Robust Weak Supervision with Variational Auto-Encoders

Francesco Tonolini [1] **Nikolaos Aletras** [1 2] **Yunlong Jiao** [1] **Gabriella Kazai** [1]

Abstract

Recent advances in programmatic weak supervision (WS) techniques allow to mitigate the enormous cost and effort of human data annotation for supervised machine learning by automating it using simple rule-based labelling functions (LFs). However, LFs need to be carefully designed, often requiring expert domain knowledge and extensive validation for existing WS methods to be effective. To tackle this, we propose the Weak Supervision Variational Auto-Encoder (WS-VAE), a novel framework that combines unsupervised representation learning and weak labelling to reduce the dependence of WS on expert and manual engineering of LFs. Our technique learns from inputs and weak labels jointly to capture the input signals distribution with a latent space. The unsupervised representation component of the WS-VAE regularises the inference of weak labels, while a specifically designed decoder allows the model to learn the relevance of LFs for each input. These unique features lead to considerably improved robustness to the quality of LFs, compared to existing methods. An extensive empirical evaluation on a standard WS benchmark shows that our WS-VAE is competitive to state-of-the-art methods and substantially more robust to LF engineering.

1. Introduction

One of the most severe bottlenecks in the successful development of supervised learning methods, including deep learning, is the cost of manual annotation of data. This task is often extremely expensive and time consuming, especially for applications where labelling requires specific domain expertise (Zhou et al., 2017; Zhang et al., 2022). To overcome this limitation, the paradigm of programmatic weak supervision (WS) has been proposed (Ratner et al., 2016a; 2017), where supervised learning models are trained without manual annotations, but using weak labels obtained from manually constructed (i.e. programmatic) rules instead. In a WS framework, domain experts do not need to label each training example individually, but just define a series of general rules, i.e., labelling functions (LFs). LFs are subsequently applied to the training data in bulk and combined to obtain estimates of the underlying true labels (Zhang et al., 2022; Ratner et al., 2017; Bach et al., 2019; Fu et al., 2020).

Despite their success, current programmatic WS techniques require high quality LFs and large amounts of data to perform comparably to supervised models trained on relatively smaller manually annotated datasets (Bach et al., 2019; Ratner et al., 2016b). In particular, most approaches make strong LF independence assumptions and require them to combine high coverage and accuracy over the training set (Ratner et al., 2017; Fu et al., 2020). This reliance on careful engineering of LFs introduces a new bottleneck to the application of deep learning at scale as designing and testing programmatic rules can become cumbersome and costly.

To mitigate these issues, we introduce the Weak Supervision Variational Auto-Encoder (WS-VAE); a new WS framework based on VAEs (Kingma & Welling, 2013; Rezende et al., 2014; Pu et al., 2016). Instead of inferring labels, the WS-VAE models the distribution of weak labels and inputs with an auto-encoder architecture. A neural network encodes input features into representation vectors where one dimension of these representation vectors is modelled as a continuous relaxation of the true hidden label. Differently from how semi-supervised VAEs handle labels (Kingma et al., 2014), we design a novel weak labels decoder, which learns to assign a different weight to each LF in each training example, based on representation consistency. This architecture results in a weak labels-guided representation of the data, where one dimension captures the task of interest.

The WS-VAE is more robust to LF design compared to existing weak supervision approaches, mainly for two reasons. Firstly, because it has to reconstruct input features, it is encouraged to maintain mutual information between features and inferred labels, which helps to ignore noise and infer accurately where weak labels are scarce. Secondly, the specifically designed weak labels decoder learns to give

[1]Amazon [2]Computer Science Department, University of Sheffield. Correspondence to: Francesco Tonolini <tonolini@amazon.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

different importance to the available weak labels in the retrieval, allowing to rely more on the informative ones and less on the inaccurate ones. In fact, our WS-VAE can even explain certain weak labels as noise and completely ignore them, if the corresponding inputs are recognised as inconsistent with the distribution of similarly labelled data. In our extensive experimental evaluation, we find WS-VAE to be a competitive WS approach with benchmark data sets and demonstrate its superior robustness to LFs engineering compared to several state-of-the-art WS methods in simulation experiments. In summary, our contributions are the following:

- We propose the use of modified VAE models to perform WS. Modelling the input features distribution jointly to inferring weak labels regularises WS and improves its robustness to LFs quality.
- We introduce a novel VAE architecture, the WS-VAE, which presents a specifically designed weak labels decoder that can adaptively assign LFs weights in different inputs. This feature allows our model to unsupervisedly estimate when certain LFs are inaccurate and further improves robustness to LFs engineering.
- We test several state-of-the-art WS models along with our model in both established benchmarks and hundreds of simulated experiments, artificially altering the quality of the LFs in the data sets. Our results show that WS-VAE is a competitive WS method (performing on par or outperforming state-of-the-art in four out of six tasks) while being substantially more robust to LFs engineering compared to other methods.

2. Background and Related Work

2.1. Weak Supervision

The general aim of weak supervision (WS) is to harvest multiple manually constructed rule-based labelling functions, which individually may have relatively low accuracy and coverage over the training data, and use them in combination to infer labels with which to train supervised models. Most WS frameworks include three main stages:

1. **Design and apply labelling functions (LFs):** Given an unlabelled training set X, domain experts first define a series of heuristic rules, each of them labelling the target data automatically, but with expected low accuracy and scarce coverage. For example, a LF may be designed to label an email as spam if a certain word is contained in the text and abstain otherwise. Each of these LFs is then applied to the whole training set, obtaining a list of weak labels $\Lambda_i = \{\lambda_{1,i}, \lambda_{2,i}, ..., \lambda_{K,i}\}$ for each training sample $x_i \in X$.
2. **Train a generative model (GM):** The obtained weak labels $\Lambda_i \in \mathbf{\Lambda}$ are combined to retrieve a probabilistic or soft label $p(y_i|\Lambda_i)$ that approximates the true hidden label $y_i \in Y$. Models to perform this step are often referred to as generative models (GMs) in the WS literature.
3. **Train an end model:** Train a final learning model to infer labels y_i from inputs x_i, using the estimates obtained from the GM as targets. Some end models are trained with approximate hard labels $\tilde{y}_i = \arg\max p(y_i|\Lambda_i)$ (Ratner et al., 2017), while others are designed to use the soft labels $p(y_i|\Lambda_i)$ directly (Devlin et al., 2018).

Most of recent work in WS focuses on stage 2 for designing GMs that efficiently combine the weak labels in a principled way (Ratner et al., 2017; Bach et al., 2019; Varma et al., 2019a; Fu et al., 2020). Different variations of GMs have also been developed for various types of data and labels, including multi-task learning (Ratner et al., 2019), sequential data (Zhan et al., 2018; Varma et al., 2019b) and generalised tasks, such as ranking and regression (Shin et al., 2021). Other work aims to exploit different types of LFs, such as continuous values (Chatterjee et al., 2019), multiple class label candidates per LF (Yu et al., 2021) and unseen related classes (Zhang et al., 2021a).

Some work also focuses on improving the other two stages (i.e. 1 and 3). The Nemo framework was recently proposed to formalise the LFs design in stage 1 as an interactive process (Hsieh et al., 2022). Stage 3 often adopts domain specific architectures to train the final model. For instance, with text data it is common to use recurrent (Gers et al., 2002) or attention based (Vaswani et al., 2017) methods as the end model (Zhang et al., 2021b). Yu et al. (2020) recently proposed an end model specifically designed for WS, in which data not covered by the LFs is introduced into training. The intersection of weak supervision and semi-supervised learning has recently been explored including methods that exploit a small amount of true labels in combination with large amounts of weak labels for learning (Karamanolakis et al., 2021; Awasthi et al., 2020; Maheshwari et al., 2020). Similarly, other methods focus on exploiting a small amount of noisy labels in combination with WS (Li et al., 2020; Goel et al., 2022).

