Labeled Data Generation with Inexact Supervision

Enyan Dai[†], Kai Shu[‡], Yiwei Sun[†], Suhang Wang[†]
† The Pennsylvania State University, ‡ Illinois Institute of Technology
{emd5759,yus162,szw494}@psu.edu,kshu@iit.edu

ABSTRACT

The recent advanced deep learning techniques have shown the promising results in various domains such as computer vision and natural language processing. The success of deep neural networks in supervised learning heavily relies on a large amount of labeled data. However, obtaining labeled data with target labels is often challenging due to various reasons such as cost of labeling and privacy issues, which challenges existing deep models. In spite of that, it is relatively easy to obtain data with *inexact supervision*, i.e., having labels/tags related to the target task. For example, social media platforms are overwhelmed with billions of posts and images with self-customized tags, which are not the exact labels for target classification tasks but are usually related to the target labels. It is promising to leverage these tags (inexact supervision) and their relations with target classes to generate labeled data to facilitate the downstream classification tasks. However, the work on this is rather limited. Therefore, we study a novel problem of labeled data generation with inexact supervision. We propose a novel generative framework named as ADDES which can synthesize high-quality labeled data for target classification tasks by learning from data with inexact supervision and the relations between inexact supervision and target classes. Experimental results on image and text datasets demonstrate the effectiveness of the proposed ADDES for generating realistic labeled data from inexact supervision to facilitate the target classification task.

KEYWORDS

Generative Model, Labeled Data Generation, Data Augmentation, Graph Neural Network

ACM Reference Format:
Enyan Dai[†], Kai Shu[‡], Yiwei Sun[†], Suhang Wang[†] . 2021. Labeled Data Generation with Inexact Supervision. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467306

1 INTRODUCTION

Deep learning technologies have achieved remarkable results in various domains such as image classification [11], object detection [29] and language translation [7]. However, training the deep neural networks relies on a large amount of labeled data, which is impractical to obtain in many domains. Taking the fake news detection for example, a news piece often requires hours of work from a professional to evaluate the credibility which leads to unaffordable time and labor costs. For applications in healthcare, it is difficult to obtain large-scale labeled data (e.g. EHR data) due to privacy issues and the scarcity of experts for labeling.

Table 1: Samples of StackOverflow.

Question Tittles	Tags	Target Labels
Laravel 4: Input::all() returns no data with $.ajax POST	ajax, laravel	php
jquery DataTables.net plugin: how to ignore rows when sorting	jquery, html	javascript

In spite of the difficulties in obtaining the accurately labeled data for target problems, the development of the internet and social media makes it easy to collect data with *inexact supervision*, i.e., labels/tags related to the target classification task. For example, users often post images and texts with self-customized tags on social media platforms such as Twitter, Facebook, and StackOverflow. Though these tags are often not the labels of target classes, they could provide inexact supervision through the relations between the tags and the labels of the target classification task. Table 1 gives two real examples of inexact supervision from StackOverflow, where questions are labeled with several tags and the target task is to assign the programming language to the questions based on their text. Obviously, these tags are not the target labels. However, they have relations with the labels of the target classes, which could provide inexact supervision for target label prediction. For instance, in the first example, the tag `laravel` shows that the question is related with `php`, because `laravel` is a framework that designed for developing `php`. In the second example, the tag `jquery`, which is a JavaScript library, suggests that the text is likely to be related to the target label `javascript`. Thus, these tags could be used to help infer target labels even though no exact supervision (data with target labels) is given. There are various applications that could benefit from inexact supervision such as image classification for Flickr and short video classification for Instagram. Therefore, it is important to study learning from inexact supervision.

The recent development of deep generative models such as generative adversarial learning (GAN) [8] and variational autoencoder [18] have shown promising results in generating realistic labeled data. The generated labeled data could be used to augment the dataset or facilitate the downstream classification tasks [1, 32, 38, 39]. For example, data augmentation by GAN is shown effective for few-shot learning [1]. Shu et al. [32] utilize the generated headlines for clickbait detection. Therefore, it is promising to develop deep generative models for generating labeled data from inexact

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467306

supervision to facilitate the training of target classifiers. However, the work on labeled data generation with inexact supervision is rather limited.

Therefore, we investigate a novel problem of labeled data generation with inexact supervision. In essence, we aim to tackle the following challenges: (i) how to extract the information of the target labels from the labels of inexact supervision classes; (ii) how to generate high-quality labeled data for classification. In an attempt to solve these two challenges, we propose a novel generative framework named as **ADDES** (labeled data generation with inexact supervision). To better infer the information for target label prediction, ADDES adopts a graph convolutional network (GCN) to capture the label relations. The information propagation among the nodes which represent different classes could utilize the supervision from labels of the inexact supervision classes. Furthermore, to obtain high-quality synthetic data, the framework is designed to utilize both the data with inexact supervision and unlabeled data. Our main contributions are:

- We study a novel problem of labeled data generation with inexact supervision for data augmentation;
- We propose a novel generative framework which could leverage data with inexact supervision, unlabeled data, and the relations between classes to generate high-quality labeled data; and
- Experiments on the benchmark datasets, including the image and text datasets, demonstrate the effectiveness of our proposed framework for labeled data generation with inexact supervision.

The rest of the paper are organized as follows. In Section 2, we introduce related work. In Section 3, we formally define the problem. In Section 4, we introduce the proposed method. In Section 5, we conduct experiments to demonstrate the effectiveness of the proposed method. In Section 6, we conclude with future work.

2 RELATED WORK

2.1 Deep Generative Model

Generative model aims to capture the distribution of the real data. Recently, deep generative models such as generative adversarial networks (GANs) [8] and variational autoencoder (VAE) [18], have attracted increasing attention as a result of their strong power in generating realistic data samples. Based on GAN and VAE, various efforts [3, 13, 17, 23, 25, 34] have been taken to generate realistic data with desired labels. For example, conditional GAN and conditional VAE are proposed to learn the conditional probability distribution of real data [23, 34]. Controlled generation of text based on VAE is also explored [3, 13]. What's more, various applications of the generative models are investigated. One major application is to generate labeled data for data augmentation [1, 32, 38, 39, 41]. For example, data augmentation based on generative adversarial networks is explored in [1]. In clickbait detection, headlines are generated to augment the data for better performance [32]. In contrast to those prior works that require large-scale accurately labeled data to learn generative models for synthesizing labeled data, we investigate a new problem of generating labeled data without the ground truth of target labels. Moreover, the proposed framework ADDES is a unified framework that could effectively synthesize images and text.

2.2 Learning from Weak Supervision

For many real-world applications, obtaining large-scale high quality labels are difficult, while it is relatively easy weak supervision [28, 46] such as noisy supervision [36, 43] and distant supervision [27]. Thus, learning from weak supervision is attracting increasing attention and various approaches are proposed [10, 27, 36, 43]. For example, Xiao et al. [43] model the relationships between images, class labels and label noises with a probabilistic graphical model and further integrate it into an end-to-end deep learning system. Han et al. [10] presents a novel deep self-learning framework to train a robust network on the real noisy datasets without extra supervision. Qin et al. [27] adopts generative adversarial training with distant supervision for relation extraction. Despite the various approaches for learning from weak supervision, the majority of them focus on noisy supervision and distant supervision. The work on learning from inexact labels is rather limited, let labeled data generation from ineact labels.

