WEAK SUPERVISION VARIATIONAL AUTO-ENCODER

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in weak supervision (WS) techniques allow to mitigate the enormous labelling cost of human data annotation for deep learning by automating it using simple rule-based labelling functions (LFs). However, LFs need to be carefully designed, often requiring expert domain knowledge to be of sufficient accuracy, cover enough data and be independent of each other for existing WS methods to be viable. In addition, WS methods often rely on small amounts of validation data with true labels to fine-tune and select models. To tackle these issues, 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 and captures the input signals distribution with an artificial latent space, leading to considerably improved robustness to LFs quality. An extensive empirical evaluation on a standard WS benchmark shows that our WS-VAE performs competitively to existing WS methods while it is substantially more robust to LF engineering.

1 INTRODUCTION

One of the most severe bottlenecks in the successful development of supervised machine 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 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 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). In addition, small amounts of ground-truth labels are further required as validation to fine-tune models and perform early stopping during training (Ratner et al., 2017; Bach et al., 2019). These modelling assumptions and practical limitations restrict the performance and applicability of WS.

To mitigate these issues, we introduce the Weak Supervision VAE (WS-VAE); a new WS framework based on variational auto-encoders (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 encoder encodes input features into representation vectors. One dimension of these representation vectors is modelled as a continuous relaxation of the true hidden label. A specifically designed decoder reconstructs the input features and the weak labels jointly from the representation vectors. This architecture results in a weak labels-guided representation of the data, where one dimension captures the task of interest. Because the model learns from inputs and weak labels jointly by exploiting its artificial latent space to capture the features distribution, it is substantially more robust to LFs of low accuracy, coverage or high correlation, as well as less reliant on validation ground-truth labels for early stopping during training.

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 sources, which individually may have relatively low accuracy and coverage over the training data, and use them in combination 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 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 $\tilde{y}_i = argmaxp(y_i|\Lambda_i)$ (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 works focused instead on exploiting 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. 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).

Other recent approaches also combine stages 2 and 3 (Ren et al., 2020; Fieraru et al., 2021). These approaches train jointly a generative model and an end model, 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 incorporates unsupervised learning into the process to mitigate the reliance on LFs of high quality to recover labels of good accuracy.

2.2 Variational Auto-Encoders

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:

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

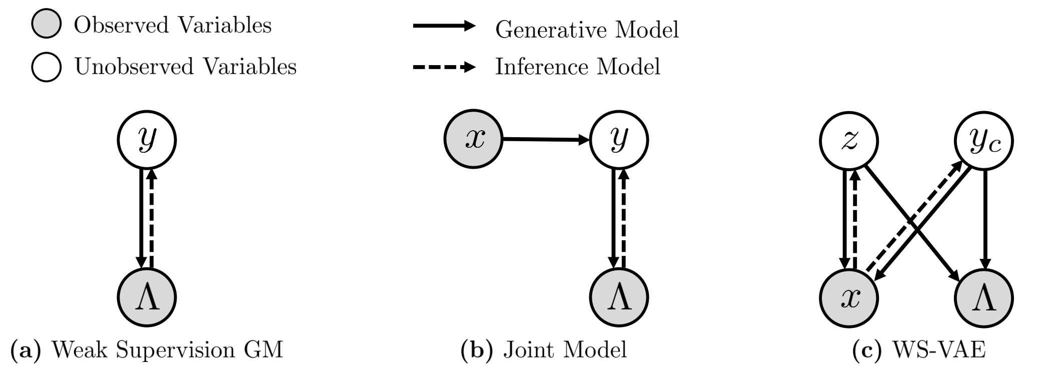


Figure 1: Graphical models for standard WS generative models compared to the proposed one: **(a)** standard weak supervision generative model, e.g. Snorkel, FlyingSquid. **(b)** joint model, such as Weasel, where the end model, mapping inputs x to labels y, is trained alongside the weak labels generative 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.

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. 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.