Other recent approaches also combine stages 2 and 3 (Ren et al., 2020; Rühling Cachay et al., 2021). These approaches train jointly a generative model and an end model by adding an agreement cost between the two. In such a way, the generative model is influenced by the end model input features during training and can derive more information from them. Our proposed approach also retains this property and can be used as a joint model. However, we incorporate unsuper-

vised learning into the process to mitigate the reliance on LFs of high quality.

2.2. Variational Auto-Encoders

VAEs are unsupervised models which aim to model the distribution of observed data $p(x)$ with a latent variable structure $p(x) = \int p(z)p(x|z)dz$, where $p(x|z)$ is typically a neural network-based model to be learned, called the decoder. Learning the weights of $p(x|z)$ is intractable in most cases, hence VAEs make use of a tractable lower bound, which they maximise in place of the exact likelihood:

$$\log p(x) \geq \mathbb{E}_{q(z|x)} \left[\log p(x|z)\right] - D_{KL}(q(z|x)||p(z)) \quad (1)$$

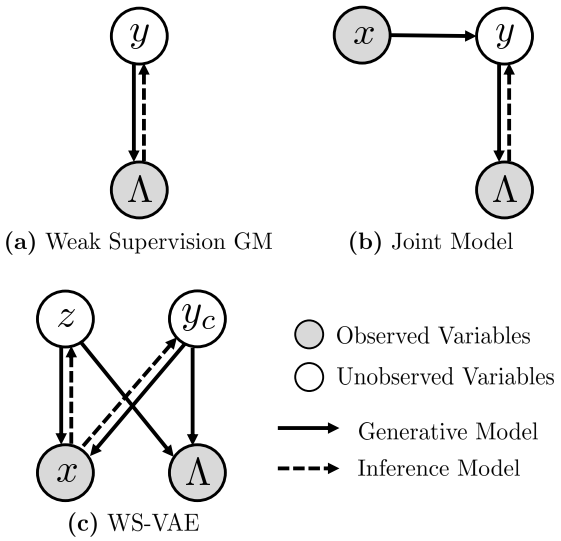
The recognition model $q(z|x)$, or encoder, is also a neural network and its weights are learned jointly with the decoder weights. Training consists of maximising the above evidence lower bound (ELBO) with respect to the encoder and decoder weights (Kingma & Welling, 2013; Pu et al., 2016).

The latent space z of a trained VAE can often represent data in a meaningful way and capture its sources of variation with simple distributions, e.g. Gaussians (Sønderby et al., 2016; Alemi et al., 2018). Significant efforts have been put into improving this capability of VAEs, generally known as disentanglement (Burgess et al., 2018; Chen et al., 2018; Gao et al., 2019; Kim & Mnih, 2018; Tonolini et al., 2020; Ding et al., 2020). In this paper, we aim to exploit such a latent space to capture the complex dependencies of labelling functions with each other and the input signals.

2.3. Weak Supervision and Generative Models

Weak labels have been used in combination with VAEs in recent work, but in the context of improving disentanglement (Vowels et al., 2020a; Margonis et al., 2020), or common factor discovery for fairness (Vowels et al., 2020b), while our method aims at modelling and retrieving the true labels associated with a specific task for the first time.

Perhaps the closest approach to our own is that of the WS-GAN, which introduces the use of a generative adversarial network (GAN) to perform weak supervision (Boecking et al., 2022). This work introduces an adaptive weak labels weighting function which learns to infer the importance of weak labels from input features. This results in an adaptive weak labels modelling similar to that of the WS-VAE decoder. Differently from this approach, however, the WS-VAE decoder infers the weak labels importance from the latent space, employing a much simpler and smaller neural network. Moreover, the WS-VAE derives this adaptive feature with a more principled approach, by modelling weak labels as noisy observations of the true label and learning the noise parameters.



(a) Weak Supervision GM (b) Joint Model (c) WS-VAE

Figure 1. Graphical models for standard WS generative models compared to the proposed one: **(a)** standard weak supervision generative model, e.g. Snorkel, FlyingSquid. **(b)** joint model, such as Weasel, where the end model, mapping inputs x to labels y, is trained alongside the weak labels generative model. **(c)** The WS-VAE model, where both inputs x and weak labels Λ are modelled with artificial latents z and hidden continuous true labels y_c.

3. The Weak Supervision Variational Auto-Encoder (WS-VAE)

We propose WS-VAE; a novel weak supervision generative model based on VAEs, which models observed inputs and weak labels jointly with a latent variable model (LVM). The WS-VAE has an auto-encoding architecture, where input signals are mapped to a latent representation space through an encoder and subsequently reconstructed through a decoder. To induce one of the latent dimensions to capture the desired task, and hence perform WS, we design a novel weak labels decoder architecture which infers the weak labels with a specific conditional distribution.

The model is trained by maximising the ELBO which balances the reconstruction of the inputs, regularisation and inference of the weak labels. Once the model is trained, the encoder can be used to infer labels from new inputs. Our framework operates in a fully unsupervised weak labelling regime, i.e., with no access to training or validation ground-truth labels. The WS-VAE is a joint model, where stages 2 and 3 (described in section 2.1) are combined, as the trained encoder can directly serve as an end model.

3.1. Problem Statement

Given an unlabelled training set $X = \{x_1, x_2, ..., x_N\}$ with no access to ground-truth labels, we aim to train a classification model $f(x)$ which infers labels $Y = \{y_1, y_2, ..., y_N\}$.

We assume to have access to K programmatic rules, i.e. the LFs. Each LF assigns a weak label $\lambda_{i,k}$ to inputs x_i, approximating the ground-truth label y_i. In our setting, we take positive and negative classes to be $+1$ and -1 respectively and abstain to be 0, meaning that $\lambda \in \{+1, 0, -1\}, y \in \{+1, -1\}$. In this paper, we focus on the binary classification case, leaving multi-class and inference extensions to future work. After the LFs have been applied to the training set X, each training input x_i has a series of weak labels assigned to it $\Lambda_i = \{\lambda_{i,1}, \lambda_{i,2}, ..., \lambda_{i,K}\}$. Given the training set of inputs $X = \{x_1, x_2, ..., x_N\}$ and the corresponding sets of weak labels $\mathbf{\Lambda} = \{\Lambda_1, \Lambda_2, ..., \Lambda_N\}$, we aim to train a classifier $f(x)$ to predict hidden ground-truth labels y from new inputs x. In the technical sections that follow, we drop the subscript i for simplicity, i.e. $x_i \to x$ and $\Lambda_i \to \Lambda$.