2.3 Multi-Label Classification

Multi-label classification is to predict a set of labels for an instance. The key challenge of multi-label learning is the overwhelming size of the possible label combinations. One straightforward way is to decompose the multi-label classification to a set of binary classification problems [2]. To achieve better performance, researchers investigate a number of methods to capture the label dependencies. For example, Wang et al. [37] use recurrent neural networks to model the high-order label dependency. Chen et al. [4] propose the ML-GCN to leverage the knowledge graph to explore the label correlation dependency. In our inexact supervision problem setting, we also assume that a data instance could have multiple labels. However, the multi-label learning assumes that all the labels of the instance are provided. We are dealing with a much more challenging problem that ground truth of target labels is totally missing in the training set, and our goal is to generate data of desired target labels for data augmentation.

2.4 Zero-Shot Learning

Zero-shot Learning (ZSL) aims to make classifications for the target categories when no data in these categories is provided. In this setting, only labeled data in the source categories is available for training. To transfer the knowledge learned from the source categories to the target categories, semantic embeddings such as word embeddings of the categories are utilized. Typical methods are to learn a compatibility function between the data and the semantic embeddings based on the source category images [6, 30, 45]. Another direction is to sample features for the target categories from semantic embeddings through generative model [20, 24, 42]. Recently, knowledge graph is adopted in zero-shot learning, which results in remarkable results [15, 21, 40]. For example, Wang et al. [40] build a graph linking the related categories together and use the GCN to predict the classifiers of the target categories from semantic embeddings. Although ZSL deals with the lack of labeled data in target categories, there is a distinct difference between zero-shot learning and inexact supervision learning. In zero-shot learning, an instance is supposed to belong to a single class. None of the supervision to the target category classification could be obtained from the seen categories data. On the contrary, we are interested in more practical scenarios in which seen labels of data could provide inexact supervision and be leveraged for target label prediction.

3 PROBLEM DEFINITION

Let $\mathcal{S} = \{l_1, l_2, ..., l_{|\mathcal{S}|}\}$ denotes the inexact supervision class set of size $|\mathcal{S}|$, and $\mathcal{T} = \{l_{|\mathcal{S}|+1}, l_{|\mathcal{S}|+2}, ..., l_{|\mathcal{S}|+|\mathcal{T}|}\}$ denotes the target class set of size $|\mathcal{T}|$. Then the whole class set is $\mathcal{W} = \mathcal{S} \cup \mathcal{T}$. Note that the target class set has no overlap with the inexact supervision class set, i.e., $\mathcal{S} \cap \mathcal{T} = \emptyset$. The label vectors $\mathbf{y}_s \in \{0,1\}^{|\mathcal{S}|}$ and $\mathbf{y}_t \in \{0,1\}^{|\mathcal{T}|}$ accordingly represent the labels of inexact supervision classes and target classes. For a data instance x^k, if $\mathbf{y}_s^k(i) == 1$, it means the instance belongs to the class l_i, otherwise not. A data instance can belong to multiple classes. The whole training set $\mathcal{D}$ contains an inexact supervision class labeled data set $\mathcal{D}_l$ consisting of n_l instances $(\mathbf{x}^l, \mathbf{y}_s^l)$ and an unlabeled data set $\mathcal{D}_u$ with n_u unlabeled instances $\mathbf{x}^u$. The total training set can be written as:

$$\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u = \{(\mathbf{x}^l, \mathbf{y}_s^l)\}_{l=1}^{n_l} \cup \{\mathbf{x}^u\}_{u=1}^{n_u}, \tag{1}$$

The labels from the whole class set $\mathcal{W}$ are correlated with each other. With the relations, the class graph $\mathcal{G} = \{\mathcal{W}, \mathcal{E}\}$ is constructed, where $\mathcal{E} \subset \mathcal{W} \times \mathcal{W}$ is set of edges linking the classes. We use $\mathbf{A} \in \mathbb{R}^{|\mathcal{W}| \times |\mathcal{W}|}$ to denote the correlation matrix. The weight $\mathbf{A}_{ij}$ indicates how likely the labels of classes l_i and l_j are both annotated to 1 in a single instance. The class embeddings matrix is denoted as $\mathbf{V} \in \mathbb{R}^{|\mathcal{W}| \times m}$, where m is the dimension of the class embeddings. For example, the class embedding can be word embedding denoting the semantic meaning of the class or one hot encoding if word embedding is not available. With the notations and definitions described here, the problem of labeled data generation with inexact supervision for data augmentation could be formulated as:

Problem 1. *Given the training set $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$ and the graph $\mathcal{G}$ with the adjacency matrix $\mathbf{A}$ and class embeddings $\mathbf{V}$, we aim to learn a generative model and produce a set of labeled data $\mathcal{D}_s = \{(\mathbf{x}^{syn_i}, \mathbf{y}_s^{syn_i}, \mathbf{y}_t^{syn_i})\}_{i=1}^{n_s}$ through the following process:*

$$f(\mathbf{A}, \mathbf{V}, \mathbf{y}_s^{syn}, \mathbf{y}_t^{syn}) \rightarrow \mathbf{x}^{syn}, \tag{2}$$

where f is the generative model required to learn.

4 METHODOLOGY

The proposed generative framework consists of three modules: an encoder $q_E(\mathbf{z}|\mathbf{x})$, a decoder $p_D(\mathbf{x}|\mathbf{z}, \mathbf{y}_s, \mathbf{y}_t)$ and a GCN-based classifier $q_C(\mathbf{y}_s, \mathbf{y}_t|\mathbf{x}, \mathcal{G})$, which are presented in Figure 2. The encoder is to learn a latent variable $\mathbf{z}$ disentangled with the inexact supervision label vector $\mathbf{y}_s$ and target label vector $\mathbf{y}_t$. The classifier utilizes the graph convolutional network to better infer $\mathbf{y}_t$ and $\mathbf{y}_s$ with the inexact supervision and unlabeled data. With the latent variable sampled from the prior $p(\mathbf{z})$ or posterior $p_E(\mathbf{z}|\mathbf{x})$, the model could synthesize a new data point $\mathbf{x}^{syn}$ corresponding to the assigned labels $(\mathbf{y}_t^{syn}, \mathbf{y}_s^{syn})$. Next, we will first introduce the probabilistic generative model for estimating data distribution followed by the deep learning framework to realize the generative model.

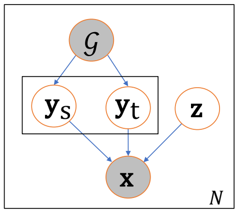


Figure 1: The Probabilistic Graphical Model of ADDES.

4.1 A Probabilistic Generative Model

Our goal is to synthesize labeled data for the learning of target label predictor when only the labels of inexact supervision classes are available. To achieve this, we assume that the data is sampled from the generative process presented in Figure 1. As shown in the figure, the data lies in a low dimension space and the latent presentation is divided into three parts: (i) $\mathbf{z}$, the latent features irrelevant with the labels; (ii) $\mathbf{y}_s$, the label vector of inexact supervision classes; (iii) $\mathbf{y}_t$, the label vector of target classes. The $\mathbf{y}_s$ and $\mathbf{y}_t$ are related with the dependency encoded by the graph $\mathcal{G}$. To generate labeled data, latent feature vector $\mathbf{z}$ is assumed to be independent with $\mathbf{y}_s$ and $\mathbf{y}_t$. With the disentangled representation, novel labeled data could be produced through varying $\mathbf{z}$, $\mathbf{y}_s$ and $\mathbf{y}_t$. Next, we give the details of the generative framework.