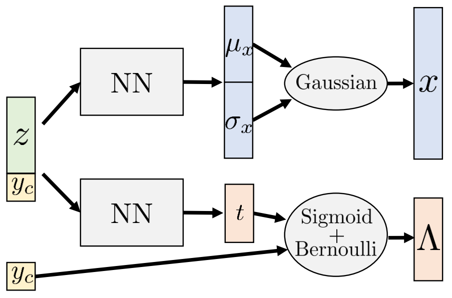
3 The Weak Supervision Variational Auto-Encoder (WS-VAE)

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 weak labelling, we design a decoder architecture which, alongside reconstructing input signals, infers the weak labels with a specific conditional distribution.

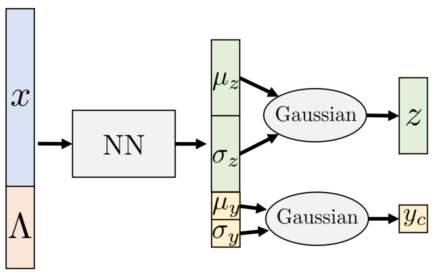
The model is trained by maximising a modified evidence lower bound (ELBO) which balances the reconstruction of the inputs, regularisation and inference of the weak labels. Once the model is trained, the encoder can be used to infer labels from new inputs. Our framework operates in a fully unsupervised weak labelling regime, i.e., with no training ground-truth labels. It can be used as a joint model, where stages 2 and 3 (described in section 2.1) are combined. The trained encoder can directly serve as end model. One can also use the WS-VAE as generative model (stage 2) and train a specialised end model (stage 3), e.g. BERT for natural language processing (NLP) tasks, using the soft labels inferred by the encoder as targets. In our experiments, we test both approaches.

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 $f(x)$ by first inferring labels $Y = \{y_1, y_2, ..., y_N\}$. We assume



(a) WS-VAE Decoder



(b) WS-VAE Encoder

Figure 2: Architectures of the WS-VAE decoder and encoder: **(a)** The decoder network. 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.

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 modelling, we take positive and negative classes to be $+1$ and -1 respectively and abstain to be 0, meaning that $\lambda \in \{+1, 0, -1\}, y \in \{+1, -1\}$. We focus on the binary classification case, where each LF can assign either a positive or negative label, or abstain. 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 wish 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 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 some latent variable of this model captures the source of variation in the observed data associated with the task. To infer this model and retrieve the source of variation associated with our task, we formalise the joint distribution of the inputs x and weak labels Λ as a latent variable model of the following form:

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

Here y_c is the latent variable associated with our task, which we model 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). 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.

Inputs 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.

Weak Labels Decoder: 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, we design the weak labels decoder $p(\Lambda|z, y)$ to be of a special form, tailored to the WS setting:

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

Here $t_k(z, y_c)$ are the temperatures of the logistic functions. They are the outputs of a neural network taking as input the latent variable z concatenated to the continuous label y_c. The weak labels $\lambda_k \in \Lambda$ are given a value 1 or -1 for the positive and negative class respectively, and 0 when abstaining.

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 learn to independently assign different importance to the classification of each weak label for each training example. In this way, the WS-VAE can ignore weak labels which are not consistent with the input representation and conversely assign higher importance to those that display meaningful correlations with the inputs. This generative model structure also allows to model correlations amongst LFs, as the common weak supervision 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's 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 features 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(b) illustrates the overall encoder architecture.

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:

$$\begin{aligned} \underset{\theta_1, \theta_2, \phi}{\arg\max} & \mathbb{E}_{q_\phi(z|x)q_\phi(y_c|x)} \left[\log p_{\theta_1}(x|z, y_c) + \gamma \log p_{\theta_2}(\Lambda|z, y_c)\right] \\ & - D_{KL}(q_\phi(z|x)||p(z)) - D_{KL}(q_\phi(y_c|x)||p(y_c)). \end{aligned} \tag{5}$$

Here θ_1, θ_2, ϕ are the weights of the neural networks constituting the decoder and encoder shown in Figure 2. To train the WS-VAE, we perform the maxmisation of equation 5 using the ADAM optimiser.