3.2. Generative Model

We take a generative modelling approach to perform WS and retrieve estimates of the hidden labels y. We assume that there is a model $p(x, \Lambda)$ which captures the joint distribution of all the data we can observe, i.e., inputs x and weak labels Λ. We also assume that a latent variable y_c of this model captures the source of variation in the observed data associated with the task. To infer this model and hence retrieve the latent y_c, we formalise the joint distribution of the inputs x and weak labels Λ as a LVM of the following form:

$$\log p(x, \Lambda) = \int\int p(z)p(y_c)p(x|z, y_c)p(\Lambda|z, y_c)dzdy_c. \tag{2}$$

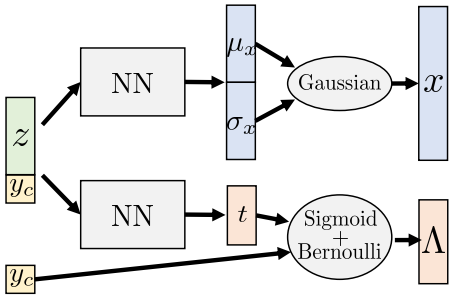
y_c is modeled to be a continuous version of the hidden true label y, such that

$$y = \begin{cases} 0, & \text{if } y_c \leq 0 \\ 1, & y_c > 0. \end{cases}$$

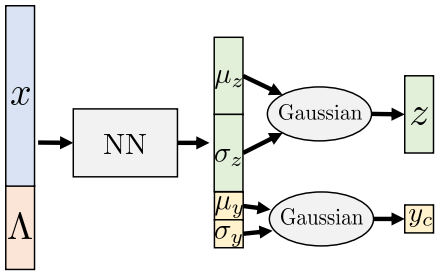
This modelling choice allows us to exploit the VAE reparametrisation trick (Kingma & Welling, 2013; Razavi et al., 2019; Kingma et al., 2014). z is a set of latent variables, which we introduce to capture the sources of variation in inputs x and weak labels Λ that are not described by the continuous label y_c, i.e. all information in the data which is not associated with the task. Each component of the generative model in equation 2 is described below:

Prior Distributions: The latent priors $p(z)$ and $p(y_c)$ are both chosen to be unit Gaussians. These distributions model the distribution we assume z and y_c to have, prior to seeing any observed data. In practice, they introduce regularisation in the continuous label y_c and latent variables z.

Input Decoder: As in standard VAEs, the input decoder $p(x|z, y_c)$ is defined as an isotropic Gaussian distribution over x, the moments of which are inferred by a neural network taking the concatenated z and y_c as input.



(a) WS-VAE Decoder



(b) WS-VAE Encoder

Figure 2. Architectures of the WS-VAE decoder and encoder: **(a)** The decoder architecture: concatenated latent variable z and continuous label y_c are passed through two distinct neural networks (NNs); one outputting Gaussian moments in the input space x and one outputting the temperatures t. The distributions of binary weak labels Λ are Bernoulli with probabilities sigmoid(y_c/t). **(b)** The encoder architecture: Input signals x are passed through a NN outputting moments of Gaussian distributions for both artificial latent variables z and continuous hidden label y_c.

Weak Label Decoder: This component is designed to be of a novel form, tailored to the WS setting. In order to induce the continuous label y_c to capture information from the classification task and be a good predictor for the hidden true labels y, the weak labels likelihood is as follows:

$$\log p(\Lambda|z, y_c) = \sum_{k}^{K} \log(\text{sigmoid}(\frac{\lambda_k y_c}{t_k(z, y_c)})). \tag{3}$$

Here $t_k(z, y_c)$ are the temperatures of the logistic functions. These are the outputs of a neural network taking as input the latent variable z concatenated to the continuous label y_c and are constrained to be positive.

The weak labels decoder (equation 3) induces the continuous label y_c to directly classify each of the weak labels $\lambda_k \in \Lambda$. However, because the temperatures t_k are inferred through a neural network from latent encodings (z, y_c), the model can assign a different weight to each

LF adaptively for each training example. This architecture results in a similar adaptive LFs weighting to that of Karamanolakis et al. (2021) in semi-supervised settings, with the important difference that our WS-VAE does not rely on ground-truth labels to infer the weights. This generative model structure also allows to model correlations amongst LFs, as the common weak supervision conditional independence assumption $p(\Lambda|y) = \prod_k^K p(\lambda_k|y)$ is relaxed to $p(\Lambda|y,z) = \prod_k^K p(\lambda_k|y,z)$.

3.3. Inference Model and Training

To train our WS-VAE, we aim to maximise the log likelihood of the observed data $\log p(x,\Lambda)$ with respect to the model parameters, i.e. the weights of the decoders. Because directly maximising the log likelihood is intractable (Kingma & Welling, 2013), we make use of a neural encoder to define a tractable lower bound (i.e., the ELBO) in its place:

$$\begin{aligned}\log p(x,\Lambda) \geq & \mathbb{E}_{q(z|x)q(y_c|x)}\left[\log p(x|z,y_c) + \log p(\Lambda|z,y_c)\right] \\ & - D_{KL}(q(z|x)||p(z)) - D_{KL}(q(y_c|x)||p(y_c))\end{aligned} \tag{4}$$

The encoders $q(z|x)$ and $q(y_c|x)$ are both Gaussian, with moments inferred by a neural network, taking as input the feature representation x. The resulting auto-encoder structure jointly models the distribution of input features x and infers the weak labels Λ, hence propagating label information between similar inputs x. This allows the WS-VAE to infer soft labels even for data not covered by any LF. Figure 2 illustrates the overall decoder and encoder architectures. To train the WS-VAE, we maximise the ELBO of equation 4 with respect to the encoder and decoders weights (details in supplementary A.1).

We hypothesise that this objective will result in improved robustness to the quality of LFs compared to existing methods because of two main reasons:

1. **Mutual Information Regularisation:** The WS-VAE ELBO (equation 4) combines a weak labels reconstruction likelihood, $\log p(\Lambda|z,y_c)$ with the standard VAE ELBO. The former induces inference of the weak labels, similarly to existing WS methods, while the latter encourages inferred continuous true labels y_c from similar inputs x to be close to each other. This encourages mutual information between y_c and x and helps to correct erroneous weak labels.

2. **Adaptive Weak Labels Cost:** The model learns to infer the temperatures $t_k(z,y_c)$ for the weak labels inference likelihoods (equation 3), which control the importance given in the overall cost to each weak label for each example. This allows the model to learn during training for which inputs x a particular LF is likely to make mistakes and adapt its importance accordingly in the optimisation.

3.4. Inferring Soft Labels

Once the WS-VAE has been trained with the available training input features X and weak labels $\mathbf{\Lambda}$, its encoder $q(y_c|x)$ can be used to generate probabilistic labels for the binary true variable y. The probability for each class is computed as follows:

$$\begin{aligned} q(y=0|x) &= \int_{-\infty}^{0} q(y_c|x)dy_c = \frac{1}{2} + \frac{1}{2}erf(\frac{-\mu_{y_c}}{\sigma_{y_c}}) \\ q(y=1|x) &= \int_{0}^{\infty} q(y_c|x)dy_c = \frac{1}{2} - \frac{1}{2}erf(\frac{-\mu_{y_c}}{\sigma_{y_c}}) \end{aligned} \tag{5}$$

Here $erf()$ indicates the standard error function and μ_{y_c} and σ_{y_c} are the mean and standard deviation of the Gaussian $q(y_c|x)$. In this way, the trained WS-VAE encoder $q(y_c|x)$ can be used as a probabilistic classifier $q(y|x)$ for a given classification task directly. Architecture and training details of the WS-VAE are presented in Supp. A.2 and B.2.

4. Experiments and Results

4.1. Data

We test our WS-VAE against several state-of-the-art WS methods on Wrench (Zhang et al., 2021b), a standard publicly available WS benchmark which consists of various tasks. We use 6 binary classification data sets including: 3 text data sets, namely, YouTube (Alberto et al., 2015), SMS (Gómez Hidalgo et al., 2006) and IMDB (Maas et al., 2011); 2 image data sets, namely Tennis Rally and Commercial (Fu et al., 2020; Zhang et al., 2021b); and 1 tabular data set, Census (Kohavi et al., 1996). Data sets details can be found in supp. B.1. For text data, pre-trained BERT encodings (Devlin et al., 2018) are taken as input features x, for image data sets input features are extracted by a ResNet-101 model pre-trained on ImageNet (He et al., 2016) and for the tabular data set raw features are used as inputs to the WS-VAE and the competing methods. WS-VAE implementation details can be found in supp. B.2.