4.1.1 Modeling Data Distribution. As shown in Figure 1, the joint distribution $p(\mathbf{x}, \mathbf{z}, \mathbf{y}_s, \mathbf{y}_t|\mathcal{G})$ could be written as:

$$p(\mathbf{x}, \mathbf{z}, \mathbf{y}_s, \mathbf{y}_t|\mathcal{G}) = p(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})p(\mathbf{y}_s, \mathbf{y}_t|\mathcal{G})p(\mathbf{z}), \tag{3}$$

where $p(\mathbf{z})$ is the prior distribution of the latent variable $\mathbf{z}$. Usually, $p(\mathbf{z})$ is chosen as normal distribution, i.e. $p(\mathbf{z}) \sim N(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix. For data with labels of inexact supervision classes, i.e., $(\mathbf{x}, \mathbf{y}_s) \in \mathcal{D}_l$, we aim to optimize the variational lower bound (ELBO) of $\log p(\mathbf{x}, \mathbf{y}_s|\mathcal{G})$ as:

$$\log p(x, \mathbf{y}_s|\mathcal{G}) \geq \mathbb{E}_q[\log \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{y}_s, \mathbf{y}_t|\mathcal{G})}{q(\mathbf{y}_t, \mathbf{z}|\mathbf{y}_s, \mathbf{x}, \mathcal{G})}], \tag{4}$$

where $q(\mathbf{y}_t, \mathbf{z}|\mathbf{y}_s, \mathbf{x}, \mathcal{G})$ is an auxiliary distribution to approximate $p(\mathbf{y}_t, \mathbf{z}|\mathbf{y}_s, \mathbf{x})$. To simplify the approximation process, we assume a factorized form of the auxiliary distribution:

$$q(\mathbf{y}_t, \mathbf{z}|\mathbf{y}_s, \mathbf{x}, \mathcal{G}) = q_C(\mathbf{y}_t|\mathbf{x}, \mathcal{G})q_E(\mathbf{z}|\mathbf{x}). \tag{5}$$

Then the ELBO of $\log p(\mathbf{x}, \mathbf{y}_s|G)$ could be re-formulated as:

$$\begin{aligned}\log p(\mathbf{x}, \mathbf{y}_s|\mathcal{G}) \geq & \mathbb{E}_q[\log p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})] - KL[q_E(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \\ & - KL[q_C(\mathbf{y}_t|\mathbf{x}, \mathcal{G})||p(\mathbf{y}_t|\mathbf{y}_s, \mathcal{G})] \\ & = \mathcal{L}_G^l(\mathbf{x}, \mathbf{y}_s),\end{aligned} \tag{6}$$

Similarly, for unlabeled instance $\mathbf{x} \in \mathcal{D}_u$, we aim to optimize the variational lower bound of $\log p(\mathbf{x}|\mathcal{G})$:

$$\begin{aligned}\log p(x|\mathcal{G}) \geq & \mathbb{E}_q[\log \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{y}_s, \mathbf{y}_t|\mathcal{G})}{q(\mathbf{y}_s, \mathbf{y}_t, \mathbf{z}|\mathbf{x}, \mathcal{G})}] \\ & = \mathbb{E}_q[\log p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})] - KL[q_E(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \\ & - KL[q_C(\mathbf{y}_t, \mathbf{y}_s|x, \mathcal{G})||p(\mathbf{y}_t, \mathbf{y}_s|\mathcal{G})] \\ & = \mathcal{L}_G^u(\mathbf{x}),\end{aligned} \tag{7}$$

With Eq.(6) and Eq.(7), the loss function on the whole training set $\mathcal{D}$ could be written as:

$$\mathcal{L}_G = \mathbb{E}_{p_l(\mathbf{x}, \mathbf{y}_s)}\mathcal{L}_G^l(\mathbf{x}, \mathbf{y}_s) + \mathbb{E}_{p_u(\mathbf{x})}\mathcal{L}_G^u(\mathbf{x}), \tag{8}$$

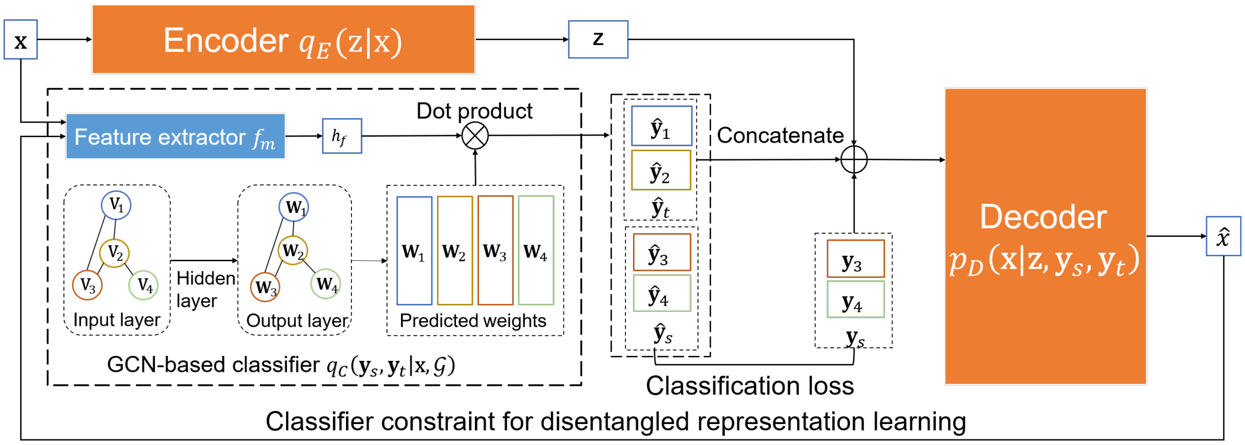


Figure 2: Overall model architecture of ADDES. It shows how to process data with labels of inexact supervision classes.

where $p_l(\mathbf{x}, \mathbf{y}_s)$ denotes the distribution of $\mathcal{D}_l$, and $p_u(\mathbf{x})$ denotes the distribution of unlabeled dataset $\mathcal{D}_u$.

4.1.2 Enforcing Disentangled Representation Learning. The representation $\mathbf{z}$ should not contain any label information so that we can vary $\mathbf{y}_s$ and $\mathbf{y}_t$ to generate labeled data by sampling from $p(\mathbf{x}|\mathbf{z}, \mathbf{y}_s, \mathbf{y}_t)$. However, since $\mathbf{x}$ covers the information of the labels, the latent variable $\mathbf{z}$ obtained from the encoder $q_E(\mathbf{z}|\mathbf{x})$ might correlate with $\mathbf{y}_s$ and $\mathbf{y}_t$. Thus, $\mathbf{y}_s, \mathbf{y}_t$ actually may not contribute to the generation of $\mathbf{x}$ as $\mathbf{z}$ already contains the label information. Therefore, we need to ensure $\mathbf{z}$ is independent with the class-attribute. To learn the disentangled representations, we add a constraint to enforce the data produced from the decoder to match the assigned labels. The objective function could be formulated as:

$$\mathcal{L}_{cons} = \mathbb{E}_{p(\mathbf{y}_s, \mathbf{y}_t) p_D(\hat{\mathbf{x}}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})}[\log q_C(\mathbf{y}_s, \mathbf{y}_t|\hat{\mathbf{x}}, \mathcal{G})], \tag{9}$$

where $\hat{\mathbf{x}} = D(\mathbf{z}, \mathbf{y}_s, \mathbf{y}_t)$ is the generated data from the decoder with sampled latent variable $\mathbf{z}$ and class-attribute. With this constraint, during training, we can vary $\mathbf{y}_s, \mathbf{y}_t$ for various data of desired labels. The regularizer will then check if the generated $\hat{\mathbf{x}}$ has the desired labels to enforce the involvement of $\mathbf{y}_s, \mathbf{y}_t$ in data generation. The distribution $p(\mathbf{y}_s, \mathbf{y}_t)$ has multiple choices. When it is the prior distribution $p(\mathbf{y}_s, \mathbf{y}_t|\mathcal{G})$. This would enable the sampled data $\mathbf{x}^{syn}$ have the desired labels. When $p(\mathbf{y}_s, \mathbf{y}_t)$ is the posterior $q_D(\mathbf{y}_t^l|\mathbf{x}^l, \mathcal{G})$ of data labeled as $\mathbf{y}_s^l$ in inexact supervision classes or posterior $q_D(\mathbf{y}_s^u, \mathbf{y}_t^u|\mathbf{x}^u, \mathcal{G})$ of unlabeled data, this constraint will assist the reconstruction of input data by providing extra semantic level supervision.

