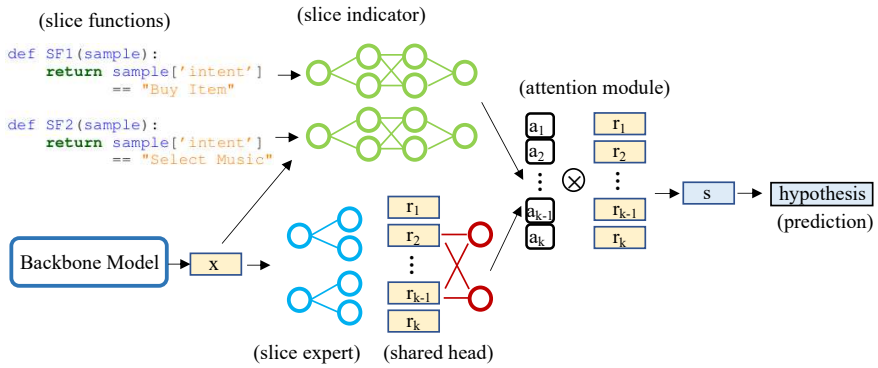


Figure 1: The slice-aware conversational skill routing model architecture for handling low traffic but critical intents. It consists of six components: (1) *slice functions* define tail intent slices that we want to monitor; (2) *backbone model* is our pre-trained skill routing model $\mathbf{P}$ that is used for feature extraction; (3) *slice indicators* are membership functions to predict if a sample query belongs to a tail slice; (4) *slice experts* aim to learn slice-specific representations; (5) *shared head* is the base task predictive layer across experts; (6) An *attention module* is used to re-weight the slice-specific representations $\mathbf{r}$ and form a slice-aware representation $\mathbf{s}$. Finally, the learned $\mathbf{s}$ is used to predict a final hypothesis (associated skill). The predicted hypothesis is used to serve a user query.

3.1 BACKBONE MODEL

We take our baseline skill routing approach ($\mathbf{P}$) as the backbone model and aim to make it a slice-aware architecture. $\mathbf{P}$ is a skill routing model, which takes in a list of routing candidates to select the most appropriate one. Each routing candidate is represented as a hypothesis with various contextual signals, such as utterance text, device type, semantic interpretation, and associated skill. While some contextual signals are common across all hypotheses, some are unique due to the presence of multiple competing semantic interpretations and skill-specific context. The core component of $\mathbf{P}$ consists of attention-based bi-directional LSTMs (Hochreiter & Schmidhuber, 1997; Graves & Schmidhuber, 2005) with fully connected layers on top of it. Formally, Let X be the set of query signals (e.g., utterance text, semantic interpretations, device type, etc.), $H = \{h_1, ..., h_n\}$ be the hypothesis list and $h_g \in H, g = [1, n]$ be the ground-truth hypothesis. The learning objective is to minimize the binary cross entropy:

$$\zeta_{base} = \mathcal{L}_{bce}(\pi(\mathcal{M}(X, H)), h_g), \tag{1}$$

```
def intent_based_slice_function_1(sample):
    return sample['intent'] == "Buy Item"

def intent_based_slice_function_2(sample):
    return sample['intent'] == "Select Music"

def intent_based_slice_function_3(sample):
    return sample['intent'] == "Buy Book"
```

Table 1: The slice functions (SFs) which split user queries into multiple data slices according to the pre-defined tail intents. The non-tail intents are in a base slice. Note SFs are only available at training stage for generating weak supervision labels. At inference stage, SFs will not be applied.

where π is a linear predictive layer which outputs a prediction over hypotheses $\hat{H} = \{\hat{h}_1, ..., \hat{h}_n\}$, and $\mathcal{M}$ is a set of multiple neural network layers, which extract the representation $\mathbf{x} \in \mathbb{R}^{n \times d}$ for a given (X, H) pair, i.e., $\mathbf{x} = \mathcal{M}(X, H)$.

To evaluate the effectiveness of trained $\mathbf{P}$, we define offline evaluation metric called replication accuracy (RA):

$$RA(\mathcal{D}_{test}) = \sum\nolimits_{(X,H,h_g) \in \mathcal{D}_{test}} \frac{\mathbb{I}(\hat{h}_g = h_g)}{|\mathcal{D}_{test}|}. \tag{2}$$

The replication accuracy measures how effectively the trained model $\mathbf{P}$ replicates the current skill routing behavior in production which is a combination of ML model and rules. Though $\mathbf{P}$ achieves high performance, replicating most of heuristic patterns, it suffers from low RA in low-volume traffic, i.e., the tail user queries. We later introduce how we extend $\mathbf{P}$ with a slice-aware component.

3.2 Slice-Aware Architecture

As presented in Figure 1, a slice-aware architecture consists of several components.

Slice Function. We first define slice (or labeling) functions to slice user queries according to intent (e.g., "Buy Book"). The selected intents have a small number of query instances, making the model $\mathbf{P}$ difficult to learn data patterns from tail intents. Each sample is assigned a slice label $\gamma \in [0, 1]$ in $\{\gamma_1, \gamma_2, ..., \gamma_k\}$ for supervision. s_1 is the base slice, and s_2 to s_k are the tail slices.

Slice Indicator. For each tail intent slice, a slice indicator (membership function) is learned to indicate whether a sample belongs to this particular slice or not. For a given representation $\mathbf{x} \in \mathbb{R}^{n \times d}$ from the backbone model, we learn $u_i = f_i(\mathbf{x}; \mathbf{w}_i^f)$, $\mathbf{w}_i^f \in \mathcal{R}^{d \times 1}$, $i \in \{1, .., k\}$ that maps $\mathbf{x}$ to $\mathbf{u} = \{u_1, ..., u_k\}$. f_i is trained with $\{\mathbf{x}, \gamma\}$ pairs with the binary cross entropy $\zeta_{ind} = \sum_i^k \mathcal{L}_{bce}(\mathbf{u}_i, \gamma_i)$.

Slice Expert. For each tail intent slice, a slice expert $g_i(\mathbf{x}; \mathbf{w}_i^g)$, $\mathbf{w}_i^g \in \mathcal{R}^{d \times d}$ is used to learn a mapping from $\mathbf{x} \in \mathbb{R}^{n \times d}$ to a slice vector $r_i \in \mathcal{R}^d$ with the samples only belonging to the tail slice. Followed by a **shared head**, which is shared across all experts and maps r_i to a prediction $\hat{h} = \varphi(r_i; \mathbf{w}_s)$, g_i and φ are learned on the base (original) task with ground-truth label h_g by $\zeta_{exp} = \sum_i^k \gamma_i \mathcal{L}_{bce}(\hat{h}, h_g)$.

Attention Module. The attention module decides how to pay special attention to the monitored slices. The distribution over slices (or attention weights) are computed based on stacked k membership likelihood $P \in \mathbb{R}^k$ and stacked k experts' prediction confidence $Q \in \mathbb{R}^{k \times c}$ as described in (Chen et al., 2019):

$$a2 = \text{SOFTMAX}(P + |Q|). \tag{3}$$

Note, the above equation is used when $c = 1$ (i.e., binary classification). As our task is a multi-class classification task where $c \geq 2$, we use an additional linear layer to transform $Q \in \mathbb{R}^{k \times c}$ to $\phi(Q) \in \mathbb{R}^k$. Finally, we experiment with the following different ways to compute attention weights, i.e., slice distribution:

$$a1 = \text{SOFTMAX}(P/\tau) \tag{4}$$

$$a2 = \text{SOFTMAX}([P + |\phi(Q)|]/\tau). \tag{5}$$

In Eq. 4, we only use the output of the indicator function (membership likelihood) in computing attention weights. In Eq. 5 we use both the membership likelihood and the transformed experts' prediction scores. The τ is a temperature parameter. In principle, smaller τ can lead to a more confident slice distribution (Wang & Niepert, 2019; Wang et al., 2021), hence we aim to examine if a small τ helps improve the routing performance.

4 EXPERIMENTS

We evaluate the skill routing model $\mathbf{P}$ with slice-based learning (SBL) (Chen et al., 2019) (we term it as $\mathbf{S}$) by performing two groups of experiments. First, we test the attention module with different methods of computing the attention weights over slices. Second, we compare the effectiveness of SBL against upsampling – a commonly used method for handling tail data.

4.1 EXPERIMENT SETUP AND IMPLEMENTATION DETAILS

We obtained live traffic from a commercial conversational AI system in production and processed the data so that individual users are not identifiable. We randomly sampled to create an adequately large data set for each training and test dataset. We further split the training set into training and validation sets with a ratio of 9:1. We used the replication accuracy (Eq. 2) to measure the model performance.

The existing production model $\mathbf{P}$ and its extension with SBL were implemented with Pytorch (Paszke et al., 2019). The hidden unit size for slice component was 128. All models were trained on AWS p3.8xlarge instances with Intel Xeon E5-2686 CPUs, 244 GB memory, and 4 NVIDIA Tesla V100 GPUs. We used Adam (Kingma & Ba, 2014) with a learning rate of 0.001 as the optimizer. Each model was trained with 10 epochs with the batch size of 256. We split the user queries into 21 data slices in total, one base slice and the rest for 20 tail intent slices. For each extracted query representation $\mathbf{x}$ for the tail intents, we add a Gaussian noise $\mathbf{x} = \mathbf{x} + \delta,\ \delta \sim \mathcal{N}(0, 0.005)$ to augment the tail queries.

4.2 EXPERIMENTS ON THE ATTENTION MECHANISMS

Table 2 shows the absolute score difference in replication accuracy between the baseline model and its SBL extension, having the baseline model's all-intent accuracy as a reference. As shown in the table, the slice-based approaches maintain the baseline performance overall, but the RA performance is lifted on the monitored tail slices. The best attention mechanism outperforms the baseline by 0.1% in tail intents' replication accuracy[1]. Tuning the temperature parameter between $\tau = 0.1$ or $\tau = 1.0$ does not significantly improve model performance on the tail intents.

Attention Methods	All Intents (%)	Tail Intents (%)
$\mathbf{P}$ (baseline model)	>99	–1.45
SBL, Eq. (4), $\tau = 1.0$	+0.01	–1.35
SBL, Eq. (5), $\tau = 1.0$	+0.01	–1.36
SBL, Eq. (4), $\tau = 0.1$	+0.01	–1.34
SBL, Eq. (5), $\tau = 0.1$	+0.01	–1.38

Table 2: The performance comparison of the baseline model $\mathbf{P}$ and its SBL extension with different attention weights in replication accuracy. All data points denote the absolute difference from the baseline model's all intents accuracy value.

[1]Given the large volume of query traffic per day, 0.1% is still a significant improvement in our system.

4.3 COMPARISON BETWEEN SLICE-BASED LEARNING AND UPSAMPLING

As upsampling is a widely used method to alleviate the tail data problem, we compare the performance between SBL and upsampling methods. Note SBL offers an additional advantage for inspecting particular tail data groups which are also critical. We denote the models as the following:

- $\mathbf{P}$ is the baseline model that is trained without applying upsampling.
- $\mathbf{S}$ is an extension of $\mathbf{P}$ (as a backbone model) to be a slice-aware model, which is trained with same training set as $\mathbf{P}$.
- $\mathbf{P}_{up}$ is the baseline model that is trained with applying upsampling.
- $\mathbf{S}_{up}$ is an extension of $\mathbf{P}_{up}$ to be a slice-aware model, which is trained with same training set as $\mathbf{P}_{up}$.

All the trained models are evaluated on the same test set. Among the aforementioned attention method choices, Eq. 4 with $\tau = 1.0$ is employed for $\mathbf{S}$ and $\mathbf{S}_{up}$. Our primary goal is to see whether $\mathbf{S}$ can improve $\mathbf{P}_{up}$.

Table 3 shows the performance comparison. When comparing $\mathbf{P}_{up}$ and $\mathbf{S}$, we can see $\mathbf{S}$ achieves slightly better performance for all intents. For the monitored tail intents, $\mathbf{S}$ achieves a slightly higher score as compared to $\mathbf{P}_{up}$.

Models	All Intents (%)	Tail Intents (%)
$\mathbf{P}$	>99	–1.41
$\mathbf{P}_{up}$	0.00	–1.47
$\mathbf{S}$	+0.01	–1.30
$\mathbf{S}_{up}$	+0.01	–1.37

Table 3: Performance comparison between the baseline model and its slice-aware architecture. $\mathbf{P}$ is the baseline model without upsampling, $\mathbf{P}_{up}$ is $\mathbf{P}$ with upsampling. $\mathbf{S}$ is the slice learning model with $\mathbf{P}$ as the backbone model, and $\mathbf{S}_{up}$ is the slice learning model with $\mathbf{P}_{up}$ as the backbone model. All data points are absolute score difference from the baseline model's all intent accuracy value.

Table 4 presents the absolute RA difference between the baseline and slice-aware models for the monitored 20 tail intents. Comparing $\mathbf{S}$ and $\mathbf{P}_{up}$, $\mathbf{S}$ improves the model performance on 14 tail intents. Compared to $\mathbf{P}_{up}$, $\mathbf{S}$ shows the comparable performance lift while effectively suppressing performance drops, for example, intent IDs 2, 3, 6, 15, and 20. As a result, $\mathbf{P}_{up}$ shows lower performance on 12 intents out of 20 (–2.41% on average), while $\mathbf{S}$ did on only 5 intents (–0.21% on average). This suggests the capability of slice-based learning in treating target intents through the slice-aware representation.

5 DISCUSSION

In our experiments, we have shown the effectiveness of SBL in terms of improving model performance on tail intent slices. It is beneficial to have ML models which are slice-aware, particularly when we want to inspect some specific and critical but low-traffic instances. Although the overall performance gain of slice-aware approach compared to the upsampling was marginal, it is worthwhile to note that the slice-aware approach was able to lift up the replication accuracy for more number of tail intents while minimizing unexpected performance degradation that was more noticeable in the upsampling approach. This result implies that the slice-aware approach has more potential in stably and evenly supporting tail intents.

On the other hand, we also note a potential limitation of SBL in the case of addressing tail intents in the industry setting. As we increase the number of tail intents, for instance to 200 intents, the model's complexity increases as well, given that an indicator function and an expert head are needed for each slice. However, this does not necessarily diminish the value of the slice-aware architecture, as the upsampling method offers no chance for us to inspect and analyze model failures on particular slices.

Tail Intent ID	$\mathbf{P}$	$\mathbf{P}_{up}$	$\mathbf{S}$	$\mathbf{S}_{up}$	Sample Size
1	>99	–0.03	0.00	–0.02	Over 10K
2	>96	–0.4	+0.04	–0.21	Over 10K
3	>96	–0.19	+0.09	–0.18	Over 10K
4	>72	+0.07	+1.98	+0.96	Over 10K
5	>99	+0.01	–0.01	0.00	Over 10K
6	>96	–0.09	+0.02	–0.11	Over 10K
7	>96	+0.03	+0.02	–0.04	Over 10K
8	>99	+0.15	+0.01	+0.19	Over 10K
9	>96	–0.24	+0.06	+0.07	Over 10K
10	>96	+0.08	–0.03	0.00	Between 1K - 10K
11	>99	0.00	0.00	0.00	Between 1K - 10K
12	>96	+0.55	–0.13	+0.46	Between 1K - 10K
13	>96	+0.36	+0.42	+0.53	Between 1K - 10K
14	>93	–0.39	–0.14	–0.42	Between 1K - 10K
15	>93	–3.29	–0.73	–1.46	Between 1K - 10K
16	>96	–1.2	0.00	–0.93	Below 1K
17	>96	–0.71	0.00	–0.71	Below 1K
18	>99	–0.16	0.00	–0.16	Below 1K
19	>99	–0.96	0.00	–1.15	Below 1K
20	>96	–21.21	0.00	–18.18	Below 1K

Table 4: Score (in %) differences in RA between the baseline and slice-aware approaches at the intent level. The baseline model's accuracy scores are rounded down to the nearest multiple of 3 percent, while the other models' are absolute score differences from the baseline ones. We denote each intent with their IDs. Sample Size is the number of random instances used for testing.

Further studies are necessary to employ and fine-tune the slice-based approach to serve tail traffic in a cost-effective way.

6 CONCLUSION

In this work, we applied and implemented the concept of slice-based learning to our skill routing model for a large-scale commercial conversational AI system. To enable the existing model to pay extra attention to selected tail intents, we tested different ways of computing slice distribution by using membership likelihood and experts' prediction confidence scores. Our experiments show that the slice-based learning can effectively and evenly improve model performance on tail intents while maintaining overall performance. We also compared the slice learning method against upsampling in terms of handling tail intents. The results suggest that slice-based learning outperforms upsampling by a small margin, while more evenly uplifting tail intents' performance. A potential future work would be to explore how to adapt SBL to monitor a large number of slices with minimum model and runtime complexity.

REFERENCES

Vivien Cabannnes, Alessandro Rudi, and Francis Bach. Structured prediction with partial labelling through the infimum loss. In *International Conference on Machine Learning*, pp. 1230–1239. PMLR, 2020.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Vincent Chen, Sen Wu, Alexander J Ratner, Jen Weng, and Christopher Ré. Slice-based learning: A programming model for residual learning in critical data slices. In *Advances in neural information processing systems*, pp. 9397–9407, 2019.

Ali El-Kahky, Xiaohu Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4067–4071. IEEE, 2014.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328. IEEE, 2008.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. Learning representations for weakly supervised natural language processing tasks. *Computational Linguistics*, 40(1):85–120, 2014.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Han Li, Sunghyun Park, Aswarth Dara, Jinseok Nam, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, and Ruhi Sarikaya. Neural model robustness for skill routing in large-scale conversational ai systems: A design choice exploration. *arXiv preprint arXiv:2103.03373*, 2021.

Ben Medlock and Ted Briscoe. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pp. 992–999, 2007.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 685–694, 2015.

Sunghyun Park, Han Li, Ameen Patel, Sidharth Mudgal, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, and Ruhi Sarikaya. A scalable framework for learning from implicit user feedback to improve natural language understanding in large-scale conversational ai systems. *arXiv preprint arXiv:2010.12251*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Gustavo Penha and Claudia Hauff. Slice-aware neural ranking. In *Proceedings of the 5th International Workshop on Search-Oriented Conversational AI (SCAI)*, pp. 1–6, 2020.

Julia Peyre, Josef Sivic, Ivan Laptev, and Cordelia Schmid. Weakly-supervised learning of visual relations. In *Proceedings of the ieee international conference on computer vision*, pp. 5179–5188, 2017.

Alessandro Prest, Cordelia Schmid, and Vittorio Ferrari. Weakly supervised learning of interactions between humans and objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):601–614, 2011.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29:3567–3575, 2016.

Alexander J Ratner, Braden Hancock, and Christopher Ré. The role of massively multi-task and weak supervision in software 2.0. In *CIDR*, 2019.

Ruhi Sarikaya. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81, 2017.

Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. An instance level analysis of data complexity. *Machine learning*, 95(2):225–256, 2014.

Cheng Wang and Mathias Niepert. State-regularized recurrent neural networks. In *International Conference on Machine Learning*, pp. 6596–6606, 2019.

Cheng Wang, Carolin Lawrence, and Mathias Niepert. Uncertainty estimation and calibration with finite-state probabilistic {rnn}s. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=9EKHN1jOlA`.

Mengqiu Wang and Christopher D Manning. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2: 55–66, 2014.

Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1): 44–53, 2018.

TABULAR DATA MODELING VIA CONTEXTUAL EMBEDDINGS

Xin Huang *
Amazon AI
xinxh@amazon.com

Ashish Khetan
Amazon AI
khetan@amazon.com

Milan Cvitkovic
PostEra
mwcvitkovic@gmail.com

Zohar Karnin
Amazon AI
zkarnin@amazon.com

ABSTRACT

We introduce TabTransformer, a new tabular data modeling architecture based on deep self-attention Transformers. Our model works by embedding categorical features in a robust and contextual manual, resulting in better prediction performance. We evaluate TabTransformer for supervised setting through extensive experiments on fifteen publicly available datasets, and conclude that it outperforms the state-of-the-art deep learning methods for tabular data by at least 1.0% on mean AUC. Furthermore, for the semi-supervised setting we develop an unsupervised pre-training and fine-tuning paradigm to learn data-driven contextual embeddings, resulting in an average 2.1% AUC lift over the state-of-the-art methods. Lastly, we demonstrate that the contextual embeddings learned from TabTransformer provide better interpretability, and are highly robust against both missing and noisy data features.

1 INTRODUCTION

Tabular data regression and classification are crucial to many real-world applications such as recommender systems (Cheng et al., 2016), online advertising (Song et al., 2019), and sales forecasting (Pavlyshenko, 2019). Many machine learning competitions such as Kaggle (Kaggle, 2020) and KDD Cup (SIGKDD, 2020) are primarily designed to solve problems in tabular domain, where various machine learning models are built to take each instance (row of tabular data) as input and map it to a target value.

The state-of-the-art for modeling tabular data is tree-based ensemble methods such as the gradient boosted decision trees (GBDT) (Chen & Guestrin, 2016). This differs from modeling image and text data where all the existing competitive models are based on deep learning (Sandler et al., 2018; Devlin et al., 2019). The tree-based ensemble models are accurate, fast to train, and easy to interpret, making them highly favourable among machine learning practitioners. However, their limitations are significant compared with deep learning models: (a) they do not allow efficient end-to-end learning of image/text encoders in presence of multi-modality along with tabular data; (b) they do not fit into the state-of-the-art semi-supervised learning framework due to unreliable probability estimation produced by basic decision tree (Tanha et al., 2017); and (c) they do not enjoy the SoTA deep learning methods (Devlin et al., 2019) to handle missing and noisy data features.

A classical and popular model that is trained using gradient descent and hence allows end-to-end learning of image/text encoders is multi-layer perceptron (MLP). The MLPs usually learn parametric embeddings to encode categorical/continuous data features. But due to their shallow architecture and context-free nature of the learned embeddings, they have the following limitations: (a) neither the model nor the learned embeddings are interpretable; (b) it is not robust against missing and noisy data (Section 3.2); (c) for semi-supervised learning, they do not achieve competitive performance (Section 3.4). Most importantly, the prediction accuracy of MLPs do not match that of tree-based models on most of the datasets (Arik & Pfister, 2019). To bridge this performance gap, researchers

*Corresponding author

have proposed various deep learning models (Arik & Pfister, 2019; Song et al., 2019; Cheng et al., 2016; Guo et al., 2018). Although these deep learning models achieve comparable prediction accuracy, they do not address all the limitations of GBDT and MLP. Furthermore, their comparisons are done in a limited setting of a handful of datasets. In particular, in Section 3.3 we show that when compared to standard GBDT on a large collection of datasets, GBDT perform significantly better than these recent models.

Different from tabular domain, the application of embeddings has been studied extensively in natural language processing. The embedding technique encodes discrete words (a categorical variable) in a dense low dimensional space, beginning from Word2Vec (Rong, 2014) with the context-free word embeddings to BERT (Devlin et al., 2019) which provides the contextual word embeddings. Based on contextual embedding, the self-attention Transformers (Vaswani et al., 2017) has achieved state-of-the-art performance on many NLP tasks. Additionally, the pre-training/fine-tuning paradigm in BERT, which pre-trains the Transformers on a large coprus of unsupervised text and fine-tunes it on downstream tasks with labeled text, has shed light on tabular data modeling in semi-supervised learning.

Motivated by the successful applications of Transformers in NLP, we adapt them in tabular domain. Particularly, TabTransformer modifies a sequence of multi-head attention-based Transformer layers on parametric embeddings to transform them into contextual embeddings, bridging the performance gap between baseline MLP and GBDT models. We investigate the effectiveness and interpretability of the resulting contextual embeddings generated by the Transformers. We find that highly correlated features (including feature pairs in the same column and cross column) result in embedding vectors that are close together in Euclidean distance, whereas no such pattern exists in context-free embeddings learned in a baseline MLP model. We also study the robustness of the TabTransformer against random missing and noisy data. The contextual embeddings make them highly robust in comparison to MLPs. Finally, we exploit the pre-training/fine-tuning methodologies from NLP and propose a semi-supervised learning approach for pre-training TabTransformer using unlabeled data and fine-tuning it on labeled data.

One of the key benefits of our proposed method for semi-supervised learning is the two independent training phases: a costly pre-training phase on unlabeled data and a lightweight fine-tuning phase on labeled data. This differs from many state-of-the-art semi-supervised methods (Chapelle et al., 2009; Oliver et al., 2018; Stretcu et al., 2019) that require a single training job including both the labeled and unlabeled data. The separated training procedure benefits the scenario where the model needs to be pretrained once but fine-tuned multiple times for multiple target variables. This scenario is in fact quite common in the industrial setting as companies tend to have one large dataset (e.g. describing customers/products) and are interested in applying multiple analyses on this data. We summarize our contributions as follows:

1. We propose TabTransformer, an architecture that provides and exploits contextual embeddings of categorical features. We provide extensive experiments showing that TabTransformer is superior to SoTA deep network models.
2. We investigate the resulting contextual embeddings and highlight their interpretability, contrasted to parametric context-free embeddings achieved by existing art.
3. We demonstrate the robustness of TabTransformer against noisy and missing data.
4. We provide and extensively study a two-phase pre-training then fine-tune procedure for tabular data, beating the state-of-the-art performance of semi-supervised learning methods.

2 Architecture and Training Process

The TabTransformer architecture comprises a column embedding layer, a stack of N Transformer layers, and a multi-layer perceptron. Each Transformer layer (Vaswani et al., 2017) consists of a multi-head self-attention layer followed by a position-wise feed-forward layer. The architecture of TabTransformer is shown below in Figure 1.

In our experiments we use standard feature engineering techniques to transform special types such as text, zipcodes, ip addresses etc., into either numeric or categorical features. Although better techniques may exist for handling special data types they are outside the scope of this paper.

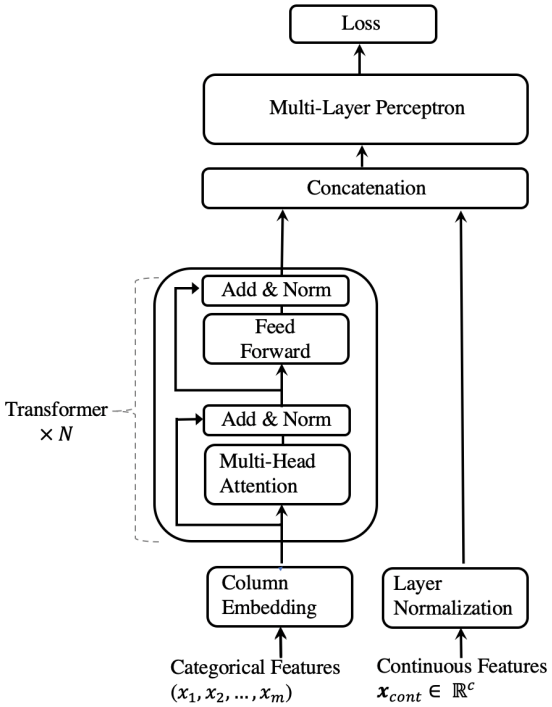


Figure 1: The architecture of TabTransformer.

Let $(\boldsymbol{x}, y)$ denote a features-target pair, where $\boldsymbol{x} \equiv \{\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{cont}}\}$ are processed features, and y is target value. The $\boldsymbol{x}_{\text{cat}}$ denotes all the categorical features and $\boldsymbol{x}_{\text{cont}} \in \mathbb{R}^c$ denotes all of the c continuous features. Let $\boldsymbol{x}_{\text{cat}} \equiv \{x_1, x_2, \cdots, x_m\}$ with each x_i being a categorical feature, for $i \in \{1, \cdots, m\}$. We embed each of the x_i categorical features into a parametric embedding of dimension d using *Column embedding*, which is explained below in detail. Let $\boldsymbol{e}_{\phi_i}(x_i) \in \mathbb{R}^d$ for $i \in \{1, \cdots, m\}$ be the embedding of the x_i feature, and $\boldsymbol{E}_\phi(\boldsymbol{x}_{\text{cat}}) = \{\boldsymbol{e}_{\phi_1}(x_1), \cdots, \boldsymbol{e}_{\phi_m}(x_m)\}$ be the set of embeddings for all the categorical features.

Next, these parametric embeddings $\boldsymbol{E}_\phi(\boldsymbol{x}_{\text{cat}})$ are passed through N Transformer layers. Each parametric embedding is transformed into contextual embedding when outputted from the top layer Transformer, through successive aggregation of context from other embeddings. We denote the sequence of N Transformer layers as a function f_θ. The function f_θ operates on parametric embeddings $\{\boldsymbol{e}_{\phi_1}(x_1), \cdots, \boldsymbol{e}_{\phi_m}(x_m)\}$ and returns the corresponding contextual embeddings $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_m\}$ where $\boldsymbol{h}_i \in \mathbb{R}^d$ for $i \in \{1, \cdots, m\}$. The contextual embeddings $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_m\}$ are concatenated along with the continuous features $\boldsymbol{x}_{\text{cont}}$ to form a vector of dimension $(d \times m + c)$. This vector is inputted to an MLP, denoted by $g_{\boldsymbol{\psi}}$, to predict the target y. Let H be the cross-entropy for classification tasks and mean square error for regression tasks. We minimize the following loss function $\mathcal{L}(\boldsymbol{x}, y)$ to learn all the TabTransformer parameters in an end-to-end learning by the first-order gradient methods. The TabTransformer parameters include $\boldsymbol{\phi}$ for column embedding, $\boldsymbol{\theta}$ for Transformer layers, and $\boldsymbol{\psi}$ for the top MLP layer.

$$\mathcal{L}(\boldsymbol{x}, y) \equiv H(g_{\boldsymbol{\psi}}(f_{\boldsymbol{\theta}}(\boldsymbol{E}_\phi(\boldsymbol{x}_{\text{cat}})), \boldsymbol{x}_{\text{cont}}), y)\,. \tag{1}$$

Below, we explain the column embedding.

Column embedding. For each categorical feature (column) i, we have an embedding lookup table $\boldsymbol{e}_{\phi_i}(.)$, for $i \in \{1, 2, ..., m\}$. For ith feature with d_i classes, the embedding table $\boldsymbol{e}_{\phi_i}(.)$ has $(d_i + 1)$ embeddings where the additional embedding corresponds to a missing value. The embedding for the encoded value $x_i = j \in [0, 1, 2, .., d_i]$ is $\boldsymbol{e}_{\phi_i}(j) = [\boldsymbol{c}_{\phi_i}, \boldsymbol{w}_{\phi_{ij}}]$, where $\boldsymbol{c}_{\phi_i} \in \mathbb{R}^\ell$ and $\boldsymbol{w}_{\phi_{ij}} \in \mathbb{R}^{d-\ell}$. The column-specific and unique identifier $\boldsymbol{c}_{\phi_i} \in \mathbb{R}^\ell$ distinguishes the classes in column i from those in the other columns. The dimension of $\boldsymbol{c}_{\phi_i}$, ℓ, is a hyper-parameter.

The use of unique identifier is innovative and particularly designed for tabular data. Rather in NLP, embeddings are element-wisely added with the positional encoding of the word in the sentence. Since, in tabular data, there is no ordering of the features, we do not use positional encodings. The strategies include both different choices for ℓ, d and element-wise adding the unique identifier and feature-value specific embeddings rather than concatenating them.

Pre-training the Embeddings. The contextual embeddings explained above are learned in end-to-end supervised training using labeled examples. For a scenario, when there are a few labeled examples and a large number of unlabeled ones, we introduce a pre-training procedure to train the Transformer layers using unlabeled data. This is followed by fine-tuning of the pre-trained Transformer layers along with the top MLP layer using the labeled data. For fine-tuning, we use the supervised loss defined in Equation 1.

We explore two different types of pre-training procedures, the masked language modeling (MLM) (Devlin et al., 2019) and the replaced token detection (RTD) (Clark et al., 2020). Given an input $\boldsymbol{x}_{\text{cat}} = \{x_1, x_2, ..., x_m\}$, MLM randomly selects $k\%$ features from index 1 to m and masks them as missing. The Transformer layers along with the column embeddings are trained by minimizing cross-entropy loss of a multi-class classifier, which predicts the original features of the masked features using contextual embeddings outputted from the top-layer Transformer.

Instead of masking features, RTD replaces the original feature by a random value of that feature. Here, the loss is minimized for a binary classifier that predicts whether or not the feature has been replaced. To compute the replacement value, the original RTD in Clark et al. (2020) uses an encoder network to sample a subset of features. The reason to use an encoder network is that there are tens of thousands of tokens in language data and a uniformly random token can be easily detected. In contrast, we use uniformly random values to replace tabular features because (a) the number of classes within each categorical feature is typically limited; (b) a different binary classifier is defined for each column rather than a shared one, as each column has its own embedding lookup table. We name the two pre-training methods as TabTransformer-MLM and TabTransformer-RTD. In our experiments, the replacement value k is set to 30.

3 Experiments

Data. We evaluate TabTransformer and baseline models on 15 publicly available binary classification datasets from the UCI repository (Dua & Graff, 2017), the AutoML Challenge (Guyon et al., 2019), and Kaggle (Kaggle, 2020) for both supervised and semi-supervised learning. Each dataset is divided into five cross-validation splits. The training/validation/testing proportion of the data for each split are $65/15/20\%$. The number of categorical features across dataset ranges from 2 to 136. In the semi-supervised experiments, for each dataset and split, p observations in the training data are uniformly sampled as the labeled data with a fixed random seed and the remaining training data are set as unlabeled set. The value of p is chosen as 50, 200, and 500, corresponding to 3 different scenarios. In the supervised experiments, each training dataset is fully labeled.

