Semi-Supervised Seq2seq Joint-Stochastic-Approximation Autoencoders With Applications to Semantic Parsing

Yunfu Song and Zhijian Ou, *Senior Member, IEEE*

***Abstract*—Developing Semi-Supervised Seq2Seq (S^4) learning for sequence transduction tasks in natural language processing (NLP), e.g. semantic parsing, is challenging, since both the input and the output sequences are discrete. This discrete nature makes trouble for methods which need gradients either from the input space or from the output space. Recently, a new learning method called joint stochastic approximation is developed for unsupervised learning of fixed-dimensional autoencoders and theoretically avoids gradient propagation through discrete latent variables, which is suffered by Variational Auto-Encoders (VAEs). In this letter, we propose seq2seq Joint-stochastic-approximation Auto-Encoders (JAEs) and apply them to S^4 learning for NLP sequence transduction tasks. Further, we propose bi-directional JAEs (called bi-JAEs) to leverage not only unpaired input sequences (which is most commonly studied) but also unpaired output sequences. Experiments on two benchmarking datasets for semantic parsing show that JAEs consistently outperform VAEs in S^4 learning and bi-JAEs yield further improvements.**

***Index Terms*—Semi-supervised learning, seq2seq, semantic parsing, joint stochastic approximation, variational auto-encoder.**

I. Introduction

RECURRENT neural network (RNN) based sequence-to-sequence (seq2seq) learning [1], aiming to mapping input sequences to output sequences, has achieved impressed performances for sequence transduction tasks in natural language processing (NLP), such as semantic parsing [2] and machine translation [3]. Particularly, semantic parsing is to map natural language (NL) utterances to formal meaning representations (MRs). By linearization, we can use seq2seq learning to handle tree-structured MRs, e.g. λ-calculus logical forms [4], Python code [5]. However, current seq2seq learning depends on supervised learning and requires an abundance of supervised data, e.g. pairs of NLs and MRs for semantic parsing, which are time-consuming and costly to collect in practice.

Manuscript received August 19, 2019; revised November 9, 2019; accepted November 12, 2019. Date of publication November 18, 2019; date of current version January 22, 2020. This work was supported by National Natural Science Foundation of China under Grant 61976122, with code released at https://github.com/thu-spmi/seq2seq-JAE. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sandro Cumani. *(Corresponding author: Zhijian Ou.)*

The authors are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China, and also with Beijing National Research Center for Information Science and Technology, Beijing 100084, China (e-mail: 769414284@qq.com; ozj@tsinghua.edu.cn).

Digital Object Identifier 10.1109/LSP.2019.2953999

There is increasing interest in developing Semi-Supervised Seq2Seq (S^4) learning to map input sequence x to output sequence z, where supervised data in the form of (x, z) pairs are limited, but there are easily-available unsupervised data. Unsupervised data in the form of unpaired input sequences are the most commonly studied form in S^4 learning, e.g. unlabeled NL utterances in semantic parsing, un-translated source language sentences in machine translation.

Developing S^4 learning methods for NLP sequence transduction tasks is challenging, since both the input and the output sequences are discrete. This discrete nature makes trouble for a class of semi-supervised learning (SSL) methods [6], [7], which use gradients with respect to x from the input space and is mainly applied for classification of continuous observations, e.g. images. Variational Auto-Encoders (VAEs) [8] represent another class of generative model based SSL methods that have been applied for S^4 learning in NLP sequence transduction tasks [9]–[11]. Roughly speaking, a VAE learns an encoder (also termed as inference model), $q_\phi(z|x)$, and a decoder (generative model), $p_\theta(x|z)$, via the variational principle. VAEs do not need gradients from the input space. But the Monte-Carlo gradient estimator for the encoder parameter ϕ is known to have high variance. For continuous z, we can utilize the reparameterization trick [8] to use gradients with respect to z from the output space to reduce variance. When z is discrete, like in S^4 learning for NLP sequence transduction tasks, reparameterization is infeasible. Alternatively, the REINFORCE trick can be employed with ad-hoc variance-reduction techniques [9], [11]. A few studies utilize the Concrete [12] or Gumbel-softmax [13] distribution as a continuous approximation to a multinomial distribution, and then use reparameterization, although the gradients of these relaxations are biased [14].