To obtain disentangled representation, $q_C(\mathbf{y}_s, \mathbf{y}_t|x, \mathcal{G})$ is used as a classifier to constrain the encoder and decoder in Eq.(9). This implies the predictions of the classifier are accurate. However, the presented loss functions may be not sufficient to model the classifier well, because the predictive distribution of the classifier on $\mathcal{D}_l$ is only optimized to follow the prior distribution $p(\mathbf{y}_t|\mathbf{y}_s, \mathcal{G})$ in Eq.(6). And the provided labels of inexact supervision classes from the $\mathcal{D}_l$ do not contribute to model $q_C(\mathbf{y}_s, \mathbf{y}_t|\mathbf{x}, \mathcal{G})$. This is undesirable because the distribution is used to get the label vectors of the input to generate or reconstruct the data. Thus, to better model the label predictive distribution and provide more reliable supervision for encoder and decoder, we add the loss function that explicitly utilizes the data with labels in inexact supervision classes:

$$\mathcal{L}_C^s = \mathbb{E}_{p_l(\mathbf{x}, \mathbf{y}_s)}[-\log q_C(\mathbf{y}_s|\mathbf{x}, \mathcal{G})]. \tag{10}$$

4.1.3 Final Objective Function. Combining the variational lower bound of the generative model, the constraint to enforce the disentangled representation learning and the additional classification loss to better model the classifier, the final objective function is:

$$\min_{\phi_E, \phi_C, \phi_D} \mathcal{L}_G + \alpha \mathcal{L}_{cons} + \beta \mathcal{L}_C^s, \tag{11}$$

where α and β are hyperparameters. And ϕ_E, ϕ_C, and ϕ_D denote the learnable parameters of the encoder, classifier, and decoder.

4.2 Deep Learning Framework of ADDES

With the generative framework given above, we introduce the details of modeling the encoder $q_E(\mathbf{z}|\mathbf{x})$, the decoder $p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})$, and the classifier $q_C(\mathbf{y}_t, \mathbf{y}_s|x, \mathcal{G})$ now.

4.2.1 Encoder and Decoder. For many applications such as images and text, both $q_E(\mathbf{z}|\mathbf{x})$ and $p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})$ could be very complex distributions. Following VAE [18], we use neural network and reparameterization trick to model $q_E(\mathbf{z}|\mathbf{x})$ and $p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})$, which are shown to be able to approximate complex distributions under mild conditions. Specifically, we assume the encoder $q_E(\mathbf{z}|\mathbf{x})$ follows Gaussian distribution with the mean and variance as the output of a neural network:

$$q_E(\mathbf{z}|\mathbf{x}) = N(z; \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2\mathbf{I}), \quad \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z = E(\mathbf{x}) \tag{12}$$

where $E(\cdot)$ is the neural network which takes $\mathbf{x}$ as input and output the mean $\boldsymbol{\mu}_z$ and standard deviation $\boldsymbol{\sigma}_z$. Then $\mathbf{z}$ can be sampled as $\mathbf{z} = \boldsymbol{\mu}_z + \boldsymbol{\sigma}_z \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is sampled from a normal distribution. Similarly, we assume the decoder $p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})$ follows Gaussian distribution with the mean and variance as the output of a deep neural network:

$$p_D(x|\mathbf{y}_s, \mathbf{y}_t, z) = N(x; \boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2\mathbf{I}), \quad \boldsymbol{\mu}_x, \boldsymbol{\sigma}_x = D(\mathbf{y}_s, \mathbf{y}_t, \mathbf{z}) \tag{13}$$

where $D(\cdot)$ is the neural network which takes $(\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})$ as input and output the mean $\boldsymbol{\mu}_x$ and standard deviation $\boldsymbol{\sigma}_x$. The structure of the $E(\cdot)$ and $D(\cdot)$ can be chosen based on the domain we are working on. For example, for image datasets, deep convolutional neutral networks could be applied. For text datasets, sequence to sequence models are good candidates.

4.2.2 *GCN-based Classifier.*

Since only the inexact supervision is available, we rely on the dependency between the labels of inexact supervision classes and target classes to infer the target labels, and the dependency is encoded in the graph $\mathcal{G}$. Graph neural networks have been demonstrated to be very effective in capturing the relationship between nodes in a graph. Therefore, to model $q_C(\mathbf{y}_t, \mathbf{y}_s|\mathbf{x}, \mathcal{G})$, we adopt Graph Convolutional Networks (GCN) for $\mathcal{G}$ and propose a GCN-based classifier. Figure 2 gives an illustration of the GCN-based classifier, which consists of two parts, i.e., a feature extractor and a GCN module. The basic idea is to learn representations of classes from $\mathcal{G}$ using GCN and the features of $\mathbf{x}$ using the feature extractor, then conduct label prediction based on these representations.

Feature extraction: To facilitate the classification by the GCN module, a low dimension representation of $\mathbf{x}$ is required. One way is to use the latent feature $\mathbf{z}$ from the encoder $E(\mathbf{x})$. However, since the encoder is expected to learn $\mathbf{z}$ that has no semantic information about the labels, directly using $\mathbf{z}$ cannot help predict labels. Thus, another feature extractor is required. The model architecture is quite flexible. For images, a CNN model such as AlexNet [19], VGG [33] and ResNet [11] can be feature extractor. For text, LSTM [12], GRU [5], CNN [16] and transformer [35] are all potential models. With the feature extractor model f_m, we could attain the representation of input $\mathbf{x}$ as

$$\mathbf{h}_f = f_m(\mathbf{x}) \in \mathbb{R}^F, \tag{14}$$

where F denotes the dimension of the extracted feature.