4.2. Baselines

We compare performance of the WS-VAE to 3 WS generative models, namely, majority voting (MV) (Zhang et al., 2021b), Snorkel (SN) (Ratner et al., 2017) and Flying Squid (FS) (Fu et al., 2020), in combination with a multi-layer perceptron (MLP) end model for all data sets and also fine tuning BERT (Devlin et al., 2018) for the text data sets. Since the WS-VAE is a joint model, it can perform inference directly after training with the weak labels. Therefore,

	YouTube	SMS	IMDB	Tennis	Commercial	Census
MV+MLP	0.83 ± 0.02	0.40 ± 0.06	0.74 ± 0.05	0.76 ± 0.07	$\mathbf{0.90 \pm 0.01}$	0.30 ± 0.01
MV+BERT	0.88 ± 0.04	0.40 ± 0.11	0.74 ± 0.01	-	-	-
SN+MLP	0.86 ± 0.01	0.38 ± 0.05	0.76 ± 0.01	0.82 ± 0.00	0.81 ± 0.05	0.42 ± 0.01
SN+BERT	0.87 ± 0.03	0.55 ± 0.11	0.74 ± 0.01	-	-	-
FS+MLP	0.81 ± 0.01	0.37 ± 0.02	$\mathbf{0.77 \pm 0.01}$	0.79 ± 0.01	0.89 ± 0.01	0.16 ± 0.01
FS+BERT	0.83 ± 0.01	0.46 ± 0.12	0.74 ± 0.00	-	-	-
Denoise	0.88 ± 0.01	$\mathbf{0.86 \pm 0.02}$	0.72 ± 0.02	0.80 ± 0.02	0.80 ± 0.03	0.14 ± 0.01
Weasel	$\mathbf{0.91 \pm 0.02}$	0.84 ± 0.06	0.71 ± 0.05	0.73 ± 0.05	0.67 ± 0.07	0.41 ± 0.02
WS-VAE (Ours)	0.90 ± 0.01	$\mathbf{0.86 \pm 0.02}$	$\mathbf{0.77 \pm 0.00}$	$\mathbf{0.83 \pm 0.01}$	0.89 ± 0.00	$\mathbf{0.48 \pm 0.06}$

Table 1. F1-scores of several WS methods compared to the proposed WS-VAE on benchmark data sets. Each experiment is repeated 5 times with a different random seed to obtain uncertainties.

	YouTube	SMS	IMDB	Tennis	Commercial	Census
MV+MLP	0.94 ± 0.03	0.66 ± 0.05	0.84 ± 0.03	$\mathbf{0.89 \pm 0.03}$	$\mathbf{0.90 \pm 0.01}$	0.82 ± 0.01
MV+BERT	0.97 ± 0.01	0.80 ± 0.07	0.62 ± 0.08	-	-	-
SN+MLP	0.96 ± 0.01	0.80 ± 0.04	0.84 ± 0.02	0.86 ± 0.00	0.81 ± 0.05	0.84 ± 0.02
SN+BERT	0.97 ± 0.01	0.84 ± 0.04	0.82 ± 0.02	-	-	-
FS+MLP	0.90 ± 0.02	0.74 ± 0.02	0.84 ± 0.02	0.86 ± 0.01	0.89 ± 0.01	0.77 ± 0.02
FS+BERT	0.96 ± 0.01	0.88 ± 0.03	0.81 ± 0.01	-	-	-
Denoise	0.97 ± 0.01	0.84 ± 0.02	0.72 ± 0.05	$\mathbf{0.89 \pm 0.01}$	0.80 ± 0.03	0.54 ± 0.04
Weasel	$\mathbf{0.99 \pm 0.01}$	0.90 ± 0.03	0.79 ± 0.04	0.87 ± 0.03	0.67 ± 0.07	0.85 ± 0.02
WS-VAE (Ours)	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.95 \pm 0.01}$	$\mathbf{0.85 \pm 0.01}$	$\mathbf{0.89 \pm 0.02}$	0.89 ± 0.00	$\mathbf{0.89 \pm 0.03}$

Table 2. AUC scores of several WS methods compared to the proposed WS-VAE on benchmark data sets. Each experiment is repeated 5 times with a different random seed to obtain uncertainties.

we also test against two recently proposed joint models; Denoise (Ren et al., 2020) and Weasel (Rühling Cachay et al., 2021). Baselines details can be found in supp. B.3. In this paper, we test models under fully unsupervised WS conditions, where we do not have access to any ground truth labels, even for validation. Both the WS-VAE and all baseline models are trained for a fixed number of iterations, without validating with ground truth labels and hence rely solely on weak labels. Details can be found in supp. B.2 and B.3.

4.3. Predictive Performance

We first test the WS-VAE on the original Wrench data, to evaluate its performance against existing WS methods. Table 1 reports the macro F1-score of end and joint models in each data set. We also report the corresponding AUC scores in table 2. Overall, we observe that our WS-VAE model obtains consistently high performance across tasks. Specifically, it matches or outperforms the F1-score of the best baseline state-of-the-art model in four out of six data sets (i.e., SMS, IMDB, Tennis and Census in table 1). In the remaining two data sets (YouTube and Commercial), it is the second best model, with an F1-score that is lower than the best baseline by only 0.01. Conversely, all baselines performed substantially lower than the average performance of other models in at least one data set, meaning that there is not a clear overall best choice amongst existing WS methods. For example, Weasel performs quite competitvely overall, being the best with the YouTube data set and second best with the SMS data set, but under-performs a simple majority voting strategy with the Commercial data set by 0.23 in F1-score. This is not the case for the WS-VAE, which either outperforms all baselines or under-performs only marginally the best model. This result validates that WS-VAE is a competitive WS method.

In particular, the WS-VAE is substantially advantageous in the Census data set, where it provides an improvement of 0.06 in F1-score compared to the best baseline (SN+MLP). We note that this large improvement is observed in the data set where the overall WS performance is the lowest and therefore the weak labels might be inaccurate and difficult to learn from. This suggests that the WS-VAE is more robust to these conditions and less reliant on the engineering of the LFs, thanks to the features described in 3.3. In the rest of the experimental section, we further investigate the robustness of the WS-VAE in depth.

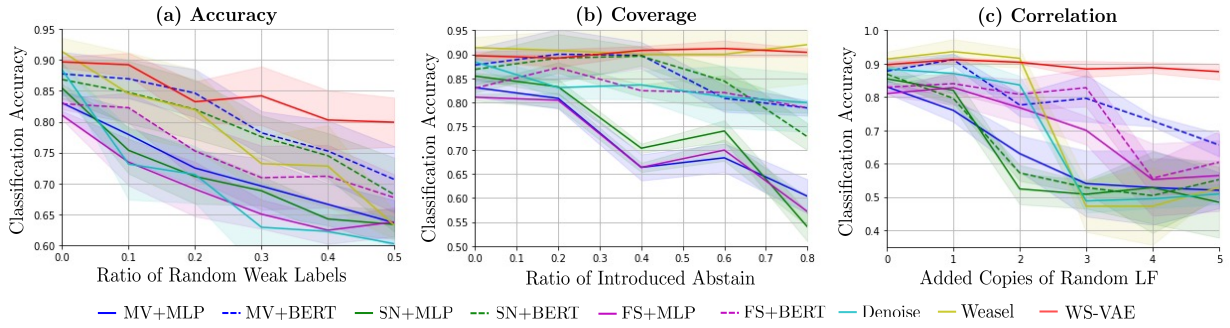


Figure 3. Classification accuracy in various LFs degradation conditions with the YouTube data set: **(a)** Classification accuracy vs. ratio of artificially added noise on the non-abstaining weak labels. **(b)** Classification accuracy vs. ratio of artificially abstained weak labels. **(c)** Classification accuracy vs. number of appended copies of randomly labelling LF.