Recently, a new learning method called joint stochastic approximation [15] is developed for learning autoencoders (or called Helmholtz machines as in [15]) and theoretically avoids gradient propagation through discrete latent variables. We call the learning principle together with the model by Joint-stochastic-approximation Auto-Encoders or JAEs for short.

In this letter, we propose seq2seq JAEs and apply them to S^4 learning for NLP sequence transduction tasks. This is more challenging than the work in [15], where both the generative and inference models are sigmoid belief networks (SBNs) and applied in generative modeling of fixed-dimensional images. In contrast, in seq2seq JAEs, the encoder itself is implemented as a seq2seq model and the decoder as another seq2seq model, similar to

seq2seq VAEs[1] in [10], [11]. This letter presents a systematic progress from ordinary fixed-dimensional JAEs to seq2seq JAEs and from unsupervised learning to semi-supervised learning. This is the first contribution of this letter.

Furthermore, in addition to unpaired input sequences (briefly called unpaired x-data), we also have easily-available unpaired output sequences (called unpaired z-data) in some tasks. For example, in parsing NL into Python code, unpaired z can be easily collected from the Python DJANGO library [5]. Given the capability of JAEs in learning with discrete latent variables, we propose bi-directional JAEs (called bi-JAEs) to leverage not only unpaired x-data but also unpaired z-data. This is the second contribution of this letter. These two unsupervised learning processes in bi-JAEs help each other. For instance in semantic parsing, learning to read meaning from text and learning to write meaning to text promote each other. This is similar to dual learning in [16] which is, however, implemented via the reinforcement learning principle. We conduct evaluations of VAEs and JAEs in S^4 learning for the task of semantic parsing over two benchmarking datasets (ATIS and DJANGO [5]). Experimental results show that JAEs consistently outperform VAEs in S^4 learning and bi-JAEs yield further improvements.

II. Method

A. Joint-Stochastic-Approximation Auto-Encoders (JAEs)

For observation x and latent variable z, consider a generative model $p_\theta(z,x) \triangleq p(z)p_\theta(x|z)$ paired with an auxiliary inference model $q_\phi(z|x)$. The key idea of Joint Stochastic Approximation (JSA) learning proposed in [15] is that in addition to maximizing with respect to (w.r.t.) θ the marginal log-likelihood, it simultaneously minimizes w.r.t. ϕ the KL divergence[2] $KL(p_\theta(z|x)||q_\phi(z|x))$ between the posteriori and the inference model, by jointly optimizing:

$$\begin{cases} \min\limits_{\theta} KL\left[\tilde{p}(x)||p_\theta(x)\right] \\ \min\limits_{\phi} KL\left[\tilde{p}(x)p_\theta(z|x)||\tilde{p}(x)q_\phi(z|x)\right] \end{cases} \tag{1}$$

where $\tilde{p}(x) \triangleq \frac{1}{m}\sum_{i=1}^{m}\delta(x-x_i)$ denotes the empirical distribution for a training dataset consisting of m observations. With the gradients being set to zeros, the optimization problem Eq. (1) can be solved by applying the stochastic approximation (SA) algorithm [17] to find the root for the following system of simultaneous equations:

$$\begin{cases} E_{\tilde{p}(x)p_\theta(z|x)}\left[\dfrac{\partial}{\partial\theta}\log p_\theta(x,z)\right] = 0 \\ E_{\tilde{p}(x)p_\theta(z|x)}\left[\dfrac{\partial}{\partial\phi}\log q_\phi(z|x)\right] = 0 \end{cases} \tag{2}$$