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:

$$\begin{aligned} q(y = 0|x) &= \int_{-\infty}^{0} q(y_c|x) dy_c = \frac{1}{2} + \frac{1}{2} erf(\frac{-\mu_{y_c}}{\sigma_{y_c}}) \\ q(y = 1|x) &= \int_{0}^{\infty} q(y_c|x) dy_c = \frac{1}{2} - \frac{1}{2} erf(\frac{-\mu_{y_c}}{\sigma_{y_c}}) \end{aligned} \tag{6}$$

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. Alternatively, the soft labels of equation 6 can be used to fine-tune a domain-specific model, such as a pre-trained BERT model for NLP classification tasks (Devlin et al., 2018). In our experiments, we test the WS-VAE in both settings. Architecture and training details of the WS-VAE are presented in Appendix A.1.

4 Experiments and Results

We test our WS-VAE on a standard publicly available WS benchmark, Wrench Zhang et al. (2021b), which consists of various NLP task, and compare to several state-of-the-art WS methods. Using the training split of the data, the WS-VAE is trained on pre-trained BERT encodings (Devlin et al., 2018) for sequence classification as inputs x and the weak labels provided as part of the data sets in the Wrench framework (see Section 4.2) as $\mathbf{\Lambda}$. After the model is trained, the encoder of the WS-VAE is used to infer soft labels as described in section 3.4. Hard labels are then computed from the soft labels by taking the class with the highest soft label and classification accuracy is computed.

4.1 Baseline Models

In our experiments, we compare with three WS generative models, in combination with two types of end models, and a joint model:

- **Majority Voting (MV):** The soft labels are a simple average of the non-abstaining labelling functions.
- **Snorkel (SN):** The Snorkel framework for WS, which trains a generative model for the weak labels Λ given the true label y and reconstructs it (Ratner et al., 2017).
- **FlyingSquid (FS):** A version of WS generative model which improves upon efficiency on Snorkel by finding a closed form solution for the GM parameters (Fu et al., 2020).

Implementation details for the GMs are presented in Appendix A.2. Each of the GMs listed above is trained with weak labels $\mathbf{\Lambda}$ from the training set and soft labels are then inferred. The soft or hard labels obtained with each generative model are then used to train two end models:

- **Multi-Layer Perception (MLP):** Pre-trained BERT encodings computed from text are used as inputs to train a single layer neural network with 100 hidden units, using hard labels from the GMs as targets.
- **BERT:** The pre-trained BERT model for sequence classification Devlin et al. (2018) is fine-tuned with the text inputs and soft labels from the GM as targets.

Implementation details for the end models are presented in Appendix A.3. We also compare with one joint model, which directly trains an end classification model with the weak labels $\mathbf{\Lambda}$:

- **Weasel:** A recently proposed joint method, where the weak supervision GM is trained jointly to the end model (Fieraru et al., 2021). This model uses BERT as a back-bone model and fine-tunes it during training. Implementation details are presented in Appendix A.4.

4.2 Data

We test all baseline models and the proposed WS-VAE on four WS text benchmark data sets taken from the WS framework Wrench: YouTube (Alberto et al., 2015), SMS (Gómez Hidalgo et al., 2006), Yelp (Zhang et al., 2015) and IMDB (Maas et al., 2011). All data sets are designed to test binary classification tasks; spam detection for the YouTube and SMS data sets, and sentiment classification for the Yelp and IMDB data sets. Each data set comprises text, which can be mapped to pre-trained BERT encodings we use as input features x, weak labels Λ and also ground truth labels y, which are only used to compute test accuracy and for validation, when experimental conditions require it.