4.4. Testing the Robustness of WS Methods

In order to evaluate the sensitivity of the WS-VAE to the engineering of LFs, we conduct a number of simulation experiments, where weak labels from the benchmark data sets are artificially modified to obtain different quality conditions with respect to the LFs. Specifically, we explore three aspects: accuracy, coverage and correlation. In real settings, LFs design efforts focus primarily on these aspects, as we want the resulting weak labels to be as correct as possible (accuracy), apply to as much data as possible (coverage) and measure different things in the data (correlation). We vary these three properties as follows:

- **Accuracy:** To degrade the accuracy of the LFs, we substitute a varying portion of the covered labels with a random binary value (positive or negative class).
- **Coverage:** To simulate low coverage, a varying portion of covered labels is set to abstain.
- **Correlation:** To simulate the effect of using uninformative LFs that are mostly correlated to each other, we append copies of a random LF to the existing set.

4.4.1. Testing Individual Robustness Properties

First, we study the effect of these 3 artificial LFs degradation individually in the YouTube data set for demonstration purposes. The YouTube weak labels are degraded as described above, varying gradually the controlling variable in each case and testing the WS-VAE and all baselines in each condition. Results are shown inf Figure 3. We make the following observations:

- **Accuracy (Figure 3(a)):** Overall, all methods suffer by gradually increasing the noise in the weak labels as expected. However, the WS-VAE remains more accurate than competing methods, maintaining on average 80% accuracy with 50% of available labels substituted with noise, while competing methods are between 65% and 71% in the same condition - a 9% difference compared to the second best (MV+BERT). This is because the WS-VAE models the correlations between inputs x and weak labels Λ and learns to ignore weak labels which appear inconsistent.
- **Coverage (Figure 3(b)):** Most baselines degrade significantly as coverage is reduced, as less labels are available for training. Weasel and our WS-VAE are instead essentially unaffected, maintaining an accuracy of around 90% across all tested conditions. This is because both our model and Weasel are able to incorporate data not covered by any LF during training. Hence, they greatly mitigate over-fitting issues.
- **Correlation (Figure 3(c)):** We also observe that baseline WS methods severely degrade when correlated noisy labels are added to the set of weak labels, with accuracy varying between 50% and 70% using five copies of a random LF. This is because the added LFs introduce multiple random weak labels in perfect agreement with each other. This leads to these random labels constituting a stronger signal than any existing informative weak labels, causing the models to overfit to them. Contrarily, the WS-VAE can capture such correlations between LFs and ignore them in the modelling, if they display no mutual information with the associated inputs. As a result, the accuracy of WS-VAE remains close to 90% as more copies of a random LF are added.

Overall, the results of Figure 3 demonstrate that as the accuracy, coverage or independence of the set of weak labels is degraded, our WS-VAE model retains substantially higher performance than existing state-of-the-art WS methods including generative and joint models. We further test the robustness of the WS-VAE model to the number of defined LFs in supplementary C.1.

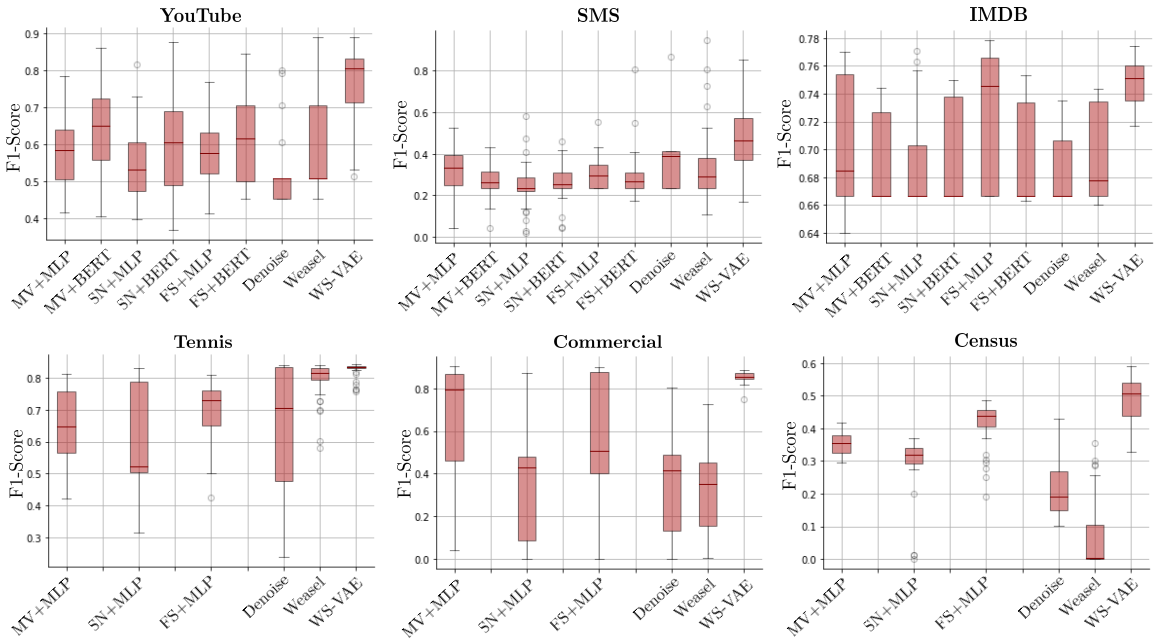


Figure 4. Distributions of macro F1-scores with randomly generated LFs properties. Distributions are built by performing 50 experiments with each data set, each with different, randomly generated LFs accuracy, coverage and correlation.

4.4.2. Testing Robustness Properties Combined

To extensively test the robustness of WS-VAE compared to the baselines, we combine all three robustness properties (accuracy, coverage and correlation), with randomly chosen intensity, on all six benchmark data sets. Given a benchmark data set with its originally computed LFs, we draw uniformly at random a ratio of weak labels to be set randomly (accuracy), a ratio of weak labels to be set to abstain (coverage) and a number of random LF copies to append to the set (correlation). We then test WS performance using all the baselines and our WS-VAE by computing macro F1-score in the test set of each task. This procedure is repeated 50 times for each benchmark data set, obtaining distributions of F1-scores with randomly altered weak labels properties. These are shown in the box plots of Figure 4. Experimental details are reported in supp. B.4.

Overall, we see that in all data sets, the distribution of F1-scores for the WS-VAE spans higher values and in many cases also displays lower variance. Differences are numerically quantified and tested for statistical significance in supp. C.2. These experiments comprise 300 unique conditions (6 datasets $\times$ 50 random LFs degradations). In 66.3% of these, the WS-VAE was the best performing model and displayed an average F1-score improvement to the best baseline in each experiment of 0.057. These results show how the WS-VAE gives better and more consistent performance with respect to different variations in weak labels quality, proving the superior robustness of WS-VAE to the engineering

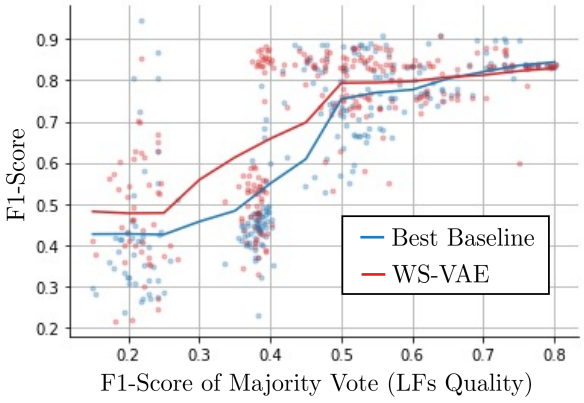


Figure 5. Model performance vs. LFs quality for all data sets and all experimental conditions. On the x-axis, we plot the macro F1-score of the raw majority vote, which we take as a proxy of the quality of the LFs used. On the y-axis, we plot the macro F1-score of the labels obtained using the WS model to be tested, which we take as a measure of model performance. The blue dots correspond to the best performing model amongst the baselines, chosen a-posteriori in each experiment. The red dots correspond to the WS-VAE. The continuous lines are obtained by performing a moving average in each case.

of LFs, compared to existing WS methods.