Basically, SA provides a mathematical framework for stochastically solving a root finding problem, which has the form of expectations being equal to zeros. The SA algorithm iterates Monte Carlo sampling and parameter updating. In each iteration t of JSA, we draw a training observation $x \sim \tilde{p}(\cdot)$ and then draw a sample z through a Markov transition which admits $p_\theta(z|x)$ as the invariant distribution, based on Metropolis Independence Sampling (MIS):

(1) Propose $z \sim q_\phi(z|x)$,

(2) Accept $z^{(t)} = z$ with probability

$$\min\left\{1, \frac{w(z)}{w(z^{(t-1)})}\right\}, w(z) = \frac{p_\theta(z|x)}{q_\phi(z|x)} \propto \frac{p_\theta(z,x)}{q_\phi(z|x)}.$$

In practice, we usually draw multiple samples for z by performing the Markov transition repeatedly, which is known as SA with multiple moves [18]. With $p_\theta(x|z)$ termed the decoder and $q_\phi(z|x)$ the encoder, we call the JSA learning method together with the model (decoder and encoder) by Joint-stochastic-approximation Auto-Encoders or JAEs for short.[3]

B. Seq2seq JAEs for S^4 Learning

To extend JAEs to Semi-Supervised Seq2Seq (S^4) learning in NLP tasks such as semantic parsing, we define x as the NL utterance, and z the linearized sequence of the MR. The prior $p(z)$ is pretrained as a RNN language model over z; the decoder $p_\theta(x|z)$ is implemented as a seq2seq model and the encoder $q_\phi(z|x)$ as another seq2seq model.

Suppose that apart from limited parallel data $\mathbb{L} = \{(x_1,z_1),\ldots,(x_k,z_k)\}$, there are large amounts of unpaired NL utterances $\mathbb{U}_x = \{x_1,\ldots,x_m\}$, with $m \gg k$. $\tilde{p}(x) \triangleq \frac{1}{m}\sum_{i=1}^{m}\delta(x-x_i)$ denotes the empirical distribution of x. We formulate the semi-supervised learning from $\mathbb{L}$ and $\mathbb{U}_x$ as jointly optimizing

$$\begin{cases} \min\limits_{\theta} KL\left[\tilde{p}(x)\|p_\theta(x)\right] - \alpha\sum_{(x,z)\in\mathbb{L}}\log p_\theta(x|z) \\ \min\limits_{\phi} KL\left[\tilde{p}(x)p_\theta(z|x)\|\tilde{p}(x)q_\phi(z|x)\right] \\ \qquad -\alpha\sum_{(x,z)\in\mathbb{L}}\log q_\phi(z|x) \end{cases} \tag{3}$$

which are defined by a combination of unsupervised and supervised criteria, similar to [8], [19]. The hyper-parameter α controls the relative weight between unsupervised and supervised criteria. Similar to Eq. (2), by deriving the gradients w.r.t. θ and ϕ and setting them to zeros, we can apply the JSA learning algorithm, as shown in Algorithm 1 but only using $\mathbb{L}$ and $\mathbb{U}_x$.

C. Seq2seq bi-JAEs for S^4 Learning

In the above, we develop JAEs for the usual form of SSL, namely utilizing $\mathbb{U}_x$ in addition to $\mathbb{L}$. By swapping the roles of x and z, we can analogously develop JAEs for another form of SSL, which utilizes $\mathbb{U}_z$ in addition to $\mathbb{L}$. To differentiate, we call the usual form as forward SSL and the latter form as backward SSL. The forward SSL uses $p_\theta(z,x) \triangleq p(z)p_\theta(x|z)$ as the generative model with $q_\phi(z|x)$ as the auxiliary inference model to exploit $\mathbb{U}_x$. To optimize $\log p_\theta(x)$ for x in $\mathbb{U}_x$, we use MIS to obtain samples from the intractable posteriori $p_\theta(z|x)$ with $q_\phi(z|x)$ as the proposal. Analogously, the backward SSL uses $q_\phi(x,z) \triangleq q(x)q_\phi(z|x)$ as the generative model with $p_\theta(x|z)$ as the auxiliary inference model to exploit $\mathbb{U}_z$. $q(x)$ denotes the prior for x, which, like $p(z)$, is pretrained as a RNN language model. To optimize $\log q_\phi(z)$ for z in $\mathbb{U}_z$, we use MIS to obtain samples from the intractable posteriori $q_\phi(x|z)$ with $p_\theta(x|z)$ as the proposal.