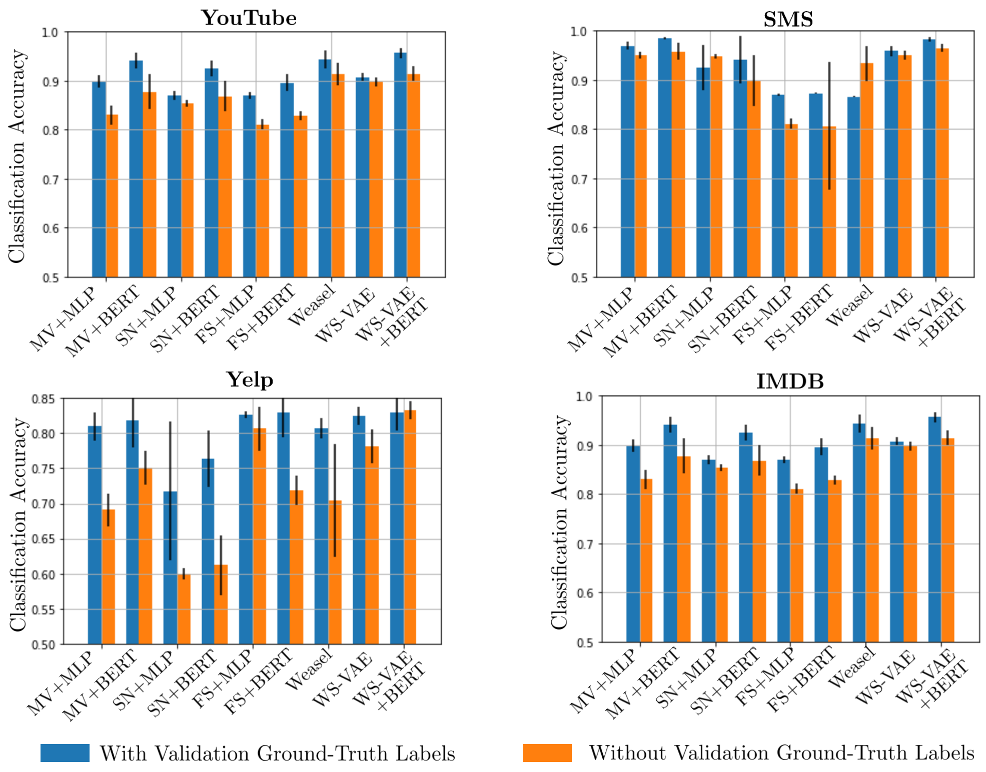


Figure 3: Weak supervision on benchmark data sets. Each of the baseline generative and end model combinations is tested, along with the proposed method. The WS-VAE is tested both as a joint model and as a GM in combination with BERT. Results are presented both for GM and end models which were trained using validation ground-truth labels for early stopping (blue) and models which were not provided any ground truth labels (orange).

4.3 Standard Benchmark Classification Results

In each experimental condition, we train the models in two different ways: (i) We assume to have access to a small amount of labelled validation data, which we use to evaluate performance during training and take the model at the iteration which maximises classification accuracy. This is the common approach to perform WS and is the default in the Wrench framework Zhang et al. (2021b); Ratner et al. (2017). (ii) We assume no labelling budget, even for validation, and simply train the models for a fixed number of iteration. The proposed WS-VAE is tested both as a joint model and as a GM, fine-tuning BERT as the end model. Figure 3 shows the predictive accuracy (over 5 runs with different random seed) for our WS-VAE models (WS-VAE and WS-VAE+BERT) and all baselines across tasks in Wrench.

- First, we observe that using a ground-truth validation set (blue bars), WS-VAE is competitive to the best performing baselines. However, using a fine-tuned BERT as an additional end model (WS-VAE+BERT), we obtain consistently better performance across all tasks. The improvement is expected, as BERT has larger capacity and is specifically designed for language tasks, but is also more computationally expensive.
- Second, we note that without using a validation set (orange bars), our WS-VAE demonstrates robust performance losing only 1-4% accuracy when validation ground-truth labels are removed. The more competitive baselines suffer from substantial drops, avareging 2-10%. From this result, we can see that existing WS methods often rely on validation ground truths and therefore need some labelling budget to be reliable, while the WS-VAE maintains good performance in truly unsupervised scenarios. This is because the WS-VAE incorporates representation of the inputs in its objective reducing over-fitting.