Setup. For TabTransformer, the hidden (embedding) dimension, the number of layers and the number of attention heads are fixed to 32, 6, and 8 respectively; these parameters are pre-selected by hyperparameter optimization (HPO) on a small number of datasets. The MLP layer sizes are set to $\{4 \times l, 2 \times l\}$, where l is the size of its input. Each competitor model is given 20 HPO rounds for each cross-validation split. For evaluation metrics, we use the Area under the curve (AUC) (Bradley, 1997). Note, the pre-training is only applied in semi-supervised scenario. We do not find much benefit in using it when the entire data is labeled. Its benefit is evident when there is a large number of unlabeled examples and a few labeled examples. Since in this scenario the pre-training provides a representation of the data that could not have been learned based only on the labeled examples.

3.1 The Effectiveness of the Transformer Layers

First, a comparison between TabTransformers and the baseline MLP is conducted in a supervised learning scenario. We remove the Transformer layers $f_{\boldsymbol{\theta}}$ from the architecture, fix the rest of the components, and compare it with the original TabTransformer. The model without the Transformer layers is equivalently an MLP. The dimension of embeddings d for categorical features is set as 32 for both models. The comparison results over 15 datasets are presented in Table 1. The TabTransformer

Table 1: Comparison between TabTransfomers and the baseline MLP. The evaluation metric is AUC in percentage.

Dataset	Baseline MLP	TabTransformer	Gain (%)
albert	74.0	75.7	**1.7**
1995_income	90.5	90.6	**0.1**
dota2games	63.1	63.3	**0.2**
hcdr_main	74.3	75.1	**0.8**
adult	72.5	73.7	**1.2**
bank_marketing	92.9	93.4	**0.5**
blastchar	83.9	83.5	-0.4
insurance_co	69.7	74.4	**4.7**
jasmine	85.1	85.3	**0.2**
online_shoppers	91.9	92.7	**0.8**
philippine	82.1	83.4	**1.3**
qsar_bio	91.0	91.8	**0.8**
seismicbumps	73.5	75.1	**1.6**
shrutime	84.6	85.6	**1.0**
spambase	98.4	98.5	**0.1**

with the Transformer layers outperforms the baseline MLP on 14 out of 15 datasets with an average 1.0% gain in AUC.

Next, we take contextual embeddings from different layers of the Transformer and compute a t-SNE plot (Maaten & Hinton, 2008) to visualize their similarity in function space. More precisely, for each dataset we take its test data, pass their categorical features into a trained TabTransformer, and extract all contextual embeddings (across all columns) from a certain layer of the Transformer. The t-SNE algorithm is then used to reduce each embedding to a 2D point in the t-SNE plot. Figure 2 (left) shows the 2D visualization of embeddings from the last layer of the Transformer for dataset *bank_marketing*. Each marker in the plot represents an average of 2D points over the test data points for a certain class. We can see that semantically similar classes are close with each other and form clusters (annotated by a set of labels) in the embedding space. For example, all of the client-based features (color markers) such as job, education level and martial status stay close in the center and non-client based features (gray markers) such as month (last contact month of the year), day (last contact day of the week) lie outside the central area; in the bottom cluster the embedding of owning a housing loan stays close with that of being default; over the left cluster, embeddings of being a student, martial status as single, not having a housing loan, and education level as tertiary get together; and in the right cluster, education levels are closely associated with the occupation types (Torpey & Watson, 2014). In Figure 2, the center and right plots are t-SNE plots of embeddings before being passed through the Transformer and the context-free embeddings from MLP, respectively. For the embeddings before being passed into the Transformer, it starts to distinguish the non-client based features (gray markers) from the client-based features (color markers). For the embeddings from MLP, we do not observe such pattern and many categorical features which are not semantically similar are grouped together, as indicated by the annotation in the plot. We also evaluate the effectiveness of TabTransformer by fitting embeddings extracted from different layers into a linear model.

3.2 The Robustness of TabTransformer

We demonstrate the robustness of TabTransformer on the noisy data and data with missing values, against the baseline MLP. We consider these two scenarios only on categorical features to specifically prove the robustness of contextual embeddings from the Transformer layers.

Noisy Data. On the test examples, we firstly contaminate them by replacing a certain number of values by randomly generated ones from the corresponding columns (features). Next, the noisy data are passed into a trained TabTransformer to compute a prediction AUC score. Results on a set of 3 datasets are presented in Figure 3. As the noisy rate increases, TabTransformer performs better

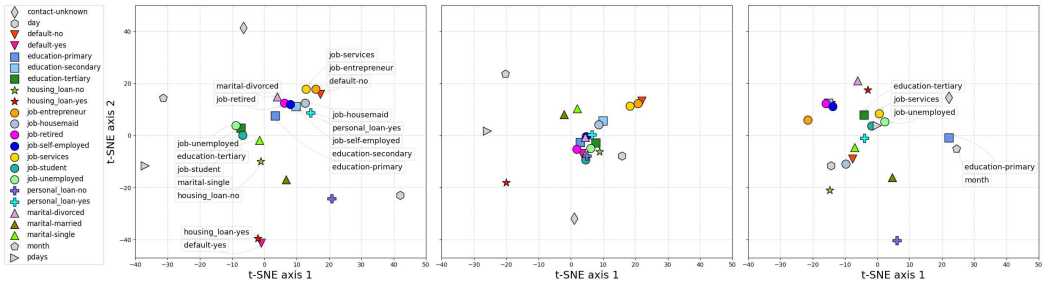


Figure 2: t-SNE plots of learned embeddings for categorical features on dataset *BankMarketing*. **Left**: TabTransformer – the embeddings generated from the last layer of the attention-based Transformer. **Center**: TabTransformer – the embeddings before being passed into the attention-based Transformer. **Right**: The embeddings learned from MLP.

in prediction accuracy and thus is more robust than MLP. In particular notice the *Blastchar* dataset where the performance is near identical with no noise, yet as the noise increases, TabTransformer becomes significantly more performant compared to the baseline. We conjecture that the robustness comes from the contextual property of the embeddings. Despite a feature being noisy, it draws information from the correct features allowing for a certain amount of correction.

Data with Missing Values. Similarly, on the test data we artificially select a number of values to be missing and send the data with missing values to a trained TabTransformer to compute the prediction score. Figure 4 shows the same patterns of the noisy data case, i.e. that TabTransformer shows better stability than MLP in handling missing values.

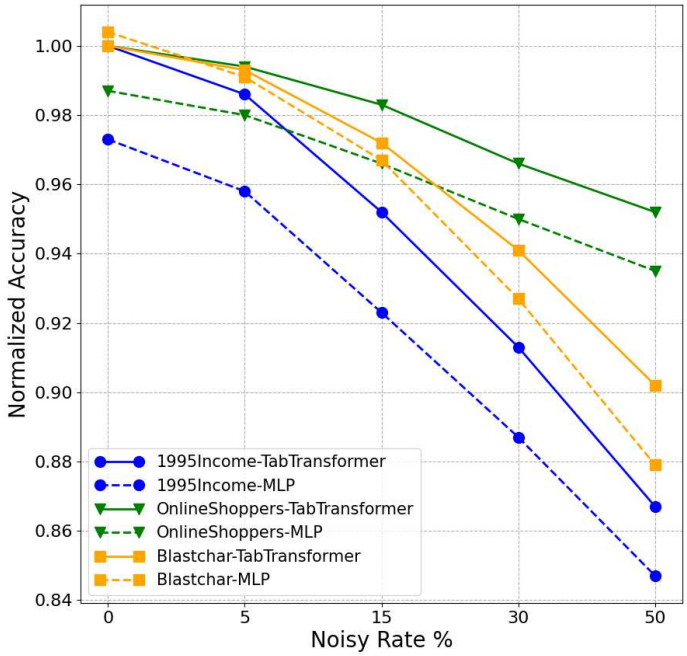


Figure 3: Performance of TabTransformer and MLP with noisy data. For each dataset, each prediction score is normalized by the score of TabTransformer at 0 noise.

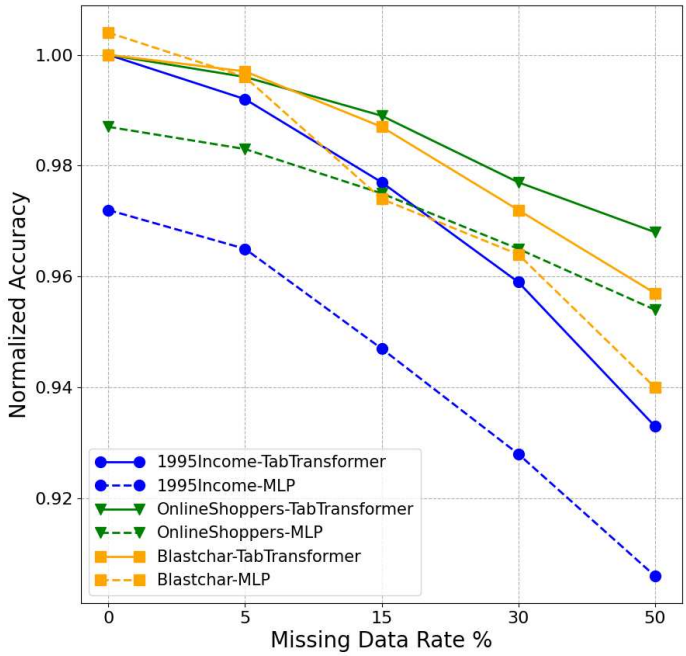


Figure 4: Performance of TabTransformer and MLP under missing data scenario. For each dataset, each prediction score is normalized by the score of TabTransformer trained without missing values.

Table 2: Model performance in supervised learning. The evaluation metric is mean $\pm$ standard deviation of AUC score over the 15 datasets for each model. Larger the number, better the result. The top 2 numbers are bold.

Model Name	**Mean AUC (%)**
TabTransformer	$\mathbf{82.8} \pm 0.4$
MLP	81.8 ± 0.4
GBDT	$\mathbf{82.9} \pm 0.4$
Sparse MLP	81.4 ± 0.4
Logistic Regression	80.4 ± 0.4
TabNet	77.1 ± 0.5
VIB	80.5 ± 0.4

3.3 Supervised Learning

Here we compare the performance of TabTransformer against following four categories of methods: (a) Logistic regression and GBDT; (b) MLP and a sparse MLP following Morcos et al. (2019); (c) TabNet model of Arik & Pfister (2019); and (d) the Variational Information Bottleneck model (VIB) of Alemi et al. (2017).

Results are summarized in Table 2. TabTransformer, MLP, and GBDT are the top 3 performers. The TabTransformer outperforms the baseline MLP with an average 1.0% gain and perform comparable with the GBDT. Furthermore, the TabTransformer is significantly better than TabNet and VIB, the recent deep networks for tabular data.

3.4 Semi-supervised Learning

We evaluate the TabTransformer under the semi-supervised learning scenario where few labeled training examples are available together with a significant number of unlabeled samples. Specifically, we compare our pretrained and then fine-tuned TabTransformer-RTD/MLM against following

Table 3: Semi-supervised learning results for 6 datasets with more than 30K data points, for different number of labeled data points. Evaluation metrics are mean AUC in percentage. Larger the number, better the result.

# Labeled data	**50**	**200**	**500**
TabTransformer-RTD	66.6 ± 0.6	70.9 ± 0.6	$\mathbf{73.1} \pm 0.6$
TabTransformer-MLM	$\mathbf{66.8} \pm 0.6$	$\mathbf{71.0} \pm 0.6$	72.9 ± 0.6
MLP (ER)	65.6 ± 0.6	69.0 ± 0.6	71.0 ± 0.6
MLP (PL)	65.4 ± 0.6	68.8 ± 0.6	71.0 ± 0.6
TabTransformer (ER)	62.7 ± 0.6	67.1 ± 0.6	69.3 ± 0.6
TabTransformer (PL)	63.6 ± 0.6	67.3 ± 0.7	69.3 ± 0.6
MLP (DAE)	65.2 ± 0.5	68.5 ± 0.6	71.0 ± 0.6
GBDT (PL)	56.5 ± 0.5	63.1 ± 0.6	66.5 ± 0.7

semi-supervised models: (a) Entropy Regularization (ER) (Grandvalet & Bengio, 2006) combined with MLP and TabTransformer; (b) Pseudo Labeling (PL) (Lee, 2013) combined with MLP, TabTransformer, and GBDT (Jain, 2017); (c) MLP (DAE): an unsupervised pre-training method designed for deep models on tabular data – the swap noise Denoising AutoEncoder (Jahrer, 2018).

The pre-training models TabTransformer-MLM, TabTransformer-RTD and MLP (DAE) are firstly pretrained on the entire unlabeled training data and then fine-tuned on labeled data. The semi-supervised learning methods, Pseudo Labeling and Entropy Regularization, are trained on the mix of labeled and unlabeled training data. To better present results, we split the set of 15 datasets into two subsets. The first set includes 6 datasets with more than 30K data points and the second set includes remaining 9 datasets.

The results are presented in Table 3 and Table 4. When the number of unlabeled data is large, Table 3 shows that our TabTransformer-RTD and TabTransformer-MLM significantly outperform all the other competitors. Particularly, TabTransformer-RTD/MLM improves over all the other competitors by at least 1.2%, 2.0% and 2.1% on mean AUC for the scenario of 50, 200, and 500 labeled data points respectively. The Transformer-based semi-supervised learning methods TabTransformer (ER), TabTransformer (PL), and the tree-based semi-supervised learning method GBDT (PL) perform worse than the average of all the models. When the number of unlabeled data becomes smaller, as shown in Table 4, TabTransformer-RTD still outperforms most of its competitors but with a marginal improvement.

Furthermore, we observe that when the number of unlabeled data is small as shown in Table 4, TabTransformer-RTD performs better than TabTransformer-MLM, thanks to its easier pre-training task (a binary classification) than that of MLM (a multi-class classification). This aligns with the finding of the ELECTRA paper (Clark et al., 2020). In Table 4, with only 50 labeled data points, MLP (ER) and MLP (PL) beat our TabTransformer-RTD/MLM. This can be attributed to that there is room to improve in our fine-tuning procedure. Particularly, our approach allows to obtain informative embeddings but does not allow the weights of the classifier itself to be trained with unlabelled data. Since this issue does not occur for ER and PL, they obtain an advantage in extremely small labelled set. We point out however that this only means that the methods are complementary and a possible follow up could combine the best of all approaches.

Both evaluation results, Table 3 and Table 4, show that our TabTransformer-RTD and Transformers-MLM models are promising in extracting useful information from unlabeled data to help supervised training, and are particularly useful when the size of unlabeled data is large.

4 Related Work

For supervised learning, standard MLPs have been applied to tabular data for many years (De Brébisson et al., 2015). For deep models designed specifically for tabular data, there are deep versions of factorization machines (Guo et al., 2018; Xiao et al., 2017), deep MLPs-based methods (Wang et al., 2017; Cheng et al., 2016; Cortes et al., 2016), Transformers-based methods (Song et al., 2019; Li et al., 2020; Sun et al., 2019), and deep versions of decision-tree-based algorithms

Table 4: Semi-supervised learning results for 9 datasets with less than 30K data points, for different number of labeled data points. Evaluation metrics are mean AUC in percentage. Larger the number, better the result.

# Labeled data	**50**	**200**	**500**
TabTransformer-RTD	78.6 ± 0.6	$\mathbf{81.6} \pm 0.5$	$\mathbf{83.4} \pm 0.5$
TabTransformer-MLM	78.5 ± 0.6	81.0 ± 0.6	82.4 ± 0.5
MLP (ER)	$\mathbf{79.4} \pm 0.6$	81.1 ± 0.6	82.3 ± 0.6
MLP (PL)	79.1 ± 0.6	81.1 ± 0.6	82.0 ± 0.6
TabTransformer (ER)	77.9 ± 0.6	81.2 ± 0.6	82.1 ± 0.6
TabTransformer (PL)	77.8 ± 0.6	81.0 ± 0.6	82.1 ± 0.6
MLP (DAE)	78.5 ± 0.7	80.7 ± 0.6	82.2 ± 0.6
GBDT (PL)	73.4 ± 0.7	78.8 ± 0.6	81.3 ± 0.6

(Ke et al., 2019; Yang et al., 2018). In particular, Song et al. (2019) apply one layer of multi-head attention on embeddings to learn higher order features. The higher order features are concatenated and inputted to a fully connected layer to make the final prediction. Li et al. (2020) use self-attention layers and track the attention scores to obtain feature importance scores. Sun et al. (2019) combine the Factorization Machine model with transformer mechanism. All 3 papers are focused on recommendation systems with input data being high dimensional and extremely sparse, which makes it hard to have a clear comparison with this paper. Recent TabNet (Arik & Pfister, 2019) is designed on the sparse feature interaction of tabular data, and has a very different mechanism than our self-attention based one. There are a few works focusing on database oriented task in tabular domain, such as column categorization (Chen et al., 2019), entity linking (Luo et al., 2018), table layout identification (Habibi et al., 2020), and table augmentation (Deng et al., 2019). However, these tasks are fundamentally different from typical ML classification problem. They do not classify individual rows of a table, but properties of the table itself, i.e., meta-data. For this reason we do not elaborate the details of these papers nor compare our results with theirs.

For semi-supervised learning, Izmailov et al. (2019) give a semi-supervised method based on density estimation and evaluate their approach on tabular data. *Pseudo labeling* (Lee, 2013) is an efficient and popular baseline method. The pseudo labeling uses the current network to infer pseudo-labels of unlabeled examples, by choosing the most confident class. These pseudo-labels are treated like human-provided labels in the cross entropy loss. *Label propagation* (Zhu & Ghahramani, 2002; Iscen et al., 2019) is a similar approach where a node's labels propagate to all nodes according to their proximity, and are used by the training model as if they were the true labels. Another standard method in semi-supervised learning is *entropy regularization* (Grandvalet & Bengio, 2005; Sajjadi et al., 2016). It adds average per-sample entropy for the unlabeled examples to the original loss function for the labeled examples. Additionally, a classical approach of semi-supervised learning is co-training (Nigam & Ghani, 2000). However, the recent approaches - entropy regularization and pseudo labeling - are typically better and more popular. A succinct review of semi-supervised learning methods in general can be found in Oliver et al. (2019); Chappelle et al. (2010).

5 Conclusion

We proposed TabTransformer, a novel deep tabular data modeling architecture for supervised and semi-supervised learning. Extensive experiments show that TabTransformer significantly outperforms recent deep networks while matching the performance of GBDT. In addition, we study a two-phase pre-training/fine-tune paradigm for tabular data, beating the state-of-the-art semi-supervised learning methods. For future work, it would be interesting to investigate them in detail.

REFERENCES

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *International Conference on Learning Representations*, abs/1612.00410, 2017. URL https://arxiv.org/abs/1612.00410.

Sercan O Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *arXiv preprint arXiv:1908.07442*, 2019. URL https://arxiv.org/abs/1908.07442.

Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning). *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

O Chappelle, B Schölkopf, and A Zien. Semi-supervised learning. adaptive computation and machine learning, 2010.

Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. Learning semantic annotations for tabular data. *arXiv preprint arXiv:1906.00781*, 2019.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1xMH1BtvB.

Corinna Cortes, Xavi Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks, 2016.

Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial neural networks applied to taxi destination prediction. In *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ECMLPKDDDC'15, pp. 40–51, Aachen, DEU, 2015. CEUR-WS.org.

Li Deng, Shuo Zhang, and Krisztian Balog. Table2vec: neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1029–1032, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pp. 529–536, 2005.

Yves Grandvalet and Yoshua Bengio. Entropy regularization. *Semi-supervised learning*, pp. 151–168, 2006.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, Xiuqiang He, and Zhenhua Dong. DeepFM: An End-to-End Wide & Deep Learning Framework for CTR Prediction. *arXiv:1804.04950 [cs, stat]*, May 2018. URL http://arxiv.org/abs/1804.04950. arXiv: 1804.04950.

Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michéle Sebag, Alexander Statnikov, Wei-Wei Tu, and Evelyne Viegas. Analysis of the automl challenge series 2015-2018. In *AutoML*, Springer series on Challenges in Machine Learning, 2019. URL https://www.automl.org/wp-content/uploads/2018/09/chapter10-challenge.pdf.

Maryam Habibi, Johannes Starlinger, and Ulf Leser. Deeptable: a permutation invariant neural network for table orientation classification. *Data Mining and Knowledge Discovery*, 34(6):1963–1983, 2020.

Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5070–5079, 2019.

Pavel Izmailov, Polina Kirichenko, Marc Finzi, and Andrew Gordon Wilson. Semi-Supervised Learning with Normalizing Flows. *arXiv:1912.13025 [cs, stat]*, December 2019. URL http://arxiv.org/abs/1912.13025. arXiv: 1912.13025.

Michael Jahrer. Porto Seguro's Safe Driver Prediction, 2018. URL https://kaggle.com/c/porto-seguro-safe-driver-prediction.

Shubham Jain. Introduction to pseudo-labelling : A semi-supervised learning technique. https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/, 2017.

Inc. Kaggle. 2020 kaggle machine learning & data science survey, 2020. URL https://www.kaggle.com/c/kaggle-survey-2020.

Guolin Ke, Jia Zhang, Zhenhui Xu, Jiang Bian, and Tie-Yan Liu. TabNN: A universal neural network solution for tabular data, 2019. URL https://openreview.net/forum?id=r1eJssCqY7.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 2, 2013.

Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. Interpretable Click-Through Rate Prediction through Hierarchical Attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 313–321, Houston TX USA, January 2020. ACM. ISBN 978-1-4503-6822-3. doi: 10.1145/3336191.3371785. URL http://dl.acm.org/doi/10.1145/3336191.3371785.

Xusheng Luo, Kangqi Luo, Xianyang Chen, and Kenny Zhu. Cross-lingual entity linking for web tables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Ari S. Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv:1906.02773 [cs, stat]*, October 2019. URL http://arxiv.org/abs/1906.02773. arXiv: 1906.02773.

Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pp. 86–93, 2000.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 3235–3246, 2018.

Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *arXiv:1804.09170 [cs, stat]*, June 2019. URL http://arxiv.org/abs/1804.09170. arXiv: 1804.09170.

Bohdan M Pavlyshenko. Machine-learning models for sales time series forecasting. *Data*, 4(1):15, 2019.

Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*, pp. 1163–1171, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

SIGKDD, 2020. URL https://www.kdd.org/kdd-cup.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management - CIKM '19*, pp. 1161–1170, 2019. doi: 10.1145/3357384.3357925. URL http://arxiv.org/abs/1810.11921. arXiv: 1810.11921.

Otilia Stretcu, Krishnamurthy Viswanathan, Dana Movshovitz-Attias, Emmanouil Platanios, Sujith Ravi, and Andrew Tomkins. Graph Agreement Models for Semi-Supervised Learning. In *Advances in Neural Information Processing Systems 32*, pp. 8713–8723. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9076-graph-agreement-models-for-semi-supervised-learning.pdf.

Qiang Sun, Zhinan Cheng, Yanwei Fu, Wenxuan Wang, Yu-Gang Jiang, and Xiangyang Xue. DeepEnFM: Deep neural networks with Encoder enhanced Factorization Machine. September 2019. URL https://openreview.net/forum?id=SJlyta4YPS.

Jafar Tanha, Maarten Someren, and Hamideh Afsarmanesh. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8:355–370, 2017.

Elka Torpey and Audrey Watson. *Education level and jobs: Opportunities by state*, 2014. URL https://www.bls.gov/careeroutlook/2014/article/education-level-and-jobs.htm.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *ADKDD@KDD*, 2017.

Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 3119–3125, Melbourne, Australia, August 2017. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3. doi: 10.24963/ijcai.2017/435. URL https://www.ijcai.org/proceedings/2017/435.

Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

ACTIVE WEASUL: IMPROVING WEAK SUPERVISION WITH ACTIVE LEARNING

Samantha Biegel[1,2], **Rafah El-Khatib**[1,3], **Luiz Otavio Vilas Boas Oliveira**[1], **Max Baak**[1] **& Nanne Aben**[1]
[1] ING Wholesale Banking Advanced Analytics, Amsterdam, The Netherlands
[2] University of Amsterdam, Amsterdam, The Netherlands
[3] Feebris, London, United Kingdom
samantha.r.biegel@gmail.com
rafah@feebris.com
{luiz.vilas.boas.oliveira, max.baak, nanne.aben}@ing.com

ABSTRACT

The availability of labelled data is one of the main limitations in machine learning. We can alleviate this using weak supervision: a framework that uses expert-defined rules $\boldsymbol{\lambda}$ to estimate probabilistic labels $p(y|\boldsymbol{\lambda})$ for the entire data set. These rules, however, are dependent on what experts know about the problem, and hence may be inaccurate or may fail to capture important parts of the problem-space. To mitigate this, we propose Active WeaSuL: an approach that incorporates active learning into weak supervision. In Active WeaSuL, experts do not only define rules, but they also iteratively provide the true label for a small set of points where the weak supervision model is most likely to be mistaken, which are then used to better estimate the probabilistic labels. In this way, the weak labels provide a warm start, which active learning then improves upon. We make two contributions: 1) a modification of the weak supervision loss function, such that the expert-labelled data inform and improve the combination of weak labels; and 2) the maxKL divergence sampling strategy, which determines for which data points expert labelling is most beneficial. Our experiments show that when the budget for labelling data is limited (e.g. ≤ 60 data points), Active WeaSuL outperforms weak supervision, active learning, and competing strategies, with only a handful of labelled data points. This makes Active WeaSuL ideal for situations where obtaining labelled data is difficult.[1]

1 INTRODUCTION

Machine learning models often require large amounts of labelled data to work properly. However, collecting large amounts of high-quality labelled data is not always straightforward. While in some cases we may be able to outsource the labelling process at a competitive price, we often require the use of expensive domain experts to do the labelling, for example due to the inherent difficulty of the task or due to privacy concerns preventing us from sharing the data externally. In these cases, it is imperative that we explore ways in which we can make use of our domain experts in a more efficient way.

One way to do so is through a process called weakly supervised learning (or weak supervision for short) (Zhou, 2018). In this process, we ask domain experts to define labelling functions: rules that they think are indicative of a given class, such as "IF $X_1 < 5$ THEN $y = 1$" (Figure 1A). These labelling functions are applied to the data, resulting in a set of weak labels $\boldsymbol{\lambda}$, which are then combined using a generative model, resulting in the probabilistic labels $p(y|\boldsymbol{\lambda})$ for all data points (Figure 1B). Finally, the probabilistic labels are used as a proxy to train a discriminative model that predicts $\mathbf{y}$ from the feature matrix $\mathbf{X}$ (Figure 1D). In short, weak supervision allows one to train a supervised model using expert-defined rules.

[1] Code for Active WeaSuL: https://github.com/SamanthaBiegel/ActiveWeaSuL

While weak supervision has shown to be very powerful (Ratner et al., 2016; Bach et al., 2019; Badene et al., 2019; Dunnmon et al., 2020), we observe that it does not always reach its optimal performance. For example, Figure 1D shows that the predicted decision boundary is rotated with respect to the optimal decision boundary. In practice, this may happen when the expert-defined rules are imprecise or cover only parts of the problem-space, when the conditional independence between weak labels is not properly specified, or due to biases introduced by the generative model.

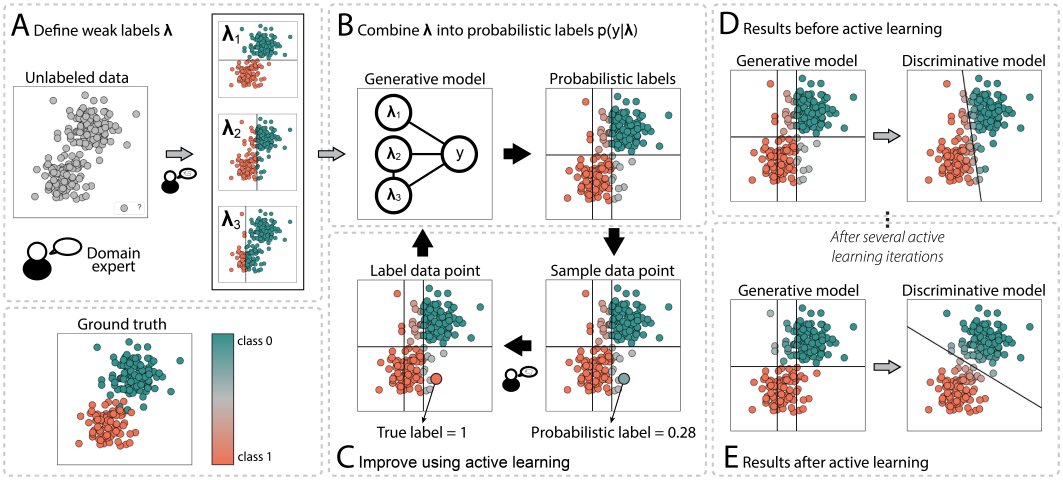


Figure 1: Overview of the Active WeaSuL approach. A) A domain expert defines the labelling functions, which when applied to data are converted to weak labels $\boldsymbol{\lambda}$. B) A generative model is used to combine the weak labels $\boldsymbol{\lambda}$ into probabilistic labels $p(y|\boldsymbol{\lambda})$. C) We use an active learning approach to iteratively select the best data point for the expert to label. This information is then used to improve the estimation of $p(y|\boldsymbol{\lambda})$. D&E) Predictions $p(y|\boldsymbol{X})$ of a discriminative model trained to predict probabilistic labels $p(y|\boldsymbol{\lambda})$ from the feature matrix $\mathbf{X}$ both D) before active learning and E) after active learning.

Inspired by Nashaat et al. (2019), we propose to address this problem by incorporating a second kind of domain expert knowledge: besides the expert-defined rules, we ask domain experts to label a small number of data points (Figure 1C). Specifically, we do this in an active learning fashion, in which we iterate over: 1) determining for which data point the model would benefit most from knowing the true label y; 2) asking an expert to label this data point for us, which we consider as the ground truth label; and 3) using this labelled data point to improve the estimation of the probabilistic labels $p(y|\boldsymbol{\lambda})$. Optionally, one can also do a fourth step, which is to train a discriminative model on the obtained probabilistic labels, using the data's features. In this way, over the course of several iterations, we gradually improve the performance of both the generative and discriminative models (Figure 1E).

We call our approach Active WeaSuL, as a nod to the combination of active learning (i.e. iteratively learning from a small number of expert-labelled data points) and weakly supervised learning (i.e. learning from expert-defined rules). To combine these two methodologies, we make two important contributions: 1) a modification to the weak supervision loss function, allowing us to use expert-labelled data to improve the combination of weak labels; and 2) the maxKL divergence sampling method, a query strategy to determine for which data points expert-labelling would be most beneficial within this framework.

We benchmark Active WeaSuL against three alternative sets of models on three data sets. We first show that Active WeaSuL outperforms the three baseline approaches; i.e. active learning by itself, weak supervision by itself, as well as the original approach to combining active learning and weak supervision by Nashaat et al. (2019), in particular when we only have budget to label a limited number of data points. Subsequently, we show the benefits of the maxKL divergence sampling strategy compared to either random sampling or sampling based on the margin from the decision boundary.

2 RELATED WORK

The combination of active learning and weak supervision has been pioneered by Nashaat et al. (2019). In their work, they also iteratively obtain expert-labelled data which they then use to improve the weak supervision model. The most important difference with our work lies in the way in which the expert-labelled data is incorporated into the weak supervision model. While Active WeaSuL uses the expert-labelled data to learn how to best combine the weak labels (thereby affecting predictions for all data points), Nashaat et al. (2019) instead use the expert-labelled data to correct the weak labels for the corresponding individual data points. Specifically, if we have three labelling functions and for one data point i we have the weak labels $\boldsymbol{\lambda}_i = [1, 1, 0]$ and an expert-label $\mathbf{y}_i = 1$, then they correct the weak labels with the expert-label by setting $\boldsymbol{\lambda}_i = [1, 1, 1]$, whereas Active WeaSuL leaves the weak labels unchanged and instead improves the way in which the weak labelling functions are combined.

We note that Nashaat et al. (2019) have also released two follow-up works (Nashaat et al., 2020a;b). In these works, they use the same combination of active learning and weak supervision as in their original work. In Nashaat et al. (2020a), they additionally use a mechanism to automatically generate labelling functions based on a small set of labelled data. In this work, we assume that we have no labelled data at the beginning, and that the labelling functions are defined by experts, hence we compare Active WeaSuL to the original work by Nashaat et al. (2019).

Besides the approach outlined above, there are several others that combine active learning and weak supervision, though with a different goal in mind. For instance, several authors have proposed to use labelled data to improve how the labelling functions are defined (Boecking et al., 2020; Awasthi et al., 2020; Chen et al., 2019; Cohen-Wang et al., 2019; Varma & Ré, 2018). A nice example of this is the work of Boecking et al. (2020), who propose a method that automatically improves the set of labelling functions based on feedback from users. In this work, we take a different angle: we assume the labelling functions themselves are fixed and instead use the expert-labelled data to improve how we combine the weak labels.

Alternatively, several authors have proposed to use weak supervision to improve active learning (Brust et al., 2020; Gonsior et al., 2020), rather than the other way around. In this approach, for a small set of data points, each point is labelled by several experts. These labels are then combined into one estimate per data point using weak supervision (i.e. by considering each expert as a labelling function). Such an approach works well in scenarios where experts cannot reliably derive the true label. We note that this serves a different goal than our work: here, we assume that the experts are in fact able to reliably determine the true labels, but that the budget for labelling is limited. We instead focus on using weak supervision to incorporate expert-defined rules into our model, thereby gaining a warm start over active learning by itself, which otherwise needs to start from scratch.