Further, we can combine these two forms to obtain bi-JAEs. Suppose that apart from limited parallel data $\mathbb{L}$ and

[1] To be clear, the encoder and decoder in a seq2seq JAE and a seq2seq VAE may be of the same structure, but they are trained via the stochastic-approximation principle and the variational principle respectively.

[2] The KL-divergence between two distributions $p(\cdot)$ and $q(\cdot)$ is defined as $KL[p||q] \triangleq \int p\log(\frac{p}{q})$.

[3] JAEs only run a few steps of transitions (the same as the number of samples drawn in VAEs) per parameter update and still have parameter convergence [17]. Thus the training time complexity of JAEs is close to that of VAEs.

Algorithm 1: Semi-Supervised Learning with bi-JAEs from $\mathbb{L}$ (Parallel Data), $\mathbb{U}_x$ (unpaired x-Data), and $\mathbb{U}_z$ (Unpaired z-Data).

repeat

Monte Carlo sampling: Draw a forward (i.e. $x \to z$) unsupervised minibatch $\mathcal{U}_x \sim \tilde{p}(x)p_\theta(z|x)$ from $\mathbb{U}_x$, a backward (i.e. $z \to x$) unsupervised minibatch $\mathcal{U}_z \sim \tilde{p}(z)q_\phi(x|z)$ from $\mathbb{U}_z$, and a supervised minibatch $\mathcal{S}$ from $\mathbb{L}$;

SA updating:

Update θ by ascending: $\frac{1}{|\mathcal{U}_x|}\sum_{(x,z)\sim\mathcal{U}_x}\frac{\partial}{\partial\theta}\log p_\theta(x|z) + \frac{1}{|\mathcal{U}_z|}\sum_{(x,z)\sim\mathcal{U}_z}\frac{\partial}{\partial\theta}\log p_\theta(x|z) + \alpha\frac{1}{|\mathcal{S}|}\sum_{(x,z)\sim\mathcal{S}}\frac{\partial}{\partial\theta}\log p_\theta(x|z)$

Update ϕ by ascending: $\frac{1}{|\mathcal{U}_x|}\sum_{(x,z)\sim\mathcal{U}_x}\frac{\partial}{\partial\phi}\log q_\phi(z|x) + \frac{1}{|\mathcal{U}_z|}\sum_{(x,z)\sim\mathcal{U}_z}\frac{\partial}{\partial\phi}\log q_\phi(z|x) + \alpha\frac{1}{|\mathcal{S}|}\sum_{(x,z)\sim\mathcal{S}}\frac{\partial}{\partial\phi}\log q_\phi(z|x)$

until convergence.

TABLE I
EXAMPLES OF NLS x AND MRS z IN THE ATIS AND DJANGO DATASETS

Dataset	Example
ATIS	NL utterance: what airline is al0 fn0 MR: airline_name (the $0 (and (flight $0) (airline $0 al0) (flight_number $0 fn0)))
DJANGO	NL utterance: sort my_list in descending order MR: sorted(my_list, reverse=True)

TABLE II
MEAN ACCURACIES % FROM 6 INDEPENDENT RUNS ON ATIS AND DJANGO FOR DIFFERENT SIZES OF LABELED DATA $\mathbb{L}$. "SUP.": SUPERVISED. "VAE": STRUCT-VAE IN [11]