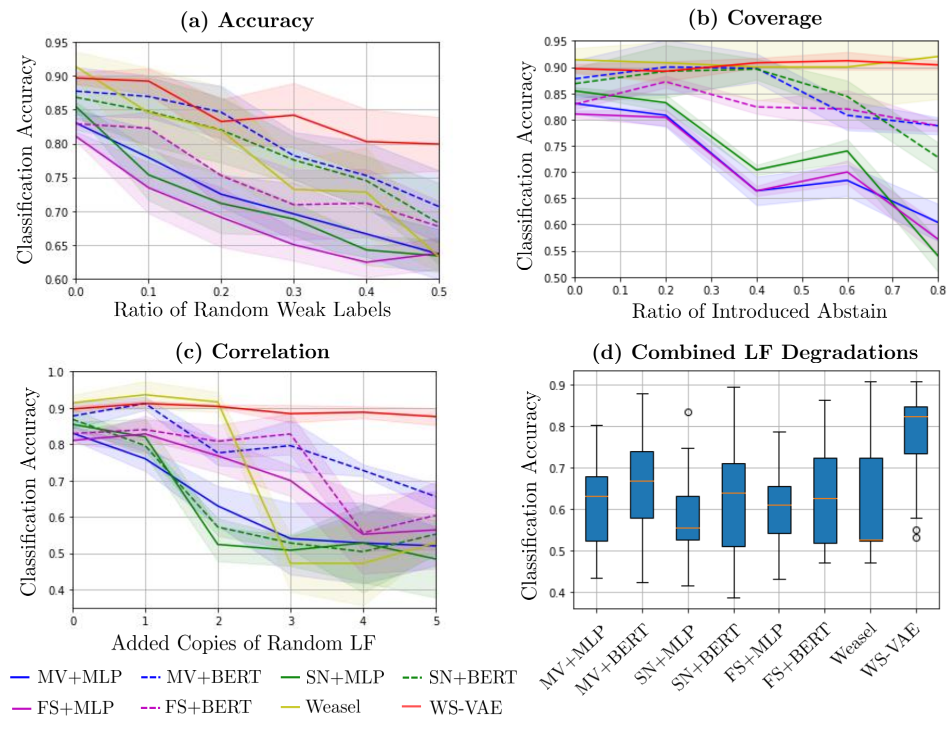


Figure 4: Classification accuracy in various LFs degradation conditions with the YouTube data set: **(a)** Classification accuracy vs. ratio of artificially added noise on the non-abstaining weak labels. **(b)** Classification accuracy vs. ratio of artificially abstained weak labels. **(c)** Classification accuracy vs. number of appended copies of randomly labelling LF. **(d)** Aggregate classification accuracy with random combinations of LF degradations. The LF degradations studied individually in a-c are combined with randomly drawn values and applied to the training LFs. The experiment is repeated 50 times to obtain aggregate distributions.

4.4 Robustness to LFs Quality

We further test the robustness of our WS-VAE and baselines to the LFs. We simulate a number of controlled settings where we artificially corrupt the quality of LFs in three aspects: accuracy, coverage and correlation. In real settings, LFs design efforts focus primarily on these aspects, as we want 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). To simulate these conditions, we modify the weak labels from the YouTube data set 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 uninformative LFs correlated to each other, we append from 1 to 5 copies of a random LF to the existing set.

We run two types of experiments, varying the parameters of the degradations described above: i) we incrementally vary each variable individually to study the effect of accuracy, coverage and correlation in isolation (Figure 4(a-c)); and ii) we make random combination of the three types of degradation and obtain aggregate distributions (Figure 4(d)). Results with the remaining 3 data sets are presented in Appendix B. For the former, we repeat each experiment 5 times to obtain errorbars, while for the latter we repeat the random degradation process 50 times to build distributions. As we are primarily interested in the fully unsupervised case, these experiments are performed without validation ground-truth labels. We make the following observations:

- **Accuracy (Figure 4(a)):** Overall, all methods suffer by gradually increasing the noise in the weak labels. 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 can learn to ignore weak labels which appear inconsistent.
- **Coverage (Figure 4(b)):** We can identify three groups of methods based on performance: i) MLP end models rapidly degrade towards $< 60\%$ as they tend to over-fit with only few labels available; ii) BERT end models limit this effect and achieve an accuracy between 70% and 80% with 80% abstaining labels. This is because pre-trained BERT is more robust to over-fitting when it is fine-tuned; iii) the two joint models, i.e., Weasel and our WS-VAE, are essentially unaffected by the decreasing coverage, remaining around 90% accuracy in all tested conditions. This is because they are both able to use features with no weak labels at all during training and greatly mitigate over-fitting.
- **Correlation (Figure 4(c)):** 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, which constitute a stronger signal than the existing weak labels, causing the models to fit to them. Contrarily, the WS-VAE can capture their correlation and ignore them in the modelling, if they display no mutual information with the associated inputs. As a result, the WS-VAE remains close to 90% accuracy as more copies of a random LF are added.
- **Combined Degradations (Figure 4(d)):** The superior robustness of the WS-VAE is clear when randomly combining the different degradations. All competing models present the bulk of their classification performance with random LFs degradation between 52% and 73%, while the WS-VAE spans a tighter distribution between 73% and 83%.