We further prove this superior robustness by studying the difference in performance as a function of LFs quality. There are several aspects that affect LFs quality, of which in this paper we analysed what we believe to be the three most important ones, and therefore it is difficult to define a single measure. However, we can use the F1-score of the raw majority vote as an approximate measure of quality. If the LFs are well engineered, i.e., they already approximate well the true labels and cover enough data, their majority vote should be in good agreement with the ground truth. Using this notion, in Figure 5 we plot WS model macro F1-score vs. raw majority vote macro F1-score for all 300 unique experimental conditions tested. We plot both the best baseline in each tested condition (blue) and the proposed WS-VAE (red).

The difference in data sets and the combination of the different types of LFs degradations that are applied randomly in each experimental setting result in significant variance in the performance of tested models. However, in Figure 5, we can identify as a clear general trend, shown by the moving averages solid lines, that the advantage of using the WS-VAE is substantial in cases where the starting weak labels are of poor quality (raw majority vote F1-score < 0.5). The difference in models' F1-score reduces as weak labels improve and eventually performance is comparable with LFs of high quality (raw majority vote F1-score ≥ 0.65). This trend is evidence again of the superior robustness of the WS-VAE. It performs comparably to the best baseline when LFs are of high quality, but loses accuracy much less rapidly as the quality of LFs degrades.

5. Limitations

In the form presented in this paper, the WS-VAE can perform programmatic weak supervision in the binary classification setting only. Our method was also implemented with fully connected networks, meaning that for text and image modalities it is necessary to compute dense vectors with pre-trained models first, such as BERT in the case of text data. This may hinder performance for uncommon data distributions, such as satellite images or text in rare languages. Because of the unique auto-encoder architecture, it is not trivial to integrate domain specific models directly in the WS-VAE, e.g., integrating pre-trained BERT into the VAE architecture.

6. Conclusion ad Future Work

We proposed WS-VAE, a novel WS framework that combines unsupervised representation learning and weak labelling. Thanks to the mutual information regularisation provided by the VAE component of the model and the adaptive architecture of its weak labels decoder, the WS-VAE is substantially more robust to the engineering of LFs. In our experiments, we thoroughly evaluate this robustness, comparing to several existing WS methods in hundreds of unique experimental conditions, generated using six data sets and tens of random modifications to their weak labels. From these experiments, we found the WS-VAE to be significantly more robust to the quality of training LFs in every data set. Furthermore, we demonstrated the effectiveness of WS-VAE by consistently matching or outperforming state-of-the-art in the majority of data sets.

In future work, we aim to extend the WS-VAE model from binary tasks to multi-class settings. This will require different designs of the continuous labels and different decoder components for the weak labels, e.g., categorical instead of Bernoulli. An additional future direction is also the extension of the WS-VAE to semi-supervised WS, where we can integrate a limited budget of ground-truth labels during training. While this is outside the scope of this paper, the LVM structure lends itself quite naturally to incorporate partial supervision (Kingma et al., 2014).

References

Alberto, T. C., Lochter, J. V., and Almeida, T. A. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pp. 138–143. IEEE, 2015.

Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurous, R. A., and Murphy, K. Fixing a broken elbo. In *International Conference on Machine Learning*, pp. 159–168. PMLR, 2018.

Awasthi, A., Ghosh, S., Goyal, R., and Sarawagi, S. Learning from rules generalizing labeled exemplars. *arXiv preprint arXiv:2004.06025*, 2020.

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

Boecking, B., Neiswanger, W., Roberts, N., Ermon, S., Sala, F., and Dubrawski, A. Generative modeling helps weak supervision (and vice versa). *arXiv preprint arXiv:2203.12023*, 2022.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Chatterjee, O., Ramakrishnan, G., and Sarawagi, S. Data programming using continuous and quality-guided labeling functions. *arXiv preprint arXiv:1911.09860*, 2019.

Chen, R. T., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ding, Z., Xu, Y., Xu, W., Parmar, G., Yang, Y., Welling, M., and Tu, Z. Guided variational autoencoder for disentanglement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7920–7929, 2020.

Fu, D., Chen, M., Sala, F., Hooper, S., Fatahalian, K., and Ré, C. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*, pp. 3280–3291. PMLR, 2020.

Gao, S., Brekelmans, R., Steeg, G., and Galstyan, A. Auto-encoding total correlation explanation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2019.

Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

Goel, A., Jiao, Y., and Massiah, J. Pars: Pseudo-label aware robust sample selection for learning with noisy labels. *arXiv preprint arXiv:2201.10836*, 2022.

Gómez Hidalgo, J. M., Bringas, G. C., Sánz, E. P., and García, F. C. Content based sms spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering*, pp. 107–114, 2006.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hsieh, C.-Y., Zhang, J., and Ratner, A. Nemo: Guiding and contextualizing weak supervision for interactive data programming. *arXiv preprint arXiv:2203.01382*, 2022.

Karamanolakis, G., Mukherjee, S., Zheng, G., and Hassan, A. Self-training with weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 845–863, 2021.

Kim, H. and Mnih, A. Disentangling by factorising. In *Proceedings of the International Conference on Machine Learning*, pp. 2649–2658, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*, 2013.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

Kohavi, R. et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pp. 202–207, 1996.

Li, J., Socher, R., and Hoi, S. C. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. 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.

Maheshwari, A., Chatterjee, O., Killamsetty, K., Ramakrishnan, G., and Iyer, R. Semi-supervised data programming with subset selection. *arXiv preprint arXiv:2008.09887*, 2020.

Margonis, V., Davvetas, A., and Klampanos, I. A. Wela-vae: Learning alternative disentangled representations using weak labels. *arXiv preprint arXiv:2008.09879*, 2020.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. Variational autoencoder for deep learning of images, labels and captions. In *Proceedings of the Advances in Neural Information Processing Systems*, 2016.

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

Ratner, A., Hancock, B., Dunnmon, J., Sala, F., Pandey, S., and Ré, C. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4763–4771, 2019.

Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29, 2016a.

Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29, 2016b.

Razavi, A., Van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Ren, W., Li, Y., Su, H., Kartchner, D., Mitchell, C., and Zhang, C. Denoising multi-source weak supervision for neural text classification. *arXiv preprint arXiv:2010.04582*, 2020.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, 2014.

Rühling Cachay, S., Boecking, B., and Dubrawski, A. End-to-end weak supervision. *Advances in Neural Information Processing Systems*, 34:1845–1857, 2021.

Shin, C., Li, W., Vishwakarma, H., Roberts, N., and Sala, F. Universalizing weak supervision. *arXiv preprint arXiv:2112.03865*, 2021.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*, 3(2), 2016.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.

Tonolini, F., Jensen, B. S., and Murray-Smith, R. Variational sparse coding. In *Uncertainty in Artificial Intelligence*, pp. 690–700. PMLR, 2020.

Varma, P., Sala, F., He, A., Ratner, A., and Ré, C. Learning dependency structures for weak supervision models. In *International Conference on Machine Learning*, pp. 6418–6427. PMLR, 2019a.