3 METHODS

3.1 WEAK SUPERVISION

In weak supervision, we estimate probabilistic labels $p(y|\boldsymbol{\lambda})$ from a set of weak labels $\boldsymbol{\lambda}$ using a generative model. Here, we use the generative model as formulated by Ratner et al. (2019). Let $\boldsymbol{\lambda}$ be the $n \times m$ matrix of weak labels, where n is the number of samples and m is the number of weak labels; let $\boldsymbol{\lambda}_{i,*}$ be the ith row of this matrix (i.e. all weak labels for data point i); and let $\boldsymbol{\lambda}_{*,j}$ be the jth column of this matrix (i.e. all data points for weak label j). Consider $\boldsymbol{\Sigma}$ the $m \times m$ covariance matrix of $\boldsymbol{\lambda}$. Consider $\mathbf{y}$ the $n \times 1$ vector of binary ground truth labels.

Next, let us define two assumptions. First, let us assume that we know the class prior $p(y)$. We can obtain $p(y)$ either through expert knowledge or by estimating it from the data (Ratner et al., 2019). Second, we assume to know which weak labels are conditionally independent of each other given y. We denote this with the set Ω, where weak label i and weak label j are conditionally independent given y for all $(i, j) \in \Omega$. We can either obtain Ω through expert knowledge, for example because we know which labelling functions are conceptually unrelated, or we can estimate Ω from the data (Varma et al., 2019).

Given the conditional independence structure Ω, we can solve the matrix completion problem by solving the objective problem from Ratner et al. (2019) as:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} ||(\mathbf{\Sigma}^{-1} + \mathbf{z}\mathbf{z}^T)_\Omega||_F \tag{1}$$

where $(.)_\Omega$ is equivalent to $(.)_{(i,j)}$ for (i, j) in Ω and $||.||_F$ is the Frobenius norm. Given the class prior $p(y)$ and the conditional independence structure Ω, we can then define a function f that transforms the parameters $\hat{\mathbf{z}}$ into probabilistic labels as:

$$\hat{p}(y|\boldsymbol{\lambda}_{i,*}) = f(\hat{\mathbf{z}}, \boldsymbol{\lambda}_{i,*}) \tag{2}$$

For more details on these equations, we refer to Ratner et al. (2019).

3.2 Modifying the generative model loss function to incorporate active learning

In this work, we use active learning to improve the generative model's estimate of $p(y|\boldsymbol{\lambda}_{i,*})$. To this end, we modify the objective function to include a penalty $Pe(\boldsymbol{z})$ that nudges the optimal parameters towards a configuration where $p(y|\boldsymbol{\lambda}_{i,*}) = \mathbf{y}_i$ for all expert-labelled data points i.

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} ||(\mathbf{\Sigma}^{-1} + \mathbf{z}\mathbf{z}^T)_\Omega||_F + \alpha Pe(\mathbf{z}) \tag{3}$$

The hyper-parameter α can be tuned such that the penalty is competitive in size with the original loss term. The penalty $Pe(\mathbf{z})$ is defined as the quadratic difference between the probabilistic label $p(y|\boldsymbol{\lambda}_{i,*}) = f(\mathbf{z}, \boldsymbol{\lambda}_{i,*})$ and the true (binary) label $\mathbf{y}_i$ for all data points i in the set of expert-labelled data D obtained through active learning:

$$Pe(\mathbf{z}) = \sum_{i \in D} (f(\mathbf{z}, \boldsymbol{\lambda}_{i,*}) - \mathbf{y}_i)^2 \tag{4}$$

In short, the penalty encourages the generative model to combine the weak labels $\boldsymbol{\lambda}$ such that the resulting probabilistic labels are concordant with the expert-labelled data, i.e. $p(y|\boldsymbol{\lambda}_{i,*}) = \mathbf{y}_i$ for all expert-labelled data points i.

3.3 The maxKL divergence active learning sampling strategy

While the modified loss function allows us to improve the estimate of $p(y|\boldsymbol{\lambda}_{i,*})$, it does not tell us which data points we should ask the expert to label. To this end, we define the maxKL divergence active learning sampling strategy, based on Kullback-Leibler (KL) divergence, which we specifically tailor to the combination of active learning and weak supervision. The main idea is that, for each iteration, we sample a data point from the region where the generative model and the expert-labelled data disagree most, as we argue that these data points will be most informative to the model (Figure 2).

Consider $\boldsymbol{\gamma}$ as the $r \times m$ matrix of unique row values in $\boldsymbol{\lambda}$, where each row $\boldsymbol{\gamma}_{i,*}$ is a unique configuration of weak label values. We refer to these rows as *buckets* that split up the problem space (Figure 2A). Let t denote the active learning iteration index. Since we label one data point per active learning iteration, t also denotes the number of data points for which we have collected expert labels so far. Using these concepts, we can define the generative model's estimate of the probability of observing y in a given bucket i at a given iteration t as:

$$\hat{p}_t(y|\boldsymbol{\gamma}_{i,*}) = f(\hat{\mathbf{z}}, \boldsymbol{\gamma}_{i,*}) \tag{5}$$

where $\hat{\mathbf{z}}$ is updated using Equation 3 at every iteration t. Additionally, we define a second estimate of the probability of observing y in a given bucket i at a given iteration t, this time using expert-labelled data only:

$$\hat{q}_t(y|\boldsymbol{\gamma}_{i,*}) = \frac{\sum_{j \in D_i} \mathbf{y}_j}{|D_i|} \tag{6}$$

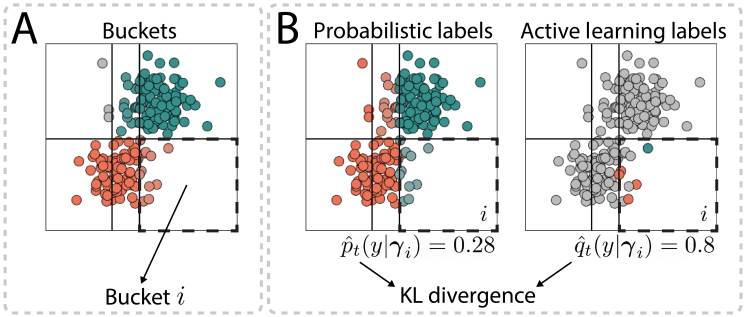


Figure 2: Overview of the active learning sampling method. A) The definition of a *bucket*, i.e. a unique configuration of weak label values. B) For each bucket i, we determine the KL divergence between the distributions of the probabilistic labels and the expert-labelled data. We then choose a data point from the bucket with the maximal divergence for expert-labelling in the next active learning iteration.

where D_i is the set of expert-labelled data points in bucket i. To prevent division by zero, we set $\hat{q}_t(y|\boldsymbol{\gamma}_{i,*}) = \text{round}(\hat{p}_{t=0}(y|\boldsymbol{\gamma}_{i,*}))$, i.e. to the generative model's estimate before any active learning iterations, rounded to the nearest binary value, when $|D_i| = 0$.

When for a bucket i the estimates $\hat{p}_t(y|\boldsymbol{\gamma}_{i,*})$ and $\hat{q}_t(y|\boldsymbol{\gamma}_{i,*})$ are very different from each other, it indicates that the generative model and the expert-labelled data are in disagreement with each other for this particular bucket. This suggests that for the next active learning iteration $t+1$ it would be highly informative to select a data point from that bucket for expert-labelling.

We quantify the difference between $\hat{p}_t(y|\boldsymbol{\gamma}_{i,*})$ and $\hat{q}_t(y|\boldsymbol{\gamma}_{i,*})$ using the KL divergence:

$$KL_{t,i}(p||q) = p \cdot \log \frac{p}{q} + (1-p) \cdot \log \frac{1-p}{1-q} \tag{7}$$

where we use $p = \hat{p}_t(y|\boldsymbol{\gamma}_{i,*})$ and $q = \hat{q}_t(y|\boldsymbol{\gamma}_{i,*})$. To prevent division by zero, we cap q to be within $[\epsilon, 1-\epsilon]$, where ϵ is a very small number.

We can now define our the maxKL divergence sampling strategy. At iteration t, we sample a data point from the bucket i that has the maximum KL divergence $KL_{t,i}$. We then ask an expert to label this data point, resulting in the updated estimates $\hat{p}_{t+1}(y|\boldsymbol{\gamma}_{i,*})$ and $\hat{q}_{t+1}(y|\boldsymbol{\gamma}_{i,*})$ for all buckets i. The complete algorithm for Active WeaSul is shown in Supplementary Algorithm 1.

4 Results

4.1 Benchmarking Active WeaSuL on artificial data

We first benchmark Active WeaSuL on the simple artificial data set illustrated in Figure 1. Essentially, this data set consists of two balanced classes, where each class is modelled by a 2-dimensional Gaussian, with a training set of 10,000 data points and a test set of 3,000 data points. Although the labels are known to us, we do not make them known to the model outside of the active learning steps. We define three labelling functions on these data: $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_3$ on the first dimension $\mathbf{X}_1$, and $\boldsymbol{\lambda}_1$ on the second dimension $\mathbf{X}_2$. Note that $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_3$ are not conditionally independent since they are both defined on $\mathbf{X}_1$, and we incorporate this information in the generative model via Ω (Equation 1). We set $\alpha = 1$ in Equation 3 since the loss of the initial generative model is close to zero.

To compare Active WeaSuL with weak supervision by itself, we consider the predictive performance of Active WeaSuL on these artificial data during the first 30 active learning iterations (Figure 3A&B). To this end, we obtain the predicted labels $\hat{\mathbf{y}}$ of the generative model of Active WeaSuL by rounding the probabilistic labels $\hat{p}_t(y|\boldsymbol{\lambda})$ to the nearest binary value, which we then compare to the ground truth labels $\mathbf{y}$ to determine the accuracy. For the generative model, we observe that the accuracy improves from 0.81 at $t = 0$ (i.e. weak supervision, without any active learning) to 0.96 within 4 active learning iterations (Figure 3A). We observe similar improvements using the discriminative

model (Figure 3B) (Supplementary Materials). This shows that combining weak supervision and active learning can indeed improve the predictive performance compared to weak supervision by itself.

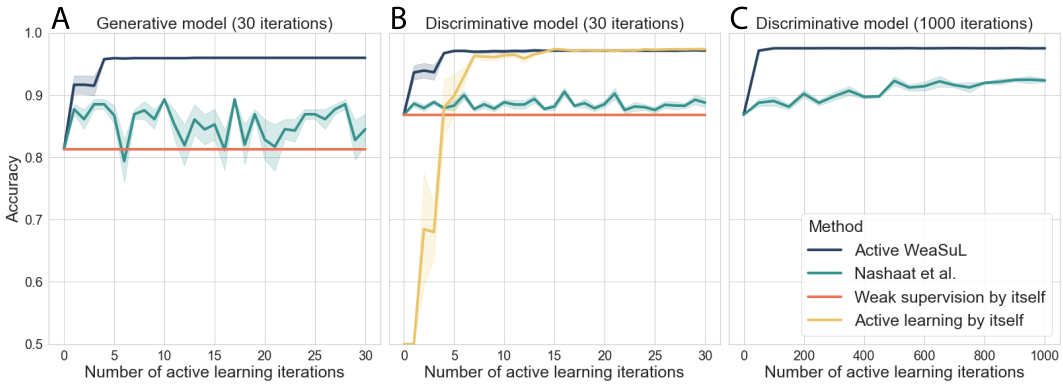


Figure 3: Predictive performance vs. active learning iterations on the artificial data set. Uncertainty bands were obtained by rerunning the experiment 10 times with a different random seed, affecting the random selection of a data point within the bucket selected by maxKL. A&B) Results for the first 30 active learning iterations for A) the generative models and B) discriminative models. C) Results for the discriminative models for Active WeaSuL and the method by Nashaat et al. (2019) for 1000 active learning iterations.

Next, we compare Active WeaSuL to active learning by itself. To this end, we train a logistic regression model to predict $\mathbf{y}$ from $\mathbf{X}$, combined with an active learning sampling strategy based on distance from the decision boundary. Figure 3B shows how this approach compares to Active WeaSuL. While active learning eventually reaches the same performance as Active WeaSuL, it requires 9 additional active learning iterations to do so. We reason that Active WeaSuL may reach its optimal performance sooner due to having a warm start: the information from the weak labels allows it to start from an accuracy of 0.87 instead of an accuracy of 0.5. Altogether, this shows that, on these data, Active WeaSuL outperforms active learning by itself.

Finally, we compare Active WeaSuL to the competing method by Nashaat et al. (2019). In the first 30 active learning iterations, their approach achieves roughly the same accuracy as weak supervision, and hence does not outperform Active WeaSuL (Figure 3A&B). When we extend the number of active learning iterations to 1000, their approach shows a slow increase in predictive performance, but still does not reach the performance that Active WeaSuL obtains after 4 iterations (Figure 3C). We note that in their own work, Nashaat et al. (2019) only report performance after obtaining labels for thousands of data points, hence our observations here are consistent with the results they reported themselves. For an interpretation of why the approach by Nashaat et al. (2019) requires so many more active learning iterations compared to Active WeaSuL, we refer to the Discussion section.

Overall, we have shown here that on the artificial data set illustrated in Figure 1, Active WeaSuL outperforms weak supervision and requires (far) fewer labelled data compared to either active learning or the approach by Nashaat et al. (2019).

4.2 Application to realistic data sets

We benchmark Active WeaSuL on two realistic data sets: a visual relationship detection task (Lu et al., 2016) and a spam detection task (Alberto et al., 2015).

In the visual relationship detection task, we predict whether a subject is sitting on top of an object based on: an image, bounding boxes of the subject and the object, and categorical variables indicating the type of object and subject (Figure 4A). The data consist of 794 data points in the training set and 185 data points in the test set. We use the bounding boxes and categorical variables to define a set of three weak labels (Supplementary Materials). Here too, we set $\alpha = 1$ in Equation 3, as the loss of the initial generative model is close to zero.

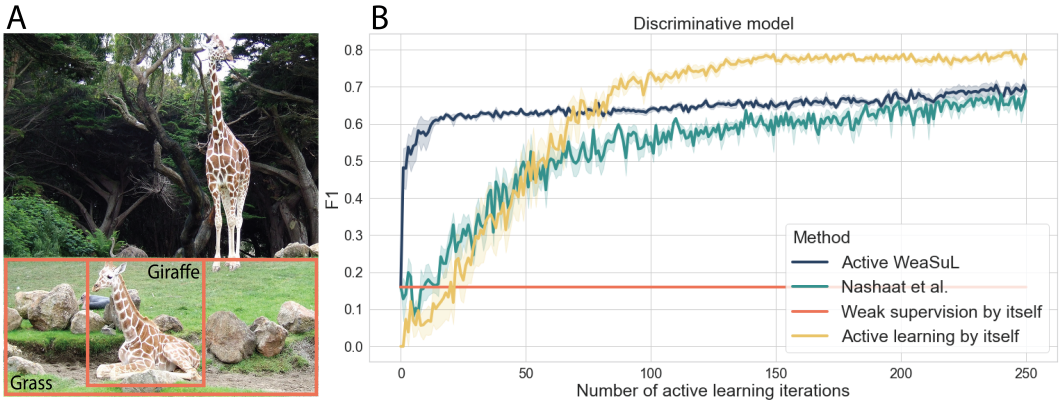


Figure 4: Predictive performance vs. active learning iterations on the visual relationship detection data set. A) Example of a visual relationship detection task: does the giraffe (subject) sit on the grass (object)? B) Results for the first 250 active learning iterations for the discriminative models.

Using these data, we compare the predictive performance of Active WeaSuL against the same set of methods as before. Because of the class imbalance, we use the F1 score to compare the methods. We focus on the predictive performance of the discriminative models, for which we use a neural network consisting of ResNet-18 (He et al., 2016) for the image data and GloVe (Pennington et al., 2014) for the categorical data (Supplementary Materials).

Active WeaSuL sharply increases its performance from 0.16 to 0.62 within 16 iterations (Figure 4B). Although active learning eventually trumps this performance, Active WeaSuL requires far fewer labelled data points to achieve a good predictive performance: it achieves a performance of 0.58 with only 7 iterations, whereas active learning requires 66 iterations and Nashaat et al. (2019) requires 77 iterations to achieve the same performance. This shows that on these data, Active WeaSuL can obtain good (though not optimal) predictive performance using very few labelled data points, thereby making it a good choice in settings where obtaining labels for a large number of data points is difficult.

The spam detection task aims to classify whether YouTube comments are spam (Alberto et al., 2015). These data consist of 1,586 training samples and 250 test samples. The weak supervision model is based on a Snorkel tutorial[2], where we use the first 7 of the 10 labelling functions defined therein. Similarly, we train the discriminative model as defined in the tutorial.

For this task, we found that Active WeaSuL's initial generative model (i.e. before any active learning iterations) had a relatively high loss of around 145. To make the penalty in Equation 3 competitive with the loss function, we suggest setting $\alpha = \max_{\mathbf{z}_i} \mathrm{d}^2 L_G / \mathrm{d}\mathbf{z}_i^2$, where $L_G = ||(\mathbf{\Sigma}^{-1} + \mathbf{z}\mathbf{z}^T)_\Omega||_F$ is the original loss of the generative model (left-hand term in Equation 3) and $\mathbf{z}_i$ are the fitted parameters in Equation 3. This is statistically motivated by interpreting the original loss function as a chi-squared function that is minimized. This results in a penalty term (right-hand term of Equation 3) that can compete with the original loss function. However, the best method to fine-tune α needs more study. Here we set the hyper-parameter $\alpha = 10^6$.

In Figure 5 we compare the predictive performance of Active WeaSuL against the same set of methods as before. We observe a similar pattern as for the other data sets: Active WeaSuL outperforms each of the baseline models. Its performance increases sharply from $F1 = 0.69$ to 0.85 within 4 iterations, which otherwise requires 27 iterations of active learning by itself, or 50 by Nashaat et al. (2019). We also observe that active learning does not overtake Active WeaSuL.

4.3 Active learning sampling strategy

Here, we zoom in on the benefits of our proposed active learning sampling strategy. To this end, we use the visual relationship detection data set to compare Active WeaSuL with three different

[2] https://github.com/snorkel-team/snorkel-tutorials/tree/master/spam

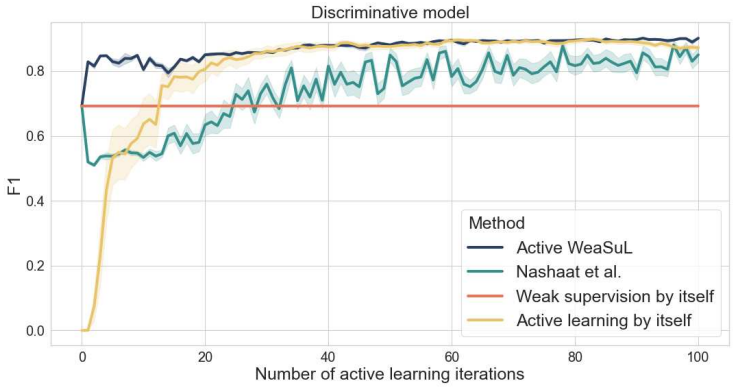


Figure 5: Predictive performance vs. active learning iterations on the spam detection data set. Results for the first 100 active learning iterations for the discriminative models.

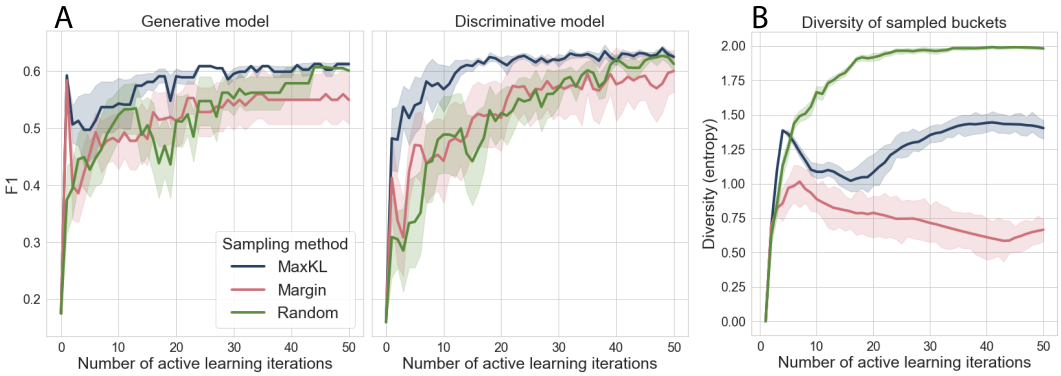


Figure 6: Comparison of the various sampling strategies on the visual relationship detection data set. A) The predictive performance for the generative model and the discriminative model. B) The diversity of the buckets that have been sampled.

sampling strategies: 1) maxKL divergence, our proposed strategy as described in Section 3; 2) margin, a strategy that samples data points whose probabilistic labels $\hat{p}_t(y|\boldsymbol{\lambda})$ are close to 0.5; and 3) random, a strategy that samples a data point from a random bucket.

Let us first consider the predictive performance of these various sampling strategies (Figure 6A). We find that the margin and random sampling strategies obtain the worst performance, suggesting that it is indeed beneficial to instead sample data points from buckets for which the model has trouble determining probabilistic labels.

An important shortcoming of the margin strategy is that it tends to repeatedly sample from the same bucket. This specifically happens when in a given bucket both classes are equally present, i.e. where $p_t(y|\boldsymbol{\lambda}) = 0.5$. Hence, compared to the other strategies, the margin strategy samples from a less diverse set of buckets, leading to labelled data that is less representative of the whole data set.

To quantify this behaviour, we define the following metric: given that we have sampled t data points, and that for a given bucket i we have sampled n_i data points, consider $r_i = n_i \,/\, t$ for each bucket i. We then summarise these fractions across buckets using entropy: $H = -\sum_i r_i \log r_i$. We note that entropy is high when the data points are sampled from a diverse set of buckets.

Indeed, we find that the margin strategy results in the lowest entropy, indicating that the data points were drawn from a non-diverse set of buckets (Figure 6B). Random sampling on the other hand leads to the highest entropy. This is in line with expectation, as random sampling should result in a highly representative set of data points (but not necessarily data points for which the model struggles). Our maxKL divergence approach results in entropy that lies between the two other

approaches, suggesting that it balances the selection of representative data points with the selection of data points for which the model struggles.

5 DISCUSSION & CONCLUSION

For all three data sets, the combination of weak supervision and active learning works because the knowledge from the expert rules is clearly informative, providing a useful warm start to solve the supervised classification problem, which active learning then improves upon. In all examples, the defined weak labels give probabilistic labels that are valuable and decent prior estimates, as visible from the resulting performance measures, albeit (slightly) biased and thus still suboptimal. The penalty term, which incorporates the manually labelled data points, pulls the probabilistic labels closer to their correct values, resulting in improved classifier performance. We argue that this happens even with only a few labelled data points, as the correct solution may be nearby, giving flexibility to move the assigned probabilistic labels.

We have shown that Active WeaSuL outperforms active learning early on, for example with ≤ 60 active learning iterations for the visual relationship detection data. We note however that active learning does eventually overtake Active WeaSuL. While early on the weak labels provide a warm start, we argue that in a later stage they may actually constrain the model too much: at some point a fully supervised model can simply learn more than a weakly supervised one. Hence, when it is possible to obtain a large number of labelled data points, one should use active learning. When the labelling budget is more limited, we suggest using Active WeaSuL.

All examples shown are straightforward classification problems where standalone active learning reaches optimal performance with only a limited number of manually labelled data points (13 for the artificial data; 150 and 40 for the realistic data). Interestingly, the number of manually labelled data points required for active learning to approach optimal performance on its own was higher in the (more complex) realistic data sets, suggesting that this number will further increase in even more complex problems. In such scenarios, the reduced labelling effort will also more prominently offset the initial effort of defining labelling functions. Altogether, we thus expect the added benefit of Active WeaSuL to further increase in more complex scenarios, specifically when the budget for requesting labelled data points is limited.

Finally, let us compare Active WeaSuL with the competing method by Nashaat et al. (2019). If for an expert-labelled data point i we have $\boldsymbol{\lambda}_{i,*} = [1, 1, 0]$ and $\mathbf{y}_i = 1$, Nashaat et al. (2019) correct the weak labels with the label provided by the expert, i.e. they assign $\boldsymbol{\lambda}_{i,*} = [1, 1, 1]$. While this clearly results in correct predictions for all expert-labelled data points, the effect on other data points is small and not necessarily beneficial. For example, the above correction increases the overall amount of agreement between weak labels, resulting in a slightly higher contribution of the third weak label $\boldsymbol{\lambda}_{*,3}$ across data points, even though it was actually mistaken here. In Active WeaSuL on the other hand, the penalty introduced in Equation 3 will actually decrease the overall contribution of $\boldsymbol{\lambda}_{*,3}$, exactly because it was mistaken. Hence, we argue that the approach of Nashaat et al. (2019) requires a large amount of active learning iterations because it relies on corrections that mostly affect individual data points, whereas Active WeaSuL requires fewer active learning iterations because it uses the expert-labelled data to learn how to best combine the weak labels.

Altogether, Active WeaSuL provides a means to make efficient use of domain expert knowledge: by combining expert-defined rules (weak supervision) with small amounts of expert-labelled data (active learning), we can train supervised models in applications where obtaining labelled data is inherently difficult.

ACKNOWLEDGEMENTS

We would like to thank the ING Wholesale Banking Advanced Analytics group, and in particular Ilan Fridman Rojas and Fabian Jansen, for their support and constructive feedback during this project.

REFERENCES

T. C. Alberto, J. V. Lochter, and T. A. Almeida. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 138–143, 2015. doi: 10.1109/ICMLA.2015.37.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. Learning from rules generalizing labeled exemplars. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020.

Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Chris Ré, and Rob Malkin. Snorkel Drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 362–375. Association for Computing Machinery, 2019. doi: 10.1145/3299869.3314036.

Sonia Badene, Kate Thompson, Jean-Pierre Lorré, and Nicholas Asher. Weak Supervision for Learning Discourse Structure. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2296–2305. Association for Computational Linguistics, 2019. doi: 10.18653/v1/d19-1234.

Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. Interactive Weak Supervision: Learning Useful Heuristics for Data Labeling. In *arXiv preprint arXiv:2012.06046v1*, 2020.

Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. Active and Incremental Learning with Weak Supervision. *KI - Künstliche Intelligenz*, 1:3, 2020. doi: 10.1007/s13218-020-00631-4.

Vincent Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christophe Re, and Li Fei Fei. Scene graph prediction with limited labels. In *Proceedings - 2019 International Conference on Computer Vision Workshop*, pp. 1772–1782, 2019. doi: 10.1109/ICCVW.2019.00220.

Benjamin Cohen-Wang, Stephen Mussmann, Alex Ratner, and Chris Ré. Interactive Programmatic Labeling for Weak Supervision. In *KDD Data Collection, Curation, and Labeling for Mining and Learning Workshop*, 2019.

Jared A Dunnmon, Alexander J Ratner, Khaled Saab, Nishith Khandwala, Matthew Markert, Hersh Sagreiya, Roger Goldman, Christopher Lee-Messer, Matthew P Lungren, Daniel L Rubin, and Christopher Ré. Cross-Modal Data Programming Enables Rapid Medical Machine Learning. *Patterns*, 1, 2020.

Julius Gonsior, Maik Thiele, and Wolfgang Lehner. WeakAL: Combining Active Learning and Weak Supervision. In *Discovery Science*, pp. 34–49. Springer International Publishing, 2020. doi: 10.1007/978-3-030-61527-7_3.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778. IEEE, 2016. doi: 10.1109/CVPR.2016.90.

Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, pp. 852–869, 2016. doi: 10.1007/978-3-319-46448-051.

Mona Nashaat, Aindrila Ghosh, James Miller, Shaikh Quader, Chad Marston, and Jean Francois Puget. Hybridization of Active Learning and Data Programming for Labeling Large Industrial Datasets. In *Proceedings - 2018 IEEE International Conference on Big Data*, pp. 46–55. IEEE, 2019. doi: 10.1109/BigData.2018.8622459.

Mona Nashaat, Aindrila Ghosh, James Miller, and Shaikh Quader. Asterisk: Generating Large Training Datasets with Automatic Active Supervision. *ACM/IMS Transactions on Data Science*, 1(2):1–25, 2020a. doi: 10.1145/3385188.

Mona Nashaat, Aindrila Ghosh, James Miller, and Shaikh Quader. Wesal: Applying active supervision to find high-quality labels at industrial scale. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020b.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014. doi: 10.3115/v1/d14-1162.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *Advances in Neural Information Processing Systems*, pp. 3574–3582, 2016.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, pp. 269. NIH Public Access, 2017.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training Complex Models with Multi-Task Weak Supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4763–4771, 2019.

Paroma Varma and Christopher Ré. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment*, volume 12, pp. 223–236. PVLDB, 2018. doi: 10.14778/3291264.3291268.

Paroma Varma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Ré. Learning dependency structures for weak supervision models. In *36th International Conference on Machine Learning*, pp. 11160–11170, 2019.

Zhi Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5: 44–53, 2018. ISSN 2053714X. doi: 10.1093/nsr/nwx106.

A APPENDIX

A.1 ACTIVE WEASUL ALGORITHM

Algorithm 1: Active WeaSuL

Input: Unlabelled data set $\mathbf{X}$, active learning budget B
Create labelling functions and apply them to the $\mathbf{X}$ to obtain the weak labels $\boldsymbol{\lambda}$;
Fit generative model (Equation 1) to get the initial parameters $\hat{\mathbf{z}}$;
Compute the probabilistic labels $\hat{p}_{t=0}(y|\boldsymbol{\lambda}) = f(\hat{\mathbf{z}}, \boldsymbol{\lambda})$;
for $t \leftarrow$ *1 to* B **do**
    Using Equations 5-7, determine the divergence $\text{KL}_{t,i}$ for all buckets i;
    Sample a data point from the bucket whose difference $\text{KL}_{t,i}$ is the largest (maxKL);
    Have an expert label the sampled data point, add it to the set of expert-labelled data D;
    Fit generative model again, now using Equation 3, resulting in an updated $\hat{\mathbf{z}}$;
    Compute $\hat{p}_t(y|\boldsymbol{\lambda}) = f(\hat{\mathbf{z}}, \boldsymbol{\lambda})$;
    Optionally, fit a discriminative model using feature matrix $\mathbf{X}$ and target $\hat{p}_t(y|\boldsymbol{\lambda})$
end
Output: Probabilistic labels $\hat{p}_t(y|\boldsymbol{\lambda})$

A.2 DISCRIMINATIVE MODEL

It is often possible to further improve on the generative model's predictive performance by training a discriminative model that predicts the probabilistic labels $\hat{p}_t(y|\boldsymbol{\lambda})$ from the feature matrix $\mathbf{X}$. This essentially allows one to define a more granular decision boundary (Figure 1D). To train the discriminative model on probabilistic labels, we use the formulation from Ratner et al. (2017).

We incorporate the information from active learning into the discriminative model in two ways. First, at each active learning iteration t, we train the discriminative model using the probabilistic labels from the generative model at iteration t. Second, for all expert-labelled data points in D, we use the binary label provided by the expert rather than the probabilistic label. Note that we do not use the discriminative model to determine which data points should be labelled next.

A.3 VISUAL RELATIONSHIP DETECTION

We have based our Visual Relationship Detection (VRD) experiment on one of the tutorials[3] in Snorkel, a popular Python package for weak supervision (Ratner et al., 2017). To perform this task with Active WeaSuL, we have made a few changes, which we describe here.

While the original VRD task is a multi-class experiment, where each class represents a relationship between a subject and an object (e.g. *sitting on, on top of, taller than*), we focus here on only one class in particular: is the subject sitting on top of an object? This turns the problem into a binary classification problem, which we can then tackle with Active WeaSuL. Since the VRD data contain quite a few duplicates (i.e. the same data point with different labels), we have removed all duplicates. We have then labelled a data point as $y = 1$ when one of its duplicates had the label *sitting on* and $y = 0$ otherwise. Finally, we have changed the labelling functions such that they are tailored towards the binary classification task (Supplementary Table 1). Note that the first two weak labels are conditionally dependent (as they are based on the same underlying data), and we incorporate this information in the generative model via Ω (Equation 1).

As in the Snorkel tutorial, we have used a neural network for our discriminative model (consisting of ResNet-18 (He et al., 2016) for the image data and GloVe (Pennington et al., 2014) for the categorical data).

To prevent overfitting, we split the training set such that 10% of the probabilistic labels are kept aside as a validation set, to be used for early stopping. We then train the model for a maximum of

[3] `https://github.com/snorkel-team/snorkel-tutorials/tree/master/visual_relation`

Table 1: Labelling functions for classifying *sitting on* for the VRD task.

	Description	Labelling function
1	relative bounding box sizes	`subject.area / object.area < 0.8`
2	distance between subject and object	`dist(subject, object) > 100`
3	object and subject category	`subject.type == 'person' AND object.type IS IN ['bench', 'chair', 'floor', 'horse', 'grass', 'table']`

100 epochs per active learning iteration, from which we choose the parameter setting corresponding to the minimal loss. The training is stopped when the minimal loss does not improve in five consecutive epochs, i.e. $patience = 5$. Note that we do not leak information in this way: we only use probabilistic labels to create the validation set, not the ground truth labels. Finally, note that the validation set is separate from the test set, hence our estimate of the predictive performance remains unbiased.

DEPENDENCY STRUCTURE MISSPECIFICATION IN MULTI-SOURCE WEAK SUPERVISION MODELS

Salva Rühling Cachay
Technical University of Darmstadt
salvaruehling@gmail.com

Benedikt Boecking
Carnegie Mellon University

Artur Dubrawski
Carnegie Mellon University

ABSTRACT

Data programming (DP) has proven to be an attractive alternative to costly hand-labeling of data. In DP, users encode domain knowledge into *labeling functions* (LF), heuristics that label a subset of the data noisily and may have complex dependencies. A label model is then fit to the LFs to produce an estimate of the unknown class label. The effects of label model misspecification on test set performance of a downstream classifier are understudied. This presents a serious awareness gap to practitioners, in particular since the dependency structure among LFs is frequently ignored in field applications of DP. We analyse modeling errors due to structure over-specification. We derive novel theoretical bounds on the modeling error and empirically show that this error can be substantial, even when modeling a seemingly sensible structure.