5 Conclusion

We proposed the WS-VAE; a novel WS framework that combines unsupervised representation learning and weak labelling. The WS-VAE models the joint distribution of inputs x and weak labels Λ with a latent variable model, representing them with a continuous label y_c, which captures the desired task, and a free latent variable z, which captures other sources of variation in the input data. Because the WS-VAE leverages representation learning of the inputs, in combination with the weak labels, it is inherently robust to the LFs and does not necessarily need validation ground-truth labels to train successfully. These properties mitigate arguably the largest outstanding bottlenecks of WS at scale; the definition of high quality LFs, which typically need to be expertly designed and fined-tuned to achieve sufficient accuracy and coverage, and be uncorrelated to each other.

In our benchmark experiments, the WS-VAE was found to consistently give competitive classification accuracy compared to existing methods over different WS benchmark tasks, proving its validity as a WS method. Additionally, when removing validation ground-truth labels for model selection in these tasks, the WS-VAE presented generally superior accuracy compared to the tested baselines, suggesting it is a fitting approach in completely unsupervised settings, where no labelling budget is available at all. The WS-VAE was also proven to be substantially more robust than competing methods to the quality of LFs. In our extensive evaluation, we controllably introduced defects in the LFs for spam classification with the YouTube data set by altering their accuracy, coverage and correlation. The WS-VAE was found to perform more consistently than any other tested method as the presence of defects is increased, with extreme cases reaching an average difference in accuracy of $9\% - 20\%$ compared to the second best method.

In future work, we aim to extend the WS-VAE model from binary tasks to multi-class tagging. 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

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

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pp. 159–168. PMLR, 2018.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. Learning from rules generalizing labeled exemplars. *arXiv preprint arXiv:2004.06025*, 2020.

Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, 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.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. Data programming using continuous and quality-guided labeling functions. *arXiv preprint arXiv:1911.09860*, 2019.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

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.

Zheng Ding, Yifan Xu, Weijian Xu, Gaurav Parmar, Yang Yang, Max Welling, and Zhuowen Tu. Guided variational autoencoder for disentanglement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7920–7929, 2020.

Mihai Fieraru, Mihai Zanfir, Teodor Szente, Eduard Bazavan, Vlad Olaru, and Cristian Sminchisescu. Remips: Physically consistent 3d reconstruction of multiple interacting people under weak supervision. *Advances in Neural Information Processing Systems*, 34, 2021.

Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*, pp. 3280–3291. PMLR, 2020.

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

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

José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, and Francisco Carrero García. Content based sms spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering*, pp. 107–114, 2006.

Cheng-Yu Hsieh, Jieyu Zhang, and Alexander Ratner. Nemo: Guiding and contextualizing weak supervision for interactive data programming. *arXiv preprint arXiv:2203.01382*, 2022.

Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan. 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.

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

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

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

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.

Ayush Maheshwari, Oishik Chatterjee, Krishnateja Killamsetty, Ganesh Ramakrishnan, and Rishabh Iyer. Semi-supervised data programming with subset selection. *arXiv preprint arXiv:2008.09887*, 2020.

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

Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In *Proceedings of the Advances in Neural Information Processing Systems*, 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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4763–4771, 2019.

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, 2016a.

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, 2016b.

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

Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. Denoising multi-source weak supervision for neural text classification. *arXiv preprint arXiv:2010.04582*, 2020.

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

Changho Shin, Winfred Li, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. Universalizing weak supervision. *arXiv preprint arXiv:2112.03865*, 2021.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*, 3(2), 2016.

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

Francesco Tonolini, Bjørn Sand Jensen, and Roderick Murray-Smith. Variational sparse coding. In *Uncertainty in Artificial Intelligence*, pp. 690–700. PMLR, 2020.

Paroma Varma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Ré. Learning dependency structures for weak supervision models. In *International Conference on Machine Learning*, pp. 6418–6427. PMLR, 2019a.