Synthesize classifiers with GCN: The GCN is to generate the parameters of classifiers for both inexact supervision classes and target classes. Each node of the graph corresponds to a class in the whole class set $\mathcal{W}$. Thus, the number of nodes is $n = |\mathcal{W}|$. The adjacency matrix of the graph is $\mathbf{A} \in \mathbb{R}^{n \times n}$. And $\mathbf{A}_{ij}$ indicates the how strong the correlation between the labels of classes l_i and l_j is. The GCN-layer updates the node features $\mathbf{H} \in \mathbb{R}^{n \times d}$ by aggregating the information from the neighbors, where d represents the dimension of the node features. The process can be written as:

$$\mathbf{H}^{l+1} = f(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{H}^l\mathbf{W}^l), \tag{15}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. f denotes the nonlinear active function. $\mathbf{W}^l \in \mathbb{R}^{d^l \times d^{l+1}}$ is the weights of the l-th layer, where d^l is the dimension of the latent feature in the l-th layer. The input of the first layer $\mathbf{V}$ could be word embeddings or one-hot embeddings of the classes. The output of the last GCN layer is $\mathbf{W} \in \mathbb{R}^{n \times F}$, which corresponds to classifier weights of the classes. The predicted scores of all the classes including inexact supervision classes and target classes could be obtained by:

$$\begin{bmatrix}\hat{\mathbf{y}}_s \\ \hat{\mathbf{y}}_t\end{bmatrix} = \sigma(\begin{bmatrix}\mathbf{W}_s \\ \mathbf{W}_t\end{bmatrix} \mathbf{h}_f), \tag{16}$$

where $\mathbf{W}_s$ and $\mathbf{W}_t$ indicate the synthesized classifier weights of the inexact supervision classes and target classes. $\hat{\mathbf{y}}_s \in \mathbb{R}^{|\mathcal{S}|}$ with the i-th element denoting the probability that the label of i-th inexact supervision class being 1. Similarly, $\hat{\mathbf{y}}_t \in \mathbb{R}^{|\mathcal{T}|}$ with the j-th element denoting the probability that the j-th target label being 1. With the parameter sharing and explicit utilization of graph structure, the inexact supervision could be propagated to the target classes to obtain reasonable classifiers.

Algorithm 1 Training Algorithm of ADDES.

```
Input: $\mathcal{D}_l = \{(\mathbf{x}^l, \mathbf{y}_s^l)\}$, $\mathcal{D}_u = \{\mathbf{x}^u\}$, $\mathbf{A}$, $\mathbf{V}$, $\alpha$ and $\beta$.
Output: $q_E(\mathbf{z}|\mathbf{x})$, $p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, \mathbf{z})$, and $q_C(\mathbf{y}_t, \mathbf{y}_s|\mathbf{x}, \mathcal{G})$
Initialize the GCN-based classifier by minimizing Eq.(10) and
  the third term of Eq.(6).
Initialize the encoder and decoder by minimizing Eq.(11) with
  the parameters of the classifier frozen.
repeat
    Randomly sample $\{\mathbf{x}_i^l\}_{i=1}^N$ from $\mathcal{D}_l$ and $\{\mathbf{x}_i^u\}_{i=1}^N$ from $\mathcal{D}_u$;
    Optimized the encoder parameters $\phi_E$, decoder parameters
      $\phi_D$ and classifier parameters $\phi_C$ by Eq.(11).
until convergence
return $q_E(\mathbf{z}|\mathbf{x})$, $p_D(\mathbf{x}|\mathbf{y}_s, \mathbf{y}_t, z)$, and $q_C(\mathbf{y}_t, \mathbf{y}_s|\mathbf{x}, \mathcal{G})$
```

4.3 Training Algorithm

The overall training algorithm of ADDES is given in Algorithm 1. Firstly, before jointly training the encoder, decoder, and classifier, these modules are separatly pretrained to have good initialization parameters. More specifically, the GCN classifier is prioritize to be optimized. Then with the classifier's parameters fixed, the encoder and decoder are pretrained with Eq.(11). Secondly, to make the gradients able to backpropagate from the decoder to the classifier, we directly input the soft labels to the decoder.

5 EXPERIMENTS

In this section, we conduct a series of experiments to validate the effectiveness of our proposed framework. They are designed to answer the following research questions:

- **RQ1** Could the proposed generative model synthesize useful labeled data as data augmentation for target label prediction?
- **RQ2** Could our proposed method bring benefits to different scenarios whose training data varies in types and sizes?
- **RQ3** Does the utilization of graph structure of labels promote the generative model learning? If it works, is it sensitive to the graph construction method?

5.1 Datasets

We conduct experiments on two publicly available datasets, including a text dataset StackOverflow and an image dataset MJSynth.

StackOverflow[1]: It contains texts of 10% of questions and answers from the Stack Overflow programming Q&A website. The majority of the questions have multiple tags. After filtering out rare tags, we obtain a tag set of size 25. Each question is labeled with 1.9 tags on average. To demonstrate ADDES could synthesize useful labeled data to facilitate the classification for various target classess, two sets from the 25 tags are set as target classes and sequently educe two datasets. Specifically, the target class sets are $\mathcal{T}_1 = \{javascript\}$ and $\mathcal{T}_2 = \{javascript, C\#, php, java\}$, which refer to as **StackOverflow-1** and **StackOverflow-2**, respectively. For both datasets, the size of $\mathcal{D}_l$ is 2k. The unlabeled set $\mathcal{D}_u$ contains 30k questions. For the test set, we randomly sample 30k questions.

MJSynth [14]: It is used for natural scene text recognition. Each image in MJSynth contains a word extracted from the text corpus.

[1] https://www.kaggle.com/stackoverflow/StackOverflow

The images are produced in a sophisticatedly designed pipeline to emulate the text in the natural scene. In the MJSynth dataset, the character label indicates if a certain letter is in the image or not. We filter out the images that contain characters other than the lower case alphabet, which makes the number of classes to 26. The number of character labels per image is 6 on average. Similar to StackOverflow, two target class sets, $\mathcal{T}_1 = \{e\}$ and $\mathcal{T}_2 = \{a, c, e\}$ are selected to build new datasets named as **MJSynth-1** and **MJSynth-2**. For both datasets, we sample 6k and 24k images as $\mathcal{D}_l$ and $\mathcal{D}_u$. Another 10k images are sampled as test sets.

5.2 Baselines

We compare our method with the following state-of-the-art baselines from the supervised classification, semi-supervised learning, multi-label learning and zero-shot learning:

- **text-CNN** [16]: Convolutional filters with different kernel sizes are applied to get the features for text classification.
- **GRU** [5]: It utilizes GRU cell to extract the text features.
- **SDANN** [14]: This is a network with five convolutional layers to process MJSynth for natural text recognition.
- **Semi-CNN** [9]: Semi-CNN utilizes unlabeled data $\mathcal{D}_u$ by adding an entropy regularization term to train the classifier.
- **ML-GCN** [4]: A state-of-the-art method applies the GCN to model the label relations for multi-label classification.
- **Zero-Shot** [40]: This is a state-of-the-art method for zero-shot learning. It transfers the knowledge learned from seen labels prediction by employing the GCN to predict the weights of classifiers for target label prediction. Thus, only the instance and labels of inexact supervision classes are required.

Aside from ML-GCN and Zero-Shot, these baselines assume labels of different classes are independent with each other. Moreover, all these methods require labels of target classes except Zero-Shot. Therefore, we develop ways to attain estimated target labels $\hat{\mathbf{y}}_t$ through inexact supervision labels $\mathbf{y}_s$ in the following subsections.