Varma, P., Sala, F., Sagawa, S., Fries, J., Fu, D., Khattar, S., Ramamoorthy, A., Xiao, K., Fatahalian, K., Priest, J., et al. Multi-resolution weak supervision for sequential data. *Advances in Neural Information Processing Systems*, 32, 2019b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vowels, M. J., Camgoz, N. C., and Bowden, R. Gated variational autoencoders: Incorporating weak supervision to encourage disentanglement. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pp. 125–132. IEEE, 2020a.

Vowels, M. J., Camgoz, N. C., and Bowden, R. Nested-vae: Isolating common factors via weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9202–9212, 2020b.

Yu, P., Ding, T., and Bach, S. H. Learning from multiple noisy partial labelers. *arXiv preprint arXiv:2106.04530*, 2021.

Yu, Y., Zuo, S., Jiang, H., Ren, W., Zhao, T., and Zhang, C. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. *arXiv preprint arXiv:2010.07835*, 2020.

Zhan, E., Zheng, S., Yue, Y., Sha, L., and Lucey, P. Generating multi-agent trajectories using programmatic weak supervision. *arXiv preprint arXiv:1803.07612*, 2018.

Zhang, J., Wang, B., Song, X., Wang, Y., Yang, Y., Bai, J., and Ratner, A. Creating training sets via weak indirect supervision. *arXiv preprint arXiv:2110.03484*, 2021a.

Zhang, J., Yu, Y., Li, Y., Wang, Y., Yang, Y., Yang, M., and Ratner, A. Wrench: A comprehensive benchmark for weak supervision. *arXiv preprint arXiv:2109.11377*, 2021b.

Zhang, J., Hsieh, C.-Y., Yu, Y., Zhang, C., and Ratner, A. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*, 2022.

Zhou, L., Pan, S., Wang, J., and Vasilakos, A. V. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.

A. WS-VAE Cost and Architecture

A.1. Optimisation of the ELBO

As is common in many VAE architectures (Burgess et al., 2018; Tomczak & Welling, 2018), we modify our ELBO to allow re-balancing of each term with hyper-parameters. In particular, we use a parameter γ to control the weight of the weak labelling reconstruction component. The WS-VAE ELBO is then maximised with respect to the different neural networks' weights:

$$\begin{aligned}\arg\max_{\theta_1,\theta_2,\phi} & \mathbb{E}_{q_\phi(z|x)q_\phi(y_c|x)}\left[\log p_{\theta_1}(x|z,y_c) + \gamma \log p_{\theta_2}(\Lambda|z,y_c)\right] \\ & -D_{KL}(q_\phi(z|x)||p(z)) - D_{KL}(q_\phi(y_c|x)||p(y_c)).\end{aligned} \tag{6}$$

Here θ_1, θ_2, ϕ are the weights of the neural networks constituting the decoder and encoder shown in Figure 2. γ is a real positive hyper-parameter which controls the weight assigned to the reconstruction of weak labels Λ relative to the standard ELBO components, i.e. reconstruction of x and KL divergence. To train the WS-VAE, we perform the maxmisation of equation 6 using the ADAM optimiser.

A.2. architecture of WS-VAE components

As schematically shown in figure 2, the WS-VAE architecture consists of three neural networks components:

- **Features Decoder:** This component consists of a fully connected neural network, which takes as input concatenated latent variables z and continuous label y_c. This neural network is made of an MLP of controllable depth and units sizes. Two linear matrices map the last hidden layer to the mean and log variance of features μ_x and $\log \sigma_x^2$.

- **Weak Labels Decoder:** This component also consists of a fully connected network, which takes as input concatenated latent variables z and continuous label y_c. This neural network is made of an MLP of controllable depth and uniots sizes. A single linear matrix maps the hidden layer to the logistic temperatures $t(z, y_c)$.

- **Features Encoder:** Similarly to the features decoder, this component consists of a fully connected neural network, which takes as input features x. This neural network is made of an MLP of controllable depth and units sizes. Four linear matrices map the hidden layer to the mean and log variance of latent variables μ_z and $\log \sigma_z^2$ and the mean and log variance of continuous label μ_{y_c} and $\log \sigma_{y_c}^2$.

B. Experimental Details

B.1. Data Sets

Our experiments are performed on the following 6 benchmark data-sets for binary classification tasks, made available with pre-computed weak labels in the Wrench framework (Zhang et al., 2021b):

- **YouTube:** Text data set of comments from YouTube for spam classification (Alberto et al., 2015). It comprises 1586 training examples and 250 test examples. 10 LFs were designed for the task.

- **IMDB:** Text data set of movie reviews from IMdb used for sentiment classification (Maas et al., 2011). It comprises 20000 training examples and 2500 test examples. 8 LFs were designed for the task.

- **SMS:** Text data set of mobile messages collected for spam classification (Gómez Hidalgo et al., 2006). It comprises 4571 training examples and 500 test examples. 73 LFs were designed for the task.

- **Tennis Rally:** Image data set of tennis match video frames, where the task is to classify if the players are in a rally (Fu et al., 2020). It comprises 6959 training examples and 1098 test examples. 6 LFs were designed for the task.

- **Commercial:** Image data set of television video frames, where the task is to classify if the frame is from a commercial (Fu et al., 2020). It comprises 64130 training examples and 7496 test examples. 4 LFs were designed for the task.

- **Census:** Tabular data set of census responses, where the task is to classify high and low income. (Kohavi et al., 1996). It comprises 10083 training examples and 16281 test examples. 83 LFs were designed for the task.

These data-sets span different training sample sizes, quantity and quality of LFs and data types, allowing us to test the WS-VAE in different regimes. Validation sample size is omitted in the above descriptions, as we do not use validation labels to fine-tune hyper parameters or perform early stopping.

B.2. WS-VAE Experimental Implementation

We keep hyper-parameters of models and training as consistent as possible across all data sets and all tested conditions, only changing layers sizes based on input features sizes. In all experiments the WS-VAE is trained with the Tensorflow ADAM optimiser for $10,000$ iterations, a batch size of 32 and an initial training rate of 0.001. With these hyper-parameters, the WS-VAE was observed to converge its cost

function in all tested conditions and all data-sets. The optimiser is set to maximise the ELBO of equation 6 with $\gamma = 100$ in all experiments, as we wish to approximately balance the reconstruction term of the weak labels Λ (4 to 83-dimensional) and that of the text features x (100 to 2048-dimensional). These settings are consistent across all data sets and experimental conditions tested. The latent dimensionality is also kept the same for all experiment and is equal to 10. The only difference of WS-VAE models across different data sets is the hidden size of the encoders and decoders; for the text data sets and the Census tabular data set, the encoder and decoder each consist of an MLP (as described in A.2) with a single 100-dimensional layer, while the weak labels decoder which infers the temperatures t_k consists of a 20-dimensional single layer MLP. For the image data sets, the structure is the same, with the difference that the encoder and decoder MLPs single layers are 1000-dimensional. These experimental settings were kept consistent across all tested conditions in all the experiments presented in this paper, with no fine-tuning or early stopping using validation ground-truth. This choice is made to truly test the robustness of the method and existing baselines to unforeseen adverse properties of the weak labels employed in truly unsupervised settings, where the labelling budget is 0.

B.3. Details of Baselines

As for the WS-VAE, in the absense of validation ground-truths, we keep hyper-parameters of models and training as consistent as possible across all data sets and all tested conditions, only changing layers sizes based on input features sizes.