Dataset	$\lvert\mathbb{L}\rvert$	SUP.	VAE	JAE	bi-VAE	bi-JAE
ATIS	500	65.1	65.3	65.8^{V}	67.1	69.5^{VJb}
	1,000	75.2	74.7	76.6^{V}	77.1	78.7^{VJb}
	2,000	80.6	80.9	82.2^{V}	82.4	83.7^{VJb}
DJANGO	1,000	50.2	49.1	50.5^{V}	52.2	52.4^{VJ}
	2,000	56.3	56.5	56.7	57.6	58.6^{VJb}
	5,000	64.7	65.1	65.1	65.9	66.0^{VJ}

For comparison, we conduct McNemar's tests. JAE is compared against VAE; bi-JAE is against VAE, JAE, and bi-VAE. "V"/"J"/"b" represents p-value< 0.01 when compared against VAE/JAE/bi-VAE.

unpaired NL utterances $\mathbb{U}_x$, there are also unpaired MRs $\mathbb{U}_z = \{z_1, \ldots, z_n\}, n >> k$. $\tilde{p}(z) \triangleq \frac{1}{n}\sum_{j=1}^{n}\delta(z - z_j)$ denotes the empirical distribution of z. We propose bi-JAEs for semi-supervised learning from $\mathbb{L}$, $\mathbb{U}_x$ and $\mathbb{U}_z$ as jointly optimizing

$$\begin{cases} \min\limits_{\theta} KL\left[\tilde{p}(x)\|p_\theta(x)\right] - \alpha\sum_{(x,z)\in\mathbb{L}}\log p_\theta(x|z) \\ \quad + KL\left[\tilde{p}(z)q_\phi(x|z)\|\tilde{p}(z)p_\theta(x|z)\right] \\ \min\limits_{\phi} KL\left[\tilde{p}(x)p_\theta(z|x)\|\tilde{p}(x)q_\phi(z|x)\right] \\ \quad + KL\left[\tilde{p}(z)\|q_\phi(z)\right] - \alpha\sum_{(x,z)\in\mathbb{L}}\log q_\phi(z|x) \end{cases} \tag{4}$$

which again can be solved by applying the JSA learning algorithm, as shown in Algorithm 1.

III. EXPERIMENTS

We conduct a series of experiments to evaluate the proposed seq2seq JAEs and bi-JAEs for S^4 semantic parsing. To compare with the state-of-the-art VAE-based method [11], we experiment with the same datasets and the same model structure (encoder or parser $q_\phi(z|x)$ and decoder $p_\theta(x|z)$).

Data setting. Two semantic parsing datasets are used. First, the ATIS dataset consists of a collection of 5,410 telephone inquiries of flight booking, and the target MRs are defined using λ-calculus logical forms. Second, the DJANGO dataset [5], which contains 18,805 lines of Python source codes extracted from Django (a web application framework), is used for code generation, i.e. parsing NL utterances into Python statements. Examples of NL utterances x and MRs z in the ATIS and DJANGO datasets are listed in Table I.

All the data in the ATIS and DJANGO datasets are labeled. Thus as in [11], we experiment for the SSL scenario by randomly drawing k examples from the training set as the parallel data $\mathbb{L}$. For experiments with JAEs, the remaining x is used to construct $\mathbb{U}_x$. For experiments with bi-JAEs, the remaining x and z are used to construct $\mathbb{U}_x$ and $\mathbb{U}_z$ respectively, without using any pairing information.

Training setting. *1)* We pretrain and fix the prior $p(z)$ as an LSTM language model[4] from all available z as in [11], which may differ for different experiments. On ATIS dataset, we estimate $p(z)$ over $\mathbb{L}$ for JAEs and over all available z in $\mathbb{L} \cup \mathbb{U}_z$ for bi-JAEs. Since all source codes in the Django web application framework (within which the DJANGO dataset is only a subset) are available for utilization, $p(z)$ is estimated over all the Django codes, for both JAEs and bi-JAEs.

2) For bi-JAEs, we pretrain and fix the prior $q(x)$ as an LSTM language model.[5] On both ATIS and DJANGO dataset, we estimate $q(x)$ over all available x in $\mathbb{L} \cup \mathbb{U}_x$.