5.3 Experiments on the Text Datasets

5.3.1 Graph Construction.

5.3.1 Graph Construction. The edges between the inexact supervision classes $l_s \in \mathcal{S}$ are built based on the conditional probability $P(l_i|l_j)$, which is the probability of an instance belonging to class l_i when it is known to be in class l_j. According to [4], we count the occurrence of label pairs from $\mathcal{D}_l$ and obtain the matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$, where $\mathbf{M}_{ij}$ represents the count of instances labeled as 1 in both l_i and l_j. Then, the conditional probability is:

$$\mathbf{A}_{ij} = P(l_i|l_j) = \frac{\mathbf{M}_{ij}}{N_j}, \quad l_i, l_j \in \mathcal{S} \tag{17}$$

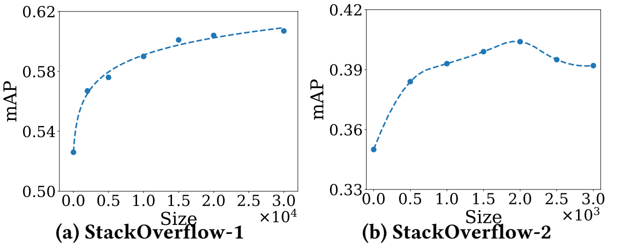
where N_j denotes the count of instances which belong to l_j in the dataset. With Eq. 17, the weights of the edges linking the inexact supervision classes could be obtained. However, due to the lacking of annotations of target labels in the training set, we are unable to build the edges between the target classes and inexact supervision classes by Eq.(17). Introducing the prior knowledge to the graph construction could solve this problem. For instance, based on the primary programming knowledge, we could add undirected edges between the target class `javascript` and inexact supervision class `ajax`. With the t-th target class denoted as l_t and its manually assigned related class set denoted as $\mathcal{R}_t$, the process of linking the target and inexact supervision classes can be formulated as:

$$\mathbf{A}_{ts} = \mathbf{A}_{st} = \begin{cases} 1 & \text{, if } l_s \in \mathcal{R}_t \\ 0 & \text{, else} \end{cases}, \tag{18}$$

where $l_s \in \mathcal{S}$ is the s-th inexact supervision class. The constructed graph could also be used to estimate the conditional probability $p(\mathbf{y}_t|\mathbf{y}_s, \mathcal{G})$ which is required in Eq.(6). The estimation formula is:

$$P(\mathbf{y}_t(i) = 1|\mathbf{y}_s, \mathcal{G}) = \begin{cases} 1 & \text{, if } \exists l_j \in \mathcal{S}, \mathbf{y}_s(j) = 1 \cap \mathbf{A}_{ij} = 1 \\ 0 & \text{, else} \end{cases}, \tag{19}$$

where $\mathbf{y}_t(i)$ denotes the label of the target class $l_i \in \mathcal{T}$, and $\mathbf{y}_s(j)$ means the label of inexact supervision class $l_j \in \mathcal{S}$.



(a) StackOverflow-1 **(b) StackOverflow-2**

Figure 3: The performance of the text-CNN w.r.t the numbers of synthetic data from ADDES added to the training set.

5.3.2 Implementation Details. The encoder of the ADDES for text generation is based on the bi-directional GRU with the hidden dimension set as 150. The mean and variance of the latent variable could be obtained from the hidden states of the GRU cell. For decoder, we adopt a global attention mechanism [22] to facilitate focusing on critical parts of the input sequence. Similarly, bi-directional GRU is applied to extract features for the GCN-based classifier. The GCN module of the classifier has two layers. One-hot embeddings are used as node attributes. The hyperparameters of ADDES are: $\alpha = 0.1, \beta = 0.1$.

5.3.3 Experimental Results. To answer **RQ1**, we synthesize a set of labeled data $\mathcal{D}_s = \{(x^{syn_i}, \mathbf{y}_s^{syn_i}, \mathbf{y}_t^{syn_i})\}_{i=1}^{n_s}$ for data augmentation, where n_s is the size of synthetic dataset. We also supplement the instance containing inexact supervision, i.e., $(x, \mathbf{y}_s) \in \mathcal{D}_l$ with estimated target label $\hat{\mathbf{y}}_t \in \{0, 1\}^{|\mathcal{T}|}$ through Eq.(19) to build estimated labeled dataset $\mathcal{D}_e$. Then the performance of the classifiers trained with $\mathcal{D}_s \cup \mathcal{D}_e$ could show whether the synthetic data could bring benefit to the classifiers for target label prediction. The performance of the models is evaluated by two metrics: mean average precision (mAP) and the average area under ROC curve (AUC).

Impacts of the size of $\mathcal{D}_s$: It has been reported that the number of synthetic data added into the training set could strongly affect the performance of supervised learning [31]. Therefore, we investigate the performance of the classifier whose training set is enlarged with the synthetic dataset $\mathcal{D}_s$ in different sizes. Here, different numbers of synthetic data mixed with 2k estimated labeled data are applied to train the text-CNN model. The results are shown in Figure 3. From Figure 3a, it is observable that the performance improves up to saturation as we add more synthetic data to the training set in StackOverflow-1. As results of StackOverflow-2 shown in Figure 3b, we could find the gain brought by synthetic labeled data $\mathcal{D}_s$

Table 2: The comparisons with the baselines and models trained with text dataset augmented with synthetic data.

Datasets	Metric	text-CNN	GRU	ML-GCN	Semi-CNN	Zero-shot	AugCNN	AugGCN
StackOverflow-1	mAP	0.528 ±0.004	0.516 ±0.007	0.565 ±0.008	0.549 ±0.004	0.541 ±0.003	0.609 ±0.003	**0.629 ±0.003**
	AUC	0.773 ±0.004	0.763 ±0.004	0.788 ±0.003	0.782 ±0.002	0.781 ±0.002	0.830 ±0.001	**0.832 ±0.003**
StackOverflow-2	mAP	0.350 ±0.003	0.342 ±0.008	0.379 ±0.005	0.376 ±0.003	0.369 ±0.002	0.400 ±0.003	**0.412 ±0.004**
	AUC	0.666 ±0.003	0.646 ±0.006	0.685 ±0.003	0.663 ±0.002	0.659 ±0.005	0.686 ±0.003	**0.703 ±0.003**

Table 3: The synthetic labeled data for StackOveflow-1.

Input Labels	Generated Text
javascript	Angular 2 Routing in plain Javascript
javascript	Different CSS depending on month and year
javascript, jquery	JQuery replace on Click in h
C++	C++ : find in set of pointers
java, json	query elasticsearch with java with JSON
C++, Android	Display image created by OpenCV on Android

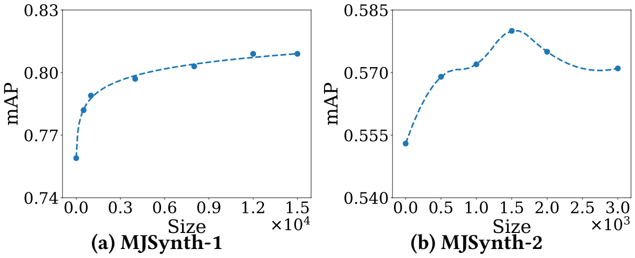
will firstly increase and then decrease as the size of $\mathcal{D}_s$ increases. This is because there are more target classes in StackOverflow-2, which makes it more challenging to generate high-quality labeled data. For both datasets, compared with the models trained without synthetic data, i.e., $|\mathcal{D}_s| = 0$, the models trained with augmented data consistently perform better. Therefore, the generated labeled text is useful as data augmentation for target label prediction.

Comparisons with baselines: We evaluate the benefits brought by synthetic data to different models , i.e., text-CNN and ML-GCN, and compare them with the baselines. More specifically, models with the same structure as text-CNN and ML-GCN are trained by adding an optimal size of $\mathcal{D}_s$ to the original estimated labeled dataset $\mathcal{D}_e$, which refer to as **AugCNN** and **AugGCN**. According to Figure 3, the size of $\mathcal{D}_s$ is set as 30k and 2k in StackOverflow-1 and StackOverflow-2, respectively. The average results and standard deviation of five runs are presented in Table 2. We could have the following observations: (i) AugCNN is better than the state-of-the-art multi-label classification method ML-GCN and the semi-supervised learning approach Semi-CNN; (ii) AugCNN and AugGCN outperform text-CNN and ML-GCN and other baselines with a large margin, which indicates the synthetic data could be helpful for various models. These observations confirm that the proposed model could generate high-quality labeled data when only inexact supervision is available in text datasets.