1 INTRODUCTION

Annotating large datasets for machine learning is expensive, time consuming, and a bottleneck for many practical applications of artificial intelligence. Recently, data programming, a paradigm that makes use of multiple weak supervision sources, has emerged as a promising alternative to manual data annotation (Ratner et al., 2016). In this framework, users encode domain knowledge into weak supervision sources, such as domain heuristics or knowledge bases, that each noisily label a subset of the data. A generative model over the sources and the latent true label is then learned. One can use the learned model to estimate *probabilistic* labels to train a *downstream model*, replacing the need to obtain ground truth labels by manual annotation of individual samples.
In practice, the sources of weak labels that users define often exhibit statistical dependencies amongst each other, e.g. sources operating on the same or similar input (some examples can be found in Table 1). Defining the correct dependency structure is difficult, thus a common approach in popular libraries (Bach et al., 2019; Ratner et al., 2019a) and related research (Dawid & Skene, 1979; Anandkumar et al., 2014; Varma & Ré, 2018; Boecking et al., 2021) is to ignore it. However, the implications of this assumption on downstream performance have not been researched in detail. Therefore, in this paper we take steps towards gaining a better understanding of the trade-offs involved.

Contributions We present novel bounds on the label model posterior and the downstream generalization risk that are explicitly influenced by misspecified higher-order dependencies. We also introduce three new higher-order dependency types, which we name *bolstering*, *negated*, and *priority* dependencies. Lastly, we empirically show that downstream test performance is highly sensitive to the user-specified dependencies, even when they make sense semantically. The finding suggests that in practice, it is advisable to only carefully model a few, if any, dependencies.

2 RELATED WORK

Data programming While the original data programming framework (Ratner et al., 2016) is based on a factor graph that support the modeling of arbitrary dependencies between LFs, more recent methods for solving for the parameters of the label model only support the modeling of pairwise correlations (Ratner et al., 2019a;b; Fu et al., 2020) — as such, losing some of the expressive power of data programming. The former extends data programming to the multi-task setting by exploiting the graph structure of the inverse covariance matrix among the sources (Ratner et al., 2019b) — in particular the fact that an entry is zero when there is no edge between the corresponding sources in the graphical model (Loh & Wainwright, 2012). The latter finds a closed-form solution for a class of

binary Ising models by using triplet methods (Fu et al., 2020). Our experimental findings suggest that practitioners may indeed benefit from simply ignoring higher-order dependencies.

Structure learning In order to automatically learn the structure between these sources, previous work optimizes the marginal pseudolikelihood of the noisy labels (Bach et al., 2017), or makes use of robust PCA to denoise the inverse covariance matrix of the sources labels into a graph structured term (Varma et al., 2019). A different approach, infers the structure through static analysis of the weak supervision sources code definitions and thereby reduces the sample complexity for learning the structure (Varma et al., 2017). In our experiments, we show that such methods should be carefully used, and may in fact lead to downstream performance losses.

Model misspecification On the side of work on model misspecification, (White, 1982) establishes that the Maximum Likelihood Estimator of a misspecified model is a consistent estimator of the learnable parameter that minimizes the KL divergence to the true distribution – if that optimal, misspecified parameter is globally identifiable. In an interesting result, (Jog & Loh, 2015) show that the KL divergence between a multivariate Gaussian distribution and a misspecified (by at least a single edge) Gaussian graphical model is bounded by a constant from below. It emphasizes the need to correctly select the model's edge structure, since otherwise the fitted distribution will differ from the true one in terms of KL divergence.

3 Problem setup

For this work we use the data programming framework as introduced in (Ratner et al., 2016), where the premise is that experts can model any higher-order dependency between labeling functions. We extend it by *negated*, *bolstering*, *priority* dependencies (definitions in the appendix B), e.g. the latter encoding the notion that one source's vote should be prioritized over the one from a noisier source. The additional dependency types are motivated by our selected LF sets that contain dependencies, such as the ones in Table 1, that we could not express before. Newer weak supervision models and model fitting approaches often only allow for pairwise correlation dependencies to be modeled (Ratner et al., 2019b; Fu et al., 2020).

Let $(x, y) \sim \mathcal{D}$ be the true data generating distribution and for simplicity assume that $y \in \{-1, 1\}$. As in (Ratner et al., 2016), users provide m labeling functions (LFs) $\lambda = \lambda(x) \in \{-1, 0, 1\}^m$, where 0 means that the LF abstained from labeling. Following (Ratner et al., 2016), we model the joint distribution of y, λ as a factor graph. To study model misspecification we compare two label models, p_θ for the conditional independent case and p_μ which models M higher-order dependencies:

$$p_\mu(\lambda, y) = \frac{1}{Z_\mu} \exp\left(\mu^T \phi(\lambda, y)\right) = Z_\mu^{-1} \exp\left(\mu_1^T \phi_1(\lambda, y) + \mu_2^T \phi_2(\lambda, y)\right), \qquad \mu \in \mathbb{R}^{m+M} \quad (1)$$

$$p_\theta(\lambda, y) = Z_\theta^{-1} \exp\left(\theta^T \phi_1(\lambda, y)\right), \qquad \theta \in \mathbb{R}^m, \quad (2)$$

where $\phi_1(\lambda, y) = \lambda y$ are the accuracy factors, $\phi_2(\cdot)$ are arbitrary, higher-order dependencies and $Z_\theta^{-1}, Z_\mu^{-1}$ are normalization constants. We assume w.l.o.g. that factors are bounded ≤ 1. Note that for ease of exposition, the models above do not model the labeling propensity factor (also known as LF coverage) . Since this factor does not depend on the label, the corresponding terms would cancel out in the quantities studied below (same goes for dependencies that do not depend on y, e.g. the *similar* factor from (Ratner et al., 2016)).

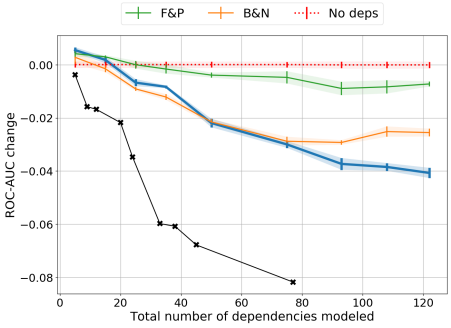
4 Theoretical analysis

Bound on the label model posterior under model misspecification We now state our bound on the probabilistic label difference (which we prove in the appendix A):

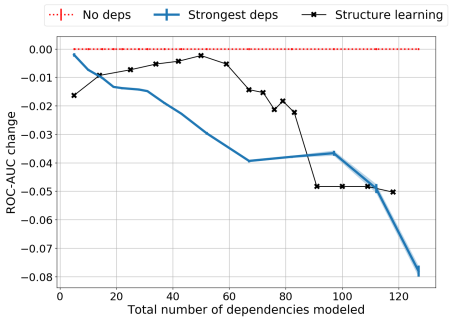
$$|p_\mu(y\ |\lambda) - p_\theta(y\ |\lambda)| \leq \frac{1}{2}||\mu_1 - \theta||_1 + \frac{1}{4}||\mu_2||_1 \quad (3)$$

This is an important quantity of interest since the probabilistic labels are used to train a downstream model. Unsurprisingly then, this quantity reappears as a main factor controlling the KL divergence and generalization risk, see below.

Suppose that the downstream model $f_w : \mathcal{X} \to \mathcal{Y}$ is parameterized by w, and that to learn w we minimize a, w.l.o.g., bounded noise-aware loss function $L(f_w(x), y) \in [0, 1]$ (that acts on the probabilistic labels). If we had access to the true labels, we would normally try to find the w that minimizes the risk: $w^* = \arg\min_w R(w) = \arg\min_w \mathbb{E}_{(x,y)\sim D}\left[L(f_w(x), y)\right]$. Since this is not the case, we instead will get the parameter $\hat{w}$ that minimizes the (empirical) noise-aware loss.



(a) IMDB (0.81)



(b) Bias Bios: professor or teacher (0.91)

Figure 1: Modeling more than a handful dependencies (as the ones in Table 1) significantly deteriorates ROC-AUC downstream performance as compared to simply ignoring them by up to 4 and 8 points. This effect intensifies as we model more dependencies. In brackets, the baseline score for the independent model ('No deps'). `B&N` means that we only model the *bolstering* and *negated* dependencies, while `F&P` means that we only model the *fixing* and *priority* dependencies. `Structure learning` means that we model the *similar*, *fixing* and *reinforcing* dependencies returned by the method from (Bach et al., 2017) for different threshold hyperparameters (the lower, the more dependencies are modeled).

Bound on the generalization risk under model misspecification While (Fu et al., 2020) provide a bound on the generalization error that accounts for model misspecification – the label model being a less expressive Ising model (i.e. no higher-order dependencies may be modeled) – the part of the bound corresponding to the misspecification is a somewhat loose KL-divergence term between the true and the misspecified models. If we instead assume that there exists a label model for some optimal parameterization of factors (not necessarily the ones that are actually modeled) that is equivalent to the true data generating distribution, we can provide a more meaningful and tight bound.

As in (Ratner et al., 2016; 2019b), assume that 1) there exists an optimal parameter of either p_μ or p_θ (say, μ^* with label model p_{μ^*}) such that sampling (λ, y) from this optimal label model is equivalent to $(\lambda, y) \sim \mathcal{D}$; and 2) the label y is independent of the features used for training f_w given the labeling function outputs λ. Differently than (Ratner et al., 2016; 2019b), the factors that are actually modeled may differ from the ones of the optimal label model. By adapting the proof of theorem 1 in (Ratner et al., 2019b) to our case where we incorrectly use a misspecified model (say, p_θ), we can bound the generalization risk as follows

$$R(\hat{w}) - R(w^*) \leq \gamma + 2||\mu_1^* - \theta||_1 + ||\mu_2^*||_1, \tag{4}$$

where γ is a bound on the empirical risk minimization error, which is not specific to our setting. We can bound the KL divergence of the two models by a similar quantity, see the appendix A.3.

Interpretation These bounds naturally involve the accuracy parameter estimation error $||\mu_1 - \theta||_1$ and the learned strength $s := ||\mu_2||_1$ of the dependencies only modeled in p_μ. Note that if we assume that the model with dependencies p_μ is the true one we can interpret s as the magnitude of the dependencies that the independent model p_θ failed to model. If on the other hand we associate the misspecified model with p_μ, then we can interpret this quantity s as the (dependency) parameter excess that was incorrectly learned by p_μ. The presented bounds are tighter than the ones from (Ratner et al., 2019b) for the L1 norm, while in addition accounting for model misspecification.

5 Experiments

5.1 Proxy for finding true dependencies in real datasets

The underlying *true* structure between two real labeling functions λ_j, λ_k is, of course, unknown. However, by using true training labels (solely for this purpose) together with the observed LF votes, we can compute resulting factor values for each data point i, to observe empirical strength of dependency factor l over a training set: $v_{j,k}^l = \sum_i \phi^l(\lambda(x_i)_j, \lambda(x_i)_k, y_i)$. Sorting dependencies l

Table 1: The strongest and weakest dependencies among the IMDB LFs, as measured by their factor value $v^l_{j,k}$ computed with respect to the true labels. In the experiment from Fig.1, we repeatedly add weaker dependencies to the set of dependencies used for learning the label model, starting with the strongest ones below.

LF_j	LF_k	factor type l	factor value $v^l_{j,k}$
best	great	bolstering	801
bad	don't waste	bolstering	110
original	bad	priority	327
recommend	terrible	priority	53
worth	not worth	fixing	238
great	nothing great	fixing	15
worth	not worth	negated	219
special	not special	negated	8
recommend	highly recommend	reinforcing	226
bad	absolutely horrible	reinforcing	7

according to $v^l_{j,k}$ in descending order, we then choose to model the top d dependencies. These are the dependencies for which the true labels provide the most evidence of being correct.

5.2 Performance deterioration due to structure over-specification

For the following experiment we use the IMDB Movie Review Sentiment dataset consisting of $n = 25$k training and test samples each (Maas et al., 2011) and manually select a set of $m = 135$ sensible LFs that label on the presence of a single word or a pair of words (i.e. uni-/bi-gram LFs). In addition we use the Bias in Bios dataset (De-Arteaga et al., 2019), and focus on the binary classification task introduced in (Boecking et al., 2021) that aims at distinguishing the biographies of professors and teachers ($n = 12294, m = 85$). We deliberately choose unigram and bigram LFs so as to create dependencies we expect to help with downstream model performance, e.g. by adding negations (e.g. "not worth") of unigrams (e.g. "worth") that we expect to fix the less precise votes of the latter when both do not abstain.

We choose different $d \in \{1, 3, 5, \ldots, 40\}$ and then model the strongest $\leq d$ dependencies of each factor l according to v^l. An example of the strongest and weakest dependencies for the IMDB dataset is shown in Table 1. For the Bias in Bios experiment, an example of a strong bolstering dependency is that the term 'PhD' appears in addition to the term 'university'. We report the test set performance of a simple 3-layer neural network trained on the probabilistic labels, averaged out over 100 runs. While for IMDB we observe a marginal boost (< 0.005) in performance when modeling the strongest $d = 1, 3$ dependencies of each factor ($5, 15$ in total), the main take-away is the following:

We find that modeling more than a handful of dependencies significantly *deteriorates* the downstream end classifier performance (by up to 8 ROC-AUC points) as compared to simply ignoring them (Fig. 1). The performance worsens as we increase d, i.e. as we model more, slightly weaker, dependencies. We reiterate that these additional dependencies still, semantically, make sense (as depicted in Table 1, where the weakest ones are modeled only for the case where the total number of dependencies $= 122$). Using the structure learning method from (Bach et al., 2017) to infer the dependency structure results in worse test performances too.

6 Discussion and Future Work

Discussion Even though this result and insight is highly relevant for practitioners, it has, to the best of our knowledge, not been explored in detail. It may come as a surprise that modeling seemingly sensible dependencies can significantly deteriorate the targeted downstream model performance. We hypothesize that this is due to the true model being close to the conditionally independent case and in our presented bounds, we indeed see that they become looser as more incorrect dependencies are modeled. In addition, more complex models often suffer of a higher sample complexity, as is also briefly noted in (Fu et al., 2020). This suggests that practitioners may 1) indeed be best served, at first, by simply ignoring (higher-order) dependencies; and 2) need to be careful when specifying

dependencies, either by hand or through structure learning algorithms, which emphasizes the need for a small ground-truth labeled validation set to compare the performance of different label models.

Future work Future work should give a theoretically precise answer to the reason for why performance deterioration is observed, conduct more extensive experiments to validate that this holds for a variety of datasets and labeling function sets, as well as better characterize the settings in which structure learning actually helps with downstream performance.

REFERENCES

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15: 2773–2832, 2014.

Stephen H. Bach, Bryan He, Alexander Ratner, and Christopher Ré. Learning the structure of generative models without labeled data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 273–282, 2017.

Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Chris Ré, and Rob Malkin. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pp. 362–375, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450356435. doi: 10.1145/3299869.3314036.

Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. Interactive weak supervision: Learning useful heuristics for data labeling. *International Conference on Learning Representations (ICLR)*, 2021.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979. ISSN 00359254, 14679876.

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 120–128, 2019.

Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. *ICML*, 2020.

Jean Honorio. Lipschitz parametrization of probabilistic graphical models. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, pp. 347–354, Arlington, Virginia, USA, 2011. AUAI Press. ISBN 9780974903972.

Varun Jog and Po-Ling Loh. On model misspecification and kl separation for gaussian graphical models. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 1174–1178. IEEE, 2015.

Po-ling Loh and Martin J Wainwright. Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. In *Advances in Neural Information Processing Systems 25*, pp. 2087–2095. Curran Associates, Inc., 2012.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29, 05 2016.

Alexander Ratner, Stephen Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: rapid training data creation with weak supervision. *The VLDB Journal*, 29, 07 2019a. doi: 10.1007/s00778-019-00552-1.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training complex models with multi-task weak supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4763–4771, 07 2019b. doi: 10.1609/aaai.v33i01.33014763.

Paroma Varma and Christopher Ré. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 12, pp. 223. NIH Public Access, 2018.

Paroma Varma, Bryan D He, Payal Bajaj, Nishith Khandwala, Imon Banerjee, Daniel Rubin, and Christopher Ré. Inferring generative model structure with static analysis. In *Advances in neural information processing systems*, pp. 240–250, 2017.

Paroma. Varma, Frederic Sala, Ann He, Alexander Ratner, and Christophe Ré. Learning dependency structures for weak supervision models. *ICML*, 2019.

Halbert White. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the Econometric Society*, pp. 1–25, 1982.

A PROOFS OF THE THEORETICAL ANALYSIS

A.1 PROBLEM SETUP RECAP

Let $(x, y) \sim \mathcal{D}$ be the true data generating distribution and for simplicity assume that $y \in \mathcal{Y} = \{-1, 1\}$. Users provide m labeling functions (LFs) $\lambda = \lambda(x) \in \{-1, 0, 1\}^m$, where 0 means that the LF abstained from labeling. We compare two label models, p_θ for the conditional independent case, and p_μ which models higher-order dependencies:

$$p_\mu(\lambda, y) = \frac{1}{Z_\mu} \exp\left(\mu^T \phi(\lambda, y)\right) = Z_\mu^{-1} \exp\left(\mu_1^T \phi_1(\lambda, y) + \mu_2^T \phi_2(\lambda, y)\right), \qquad \mu \in \mathbb{R}^{m+M} \tag{5}$$

$$p_\theta(\lambda, y) = Z_\theta^{-1} \exp\left(\theta^T \phi_1(\lambda, y)\right), \qquad \theta \in \mathbb{R}^m, \tag{6}$$

where $\phi_1(\lambda, y) = \lambda y$ are the accuracy factors, $\phi_2(\cdot)$ are arbitrary, higher-order dependencies and $Z_\theta^{-1}, Z_\mu^{-1}$ are normalization constants. We assume w.l.o.g. that factors are bounded ≤ 1. Using an unlabeled dataset $X = \{x_i\}_{i=1}^n$ of n data points $x_i \in \mathcal{X}$ to which we each apply the m user provided labeling functions, we attain the label matrix $\Lambda \in \{-1, 0, 1\}^{n \times m}$. With Λ we then train the label model to get a set of n probabilistic labels with which we supervise the downstream model $f_w : \mathcal{X} \to \mathcal{Y}$.

Lemma 1 (Sigmoid posterior). With $\sigma(x) = \frac{1}{1+\exp(-x)}$ being the sigmoid function, it holds that

$$p_\mu(y \,|\lambda) = \sigma\left(2\mu_1^T \phi_1(\lambda, y) + \mu_2^T \left(\phi_2(\lambda, y) - \phi_2(\lambda, -y)\right)\right) \tag{7}$$

$$p_\theta(y \,|\lambda) = \sigma\left(2\theta^T \phi_1(\lambda, y)\right). \tag{8}$$

Proof

$$\begin{aligned}
p_\mu(y \,|\lambda) &= \frac{p_\mu(\lambda, y)}{p_\mu(\lambda)} = \frac{p_\mu(\lambda, y)}{\sum_{\tilde{y} \in \mathcal{Y}} p_\mu(\lambda, \tilde{y})} \\
&= \frac{Z_\mu^{-1} \exp\left(\mu^T \phi(\lambda, y)\right)}{\sum_{\tilde{y} \in \mathcal{Y}} Z_\mu^{-1} \exp\left(\mu^T \phi(\lambda, \tilde{y})\right)} \\
&= \frac{\exp\left(\mu^T \phi(\lambda, y)\right)}{\sum_{\tilde{y} \in \mathcal{Y}} \exp\left(\mu^T \phi(\lambda, \tilde{y})\right)} \\
&= \frac{\exp\left(\mu^T \phi(\lambda, y)\right)}{\exp\left(\mu^T \phi(\lambda, y)\right) + \exp\left(\mu^T \phi(\lambda, -y)\right)} \\
&= \frac{1}{1 + \exp\left(\mu^T \left(\phi(\lambda, -y) - \phi(\lambda, y)\right)\right)} \\
&= \sigma\left(\mu^T \left(\phi(\lambda, y) - \phi(\lambda, -y)\right)\right) \\
&= \sigma\left(2\mu_1^T \phi_1(\lambda, y) + \mu_2^T \left(\phi_2(\lambda, y) - \phi_2(\lambda, -y)\right)\right),
\end{aligned}$$

where in the last step we used the fact that the accuracy factors are odd functions, i.e. $\phi_1(\lambda, -y) = -\lambda y = -\phi_1(\lambda, y)$. Eq. 8 follows by the same argumentation.

A.2 PROOF OF THE BOUND ON THE LABEL MODEL POSTERIOR

Bound Our bound on the probabilistic label difference between the two models above is:

$$|p_\mu(y\,|\lambda) - p_\theta(y\,|\lambda)| \leq \frac{1}{2}||\mu_1 - \theta||_1 + \frac{1}{4}||\mu_2||_1 \tag{9}$$

Proof Using Lemma 1 we have that

$$|p_\mu(y\,|\lambda) - p_\theta(y\,|\lambda)| = \left|\sigma\left(2\mu_1^T\phi_1(\lambda,y) + \mu_2^T\left(\phi_2(\lambda,y) - \phi_2(\lambda,-y)\right)\right) - \sigma\left(2\theta^T\phi_1(\lambda,y)\right)\right|$$

By the mean value theorem it follows that for some c between the arguments of σ above

$$\begin{aligned}
&= \sigma'(c)\left|\left(2\mu_1^T\phi_1(\lambda,y) + \mu_2^T\left(\phi_2(\lambda,y) - \phi_2(\lambda,-y)\right)\right) - 2\theta^T\phi_1(\lambda,y)\right| \\
&= \sigma'(c)\left|2\left(\mu_1-\theta\right)^T\phi_1(\lambda,y) + \mu_2^T\left(\phi_2(\lambda,y) - \phi_2(\lambda,-y)\right)\right|
\end{aligned}$$

Using the triangle inequality and the fact that $\max_x \sigma'(x) = \max_x \sigma(x)(1-\sigma(x)) = \frac{1}{4}$, we can now bound this expression as follows

$$\leq \frac{1}{2}\left|\left(\mu_1-\theta\right)^T\phi_1(\lambda,y)\right| + \frac{1}{4}\left|\mu_2^T\left(\phi_2(\lambda,y) - \phi_2(\lambda,-y)\right)\right|$$

finally, since the defined higher-order dependencies are indicator functions $\neq 0$ for only one $y \in \mathcal{Y}$, and if $||q||_\infty \leq 1$ then $|x^Tq| = |\sum_i x_i q_i| \leq \sum_i |x_i q_i| \leq \sum_i |x_i| = ||x||_1$, this reduces to

$$\leq \frac{1}{2}||\mu_1 - \theta||_1 + \frac{1}{4}||\mu_2||_1.$$

A.3 PROOF OF THE BOUND ON THE KL DIVERGENCE

Bound

$$KL\left(p_\mu(y\,|\lambda)\;||\;p_\theta(y\,|\lambda)\right) \leq 2||\mu_1 - \theta||_1 + ||\mu_2||_1 \tag{10}$$

Proof We adapt Theorem 7 of (Honorio, 2011) to give a bound on the KL divergence between the two label model posterior's. First note that with the same line of argumentation as in A.2 we have that

$$\begin{aligned}
|\log p_\mu(y\,|\lambda) - \log p_\theta(y\,|\lambda)| &= (\log\sigma)'(c)\left|2\left(\mu_1-\theta\right)^T\phi_1(\lambda,y) + \mu_2^T\left(\phi_2(\lambda,y) - \phi_2(\lambda,-y)\right)\right| \\
&\leq 2||\mu_1 - \theta||_1 + ||\mu_2||_1,
\end{aligned}$$

where we use that the derivative of $\log\sigma(\cdot)$ is $(1+\exp(x))^{-1} \in (0,1)$, and is bounded by 1. Next

$$\begin{aligned}
KL\left(p_\mu(y\,|\lambda)\;||\;p_\theta(y\,|\lambda)\right) &= \sum_{\lambda\in L} p_\mu(\lambda)\sum_{y\in\mathcal{Y}} p_\mu(y|\lambda)\cdot\log\left(\frac{p_\mu(y|\lambda)}{p_\theta(y|\lambda)}\right) \\
&= \sum_{\lambda\in L} p_\mu(\lambda)\sum_{y\in\mathcal{Y}} \frac{p_\mu(\lambda,y)}{p_\mu(\lambda)}\cdot\log\left(\frac{p_\mu(y|\lambda)}{p_\theta(y|\lambda)}\right) \\
&= \sum_{\lambda\in L}\sum_{y\in\mathcal{Y}} p_\mu(\lambda,y)\cdot\left(\log p_\mu(y|\lambda) - \log p_\theta(y|\lambda)\right) \\
&\leq \sum_{\lambda\in L}\sum_{y\in\mathcal{Y}} p_\mu(\lambda,y)\cdot\left|\log p_\mu(y|\lambda) - \log p_\theta(y|\lambda)\right| \\
&\leq \left(2||\mu_1-\theta||_1 + ||\mu_2||_1\right)\sum_{\lambda\in L}\sum_{y\in\mathcal{Y}} p_\mu(\lambda,y) \\
&= 2||\mu_1-\theta||_1 + ||\mu_2||_1
\end{aligned}$$

A.4 PROOF OF THE GENERALIZATION RISK BOUND

We now adapt Theorem 1 from (Ratner et al., 2019b) to the setting with model misspecification and assume like in (Ratner et al., 2016; 2019b) that

1. there exists an optimal parameter of either p_μ or p_θ such that sampling (λ, y) from this optimal label model is equivalent to $(\lambda, y) \sim \mathcal{D}$.
2. the label y is independent of the features used for training f_w given the labeling function outputs λ, i.e. the LF labels provide sufficient signal to identify the label.

For 1. we now assume without loss of generality, that $p_{\mu^*}(\lambda, y) = p_{\mathcal{D}}(\lambda, y)$ for an optimal parameter $\mu^* \in \mathbb{R}^{m+M}$, and that we incorrectly use the misspecified label model p_θ. For the reversed roles (i.e. p_θ is the true model and p_μ is misspecified), the following arguments are symmetric.
Suppose that the downstream model $f_w : \mathcal{X} \to \mathcal{Y}$ is parameterized by w, and that to learn w we minimize a, w.l.o.g., bounded loss function $L(f_w(x), y) \in [0, 1]$. If we had access to the true labels, we would normally try to find the w that minimizes the risk:

$$w^* = \arg\min_w R(w) = \arg\min_w \mathbb{E}_{(x,y)\sim D} \left[L(f_w(x), y) \right]. \tag{11}$$

Since this is not the case, we instead minimize the noise-aware loss:

$$\tilde{w} = \arg\min_w R_\theta(w) = \arg\min_w \mathbb{E}_{(x,y)\sim \mathcal{D}} \left[\mathbb{E}_{\tilde{y}\sim p_\theta(\cdot|\lambda)} \left[L(f_w(x), \tilde{y}) \right] \right]. \tag{12}$$

In practice we will get the parameter $\hat{w}$ that minimizes the empirical noise-aware loss over the unlabeled dataset $X = \{x_1, ..., x_n\}$:

$$\hat{w} = \arg\min_w \hat{R}_\theta(w) = \arg\min_w \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\tilde{y}\sim p_\theta(\cdot|\lambda(x_i))} \left[L(f_w(x_i), \tilde{y}) \right]. \tag{13}$$

Since the empirical risk minimization error is not specific to our setting and can be done with standard methods, we simply assume that the error $|R_\theta(\tilde{w}) - R_\theta(\hat{w})| \leq \gamma(n)$, where $\gamma(n)$ is a decreasing function of the unlabeled dataset size n.

Bound By adapting the proof of theorem 1 in (Ratner et al., 2019b) to our case with model misspecification involved, we can bound the generalization risk as follows

$$R(\hat{w}) - R(w^*) \leq \gamma(n) + 2||\mu_1^* - \theta||_1 + ||\mu_2^*||_1. \tag{14}$$

Note that the bound from (Ratner et al., 2019b) is mistakenly too tight by a factor of 2 (due to the last step in the proof below that is partly overseen).

Proof First, using the law of total expectation, followed by our assumptions 2. and 1., in that order, we have that:

$$\begin{aligned}
R(w) &= \mathbb{E}_{(x',y')\sim D} \left[R(w) \right] \\
&= \mathbb{E}_{(x',y')\sim D} \left[\mathbb{E}_{(x,y)\sim D} \left[L(f_w(x'), y) | x = x' \right] \right] \\
&= \mathbb{E}_{(x',y')\sim D} \left[\mathbb{E}_{(x,y)\sim D} \left[L(f_w(x'), y) | \lambda(x) = \lambda(x') \right] \right] \\
&= \mathbb{E}_{(x',y')\sim D} \left[\mathbb{E}_{\tilde{y}\sim p_{\mu^*}(\cdot|\lambda)} \left[L(f_w(x'), \tilde{y}) \right] \right] \\
&= R_{\mu^*}(w)
\end{aligned}$$

Using the result above that $R = R_{\mu^*}$ and adding and subtracting terms we have:

$$\begin{aligned}
R(\hat{w}) - R(w^*) &= R_{\mu^*}(\hat{w}) - R_{\mu^*}(w^*) \\
&= R_{\mu^*}(\hat{w}) + R_\theta(\hat{w}) - R_\theta(\hat{w}) + R_\theta(\tilde{w}) - R_\theta(\tilde{w}) - R_{\mu^*}(w^*)
\end{aligned}$$

since $\tilde{w}$ minimizes the noise-aware risk, i.e. $R_\theta(\tilde{w}) \leq R_\theta(w^*)$, we have that:

$$\begin{aligned}
&\leq R_{\mu^*}(\hat{w}) + R_\theta(\hat{w}) - R_\theta(\hat{w}) + R_\theta(w^*) - R_\theta(\tilde{w}) - R_{\mu^*}(w^*) \\
&\leq |R_\theta(\hat{w}) - R_\theta(\tilde{w})| + |R_{\mu^*}(\hat{w}) - R_\theta(\hat{w})| + |R_\theta(w^*) - R_{\mu^*}(w^*)| \\
&\leq \gamma(n) + 2 \max_{w'} |R_{\mu^*}(w') - R_\theta(w')|
\end{aligned}$$

The main term $|R_{\mu^*}(w') - R_\theta(w')|$ we now have in the bound, is the difference between the true/expected noise-aware losses given the true label model parameter μ^* and the estimated parameter

θ for the misspecified model. We now bound this quantity:

$$\begin{aligned}
|R_{\mu^*}(w') - R_\theta(w')| &= \left|\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{\tilde{y}\sim p_{\mu^*}(\cdot|\lambda)}\left[L(f_w(x),\tilde{y})\right] - \mathbb{E}_{\tilde{y}\sim p_\theta(\cdot|\lambda)}\left[L(f_w(x),\tilde{y})\right]\right]\right| \\
&= \left|\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\sum_{y'\in\mathcal{Y}} L(f_w(x),y')\left(p_{\mu^*}(y'\,|\lambda) - p_\theta(y'\,|\lambda)\right)\right]\right| \\
&\leq \sum_{y'\in\mathcal{Y}} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[|L(f_w(x),y')\left(p_{\mu^*}(y'\,|\lambda) - p_\theta(y'\,|\lambda)\right)|\right] \\
&\leq \sum_{y'\in\mathcal{Y}} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[|p_{\mu^*}(y'\,|\lambda) - p_\theta(y'\,|\lambda)|\right] \\
&\leq |\mathcal{Y}| \max_{y'} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[|p_{\mu^*}(y'\,|\lambda) - p_\theta(y'\,|\lambda)|\right]
\end{aligned}$$

We can now use our bound from (3) and get that:

$$\leq 2\left(\frac{1}{2}||\mu_1^* - \theta||_1 + \frac{1}{4}||\mu_2^*||_1\right) = ||\mu_1^* - \theta||_1 + \frac{1}{2}||\mu_2^*||_1$$

Plugging this back into the term for the generalization risk gives the desired result:

$$\begin{aligned}
R(\hat{w}) - R(w^*) &\leq \gamma(n) + 2\max_{w'}|R_{\mu^*}(w') - R_\theta(w')| \\
&\leq \gamma(n) + 2||\mu_1^* - \theta||_1 + ||\mu_2^*||_1.
\end{aligned}$$

B Factor Definitions

We supplement the factor definitions of higher-order dependencies used in this paper. The first two stem from (Ratner et al., 2016), the rest we defined ourselves for the conducted experiments, and where motivated by frequently occurring dependency patterns, as the ones in Table 1, that are not covered by (Ratner et al., 2016). Whenever a factor $\phi_{j,k}(\lambda, y)$ is not symmetric (all factors, besides *bolstering*), we define it so that LF_k acts on (e.g. *negates*) LF_j.
For the *fixing* dependency we have:

$$\phi_{j,k}^{Fix}(\lambda, y) = \begin{cases} +1 & \text{if } \lambda_j = -y \wedge \lambda_k = y \\ -1 & \text{if } \lambda_j = 0 \wedge \lambda_k \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

for the *reinforcing* one:

$$\phi_{j,k}^{Rei}(\lambda, y) = \begin{cases} +1 & \text{if } \lambda_j = \lambda_k = y \\ -1 & \text{if } \lambda_j = 0 \wedge \lambda_k \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

for the *priority* factor:

$$\phi_{j,k}^{Pri}(\lambda, y) = \begin{cases} +1 & \text{if } \lambda_j = -y \wedge \lambda_k = y \\ -1 & \text{if } \lambda_j = y \wedge \lambda_k = -y \\ 0 & \text{otherwise} \end{cases}$$

for the *bolstering*:

$$\phi_{j,k}^{Bol}(\lambda, y) = \begin{cases} +1 & \text{if } \lambda_j = \lambda_k = y \\ -1 & \text{if } \lambda_j = \lambda_k \neq y \vee \lambda_j = -\lambda_k \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

and, finally, for the *negated* factor:

$$\phi_{j,k}^{Neg}(\lambda, y) = \begin{cases} +1 & \text{if } \lambda_j = -y \wedge \lambda_k = y \\ -1 & \text{if } (\lambda_j = y \wedge \lambda_k = -y) \vee \lambda_j = \lambda_k \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

WEAKLY-SUPERVISED GROUP DISENTANGLEMENT USING TOTAL CORRELATION

Linh Tran[1,2,*], **Saeid Asgari Taghanaki** [1], **Amir Hosein Khasahmadi**[1], **Aditya Sanghi**[1]

[1]*Autodesk AI Lab*
[2]*Imperial College London*

ABSTRACT

Learning disentangled representations that uncover factors of variation in data remains an ongoing key challenge in representation learning. Recent concerns about the feasibility of learning disentangled representations in an unsupervised fashion have motivated a shift toward weak supervision. One way to incorporate weak supervision is through *match pairing*, i.e., using observations as pairs that share at least one factor of variation. Existing match pairing approaches only consider the structural constraints with an average approximate posterior over observations of a shared group. We show the limitations of these approaches and propose a novel formulation to enforce disentangled representations of groups through total correlation, which improves overall disentanglement on various image datasets.