3) For model initialization, supervised pretraining of $q_\phi(z|x)$ and $p_\theta(x|z)$ are conducted, using the labeled data $\mathbb{L}$.

4) Throughout all experiments, we set $\alpha = 10$ in Eq.(3) and Eq.(4) without much tuning. During each SA iteration, MIS is repeated 5 times, both for $q_\phi(z|x)$ (given x from $\mathbb{U}_x$) and for $p_\theta(x|z)$ (given z from $\mathbb{U}_z$). We employ the Adam optimizer, and apply early stopping. As in [11], we reload the best model and halve the learning rate (initialized to 0.001) when the performance on the development set (evaluated per-epoch) does not increase within 5 epochs. This operation of model reloading and learning rate halving is repeated for 5 times.

A. Main Results

Table II list the results on ATIS and DJANGO datasets with varying amounts of labeled data $\mathbb{L}$. The evaluation metric is the exact-match **accuracy** [11]. To compare with the VAE-based method [11], we use the source code released by [11] and report our best reproduced results. We also implement bi-VAEs as a counterpart of bi-JAEs. The main observations are as follows:

1) By using additional unpaired data, JAEs and bi-JAEs significantly surpass the purely supervised baseline.
2) Under the same setting, JAEs outperform VAEs significantly and consistently on ATIS. On DJANGO, some mean accuracies are close, but the standard deviations (SDs) for VAEs are much larger than for JAEs. The SDs for VAEs and JAEs are 2.2/2.9/2.9 and 0.8/0.7/0.6

[4]This prior is needed to evaluate z-samples proposed by $q_\phi(z|x)$ in MIS.

[5]This prior is needed to evaluate x-samples proposed by $p_\theta(x|z)$ in MIS.

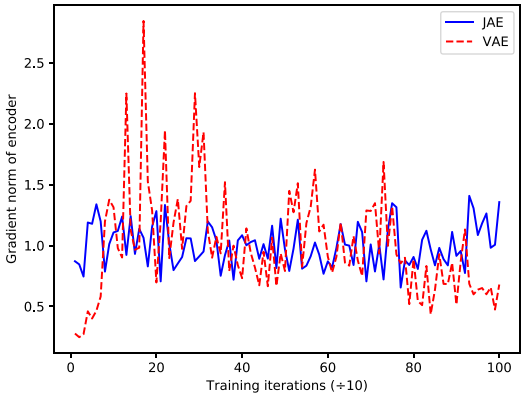

Fig. 1. Comparing the gradient norm w.r.t. ϕ between JAEs and VAEs on DJANGO ($|\mathbb{L}| = 5000$). We normalize the mean gradient norm of all iterations to 1, making the scale of the gradient norm between JAEs and VAEs comparable. Each value in the plot of the gradient norm is averaged over 10 iterations. We can see that the gradients of VAEs are more noisy than JAEs, due to the REINFORCE trick used in VAEs.

TABLE III
ACCURACIES % WITH AND W/O SAMPLE CACHING FOR JAES AND bi-JAES

Dataset	Cache?	JAE	bi-JAE
ATIS	×	76.5	78.1
($|\mathbb{L}| = 1,000$)	✓	76.6	78.7
DJANGO	×	64.8	65.8
($|\mathbb{L}| = 5,000$)	✓	65.1	66.0

respectively for $|\mathbb{L}| = 1000/2000/5000$. Additionally, we plot the gradient norms of the encoder parameter for VAEs and JAEs during training on DJANGO in Fig. 1. It is clear that the gradients during training with VAEs are more noisy, presumably due to the use of the REINFORCE trick. Overall, these results show that JAEs achieves better and stabler results than VAEs. By comparing bi-VAEs and bi-JAEs, we have similar observations.