Visualization of the synthetic text: Some samples of generated labeled data for StackOverflow-1 whose target class set is {*javascript*} are reported in Table 3. It shows that we could produce realistic text which contains the target label `javascript`. Furthermore, the model could also generate the text with multiple labels.

5.4 Experiments on the Image Datasets

5.4.1 Graph Construction. We aim to assign links between the labels that often occur together. As each image in the datasets contains a word, the co-occurrence probability of labels should be the same as the probability that two letters appear together in a word. Therefore, we use a common word corpus YAWL[2] to calculate the correlation matrix by Eq.(17) Moreover, the estimation of the conditional probability $p(\mathbf{y}_t|\mathbf{y}_s, \mathcal{G})$ could be attained through the YAML corpus. Specially, We train a random forest model on YAML to build the estimated labeled dataset $\mathcal{D}_e$. It predicts the target label vector $\mathbf{y}_t$ based on the inexact supervision label vector $\hat{\mathbf{y}}_s$ of $\mathcal{D}_l$.



(a) MJSynth-1 **(b) MJSynth-2**

Figure 4: The performance of the SDANN w.r.t the numbers of synthetic data from ADDES added to the training set.

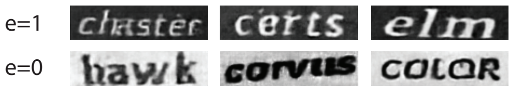


Figure 5: The synthetic images of ADDES on MJSynth-1

5.4.2 Implementation Details. The encoder is composed of 5 convolutional layers which contain 64, 128, 256, 512, and 512 filters with the kernel size and stride set as 4 and 2. The mean and variance of the latent variable are obtained from the output of the global max pooling layer of the encoder. The structure of the decoder is symmetrical to the encoder. Transpose convolution with stride 2 is used to upsample the feature map in the decoder. For the GCN-based classifier, the feature extractor has the same structure as the encoder. And there is one hidden layer with the filer size set as 128 in the classifier. The hyperparameters are set as: $\alpha = 100, \beta = 0.1$.

5.4.3 Experimental Results. The proposed framework could also generate useful labeled data for data augmentation in image classification. Similar to the text datasets, we investigate impacts of size of the synthetic labeled data $\mathcal{D}_s$ to the models utilizing synthetic data. Then, we compare the performance of the baselines and the models training with augmented data to demonstrate the generated labeled data could facilitate the learning of image classifier.

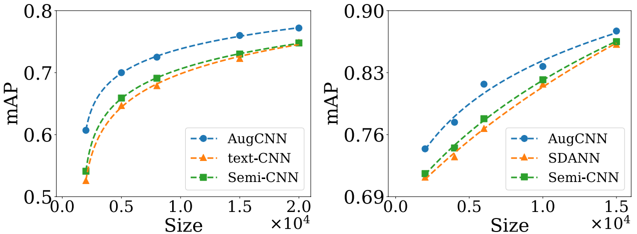
Impacts of size of $\mathcal{D}_s$: The Figure 4 shows the trend of the performance of the SDANN with the increase of augmented images. We could observe that the synthetic data could improve the performance of the model for both image datasets. The trend of the curves is in line with the results of text datasets. The evident improvements after introducing a reasonable number of synthetic images demonstrate the validity of the generated labeled data.

Comparisons with baselines: We train SDANN and ML-GCN with the augmented dataset $\mathcal{D}_e \cup \mathcal{D}_s$ to get **AugCNN** and **AugGCN** and find significant improvements compared with the baselines. From Table 4, we could have similar observations in image datasets: (i) AugCNN outperforms SDANN and Semi-CNN with a large margin and even performs better than the sophisticatedly designed

[2] http://freshmeat.sourceforge.net/projects/yawl/

Table 4: The results of the baselines and the classifiers trained with augmented data on the image datasets.

Dataset	Metric	SDANN	ML-GCN	Semi-CNN	Zero-shot	AugCNN	AugGCN
MJSynth-1	mAP	0.767 ±0.004	0.788 ±0.005	0.776 ±0.003	0.700 ±0.001	0.810 ±0.004	**0.816 ±0.002**
	AUC	0.640 ±0.005	0.665 ±0.006	0.650 ±0.002	0.556 ±0.003	0.696 ±0.005	**0.701 ±0.003**
MJSynth-2	mAP	0.555 ±0.005	0.572 ±0.005	0.562 ±0.001	0.539 ±0.006	0.584 ±0.005	**0.606 ±0.004**
	AUC	0.584 ±0.004	0.606 ±0.007	0.593 ±0.002	0.567 ±0.005	0.620 ±0.004	**0.640 ±0.005**



(a) mAP on StackOverflow-1 (b) mAP on MJSynth-1

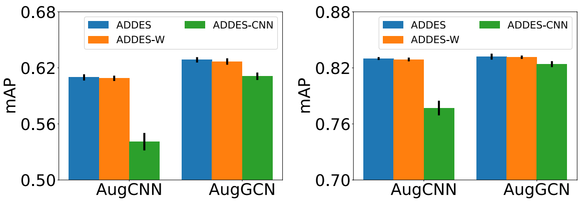
Figure 6: Impacts of the size of $\mathcal{D}_l$ to our proposed model.

model ML-GCN, which shows our generative model utilizes unlabeled data and label relations well; (ii) AugGCN achieves the best results among all the classifiers. It indicates benefits of the synthetic labeled images are beyond the simple SDANN. These observations demonstrate the effectiveness of ADDES in labeled images generation with inexact supervision.

Visualization: Samples of synthetic data on MJSynth-1 are presented in Figure 5. The target label set is $\{e\}$. The first row are samples generated with $\{e\}$ set to 1 and the second row are samples with $\{e\}$ set to 0. We could observe that ADDES generate realistic data to facilitate the training of classifiers for target label prediction.

5.5 Impacts of the Size of $\mathcal{D}_l$

Prior studies have shown that some semi-supervised learning methods and data augmentation methods are sensitive to the size of labeled data [26, 44]. To demonstrate our proposed model could synthesize useful labeled data to facilitate the targets label prediction regardless of the size of $\mathcal{D}_l$, we conduct experiments on StackOverflow-1 and MJSynth-1 to answer **RQ2**. We select the sizes of $\mathcal{D}_l$ ranging from 2k to 20k. The results are shown in Figure 6. To make fair comparisons, we compare text-CNN/SDANN, Semi-CNN, and AugCNN, which have the same network structure. From Figure 6, we could observe that in both text and image datasets, Semi-CNN makes negligible improvements compared with text-CNN. On the contrary, with the synthetic labeled data included in the training set, AugCNN consistently outperforms the other baselines with a clear margin regardless of the size of $\mathcal{D}_l$. It shows that our proposed method could benefit the scenarios varying in data types and sizes.



(a) mAP (b) AUC

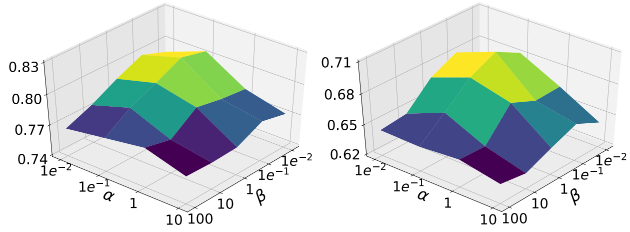
Figure 7: Comparisons with ADDES and its variants.