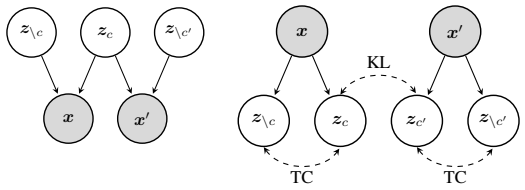
1 INTRODUCTION

Decomposing data into disjoint independent factors of variations and thus learning disentangled representations is essential for interpretable and controllable machine learning Shu et al. (2019). Recent works have shown the usefulness of disentangled representation with respect to abstract reasoning (van Steenkiste et al. (2019)), fairness (Locatello et al. (2019a); Creager et al. (2019)), reinforcement learning (Higgins et al. (2017b)) and general predictive performance (Locatello et al. (2019b)). Even though unsupervised disentanglement methods (Higgins et al. (2017a); Kim & Mnih (2018); Chen et al. (2018)) have shown promising results to learn disentangled representations, Locatello et al. (2019b) showed in a rigorous study that it is impossible to disentangle variations of data without any supervision or inductive bias. Since then, there has been a shift toward weakly supervised disentanglement learning Locatello et al. (2019b), Shu et al. (2019) such as *match pairing* (Shu et al. (2019)) which uses paired observations during optimization. In this work, we present a framework to learn group-disentangled representations using total correlation in a weakly-supervised setting. Our work can be considered learning different levels of weakly-supervised group disentanglement with total correlation. Closely related work is the one of Creager et al. (2019) which proposed to minimize the mutual information between the sensitive latent variable and sensitive labels. Similarly, Klys et al. (2018) proposed to minimize mutual information between the latent variable and a conditional subspace. Both works require either supervised labels or conditions to estimate the mutual information, whereas we only use *weak* supervision for learning disentangled group representations. Locatello et al. (2020) proposed to disentangle groups of variations with only knowing the number of common groups which can be considered as a complementary component to our method. We show that our approach can flexibly disentangle between and within groups of factors of variation. Further, we demonstrate that we improve on disentanglement for various image datasets.

In summary, we make the following contributions:

1. We show limitations of existing group disentanglement approaches (Bouchacourt et al. (2018) and Hosoya (2019)) in terms of latent variable collapse and batch size sensitivity and propose a weakly-supervised way for addressing these weaknesses.
2. We propose a new way of learning disentangled representations from paired observations using total correlation. We also show how to enforce different levels of inter-group and intra-group disentanglement through total correlation.

[*]Corresponding author: linh.tran@autodesk.com



(a) Generative (b) Inference

Figure 1: **The proposed generative and inference model.** Shaded nodes denote observed quantities, and unshaded nodes represent unobserved (latent) variables. Dotted arrows represent either minimizing the TC or the KL divergence between variables.

2 Background

VAEs are latent variable models and aim to learn latent variables $\boldsymbol{z}$ which should capture information about the observations $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$. They are trained to maximize the evidence lower bound (ELBO) given as $\log p(\boldsymbol{x}) \geq \mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))$. To be consistent with the works of Bouchacourt et al. (2018) and Hosoya (2019), let us assume that the observations $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ are collected in $\mathcal{G}$ distinct groups. Within a group, all observations share some *fixed* factors of variations. Each group $g \in \mathcal{G}$ splits $\mathcal{X}$ into disjoint partitions with arbitrary sizes. For simplicity, we define two groups g_C and $g_{\setminus C}$; where g_C represents information about the actual content whereas $g_{\setminus C}$ represents any variation not contained in C. Given a pair of observations $(\boldsymbol{x}, \boldsymbol{x}')$ which share group factors c, we define two variables $\boldsymbol{z} = (\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c})$ and $\boldsymbol{z}' = (\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c})$ to capture content $(\boldsymbol{z}_c, \boldsymbol{z}_{c'})$ and non-content information $(\boldsymbol{z}_{\setminus c}, \boldsymbol{z}_{\setminus c'})$, e.g. style or background. (Bouchacourt et al. (2018); Hosoya (2019)) learn a group-specific latent variable $\bar{\boldsymbol{z}}_{c,c'}$ by averaging over the corresponding content latent variable $\boldsymbol{z}_c, \boldsymbol{z}_{c'}$ during inference. The modified objective considers the ELBO of paired observations $(\boldsymbol{x}, \boldsymbol{x}')$ i.i.d. sampled from group g_C

$$\begin{aligned}\mathcal{L}_{\text{WS-ELBO}}(\boldsymbol{x}, \boldsymbol{x}'; \theta, \phi) = \mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{x}|\bar{\boldsymbol{z}}_{c,c'}, \boldsymbol{z}_{\setminus c})] + \mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{x}'|\bar{\boldsymbol{z}}_{c,c'}, \boldsymbol{z}_{\setminus c'})] \\ - \beta \mathrm{D}_{\mathrm{KL}}(q_\phi(\bar{\boldsymbol{z}}_{c,c'}, \boldsymbol{z}_{\setminus c}|\boldsymbol{x}) \parallel p(\boldsymbol{z})) - \beta \mathrm{D}_{\mathrm{KL}}(q_\phi(\bar{\boldsymbol{z}}_{c,c'}, \boldsymbol{z}_{\setminus c'}|\boldsymbol{x}') \parallel p(\boldsymbol{z})),\end{aligned} \tag{1}$$

where $\bar{\boldsymbol{z}}_{c,c'}$ is sampled from either a Normal distribution over the average of learned means and covariances (Hosoya (2019)) or a product of Normal distributions (Bouchacourt et al. (2018)).

3 Learning Group Similarities using Total Correlation

Considering the setting in Section 2, we would like 1) z_c to be highly correlated with group C and $z_{\setminus c}$ to be highly correlated with group $\setminus C$ and 2) $\boldsymbol{z}_c \approx \boldsymbol{z}_{c'}$ if the paired observations share the same content c or $\boldsymbol{z}_{\setminus c} \approx \boldsymbol{z}_{\setminus c'}$ if the paired observations share the same non-content $\setminus c$. In what follows, we describe our approach with the generative and inference model visualized in Figure 1. Although existing works showed promising results, in practice, minimizing the objective in (1) will not necessarily fulfill the first requirement, i.e., the model will learn representations z_c and $z_{\setminus c}$ that are uncorrelated with each other. Therefore, along with maximizing the variational lower bound, we propose to minimize the total correlation between latent variables z_c and $z_{\setminus c}$.

The total correlation of $\boldsymbol{z}_c$ and $\boldsymbol{z}_{\setminus c}$ is defined as

$$\mathrm{TC}(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c}) = \mathrm{D}_{\mathrm{KL}}(q(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c}) \parallel \bar{q}(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c})) \tag{2}$$

where $\bar{q}(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c})$ denotes the desired factorization of the aggregated posterior $q(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c})$. With different factorization, we can enforce different levels of disentanglement:

1. Inter-group disentanglement: $[\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c}]$ is said to be disentangled if its aggregate posterior factorizes as $\bar{q}(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c}) = q(\boldsymbol{z}_c) \cdot q(\boldsymbol{z}_{\setminus c})$. Note that under this disentanglement criteria, each $\boldsymbol{z}_c$ and $\boldsymbol{z}_{\setminus c}$ can be correlated among themselves. However, they must be independent of each other.

(a) Inter-group disentanglement ($q(z_c) \cdot q(z_{\setminus c})$) (b) Inter- and intra-group disentanglement ($q(z_c) \cdot \prod_j q(z_{\setminus c,i})$) (c) Total disentanglement ($\prod_i q(z_{c,j}) \cdot \prod_j q(z_{\setminus c,i})$)

Figure 2: **Different factorizations encourage different levels of disentanglement.** The latent variable $\boldsymbol{z}_c$ representing content information is visualized as red tiles whereas $\boldsymbol{z}_{\setminus c}$ representing non-content information is visualized as blue tiles. The different shades of colors represents correlation to different factors of variation.

2. Inter-group disentanglement and intra-group disentanglement of one group: $[z_c, \boldsymbol{z}_{\setminus c}]$ and $\boldsymbol{z}_{\setminus c}$ are disentangled if the aggregate posterior factorizes as $q(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c}) = q(\boldsymbol{z}_c) \cdot \prod_j q(\boldsymbol{z}_{\setminus c,i})$. With this criteria, $\boldsymbol{z}_c$ is still free to co-vary together, but must be independent from all $\boldsymbol{z}_{\setminus c,i}$. Further each dimension of $\boldsymbol{z}_{\setminus c}$ is disentangled. This kind of disentanglement was also used by Creager et al. (2019).
3. Inter-group disentanglement and intra-group disentanglement of all groups: We can enforce total disentanglement if the aggregate posterior factorizes as $q(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c}) = \prod_i q(\boldsymbol{z}_{c,j}) \cdot \prod_j q(\boldsymbol{z}_{s,i})$. This is equivalent to disentanglement achieved in the FactorVAE objective Kim & Mnih (2018).

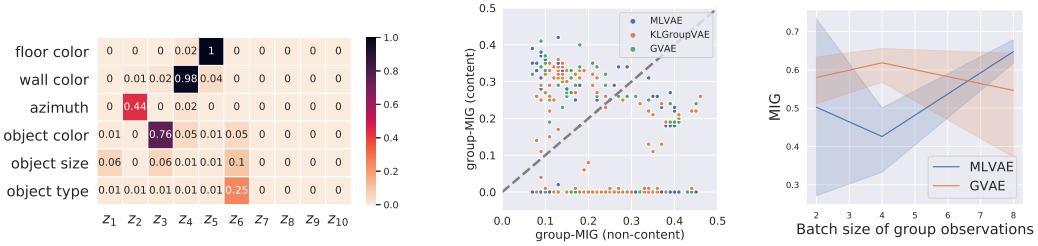
A visualization of these levels can be found in Figure 2. These types of disentanglement are useful for cases where only high-level labels are available, e.g., content vs. style, and only some groups can be further disentangled. Existing works have addressed the second requirement for group disentanglement ("shared observations have approx. same corresponding group latent variable") by using some average over the content latent variable. However, this estimate is biased and requires a certain amount of observations that share the same factors. This might be difficult with sparse, incomplete, and small datasets. We propose a KL-based regularization between the group latent variables of paired observations. This has an analytical form when the latent variables are Normal distributed and does not have any batch-size dependency. Using the total correlation (TC) and a KL term on the group latent variable, the objective becomes

$$\begin{aligned}\mathcal{L} = \sum_{\tilde{\boldsymbol{x}}=\boldsymbol{x},\boldsymbol{x}'} \Bigg(\underbrace{\mathbb{E}_{q_\phi}[\log p_\theta(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{z}}_c, \tilde{\boldsymbol{z}}_{\setminus c})]}_{\text{reconstruction}} - \underbrace{\mathrm{D}_{\mathrm{KL}}(q_\phi(\tilde{\boldsymbol{z}}_c, \tilde{\boldsymbol{z}}_{\setminus c}|\tilde{\boldsymbol{x}}) \parallel p(\boldsymbol{z}))}_{\text{KL between approx. posterior and prior}} \Bigg) \\ - \sum_{\tilde{\boldsymbol{z}}=\boldsymbol{z},\boldsymbol{z}'} \underbrace{\beta \cdot \mathrm{TC}(\tilde{\boldsymbol{z}}_c, \tilde{\boldsymbol{z}}_{\setminus c})}_{\text{total correlation}} - \underbrace{\gamma \cdot \mathrm{D}_{\mathrm{KL}}(q(\boldsymbol{z}_g) \parallel q(\boldsymbol{z}_{g'}))}_{\text{KL between shared group latent variables}} ,\end{aligned} \tag{3}$$

where $g \in \{c, \setminus c\}$ is the group which is being shared by the paired observations $(\boldsymbol{x}, \boldsymbol{x}')$. For evaluation, we used a binary adversary which approximates the log density ratio (Kim & Mnih (2018)) to estimate the total correlation loss. We use an adversarial network which attempts to classify between "true" samples from the aggregate posterior $q(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c})$ and "fake" samples from the product of the marginals $\bar{q}(\boldsymbol{z}_c, \boldsymbol{z}_{\setminus c})$. The latent variables are independent from each other if the samples are indistinguishable and the adversary cannot do it better than random chance.

4 Evaluation

Following the experimental setup in (Locatello et al. (2019b); Chen et al. (2018)), we treated learning disentangled representations as a statistical problem instead of empirical risk minimization and hence, did not use the separate train and test sets. For evaluation, we used two datasets, namely, 3DShapes (Burgess & Kim (2018)) and dSprites (Matthey et al. (2017)). We compare our model, *group-tcVAE*, with MLVAE, Bouchacourt et al. (2018), and GVAE, Hosoya (2019). Locatello et al. (2020) have already shown that both works by Bouchacourt et al. (2018) and Hosoya (2019) are superior to the unsupervised disentanglement approaches, hence, we do not compare with them. We quantitatively compare the strength of disentanglement with the Mutual Information Gap (MIG) (Chen et al. (2018)). Further, we introduce *group-MIG*, a metric based on MIG, which quantitatively estimates the mutual information between groups and corresponding latent variables. Formally, we define group-MIG as $\frac{1}{K} \sum_{k=1}^{K} \frac{1}{H(v_k)} (\max I(z_{i=f_g(v_k)}; v_k) - \max I(z_{i \neq f_g(v_k)}; v_k))$ where K is the

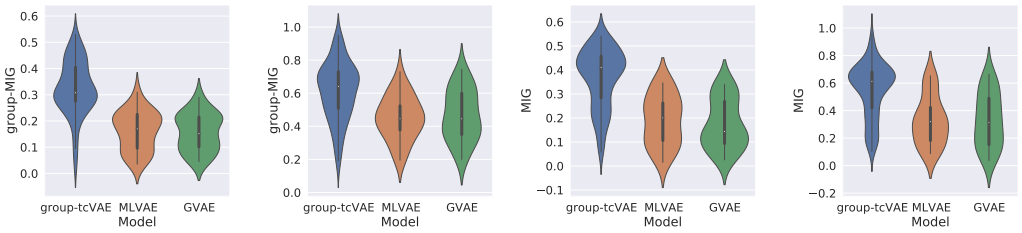


(a) 3DShapes: MI between latent dimensions and factors of variation of trained GVAE model with MIG $= 0.55$ and group-MIG $= 0.44$.

(b) dSprites: group-MIG of content and non-content information for all hyperparameter runs for MLVAE and GVAE.

(c) dSprites: MIG w.r.t. different number of shared observations for MLVAE and GVAE.

Figure 3: **Collapse and sensitity of existing weakly-supervised disentanglement models.** In all the sub-figures higher is better.



(a) dSprites (b) 3DShapes (c) dSprites (d) 3DShapes

Figure 4: **Comparisons between group-tcVAE and comparisons.** Density plots of group-MIG and MIG for group-tcVAE, MLVAE and GVAE) over all runs (*higher is better*).

number of known factors, v_k is the ground truth factor, $f_g(v_k) \in \{c, \setminus c\}$ returns the group that the factor belongs to and $I(z; v_k)$ is an empirical estimate of mutual information between continuous variable z and v_k.

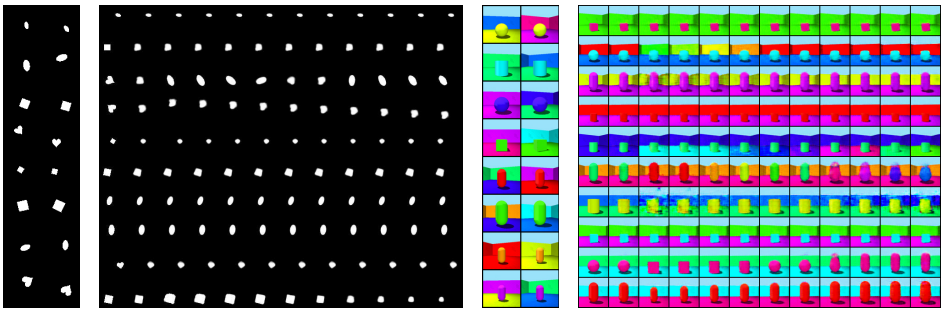
4.1 Empirical analysis of existing weakly-supervised methods

Collapse of content latent variable. The latent variable $\boldsymbol{z}_c$ can either collapse to a single factor of variation or it might even contain almost no information to any content. We visualize such behavior in Figure 3 (a) on a GVAE model trained on 3DShapes with two groups of variations c ={object color, object size and object type} and $\setminus c$ ={floor color, wall color, azimuth}. Ideally, $\boldsymbol{z}_1 - \boldsymbol{z}_5$ contains high mutual information with group factors $\setminus c$ and $\boldsymbol{z}_6 - \boldsymbol{z}_{10}$ contains high mutual information with group factors c. However, most information is captured in $\boldsymbol{z}_1 - \boldsymbol{z}_5$, whereas only a little information about object type is contained in $\boldsymbol{z}_6$. As shown in Figure 3 (b), we make similar observations with dataset dSprites in which both MLVAE and GVAE fail to capture content-specific information in the corresponding latent variable.

Sensitivity to group batch size. In practice, always having a certain number of observations that share the same group variations might be difficult, which is a requirement for MLVAE and GVAE with dSprites. This results in performance degradation and high variance (Figure 3(c)).

4.2 Weakly-supervised disentanglement

We perform an extensive evaluation on group-tcVAE to assess its performance in comparison to MLVAE and GVAE. For MLVAE and GVAE, we experimented with the hyperparameters as in Locatello et al. (2020). For group-tcVAE, we used the hyperparameter ranges $\beta = [10, 20, 30, 40, 50, 100]$, $\lambda = [1, 8, 16, 32, 64]$ and a batch size of 32 paired observations ($= 32 \times 2$). For all models, we performed five runs with different random seeds. We plotted all results in Figure 4. For both group-MIG



(a) (b) (c) (d)

Figure 5: **Qualitative results of group-tcVAE.** Training samples (a, c) and reconstruction as well as interpolations (b, d) for dSprites (a, b) and 3DShapes (c, d). Each row represents a pair of observations in (a, c). For the interpolations, each i-th row represents interpolating over only the i-th dimension of $\boldsymbol{z}$.

and MIG, group-tcVAE outperforms MLVAE and GVAE w.r.t. average and best MIG and group-MIG. With dSprites, group-tcVAE almost doubles the average and best performance, whereas with 3DShapes we observe an increase of at least 10% w.r.t. group-MIG and MIG. For the best performing models, we also plotted training samples and qualitative results in Figure 5.

5 Conclusion

We have analyzed existing weakly-supervised disentanglement models and identified challenges w.r.t. latent variable collapse and batch size sensitivity. We proposed a new framework based on total correlation for weakly-supervised disentanglement and showed through empirical evaluations on image datasets that our model improves learning disentangled representations. For future work, we plan to apply our proposed framework to challenging real-world data sets and non-image domains and extend it to semi weakly-supervised and active learning settings.

References

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Chris Burgess and Hyunjik Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in neural information processing systems*, pp. 2610–2620, 2018.

Elliot Creager, David Madras, Jörn-Henrik Jacobsen, Marissa A Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. Flexibly fair representation learning by disentanglement. *arXiv preprint arXiv:1906.02589*, 2019.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017a.

Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: improving zero-shot transfer in reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1480–1490. PMLR, 2017b.

Haruo Hosoya. Group-based learning of disentangled representations with generalizability for novel contents. In *IJCAI*, pp. 2506–2513, 2019.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Jack Klys, Jake Snell, and Richard S. Zemel. Learning latent subspaces in variational autoencoders. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6445–6455, 2018.

Francesco Locatello, Gabriele Abbati, Thomas Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the fairness of disentangled representations. In *Advances in Neural Information Processing Systems*, pp. 14611–14624, 2019a.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019b.

Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. *arXiv preprint arXiv:2002.02886*, 2020.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Rui Shu, Yining Chen, Abhishek Kumar, Stefano Ermon, and Ben Poole. Weakly supervised disentanglement with guarantees. *arXiv preprint arXiv:1910.09772*, 2019.

Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14222–14235, 2019.

USING SYSTEM CONTEXT INFORMATION TO COMPLEMENT WEAKLY LABELED DATA

Matthias Meyer & Lothar Thiele
Computer Engineering and Networks Laboratory
ETH Zurich
Zurich, Switzerland
{matthmey,thiele}@ethz.ch

Michaela Wenner & Fabian Walter
Laboratory of Hydraulics, Hydrology and Glaciology
ETH Zurich/WSL Birmensdorf
{wenner,walter}@vaw.baug.ethz.ch

Clément Hibert
Institut de Physique du Globe de Strasbourg
University of Strasbourg
hibert@unistra.fr

ABSTRACT

Real-world datasets collected with sensor networks often contain incomplete and uncertain labels as well as artefacts arising from the system environment. Complete and reliable labeling is often infeasible for large-scale and long-term sensor network deployments due to the labor and time overhead, limited availability of experts and missing ground truth. In addition, if the machine learning method used for analysis is sensitive to certain features of a deployment, labeling and learning needs to be repeated for every new deployment. To address these challenges, we propose to make use of system context information formalized in an information graph and embed it in the learning process via contrastive learning. Based on real-world data we show that this approach leads to an increased accuracy in case of weakly labeled data and leads to an increased robustness and transferability of the classifier to new sensor locations.

1 INTRODUCTION

Classifiers based on artificial neural networks have proven to be very effective across domains, however their applicability to real-world data is limited by the requirement of a clean and comprehensive dataset (Tsipras et al., 2020). Unfortunately, real-world datasets often contain artefacts arising from the system environment and contain incomplete and uncertain labels. One example of machine learning applications is natural hazard monitoring for slope failure detection (Hammer et al., 2013; Dammeier et al., 2016). Here, high misclassification requires careful retraining and post-processing (Hibert et al., 2017). In this setting, comprehensive manual annotations are infeasible for large-scale and long-term sensor network deployments due to the labor and time overhead (Meyer et al., 2019).

Hence, the process is error-prone and requires significant domain expertise. However, experts might not be available throughout the whole deployment periods of the sensor network, which inevitably leads to an annotation set containing noisy annotations limited in time and/or subset of sensors. In addition, as long as the learned features and classifier are sensitive to the detailed properties of the subsurface and the sensors, labeling and learning needs to be repeated for every new installation or classifier performance is decreased (Wenner et al., 2021). Therefore, there is a close link between weakly labeled data and robustness with respect to certain feature variations.

Fortunately, real-world deployments provide additional sources of information which could be beneficial for learning, such as correlation of sensor data due to sensor proximity. However, these information cannot be easily captured by the prevailing data/annotation pairs used for learning. Similarity learning (Schroff et al., 2015; Meyer et al., 2017), such as contrastive learning (He et al., 2020; Chen et al., 2020; Saeed et al., 2020) allows to establish relations between data pairs. However, their capability to integrate system context information is limited.

To address these challenges, we propose to transfer the concept of knowledge graphs (Hogan et al., 2021) to learning by using it for storing information about data similarity. Moreover, we extend the prevailing data/annotation learning concept to allow any data point to be an annotation for any other data point. This is accomplished by utilizing the following concepts: (i) injecting all available knowledge in form of an information graph and sampling from it, (ii) transforming the data into a common representation and (iii) the use of contrastive learning to train the system. We show that using these concepts to formalize system context information and using the additional knowledge in the learning phase leads to an increased accuracy in case of weakly labeled data and leads to an increased robustness and transferability of the classifier to new sensor locations. [1]

Our main contributions are:

- We present a method which uses system context information to counteract the negative impact of few and weak labels by combining contrastive learning with an information graph.
- We present a unified learning process in which annotations are encoded as Gaussian random vectors to treat them similar to data.
- We demonstrate on a dataset gathered from a real-world deployment in the Swiss alps, how the method can be used to train a classifier with improved generalization performance across sensors with diverging characteristics.

2 DATASET

In this work, we use data from a real-world deployment of seismic sensors at Illgraben, Switzerland. The sensor array consists of 8 seismometer (ILL01-08), each having three channels, one vertical and two horizontal. The sensors are deployed at distances of hundreds of meters up to several kilometers away from the area of interest. We aim to distinguish seismic signals from 3 different types of events namely earthquakes, slope failures and noise signals. The Illgraben event catalog was created by visual inspection of the vertical channel of the seismic waveforms and their spectrograms by experts. The earthquake catalogs provided by the Swiss Seismological Service (SED) and the European-Mediterranean Seismological Center (EMSC) served as additional ground truth for providing correct earthquake labels. The Illgraben event catalog consists of 320 to 560 time segments per station each containing an event, summing up to 32.5 hours of labelled seismic data recorded at a sampling frequency of 100 Hz. In addition, the dataset contains randomly sampled, verified time segments without activity with a total duration approximately equal to the event segment's total duration.

3 METHOD

In our scenario, two major issues need to be addressed, namely (i) few and weak labels and (ii) classifier robustness. The first issue requires an improvement and extension of the annotation set, the latter requires that the learning method can adapt to out-of-distribution samples.

In contrast to real-time classification, environmental monitoring usually relies on post-processing of a long-term dataset. Therefore, an extensive dataset is usually available for training albeit not always thoroughly annotated. In our scenario, we can make use of non-annotated data by using general assumptions about the specific sensor deployment, for example about sensor proximity: The same event is captured by multiple seismometer channels and possibly multiple stations, but with different signal signatures. These differences are caused by groundwave propagation as well as properties of the seismometers, for example ground coupling. Thus, we obtain "different views" of the same event. Contrastive learning has shown to benefit from such different views. Intuitively speaking, contrastive learning achieves an embedding of data samples in a latent space by moving representations of different views of the same event closer together while increasing the distances of representations of different events.

To combine contrastive learning with all available information, we propose to make use of an information graph, which holds annotations as well information about the relation between time segments, channels and stations. We expect that by using system context information we simultaneously

[1] Further content is available at https://matthiasmeyer.xyz/system-context-info/

(i) improve our annotation set by adding additional information to each annotated segment and (ii) include non-annotated, out-of-distribution samples in the training process.

The information graph is filled by subdividing the seismic signals into segments using a window length T_w. Each segment is represented by a node in the information graph. An edge is introduced between segments A and B if the segments overlap in time and A is from a different station or different channel than segment B. To reduce the possibility of learned shortcuts (Fonseca et al., 2020) no edges to segments of the same channel are added.

The information graph is used to train a model $f(\cdot)$ which embeds each segment x_i into a common space $z_i = f(x_i)$, with $\boldsymbol{z}_i \in \mathbb{R}^d$. Similar to related work, we separate the model into an encoder and encoder head (Chen et al., 2020). As illustrated in Fig. 1, a fixed number N_e of edges is sampled from the graph for every batch during training and connected data segments are loaded. Any duplicate segments are removed from the batch before computing $f(\cdot)$, leading to a number of data segments in the batch of $N \leq 2N_e$. Each data segment is encoded by the encoder and subsequently transformed by the encoder head into an embedding vector $\boldsymbol{z}_i$.

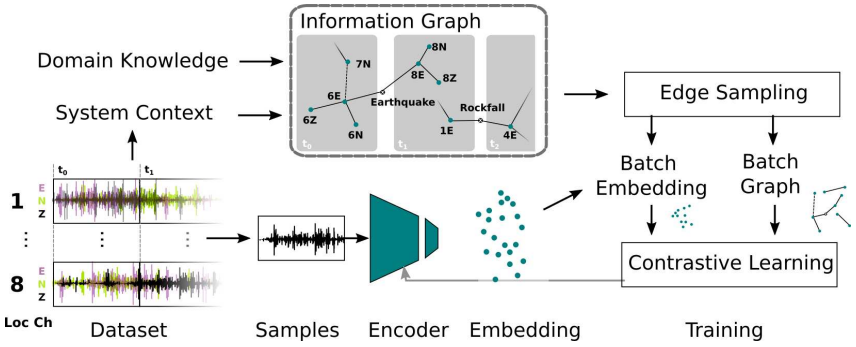


Figure 1: The central entity is the information graph which combines knowledge from domain experts, for example annotations or signal propagation behaviour, as well as dataset-specific knowledge of each data segment, for example **loc**ation, **ch**annel, time. The combination of contrastive loss, information graph and encoder allows to learn a suitable embedding for the classification task.

By sampling the edges we construct a subgraph of the information graph with non-negative adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{N \times N}$. To avoid that second order neighbours of a node have a detrimental impact on training by not being directly connected, we add second order neighbors by $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{A}^2$. In this setting, we define the contrastive loss between a pair s, t (source and target of an edge), with the adjacency matrix $\boldsymbol{B}$ as follows:

$$L_{s,t} = -B_{s,t} \log \frac{\exp(\phi(\boldsymbol{z}_s, \boldsymbol{z}_t)/\tau)}{\sum_{n=1}^{N} \mathbb{1}_{[B_{s,n}=0]} \exp(\phi(\boldsymbol{z}_s, \boldsymbol{z}_n)/\tau)} \tag{1}$$

where $B_{s,t}$ represents the weight of the edge connecting s, t. $\phi(\cdot)$ is a similarity function, which in our implementation is the Cosine similarity. τ is a temperature scaling. The indicator function $\mathbb{1}$ is evaluating to 1 iff $B_{s,n}$ is zero.

Annotations are considered in the information graph by introducing N_c anchor nodes, where N_c equals the number of classes. Each segment belonging to a class is connected to the anchor node of that class by an edge. Two strategies can be employed to compute Eq. 1 for an edge with an annotation anchor node.

The first option makes use of the fact that $\boldsymbol{B}$ contains second order neighbors meaning that all nodes sharing an edge with an annotation are also directly connected. Thus, the edges with an annotation node can be skipped while computing Eq. 1 but representations with the same annotation are still moved closer together, which resembles the work by Khosla et al. (2020).

The second option is to introduce a high-dimensional L2-normalized Gaussian random vector $\boldsymbol{a}^{(c)} \in \mathbb{R}^d$ for class c into the batch which acts as the target z_t during computation of Eq. 1. The vectors

	Accuracy			
	all	all+SC	one-station	one-station+SC
Random Forest	86.4 %	-	n.a.	-
ResNet18+XE	91.3 % $\pm$0.5	-	50.0 % $\pm$15.0	-
ResNet18+IG+links	92.3 % $\pm$0.3	93.8 $\pm$0.3	44.0 % $\pm$15.6	84.1 % $\pm$2.6
ResNet18+IG+anchors	92.0 % $\pm$0.4	93.9 $\pm$0.6	47.9 % $\pm$16.7	85.0 % $\pm$1.8

Table 1: Classifier accuracies for different sets of available training annotations. Either all annotations (*all*) or only annotations for one station are available (*one-station*). Additionally, the information graph (IG) is used with or without system context information (*SC*).

are fixed at the beginning of the training. Here, we make use of the fact that any two random high-dimensional vectors are almost orthogonal to each other with high probability (Blum et al., 2020), thus data points of different classes are trained to move "far away" from each other. The first strategy will be referred to as *link*, the latter as *anchor* in the following evaluation.

4 Experimental evaluation

We evaluate the proposed approach by performing an ablation study and comparing the system to a random forest classifier, which is best practice for slope failure detection (Wenner et al., 2021).

For the ablation study we use a classifier based on a single-channel variant of ResNet18 (He et al., 2015) as encoder in combination with an MLP with 1 hidden layer as encoder head. During classification the encoder head is replaced with a classification head differing only in output size. Input to the ResNet18 is a log-compressed spectrogram of the seismic data. For more implementation details please refer to the Appendix A. The ResNet18 model is trained with three methods, cross-entropy loss between the output of the classification head and the ground truth (Resnet18+XE), contrastive pretraining using the information graph (IG) and either the link (Resnet18+IG+link) or anchor (Resnet18+IG+anchor) strategy. Subsequently, the classification head is trained using cross-entropy loss. We compare training with system context information (SC) and training without it.

The benefit of our approach for the weakly-labeled setting is evaluated by using the available training annotations of all stations. The experiments are repeated 5 times and mean and standard deviation are reported. Robustness is evaluated by training the model variants when only a subset of the annotations are available. While the whole training data is available to train a classifier only the annotations for one of each of the 8 seismic stations can be used. All reported accuracies are based on evaluation on the test set using all stations and are reported as mean and standard deviation of all one-station evaluations.