3) By using additional unpaired z-data, bi-JAEs significantly outperform JAEs, over both ATIS and DJANGO. The gains are larger when $|\mathbb{L}|$ is small (limited availability of parallel data). It is interesting to see that bi-VAEs also outperform VAEs. By utilizing additional unpaired z-data, the proposed bi-directional learning clearly yield further improvements for S^4 learning.

B. Ablation Study

Chain persistence. Note that in MIS, we need to cache z (i.e. $z^{(t-1)}$) for each x in $\mathbb{U}_x$ to make a persistent Markov Chain, so that the invariant distribution of the Markov Chain will be $p_\theta(z|x)$. Caching z provides theoretical convergence guarantee of the JSA algorithm, but on the other hand, may bring some extra memory cost. So we evaluate the performance without caching, i.e. we substitute the cached z with the first sample proposed from $q_\phi(z|x)$. For bi-JAEs, we need to cache x for each z in $\mathbb{U}_z$; so in the no-caching experiment, we use the first sample proposed from $p_\theta(x|z)$. We experiment with semi-supervised learning on ATIS with $|\mathbb{L}| = 1,000$ and DJANGO with $|\mathbb{L}| = 5,000$. It can be seen from the results in Table III that the performances with caching are indeed better than with no-caching, though slightly. The result also suggests that training without caching could be used if there is a large amount of unlabeled data and the memory cost need be controlled.

TABLE IV
ACCURACIES % WITH DIFFERENT MIS SETTINGS FOR JAES AND bi-JAES

Dataset	Method	M=1	M=5, Last	M=5, All
ATIS	JAE	75.1	75.8	76.6
($|\mathbb{L}| = 1,000$)	bi-JAE	76.8	77.8	78.7
DJANGO	JAE	65.0	64.5	65.1
($|\mathbb{L}| = 5,000$)	bi-JAE	66.1	65.6	66.0

TABLE V
EXAMINING THE EFFECT OF ARTIFICIAL PAIRING BETWEEN $\mathbb{U}_x$ AND $\mathbb{U}_z$ FOR bi-JAES. $\mathbb{U}_1$ + $\mathbb{U}_1$ CONTAINS ARTIFICIAL PAIRING, WHILE $\mathbb{U}_1$ + $\mathbb{U}_2$ NOT

Dataset	$\mathbb{U}_1$ + $\mathbb{U}_1$	$\mathbb{U}_1$ + $\mathbb{U}_2$
ATIS ($|\mathbb{L}| = 1,000$)	77.8	77.4
DJANGO ($|\mathbb{L}| = 5,000$)	65.6	65.7

MIS setting. During each SA iteration, we use multiple moves along the MIS based Markov Chain to obtain samples from $p_\theta(z|x)$ (and from $q_\phi(x|z)$ in bi-JAEs). Denote the number of moves by M. In SA updating, we can use the last sample (denoted as "Last") or use all the samples from the multiple moves (denoted as "All"). Taking $M = 5$ as an example, for each unpaired x or z, we use one sample to calculate gradients for "$M = 5$, Last," and 5 samples for "$M = 5$, All". The results for different MIS settings are listed in Table IV. On ATIS, "$M = 5$, Last" is better than "$M = 1$," revealing that more moves in SA iterations could improve mixing of the Markov Chain. "$M = 5$, All" is better than "$M = 5$, Last," indicating that the gradients calculated by more samples are less noisy. On DJANGO, the difference between the three MIS settings is slight, presumably because the DJANGO dataset is more complex and MIS setting may not be the major factor affecting performance.