Paroma Varma, Frederic Sala, Shiori Sagawa, Jason Fries, Daniel Fu, Saelig Khattar, Ashwini Ramamoorthy, Ke Xiao, Kayvon Fatahalian, James Priest, et al. Multi-resolution weak supervision for sequential data. *Advances in Neural Information Processing Systems*, 32, 2019b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. 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.

Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. Nestedvae: Isolating common factors via weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9202–9212, 2020b.

Peilin Yu, Tiffany Ding, and Stephen H Bach. Learning from multiple noisy partial labelers. *arXiv preprint arXiv:2106.04530*, 2021.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. *arXiv preprint arXiv:2010.07835*, 2020.

Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, and Patrick Lucey. Generating multi-agent trajectories using programmatic weak supervision. *arXiv preprint arXiv:1803.07612*, 2018.

Jieyu Zhang, Bohan Wang, Xiangchen Song, Yujing Wang, Yaming Yang, Jing Bai, and Alexander Ratner. Creating training sets via weak indirect supervision. *arXiv preprint arXiv:2110.03484*, 2021a.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. Wrench: A comprehensive benchmark for weak supervision. *arXiv preprint arXiv:2109.11377*, 2021b.

Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*, 2022.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.

Appendix

A Experimental Details

A.1 WS-VAE Architecture and Training

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. For all experiments, this neural network is made of a single hidden layer with 100 units. Two linear matrices map the 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 neural network, which takes as input concatenated latent variables z and continuous label y_c. For all experiments, this neural network is made of a single hidden layer with 20 units. A single linear matrix maps the hidden layer to the logistic temperatures $t(z, y_c)$.
- **Features Encoder:** This component consists of a fully connected neural network, which takes as input features x. For all experiments, this neural network is made of a single hidden layer with 100 units. 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 $\log \sigma_{y_c}^2$.

In all experiments the WS-VAE is trained with the Tensorflow ADAM optimiser for $5,000$ iterations, a batch size of 32 and an initial training rate of 0.001. When performing early stopping, the model is evaluated every 250 iterations. The optimiser is set to maximise the ELBO of equation 5 with $\gamma = 100$, as we wish to approximately balance the reconstruction term of the weak labels Λ (8 to 73-dimensional) and that of the text features x (728-dimensional). These settings are consistent across all data sets and experimental conditions tested.

A.2 Details of WS Generative Models Baselines

- **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 as is default in the Wrench framework, with parameters $l_r = 0.01$, $l_2 = 0$ and $n_{epochs} = 10$.
- **Flying Squid:** In all experiments, the Flying Squid algorithm was implemented as is default in the Wrench framework, with the same parameters as Snorkel; $l_r = 0.01$, $l_2 = 0$ and $n_{epochs} = 10$.

A.3 Details of 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. If early stopping is applied with validation data, the model is evaluated every 10 iterations and the best performing model over the validation set is picked (standard in Wrench framework).
- **BERT:** 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 model. 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}. If early stopping is applied with validation data, the model is evaluated every 10 iterations with a random batch of 64 validation samples and the best performing model is picked (standard in Wrench framework).

A.4 DETAILS OF WEASEL

The joint model Weasel is implemented in the Wrench framework with default parameters; hidden size of 100, a BERT back-bone, temperature 1 and 0.3 dropout. In all experiments, the Weasel model is trained with the ADAMW optimiser for $2,000$ iterations, a batch size of 4 and initial training rate 5×10^{-5}. If early stopping is applied with validation data, the model is evaluated every 10 iterations with a random batch of 8 validation samples and the best performing model is picked (standard in Wrench framework). The batch sizes for training and early stopping evaluation are set lower than for the BERT end model as the Weasel model has higher GPU memory requirement and could not run on our hardware set-up with larger batch sizes.