The results presented in Table 1 show that the ResNet18 classifiers outperform the random forest classifier in the weakly-labeled setting (*all* and *all+SC*). The *all* column, illustrates that training using contrastive pretraining improves the performance significantly in comparison to the random forest classifier but only slightly in comparison to using cross-entropy loss (ResNet18+XE). However, if we include the system context information the accuracy improves significantly (*all+SC* column), demonstrating our method's applicability to weakly labeled data.

In the robustness experiments, all classifiers show a comparable bad performance of around 50 % on average if only annotated data of one station is used (*one-station*). If more non-annotated data from other stations is available, our method takes advantage of the system context information (SC) stored in the information graph (*one-station+SC*) and the average accuracy rises to over 84 %. The increase comes from a better generalization to other sensors, as illustrated in Fig. 2. The left figure illustrates the poor generalization of ResNet18+XE to other stations than the one used for training. If, however, non-annotated data from other stations and system context information is available, our method increases classifier performance on all stations, thus demonstrating and increased robustness.

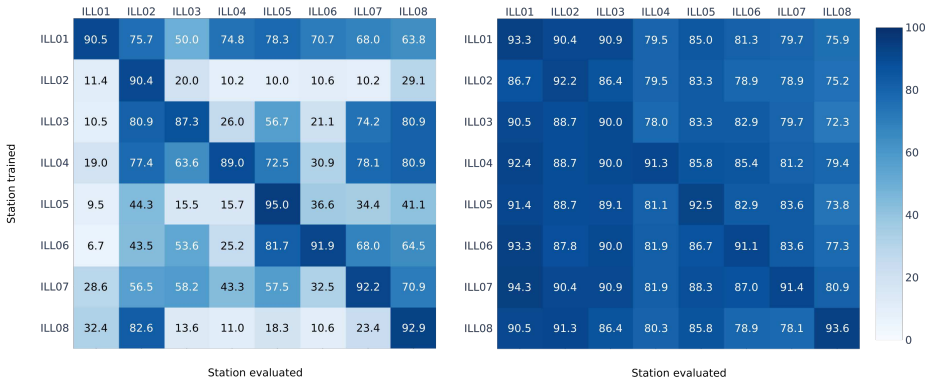


Figure 2: Each row indicates of which station the annotated training subset was used. Each column indicates the respective score on the subset of the test dataset for each station. (Left): Results for ResNet18+XE. (Right): Results for ResNet18+IG+anchor.

5 Conclusion

In this paper we have presented a novel approach to learn with weakly labeled data for the case of mass movement monitoring. By using contrastive learning we can increase the classification accuracy compared to the reference implementation. Moreover, the presented method unifies data and annotation representations and thus inherently allows to integrate additional system information into the learning process. This additional information leads to a strong performance increase in a setting with limited annotations and diverging sensor characteristics, demonstrating increased robustness across sensors.

REFERENCES

Avrim Blum, John Hopcroft, and Ravi Kannan. *Foundations of Data Science*. Cambridge University Press, first edition, January 2020. ISBN 978-1-108-75552-8 978-1-108-48506-7. doi: 10.1017/9781108755528.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, February 2020.

Franziska Dammeier, Jeffrey R. Moore, Conny Hammer, Florian Haslinger, and Simon Loew. Automatic detection of alpine rockslides in continuous seismic data using hidden Markov models. *Journal of Geophysical Research: Earth Surface*, 2016. ISSN 21699011. doi: 10.1002/2015JF003647.

Eduardo Fonseca, Diego Ortego, Kevin McGuinness, Noel E. O'Connor, and Xavier Serra. Unsupervised Contrastive Learning of Sound Event Representations. *arXiv:2011.07616 [cs, eess]*, November 2020.

C. Hammer, M. Ohrnberger, and D. Fäh. Classifying seismic waveforms from scratch: A case study in the alpine environment. *Geophysical Journal International*, 2013. ISSN 0956540X. doi: 10.1093/gji/ggs036.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Clément Hibert, Floriane Provost, Jean Philippe Malet, Alessia Maggi, André Stumpf, and Valérie Ferrazzini. Automatic identification of rockfalls and volcano-tectonic earthquakes at the Piton de la Fournaise volcano using a Random Forest algorithm. *Journal of Volcanology and Geothermal Research*, 2017. ISSN 03770273. doi: 10.1016/j.jvolgeores.2017.04.015.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *arXiv:2003.02320 [cs]*, January 2021.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised Contrastive Learning. *arXiv:2004.11362 [cs, stat]*, December 2020.

Matthias Meyer, Jan Beutel, and Lothar Thiele. Unsupervised Feature Learning for Audio Analysis. *arXiv:1712.03835 [cs]*, December 2017.

Matthias Meyer, Samuel Weber, Jan Beutel, and Lothar Thiele. Systematic identification of external influences in multi-year microseismic recordings using convolutional neural networks. *Earth Surface Dynamics*, 7(1):171–190, February 2019. ISSN 2196-6311. doi: 10.5194/esurf-7-171-2019.

F. Provost, C. Hibert, and J. P. Malet. Automatic classification of endogenous landslide seismicity using the Random Forest supervised classifier. *Geophysical Research Letters*, 2017. ISSN 19448007. doi: 10.1002/2016GL070709.

Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive Learning of General-Purpose Audio Representations. *arXiv:2010.10915 [cs, eess]*, October 2020.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, June 2015. doi: 10.1109/CVPR.2015.7298682.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. From ImageNet to Image Classification: Contextualizing Progress on Benchmarks. *arXiv:2005.11295 [cs, stat]*, May 2020.

M. Wenner, C. Hibert, A. van Herwijnen, L. Meier, and F. Walter. Near-real-time automated classification of seismic signals of slope failures with continuous random forests. *Natural Hazards and Earth System Sciences*, 21(1):339–361, 2021. doi: 10.5194/nhess-21-339-2021. URL https://nhess.copernicus.org/articles/21/339/2021/.

Table 2: Random forest parameters

Number of trees	400
Split quality measure	Gini criterion
Minimum number of samples required to be a leaf node	1
Minimum number of samples for an internal node to be split	2

A Implementation Details

A.1 Details to Contrastive Learning with Information Graph

The seismic signals are subdivided into segments using a window length T_w and a stride T_h. The data subset for pre-training uses $T_w = 30s$, $T_h = 30s$, the subset for fine-tuning and the test set use $T_w = 30s$, $T_h = 15s$. Each segment's annotation is determined using the Illgraben event catalogue, which is split into training and test set with a ratio of approx. 70/30. Linear detrend is applied to the seismic signal before it is transformed into a log-compressed spectrogram with window length of 2.56 s and stride of 0.08 s. No data augmentation is applied. As encoder we use a single-channel variant of ResNet18 (He et al., 2015), without the final linear layer. The output of the encoder is a 512-dimensional vector, which is then passed through the encoder head, consisting of a MLP with a hidden layer of size 512, batch normalization and ReLU non-linearity. The encoder head's output size is $d = 128$ and L2 normalized. During fine-tuning, the same encoder head architecture (with random initialization) is used as a classification head with an output size equal to the number of classes N_c.

In the supervised training we train encoder and classification head jointly using cross-entropy loss.

For semi-supervised training (*all+SC* and *one-station+SC*), we train the encoder and encoder head with contrastive loss, then for fine-tuning we replace the encoder head with a classification head and train the classification head with cross-entropy loss while keeping the encoder weights fixed. In every epoch we first train with contrastive loss, then fine-tune the classification head. For optimization we use SGD with momentum 0.9 and weight decay 10^{-4} and a batch size of 128. The temperature coefficient τ is set to 0.1. We use a cosine annealing scheduler. In our experiments the edge weights of the information graph are 1.

To counter class-imbalance, each batch contains the same number of examples for each class. During semi-supervised training the non-annotated data out-weights the annotated data by a factor of approx. 4.5 in each batch. We select our model based on a validation set which is 20% of the training set, except for the one-station+SC experiment. Here, we select model from the last epoch, since model selection on the one-station subset would deteriorate the generalization effect. More details, e.g. the code and hyperparameters for individual experiments will be made available on the paper's project page.

A.2 Details to Random Forest Classifier

FollowingProvost et al. (2017) and Wenner et al. (2021) we computed a total of 55 signal characteristics in the time and frequency domain, e.g., information on the signal form and dominant frequencies. For a complete description of the chosen features see Wenner et al. (2021). We performed a three-fold-cross-validation grid search to optimize classifier performance. Final parameters are presented in Table 2.

WEAKLY SUPERVISED MULTI-TASK LEARNING FOR CONCEPT-BASED EXPLAINABILITY

Catarina Belém, Vladimir Balayan, Pedro Saleiro & Pedro Bizarro
Feedzai, Lisbon, Portugal
<first>.<last>@feedzai.com

ABSTRACT

In ML-aided decision-making tasks, such as fraud detection or medical diagnosis, the human-in-the-loop, usually a domain-expert without technical ML knowledge, prefers high-level concept-based explanations instead of low-level explanations based on model features. To obtain faithful concept-based explanations, we leverage multi-task learning to train a neural network that jointly learns to predict a decision task based on the predictions of a precedent explainability task (*i.e.*, multi-label concepts). There are two main challenges to overcome: concept label scarcity and the joint learning. To address both, we propose to: i) use expert rules to generate a large dataset of noisy concept labels, and ii) apply two distinct multi-task learning strategies combining noisy and golden labels. We compare these strategies with a fully supervised approach in a real-world fraud detection application with few golden labels available for the explainability task. With improvements of 9.26% and of 417.8% at the explainability and decision tasks, respectively, our results show it is possible to improve performance at both tasks by combining labels of heterogeneous quality.

1 INTRODUCTION

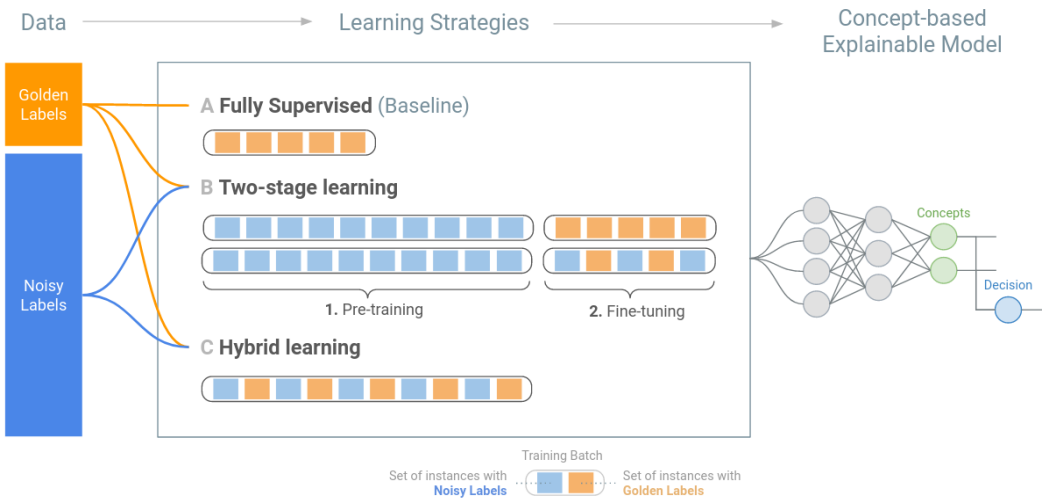


Figure 1: Weakly supervised multi-task learning strategies for concept-based explainability: (A) *baseline* strategy resorts exclusively to golden explainability labels; (B) *two-stage learning* strategy (1) uses noisy explainability labels to pre-train a base model and (2) fine-tuning either using purely golden batches or hybrid ones; (C) *hybrid learning* strategy only uses hybrid batches of golden and noisy explainability labels.

The AI black-box paradigm has led to a growing demand for model explanations (Ribeiro et al., 2016; Lundberg & Lee, 2017). Concept-based explainability emerges as a promising family of

methods addressing the information needs of humans-in-the-loop without technical ML knowledge. It concerns the generation of high-level concept-based explanations (*e.g.*, "Suspicious payment") rather than low-level explanations based on model features (*e.g.*, "MCC=7801").

Concept-based explainability can be implemented using a multi-task learning approach (Kim et al., 2018; Melis & Jaakkola, 2018; Ghorbani et al., 2019; Koh et al., 2020). With such implementation both the decision and the explanation are learned jointly. We refer to the individual tasks as decision (or predictive) task and explainability task. Likewise, we refer to the annotation types used throughout learning as decision (or class) labels and concepts (or explainability) labels, respectively.

There are two main challenges towards this approach: concept label scarcity and learning to jointly predict the decision and the concepts that feed that decision. On the one hand, the creation of golden (or human) labeled datasets remains an arduous and expensive task irrespective of the application domain. On the other hand, the joint learning depends on several factors (*e.g.*, learning rates and/or dominance relationships of the involved tasks) and, if done incautiously, may cause deterioration of the predictions' quality. Concept-based explainability methods must provide high-level domain knowledge explanations without compromising the quality of the conventional classification task.

This work aims to implement multi-task learning for concept-based explainability in the context of a real-world e-commerce fraud detection application. To overcome the aforementioned challenges, we first resort to weak supervision. Based on a few rule-based predictors available *off-the-shelf* in historical production data, we are able to automatically generate noisy concept labels for datasets with millions of instances. Although imprecise (or weak), these noisy explainability labels prove valuable assets in training (deep) concept-based explainability models.

Finally, since we also had access to a small set of golden explainability labels, we set out to explore learning strategies to enhance joint task performance. In particular, we explore the impact of combining different types of supervision (*i.e.*, weak and full) when training deep learning models. Figure 1 summarizes the learning strategies we apply.

2 Preliminaries on Concept-based Explainability

Concept-based explainability consists of producing explanations in the format of high-level domain knowledge concepts. Following this definition, human specialists help devise a concepts taxonomy with all the relevant concepts for a specific task. These concepts closely reflect the expert's reasoning process when performing the task and therefore are perceived as suitable explanations.

Implementation-wise, this explainability paradigm can be incorporated into deep neural networks in the form of multi-task learning (Ruder, 2017; Zhang & Yang, 2017; Melis & Jaakkola, 2018). To this end, the model is enlarged with an explainability (or semantic) task and the learning process is modified to allow for the joint learning of the existing decision (or predictive) task and the explainability task. In practice, depending on the tasks affinity, multi-task learning often boosts individual task performance (Vandenhende et al., 2021; Zhang & Yang, 2017). Let $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}_D^{(i)}, \boldsymbol{y}_E^{(i)})\}_{i=1}^N$ denote a dataset with N instances with d-dimension feature vector $\boldsymbol{x} \in \mathbb{X} = \mathbb{R}^d$, m-dimension decision label vector $\boldsymbol{y}_D \in \mathbb{Y}_D = \{0,1\}^m$, and k-dimension explanations $\boldsymbol{y}_E \in \mathbb{Y}_E = \{0,1\}^k$. The decision task is thus modeled through an m multi-classification task, whereas the explainability task is modeled as a multi-label classification task with k concepts. Jointly learning the decision and explainability tasks comes down to learning the function $f^* : \mathbb{X} \rightarrow (\mathbb{Y}_D, \mathbb{Y}_E)$.

Figure 2 shows a hard-parameter sharing approach towards achieving concept-based explainability (Vandenhende et al., 2021). In practice, we force both tasks to share the parameters of the initial layers and keep specialized output layers for each individual task. The hierarchy observed in the output layers presupposes the explainability task carries pertinent information to the decision layer that is not explicit in the input data. Conversely, removing this dependency and learning both tasks in parallel may lead to learning decisions that are decoupled from the explanations. A (deep) feed-forward network with L hidden layers defines a mapping $f(\boldsymbol{x}; \boldsymbol{\theta}_{1:\mathrm{L}})$, where $\boldsymbol{\theta}_{1:\mathrm{L}}$ denotes all the parameters up to the L^{th} layer. Parameterized by $\boldsymbol{\theta}_E$, the explainability layer that follows defines the mapping $\hat{\boldsymbol{y}}_E = f_E(f(\boldsymbol{x}; \boldsymbol{\theta}_{1:\mathrm{L}}); \boldsymbol{\theta}_E)$, hence producing a k-dimensional vector with uncalibrated probabilities for each concept. These probabilities stem from applying the sigmoid activation function (one per each neuron unit). In addition to being explanatory, the concepts vector $\hat{\boldsymbol{y}}_E$ also serves

the decision layer, $\hat{\boldsymbol{y}}_D = f_D(f_E(\boldsymbol{x}; \boldsymbol{\theta}_{1:\mathrm{E}}); \boldsymbol{\theta}_D)$ with $\boldsymbol{\theta}_D$ denoting the parameters of the decision layer. The m-dimension probability vector $\hat{\boldsymbol{y}}_D$ results from applying the softmax function.

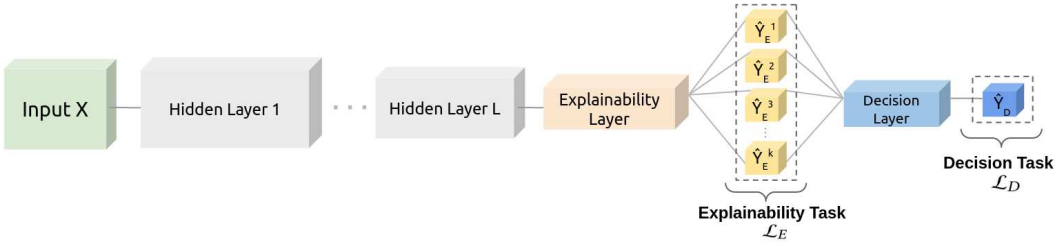


Figure 2: Concept-based explainable feedforward model architecture: The explainability layer produces the concepts (yellow), which are the inputs for the decision layer. The concepts (explanations of the decision task) are also outputs of the network. Colors indicate layer type: input vector (green); hidden layer (grey); explainability layer (orange); decision layer (blue); output vectors (dashed box).

Learning f^* requires mastering both the explainability and the decision tasks. One solution is to minimize the cross-entropy loss of each task, henceforth denoted $\mathcal{L}_E(\hat{\boldsymbol{y}}_E, \boldsymbol{y}_E)$ and $\mathcal{L}_D(\hat{\boldsymbol{y}}_D, \boldsymbol{y}_D)$, and combine them into a meta-loss $\mathcal{L}$ as defined in Equation 1. Here, $\alpha \in [0, 1]$ weighs the relative importance of the decision task over the explainability task and, thus targets different explainability-accuracy trade-offs.

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}_E, \boldsymbol{y}_D) = \alpha\, \mathcal{L}_D(\hat{\boldsymbol{y}}_D, \boldsymbol{y}_D) + (1 - \alpha)\, \mathcal{L}_E(\hat{\boldsymbol{y}}_E, \boldsymbol{y}_E) \tag{1}$$

During training, the model uses backward propagation of errors (*backprop*) together with mini-batch gradient descent algorithm to minimize $\mathcal{L}$ (Rumelhart et al., 1986). Obviously, the proposed models' generalization capacity heavily depends on the training data availability and its quality. The following section details how different approaches can be used to reach reasonable predictive performance at both tasks when departing from a small golden explainability task dataset (low-resources setting) and a large golden decision task dataset (high-resources setting).

3 Methodology

In this section, we start by identifying the key properties of concept-based explainability tasks that propel us into adopting a weak supervision approach. Afterwards, we propose a knowledge-based labeling technique that produces numerous but imprecise (noisy) concept labels. Finally, we put forward two learning strategies to better exploit the available labels (*i.e.*, using both noisy and golden concept labels).

3.1 Properties for Weak Supervision

Despite empirical success in low-resources natural language tasks (Mintz et al., 2009; Zeng et al., 2015), weak supervision is seldom applied in algorithmic decision-making tasks. We identify three main characteristics, inherent to most industry AI solutions, that incentivize the use of weak supervision to make feasible the concept-based explainability paradigm.

Abundant Golden Decision Labels. Learning the mapping $f^* : \mathbb{X} \rightarrow (\mathbb{Y}_D, \mathbb{Y}_E)$ entails having labels for both the decision and the associated concepts. In many industry settings, it is relatively straightforward to obtain massive golden labeled datasets (hundreds of thousands or millions of instances) for the decision task. For example, modern financial fraud prevention systems are already designed to persist the outcome of payment transactions (legitimate or fraudulent).

Scarce Golden Explainability Labels. Many systems are unprepared (or lack the infrastructure) for capturing specific concept annotations. Even in cases where companies do accrue information about the human-in-the-loop thinking process, it is frequently done in an impromptu and unstructured fashion, making it difficult and impractical to automatically process. Alternatively, recruiting people to hand-curate tens of thousands of instances can quickly become prohibitively time-consuming and expensive (Mintz et al., 2009) – a cost that is further exacerbated in multi-label settings.

Availability of domain knowledge information. At the same time, modern AI-powered systems often co-exist with rule systems. For instance, in a fraud prevention solution, rules-based systems can be very effective in short-listing payment transactions based on the triggered rules. Moreover, enlarging the set of rules with additional business constraints (*e.g.*, automatically reject transactions with a specific IP) is trivial.

3.2 DISTANT SUPERVISION

Previous research works adopt weak supervision strategies to overcome the label scarcity problem (Mintz et al., 2009; Zeng et al., 2015). In particular, they draw heuristics based on "distant" systems, such as databases and dictionaries, to automatically create abundant and imprecise labeled datasets. This technique is also known as *distant supervision* (Mintz et al., 2009; Go et al., 2009).

Our work applies a similar technique to overcome the concepts label scarcity inherent to concept-based explainability. We use *distant supervision* to heuristically extract imprecise proxy annotations for the concepts. We draw on the information left by rules-based systems that co-exist with real-world AI-based solutions. Through the extraction of semantic information within each rule, we build one-to-many mappings of a set of rules to the corresponding concepts in the taxonomy. Next, to prevent (some) label noise, domain experts validate the correctness and significance of these mappings.

Table 1: *Rule-to-concept* mapping examples

Rule description	**Mapped concepts**
Order contains risky product styles.	Suspicious Items
User tried *n* different cards last week.	Suspicious Customer, Suspicious Payment

Table 1 presents two validated *rule-to-concept* mappings in an *e-commerce* fraud detection use case. Each rule comprises a human readable description with enough domain knowledge to discern the most adequate fraud concepts. A single rule may be linked with more than one domain concept in the fraud taxonomy. Consider a payment transaction $x \in \mathbb{X}$ for which no concept labels exist and for which both rules in the table are activated. The proposed approach automatically attributes the labels "Suspicious Items" (resulting from the first rule), "Suspicious Customer", and "Suspicious Payment" (resulting from the second rule) to x. The true potential of this technique lies in its ability to bulk annotate large (pre-existing) data volumes, thus allowing us to quickly create multi-label datasets. Despite still requiring human effort to create these associations, the total human effort is negligible when compared with the manual annotation of the same volume of data.

As previously mentioned, this work pressuposes the existence of a large dataset encompassing information about the set of triggered rules as well as the decision labels (*i.e.*, the labels for the decision task $\mathbb{Y}_D$). Then, using the described distant supervision approach, we bulk assign noisy labels $\boldsymbol{y}_E \in \{0,1\}^k$. We obtain the final weakly labeled dataset, $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}_D^{(i)}, \boldsymbol{y}_E^{(i)})\}_{i=1}^N$, ready to be used for model training. We expect this partially-weak-partially-full supervised dataset to yield significant performance gains in terms of the explainability task when compared to having no labeled data. The ensuing section focus on how to leverage the weak explainability labels obtained to improve performance through learning.

3.3 LEARNING STRATEGIES

From an empirical standpoint, we consider the interplay between weak (or distant) supervision and multi-task learning: *Are we able to outperform the fully supervised baseline*? To answer this question, we devise two learning strategies to combine the explainability labels' quality differently. With these strategies, the model is given access to larger datasets, which may benefit its generalization capabilities at both tasks.

3.3.1 TWO-STAGE LEARNING STRATEGY

We suggest separating the learning process in two sequential stages: the *pre-training stage* and the *fine-tuning stage*. The former refers to the training of a base model using the large but noisy dataset, whereas the latter intends to specialize the base model using golden labels. Technically speaking, we use a transfer learning technique (Weiss et al., 2016) in which we learn the model's parameters on a related dataset (the noisy dataset) and use it to obtain better performing models on the smaller target dataset (the golden-labeled dataset). In practice, we assume that, by tuning the model with a small set of examples labeled by domain-experts, it will entail improved explainability.

Notwithstanding the stated improvements on convergence and training speed during model training, if applied naively, the proposed approach can cause performance decay (Wang et al., 2018) – an unpractical scenario specially in high-stakes AI applications. This is often the case when the datasets on each stage are drawn from different distributions. It may also occur that upon transferring this knowledge to the target dataset, the model ends up completely discarding previous information. For instance, using a learning rate value that causes steep updates or even iterating for many epochs, can be too aggressive and cause the model to unlearn the decision task. Consequently, we suggest freezing the hidden layers of the concept-based explainability model (grey layers in Figure 2) and only have the task-specific layers being tweaked and learned in the *fine-tuning stage*.

3.3.2 HYBRID LEARNING

Depending on the real-world application, the explainability task is likely to assume an auxiliary role in the learning process. When training this task on the noisy labels, we risk learning a highly biased model. Although fine-tuning may help to pay off for some of the introduced bias, this strategy can still be suboptimal.

For that reason, we test a *hybrid* learning strategy with the intent to promote faster and better results. Rather than using fully distantly supervised batches to train the multi-task model, we create mixed batches with part golden, part noisy concept labels. As a consequence, we assume that gradient updates tend to be more informative and less prone to capture noise.

4 EXPERIMENTS AND RESULTS

We evaluate and compare the proposed learning strategies in a real-world e-commerce fraud detection application. The main task, *i.e.*, the traditional decision task, aims to discern fraudulent from legitimate payment transactions (a binary classification setting). Conversely, the explainability task is perceived as an auxiliary task, whose goal is to improve the human-in-the-loop's decision-making. Using a total of 14 domain concepts (extracted from a fraud patterns taxonomy), the explainability task concerns the attribution of the corresponding concepts to the transaction (a multi-label classification setting).

Datasets. We use a privately held dataset totalling approximately 6 million payment transactions. Each transaction consists of information about the purchase (*e.g.*, number of items, shipping address), the fraud decision label, and the information about the triggered rules. We apply the distant supervision technique (described in Section 3.2) to obtain the *noisy explainability* labels. Additionally, we have access to smaller subset of the dataset with human-annotated labels for both tasks, which totals approximately 1.3k transactions, 37% of which are fraudulent. Note that all labels referring to the fraud decision task are golden and, henceforth, denoted as *golden decision* labels. Conversely, the explainability task spans both a small *golden explainability* dataset and a large *noisy explainability* dataset. Figure 3 depicts the datasets timeline and the evaluation splits of our experiments. We follow a three-way holdout evaluation split composed of training (only 2% of instances are fraudulent), validation (4% of fraud prevalence), and a test set (4% fraudulent events). These are used for training different model configurations, for selecting the decision label threshold, and for comparison between the generalization capabilities of each variant.

Learning Variants. We evaluate a total of three settings (all of which depicted in Figure 1) using two random seeds: (1) full supervision, which trains under a fully supervised low-resources setting; (2) *two-stage learning*, which first pre-trains a base model using abundant noisy explainability labels

and is then refined using few golden explainability labels; and (3) *hybrid learning*, which combines both noisy and golden explainability labels into the same batch during learning.

Hyperparameter Optimization. We run the same hyperparameter grid for each of the evaluated variants, in which we vary the number and dimension of hidden layers, learning rate, as well as the relative weight α of the importance task over the explainability task (see Section 2). A second hyperparameter grid is defined and used during the fine-tuning phase of the *two-stage* learning strategy. In this grid, we vary the number of epochs, batch size, and learning rate.

Metrics. We evaluate models in terms of their predictive performance at both tasks in the golden test set. For the decision task, we are restricted by business requirements to measure fraud recall at 5% false positive rate (FPR), henceforth, abbreviated as recall@5%. Conversely, we do not have any business constraint on the explainability performance metric. Instead, we use the mean Average Precision (mAP) because of its applicability and usefulness in real-world scenarios. In particular, it focus on the number of correctly predicted concepts and does not impose restrictions on the explanation size (*i.e.*, how many concepts each explanation should contain).

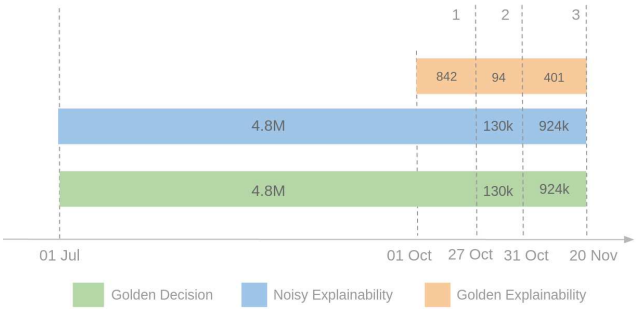


Figure 3: Datasets timeline and corresponding evaluation splits. Models are trained in the training set (1), thresholds determined in the validation set (2), and models' performance compared in the test set (3).

4.1 Fully supervised Learning

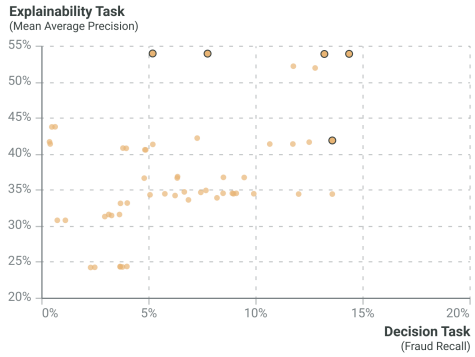
The baseline represents an aggravated low-resources setting where only a small fraction of the dataset has golden labels for both tasks and there is no other information to take advantage of. Thus, we cast the concept-based explainability multi-task problem in a supervised fashion and train models in 842 payment transactions. In this case, we fix the fraud prevalence in the batch size at 37%. Figure 4a shows the explainability-accuracy trade-off obtained in the golden test sets for all the models obtained. The larger sized points represents the Pareto optimal models (*i.e.*, the optimal trade-offs between both tasks) at the two different runs.

Considering the performance at the decision task, most baseline models struggle to discern fraudulent from legitimate transactions with the best model achieving approximately 15% recall@5%. Similar results can be observed for the explainability task, though with seemingly larger mAP values. Indeed, only six models achieve mAP values larger than 50%.

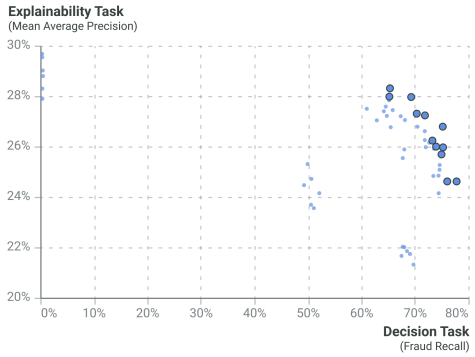
In general, the results seem to corroborate the idea that (deep) neural networks need massive datasets to reach peek performance. Training models for longer and more epochs in low-resource scenarios quickly results in model overfitting as observed by the higher model density in the bottom-left region of the plot. We also observe that simpler models (*i.e.*, models with fewer layers and lower learning rates) reach the best compromises in terms of recall@5% and mAP.

4.2 Two-stage Learning

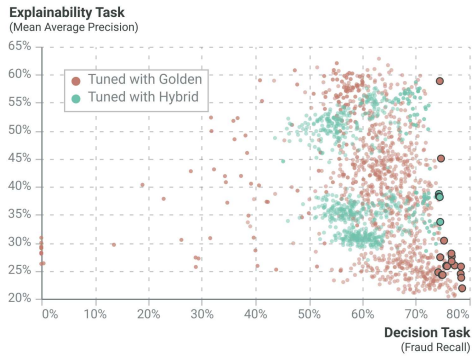
This learning strategy is carried in two stages. The first one, dubbed the *pre-training stage*, considers a high-resources scenario with approximately 4.8 million labeled transactions. This data contains both accurate golden decision labels and imprecise concept labels (or *noisy explainability* labels). Using these labels, we run the hyperparameter grid training 27 models per random seed. From this



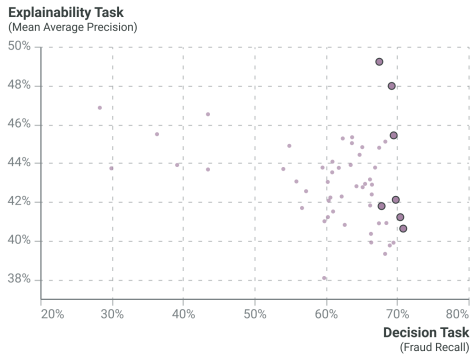
(a) Fully supervised learning strategy.



(b) Two-stage learning strategy (pre-training).



(c) Two-stage strategy learning (fine-tuning).



(d) Hybrid learning strategy.

Figure 4: Explainability and predictive accuracy performance results of each learning strategy in golden test sets. These comprise the results of both random seeds. The larger-sized points represent optimal trade-offs of each learning strategy.

pool of models, we pick the Pareto optimal ones, *i.e.*, the ones with the best explainability-accuracy trade-off. The selected model (dubbed base models) are then used in a second stage involving different fine-tuning configurations and a small golden explainability dataset.

Pre-training results. Figure 4b shows the results for the first stage. We find a significant increase in the decision task performance when compared with the former fully supervised approach. As expected, all the models achieve reasonably high values of fraud recall@5% above 50%. This can be explained by the size of the training datasets (in the order of millions of transactions). Comparing with the baseline, this represents a boost of at least 200% in fraud recall@5%. Additionally, we observe that some models present very low values for decision task, while maintaining relatively "high" explainability. We find that these models had small learning rate and weighed more heavily the explainability task.

On the other hand, we see a tendency towards lower values at the explainability task, with mAP values falling down between 20% and 30%. Comparing to the baseline, this performance metric deteriorates significantly. One possible reason is due to the noise associated with the explainability labels, as evidenced by the low Jaccard similarity index between both types of explainability labels. Nonetheless, from a business perspective, the obtained trade-offs are better than the fully supervised results, since the performance at the decision task is significantly higher.