Unpaired data in bi-JAEs. In experiments with bi-JAEs on ATIS and DJANGO which both consist of all labeled data, we first randomly create the parallel data $\mathbb{L}$, then use the remaining x and z to create $\mathbb{U}_x$ and $\mathbb{U}_z$ respectively. The unpaired data $\mathbb{U}_x$ and $\mathbb{U}_z$ obtained in this way are in fact paired, which seldom occur in realistic tasks. The following experiment is conducted to examine the effect of this artificial pairing between $\mathbb{U}_x$ and $\mathbb{U}_z$ for bi-JAEs. Apart from $\mathbb{L}$ ($|\mathbb{L}| = 1{,}000$ for ATIS and $|\mathbb{L}| = 5{,}000$ for DJANGO), we split the remaining data into two parts of equal sizes, denoted as $\mathbb{U}_1$ and $\mathbb{U}_2$ respectively. Two settings to create $\mathbb{U}_x$ and $\mathbb{U}_z$ are tested. The first uses x from $\mathbb{U}_1$ and z also from $\mathbb{U}_1$ (denoted as $\mathbb{U}_1$ + $\mathbb{U}_1$) to create $\mathbb{U}_x$ and $\mathbb{U}_z$ respectively, which are in fact paired. The second uses x from $\mathbb{U}_1$ and z from $\mathbb{U}_2$ (denoted as $\mathbb{U}_1$ + $\mathbb{U}_2$), thus without artificial pairing between $\mathbb{U}_x$ and $\mathbb{U}_z$. It can be seen from the results in Table V that the performance difference between the two settings is weak. The artificial pairing does not inadvertently magnify bi-JAEs. This also suggests that $\mathbb{U}_x$ and $\mathbb{U}_z$ can be created completely separately in real tasks.

IV. CONCLUSION

The contribution of this letter is that we present a solid advance from unsupervised learning of ordinary fixed-dimensional JAEs [15] to S^4 learning of seq2seq JAEs, and further we propose bi-directional JAEs to leverage not only unpaired x-data but also unpaired z-data. The effects of sample caching, different MIS settings and artificial pairing between $\mathbb{U}_x$ and $\mathbb{U}_z$ for the challenging S^4 learning are examined. Experimental results on two semantic parsing benchmarking datasets demonstrate the superiority of JAEs and bi-JAEs over their VAE counterparts.

REFERENCES

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[2] J. Andreas, A. Vlachos, and S. Clark, "Semantic parsing as machine translation," in *Proc. Assoc. Comput. Linguistics*, 2013, pp. 47–52.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Representations*, 2015.

[4] L. Zettlemoyer and M. Collins, "Online learning of relaxed CCG grammars for parsing to logical form," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2007, pp. 678–687.

[5] Y. Oda *et al.*, "Learning to generate pseudo-code from source code using statistical machine translation (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2015, pp. 574–584.

[6] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.

[7] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1979–1993, Aug. 2019.

[8] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3581–3589.

[9] Y. Miao and P. Blunsom, "Language as a latent variable: Discrete generative models for sentence compression," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 319–328.

[10] T. Kočiskỳ *et al.*, "Semantic parsing with semi-supervised sequential autoencoders," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1078–1087.

[11] P. Yin, C. Zhou, J. He, and G. Neubig, "Structvae: Tree-structured latent variable models for semi-supervised semantic parsing," in *Proc. Assoc. Comput. Linguistics*, 2018, pp. 754–765.

[12] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. Int. Conf. Learn. Representations*, 2017.

[13] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Int. Conf. Learn. Representations*, 2017.

[14] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick, "Improved variational autoencoders for text modeling using dilated convolutions," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3881–3890.

[15] H. Xu and Z. Ou, "Joint stochastic approximation learning of Helmholtz machines," in *Proc. Int. Conf. Learn. Representations Workshop Track*, 2016.

[16] D. He *et al.*, "Dual learning for machine translation," in *Proc. Conf. Neural Inf. Process. Syst.*, 2016, pp. 820–828.

[17] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 1951.

[18] B. Wang, Z. Ou, and Z. Tan, "Learning trans-dimensional random fields with applications to language modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 876–890, Apr. 2018.

[19] H. Larochelle, M. I. Mandel, R. Pascanu, and Y. Bengio, "Learning algorithms for the classification restricted Boltzmann machine," *J. Mach. Learn. Res.*, vol. 13, pp. 643–669, Mar. 2012.