WS Generative Models

- **Majority Voting:** Majority voting has no hyper-parameters and was used as implemented in the Wrench framework (Zhang et al., 2021b).
- **Snorkel:** In all experiments, the Snorkel algorithm was implemented with parameters $l_r = 0.01$, $l_2 = 0$ and $n_{epochs} = 1000$. The model is always optimised for the whole training steps without any validated early stopping, i.e. no validation with ground-truth labels.
- **Flying Squid:** In all experiments, the Flying Squid algorithm was implemented with the same parameters as Snorkel; $l_r = 0.01$, $l_2 = 0$ and $n_{epochs} = 1000$. as for Snorkel, no validtion for early stopping was implemented.

End Models

- **MLP:** The MLP end model is trained within the Wrench framework. A fully connected single layer neural network takes as input the pre-trained BERT features and outputs the class label. The model is trained with the ADAM optimiser for $20,000$ iterations with batch size 128 and initial training rate 0.001. As for the components of the WS-VAE, the MLP is composed of a single layer, which is 100-dimensional for experiments with text data and the Census data set and is 1000-dimensional with image data. This results in the MLP end model to be directly comparable to the WS-VAE encoder used for inference upon testing. As for the WS-VAE and the GMs, this end model is trained for the whole iterations, without performing early stopping through validation.
- **BERT:** In experiments with text data, we also fine-tune BERT as an end model, as is common in WS settings (Zhang et al., 2021b). A pre-trained BERT model for sequence classification (Devlin et al., 2018) with 128 max tokens is fine-tuned with the training set and the labels obtained with the WS generative models. The model is trained with the ADAMW optimiser in Wrench for $1,000$ iterations, with batch size 8 and initial training rate 5×10^{-5}. Here too, no validation ground-truth labels are assumed to be available and the model is fine-tuned for the whole iteration count.

Details of Joint Models

We compare to two WS joint models; Denoise (Ren et al., 2020) and Weasel (Rühling Cachay et al., 2021). Both are implemented in the Wrench framework (Zhang et al., 2021b). For text data sets, both models fine-tune a pre-trained BERT model as their backbone, while for image and tabular data sets they have a single layer MLP of 1000 and 100 units respectively. Denoise is implemented with Snorkel as internal label model and is trained with ADAMW for 10000 iterations, a batch size of 8, initial training rate of 5×10^{-5} and hyper-parameters $\alpha = 0.6$, $c_1 = 0.2$ and $c_2 = 0.7$ (default in Wrench implementation with YouTube data-set). Weasel is trained with ADAMW as well, for 4000 iterations, a batch size of 8 and initial training rate of 5×10^{-5}.

With all baselines, as with the proposed WS-VAE, we assume no ground-truth labels are available, even for validation. This means that hyper-parameters are fixed across all tested conditions, and largely also across data-sets, aside of model differences due to size and nature of data, and are not tuned or cross-validated with validation ground-truth in each case. It also means that we cannot evaluate task performance during training and perform early stopping at the best performing model, but we train to the last iteration in every experiment. This choice is to simulate and evaluate robustness in the fully unsupervised WS case.

	Mean Diff. from Best Baseline	Median Diff. from Best Baseline	p-value w. Best Baseline	Times Better than Best Baseline	Times Better than Any
YouTube	+0.109	+0.156	2.0×10^{-4}	84%	72%
SMS	+0.191	+0.194	2.0×10^{-8}	86%	66%
IMDB	+0.144	+0.085	4.5×10^{-4}	94%	56%
Tennis	+0.031	+0.018	3.5×10^{-4}	74%	58%
Commercial	+0.252	+0.065	4.2×10^{-7}	70%	58%
Census	+0.070	+0.067	4.1×10^{-7}	80%	80%

Table 3. Difference of macro F1-score distribution between WS-VAE and best baseline (chosen using highest mean F1-score). The table reports (left to right) the difference in the means of F1-score distributions between WS-VAE and best baseline, the difference in F1-scores distributions medians, the two-sample p-value between the two distribution, for statistical significance, the percentage of experiments in which WS-VAE resulted in a higher F1-score than the best baseline and the percentage of experiments in which the WS-VAE resulted in a higher F1-score than any baseline.

B.4. Details of LFs Degradation Experiments

Given a benchmark data set with its original weak labels from the Wrench framework (Zhang et al., 2021b), the experiments of Figure 4 were obtained by performing the following steps:

- **Alter Accuracy:** A ratio η of weak labels to randomly set to a random value is drawn from a uniform distribution between 0 and η_{max}. The non-abstaining training weak labels in the set are set to a random value with probability η. For YouTube, Tennis and Commercial $\eta_{max} = 0.5$, while for SMS, IMDB and Census $\eta_{max} = 0.25$.
- **Alter Coverage:** A ratio τ of weak labels to randomly set to abstain is drawn from a uniform distribution between 0 and τ_{max}. The non-abstaining training weak labels in the set are set to abstain with probability τ. For YouTube, Tennis and Commercial $\tau_{max} = 0.8$, while for SMS, IMDB and Census $\tau_{max} = 0.4$.
- **Add Correlated LFs:** a full coverage random list of weak labels is generated for the training set. Then, an integer N_c is drawn from a uniform distribution between 0 and 5. The list of random weak labels is appended N_c times to the matrix of existing training weak labels $\mathbf{\Lambda}$.
- **Train Models:** The resulting weak labels, along with the corresponding inputs, are used to train all baselines WS methods and the proposed WS-VAE.
- **Test Models:** The final labelling models are run over the test set and macro F1-scores with the test ground-truth labels are computed.

This procedure is repeated with a different random seed 50 times per data set to obtain the distributions shown in Figure 4.

C. Additional Results

C.1. Number of Labelling Functions Ablation

We perform a similar ablation experiment to those of Figure 3, where we vary the number of available LFs to perform programmatic WS on the YouTube data set. Experimental conditions are generated as follows: i) a number N_{LF} of available LFs is selected, ii) we draw N_{LF} LFs at random from the available set and iii) we perform WS using all baselines methods and the WS-VAE with the available weak labels. We repeat this procedure 5 times with a different random ssub-set of LFs to obtain error bars and gradually increase N_{LF} from 3 to 10 (All available LFs). Results are shown in Figure

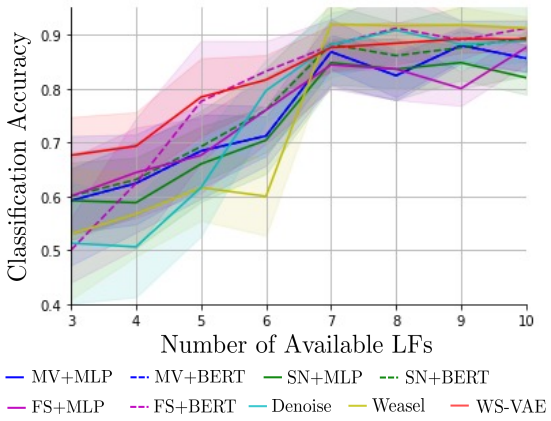


Figure 6. Classification accuracy on the YouTube data set test set at varying number of available LFs. Error bars are obtained by repeating experiments 5 times, each drawing a new random sub-set of LFs to be made available.

The WS-VAE is, on average, more accurate when less LFs are available (3-6) than competing baselines. All methods perform similarly as more LFs are made available (7- 10), when the quality and quantity of available weak labels is

relatively high. This result is in line with the rest of the ablation studies shown in Figure 3 and further proves the superior robustness of the WS-VAE to the engineering of LFs.

C.2. Comparison of Performance Distributions Under LFs Degradation

We report in table 3 measures of difference between the distributions of F1-scores computed in the combined degradations experiments and shown as box plots in Figure 4. We report several numerical metrics of the difference between the distribution of performance obtained with the WS-VAE and the best baseline, chosen as the baseline with the highest mean F1 score.