Fine-tuning results. For this step, we selected all the base models (as discussed in Section 4.1) representing the best trade-offs (the larger sized points in the Figure 4b). The second stage envisions the amelioration of base models' explainability performance through fine-tuning and different learning approaches. In particular, we experiment fine-tuning with fully supervised batches (*i.e.*, pure golden

label batches), as well as with a more hybrid version (*i.e.*, involving both noisy and golden labels in the batch)

Figure 4c exhibits the results for all fine-tuned models. We can observe that both learning strategies boost the pre-trained model's performance with regards the explainability task, while maintaining similar values for decision task. However, only very few models were able to achieve better fraud recall than the base models. When comparing the performance of fine-tuned models with supervised baseline models (see Figure 4a), we observe not only substantial performance improvements in terms of fraud recall, but also significant increases at the explainability task.

Interestingly, we also observe two big clusters of models trained with the hybrid fine-tuning variant (green colored dots). Despite having the same range of values for decision task with values $[0.45, 0.8]$, these show two different ranges in terms of explainability performance: $[0.28, 0.4]$ and $[0.45, 0.6]$. This can be explained with the increase of the golden labels percentage in the batches, *i.e.*, the greater the golden labels' fraction in the batch, the higher the values at the explainability task.

In conclusion, comparing this strategy to the fully supervised approach (baseline), the two-stage learning strategy seems to be a strong proposal to tackle the concept-based explainability problem while using a multi-task learning approach.

4.3 HYBRID LEARNING

The last learning strategy concerns the creation of hybrid training batches that include both noisy and some pre-defined fraction of golden explainability labels. All models are trained from scratch using these hybrid batches. In this experiment, we used batches of which 10% of its size consisted of golden explainability labels. Figure 4d shows the obtained results. We find that most trained models achieve fraud recall values above 54%. When compared to the fully baseline (see Figure 4a), models trained with the hybrid learning strategy have a substantial improvement in decision task performance, while maintaining reasonable values for explainability.

Although these models achieve higher explainability values when compared to the two-stage pre-trained models (see Figure 4b), they attain lower values at the decision task (*i.e.*, lower fraud recall values). Moreover, when compared to fine-tuned models (see Figure 4c), hybrid learning models seem to perform worse at both tasks.

4.4 FINAL COMPARISON

In this section, we draw a comparison between each learning strategy in the test set. The results are shown in Figure 5. These seem to corroborate the idea that it is indeed possible to jointly learn a predictive (decision) task and the associated explanations. Moreover, when compared with the low-resources fully supervision learning strategy, both proposed strategies seem to lead to significant improvements in terms of decision performance at a reduced cost in the explainability performance. In fact, in the case of the two-stage learning strategy we observe that it is possible to improve performance at both tasks simultaneously.

The boost in the decision task over the baseline is of no surprise, since the baseline is restricted to use a very small dataset (with 842 transactions) with golden concept (or explainability) labels for both tasks. As a result, the model is not able to generalize well for unforeseen instances in the test set, thus presenting lower decision performance. On the other hand, when using weakly-labeled concepts we are able to bulk annotate massive golden decision datasets with the corresponding noisy explainability labels. Having a larger pool of golden labels, these models are able to generalize better.

Considering the hybrid learning strategy, this seems to yield consistently good trade-offs between explainability and decision tasks. On the other hand, the two-stage learning strategy spans a wider region of the explainability-accuracy solution space, reaching the best trade-offs in terms of decision task, as well as at the explainability task. Interestingly, we observe that the best trade-off[1] is achieved by a model, trained with the two-stage learning strategy and tuned with pure golden batches. This

[1]Due to business constraints, the best trade-off is the one that does not hurt performance too much and that attains reasonably high explainability performance.

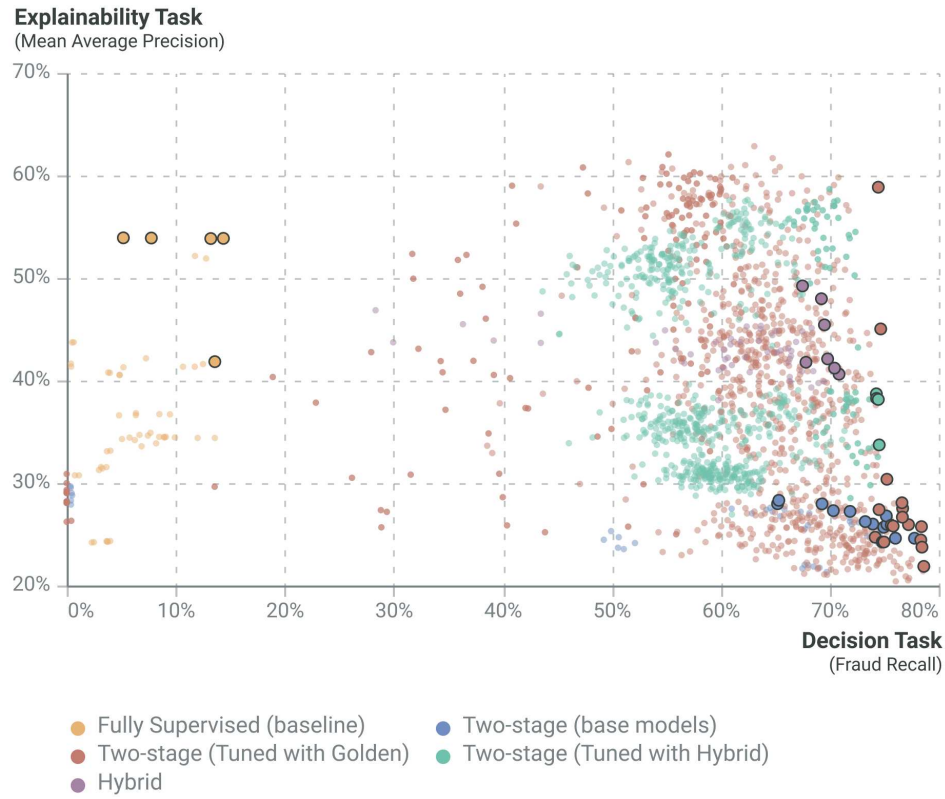


Figure 5: Explainability-Accuracy comparison between the learning strategies in the test set. These comprise the results of all random seeds. The larger-sized points represent optimal trade-offs.

model (top-right larger-sized orange point) achieves considerably high values in explainability, while mantaining high values at the decision task.

Finally, these results show that it is possible to learn more efficiently in a low-resources settings when using weak supervision to produce a higher-resources and noisier dataset. Moreover, depending on the importance of the two tasks, we conclude empirically that both proposed learning strategies can be used in practice with satisfactory results. Both two-stage and hybrid learning strategies improves significantly the performance on decision task at reduced (and so times at no) cost in explainability performance. We also argue that there is no one-size-fits-all learning strategy and that further experiments comparing the two proposed learning strategies are required (*e.g.*, use more random seeds, explore a wider region of the models' hyperparameter space, explore different golden label fractions for the hybrid strategy).

5 Conclusions

Concept-based explainability is particularly useful to explain the predictions of a black-box ML model to a non-technical, but domain expert, human-in-the-loop. A natural approach consists of training a (deep) neural network to jointly learn the predictions of a decision task and associated concepts. However, this approach faces two main challenges: concept label scarcity and the joint learning itself. We proposed to overcome these issues through the use of weak supervision approach that leverages available off-the-shelf information about expert rules to generate noisy concept labels. Furthermore, having access to a small set of golden concept labels, we devise two learning strategies to better exploit the different concept label signals (noisy and golden) during training. When comparing with the low-resources fully supervised approach, obtained results (in a e-commerce fraud detection use case) show it is possible to improve decision task performance with no (or very reduced) costs in the explainability task performance.

ACKNOWLEDGMENTS

The project CAMELOT (reference POCI-01-0247-FEDER-045915) leading to this work is co-financed by the ERDF - European Regional Development Fund through the Operational Program for Competitiveness and Internationalisation - COMPETE 2020, the North Portugal Regional Operational Program - NORTE 2020 and by the Portuguese Foundation for Science and Technology - FCT under the CMU Portugal international partnership. The authors would also like to thank Beatriz Malveiro and João Palmeiro for their help with graphics editing.

REFERENCES

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, pp. 9277–9286, 2019.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *35th International Conference on Machine Learning, ICML 2018*, 6:4186–4195, 2018.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/koh20a.html.

Scott M. Lundberg and Su In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 2017-Decem(Section 2):4766–4775, 2017. ISSN 10495258.

David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pp. 7775–7784, 2018.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 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*, pp. 1003–1011, 2009.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-August-2016:1135–1144, 2016. doi: 10.1145/2939672.2939778.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. ISSN 1939-3539. doi: 10.1109/tpami.2021.3054719. URL http://dx.doi.org/10.1109/TPAMI.2021.3054719.

Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime G. Carbonell. Characterizing and avoiding negative transfer. *CoRR*, abs/1811.09751, 2018. URL http://arxiv.org/abs/1811.09751.

Karl Weiss, Taghi Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3, 05 2016. doi: 10.1186/s40537-016-0043-6.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1753–1762, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1203. URL https://www.aclweb.org/anthology/D15-1203.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *CoRR*, abs/1707.08114, 2017. URL http://arxiv.org/abs/1707.08114.

PERVASIVE LABEL ERRORS IN TEST SETS DESTABILIZE MACHINE LEARNING BENCHMARKS

Curtis G. Northcutt
ChipBrain, MIT
Boston, MA, USA
curtis@chipbrain.com
cgn@mit.edu

Anish Athalye
Dept. of EECS
MIT
Cambridge, MA, USA
aathalye@mit.edu

Jonas Mueller
Amazon Web Services
East Palo Alto, CA, USA
jonaswmueller@gmail.com

ABSTRACT

We identify label errors in the *test* sets of 10 of the most commonly-used computer vision, natural language, and audio datasets, and subsequently study the potential for these label errors to affect benchmark results. Errors in test sets are numerous and widespread: we estimate an average of 3.4% errors across the 10 datasets,[1] where for example 2916 label errors comprise 6% of the ImageNet validation set. Putative label errors are identified using confident learning algorithms and then human-validated via crowdsourcing (54% of the algorithmically-flagged candidates are indeed erroneously labeled). Traditionally, machine learning practitioners choose which model to deploy based on test accuracy — our findings advise caution here, proposing that judging models over correctly labeled test sets may be more useful, especially for noisy real-world datasets. Surprisingly, we find that lower capacity models may be practically more useful than higher capacity models in real-world datasets with high proportions of erroneously labeled data. For example, on ImageNet with corrected labels: ResNet-18 outperforms ResNet-50 if the prevalence of originally mislabeled test examples increases by just 6%. On CIFAR-10 with corrected labels: VGG-11 outperforms VGG-19 if the prevalence of originally mislabeled test examples increases by just 5%.

1 INTRODUCTION

Large labeled data sets have been critical to the success of supervised machine learning across the board in domains such as image classification, sentiment analysis, and audio classification. Yet, the processes used to construct datasets often involve some degree of automatic labeling or crowdsourcing, techniques which are inherently error-prone (Sambasivan et al., 2021). Even with controls for error correction (Kremer et al., 2018; Zhang et al., 2017), errors can slip through. Prior work has considered the consequences of noisy labels, usually in the context of *learning* with noisy labels, and usually focused on noise in the *train* set. Some past research has concluded that label noise is not a major concern, because of techniques to learn with noisy labels (Patrini et al., 2017; Natarajan et al., 2013), and also because deep learning is believed to be naturally robust to label noise (Rolnick et al., 2017; Sun et al., 2017; Huang et al., 2019; Mahajan et al., 2018).

However, label errors in *test* sets are less-studied and have a different set of potential consequences. Whereas *train* set labels in a small number of machine learning datasets, e.g. in the ImageNet dataset, are well-known to contain errors (Northcutt et al., 2021; Shankar et al., 2020; Hooker et al., 2019), labeled data in *test* sets is often considered "correct" as long as it is drawn from the same distribution as the train set — this is a fallacy — machine learning *test* sets can, and do, contain pervasive errors and these errors can destabilize ML benchmarks.

Researchers rely on benchmark test datasets to evaluate and measure progress in the state-of-the-art and to validate theoretical findings. If label errors occurred profusely, they could potentially undermine the framework by which we measure progress in machine learning. Practitioners rely on their own real-world datasets which are often more noisy than carefully-curated benchmark datasets.

[1] To view the mislabeled examples in these benchmarks, go to https://labelerrors.com.

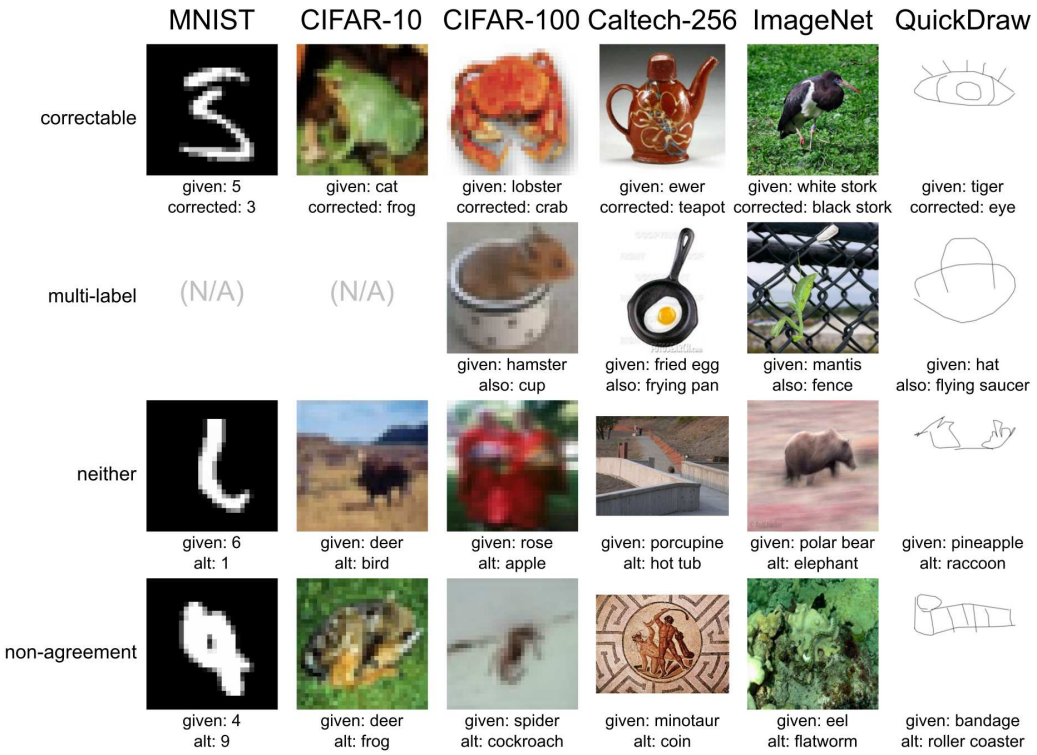


Figure 1: An example label error from each category (Sec. 4) for image datasets. The figure shows given labels, human-validated corrected labels, also the second label for multi-class data points, and CL-guessed alternatives. A browser for all label errors across all 10 datasets is available at https://labelerrors.com. Errors from text and audio datasets are also included on the website.

Label errors in these test sets could potentially lead practitioners to incorrect conclusions about which models actually perform best in the real world.

We present the first study that identifies and systematically analyzes label errors across 10 commonly-used datasets across computer vision, natural language processing, and audio processing. Unlike prior work on noisy labels, we do not experiment with synthetic noise but with naturally-occurring errors. Rather than exploring a novel methodology for dealing with label errors, which has been extensively studied in the literature (Cordeiro & Carneiro, 2020), this paper aims to characterize the prevalence of label errors in the test data of popular benchmarks used to measure ML progress, and we subsequently analyze practical consequences of these errors, and in particular, their effects on model selection. Using *confident learning* (Northcutt et al., 2021), we algorithmically identify putative label errors in test sets at scale [2], and we validate these label errors through human evaluation, estimating an average of 3.4% errors. We identify, for example, 2916 (6%) errors in the ImageNet validation set (which is *commonly used as a test set*), and estimate over 5 million (10%) errors in QuickDraw. Figure 1 shows examples of validated label errors for the image datasets in our study.

We use ImageNet and CIFAR-10 as case studies to understand the consequences of test set label errors on benchmark stability. While there are numerous erroneous labels in these benchmarks' test data, we find that relative rankings of models in benchmarks are unaffected after removing or correcting these label errors. However, we find that these benchmark results are *unstable*: higher-capacity models (like NasNet) undesirably reflect the distribution of systematic label errors in their predictions to a far greater degree than models with fewer parameters (like ResNet-18), and this effect *increases* with the prevalence of mislabeled test data. This is not traditional overfitting. Larger models are

[2]To find all label errors, we use the cleanlab implementation of confident learning open-sourced at: https://github.com/cgnorthcutt/cleanlab

able to generalize better to the given noisy labels in the test data, but this is problematic because these models produce *worse* predictions than their lower-capacity counterparts when evaluated on the corrected labels for mislabeled test examples.

In real-world settings with high proportions of erroneously labeled data, lower capacity models may thus be practically more useful than their higher capacity counterparts. For example, it may appear NasNet is superior to ResNet-18 based on the test accuracy over originally given labels, but NasNet is in fact worse than ResNet-18 based on the test accuracy over corrected labels. Since the latter form of accuracy is what matters in practice, ResNet-18 should actually be deployed instead of NasNet here – but this is unknowable without correcting the test data labels.

To evaluate how benchmarks of popular pre-trained models change, we incrementally increase the noise prevalence by controlling for the proportion of correctable (but originally mislabeled) data within the test dataset. This procedure allows us to measure the noise prevalence in each test set where benchmark rankings change. For example, on ImageNet with corrected labels: ResNet-18 outperforms ResNet-50 if the prevalence of originally mislabeled test examples increases by just 6%.

Our contributions include:

1. Using a simple algorithmic + crowdsourcing pipeline to identify and validate label errors, we discover label errors are pervasive in test sets of popular benchmarks used in nearly all machine learning research.
2. We provide a cleaned and corrected version of each test set [3], in which a large fraction of the label errors have been corrected by humans. We hope future research on these benchmarks will use this improved test data instead of the original erroneous labels.
3. We analyze the implications of pervasive test set label errors. We find that higher capacity models perform better on the subset of incorrectly-labeled test data in terms of their accuracy on the original labels (i.e., what one traditionally measures), but these models perform worse on this subset than their simpler counterparts in terms of their accuracy on corrected labels (i.e., what one cares about in practice, but cannot measure without the manually-corrected test data we provide).
4. In case studies with commonly-used benchmark datasets, we identify the prevalence of originally mislabeled test data needed to destabilize ML benchmarks, i.e., for low-capacity models to outperform high-capacity models. We discover that merely slight increases in the test label error prevalence would cause model selection to favor the wrong model based on standard test accuracy.

Our findings imply ML practitioners might benefit from correcting test set labels to benchmark how their models will perform in real-world deployment, and by using simpler/smaller models in applications where labels for their datasets tend to be noisier than the labels in gold-standard benchmark datasets. One way to ascertain whether a dataset is noisy enough to suffer from this effect is to correct at least the test set labels, e.g. using our straightforward approach.

2 Background and Related work

Experiments in learning with noisy labels (Patrini et al., 2016; Van Rooyen et al., 2015; Natarajan et al., 2013; Jindal et al., 2016; Sukhbaatar et al., 2015) suffer a double-edged sword: either synthetic noise must be added to clean training data to measure performance on a clean test set, at the expense of modeling *actual* real-world label noise (Jiang et al., 2020), or, a naturally noisy dataset is used and accuracy is measured on a noisy test set. In the noisy WebVision dataset (Li et al., 2017), accuracy on the ImageNet validation is often reported as a "clean" test set, however, related works (Recht et al., 2019; Northcutt et al., 2021; Tsipras et al., 2020; Hooker et al., 2019) have already shown the existence of label issues in ImageNet. Unlike these works, we focus exclusively on existence and implications of label errors in the test set, and extend our analysis to many types of datasets. Although extensive pior work deals with label errors in the *training* set (Frénay & Verleysen, 2014;

[3] A corrected version of each test set is provided at https://github.com/cgnorthcutt/label-errors.

Cordeiro & Carneiro, 2020), much less work has been done to understand the implications of label errors in the *test set*.

Crowd-sourced curation of labels via multiple human workers (Zhang et al., 2017; Dawid & Skene, 1979; Ratner et al., 2016) is a common method for validating/correcting label issues in datasets, but it can be exorbitantly expensive for large datasets. To circumvent this issue, we only validate subsets of datasets by first estimating which examples are most likely to be mislabeled. To achieve this, we lean on a number of contributions in uncertainty quantification for finding label errors based on prediction/label agreement via confusion matrices (Xu et al., 2019; Hendrycks et al., 2018; Chen et al., 2019; Lipton et al., 2018), however these approaches lack either robustness to class imbalance or theoretical support for realistic settings with *asymmetric, non-uniform noise*. For robustness to class imbalance and theoretical support for exact uncertainty quantification, we use the model-agnostic framework, confident learning (CL) (Northcutt et al., 2021), to estimate which labels are erroneous across diverse datasets. Northcutt et al. (2021) have demonstrated that CL more accurately identifies label errors than other label-error identification methods. Unlike prior work that only models symmetric label noise (Van Rooyen et al., 2015), we quantify class-conditional label noise with CL, validating the correctable nature of the label errors via crowdsourced workers. Human validation confirms the noise in common benchmark datasets is indeed primarily systematic mislabeling, not just random noise or lack of signal (e.g. images with fingers blocking the camera).

Datasets An overview of each dataset, how it was created, and any alterations we made for the experiments in this paper, is discussed in Appendix A.

3 IDENTIFYING LABEL ERRORS IN BENCHMARK DATASETS

Here we summarize our algorithmic label error identification performed prior to crowd-sourced human verification. The primary contribution of this section is not in the methodology, which is covered extensively in (Northcutt et al., 2021), but in its utilization as a *filtering* process to significantly (often as much as 90%) reduce the number of examples requiring human validation in the next step.

To identify label errors in a test dataset with n examples and m classes, we first characterize label noise in the dataset using the confident learning (CL) framework (Northcutt et al., 2021) to estimate $\boldsymbol{Q}_{\tilde{y},y^*}$, the $m \times m$ discrete joint distribution of observed, noisy labels, $\tilde{y}$, and unknown, true labels, y^*. Inherent in $\boldsymbol{Q}_{\tilde{y},y^*}$ is the assumption that noise is class-conditional (Angluin & Laird, 1988), depending only on the latent true class, not the data. This assumption is commonly used (Goldberger & Ben-Reuven, 2017; Sukhbaatar et al., 2015) because it is reasonable. For example, in ImageNet, a *tiger* is more likely to be mislabeled *cheetah* than *flute*.

The diagonal entry, $\hat{p}(\tilde{y}{=}i, y^*{=}i)$, of matrix $\boldsymbol{Q}_{\tilde{y},y^*}$ is the probability that examples in class i are correctly labeled. Thus, if the dataset is error-free, then $\sum_{i\in[m]} \hat{p}(\tilde{y}{=}i, y^*{=}i) = 1$. The fraction of label errors is $\rho = 1 - \sum_{i\in[m]} \hat{p}(\tilde{y}{=}i, y^*{=}i)$ and the number of label errors is $\rho \cdot n$. To find label errors, we choose the top $\rho \cdot n$ examples ordered by the normalized margin: $\hat{p}(\tilde{y}{=}i; \boldsymbol{x}, \boldsymbol{\theta}) - \max_{j\neq i} \hat{p}(\tilde{y}{=}j; \boldsymbol{x}, \boldsymbol{\theta})$ (Wei et al., 2018). Table 1 shows the number of CL guessed label issues for each test set across ten popular ML benchmark datasets. Confident learning estimation of $\boldsymbol{Q}_{\tilde{y},y^*}$ is summarized in the appendices in (Sec. B).

Computing out-of-sample predicted probabilities Estimating $\boldsymbol{Q}_{\tilde{y},y^*}$ for CL noise characterization requires two inputs for each dataset: (1) out-of-sample predicted probabilities $\hat{\boldsymbol{P}}_{k,i}$ ($n \times m$ matrix) and (2) the test set labels $\tilde{y}_k$. We observe the best results computing $\hat{\boldsymbol{P}}_{k,i}$ by pre-training on the train set, then fine-tuning (all layers) on the test set using cross-validation to ensure $\hat{\boldsymbol{P}}_{k,i}$ is out-of-sample. If pre-trained models are open-sourced (e.g. ImageNet), we use them instead of pre-training ourselves. If the dataset did not have an explicit test set (e.g. QuickDraw and Amazon Reviews), we skip pre-training, and compute $\hat{\boldsymbol{P}}_{k,i}$ using cross-validation on the entire dataset. For all datasets, we try common models that achieve reasonable accuracy with minimal hyper-parameter tuning, and use the model yielding the highest cross-validation accuracy, reported in Table 1.

Using this approach allows us to find label errors without manually checking the entire test set, because CL identifies potential label errors automatically.

Table 1: Test set errors are prominent across common benchmark datasets. Errors are estimated using confident learning (CL) and validated by human workers on Mechanical Turk.

Dataset	Modality	Size	Model	Test Set Errors				
				CL guessed	MTurk checked	validated	estimated	% error
MNIST	image	10,000	2-conv CNN	100	100 (100%)	15	-	0.15
CIFAR-10	image	10,000	VGG	275	275 (100%)	54	-	0.54
CIFAR-100	image	10,000	VGG	2235	2235 (100%)	585	-	5.85
Caltech-256	image	30,607	ResNet-152	4,643	400 (8.6%)	65	754	2.46
ImageNet*	image	50,000	ResNet-50	5,440	5,440 (100%)	2,916	-	5.83
QuickDraw	image	50,426,266	VGG	6,825,383	2,500 (0.04%)	1870	5,105,386	10.12
20news	text	7,532	TFIDF + SGD	93	93 (100%)	82	-	1.11
IMDB	text	25,000	FastText	1,310	1,310 (100%)	725	-	2.9
Amazon	text	9,996,437	FastText	533,249	1,000 (0.2%)	732	390,338	3.9
AudioSet	audio	20,371	VGG	307	307 (100%)	275	-	1.35

*Because the ImageNet test set labels are not publicly available, the ILSVRC 2012 validation set is used.

Table 2: Mechanical Turk validation confirming the existence of pervasive label errors and categorizing the types of label issues.

Dataset	Test Set Errors Categorization					
	non-errors	errors	non-agreement	correctable	multi-label	neither
MNIST	85	15	2	10	-	3
CIFAR-10	221	54	32	18	0	4
CIFAR-100	1650	585	210	318	20	37
Caltech-256	335	65	25	22	5	13
ImageNet	2524	2916	598	1428	597	293
QuickDraw	630	1870	563	1047	20	240
20news	11	82	43	22	12	5
IMDB	585	725	552	173	-	-
Amazon	268	732	430	302	-	-
AudioSet	32	275	-	-	-	-

4 VALIDATING LABEL ERRORS

We validated the algorithmically identified label errors with a Mechanical Turk study. For three datasets with a large number of errors (Caltech-256, QuickDraw, and Amazon Reviews), we checked a random sample; for the rest, we checked all identified errors.

We presented workers with hypothesized errors and asked them whether they saw the (1) given label, (2) the top CL-predicted label, (3) both labels, or (4) neither label in the example. To aid the worker, the interface included high-confidence examples drawn from the training set of the given class and the CL-predicted class. Figure 2 shows the Mechanical Turk worker interface, showing a data point from the CIFAR-10 dataset.

Each CL-identified label error was independently presented to five workers. We consider the example validated (an "error") if fewer than three of the workers agree that the data point has the given label (*agreement threshold = 3 of 5*) , otherwise we consider it to be a "non-error" (i.e. the original label was correct). We further categorize the label errors, breaking them down into (1) "correctable", where a majority agree on the CL-predicted label; (2) "multi-label", where a majority agree on both labels appearing; (3) "neither", where a majority agree on neither label appearing; and (4) "non-agreement", a catch-all category for when there is no majority. Table 2 summarizes the results, and Figure 1 shows examples of validated label errors from image datasets.

5 IMPLICATIONS OF LABEL ERRORS IN TEST DATA

Finally, we consider how these pervasive test set label errors may affect ML practice in real-world applications. To clarify the discussion, we first introduce some useful terminology.

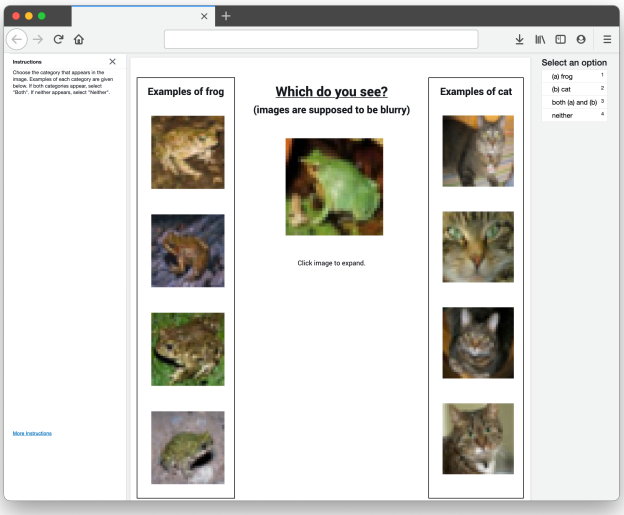


Figure 2: Mechanical Turk worker interface showing an example from CIFAR-10 (with given label "cat"). For each data point algorithmically identified as a potential label error, the interface presents the data point, along with examples belonging to the given class. The interface also shows data points belonging to the confidently predicted class. Either the given is shown as option (a) and predicted is shown as option (b), or vice versa (chosen randomly). The worker is asked whether the image belongs to class (a), (b), both, or neither.

Definition 1 (original accuracy, $\tilde{A}$)**.** *The accuracy of a model's predicted labels over a given dataset, computed with respect to the original labels present in the dataset. Measuring this over the test set is the standard way practitioners evaluate their models today.*

Definition 2 (corrected accuracy, A^*)**.** *The accuracy of a model's predicted labels, computed with respect to a new version of the given dataset in which previously identified erroneous labels have been corrected through human revision to the true class when possible and removed when not. Measuring this over the test set is preferable to $\tilde{A}$ for evaluating models (because A^* better reflects performance in real-world applications).*

In the following definitions, "$\setminus$" denotes a set difference, $\mathcal{D}$ denotes the full test dataset, and $\mathcal{B} \subset \mathcal{D}$ denotes the subset of benign test examples that CL did *not* flag as likely label errors.

Definition 3 (unknown-label set, $\mathcal{U}$)**.** *The subset of CL-flagged test examples for which human labelers could not agree on a correct label ($\mathcal{U} \subset \mathcal{D} \setminus \mathcal{B}$). This includes examples where human reviewers agreed that multiple classes or none of the classes are appropriate.*

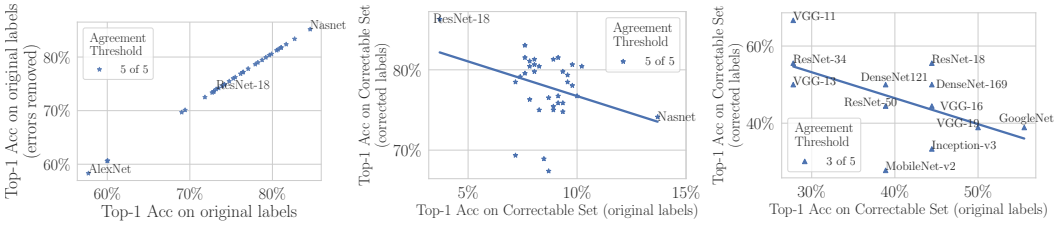
Definition 4 (pruned set, $\mathcal{P}$)**.** *The remaining test data after removing $\mathcal{U}$ from $\mathcal{D}$ ($\mathcal{P} = \mathcal{D} \setminus \mathcal{U}$).*

Definition 5 (correctable set, $\mathcal{C}$)**.** *The subset of CL-flagged examples for which human-validation reached consensus on a different label than the originally given label ($\mathcal{C} = \mathcal{P} \setminus \mathcal{B}$).*

Definition 6 (noise prevalence, N)**.** *The percentage of the* pruned set *comprised of the* correctable set*, i.e. what fraction of data received the wrong label in the original benchmark when a clear alternative ground-truth label should have been assigned (disregarding any data for which humans failed to find a clear alternative). Here we operationalize noise prevalence as $N = \frac{|\mathcal{C}|}{|\mathcal{P}|}$.*

These definitions imply $\mathcal{B}, \mathcal{C}, \mathcal{U}$ are disjoint with $\mathcal{D} = \mathcal{B} \cup \mathcal{C} \cup \mathcal{U}$, and also $\mathcal{P} = \mathcal{B} \cup \mathcal{C}$. In subsequent experiments, we report corrected test accuracy over $\mathcal{P}$ after correcting all of the labels in $\mathcal{C} \subset \mathcal{P}$. We ignore the unknown-label set $\mathcal{U}$ (and no longer include those examples in our estimate of noise prevalence) because it is unclear how to measure *corrected accuracy* for examples whose true underlying label remains ambiguous. Thus the *noise prevalence* reported throughout this section differs from the fraction of label errors originally found in each of the test sets.

A major issue in ML today is that one only sees the original test accuracy in practice, whereas one would prefer to base modeling decisions on the corrected test accuracy instead. Our subsequent discussion highlights the potential implications of this mismatch. What are the consequences of test



(a) ImageNet Val set Acc. (b) ImageNet Correctable set Acc. (c) CIFAR-10 Correctable set Acc.