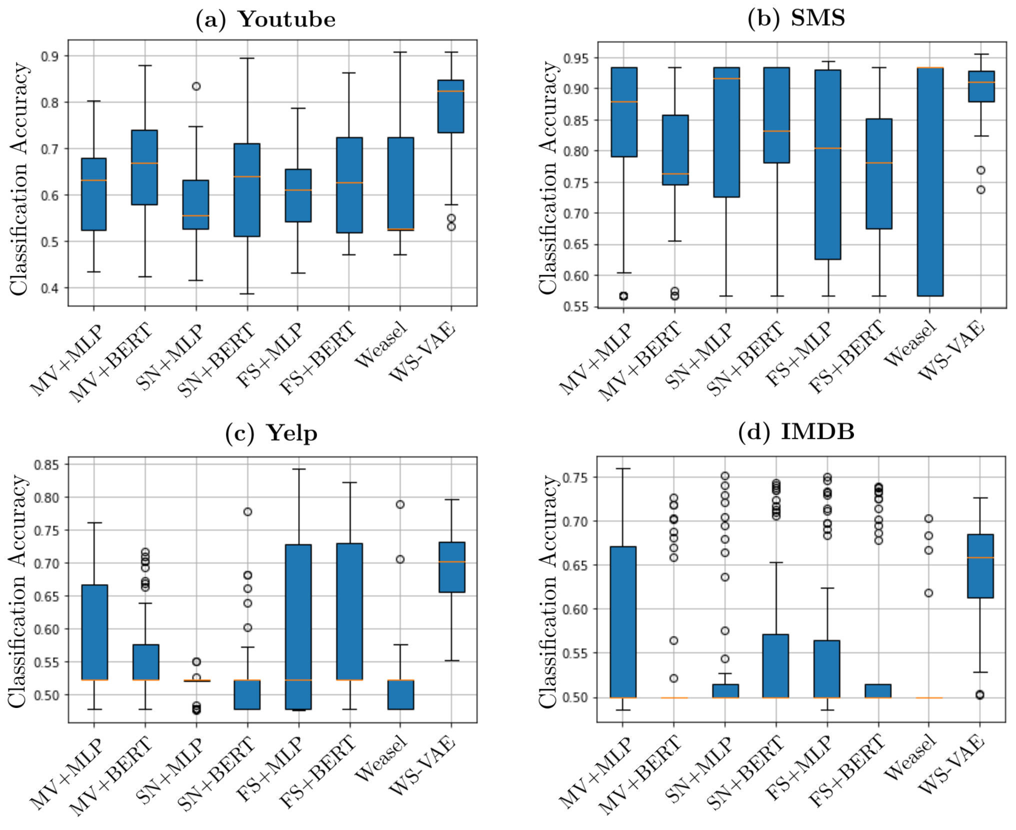


Figure 5: Classification performance box plots with random combinations of LF degradation conditions.

B LF Robustness Results on All Data Sets

We repeat the experiment of Figure 4(d) with each of the four data sets described in section 4.2. For each data set, as for the experiments with the YouTube data set of Figure 4(d), random combinations of LF degradation parameters are drawn at random and applied to the data set. The ratio of random substitution is drawn uniformly from the interval $\{0, 0.5\}$, the ratio of artificially abstained labels is drawn uniformly from the interval $\{0, 0.8\}$ and the number of copies of a randomly generated set of weak labels is drawn uniformly in the interval $\{0, 5\}$. Random combinations are drawn 50 times, each time all baseline methods and the proposed WS-VAE are trained using the resulting weak labels and results are aggregated to obtain performance distributions. Results are shown in Figure 5.

The substantially superior robustness of the WS-VAE demonstrated in the combined degradations results with the YouTube data set of Figure 4(d) is confirmed in the remaining three data sets (SMS, Yelp and IMDB), as in all cases the WS-VAE results in a distribution of performance which is both higher in its average and tighter. With the SMS data set, the bulk of the WS-VAE distribution of accuracy is between 88% and 93%, while those of competing methods are both lower in their average and vary substantially more for the same random degradations, with the second best (MV+MLP) spanning 79% to 94%. The same general trend is observed with the Yelp data set, with the WS-VAE distribution presenting the box boundaries between 66% and 73% accuracy, while the second best (FS+BERT) spans a much larger interval between 53% and 73% accuracy. With the IMDB data set, the competing methods were found to be even more fragile to LF design, with almost all methods often falling to approximately 50% accuracy (random guessing) with random degradations, while the WS-VAE presents the box boundaries of its accuracy distribution between 61% and 68%.