5.6 Ablation Study

To answer **RQ3**, we conduct ablation studies on StackOverflow-1 to investigate the importance of the GCN-based classifier and its sensitivity to the graph construction methods. Specifically, we compare our model with the following variants of ADDES:

- **ADDES-CNN**: It replaces the GCN-based classifier in ADDES with a multi-label classifier to obtain $q_C(\mathbf{y}_t, \mathbf{y}_s|\mathbf{x})$. It treats the prediction of multiple labels as isolation tasks.
- **ADDES-W**: ADDES-W builds weighted graph thorough the ground truth of training data in whole class set through Eq.(17) for the GCN-based classifier. The weights of edges between classes indicate their co-occurrence rates.

The performance of the AugCNN and AugGCN which utilize 30k synthetic data from ADDES and its variants is presented in Figure 7. As we can see, if we eliminate the GCN module, the gain brought by the synthetic data will significantly decrease ($p < 0.005$, t-test). However, we could find that the synthetic data of ADDES and ADDES-W shows no significant difference for data augmentation. From these observations, we could conclude that (1) the information aggregation from the inexact supervision to target labels contributes to better generative model for inexact supervision; (2) The graph utilizing prior knowledge to obtain binary weights between target classes and inexpensive supervision classes shows no difference with the graph completely built by the labels co-occurrence probability in modeling the data with inexact supervision.



(a) mAP of AugCNN (b) AUC of AugCNN

Figure 8: Parameter sensitivity analysis.

5.7 Parameter Sensitivity

The proposed framework includes two important hyperparameters, i.e., α controlling the contribution of the constraint for disentangled representation learning, β controlling the contribution of the inexact supervision labels to model the classifier in ADDES. We investigate the impacts of these two parameters on target label prediction on MJSynth-1 with the number of synthetic data set as 30k. We vary α as $\{0.01, 0.1, 1, 10\}$ and β as $\{0.01, 0.1, 1, 10, 100\}$. Then, We obtain AugCNN models with the synthetic data from the generative models. The results are presented in Fig. 8. With the increase of α, the performance first increase then decrease. The same trend also exhibits in β. And when both α and β ranges from 0.01 to 1, the generative model shows consistently good performance.

6 CONCLUSION AND FUTURE WORK

In this paper, we investigate a novel problem of labeled data generation with inexact supervision. It is a potential direction to cope with the deficiency of labeled data for deep learning. To deal with this problem, we propose a novel generative framework ADDES to generate data labeled in both target and inexact supervision class set. Extensive experimental results on image and text datasets demonstrated the effectiveness of the ADDES in synthesizing high-quality labeled data for the target label prediction. Further experiments are conducted to understand the contributions of each component of ADDES and its parameter sensitivity. There are several interesting directions which need further investigation. First, in this paper, we assume the inexact supervision labels are clean. However, the labels could be noisy as they are crawled from social media. Thus, one direction is to investigate labeled data generation with inexact and inaccurate supervision. Second, there are many different ways in constructing the graph. We would like to study automatic methods to construct the graph linking the related labels.

7 ACKNOWLEDGEMENTS

This material is based upon work supported by, or in part by, the National Science Foundation (NSF) under grant #IIS-1909702, #IIS1955851. The findings and conclusions in this paper do not necessarily reflect the view of the funding agency.

REFERENCES

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).

[2] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. 2004. Learning multi-label scene classification. *Pattern recognition* 37, 9 (2004), 1757–1771.

[3] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* (2015).

[4] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. 2019. Multi-Label Image Recognition with Graph Convolutional Networks. In *CVPR*. 5177–5186.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[6] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NeurIPS*. 2121–2129.

[7] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*. 1243–1252.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NeurIPS*. 2672–2680.

[9] Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *NeurIPS*. 529–536.

[10] Jiangfan Han, Ping Luo, and Xiaogang Wang. 2019. Deep self-learning from noisy labels. In *CVPR*. 5138–5147.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

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

[13] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*. JMLR. org, 1587–1596.

[14] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. In *Workshop on Deep Learning, NIPS*.

[15] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P. Xing. 2019. Rethinking Knowledge Graph Propagation for Zero-Shot Learning. In *CVPR*.

[16] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[17] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *NeurIPS*. 3581–3589.

[18] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. arXiv:stat.ML/1312.6114

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*. 1097–1105.

[20] Vinay Kumar Verma, Gundeep Arora, Ashish Mishra, and Piyush Rai. 2018. Generalized Zero-Shot Learning via Synthesized Examples. In *CVPR*.

[21] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. [n.d.]. Multi-label Zero-Shot Learning with Structured Knowledge Graphs. In *CVPR* (2018-06). IEEE, 1576–1585. https://doi.org/10.1109/CVPR.2018.00170

[22] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[23] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[24] Ashish Mishra, Shiva Krishna Reddy, Anurag Mittal, and Hema A. Murthy. 2018. A Generative Model for Zero Shot Learning Using Conditional Variational Autoencoders. In *CVPR Workshops*.

[25] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier gans. In *ICML*. JMLR. org, 2642–2651.

[26] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. 2018. Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*. 3235–3246.

[27] Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Dsgan: Generative adversarial training for distant supervision relation extraction. *arXiv preprint arXiv:1805.09929* (2018).

[28] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *VLDB*, Vol. 11. 269.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*. 91–99.

[30] Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *ICML*. 2152–2161.

[31] Hoo-Chang Shin, Neil A Tenenholtz, Jameson K Rogers, Christopher G Schwarz, Matthew L Senjem, Jeffrey L Gunter, Katherine P Andriole, and Mark Michalski. [n.d.]. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *International workshop on simulation and synthesis in medical imaging*. 1–11.

[32] Kai Shu, Suhang Wang, Thai Le, Dongwon Lee, and Huan Liu. 2018. Deep headline generation for clickbait detection. In *ICDM*. IEEE, 467–476.

[33] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[34] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NeurIPS*. 3483–3491.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*. 5998–6008.

[36] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. 2017. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*. 839–847.

[37] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. 2016. Cnn-rnn: A unified framework for multi-label image classification. In *CVPR*. 2285–2294.

[38] Wentao Wang, Tyler Derr, Yao Ma, Suhang Wang, Hui Liu, Zitao Liu, and Jiliang Tang. 2020. Learning from Incomplete Labeled Data via Adversarial Data Generation. In *ICDM*. IEEE, 1316–1321.

[39] Wentao Wang, Suhang Wang, Wenqi Fan, Zitao Liu, and Jiliang Tang. 2020. Global-and-Local Aware Data Generation for the Class Imbalance Problem. In *SDM*. SIAM, 307–315.

[40] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. 2018. Zero-Shot Recognition via Semantic Embeddings and Knowledge Graphs. In *CVPR*.

[41] Yilin Wang, Suhang Wang, Guojun Qi, Jiliang Tang, and Baoxin Li. 2018. Weakly supervised facial attribute manipulation via deep adversarial network. In *WACV*. IEEE, 112–121.

[42] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. 2019. f-VAEGAN-D2: A feature generating framework for any-shot learning. In *CVPR*. 10275–10284.

[43] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *CVPR*. 2691–2699.

[44] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. (2019).

[45] Li Zhang, Tao Xiang, and Shaogang Gong. 2017. Learning a deep embedding model for zero-shot learning. In *CVPR*. 2021–2030.

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