Figure 3: Benchmark ranking comparison of 34 models pre-trained on ImageNet and 13 pre-trained on CIFAR-10 (more details in Tables S2 and S1 and Fig. S1, in the Appendix). Benchmarks are unchanged by removing label errors (a), but change drastically on the Correctable set with original (erroneous) labels versus corrected labels, e.g. Nasnet: 1/34 $\rightarrow$ 29/34, ResNet-18: 34/34 $\rightarrow$ 1/34.

set label errors? Figure 3 compares performance on the ImageNet validation set, *commonly used in place of the test set*, of 34 pre-trained models from the PyTorch and Keras repositories. Figure 3a confirms the observations of Recht et al. (2019); benchmark conclusions are largely unchanged by using a corrected test set, i.e. in our case by removing errors.

However, we find a surprising result upon closer examination of the models' performance *on the erroneously labeled data*, which we call the "correctable set" $\mathcal{C}$. When evaluating models *only* on the subset of test examples in $\mathcal{C}$, models which perform best on the original (incorrect) labels perform the worst on corrected labels. For example, ResNet-18 (He et al., 2016) significantly outperforms NasNet (Zoph et al., 2018) in terms of corrected accuracy over $\mathcal{C}$, despite exhibiting far worse original test accuracy. The change in ranking can be dramatic: Nasnet-large drops from ranking 1/34 $\rightarrow$ 29/34, Xception drops from ranking 2/34 $\rightarrow$ 24/34, ResNet-18 increases from ranking 34/34 $\rightarrow$ 1/34, and ResNet-50 increases from ranking 20/24 $\rightarrow$ 2/24 (see Table S1 in the Appendices). We verified that the same trend occurs independently across 13 models pre-trained on CIFAR-10 (Figure 3c), e.g. VGG-11 significantly outperforms VGG-19 (Simonyan & Zisserman, 2014) in terms of corrected accuracy over $\mathcal{C}$. Note that all numbers reported here are over subsets of the held-out test data, so this is not overfitting in the classical sense.

This phenomenon, depicted in Figures 3b and 3c, may indicate two key insights: (1) lower-capacity models provide unexpected regularization benefits and are more resistant to learning the asymmetric distribution of noisy labels, (2) over time, the more recent (larger) models have architecture/hyperparameter decisions that were made on the basis of the (original) test accuracy. Learning to capture systematic patterns of label error in their predictions allows these models to improve their original test accuracy, but this is not the sort of progress ML research should aim to achieve. Harutyunyan et al. (2020); Arpit et al. (2017) have previously analyzed phenomena similar to (1), and here we demonstrate that this issue really does occur for the models/datasets widely used in current practice. (2) is an undesirable form of overfitting, albeit not in the classical sense (as the original test accuracy can further improve through better modeling of label errors), but rather overfitting to the specific benchmark (and quirks of the original label annotators) such that accuracy improvements for erroneous labels may not translate to superior performance in a deployed ML system.

This phenomenon has important practical implications for real-world datasets with greater noise prevalence than the highly curated benchmark data studied here. In these relatively clean benchmark datasets, the noise prevalence is an underestimate as we could only verify a subset of our candidate label errors rather than all test labels, and thus the potential gap between original vs. corrected test accuracy is limited for these particular benchmarks. However, this gap increases proportionally for data with more (correctable) label errors in the test set.

To evaluate how benchmarks of popular pre-trained models change, we randomly and incrementally remove correctly-labeled examples, one at a time, until only the original set of mislabeled test data (with corrected labels) is left. We create alternate versions (subsets) of the pruned benchmark test data $\mathcal{P}$, in which we additionally randomly omit some fraction, x, of $\mathcal{B}$ (the test examples that were not identified to have label errors). This effectively increases the proportion of the resulting test dataset comprised of the correctable set $\mathcal{C}$, and reflects how test sets function in applications with greater prevalence of label errors. If we remove a fraction x of benign test examples (in $\mathcal{B}$) from $\mathcal{P}$, we estimate the noise prevalence in the new (reduced) test dataset to be $N = \frac{|\mathcal{C}|}{|\mathcal{P}|-x|\mathcal{B}|}$. By varying

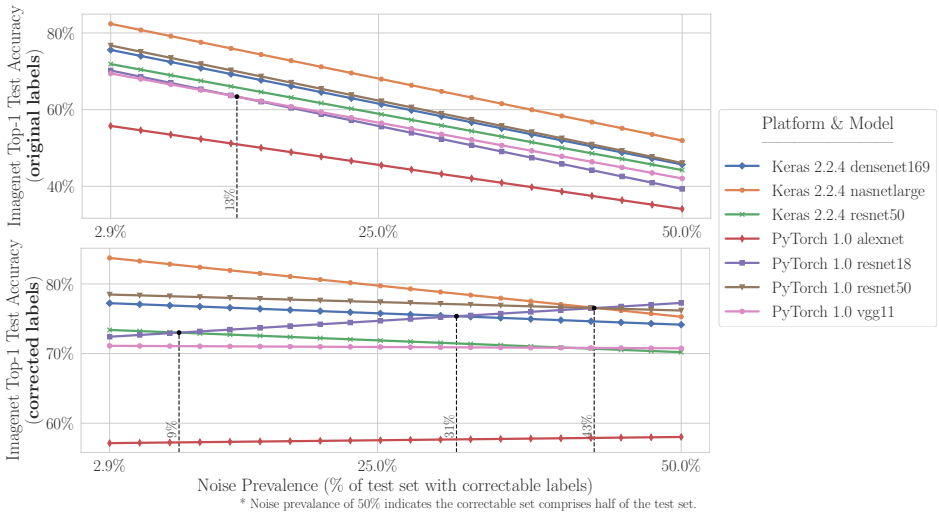


Figure 4: ImageNet top-1 original accuracy (top panel) and corrected accuracy (bottom panel) vs Noise Prevalence (agreement threshold = 3). Vertical lines indicate noise levels at which the ranking of two models changes (in terms of original/corrected accuracy). The left-most point ($N = 2.9\%$) on the x-axis is $|\mathcal{C}|/|\mathcal{P}|$, i.e. the (rounded) estimated noise prevalence of the pruned set, $\mathcal{P}$. The leftmost vertical dotted line in the bottom panel is read, "The Resnet-50 and Resnet-18 benchmarks cross at noise prevalence $N = 8.6\%$, implying Resnet-18 outperforms Resnet-50 when N increases by around 6% relative to the original pruned test data ($N = 2.9\%$ originally, c.f. Table 2).

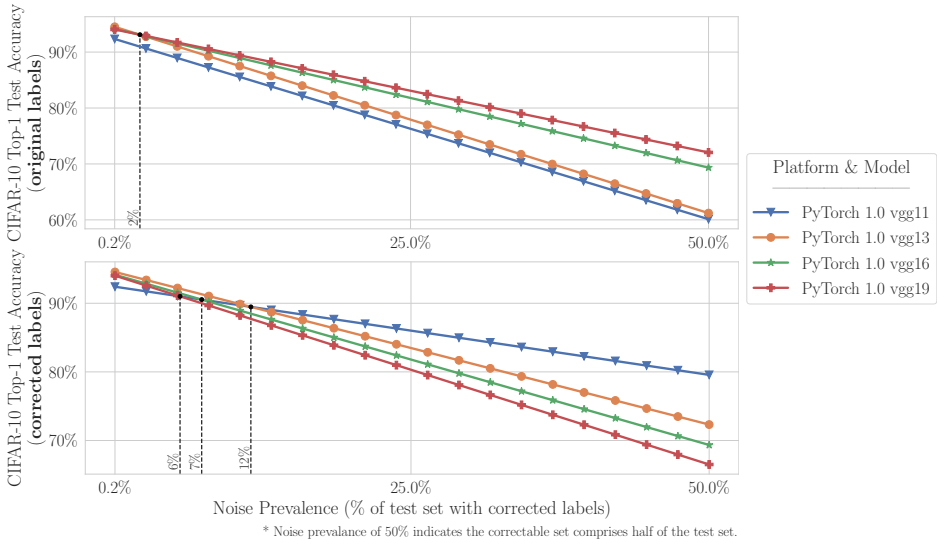


Figure 5: CIFAR-10 top-1 original accuracy (top panel) and corrected accuracy (bottom panel) vs Noise Prevalence (agreement threshold = 3). For additional details, see the caption of Fig. 4.

x from 0 to 1, we can simulate any noise prevalence ranging from $|\mathcal{C}|/|\mathcal{P}|$ to 1. We operationalize averaging over all choices of removal by linearly interpolating from benchmark accuracies on the corrected test set ($\mathcal{P}$, with corrected labels for the subset $\mathcal{C}$) to accuracies on the erroneously labeled subset ($\mathcal{C}$, with corrected labels).

For a given model, $\mathcal{M}$, its resulting accuracy (as a function of x) over the reduced test data is given by $A(x;\mathcal{M}) = \frac{A_{\mathcal{C}}(\mathcal{M})\cdot|\mathcal{C}|+(1-x)\cdot A_{\mathcal{B}}(\mathcal{M})\cdot|B|}{|\mathcal{C}|+(1-x)\cdot|B|}$, where $A_{\mathcal{C}}(\mathcal{M})$ and $A_{\mathcal{B}}(\mathcal{M})$ denote the (original or corrected) accuracy over the correctable set and benign set, respectively (accuracy before removing any examples). Here $A_{\mathcal{B}} = A^*_{\mathcal{B}} = \tilde{A}_{\mathcal{B}}$ because no erroneous labels were identified in $\mathcal{B}$. The expectation is taken over which fraction x of examples are randomly removed from $\mathcal{B}$ to produce the

reduced test set: the resulting expected accuracy, $A(x; \mathcal{M})$, is depicted on the y-axis of Figures 4-5. As our removal of test examples was random from the non-mislabeled set, we expect this reduced test data is representative of test sets that would be used in applications with a similarly greater prevalence of label errors. Note that we ignore non-correctable data with unknown labels ($\mathcal{U}$) throughout this analysis, as it is unclear how to report a better version of the accuracy for such ill-specified examples.

Over alternative (reduced) test sets created by imposing increasing degrees of noise prevalence in ImageNet/CIFAR-10, Figures 4-5 depict the resulting original (erroneous) test set accuracy and corrected accuracy of the models, expected on each alternative test set. For a given test set (i.e. point along the x-axis of these plots), the vertical ordering of the lines indicates how models would be favored based on original accuracy or corrected accuracy over this test set. Unsurprisingly, we see that more flexible/recent architectures tend to be favored on the basis of original accuracy, regardless of which test set (of varying noise prevalence) is considered. This aligns with conventional expectations that powerful models like NasNet will outperform simpler models like ResNet-18. However, if we shift our focus to the corrected accuracy (i.e. what actually matters in practice), it is no longer the case that more powerful models are reliably better than their simpler counterparts: the performance strongly depends on the degree of noise prevalence in the test data. For datasets where label errors are common, a practitioner is more likely to select a model (based on original accuracy) that is not actually the best model (in terms of corrected accuracy) to deploy.

Finally, we note that this analysis only presents a loose lower bound on the magnitude of these issues. We only identified a subset of the actual correctable set as we are limited to human-verifiable label corrections for a subset of data candidates (algorithmically prioritized via confident learning). Because the actual correctable sets are likely larger, our noise prevalence estimates are optimistic in favor of higher capacity models. Thus, the true gap between between corrected vs. original accuracy may be larger and of greater practical significance, even for the gold-standard benchmark datasets considered here. For many application-specific datasets collected by ML practitioners, the noise prevalence will be greater than the numbers presented here: thus, it is imperative to be cognizant of the distinction between corrected vs. original accuracy, and to utilize careful data curation practices, perhaps by allocating more of an annotation budget to ensure higher quality labels in the test data.

6 Conclusion

Traditionally, ML practitioners choose which model to deploy based on test accuracy — our findings advise caution here, proposing that judging models over correctly labeled test sets may be more useful, especially for noisy real-world datasets. Small increases in the prevalence of originally mislabeled test data can destabilize ML benchmarks, indicating that low-capacity models may actually outperform high-capacity models in noisy real-world applications, even if their measured performance on the original test data may be worse. This gap increases as the prevalence of originally mislabeled test data increases. It is imperative to be cognizant of the distinction between corrected vs. original test accuracy, and to follow dataset curation practices that maximize high-quality test labels, even if budget constraints limit you to lower-quality training labels.

This paper shares new findings about pervasive label errors in test sets and their effects on benchmark stability, but does not address whether the apparent overfitting of high-capacity models versus low-capacity models is due to overfitting to train set noise, overfitting to validation set noise during hyper-parameter tuning, or heightened sensitivity to train/test label distribution shift that occurs when test labels are corrected. An intuitive hypothesis is that high-capacity models more closely fit all statistical patterns present in the data, including those patterns related to systematic label errors that models with more limited capacity are less capable of closely approximating. A rigorous analysis to disambiguate and understand the contribution of each of these causes and their effects on benchmarking stability is a natural next step, which we leave for future work. How to best allocate a given human relabeling budget between training and test data also remains an open question.

Acknowledgements

This work was supported in part by funding from the MIT-IBM Watson AI Lab.

REFERENCES

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning (ICML)*, 2019.

F. R. Cordeiro and G. Carneiro. A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations? In *Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 9–16, 2020.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28 (1):20–28, 1979.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014. ISSN 21622388. doi: 10.1109/TNNLS.2013.2292894.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, LA, 2017.

Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations (ICLR)*, 2017.

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. URL http://authors.library.caltech.edu/7694.

Patrick J Grother. Nist special database 19 handprinted forms and characters database. *National Institute of Standards and Technology*, 1995.

David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

Hrayr Harutyunyan, Kyle Reing, Greg Ver Steeg, and Aram Galstyan. Improving generalization by controlling label-noise information in neural network weights. In *International Conference on Machine Learning*, pp. 4071–4081. PMLR, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Sara Hooker, Aaron Courville, Yann Dauphin, and Andrea Frome. Selective brain damage: Measuring the disparate impact of model pruning. *arXiv preprint arXiv:1911.05248*, 2019.

W Ronny Huang, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations. *arXiv preprint arXiv:1906.03291*, 2019.

Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In Hal Daumé III and Aarti Singh (eds.), *ICML*, volume 119 of *PMLR*, pp. 4804–4815. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/jiang20c.html.

Ishan Jindal, Matthew Nokleby, and Xuewen Chen. Learning deep networks from noisy labels with dropout regularization. In *International Conference on Data Mining (ICDM)*, 2016.

Jan Kremer, Fei Sha, and Christian Igel. Robust active label correction. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of Machine Learning Research (PMLR)*, volume 84 of *Proceedings of Machine Learning Research*, pp. 308–316, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR. URL http://proceedings.mlr.press/v84/kremer18a.html.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv:1708.02862*, 2017.

Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International Conference on Machine Learning (ICML)*, 2018.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Annual Conference of the Association for Computational Linguistics (ACL)*, pp. 142–150, Portland, Oregon, USA, June 2011. Annual Conference of the Association for Computational Linguistics (ACL). URL http://www.aclweb.org/anthology/P11-1015.

D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. *ArXiv*, May 2018.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Special Interest Group on Information Retrieval (SIGIR)*, pp. 43–52, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3621-5. doi: 10.1145/2766462.2767755. URL http://doi.acm.org/10.1145/2766462.2767755.

Tom Mitchell. Twenty newsgroups dataset, 1999. URL https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2013.

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research (JAIR)*, 2021.

Giorgio Patrini, Frank Nielsen, Richard Nock, and Marcello Carioni. Loss factorization, weakly supervised learning and label noise robustness. In *International Conference on Machine Learning (ICML)*, pp. 708–717, 2016.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning (ICML)*, pp. 5389–5400, 2019.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv:1705.10694*, 2017.

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. "Everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *Human Factors in Computing Systems (CHI)*, 2021.

Vaishaal Shankar, Rebecca Roelofs, Horia Mania, Alex Fang, Benjamin Recht, and Ludwig Schmidt. Evaluating machine accuracy on ImageNet. In Hal Daumé III and Aarti Singh (eds.), *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8634–8644. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/shankar20c.html.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. In *International Conference on Learning Representations (ICLR)*, 2015.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era, 2017. URL https://arxiv.org/pdf/1707.02968.pdf.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. From imagenet to image classification: Contextualizing progress on benchmarks. In *International Conference on Machine Learning*, pp. 9625–9635. PMLR, 2020.

Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. On the margin theory of feedforward neural networks. *arXiv:1810.05369*, 2018.

Wikipedia contributors. List of datasets for machine learning research — wikipedia, the free encyclopedia, 2020. URL https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research. Online; accessed 22-October-2018.

Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 6225–6236. Curran Associates, Inc., 2019.

Jing Zhang, Victor S Sheng, Tao Li, and Xindong Wu. Improving crowdsourced label quality using noise correction. *IEEE transactions on neural networks and learning systems*, 29(5):1675–1688, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

APPENDIX:PERVASIVE LABEL ERRORS IN TEST SETS DESTABILIZE MACHINE LEARNING BENCHMARKS

A DATASETS

We select 10 of the most-cited, open-source datasets created in the last 20 years from the Wikipedia List of ML Research Datasets (Wikipedia contributors, 2020), with preference for diversity across computer vision, NLP, sentiment analysis, and audio modalities. Citation counts were obtained via the Microsoft Cognitive API. In total, we evaluate six visual datasets: MNIST, CIFAR-10, CIFAR-100, Caltech-256, ImageNet, and QuickDraw; three text datasets: 20news, IMDB, and Amazon Reviews; and one audio dataset: AudioSet.

A.1 DATASET DETAILS

For each of the datasets we investigate, we summarize the original data collection and labeling procedure as they pertain to potential label errors.

MNIST (Lecun et al., 1998). MNIST is a database of binary images of handwritten digits. The dataset was constructed from Handwriting Sample Forms distributed to Census Bureau employees and high school students; the ground-truth labels were determined by matching digits to the instructions of the task in order to copy a particular set of digits (Grother, 1995). Label errors may arise from failure to follow instructions or from handwriting ambiguities.

CIFAR-10 / CIFAR-100 (Krizhevsky, 2009). The CIFAR-10 and CIFAR-100 datasets are collections of small 32×32 images and labels from a set of 10 or 100 classes, respectively. The images were collected by searching the internet for the class label. Human labelers were instructed to select images that matched their class label (query term) by filtering out mislabeled images. Images were intended to only have one prominent instance of the object, but could be partially occluded as long as it was identifiable to the labeler.

Caltech-256 (Griffin et al., 2007). Caltech-256 is a database of images and classes. Images were scraped from image search engines. Four human labelers were instructed to rate the images into "good," "bad," and "not applicable," eliminating the images that were confusing, occluded, cluttered, artistic, or not an example of the object category from the dataset.

ImageNet (Deng et al., 2009). ImageNet is a database of images and classes. Images were scraped by querying words from WordNet "synonym sets" (synsets) on several image search engines. The images were labeled by Amazon Mechanical Turk workers who were asked whether each image contains objects of a particular given synset. Workers were instructed to select images that contain objects of a given subset regardless of occlusions, number of objects, and clutter to "ensure diversity" in the dataset's images.

QuickDraw (Ha & Eck, 2017). The Quick, Draw! dataset contains more than 1 billion doodles collected from users of an experimental game to benchmark image classification models. Users were instructed to draw pictures corresponding to a given label, but the drawings may be "incomplete or may not match the label." Because no explicit test set is provided, we study label errors in the entire dataset to ensure coverage of any test set split used by practitioners.

20news (Mitchell, 1999). The 20 Newsgroups dataset is a collection of articles posted to Usenet newsgroups used to benchmark text classification and clustering models. The label for each example is the newsgroup it was originally posted in (e.g. "misc.forsale"), so it is obtained during the overall data collection procedure.

IMDB (Maas et al., 2011). The IMDB Large Movie Review Dataset is a collection of movie reviews to benchmark binary sentiment classification. The labels were determined by the user's review: a score ≤ 4 out of 10 is considered negative; ≥ 7 out of 10 is considered positive.

Amazon Reviews (McAuley et al., 2015). The Amazon Reviews dataset is a collection of textual reviews and 5-star ratings from Amazon customers used to benchmark sentiment analysis models. We use the 5-core (9.9 GB) variant of the dataset. **Modifications**: In our study, 2-star and 4-star reviews are removed due to ambiguity with 1-star and 5-star reviews, respectively. If these reviews were left in the dataset, they could inflate error counts. Because no explicit test set is provided, we study label errors in the entire dataset to ensure coverage of any test set split used by practitioners.

AudioSet (Gemmeke et al., 2017). AudioSet is a collection of 10-second sound clips drawn from YouTube videos and multiple labels describing the sounds that are present in the clip. Three human labelers independently rated the presence of one or more labels (as "present," "not present," and "unsure"), and majority agreement was required to assign a label. The authors note that spot checking revealed some label errors due to "confusing labels, human error, and difference in detection of faint/non-salient audio events."

B Details of Confident Learning (CL) for finding label errors

Here we summarize CL joint estimation and how it is used to algorithmically flag candidates with likely label errors for subsequent human review. An unnormalized representation of the joint distribution between observed and true label, called the *confident joint* and denoted $\boldsymbol{C}_{\tilde{y},y^*}$, is estimated by counting all the examples with noisy label $\tilde{y}=i$, with high probability of actually belonging to label $y^*=j$. This binning can be expressed as:

$$\boldsymbol{C}_{\tilde{y},y^*} = |\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i}:\ \hat{p}(\tilde{y}=j;\boldsymbol{x},\boldsymbol{\theta}) \geq t_j\}|$$

where $\boldsymbol{x}$ is a data example (e.g. an image), $\boldsymbol{X}_{\tilde{y}=i}$ is the set of examples with noisy label $\tilde{y}=i$, $\hat{p}(\tilde{y}=j;\boldsymbol{x},\boldsymbol{\theta})$ is the out-of-sample predicted probability that example $\boldsymbol{x}$ actually belongs to noisy class $\tilde{y}=j$ (even though its given label $\tilde{y}=i$) for a given model $\boldsymbol{\theta}$. Finally, t_j is a per-class threshold that, in comparison to other confusion matrix approaches, provides robustness to heterogeneity in class distributions and class distributions, defined as:

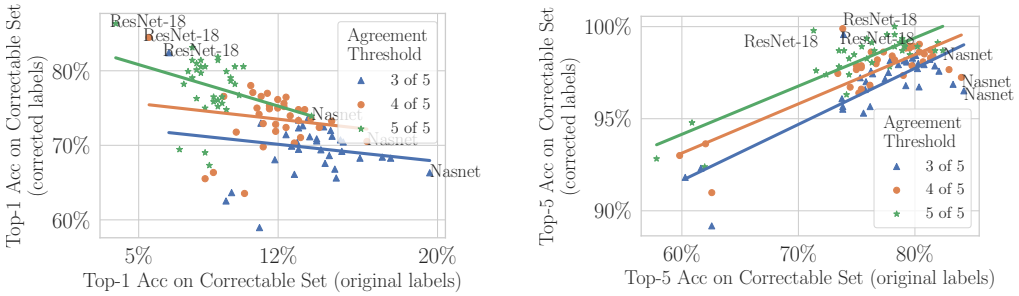
$$t_j = \frac{1}{|\boldsymbol{X}_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y}=j;\boldsymbol{x},\boldsymbol{\theta}) \tag{1}$$

A caveat occurs when an example is confidently counted into more than one bin. When this occurs, the example is only counted in the $\arg\max_{l \in [m]} \hat{p}(\tilde{y}=l;\boldsymbol{x},\boldsymbol{\theta})$ bin.

$\boldsymbol{Q}_{\tilde{y},y^*}$ is estimated by normalizing $\boldsymbol{C}_{\tilde{y},y^*}$, as follows:

$$\hat{\boldsymbol{Q}}_{\tilde{y}=i,y^*=j} = \frac{\frac{\boldsymbol{C}_{\tilde{y}=i,y^*=j}}{\sum_{j \in [m]} \boldsymbol{C}_{\tilde{y}=i,y^*=j}} \cdot |\boldsymbol{X}_{\tilde{y}=i}|}{\sum_{i \in [m], j \in [m]} \left(\frac{\boldsymbol{C}_{\tilde{y}=i,y^*=j}}{\sum_{j \in [m]} \boldsymbol{C}_{\tilde{y}=i,y^*=j}} \cdot |\boldsymbol{X}_{\tilde{y}=i}| \right)} \tag{2}$$

The numerator calibrates $\sum_j \hat{\boldsymbol{Q}}_{\tilde{y}=i,y^*=j} = |\boldsymbol{X}_i| / \sum_{i \in [m]} |\boldsymbol{X}_i|, \forall i \in [m]$ so that row-sums match the observed prior over noisy labels. The denominator makes the distribution sum to 1.



(a) Top-1 Accuracy. (b) Top-5 Accuracy.

Figure S1: Benchmark ranking comparison of 34 pre-trained models on the ImageNet val set (used as test data here) for various settings of the agreement threshold. Top-5 benchmarks are unchanged by removing label errors (a), but change drastically on the correctable subset with original (erroneous) labels versus corrected labels. Corrected test set sizes: 1428 (▲), 960 (●), 468 (⋆).

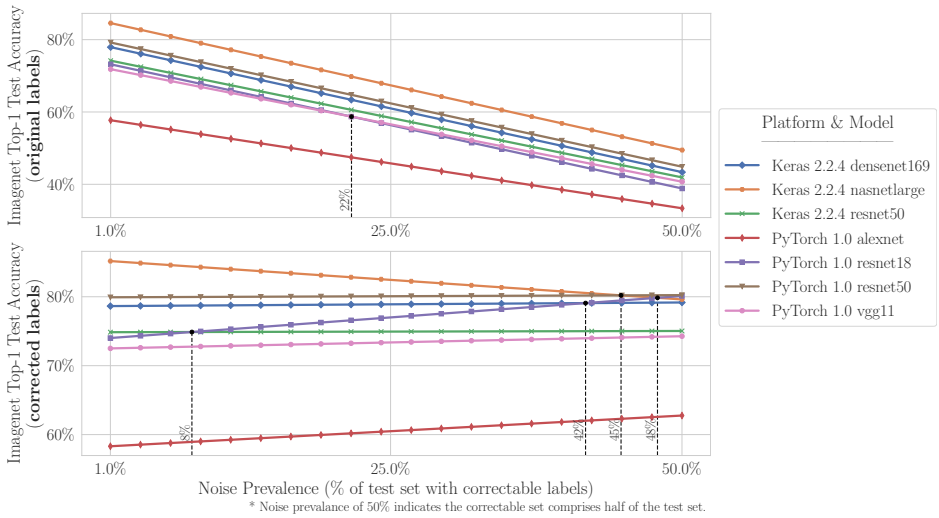


Figure S2: ImageNet top-1 original accuracy (top panel) and top-1 corrected accuracy (bottom panel) vs Noise Prevalence with agreement threshold = 5 (instead of threshold = 3, c.f., Fig. 4).

C CASE STUDY BENCHMARKING DETAILS

Figure 3 depicts how the benchmarking rankings on the correctable subset of ImageNet examples change significantly for an *agreement threshold* $= 5$, meaning 5 of 5 human raters need to independently select the same alternative label for that data point and a new label to be included in the accuracy evaluation. To ascertain that the results of this figure are not due to the setting of the agreement threshold, the results for all three settings of the agreement threshold are shown in Sub-figure S1b. Observe the negative correlation (for top-1 accuracy) occurs in all three settings. Furthermore, observe that this negative correlation no longer holds when top-5 accuracy is used (shown in S1a), likely because many of these models use a loss which maximizes (and overfits to noise) based on top-1 accuracy, not top-5 accuracy. Regardless of whether top-1 or top-5 accuracy is used, model benchmark rankings change significantly on the correctable set in comparison to the original test set (see Table S1).

The dramatic changes in ranking shown in Table S1 may be explained by overfitting to the validation when these models are trained, which can occur inadvertently during hyper-parameter tuning, or by overfitting to the noise in the training set. The benchmarking experiment was replicated on CIFAR-10 in addition to ImageNet. The individual accuracies for CIFAR-10 are reported in Table S2.

Table S1: Individual accuracy scores for Sub-figure 3b with *agreement threshold = 3 of 5*. Acc@1 stands for the (top-1 validation) original accuracy on the correctable set, in terms of original ImageNet examples and labels. *cAcc@1* stands for the (top-1 validation) corrected accuracy on the correctable set of ImageNet examples with correct labels. To be corrected, at least 3 of 5 Mechanical Turk raters had to independently agree on a new label, proposed by us using the class with the $\arg\max$ probability for the example.

Platform	Model	Acc@1	cAcc@1	Acc@5	cAcc@5	Rank@1	cRank@1	Rank@5	cRank@5
PyTorch 1.0	resnet18	6.51	82.42	73.81	99.58	34	1	30	1
PyTorch 1.0	resnet50	13.52	73.74	79.97	98.46	20	2	11	2
PyTorch 1.0	vgg19_bn	13.03	73.39	79.97	97.97	23	3	10	9
PyTorch 1.0	vgg11_bn	11.13	72.97	76.26	97.55	30	4	22	15
PyTorch 1.0	resnet34	13.24	72.62	77.80	98.11	21	5	18	6
PyTorch 1.0	densenet169	14.15	72.55	79.62	98.32	16	6	12	3
PyTorch 1.0	densenet121	14.29	72.48	78.64	97.97	14	7	16	11
PyTorch 1.0	vgg19	13.03	72.34	79.34	98.04	22	8	13	8
PyTorch 1.0	resnet101	14.64	71.99	81.16	98.25	11	9	5	4
PyTorch 1.0	vgg16	12.39	71.43	77.52	97.20	28	10	19	19
PyTorch 1.0	densenet201	14.71	71.22	80.81	97.97	10	11	6	10
PyTorch 1.0	vgg16_bn	13.59	71.15	77.87	97.41	19	12	17	17
Keras 2.2.4	densenet169	13.94	70.87	78.85	98.18	17	13	15	5
PyTorch 1.0	densenet161	15.13	70.73	80.11	98.04	7	14	8	7
Keras 2.2.4	densenet121	13.94	70.59	76.40	97.48	18	15	20	16
PyTorch 1.0	resnet152	15.27	70.45	81.79	97.83	5	16	4	12
PyTorch 1.0	vgg11	12.96	70.38	75.49	97.27	25	17	27	18
PyTorch 1.0	vgg13_bn	12.68	69.89	75.84	96.99	27	18	25	20
PyTorch 1.0	vgg13	13.03	69.47	76.40	96.78	24	19	21	24
Keras 2.2.4	nasnetmobile	14.15	69.40	79.27	96.85	15	20	14	21
Keras 2.2.4	densenet201	15.20	69.19	80.11	97.76	6	21	9	13
Keras 2.2.4	mobilenetV2	14.57	68.63	75.84	96.57	12	22	24	26
Keras 2.2.4	inceptionresnetv2	17.23	68.42	83.40	96.85	3	23	2	22
Keras 2.2.4	xception	17.65	68.28	82.07	97.62	2	24	3	14
Keras 2.2.4	inceptionv3	16.11	68.28	80.25	96.78	4	25	7	23
Keras 2.2.4	vgg19	11.83	68.07	73.95	95.52	29	26	29	30
Keras 2.2.4	mobilenet	14.36	67.58	73.60	96.08	13	27	31	27
Keras 2.2.4	resnet50	14.85	66.81	76.12	95.73	9	28	23	28
Keras 2.2.4	nasnetlarge	19.61	66.32	84.24	96.57	1	29	1	25
Keras 2.2.4	vgg16	12.82	66.11	74.09	95.66	26	30	28	29
PyTorch 1.0	inception_v3	14.92	65.62	75.56	95.38	8	31	26	31
PyTorch 1.0	squeezenet1_0	9.66	63.66	60.50	91.88	32	32	34	33
PyTorch 1.0	squeezenet1_1	9.38	62.54	61.97	92.30	33	33	33	32
PyTorch 1.0	alexnet	11.06	58.96	62.61	89.29	31	34	32	34

Table S2: Individual CIFAR-10 accuracy scores for Sub-figure 3c with *agreement threshold = 3 of 5*. Acc@1 stands for the top-1 validation accuracy on the correctable set ($n = 18$) of original CIFAR-10 examples and labels. See Table S1 caption for more details. Discretization of accuracies occurs due to the limited number of corrected examples on the CIFAR-10 test set.

Platform	Model	Acc@1	cAcc@1	Acc@5	cAcc@5	Rank@1	cRank@1	Rank@5	cRank@5
PyTorch 1.0	googlenet	55.56	38.89	94.44	94.44	1	10	13	13
PyTorch 1.0	vgg19_bn	50.00	38.89	100.00	100.00	2	11	7	7
PyTorch 1.0	densenet169	44.44	50.00	100.00	100.00	5	4	2	2
PyTorch 1.0	vgg16_bn	44.44	44.44	100.00	100.00	3	8	5	5
PyTorch 1.0	inception_v3	44.44	33.33	100.00	100.00	6	12	8	8
PyTorch 1.0	resnet18	44.44	55.56	94.44	100.00	4	2	10	10
PyTorch 1.0	densenet121	38.89	50.00	100.00	100.00	8	5	3	3
PyTorch 1.0	densenet161	38.89	50.00	100.00	100.00	9	6	4	4
PyTorch 1.0	resnet50	38.89	44.44	100.00	100.00	7	9	6	6
PyTorch 1.0	mobilenet_v2	38.89	27.78	100.00	100.00	10	13	9	9
PyTorch 1.0	vgg11_bn	27.78	66.67	100.00	100.00	11	1	1	1
PyTorch 1.0	resnet34	27.78	55.56	94.44	100.00	13	3	11	11
PyTorch 1.0	vgg13_bn	27.78	50.00	94.44	100.00	12	7	12	12